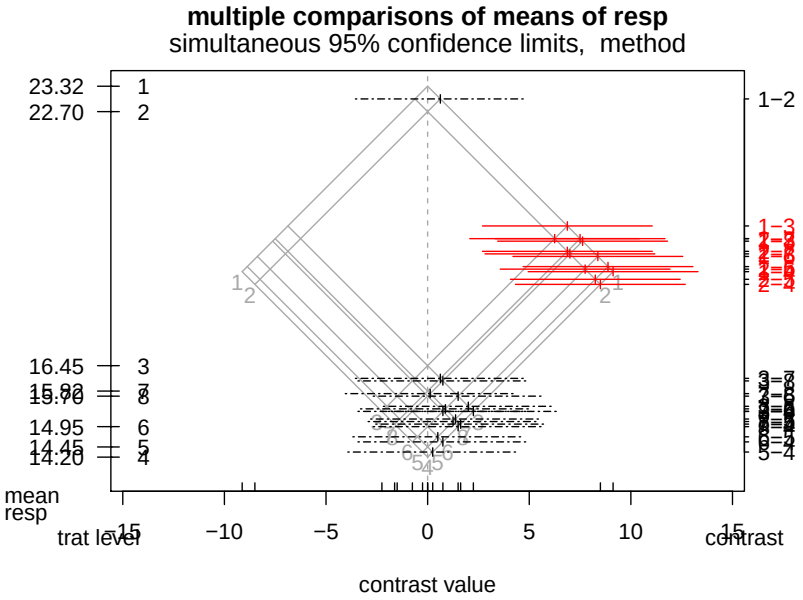

Embrapa Arroz e Feijão

Estatística Experimental Aplicada

de 07 à 11 de novembro de 2011

Walmes Marques Zeviani
Laboratório de Estatística e Geoinformação (LEG)
Departamento de Estatística
Universidade Federal do Paraná

Documento concluído em 6 de novembro de 2011 às 15:04:09 – Laboratório de Estatística e Geoinformação – Universidade Federal do Paraná. Acesse a página do Curso em `http://www.leg.ufpr.br/cursorcnpaf`.

Sumário

1 Manipulação de objetos e funções

1.1 Instalação do R

```
.
#-------------------------------------------------------------------------------------------
# página do R, onde estão os pacotes, tutoriais e arquivos de instalação
browseURL(URLencode("http://www.r-project.org/"))
                                                                                           #
#-------------------------------------------------------------------------------------------
# link direto para a página de download da versão para Windows
browseURL(URLencode("http://cran.stat.ucla.edu/bin/windows/base/"))
                                                                                           #
#-------------------------------------------------------------------------------------------
# documento com instruções de intalação e primeiros passos
browseURL(URLencode("http://cran.r-project.org/doc/contrib/Itano-installation.pdf"))
                                                                                           #
#-------------------------------------------------------------------------------------------
# página do R Studio, a interface do momento
browseURL(URLencode("http://www.rstudio.org/"))
                                                                                           #
#-------------------------------------------------------------------------------------------
# página da [R-br], a lista Brasileira oficial de usuários do programa R
browseURL(URLencode("http://www.leg.ufpr.br/rbr"))
                                                                                           #
#-------------------------------------------------------------------------------------------
# curiosidades sobre o R, manchete no New York Times e lista de abreviações das funções
browseURL(URLencode("http://www.nytimes.com/2009/01/07/technology/business-computing/07program.html"))
browseURL(URLencode("http://jeromyanglim.blogspot.com/2010/05/abbreviations-of-r-commands-explained.html"))
                                                                                           #
#-------------------------------------------------------------------------------------------
# blogs/sites para consulta sobre R
browseURL(URLencode("http://www.r-bloggers.com/"))
browseURL(URLencode("http://r-project.markmail.org/search/?q="))
browseURL(URLencode("http://www.statmethods.net/index.html"))
browseURL(URLencode("http://zoonek2.free.fr/UNIX/48_R/all.html"))
browseURL(URLencode("http://addictedtor.free.fr/graphiques/"))
browseURL(URLencode("http://www.leg.ufpr.br/"))
browseURL(URLencode("http://www.leg.ufpr.br/cursorufgd"))
                                                                                           #
#-------------------------------------------------------------------------------------------
.
```

1.2 Primeiros passos no R: como pedir ajuda

Antes de qualquer coisa, iniciaremos o aprendizado de R com as formas de pedir ajuda e fazer acesso as fontes disponíveis de informação sobre pacotes e funções.

```
.
#-------------------------------------------------------------------------------------------
# quando você só sabe algo sobre
apropos("tukey") # procura dentre as funções locais aquela que tem tukey no nome
apropos("help")  # essa função aceita expressões regulares
                                                                                           #
#-------------------------------------------------------------------------------------------
# fazendo a busca do termo
help(TukeyHSD)                   # documentação é apresentada no editor
help(TukeyHSD, help_type="html") # documentação é apresentada no navegador de internet
?TukeyHSD                        # é o mesmo que help(TukeyHSD)
                                                                                           #
#-------------------------------------------------------------------------------------------
# buscando em pacotes o termo
help.search("Tukey") # faz uma busca ampla do termo, inclusive em pacotes não carregados
??Tukey              # o mesmo que help.search("Tukey")
                                                                                           #
#-------------------------------------------------------------------------------------------
# fazendo a busca na web
RSiteSearch("Tukey") # faz busca na documentação online no R
                                                                                           #
#-------------------------------------------------------------------------------------------
# procurar implementações de métodos/funções/pacotes no R
browseURL(URLencode("http://www.inside-r.org/")) # para buscar métodos implementados no R
                                                                                           #
#-------------------------------------------------------------------------------------------
.
```

1.3 Como o R funciona: criação, atribuição, acesso e modificação de objetos

```
.
#---------------------------------------------------------------------------------------
# criação de vetores, sequências lógicas e números aleatórios
c(2,4,7,3,8,9)          # um vetor
1:7                     # uma sequencia regular de passo 1
seq(0, 20, by=2.4)      # uma sequencia regular de passo 2.4
seq(0, 20, length=4)    # uma sequencia regular de 4 elementos
help(seq, help_type="html")
rep(1:3, times=3)       # repete o vetor todo 3 vezes
rep(1:3, each=3)        # repete cada elemento 3 vezes
rnorm(5, 3, 2)          # números aletórios normais média=3 desvio=2
rnorm(5, sd=2, mean=3)  # o mesmo
rnorm(5, mean=3, sd=2)  # o mesmo
runif(5)                # número aletórios uniformes min=0, max=1
                                                                                      #
#---------------------------------------------------------------------------------------
# matrizes
matrix(c(1,5,38,400), 2, 2)      # matriz 2x2
matrix(1:6, 2, 3)                # matriz 2x3
matrix(rnorm(9), 3, 3)           # matriz 3x3
matrix(c("a","c","b","j"), 2, 2) # matriz 2x2
                                                                                      #
#---------------------------------------------------------------------------------------
# estrutura de dados (planilha)
data.frame(A=1:4, B=runif(4), C=letters[1:4])
data.frame(trat=c(1:2,1:2), bloc=rep(1:2, e=2))
expand.grid(cult=c("A","B"), bloc=c("I","II","III"), dose=1:3)
                                                                                      #
#---------------------------------------------------------------------------------------
# listas
list(A=rnorm(4),
     B=matrix(1:4,2,2),
     C=data.frame(a=1:4, b=runif(4), c=letters[1:4]),
     D="O R é livre")
                                                                                      #
#---------------------------------------------------------------------------------------
# atribuição, acesso e modificação de vetores
x <- seq(12, 18, 2); x
x[1]
x[2:3]
x[-4]
x[3:4] <- c(20,22); x
x <- c(x, 40, 89, 132)
x
                                                                                      #
#---------------------------------------------------------------------------------------
# criando vetores com scan()
x <- scan()
                                                                                      #
#---------------------------------------------------------------------------------------
# atribuição, acesso e modificação de matrizes
x <- matrix(rnorm(9), 3, 3); x
x[1,]
x[,1]
x[2,2]
x[-3,-3]
x[3,1] <- 19; x
x[3,1] <- "19"; x
                                                                                      #
#---------------------------------------------------------------------------------------
# atribuição, acesso e modificação de tabelas de dados
x <- data.frame(A=1:4, B=runif(4), C=letters[1:4]); x
x[,1]
x[,"A"]
x[1,2]
x[-3,-3]
x[1,"A"] <- "200"
x$A
                                                                                      #
#---------------------------------------------------------------------------------------
# digitando dados com edit()
x <- edit(data.frame())
                                                                                      #
#---------------------------------------------------------------------------------------
# atribuição, acesso e modificação de "planilhas"
x <- list(A=rnorm(4),
          B=matrix(1:4,2,2),
```

```
          C=data.frame(a=1:4, b=runif(4), c=letters[1:4]))
x
x[[1]]
x[[3]][,1]
x$B
x[["C"]]
x[["C"]][1,1] <- 0
                                                                                                #
#-------------------------------------------------------------------------------------------
.
```

1.4 Informações sobre objetos (atributos)

```
.
#-------------------------------------------------------------------------------------------
# como obter informações sobre um objeto? Diga-me o que tu és que digo o que farei contigo

#-------------------------------------------------------------------------------------------
# para vetores
v <- c(a=1, b=2, c=3)
v
length(v)  # dimensão/comprimento/número e elementos
class(v)   # classe do objeto
typeof(v)  # tipo de elemento que o objeto armazena
names(v)   # nome dos elementos
                                                                                                #
#-------------------------------------------------------------------------------------------
# para matrizes
m <- matrix(1:3,2,2)
m
dim(m)      # dimensões
nrow(m)     # número de linhas
ncol(m)     # número de colunas
class(m)    # classe do objeto
colnames(m) # nome das colunas
rownames(m) # nome das linhas
colnames(m) <- c("prod","peso") # modifica o nome das colunas
rownames(m) <- c("M","F")       # modifica o nome das linhas
colnames(m)
m
                                                                                                #
#-------------------------------------------------------------------------------------------
# para data.frames
d <- expand.grid(A=1:2, B=c("A","B"))
dim(d)   # dimensões
nrow(d)  # número de linhas
ncol(d)  # número de colunas
names(d) # nome das colunas
names(d) <- c("trat", "bloc") # modifica o nome das colunas
d
                                                                                                #
#-------------------------------------------------------------------------------------------
# para listas
l <- list(A=rnorm(4), B=matrix(1:4,2,2))
length(l) # número de elementos/número de slots/número de "abas"
class(l)  # classe do objeto
names(l)  # nome dos elementos/nome dos slots/nome das "abas"
l
                                                                                                #
#-------------------------------------------------------------------------------------------
# como saber praticamente tudo sobre um objeto?
str(v)
str(m)
str(d)
str(l)
ls()   # lista os objetos da memória
                                                                                                #
#-------------------------------------------------------------------------------------------
.
```

1.5 Fazendo operações matemáticas e a lógica da reciclagem

```
.
#-----------------------------------------------------------------------------------------
# as operações fundamentais e mais
1+100   # soma
3-5     # subtração
2*8     # produto
3/4     # divisão
2^3     # potenciação, também vale 2**3
sqrt(4) # raíz quadrada
exp(3)  # número neperiano 2.71... veja 2.71^3
log(10)           # logartimo base e
log10(1000)       # logartimo base 10
log(30, base=2.2) # logartimo base qualquer
                                                                                        #
#-----------------------------------------------------------------------------------------
# as operações em vetores e a lei da recliclagem
x <- 1:3
x-1
x+c(5:7)
x/3
x/c(1:2)
x^2
log(x)
                                                                                        #
#-----------------------------------------------------------------------------------------
# as operações com matrizes
x <- matrix(1:4, 2, 2)
y <- matrix(4:1, 2, 2)
z <- matrix(1:6, 3, 2)
x*10  # produto por constante
x-4   # subtração por uma contante
x+y   # soma de matrizes (elementwise)
x*y   # produto de elementos (elementwise)
x%*%y # produto de matrizes
x+z
z%*%x
det(x)   # determinante
diag(x)  # elementos da diagonal
solve(x) # inversa
t(z)     # transposta
                                                                                        #
#-----------------------------------------------------------------------------------------
# exemplo de operações com matrizes: estimação de parâmetros em modelo linear
x <- 0:12                 # valores da variável indendente
y <- x+rnorm(x,0,1)       # valores da variável aleatória dependente
plot(x, y)                # diagrama de dispersão de y em função de x
X <- cbind(b0=1, b1=x)    # matriz do modelo de regressão linear E(Y|x) = b0+b1*x
beta <- solve(t(X)%*%X)%*%t(X)%*%y # ou solve(crossprod(X), crossprod(X, y))
beta  # parâmetros estimados pelo método dos mínimos quadrados
                                                                                        #
#-----------------------------------------------------------------------------------------
# operações trigonométricas, (antigamente) útil para transformação de dados
x <- seq(0, 2, 0.5) # sequência regular de 0 à 2 com incremento 0.5
pi          # constante da trigonometria 3.1415
sin(x*pi)   # função seno, calcula o seno de um ângulo (em radianos)
cos(x*pi)   # função cosseno, calcula o coseno de um ângulo (em radianos)
tan(x*pi)   # função tangente, calcula a tangente de um ângulo (em radianos)
asin(1)/pi  # arcoseno, é o inverso do seno, dado um ângulo retorna o radiano
acos(1)/pi  # arcocosseno, é o inverso do cosseno, dado um ângulo retorna o radiano
atan(1)/pi  # arcotangente, é o inverso da tangente, dado um ângulo retorna o radiano
                                                                                        #
#-----------------------------------------------------------------------------------------
.
```

1.6 Operações estatísticas

```
.
#-----------------------------------------------------------------------------------------
# em vetores
x <- rnorm(1000, 80, 3) # valores realizados de uma distribuição normal
mean(x)        # média (average)
sum(x)         # soma
var(x)         # variância amostral (denominador n-1)
sd(x)          # desvio padrão amostral (denominador n-1)
median(x)      # mediana
max(x)         # máximo
```

```
min(x)         # mínimo
range(x)       # extremos
diff(range(x)) # amplitude
summary(x)     # resumo: extremos, 1º, 2º e 3º quartis e média
IQR(x)         # distância interquatílica
quantile(x)    # quantis/percentis
plot(x)        # diagrama de dispersão, valores ~ ordem
hist(x)        # histograma/distribuição de frequências
# ver pacote fBasics para mais funções de análise descritiva
                                                                                        #
#---------------------------------------------------------------------------------------
# operações em matrizes para obter quantidades marginais em linhas e colunas
x <- matrix(rnorm(20), 4, 5)
colSums(x)  # soma por colunas
rowMeans(x) # média por linhas
mean(x)     # média entre todos os elementos
var(x)      # matriz de covariância amostral
cor(x)      # matriz de correlação amostral
sd(x)       # vetor de desvios-padrões amostrais por coluna
apply(x, MARGIN=1, FUN=var)    # variância marginal nas linhas
apply(x, 2, median)            # mediana marginal nas colunas
                                                                                        #
#---------------------------------------------------------------------------------------
# operações com data.frames para obter quantidades separadas por categorias
x <- expand.grid(produto=c("controle","tratado"), # coluna com níveis do fator produto
                 nitro=c("presente","ausente"),   # coluna com níveis do fator nitrogênio
                 rep=1:10)                        # coluna com índice da repetição
x # data.frame gerada pelo produto cartesiano dos níveis (cruzamento completo entre níveis)
x$resp <- rnorm(nrow(x), 0, 1) # resposta obervada (hipotético)
x
tapply(x$resp, x$produto, mean) # calcula a média da resp separado por níveis de produto
tapply(resp, produto, mean)     # retorna uma mensagem de erro
ls()                            # lista os objetos criados, esclarece a mensagem de erro
with(x, tapply(resp, produto, mean))                # o mesmo de um jeito mais econômico
with(x, tapply(resp, list(produto, nitro), sum))    # fazendo para a combinação dos níveis
with(x, aggregate(resp, list(produto, nitro), mean)) # o mesmo numa saída diferente
                                                                                        #
#---------------------------------------------------------------------------------------
.
```

1.7 Construindo funções simples

```
.
#---------------------------------------------------------------------------------------
# criação e uso de funções simples
f0 <- function(x, y){ # x e y são os argumentos da função
  (x+y)^2             # código entre { } é o corpo, são os procedimentos executados
}
class(f0)
args(f0)     # argumentos que devem ser fonecidos para a função
body(f0)     # o corpo da função
f0(3, 2)     # uso da função com escalares
f0(1:3, 0:2) # uso da função com vetores
f0(1:3, 2)   # uso da função com vetor e escalar
                                                                                        #
#---------------------------------------------------------------------------------------
# exemplo: função para obtenção das raízes de uma função de 2 grau
baskara <- function(a,b,c){ # argumentos da função
  ## a, b, c: escalares constantes do polinômio de 2º grau a+bx+cx²
  x1 <- (-b-sqrt(b^2-4*a*c))/(2*a) # 1ª raiz/solução
  x2 <- (-b+sqrt(b^2-4*a*c))/(2*a) # 2ª raiz/solução
  return(c(x1, x2))                # vetor retornado como resultado pela função
}
                                                                                        #
#---------------------------------------------------------------------------------------
# aplicando a função recém criada
baskara(-3,2,1)
baskara(3,2,1)
                                                                                        #
#---------------------------------------------------------------------------------------
# gráfico das funções
curve(3*x^2+2*x+1, -1, 2)             # faz gráfico de funções paramétricas de uma variável
curve(-3*x^2+2*x+1, -1, 2)
abline(h=0, v=baskara(-3,2,1), lty=2) # adiciona linhas verticais ao gráfico
                                                                                        #
#---------------------------------------------------------------------------------------
```

```
# exemplo: função para obtenção da nota necessária para ser aprovado na 3 prova
nota3 <- function(nota1, nota2){ # argumentos da função
  ## nota1 e nota2: notas do aluno nas 2 primeiras avaliações
  n3 <- 21-nota1-nota2 # nota mínima necessária na 3º prova para ser aprovado
  if(n3<=10){
    cat("nota mínima:", n3, "(pode ser aprovado sem exame)")
  } else {
    cat("nota mínima:", n3, "(terá que fazer o exame)")
  } # o resultado da função é a útima conta realizada
}
nota3(3,5)
nota3(8,9.5)
                                                                                        #
#---------------------------------------------------------------------------------------
.
```

2 Importação de dados e análise exploratória

2.1 Importando dados

```
.
#---------------------------------------------------------------------------------------
# como importar/ler dados?
apropos("read")                    # retorna as funções que possuem o termo "read" no nome
help(read.table, help_type="html") # acessa a documentação da função pelo navegador
                                                                                        #
#---------------------------------------------------------------------------------------
# onde os dados devem estar?
getwd()                                          # o seu diretório de trabalho atual
setwd("/home/walmes/Documentos/Curso R Embrapa")  # alterar o seu diretório de trabalho
                                                                                        #
#---------------------------------------------------------------------------------------
# importando dados, descomente a opção que for conveniente
#soja <- read.table("http://www.leg.ufpr.br/~walmes/cursoR/cnpaf/soja.txt", header=TRUE, sep="\t", dec=",")
#soja <- read.table("soja.txt", header=TRUE, sep="\t", dec=",")
#soja <- read.table("../dados/soja.txt", header=TRUE, sep="\t", dec=",")
class(soja)  # classe do objeto
names(soja)  # nomes das colunas
dim(soja)    # dimensões
str(soja)    # estrutura
head(soja)   # cabeçalho
soja         # todos os registros (ter cuidado quando a base for grande)
                                                                                        #
#---------------------------------------------------------------------------------------
# exploração númerica, médias por nível de potássio, água e potássio:água
with(soja, tapply(rengrao, list(potassio), mean))
with(soja, tapply(rengrao, list(agua), mean))
with(soja, tapply(rengrao, list(potassio, agua), mean))
                                                                                        #
#---------------------------------------------------------------------------------------
# selecionando subconjuntos dos dados de acordo com os níveis das categorias
subset(soja, potassio==0)
subset(soja, bloco=="I")
subset(soja, potassio==0 & bloco=="I")
                                                                                        #
#---------------------------------------------------------------------------------------
# selecionando subconjunto dos dados por valores das respostas
subset(soja, rengrao<15)
subset(soja, rengrao<15 & pesograo<11)
                                                                                        #
#---------------------------------------------------------------------------------------
# um pouco sobre perguntas lógicas
1==1  # é igual?
2==1
1!=3  # é diferente?
3!=3
1<2   # é menor?
1<1
1<=1  # é menor e igual?
1<=1 & 2>1 # é menor e igual E maior?
1<=1 & 1>1
1<3 | 2<3  # é menor OU menor?
1<3 | 4<3
5<3 | 4<3
"joão"=="João" # R é case sensitive
```

```
"joão"=="joao"
                                                                                                #
#-------------------------------------------------------------------------------------------
.
```

2.2 Explorações gráficas (1)

```
.
#-------------------------------------------------------------------------------------------
# matriz de diagramas de dispersão, útil para uma inspeção macro sobre relações
# também ver a função car::scatterplotMatrix(soja)
pairs(soja)
                                                                                                #
#-------------------------------------------------------------------------------------------
# gráficos simples de dispersão (rótulos, cores, simbolos, tamanhos)
plot(rengrao~potassio, data=subset(soja, agua==50))
plot(rengrao~potassio, data=subset(soja, agua==50),
     xlab="Dose de potássio", ylab="Rendimento de grãos",
     col=2, pch=19, cex=1.2)
                                                                                                #
#-------------------------------------------------------------------------------------------
# boxplot (subconjuntos e cores)
boxplot(rengrao~potassio, data=subset(soja, agua==50))
boxplot(rengrao~potassio, data=soja, col="yellow")
                                                                                                #
#-------------------------------------------------------------------------------------------
# todos níveis de água ao mesmo tempo (título)
par(mfrow=c(1,3)) # divide a janela gráfica
plot(rengrao~potassio, data=subset(soja, agua==37.5), main="37.5% de água")
plot(rengrao~potassio, data=subset(soja, agua==50), main="50.0% de água")
plot(rengrao~potassio, data=subset(soja, agua==62.5), main="62.5% de água")
                                                                                                #
#-------------------------------------------------------------------------------------------
# gráficos de barras (adição de texto)
par(mfrow=c(1,1)) # restaura a janela gráfica
pot.m <- with(soja, tapply(rengrao, potassio, mean))
pot.m
bp <- barplot(pot.m) # alterar para ylim=c(0,32)
text(bp, pot.m, label=round(pot.m, 3), pos=3) # pos=3
title("Médias dos tratamentos")
box()
                                                                                                #
#-------------------------------------------------------------------------------------------
# melhorando o aspecto
bp <- barplot(pot.m, ylim=c(0,33), col="seagreen",
              xlab="Dose de potássio", ylab="Rendimento médio de grãos")
text(bp, pot.m, label=round(pot.m, 3), pos=3, font=3)
title("Médias dos tratamentos")
box()
                                                                                                #
#-------------------------------------------------------------------------------------------
.
```

2.3 Explorações gráficas (2)

```
.
#-------------------------------------------------------------------------------------------
# lendo novos dados
#agr <- read.table("http://www.leg.ufpr.br/~walmes/cursoR/cnpaf/agreg.txt", header=TRUE, sep="\t")
#agr <- read.table("agreg.txt", header=TRUE, sep="\t")
#agr <- read.table("../dados/agreg.txt", header=TRUE, sep="\t")
names(agr)
str(agr)
                                                                                                #
#-------------------------------------------------------------------------------------------
# qual a relação marginal entre as variáveis?
pairs(agr)
# impressões visuais:
# porque perimetro só assume 4 valores se é uma variável contínua?
# perimetro aumenta aumenta com maior ou menor eixo (lógico).
# aspecto e roundness possuem associação fraca.
# maior eixo e menor eixo apresentam a restrição de que maior eixo>menor eixo (lógico)
```

```
# todas essas conclusões podem mudar se separarmos os dados para os níveis de profundidade
                                                                                                #
#-------------------------------------------------------------------------------------------
# qual a distribuição de frequência? (cores, número de classes, tipo de frequência)
hist(agr$roundness)             # número de classe usa regra de Sturges
hist(agr$roundness, freq=FALSE) # usa densidade=freq_rel/ampl_clase, vale prob=TRUE
plot(density(agr$roundness))    # estimação suave da dist de freq (moderno)
rug(agr$roun)                   # adiciona os traços na margem inferior
                                                                                                #
#-------------------------------------------------------------------------------------------
# os dados têm distribição normal? como checar?
par(mfrow=c(1,2))
qqnorm(agr$roundness); qqline(agr$roundness)
qqnorm(agr$aspecto); qqline(agr$aspecto)
                                                                                                #
#-------------------------------------------------------------------------------------------
# gráfico qq por categoria
with(subset(agr, profundidade==5), { qqnorm(roundness); qqline(roundness) })
with(subset(agr, profundidade==20), { qqnorm(roundness); qqline(roundness) })
                                                                                                #
#-------------------------------------------------------------------------------------------
# gráficos para visualizar a aderência de uma distribuição (dois comandos por linha)
qqnorm(scale(agr$roundness), asp=1); qqline(scale(agr$roundness))
hist(scale(agr$roundness), freq=FALSE)
curve(dnorm(x), add=TRUE, col=2); lines(density(scale(agr$roundness)), col=3)
                                                                                                #
#-------------------------------------------------------------------------------------------
# o que fazer? tranformar? qual tranformação? raiz? log?
require(MASS) # faz a requisição do pacote MASS, suas funções estarão disponíveis
                                                                                                #
#-------------------------------------------------------------------------------------------
# faz a estimação do parâmetro lambda para aplicar a transformação boxcox
agr5 <- subset(agr, profundidade==5) # atribui dados à um objeto
layout(1)
boxcox(agr5$roundness~1, lambda=seq(-1,6,l=100))
qqnorm(agr5$roundness^4); qqline(agr5$roundness^4)
                                                                                                #
#-------------------------------------------------------------------------------------------
# aplica o teste de normalidade de shapiro wilk
shapiro.test(agr5$roundness)
shapiro.test(agr5$roundness^4)
shapiro.test(sqrt(agr5$roundness))
shapiro.test(log(agr5$roundness))
                                                                                                #
#-------------------------------------------------------------------------------------------
# o que fazer em casos como esse? qual a causa do afastamento da normalidade?
boxcox(agr5$aspecto~1, lambda=seq(-1,6,l=100))
qqnorm(agr5$aspecto^3); qqline(agr5$aspecto^3)
                                                                                                #
#-------------------------------------------------------------------------------------------
.
```

2.4 Explorações gráficas (3)

```
.
#-------------------------------------------------------------------------------------------
# biblioteca para gráficos
require(lattice)
                                                                                                #
#-------------------------------------------------------------------------------------------
# de volta aos dados de soja
xyplot(rengrao~potassio, groups=agua, data=soja)
xyplot(rengrao~potassio, groups=agua, data=soja, type=c("p","a"))
xyplot(rengrao~potassio|agua, data=soja, type=c("p","a"))
xyplot(rengrao~potassio|agua, data=soja, type=c("p","smooth"))
                                                                                                #
#-------------------------------------------------------------------------------------------
# de volta aos dados de agragados
qqmath(~roundness, groups=profundidade, data=agr)
qqmath(~roundness|profundidade, data=agr)
qqmath(~roundness+aspecto|profundidade, data=agr)
                                                                                                #
#-------------------------------------------------------------------------------------------
# histograma
histogram(~roundness|profundidade, data=agr)
densityplot(~roundness+aspecto|profundidade, data=agr)
```

```
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# matriz de dispersão
str(agr)
splom(agr[,-1], group=agr$profundidade)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# nota importante sobre normalidade! os dados abaixo têm distribuição normal?
m <- gl(15,8)                    # 15 níveis com 8 repetições de cada
x <- rnorm(m, as.numeric(m), 0.1) # E(Y|x) = x
xp <- qqnorm(x); qqline(x)
rug(xp$x)
rug(xp$y, side=2)
m0 <- lm(x~m)
xp <- qqnorm(residuals(m0)); qqline(residuals(m0))
rug(xp$x)
rug(xp$y, side=2)
shapiro.test(x)              # ver sobre esse teste o grau de liberdade
shapiro.test(residuals(m0))
                                                                                                    #
#-----------------------------------------------------------------------------------------------
.
```

2.5 Gráficos interativos sobre normalidade e correlação

```
.
#-----------------------------------------------------------------------------------------------
# carrega o pacote rpanel com funções para manipulação interativa de gráficos
require(rpanel)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# função para ilustrar o impacto do efeito dos tratamentos no teste de normalidade
panel.norm <- function(panel){
  mm <- seq(0, by=panel$h1, length.out=panel$nlev)
  m <- rep(mm, panel$nrep)
  set.seed(panel$seed)
  x <- rnorm(length(m), m, 1)
  par(mfrow=c(1,2))
  qq <- qqnorm(x, main="com os dados", col=as.numeric(as.factor(mm)))
  qqline(x); rug(qq$x); rug(qq$y, side=2)
  sh1 <- shapiro.test(x)$p.value
  mtext(sh1)
  m0 <- aov(x~factor(m))
  r <- residuals(m0)
  pp <- qqnorm(r, main="com os resíduos", col=as.numeric(as.factor(mm)))
  qqline(r); rug(pp$x); rug(pp$y, side=2)
  sh2 <- shapiro.test(r)$p.value
  mtext(sh2)
}
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# faz uso do painel criado
panel <- rp.control(nlev=10, nrep=10, seed=1111) # variáveis que são fixas
rp.slider(panel, h1, 0, 10, initval=0, showvalue=TRUE, action=panel.norm) # controladas
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# função para ilustrar o impacto das médias dos tratamentos no teste de correlação entre
# duas respostas avaliadas
panel.cor <- function(panel){
  V <- matrix(c(1,panel$cor,panel$cor,1),2,2)
  v <- chol(V)
  mm <- seq(0, by=panel$h1, length.out=panel$nlev)
  m <- rep(mm, panel$nrep)
  set.seed(panel$seed)
  x <- matrix(rnorm(2*length(m)), ncol=2)
  x <- t(t(v)%*%t(x))
  x[,1] <- x[,1]+m; x[,2] <- x[,2]+m
  par(mfrow=c(1,2))
  plot(x, col=as.numeric(as.factor(m)))
  points(mm, mm, pch=19, cex=1.2, col=as.numeric(as.factor(mm)))
  legend("topleft", legend=cor.test(x[,1], x[,2])$est, bty="n")
  m0 <- aov(x~factor(m))
  r <- residuals(m0)
  plot(r, col=as.numeric(as.factor(m)))
  legend("topleft", legend=cor.test(r[,1], r[,2])$est, bty="n")
}
```

```
panel <- rp.control(nlev=10, nrep=10, seed=1234)
#-------------------------------------------------------------------------------------
# faz uso do painel criado
rp.slider(panel, h1, 0, 5, initval=0, showvalue=TRUE, action=panel.cor)     # fixas
rp.slider(panel, cor, -1, 1, initval=0.2, showvalue=TRUE, action=panel.cor) # controladas
#-------------------------------------------------------------------------------------
.
```

3 Estatística descritiva

3.1 Distribuição de frequências

```
#-------------------------------------------------------------------------------------
# importação de dados de competição de híbridos de milho
#hib <- read.table("http://www.leg.ufpr.br/~walmes/cursoR/cnpaf/hibridos.txt", header=TRUE, sep="\t")
#hib <- read.table("hibridos.txt", header=TRUE, sep="\t")
#hib <- read.table("../dados/hibridos.txt", header=TRUE, sep="\t")
str(hib)
                                                                                    #
#-------------------------------------------------------------------------------------
# para verificar os níveis que as variáveis categóricas assumiram
levels(hib$grao)
nlevels(hib$grao)
levels(hib$resistencia)
nlevels(hib$resistencia)
lapply(list(grao=hib$grao, resistencia=hib$resistencia), levels)
lapply(as.list(hib[,c("grao","resistencia")]), levels)
                                                                                    #
#-------------------------------------------------------------------------------------
# para obter a distribuição de frequência dos níveis
tapply(hib$grao, hib$grao, length) # distribuição absoluta de frequência
table(hib$grao)                    # o mesmo com uma função apropriada
prop.table(table(hib$grao))        # distribuição relativa de frequência
                                                                                    #
#-------------------------------------------------------------------------------------
# gráficos para distribuição de frequência de variáveis categóricas
fr <- prop.table(table(hib$grao))
str(fr)     # note que é um vetor nomeado
barplot(fr) # gráfico de barras, opções: col=, dens=, (x|y)lim=, (x|y)lab=, main=...
pie(fr)     # gráfico de setores, opções: col=, dens=, main=...
                                                                                    #
#-------------------------------------------------------------------------------------
# gráficos usando diversas opções
barplot(fr) # faz o gráfico básico, necessário para extração dos limites dessa janela
alt <- 3*strheight("palavra") # altura de uma palavra com relação ao último gráfico feito
lim <- par()$usr              # os limites dos eixos do último gráfico feito
bp <- barplot(fr, ylim=lim[3:4]+c(0,alt),
              col=c("springgreen","limegreen","seagreen"),
              xlab="Tipo de grão", ylab="Frequência relativa")
text(bp, fr, labels=fr, pos=3)
title("Distribuição de frequências")
mtext("Valores obtidos em uma amostra de 32 registros", side=3, lin=0.5, font=3)
box()
# apesar de bonito, um gráfico com 3 barras apresenta baixa relação informação/espaço
                                                                                    #
#-------------------------------------------------------------------------------------
# gráfico de setores
pie(fr, labels=paste(names(fr), " (",100*fr,"%)", sep=""),
    col=c("springgreen","limegreen","seagreen"),
    main="Distribuição de frequências")
mtext("Valores obtidos em uma amostra de 32 registros", side=3, lin=0.5, font=3)
                                                                                    #
#-------------------------------------------------------------------------------------
# distribuição de frequência para variáveis discretas (ciclo é discreto? reflita)
sort(unique(hib$ciclo)) # valores distintos observados
table(hib$ciclo)        # distribuição de frequência absoluta
typeof(hib$ciclo)       # tipo dos valores
                                                                                    #
#-------------------------------------------------------------------------------------
# gráficos de distribuição de frequência
fr <- table(hib$ciclo)
barplot(fr) # ops, não reconheceu que 67 e 75 não estão presentes
plot(fr)    # agora sim!
pie(fr)     # completamente sem sentido usar esse gráfico
```

```
# pode-se agrupar os valores em classes
                                                                                                  #
#-----------------------------------------------------------------------------------------------
# distribuição de frequência para variáveis contínuas
summary(hib$rendimento)
range(hib$rendimento)
sort(unique(hib$rendimento)) # cada valor ocorre uma única vez?
table(hib$rendimento)        # sim!
rclass <- cut(hib$rendimento, c(3900,4500,5000,5500,6000,6400), dig=4)
str(rclass)   # as classes se tornan níveis de uma variável categórica
table(rclass) # distribuição de frequâncias absolutas
                                                                                                  #
#-----------------------------------------------------------------------------------------------
# gráficos de distribuição de frequência
fr <- table(rclass)
barplot(fr)          # mas as classes tem limites encontrados
barplot(fr, space=0) # mas as classes tem amplitudes diferentes
# barplot é para níveis categóricos, para contínua deve-se usar o histograma
                                                                                                  #
#-----------------------------------------------------------------------------------------------
# histograma
hist(hib$rendimento)                                          # divisão default em classes
hist(hib$rendimento, breaks=c(3900,4500,5000,5500,6000,6400)) # definida pelo usuário
hist(hib$rendimento, breaks=c(3900,4500,5000,5500,6000,6400), prob=TRUE) # densidade
rug(hib$rendimento)
                                                                                                  #
#-----------------------------------------------------------------------------------------------
# histograma: amplitude de classe
require(rpanel)
hist.panel <- function(panel){
  hist(panel$x, breaks=seq(panel$from, panel$to, length=panel$length))
  panel
}
panel <- rp.control(x=hib$rendimento, from=3900, to=6400)
rp.slider(panel, length, 3, 12, resolution=1, initval=3, action=hist.panel)
# histograma tem sido substituido pelo gráfico de densidade empírica
                                                                                                  #
#-----------------------------------------------------------------------------------------------
# densidade empírica
plot(density(hib$rendimento))
plot(density(hib$rendimento, bw=200)) # mexe na largura da banda
rug(hib$rendimento)
                                                                                                  #
#-----------------------------------------------------------------------------------------------
# tipos de kernel
kernels <- eval(formals(density.default)$kernel) # extrai o tipo de kernels
kernels
plot(density(0, from=-1.2, to=1.2, width=1, kernel="gaussian"), type="l",
     ylim = c(0, 2), xlab="", main="R's density() kernels with width = 1")
for(i in 2:length(kernels))
  lines(density(0, width=1, kernel=kernels[i]), col=i)
legend(0.5, 2.0, legend=kernels, col=seq(kernels), lty=1, bty="n")
                                                                                                  #
#-----------------------------------------------------------------------------------------------
# gráfico de densidade: largura de banda e tipo de função kernel
density.panel <- function(panel){
  aux <- density(panel$x, width=panel$width, kernel=panel$kernel)
  plot(aux)
  rug(panel$x)
  lim <- par()$usr
  arrows(panel$c0-0.5*panel$width, 0, panel$c0+0.5*panel$width, 0,
         length=0.1, code=3, angle=90, col=2)
  y0 <- approx(aux$x, aux$y, xout=panel$c0)
  arrows(panel$c0, 0, panel$c0, y0$y, length=0.1, col=2)
  d <- density(panel$c0, width=panel$width, kernel=panel$kernel)
  lines(d$x, d$y/panel$n, col=2)
  panel
}
panel <- rp.control(x=hib$rendimento, n=length(hib$rendimento))
rp.slider(panel, width, 10, 1000, initval=150, showvalue=TRUE, action=density.panel)
rp.slider(panel, c0, 3500, 6500, initval=5000, showvalue=TRUE, action=density.panel)
rp.radiogroup(panel, kernel,
              c("gaussian","epanechnikov","rectangular","triangular","biweight",
                "cosine","optcosine"), action=density.panel)
                                                                                                  #
#-----------------------------------------------------------------------------------------------
# obtendo valores de área abaixo da curva de densidade = probabilidade
dens <- density(hib$rendimento, bw=150)
plot(dens)
```

```
rug(hib$rendimento)
fdens <- approxfun(dens$x, dens$y) # cria uma função que é interpolação linear
i <- 4500; s <- 5500               # limites inferior e superior
integrate(fdens, i, s)             # faz uma integração numérica
polygon(c(i,dens$x[dens$x>i & dens$x<s],s),
        c(0,dens$y[dens$x>i & dens$x<s],0), col="gray90")
i <- 3600; s <- 6800               # limites inferior e superior
integrate(fdens, i, s)             # faz uma integração numérica
polygon(c(i,dens$x[dens$x>i & dens$x<s],s),
        c(0,dens$y[dens$x>i & dens$x<s],0), col="gray90")
                                                                                        #
#---------------------------------------------------------------------------------------
# juntando histograma com densidade
hist(hib$rendimento, freq=FALSE)
lines(density(hib$rendimento), col=2)
rug(hib$rendimento)
                                                                                        #
#---------------------------------------------------------------------------------------
# distribuição de frequências acumuladas
h <- hist(hib$rendimento, plot=FALSE)
str(h)
h$counts <- cumsum(h$counts)
h$itensities <- h$density
plot(h, freq=TRUE, main="(Cumulative) histogram of x",
     col="navajowhite2", border="turquoise3")
box()
rug(hib$rendimento)
                                                                                        #
#---------------------------------------------------------------------------------------
# distribuição acumulada empírica
plot(ecdf(hib$rendimento))
rug(hib$rendimento)
                                                                                        #
#---------------------------------------------------------------------------------------
# momento naftalina: diagrama de ramos e folhas
stem(x)
                                                                                        #
#---------------------------------------------------------------------------------------
.
```

3.2 Medidas de posição, dispersão e forma

```
.
#---------------------------------------------------------------------------------------
# média amostral
x <- hib$rendimento # vetor com valores observados
mean(x)             # calcula a média da amostra
                                                                                        #
#---------------------------------------------------------------------------------------
# média ponderada amostral, uso em dados agrupados
ht <- hist(hib$rendimento, breaks=c(3900,4500,5000,5500,6000,6400))
str(ht)                           # armazena os pontos médios e pesos
weighted.mean(ht$mids, ht$counts) # calcula a média para dados agrupados
                                                                                        #
#---------------------------------------------------------------------------------------
# variância (desvio padrão) amostral
var(x) # calcula a variância amostral, denominador n-1
sd(x)  # desvio padrão amostral
                                                                                        #
#---------------------------------------------------------------------------------------
# mediana amostral
median(x)
                                                                                        #
#---------------------------------------------------------------------------------------
# desvio absoluto da mediana
sum(abs(x-median(x)))/length(x)
                                                                                        #
#---------------------------------------------------------------------------------------
# coeficiente de variação (cv%)
100*sd(x)/mean(x)
                                                                                        #
#---------------------------------------------------------------------------------------
# máximo, mínimo e amplitude
max(x)         # calcula o máximo
min(x)         # calcula o mínimo
range(x)       # ambos
```

```
diff(range(x)) # faz a diferença consecutiva entre elementos, retorna a amplitude
IQR(x)         # amplitude interquatílica
fivenum(x)     # 5 números de Tukey: 3 quartis e 2 extremos
                                                                                                #
#-------------------------------------------------------------------------------------------
# quartis
summary(x)
                                                                                                #
#-------------------------------------------------------------------------------------------
# percentis
quantile(x, probs=c(0.05,0.1,0.25,0.5,0.75,0.9,0.95)) # retorna os quantis, default é o 7
                                                                                                #
#-------------------------------------------------------------------------------------------
# existem 9 formas de calcular os quantis, veja os resultados
help(quantile, help_type="html")
mapply(quantile, type=1:9, MoreArgs=list(x=x, probs=c(5,10,25,50,75,90,95)/100))
                                                                                                #
#-------------------------------------------------------------------------------------------
# momentos de ordem r
m.r <- function(x, r){
  m <- mean(x)
  d <- (x-m)^r
  sum(d)/length(d)
}
m.r(x, 2) # variância populacional
var(x)*(length(x)-1)/length(x)
m.r(x, 3) # assimetria
m.r(x, 4) # curtose
                                                                                                #
#-------------------------------------------------------------------------------------------
# coeficiente de assimetria
m.r(x, 3)/var(x)^(3/2)
                                                                                                #
#-------------------------------------------------------------------------------------------
# coeficiente de curtose
m.r(x, 4)/var(x)^2-3
                                                                                                #
#-------------------------------------------------------------------------------------------
# todas essas funções estão disponíveis no pacote fBasics
install.packages("fBasics", dependencies=TRUE) # instala pacotes com dependências
require(fBasics)                               # carrega o pacote para o uso
basicStats(x)                                  # aplica a função do pacote
basicStats                                     # exibe o código da função (posso modificar)
                                                                                                #
#-------------------------------------------------------------------------------------------
# gráfico de caixas (boxplot)
boxplot(x)
                                                                                                #
#-------------------------------------------------------------------------------------------
# gráfico de caixa entalhado (notched boxplot)
boxplot(x, notch=TRUE) # é um IC para a mediana
                                                                                                #
#-------------------------------------------------------------------------------------------
# critério de expulsão dos pontos do boxplot
bp.panel <- function(panel){
  x <- rep(1:10, 2)
  x[2] <- panel$extreme
  gr <- gl(2, 10)
  bp <- boxplot(x~gr, range=panel$range, plot=FALSE)
  inf <- bp$stats[4,2]
  sup <- inf+panel$range*diff(bp$stats[c(2,4),2])
  ylim <- extendrange(r=c(min(x), max(c(x,sup))), f=0.05)
  boxplot(x~gr, range=panel$range, ylim=ylim)
  arrows(1.5, inf, 1.5, sup, angle=90, code=3, length=0.1)
  panel
}
panel <- rp.control()
rp.slider(panel, range, 0.5, 4, initval=1.5, showvalue=TRUE, action=bp.panel)
rp.slider(panel, extreme, 10, 20, initval=10, showvalue=TRUE, action=bp.panel)
                                                                                                #
#-------------------------------------------------------------------------------------------
.
```

3.3 Mais alguns gráficos

```
.
#-------------------------------------------------------------------------------------------
# gráfico para duas contínuas, diagrama de dispersão
plot(altura~espiga, data=hib)
                                                                                           #
#-------------------------------------------------------------------------------------------
# gráfico para contínua em função de categórica
plot(rendimento~resistencia, data=hib)
                                                                                           #
#-------------------------------------------------------------------------------------------
# gráfico para categórica em função de categórica
mosaicplot(~grao+resistencia, data=hib) # grao, resistencia|grao
mosaicplot(~resistencia+grao, data=hib) # resistencia, grao|resistencia
# na situação de independência a malha é regular
                                                                                           #
#-------------------------------------------------------------------------------------------
# gráficos de séries no tempo
met <- read.table("../dados/metereologia.txt", header=TRUE, sep="\t", stringsAsFactors=FALSE)
str(met)
names(met) <- tolower(names(met)) # passa os nomes para minúsculas
met$data <- as.Date(met$data)     # converte para data
                                                                                           #
#-------------------------------------------------------------------------------------------
# gráfico da temperatura
plot(temp~data, data=met)
plot(temp~data, data=met, type="l")
                                                                                           #
#-------------------------------------------------------------------------------------------
# conversão com a função as.POSIX*
met$data <- as.POSIXct(format(met$data, "%Y-%m-%d"), format="%Y-%m-%d")
plot(temp~data, data=met, type="l", xaxt="n")
axis.POSIXct(1, at=seq(as.POSIXct("2009-10-01"), as.POSIXct("2011-10-01"), by="month"), format="%B", col=2)
plot(temp~data, data=met, type="l", xaxt="n")
axis.POSIXct(1, at=seq(as.POSIXct("2009-10-01"), as.POSIXct("2011-10-01"), by="month"), format="%m/%y", col=2)
                                                                                           #
#-------------------------------------------------------------------------------------------
# duas séries em um mesmo gráfico
par(mar=c(5.1,4.1,4.1,4.1))
plot(temp~data, data=met, type="l", xaxt="n")
par(new=TRUE)
plot(urel~data, data=met, type="l", xaxt="n", yaxt="n", ylab=NA, col=2)
axis(4, col=2)
mtext("urel", side=4, line=3)
axis.POSIXct(1, at=seq(as.POSIXct("2009-10-01"), as.POSIXct("2011-10-01"), by="month"), format="%m/%y")
                                                                                           #
#-------------------------------------------------------------------------------------------
# série de máximo e mínimo
matplot(met$data, cbind(met$tmax, met$tmin), type="l", xaxt="n",
        ylab="Amplitude térmica diária", xlab="Data (mês/ano)")
axis.POSIXct(1, at=seq(as.POSIXct("2009-10-01"), as.POSIXct("2011-10-01"), by="month"), format="%m/%y")
polygon(c(met$data, rev(met$data)), c(met$tmin, rev(met$tmax)), col="gray90")
                                                                                           #
#-------------------------------------------------------------------------------------------
# gráfico de teias
require(fmsb)
uis <- function(x){# unit interval scale
  (x-min(x))/diff(range(x))
}
aux <- apply(hib[,2:5], 2, uis)
radarchart(as.data.frame(aux[1:5,]), maxmin=FALSE)
                                                                                           #
#-------------------------------------------------------------------------------------------
# diagrama ternário para mistura de 3 componentes
require(plotrix)
data(soils)
triax.plot(soils, show.grid=TRUE)
                                                                                           #
#-------------------------------------------------------------------------------------------
# gráfico radial, útil (às vezes) para dados ciclicos
aux <- met
str(aux)
x <- seq(0, 4*pi, l=nrow(aux))
sx <- aux$temp
radial.plot(sx, x, rp.type="p", line.col=1, radial.lim=c(0,40),
            labels=c(10:12,1:9))
                                                                                           #
#-------------------------------------------------------------------------------------------
.
```

4 Regressão linear

4.1 Preparação dos dados

```
.
#-------------------------------------------------------------------------------------------
# importando dados
#dap <- read.table("http://www.leg.ufpr.br/~walmes/cursoR/cnpaf/dap.txt", header=TRUE, sep="\t")
dap <- read.table("../dados/dap.txt", header=TRUE, sep="\t")
str(dap)
names(dap) <- c("d","h")
                                                                                          #
#-------------------------------------------------------------------------------------------
# associação entre as variáveis
plot(h~d, dap) # ou plot(dap), só plota os casos completos
sum(!is.na(dap$d))
sum(!is.na(dap$h))
                                                                                          #
#-------------------------------------------------------------------------------------------
# criando novas variáveis para ajustar modelos para predição de h via funções de d
dap$d2 <- dap$d^2
dap <- transform(dap, d2=d^2, d3=d^3, dr=sqrt(d), dl=log(d), di=1/d, di2=1/d^2)
str(dap)
                                                                                          #
#-------------------------------------------------------------------------------------------
# buscando por relações marginais
plot(dap) # usa pairs()
                                                                                          #
#-------------------------------------------------------------------------------------------
# ordena os dados e deixa apenas os registros completos (linha cheia)
dap <- dap[order(dap$d),]          # ordenar para evitar efeito espaguete quando plotar
dapcc <- dap[complete.cases(dap),] # nova base que contém d e h observados
rownames(dapcc) <- NULL            # reseta o nome das linhas
head(dapcc)
str(dapcc)
                                                                                          #
#-------------------------------------------------------------------------------------------
.
```

4.2 Regressão linear simples

```
.
#-------------------------------------------------------------------------------------------
# ajustando a equação da reta (regressão linear simples)
m0 <- lm(h~d, data=dapcc)
summary(m0)  # quadro de estimativas dos parâmetros
                                                                                          #
#-------------------------------------------------------------------------------------------
# matriz do modelo
head(model.matrix(m0))
                                                                                          #
#-------------------------------------------------------------------------------------------
# coisas que o objeto m0 armazena
names(m0)
str(m0)
                                                                                          #
#-------------------------------------------------------------------------------------------
# verificando a qualidade do ajuste
plot(h~d, dapcc) # xlab=, ylab=
lines(fitted(m0)~d, dapcc, col="black", lty=2, lwd=2) # adiciona a linha ajustada
abline(m0, col=3, lty=2)                              # adiciona a linha ajustada
                                                                                          #
#-------------------------------------------------------------------------------------------
# análise de resíduos para verificar as pressuposições do modelo, outlier (de graça!)
par(mfrow=c(2,2))  # divide a janela gráfica em 4 na disposição 2x2
plot(m0)           # apresenta os gráficos de diagnóstico
layout(1)          # volta para 1 gráfico por janela
                                                                                          #
#-------------------------------------------------------------------------------------------
# fazer um close up em apenas um gráfico
plot(m0, which=2) # apenas o gráfico de probabilidade normal
                                                                                          #
#-------------------------------------------------------------------------------------------
# todos os gráficos de análise de resíduos
```

```
par(mfrow=c(3,2))   # divide a janela gráfica em 6 na disposição 3x2
plot(m0, which=1:6) # apresenta os gráficos de diagnóstico
layout(1)           # volta para 1 gráfico por janela
                                                                                #
#-------------------------------------------------------------------------------
.
```

4.3 Regressão linear múltipla

```
.
#-------------------------------------------------------------------------------
# ajuste do modelo quadrático
m1 <- lm(h~d+d2, data=dapcc) # ou lm(h~d+I(d^2), data=dapcc)
head(model.matrix(m1))
summary(m1)
anova(m1)
                                                                                #
#-------------------------------------------------------------------------------
# gráficos
layout(matrix(c(1,1,2,3,4,5),2,3)) # reparte em 6 para caber 5 gráficos
plot(h~d, dapcc)
lines(fitted(m1)~d, dapcc, col=2)
plot(m1)
                                                                                #
#-------------------------------------------------------------------------------
# modelo cúbico
m2 <- lm(h~d+d2+d3, data=dapcc) # ou lm(h~d+I(d^2)+I(d^3), data=dapcc)
head(model.matrix(m2))
summary(m2)
anova(m2)
                                                                                #
#-------------------------------------------------------------------------------
# gráficos
plot(h~d, dapcc)
lines(fitted(m2)~d, dapcc, col=2)
plot(m2)
                                                                                #
#-------------------------------------------------------------------------------
# modelo recíproco
m3 <- lm(h~d+di, data=dapcc)
summary(m3)
                                                                                #
#-------------------------------------------------------------------------------
# gráficos
plot(h~d, dapcc); lines(fitted(m3)~d, dapcc, col=2); plot(m3)
                                                                                #
#-------------------------------------------------------------------------------
# modelo quadrado do recíproco
m4 <- lm(h~d+di2, data=dapcc)
summary(m4)
                                                                                #
#-------------------------------------------------------------------------------
# gráficos
plot(h~d, dapcc); lines(fitted(m4)~d, dapcc, col=2); plot(m4)
                                                                                #
#-------------------------------------------------------------------------------
# modelo raíz quadrada
m5 <- lm(h~d+dr, data=dapcc)
summary(m5)
plot(h~d, dapcc); lines(fitted(m5)~d, dapcc, col=2); plot(m5)
                                                                                #
#-------------------------------------------------------------------------------
# modelo logarítmo
m6 <- lm(h~d+dl, data=dapcc)
summary(m6)
plot(h~d, dapcc); lines(fitted(m6)~d, dapcc, col=2); plot(m6)
                                                                                #
#-------------------------------------------------------------------------------
.
```

4.4 Para ajustar vários modelos de uma única vez

```
.
#-------------------------------------------------------------------------------------------
# cria a lista com as fórmulas dos modelos
formulas <- list(linear=h~d, quadratico=h~d+d2, cubico=h~d+d2+d3,
                 reciproco=h~d+di, reciproco2=h~d+di2,
                 raiz=h~d+dr, log=h~d+dl)
formulas
                                                                                          #
#-------------------------------------------------------------------------------------------
# com a lapply(), o ajuste será feito usando cada uma das fórmulas da lista
ajustes <- lapply(formulas,
                  function(f){
                    m <- lm(f, data=dap); m
                  })
names(ajustes)
                                                                                          #
#-------------------------------------------------------------------------------------------
# com a lapply() é possível obter todos os resultados do ajuste e gráficos
lapply(ajustes, summary)
lapply(ajustes, anova)
par(mfrow=c(3,3))
lapply(ajustes, plot, which=1)
layout(1)
r2 <- lapply(ajustes, function(a){ summary(a)$adj.r.squared })
r2
unlist(r2)
do.call(c, r2)
plot(unlist(r2), type="h")
                                                                                          #
#-------------------------------------------------------------------------------------------
.
```

4.5 Exportando os resultados para arquivos de texto e imagens

```
.
#-------------------------------------------------------------------------------------------
# exportando os resultados numéricos
sink("tabelaestimativas.txt")
lapply(ajustes, summary)
sink()
                                                                                          #
#-------------------------------------------------------------------------------------------
# exportando os gráficos de resíduos para um modelos apenas, em png
png("residuos.png") # png("residuos.png", width=400, height=400)
par(mfrow=c(2,2))
plot(m0)
dev.off() # fecha o dispositivo gráfico aberto para salvar a figura em png
                                                                                          #
#-------------------------------------------------------------------------------------------
# salvando em pdf
pdf("residuos.pdf")
par(mfrow=c(2,2))
plot(m0)
dev.off()
                                                                                          #
#-------------------------------------------------------------------------------------------
# salvar todos os gráficos de resíduos
pdf("todos.pdf")
par(mfrow=c(2,2))
lapply(ajustes, plot)
dev.off()
                                                                                          #
#-------------------------------------------------------------------------------------------
.
```

4.6 Procedimentos para seleção de modelos/variáveis

```
.
#-------------------------------------------------------------------------------------------
# modelo com todas as variáveis
m7 <- lm(h~., data=dapcc) # o ponto "." indica "todas as variáveis restante do arquivo"
summary(m7)
```

```
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# gráficos
layout(matrix(c(1,1,2,3,4,5),2,3))
plot(h~d, dapcc); lines(fitted(m7)~d, dapcc, col=2); plot(m7)
layout(1)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# seleção de modelos/variáveis (both, forward, backward)
step(m7, direction="both")                          # por padrão é o AIC: ll-2*p
step(m7, direction="both", k=log(nrow(dapcc)))      # assim é o BIC: ll-log(n)*p
step(m7, direction="both", k=4)                     # você usar um peso arbitrário
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# modelo m5 foi escolhido pelo critério BIC
summary(m5)
anova(m5)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# gráficos
layout(matrix(c(1,1,2,3,4,5),2,3))
plot(h~d, dapcc); lines(fitted(m5)~d, dapcc, col=2); plot(m5)
layout(1)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
.
```

4.7 Estudo e remoção de pontos discrepantes/influentes

```
.
#-------------------------------------------------------------------------------------------------
# medidas de influência das observações (leave one out)
inf <- influence.measures(m5)
inf
summary(inf) # resumo das observações influentes pelas diferentes medidas de influência
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# sinalizando os pontos influentes
str(inf)                 # estrutura do objeto
dfits <- inf$is.inf[,4]  # influentes pelo DFITS (influência sobre ajuste)
plot(h~d, dapcc)
lines(fitted(m5)~d, dapcc, col=2)
with(dapcc, points(d[dfits], h[dfits], col=2, pch=19))
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# identificar/remover os pontos discrepantes/influentes manualmente (critério olho)
plot(residuals(m5)~d, dapcc)
id <- identify(dapcc$d, residuals(m5))
id
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# refazer a análise com os pontos removidos
dapcc2 <- dapcc[-id,]
str(dapcc2)
m5b <- lm(h~d+dr, data=dapcc2)
summary(m5b)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# gráficos
layout(matrix(c(1,1,2,3,4,5),2,3))
plot(h~d, dapcc2)
lines(fitted(m5b)~d, dapcc2, col=2) # modelo com as remoções
lines(fitted(m5)~d, dapcc, col=3)   # modelo com todas observações
plot(m5b)
layout(1)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# denconfia da normalidade? devemos tranformar? qual transformação usar?
plot(m5b, which=2)
require(MASS)
layout(1)
boxcox(m5b, lambda=seq(0.5,2,l=100)) # intervalo contém o 1?
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# qual o resultado dos testes?
shapiro.test(rstudent(m5b))
```

```
ks.test(rstudent(m5b), "pnorm")
                                                                                                    #
#-------------------------------------------------------------------------------------------------
.
```

4.8 Predição de valores a partir do modelo escolhido

```
.
#-------------------------------------------------------------------------------------------------
# tudo para encontrar o modelo, vamos predizer a artura das árvores e salvar num arquivo
hpred <- predict(m5b, newdata=dap)
str(hpred)
dap$hpred <- hpred
str(dap)
write.table(dap, "dap.xls", sep="\t", quote=FALSE, row.names=FALSE, dec=".")
                                                                                                    #
#-------------------------------------------------------------------------------------------------
.
```

4.9 Representação gráfica do ajuste

```
.
#-------------------------------------------------------------------------------------------------
# escolhendo o intervalo de predição
range(dapcc2$d)
d.new <- seq(4, 30, length=100)
d.new
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# fazendo predição com intervalo de confiança e predição futura
Yp <- predict(m5b, newdata=data.frame(d=d.new, dr=sqrt(d.new)), interval="confidence")
Yf <- predict(m5b, newdata=data.frame(d=d.new, dr=sqrt(d.new)), interval="prediction")
head(Yp)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# plotando
layout(1)
plot(h~d, dapcc2, xlab="DAP (cm)", ylab="Altura (m)") # diagrama de dispersão
matlines(d.new, Yp, col=c(1,2,2), lty=c(1,2,2))       # bandas de confiança
matlines(d.new, Yf, col=c(1,3,3), lty=c(1,3,3))       # bandas de predição
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# fazendo anotações dentro do gráfico, incluindo a legenda das linhas
legend("topleft",
       c("Predito","ICpredito","ICobsfutura"),
       lty=c(1,2,3), col=c(1,2,3), bty="n")
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# inclusão da expressão do modelo
co <- format(c(abs(coef(m5b)), summary(m5b)$r.squared), digits=3)
co <- gsub(" ", "", co)
sinal <- ifelse(coef(m5b)<0, "-", "+")
co[seq_along(sinal)] <- paste(sinal, co[seq_along(sinal)])
text(20, 15,
     label=substitute(hat(h)==b0~b1*d~b2*sqrt(d)~~~~(R^2==r2),
       list(b0=co[1], b1=co[2], b2=co[3], r2=co[4])), bty="n")
text(par()$usr[2], par()$usr[3], adj=c(1.03,-0.3), offset = 2.5, # no cantinho de baixo
     label=substitute(hat(h)==b0~b1*d~b2*sqrt(d)~~~~(R^2==r2),
       list(b0=co[1], b1=co[2], b2=co[3], r2=co[4])))
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# mais sobre gráficos no R
demo(plotmath)
demo(graphics)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
.
```

4.10 Mais sobre análise de resíduos e R^2 (quarteto de Anscombe)

```
.
#-------------------------------------------------------------------------------------------
# mais sobre resíduos e R2
data(anscombe) # disponibiliza o arquivo para uso
form <- list(a1=y1~x1, a2=y2~x2, a3=y3~x3, a4=y4~x4)
ajus <- lapply(form, function(a) lm(a, data=anscombe))
lapply(ajus, summary)
lapply(ajus, data)
anscombe
                                                                                          #
#-------------------------------------------------------------------------------------------
# gráficos
par(mfrow=c(4,5), oma=c(0,0,0,0), mar=c(2,2,2,2))
lapply(ajus,
       function(a){
         plot(a$model[,1]~a$model[,2])
         abline(a, col=2)
         plot(a)
       })
                                                                                          #
#-------------------------------------------------------------------------------------------
# o significado dos leverages
cbind(sapply(ajus, hatvalues), anscombe[,1:4])
                                                                                          #
#-------------------------------------------------------------------------------------------
# mais sobre medidas de influência (animação)
require(rpanel)
help(rp.control, help_type="html")
data(CofE)
attach(CofE)
rp.regression(Employ, Giving)
                                                                                          #
#-------------------------------------------------------------------------------------------
.
```

4.11 Sobre interpretação de modelos de regressão polinomial

```
#-------------------------------------------------------------------------------------------
# simulando uma resposta gerada por polinômio com a seguinte lei
dev.off()
b0 <- 1; b1 <- 5; b2 <- -0.2; b3 <- 0.002    # valores paramétricos/populacionais
x <- seq(0,75,by=5)                          # valores de x avaliados
det.seed(504)
y <- b0+b1*x+b2*x^2+b3*x^3+rnorm(x,0,25)     # resposta observada
plot(y~x)
curve(b0+b1*x+b2*x^2+b3*x^3, add=TRUE)       # comportamento populacional
                                                                                          #
#-------------------------------------------------------------------------------------------
# ajustar modelo fazendo translação do x
d <- seq(-20,20,by=5); names(d) <- d
m0.list <- lapply(d,
                  function(d){
                    z <- x-d; z2 <- z^2; z3 <- z^3
                    m0 <- lm(y~z+z2+z3); m0
                  })
do.call(c, lapply(m0.list, function(m0) summary(m0)$r.squared))  # R^2
lapply(m0.list, function(m0) summary(m0)$coeff)                  # estimativas/testes marginais
lapply(m0.list, function(m0) cov2cor(vcov(m0)))                  # matriz de correlação
lapply(m0.list, function(m0) anova(m0))                          # anova sequencial
lapply(m0.list, function(m0) model.matrix(m0)[1:3,])             # matrizes dos modelos
                                                                                          #
#-------------------------------------------------------------------------------------------
.
```

4.12 Regressão linear múltipla (2)

```
.
#-------------------------------------------------------------------------------------------
# importação e higienização
```

```
chi <- read.table("../dados/chimarrita.txt", header=TRUE, sep="\t")
chi <- chi[complete.cases(chi),]
str(chi)
                                                                                                    #
#-------------------------------------------------------------------------------------------
# modelar o tamanho da lesão60 como função das demais variáveis observadas no instante 0
vars <- c("lesao60", grep("[a-z]0$", names(chi), value=TRUE))
chi2 <- subset(chi, fer=="sim", select=vars)
str(chi2)
car::scatterplotMatrix(chi2)
round(cor(chi2),3)
                                                                                                    #
#-------------------------------------------------------------------------------------------
# ajuste do modelo completo, todos os efeitos principais e interações duplas
m0 <- lm(lesao60~(.)^2, data=chi2)
summary(m0)
par(mfrow=c(2,2))
plot(m0)
layout(1)
                                                                                                    #
#-------------------------------------------------------------------------------------------
# usar um step
m1 <- step(m0, k=log(nrow(chi2))) # usando o BIC
summary(m1)
anova(m1, m0)
par(mfrow=c(2,2))
plot(m1)
layout(1)
                                                                                                    #
#-------------------------------------------------------------------------------------------
# fazendo gráficos para predição da resposta
apply(chi2, 2, range) # amplitude das variáveis da amostra
pred <- with(chi2, expand.grid(cor0=seq(min(cor0), max(cor0), l=10),
                               ms0=seq(min(ms0), max(ms0), l=10),
                               ss0=seq(min(ss0), max(ss0), l=3),
                               b0=seq(min(b0), max(b0), l=3)))
str(pred)
pred$lesao60 <- predict(m1, newdata=pred)
                                                                                                    #
#-------------------------------------------------------------------------------------------
# gráfico de 3 dimensões
require(lattice)
wireframe(lesao60~cor0+ms0|ss0+b0, data=pred)
wireframe(lesao60~cor0+ms0|ss0+b0, data=pred, drape=TRUE)
                                                                                                    #
#-------------------------------------------------------------------------------------------
# melhorando o aspecto visual
require(latticeExtra)
colscale <- colorRampPalette(c("yellow", "red"), space="rgb")
wf <- wireframe(lesao60~cor0+ms0|ss0+b0, data=pred,
                drape=TRUE, col.regions=colscale(90))
useOuterStrips(wf,
               strip=strip.custom(bg="gray90"),
               strip.left=strip.custom(bg="gray50"))
                                                                                                    #
#-------------------------------------------------------------------------------------------
# pode-se fazer um componentes principais, análise fatorial, arvore de regressão...
                                                                                                    #
#-------------------------------------------------------------------------------------------
.
```

4.13 Regressão para mistura de duas componentes

```
.
#-------------------------------------------------------------------------------------------
# dados de mistura de duas componentes: restrição que a soma dá um, predição é para x
# no intervalo unitário, reparemetrização torna mais clara a interpretação
mis <- read.table("../dados/biomassa.txt", header=TRUE, sep="\t")
str(mis)
names(mis) <- tolower(names(mis)) # passando nomes das colunas para minúsculo
                                                                                                    #
#-------------------------------------------------------------------------------------------
# por enquanto vamos análisar apenas uma espécie e sem usar os tratamentos adicionais
mis2 <- subset(mis, pla=="E" & !trat%in%c("BJ","Na +"))
str(mis2)
mis2
```

```
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# fazendo o gráficos de todas as respostas
require(reshape) # help(melt, package="reshape", help_type="html")
aux <- melt(mis2, id.vars="k", measure.vars=8:20, variable_name="resposta") # empilhamento
str(aux)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# gráfico
require(lattice)
xyplot(value~k|resposta, data=aux, type=c("p","a","smooth","r"), scales="free")
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# análise do modelo de regressão quadrático em K com parametrização usual
m0 <- lm(dc~k+I(k^2), data=mis2)
summary(m0)
par(mfrow=c(2,2)); plot(m0); layout(1)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# análise do modelo de regressão (quadrático) mas com a parametrização de mistura
m1 <- lm(dc~-1+k+na+k:na, data=mis2)
m1 <- lm(dc~-1+k+I(1-k)+k:I(1-k), data=mis2)
summary(m1) # vantagem é a interpretação dos parâmetros
par(mfrow=c(2,2)); plot(m1); layout(1)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# gráfico do ajuste do modelo
k <- seq(0,1,l=25)
y <- predict(m1, newdata=data.frame(k=k), interval="confidence")
plot(dc~k, data=mis2)
matlines(k, y, type="l", col=c(1,2,2), lty=c(1,2,2))
                                                                                                    #
#-----------------------------------------------------------------------------------------------
.
```

4.14 Contrastes em um modelo de regressão linear

```
.
#-----------------------------------------------------------------------------------------------
# a dose de melhor mistura é diferente dos componentes puros?
kmax <- (m1$coef[3]+m1$coef[1]-m1$coef[2])/(2*m1$coef[3])
names(kmax) <- NULL # retira que ele vai atrapalhar lá na frente
abline(v=kmax)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# fazendo os contrastes matricialmente
b <- coef(m1)
m <- rbind(k0=c(0,1,0*1),
           kmax=c(kmax,1-kmax,kmax*(1-kmax)))
mc <- as.matrix(apply(m, 2, diff)); mc    # vetor coluna dos coeficientes do contraste
est <- t(mc)%*%b; est                     # estimativa do contraste
vco <- t(mc)%*%vcov(m1)%*%mc; sqrt(vco)   # variância, erro-padrão do contraste
Fc <- est%*%solve(vco)%*%t(est); sqrt(Fc) # valor F, t calculado do contraste
pf(Fc, 1, df.residual(m1), lower.tail=FALSE)      # p-valor F
pt(sqrt(Fc), df.residual(m1), lower.tail=FALSE)*2 # p-valor t
#-----------------------------------------------------------------------------------------------
# usando a gmodels::glh.test()
require(gmodels)
glh.test(m1, c(mc))
#-----------------------------------------------------------------------------------------------
# usando a gmodels::estimable(), detalhe: o vetor passado não pode ter atributo names
estimable(m1, cm=c(1,0,0))                              # potássio puro
estimable(m1, cm=c(0,1,0))                              # sódio puro
estimable(m1, cm=c(0.5,0.5,0.25))                       # mistura 1:1
estimable(m1, cm=c(kmax,1-kmax,kmax*(1-kmax)))          # melhor mistura (ops!)
estimable(m1, cm=c(0.66,0.33,0.66*0.33))                # passando os valores na mão
estimable(m1, cm=as.vector(c(kmax,1-kmax,kmax*(1-kmax)))) # retirando os nomes
estimable(m1, cm=as.vector(mc))                         # passando um contraste
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# usando a contrast::contrast(), vantagem: simples de usar
require(contrast)
contrast(m1, list("k"=kmax, "I(1 - k)"=1-kmax))
contrast(m1, list("k"=kmax, "I(1 - k)"=1-kmax), list("k"=0, "I(1 - k)"=1-0))
contrast(m1, list("k"=kmax, "I(1 - k)"=1-kmax), list("k"=1, "I(1 - k)"=1-1))
contrast(m1, list("k"=1, "I(1 - k)"=1-1), list("k"=0, "I(1 - k)"=1-0))
```

```
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# se o número de contrastes for alto, no teste t eleva-se rapidamente a taxa de erro tipo 1
                                                                                                    #
#-------------------------------------------------------------------------------------------------
.
```

4.15 Intervalo de confiança para uma função de parâmetros

```
.
#-------------------------------------------------------------------------------------------------
# a dose do ponto de máximo é uma função de parâmetros, como obter um IC para ela
(m1$coef[3]+m1$coef[1]-m1$coef[2])/(2*m1$coef[3])
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# uma possível solução seria usar o método delta, de fácil uso e disponível na car
require(car)
coef(m1)
dm <- deltaMethod(m1, g="0.5*(b3+b1-b2)/b3", parameterNames=c("b1","b2","b3"))
dm
dm <- unlist(dm)
dm[1]+c(-1,0,1)*1.96*dm[2] # é uma aproximação de 1 ordem
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# podemos fazer um intervalo de confiança baseado em amostras bootstrap
# (não paramétrico, faz amostragem com reposição e estima os parâmetros)
boot <- bootCase(m1, B=1000) # demora um pouco
str(boot)                    # são as amostras dos parâmetros
colnames(boot) <- names(coef(m1))
apply(boot, 2, quantile, probs=c(0.025, 0.975))
t(confint(m1))
par(mfrow=c(3,1))
apply(boot, 2, function(x) hist(x))
apply(boot, 2, function(x) plot(density(x)))
layout(1)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# agora é aplicar a função a cada linha da amostra e fazer o IC
kmax.a <- apply(boot, 1, function(x){ 0.5*(x[3]+x[1]-x[2])/x[3] })
plot(density(kmax.a, n=512, from=-1, to=2), xlim=c(0,1.5))
rug(kmax.a)
kmax.ic <- quantile(kmax.a, probs=c(0.025,0.975))
kmax.ic
abline(v=kmax.ic) # IC fora de (0,1) porque b3 é não significativo
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# reduzindo a variabilidade de propósito apenas para mostrar a utilidade do método
# ISSO É PARA MOSTRAR O PROCEDIMENTO, NÃO SE DEVE FAZER
mis2$dc2 <- fitted(m1)+0.8*residuals(m1) # somando o ajustado à 0.8 do desvio
plot(dc2~k, mis2)
m2 <- lm(dc2~-1+k+I(1-k)+k:I(1-k), data=mis2)
coef(m2)
boot <- bootCase(m2, B=1000)
kmax.a <- apply(boot, 1, function(x){ 0.5*(x[3]+x[1]-x[2])/x[3] })
plot(density(kmax.a, n=512, from=-1, to=2), xlim=c(0,1.5))
rug(kmax.a)
kmax.ic <- quantile(kmax.a, probs=c(0.025,0.975))
kmax.ic
abline(v=kmax.ic) # IC fora de (0,1) porque b3 é não significativo
                                                                                                    #
#-------------------------------------------------------------------------------------------------
.
```

Análisar os dados do Maycom (diametro do caule, matéria seca das folhas, matéria seca do caule, fazer uma análise fatorial). Esses dados possem o número de folhas ao longo do tempo. É possível usar a contagem subdispersa e incluir efeito aleatório!

5 Regressão não linear

5.1 Motivação

```
.
#-------------------------------------------------------------------------------------------
# dados de motivação
#  dia: dias após aplicação do nematicída
#  eclod: número de ovos eclodidos de nematóide
lines <- "
  dia  eclod
    2  13.00
    4  56.50
    6  97.50
    8 168.00
   10 246.50
   12 323.00
   14 374.00
   16 389.00
"
da <- read.table(textConnection(lines), header=TRUE); closeAllConnections()
str(da)
plot(eclod~dia, da)
                                                                                          #
#-------------------------------------------------------------------------------------------
# ajuste de modelos lineares e não lineares
new <- data.frame(dia=seq(0,30,l=100))
plot(eclod~dia, da, xlim=c(0,30), ylim=c(0,600))
                                                                                          #
#-------------------------------------------------------------------------------------------
# modelo linear da reta
m0 <- lm(eclod~dia, da)
lines(predict(m0, newdata=new)~new$dia, col=1)
                                                                                          #
#-------------------------------------------------------------------------------------------
# modelo polinômio cúbico
m1 <- lm(eclod~poly(dia, 3), da)
lines(predict(m1, newdata=new)~new$dia, col=2)
                                                                                          #
#-------------------------------------------------------------------------------------------
# modelo não linear (logístico)
m2 <- nls(eclod~SSlogis(dia, Asym, xmid, scal), data=da)
lines(predict(m2, newdata=new)~new$dia, col=3)
                                                                                          #
#-------------------------------------------------------------------------------------------
# lineares:
# * em termos de ajuste: é uma aproximação local e portanto a predição fora do intervalo
# das covariáveis não deve ser feita
# * em termos de interpretação: é um modelo empírico, relevância teórica, interpretação em
# tem termos de taxa por unidade de incremento, nos polinômios interpretação é complicada
# * em termos de inferência: estimadores de MQ0/verossimilhança possuem expressão analítica
# inferências em pequenas amostras são exatas
                                                                                          #
#-------------------------------------------------------------------------------------------
# não lineares
# * em termos de ajuste: é um modelo globa e portanto a predição fora do é justificada
# * em termos de interpretação: é um modelo com obtido com suporte teórico, os parâmetros
# possuem significado que agrega interpretação
# * em termos de inferência: estimadores de MQ0/verossimilhança usam procedimentos
# numéricos (gargalo do chute inicial), inferências em pequenas amostras são aproximadas e
# dependem da parametrização do modelo e dos dados
                                                                                          #
#-------------------------------------------------------------------------------------------
.
```

5.2 Definição

```
.
#-------------------------------------------------------------------------------------------
# as derivadas de y em relação a theta não são livres de theta
# y = A*x/(B+x) : modelo michaelis mentem
mm <- expression(A*x/(B+x)) # expressão do modelo
D(mm, "A") # derivada simbólica de uma expressão
D(mm, "B")
                                                                                          #
#-------------------------------------------------------------------------------------------
# y = A/(1+exp(B+C*x)) : modelo logístico
logis <- expression(A/(1+exp(B+C*x)))
D(logis, "A")
D(logis, "B")
```

```
D(logis, "C")
mapply(D, name=c("A","B","C"), expr=logis) # calcula todas as derivadas de uma vez só
                                                                                                    #
#-------------------------------------------------------------------------------------------
.
```

5.3 Exemplo de modelos não lineares

```
.
#-------------------------------------------------------------------------------------------
# modelo michaelis mentem: f(x) = A*x/(B+x)
#  A: assintota (lim f(x) para x -> infinito)
#  B: tempo de meia vida (x para o qual f(x) = A/2)
layout(1)
A <- 10; B <- 3
curve(A*x/(B+x), 0, 50, ylim=c(0,10), col=2, lwd=3)
abline(h=c(A, A/2), v=B, lty=3)
                                                                                                    #
#-------------------------------------------------------------------------------------------
# se dividirmos numerador e denominador por B obtemos o ?monomolecular: f(x) = C*x/(1+x/B)
#   C: é a rezão entre assintota e tempo de meia vida pois C = A/B
#   B: é o tempo de meia vida
C <- A/B
curve(C*x/(1+x/B), add=TRUE, col=4, lwd=2, lty=2)
# além de mudar intepretação (nenhuma intepretação cartesiana ou biológica para C), muda-se
# as propriedades de estimação e inferência
                                                                                                    #
#-------------------------------------------------------------------------------------------
# modelo logístico, f(x) = A/(1+exp((B-x)/C))
#  A: assíntota (lim f(x) para x -> infinito)
#  B: tempo de meia vida (x para o qual f(x) = A/2)
#  C: ~escala, tamanho do intervalo de variação em x onde f(x) não é plana divido por 10
#  o valor (0.09886*A/C) é a aproximadamente a taxa média nesse intervalo de x
A <- 10; B <- 25; C <- 5
curve(A/(1+exp((B-x)/C)), 0, 50, col=2, lwd=3)
abline(h=c(A, A/2), v=B, lty=3)
rect(xleft=B-C*10/2, 0, B+C*10/2, A, density=5)
abline(a=A*(-B+C*10/2)/(10*C), b=A/(10*C), col=3)
xg <- seq(B-C*10/2, B+C*10/2,l=100)
yg <- A/(1+exp((B-xg)/C))
mean(diff(yg)/diff(xg))
0.09886*A/C
                                                                                                    #
#-------------------------------------------------------------------------------------------
# modelo resposta platô: f(x) = A+B*x se x<x0 e A+B*x0 se x>=x0
A <- 1; B <- 0.5; x0 <- 5
curve(A+B*x*(x<x0)+B*x0*(x>=x0),0, 20, col=2, lwd=3)
abline(h=c(A, A+B*x0), v=x0, lty=3)
                                                                                                    #
#-------------------------------------------------------------------------------------------
# modelo de produção-competição (Bleasdale & Nelder, 1960): f(x) = x*(A+B*x)^(-1/C)
#  A: ?
#  B: ?
#  C: ?
#  o valor x = A/(B*(1/C-1)) é o ponto de máximo da função, para C<1
A <- 10; B <- 2; C <- 0.5
curve(x*(A+B*x)^(-1/C), 0, 50, col=2, lwd=3); abline(v=A/(B*(1/C-1)))
C <- 1
curve(x*(A+B*x)^(-1/C), 0, 50, col=2, lwd=3)
C <- 2
curve(x*(A+B*x)^(-1/C), 0, 50, col=2, lwd=3)
                                                                                                    #
#-------------------------------------------------------------------------------------------
# modelo de curva de água (van Genuchten, 1980): f(x) = B+(A-B)/(1+(C*10^x)^n)^(1-1/n)
A <- 0.7; B <- 0.3; C <- 1.3; D <- 1.6
curve(B+(A-B)/(1+(C*10^x)^D)^(1-1/D), -3, 4, col=2, lwd=3)
abline(h=c(A,B), lty=3)
par(new=TRUE)
curve(eval(D(expression(B+(A-B)/(1+(C*10^x)^D)^(1-1/D)), "x")),
      -3, 4, yaxt="n", xaxt="n")
axis(4)
                                                                                                    #
#-------------------------------------------------------------------------------------------
.
```

5.4 Uso de recursos gráficos para entender o significado dos parâmetros

```
.
#-------------------------------------------------------------------------------------------
# pacote que permite a construção de interfaces gráficas
library(gWidgetsRGtk2)
options("guiToolkit"="RGtk2")
                                                                                           #
#-------------------------------------------------------------------------------------------
# modelo michaelis mentem (reações químicas, liberação de nutrientes no solo)
limits <- list(A=c(0,20), B=c(0,6))
plottest <- function(...){
  curve(svalue(A)*x/(svalue(B)+x), 0, 15, ylim=c(0,10))
  abline(v=svalue(B), h=svalue(A)/c(1,2), col=2)
}
                                                                                           #
#-------------------------------------------------------------------------------------------
# função que constrói a janela gráfica com deslizadores
#func <- '
w <- gwindow("Slider and spinbox example")
tbl <- glayout(cont=w)
for(i in 1:length(limits)){
  tbl[i,1] <- paste("Slide to adjuste parameter", names(limits)[i])
  tbl[i,2, expand=TRUE] <- (assign(names(limits)[i],
             gslider(from=limits[[i]][1], to=limits[[i]][2],
                     by=diff(limits[[i]])/20, value=mean(limits[[i]]),
                     container=tbl, handler=plottest)))
}
plottest()
#'; writeLines(func, "func.R")
                                                                                           #
#-------------------------------------------------------------------------------------------
# modelo logístico (curva de crescimento)
limits <- list(A=c(0,20), B=c(10,40), C=c(0.5,7))
plottest <- function(...){
  AA <- svalue(A); BB <- svalue(B); CC <- svalue(C)
  curve(AA/(1+exp((BB-x)/CC)), 0, 50, ylim=c(0,15))
  abline(h=c(AA, AA/2), v=BB, lty=3, col=2)
  rect(xleft=BB-CC*10/2, 0, BB+CC*10/2, AA, border="red")
}
source("func.R")
                                                                                           #
#-------------------------------------------------------------------------------------------
# modelo resposta platô (ensaios com fetilizante)
limits <- list(A=c(0,2), B=c(0,2), x0=c(2,7))
plottest <- function(...){
  AA <- svalue(A); BB <- svalue(B); x00 <- svalue(x0)
  curve(AA+BB*x*(x<x00)+BB*x00*(x>=x00), 0, 20, ylim=c(0,3))
  abline(h=c(AA, AA+BB*x00), v=x00, col=2)
}
source("func.R")
                                                                                           #
#-------------------------------------------------------------------------------------------
# modelo de produção-competição (Bleasdale & Nelder, 1960)
limits <- list(A=c(0,20), B=c(0,2), C=c(0,2))
plottest <- function(...){
  AA <- svalue(A); BB <- svalue(B); CC <- svalue(C)
  curve(x*(AA+BB*x)^(-1/CC), 0, 50, ylim=c(0,0.7))
  abline(v=AA/(BB*(1/CC-1)), col=2)
}
source("func.R")
                                                                                           #
#-------------------------------------------------------------------------------------------
# modelo de curva de água no solo (van Genuchten, 1980)
limits <- list(A=c(0.5,1), B=c(0,0.5), C=c(0,5), D=c(1,6))
plottest <- function(...){
  curve(svalue(B)+(svalue(A)-svalue(B))/(1+(svalue(C)*10^x)^svalue(D))^(1-1/svalue(D)),
        -3, 4, ylim=c(0,1))
}
source("func.R")
                                                                                           #
#-------------------------------------------------------------------------------------------
.
```

5.5 Estimação de parâmetros em modelos não lineares

```
.
#-------------------------------------------------------------------------------------------
# como funciona o procedimento iterativo para estimar parâmetros?
# exemplo com o modelo michaelis mentem e dados de mentirinha
theta <- c(A=10, B=3)
da <- data.frame(x=seq(1,20,2))
da$y <- theta["A"]*da$x/(theta["B"]+da$x)+rnorm(da$x,0,0.2)
plot(y~x, da)
                                                                                          #
#-------------------------------------------------------------------------------------------
# sequência de estimativas até a convergência do procedimento de estimação
# caminho pela superfície de mínimos quadrados
sqe <- function(A, B, y, x){ hy <- (A*x)/(B+x); sum((y-hy)^2) }
SQE <- Vectorize(sqe, c("A", "B"))
A.grid <- seq(0,40,l=100)
B.grid <- seq(0,20,l=100)
sqe.surf <- outer(A.grid, B.grid, SQE, da$y, da$x)
contour(A.grid, B.grid, sqe.surf, levels=(1:35)^2,
        xlab="A", ylab="B", col="gray70")
#start.list <- locator(n=4);
#start.list <- split(do.call(cbind, start.list), f=1:4)
#start.list <- lapply(start.list, function(x){names(x) <- c("A","B"); x})
start.list <- list(s1=c(A=0.1,B=0.1), s2=c(A=40,B=20),
                   s3=c(A=35,B=2.5), s4=c(A=18,B=18))
par(mfrow=c(2,2))
for(lis in 1:4){
  contour(A.grid, B.grid, sqe.surf, levels=(seq(1,35,2))^2,
        xlab="A", ylab="B", col="gray70")
  sink("trace.txt")
  n0 <- nls(y~A*x/(B+x), data=da, start=start.list[[lis]], trace=TRUE)
  sink()
  trace <- read.table("trace.txt")
  for(i in seq(nrow(trace)-1)){
    arrows(trace[i,"V3"], trace[i,"V4"],
           trace[i+1,"V3"], trace[i+1,"V4"],
           col=2, length=0.1)
    abline(v=trace[i+1,"V3"], h=trace[i+1,"V4"], col="orange", lty=3)
    Sys.sleep(1)
    print(c(i, trace[i+1,"V3"], trace[i+1,"V4"]))
  }
}
                                                                                          #
#-------------------------------------------------------------------------------------------
# olhando a convergência pelo gráficos dos observados vs preditos
for(lis in 1:4){
  sink("trace.txt")
  n0 <- nls(y~A*x/(B+x), data=da, start=start.list[[lis]], trace=TRUE)
  sink()
  plot(y~x, da)
  trace <- read.table("trace.txt")
  for(i in seq(nrow(trace))){
    curve(trace[i,"V3"]*x/(trace[i,"V4"]+x), add=TRUE, col=2)
    Sys.sleep(1)
  }
}
layout(1)
                                                                                          #
#-------------------------------------------------------------------------------------------
# curva ajustada
plot(y~x, da)
curve(coef(n0)["A"]*x/(coef(n0)["B"]+x), add=TRUE, col=2)
                                                                                          #
#-------------------------------------------------------------------------------------------
# estimativas dos parâmetros
summary(n0)
                                                                                          #
#-------------------------------------------------------------------------------------------
.
```

5.6 Ajuste de modelo não linear aos dados de DAP

```
.
#-------------------------------------------------------------------------------------------
# importando dados
#dap <- read.table(file.choose(), header=TRUE) # no windows vai abrir janela de procura
#dap <- read.table("http://www.leg.ufpr.br/~walmes/cursoR/cnpaf/dap.txt", header=TRUE)
```

```
dap <- read.table("../dados/dap.txt", header=TRUE)
names(dap) <- c("d","h")
str(dap)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# ordenando e tomando só os casos completos
dap <- dap[order(dap$d),]
dapcc <- dap[complete.cases(dap),]
str(dapcc)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# análise gráfica exploratória dos dados
plot(h~d, dapcc)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# análise gráfica do modelo candidato h = b0*(1-exp(b1*d))^b2
start <- list() # lista vazia irá armazenar os valores do último movimento de mouse
limits <- list(b0=c(25,35), b1=c(0,0.5), b2=c(0.7, 1.3)) # amplitude de variação
plottest <- function(...){
  plot(h~d, dapcc)
  curve(svalue(b0)*(1-exp(-svalue(b1)*x))^svalue(b2), add=TRUE, col=2)
  start <<- list(b0=svalue(b0), b1=svalue(b1), b2=svalue(b2))
}
source("func.R")
start
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# ajustar o modelo não linear (com os bons chutes do procedimento gráfico que só o R tem!)
n0 <- nls(h~b0*(1-exp(-b1*d))^b2, data=dapcc, start=start, trace=TRUE)
summary(n0)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# passando os chutes sem procedimento gráfico
n0 <- nls(h~b0*(1-exp(-b1*d))^b2, data=list(A=35, B=0.1, C=1.3), start=start, trace=TRUE)
summary(n0)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# ajustar o modelo não linear (com chutes sem noção, ver os tipos de mensagem de erro)
n1 <- nls(h~b0*(1-exp(-b1*d))^b2, data=dapcc,
          start=list(b0=35, b1=0, b2=1.3), trace=TRUE) # b1=0: exp(0*x) = 1 constante
n1 <- nls(h~b0*(1-exp(-b1*d))^b2, data=dapcc,
          start=list(b0=35, b1=-1, b2=1.3), trace=TRUE) # b1<0: problema na derivada numérica
n1 <- nls(h~b0*(1-exp(-b1*d))^b2, data=dapcc,
          start=list(b0=35, b1=0.1, b2=-1), trace=TRUE) # vai para região singular
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# verificação do ajuste
plot(h~d, dapcc)
lines(fitted(n0)~d, dapcc, col=2)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# não temos os gráficos de resíduos prontos para modelos não lineares, vamos contruí-los
# extraindo valores
r.cru <- residuals(n0)
var(r.cru)
r.pad <- residuals(n0, type="pearson")
var(r.pad)
fitd <- fitted(n0)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# fazendos os gráficos
par(mfrow=c(1,3))
plot(r.cru~fitd)
abline(h=0, lty=3)
scatter.smooth(sqrt(abs(r.pad))~fitd)
qqnorm(r.pad); qqline(r.pad, lty=2)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
.
```

5.7 Intervalo de confiança e teste de hipótese para modelos aninhados

```
.
#-----------------------------------------------------------------------------------------------
# intervalo de confiança para as estimativas
confint.default(n0)  # intervalo assintótico sempre é simétrico
```

```
confint(n0)          # observar os valores para entender o que o perfilhamento
                                                                                    #
#-------------------------------------------------------------------------------------
# o intervalo de confiança perfilhado para um parâmetro
prof <- profile(n0)
prof$b0
layout(1)
plot(prof$b0[,1]~prof$b0[,2][,1], type="l")
abline(h=c(-1.96,0,1.96), v=c(29.84, 31.93, 36.96), col=2)
                                                                                    #
#-------------------------------------------------------------------------------------
# o intervalo de confiânça de b2 contém o 1, será que preciso de b2?
n1 <- nls(h~b0*(1-exp(-b1*d)), data=dapcc, start=list(b0=30, b1=0.1))
summary(n1)
                                                                                    #
#-------------------------------------------------------------------------------------
# como ficou?
layout(1)
plot(h~d, dapcc)
lines(fitted(n0)~d, dapcc, col=2)
lines(fitted(n1)~d, dapcc, col=3)
                                                                                    #
#-------------------------------------------------------------------------------------
# teste da razão de verossimilhança para H0: b2=1
anova(n1, n0)
                                                                                    #
#-------------------------------------------------------------------------------------
# comparar o ajuste do modelo não linear com o linear escolhido na classe dos lineares
-2*c(logLik(n1))+2*length(coef(n1))
#help(AIC, help_type="html")
# AIC (k=2 parâmetros)
# -2*c(logLik(m5))+2*3
# 966.5362 (k=3 parâmetros)
-2*c(logLik(n1))+log(nrow(dapcc))*length(coef(n1))
# BIC (k=2 parâmetros)
-2*c(logLik(m5))+length(coef(m5))*log(nrow(dapcc))
                                                                                    #
#-------------------------------------------------------------------------------------
# R2 em modelos não lineares (danger!, o modelo trivial deve ser um modelo aninhado)
R2 <- function(nls.obj){
  da <- eval(nls.obj$data)
  resp.name <- all.vars(summary(nls.obj)$formula)[1]
  names(da)[which(names(da)==resp.name)] <- "y"
  sqn <- deviance(nls.obj)
  sqe <- deviance(lm(y~1, da))
  1-(sqn/sqe)
}
R2(n0)
R2(n1)
                                                                                    #
#-------------------------------------------------------------------------------------
.
```

5.8 Análise de resíduos para modelo de regressão não linear

```
.
#-------------------------------------------------------------------------------------
# extrai matrix de derivadas primeiras avaliada em theta, é como a matrix X dos lineares
F <- attr(n1$m$fitted(), "gradient")
F
                                                                                    #
#-------------------------------------------------------------------------------------
# com essa matriz ajustamos um lm para ter gráficos de resíduos, !!remover intercepto!!
m0 <- lm(h~-1+F, data=dapcc) # summary(), anova() não servem para nada
                                                                                    #
#-------------------------------------------------------------------------------------
# gráfico de análise dos resíduos
par(mfrow=c(2,2))
plot(m0)
mtext("Análise de resíduos para modelo de regressão não linear",
      outer=TRUE, line=-2, cex=1.4)
layout(1)
                                                                                    #
#-------------------------------------------------------------------------------------
.
```

5.9 Ajuste simultâneo de duas curvas (de crescimento)

```
.
#-------------------------------------------------------------------------------------
# curve de crescimento de aves para 2 sistemas de resfriamento de galpão
frango <- expand.grid(dia=2:42, sistema=factor(c("A","B")))
frango$peso <- c(  80.18145,   89.98167,  132.14629,  192.04534,  167.68245,  191.45191,
                  220.74227,  212.98519,  230.82651,  346.32728,  391.14474,  407.79706,
                  441.54167,  499.63470,  575.36996,  603.35279,  678.09090,  763.96071,
                  787.66652,  921.68731,  959.13005, 1069.59008, 1150.70054, 1269.26359,
                 1313.35194, 1419.24574, 1532.63279, 1647.94630, 1722.91144, 1832.84384,
                 1921.09935, 1960.50372, 2062.17519, 2204.45014, 2258.73203, 2311.79432,
                 2466.26338, 2505.48039, 2521.81638, 2625.00725, 2728.60234,  201.41506,
                  240.71230,  289.29251,  215.56332,  294.79948,  297.17629,  346.07243,
                  358.03428,  393.36050,  388.47739,  477.51108,  420.89742,  490.44854,
                  605.53948,  629.18954,  659.28526,  713.87248,  773.69469,  887.45404,
                  943.04904,  970.29292,  980.20056, 1142.43274, 1197.28398, 1187.79456,
                 1243.54212, 1340.48431, 1453.78205, 1542.45519, 1596.08595, 1702.33500,
                 1801.46693, 1847.62131, 1860.69871, 2018.38835, 2046.97753, 2077.06034,
                 2236.60287, 2238.75234, 2302.30264, 2354.35641)
                                                                                      #
#-------------------------------------------------------------------------------------
# análise gráfica exploratória
require(lattice)
xyplot(peso~dia, groups=sistema, data=frango, type=c("p","smooth"))
xyplot(peso~dia|sistema, data=frango, type=c("p","smooth"))
                                                                                      #
#-------------------------------------------------------------------------------------
# ajuste de curvas individuais com modelo logístico
nA <- nls(peso~A/(1+exp(-(dia-d50)/S)),
          data=subset(frango, sistema=="A"),
          start=list(A=3000, d50=25, S=10))
summary(nA)
                                                                                      #
#-------------------------------------------------------------------------------------
nB <- nls(peso~A/(1+exp(-(dia-d50)/S)),
          data=subset(frango, sistema=="B"),
          start=list(A=3000, d50=25, S=10))
summary(nB)
                                                                                      #
#-------------------------------------------------------------------------------------
# fazer o ajuste das duas curvas num único nls(), estimativa do QMR é mais consistente
nAB <- nls(peso~A[sistema]/(1+exp(-(dia-d50[sistema])/S[sistema])),
           data=frango,
           start=list(
             A=c(3200,3200),
             d50=c(28,30),
             S=c(8,10)))
summary(nAB)
                                                                                      #
#-------------------------------------------------------------------------------------
# fazer o gráfico dos valores ajustados/preditos
new <- expand.grid(dia=0:70, sistema=factor(c("A","B")))
new$fit <- predict(nAB, newdata=new)
                                                                                      #
#-------------------------------------------------------------------------------------
# gráfico
layout(1)
with(frango, plot(peso~dia, col=sistema, xlim=c(0,70), ylim=c(0,3200)))
with(subset(new, sistema=="A"), lines(dia, fit))
with(subset(new, sistema=="B"), lines(dia, fit, col=2))
                                                                                      #
#-------------------------------------------------------------------------------------
.
```

5.10 Teste de hipótese para modelos aninhados

```
.
#-------------------------------------------------------------------------------------
# hipóteses: H0A: A1=A2, H0B: d501=d502, H0C: S1=S2
confint.default(nAB) # baseado em normalidade assintótica
confint(nAB)         # baseado em perfil de verossimilhança (modelo logístico bem "linear")
                                                                                      #
#-------------------------------------------------------------------------------------
# testar H0A: ajustar um modelo em que A seja igual para os dois sistemas
```

```
nAB2 <- nls(peso~A/(1+exp(-(dia-d50[sistema])/S[sistema])),
            data=frango,
            start=list(
              A=c(3200),
              d50=c(28,30),
              S=c(8,10)))
summary(nAB2)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# empregar o teste da razão de verossimilhança para testar H0A: A1=A2
anova(nAB2, nAB)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# fazer o gráfico dos valores ajustados/preditos
new <- expand.grid(dia=0:70, sistema=factor(c("A","B")))
new$fit <- predict(nAB2, newdata=new)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# gráfico
layout(1)
with(frango, plot(peso~dia, col=sistema, xlim=c(0,70), ylim=c(0,3200)))
with(subset(new, sistema=="A"), lines(dia, fit))
with(subset(new, sistema=="B"), lines(dia, fit, col=2))
                                                                                                    #
#-----------------------------------------------------------------------------------------------
.
```

5.11 Intervalo de confiança e teste de hipótese para diferença de duas curvas em um ponto

```
.
#-----------------------------------------------------------------------------------------------
# estimar a diferença de peso aos 42 entre os sistemas ("teste de médias")
coef(nAB2) # estimativas
vcov(nAB2) # matriz de covariância das estimativas
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# "converter" o modelo para linear, permitido porque o inferência em não linear é linear
F <- attr(nAB2$m$fitted(), "gradient")
m0 <- lm(peso~-1+F, data=frango)
coef(m0) # não faz muito sentido, mas é resultado de uma rotação
round(vcov(nAB2),3) # modelo não linear original
round(vcov(m0),3)   # sua versão linear (mesma matriz)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# precisamos montar a matriz do contraste, ou seja, a F* para os valores x*
model <- deriv3(~i*A/(1+exp((d501-dia)/S1))+(1-i)*A/(1+exp((d502-dia)/S2)),
                c("A","d501","d502","S1","S2","i"),
                function(A, d501, d502, S1, S2, i, dia){ NULL })
lA <- as.list(c(coef(nAB2), i=1, dia=42)) # sistema A aos 42 dias
lB <- as.list(c(coef(nAB2), i=0, dia=42)) # sistema B aos 42 dias
FA <- do.call(model, lA)     # predito, F e H para sistema A aos 42 dias
FB <- do.call(model, lB)     # predito, F e H para sistema B aos 42 dias
c(pesoA=FA[1], pesoB=FB[1])  # o peso dos animais aos 42 dias
CC <- attr(FA, "gradient")[1:5]-attr(FB, "gradient")[1:5] # matriz do contraste
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# estimativa do contraste
CC%*%coef(m0)
diff(predict(nAB2, newdata=data.frame(sistema=c("B","A"), dia=42)))
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# usando a gmodels::estimable()
require(gmodels)
estimable(m0, rbind(H0D=CC), conf.int=0.95)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# e o R²? é para o modelo todo, e não um para cada curva
R2(nAB2)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
.
```

5.12 Ajuste de modelos não lineares com a library{nlme}

```
.
#-------------------------------------------------------------------------------------------
# dados de número de nematóides ecloditos em função dos dias e dose de nematícida
nema <- expand.grid(dia=seq(2,16,2), dose=gl(4,1,labels=c(0,1,5,10)))
nema$eclod <- c(13, 56.5, 97.5, 168, 246.5, 323, 374, 389, 7, 26, 64.5, 126, 207.5,
                282, 334, 343, 5, 21.5, 45.5, 79, 118.5, 146, 167.5, 174.5, 3.25,
                9.25, 12.5, 20.5, 32.25, 39.25, 40.25, 42.25)
str(nema)
xyplot(eclod~dia, groups=dose, data=nema, type="b", auto.key=TRUE)
                                                                                          #
#-------------------------------------------------------------------------------------------
# carrega o pacote nlme (do grupo dos recomendados)
require(nlme)
                                                                                          #
#-------------------------------------------------------------------------------------------
# fazendo ajustes simultâneos, a SSlogis é uma selfStart
nema <- groupedData(eclod~dia|dose, data=nema) # expressa a estrutura do dado
nema$dose <- factor(nema$dose, ordered=FALSE)
n0 <- nlsList(eclod~SSlogis(dia, Asym, xmid, scal), data=nema)
summary(n0)
plot(augPred(n0)) # faz o gráfico com preditos e observados
coef(n0)
                                                                                          #
#-------------------------------------------------------------------------------------------
# na nlsList não tem como restrigir a estimação, então usamos a gnls
# podemos passar o objeto nlsList ou montar sem atalhos
gn0 <- gnls(eclod~SSlogis(dia, Asym, xmid, scal),
            data=nema,
            params=Asym+xmid+scal~dose,
            start=c(44,130,320,380, 7.3,1,1,1, 2.3,0,0,0))
anova(gn0, type="marginal")  # é um teste de Wald, testa se o 3 desvios do intercepto=0
                                                                                          #
#-------------------------------------------------------------------------------------------
# novos valores de dia para a predição de eclod
new <- expand.grid(dia=seq(0,20,0.2), dose=levels(nema$dose))
new$eclod <- predict(gn0, newdata=new)
xyplot(eclod~dia, groups=dose, data=new, type="l")
                                                                                          #
#-------------------------------------------------------------------------------------------
# incluir todos os resultados em um único gráfico
tudo <- rbind(nema, new)
tudo$tipo <- rep(c("obs","fit"), c(nrow(nema),nrow(new)))
xyplot(eclod~dia|factor(dose), groups=tipo, data=tudo,
       distribute.type=TRUE, type=c("l","p","g"),
       main="O revisor que pede gráficos no Excel deve ir preso!",
       sub="Os gráficos do R são científicos!",
       xlab="Período após a aplicação dos nematicídas (dias)",
       ylab="Nematóides eclodidos",
       key=list(x=0.8, y=0.9,
         lines=list(lty=c(NULL,1), col=c("#0080ff","#ff00ff")),
         text=list(c("ajustado","observado"))),
       layout=c(4,1)#, scales=list(y="free")
       )
                                                                                          #
#-------------------------------------------------------------------------------------------
.
```

5.13 Teste de hipótese para modelos aninhados (2)

```
.
#-------------------------------------------------------------------------------------------
# testar a hipótese H0: scal_i=scal para todo i, ou seja, scal comum
gn1 <- gnls(eclod~SSlogis(dia, Asym, xmid, scal),
            data=nema,
            params=list(Asym+xmid~dose, scal~1),
            start=c(44,130,320,380, 7.3,1,1,1, 2.3))
anova(gn1, gn0) # testa a H0 por LRT, note p-valor diferente do anova(..., "marginal")
                                                                                          #
#-------------------------------------------------------------------------------------------
.
```

5.14 Ajuste do modelo duplo van Genuchten (7 parâmetros)

```
.
#-------------------------------------------------------------------------------------------
# dados para a Curva de Retenção de Água, umidade (theta) e tensão (psi)
cra <- data.frame(psi=c(0.01,1,2,4,6,8,10,33,60,100,500,1500,2787,4727,6840,7863,
                    9030,10000,10833,13070,17360,21960,26780,44860,69873,74623,
                    87287,104757,113817,147567,162723,245310,262217,298223),
                  theta=c(0.779,0.554,0.468,0.406,0.373,0.36,0.344,0.309,0.298,
                    0.292,0.254,0.241,0.236,0.233,0.223,0.202,0.172,0.187,0.138,
                    0.098,0.07,0.058,0.052,0.036,0.029,0.0213,0.0178,0.0174,
                    0.0169,0.0137,0.0126,0.0109,0.0106,0.0053))
                                                                                          #
#-------------------------------------------------------------------------------------------
# gráfico dos valores observados
plot(theta~log10(psi), data=cra)
                                                                                          #
#-------------------------------------------------------------------------------------------
# função do modelo duplo van Genuchten, 7 parâmetros, cria a função do modelo f(x,theta)
dvg <- function(x, ts, ti, tr, ae, at, ne, nt){
  tr+(ti-tr)/((1+(at*10^x)^nt)^(1-1/nt))+(ts-ti)/((1+(ae*10^x)^ne)^(1-1/ne))
}
dvg(log10(c(1,10,100,1000,10000)),         # log 10 das tensões
    0.7, 0.2, 0.05, 1.3, 0.0001, 1.5, 3.5) # valor dos parâmetros
                                                                                          #
#-------------------------------------------------------------------------------------------
.
```

5.15 Procedimentos gráficos: `rpanel`, `manipulate` e `gWidgetsRGtk2`

```
.
#-------------------------------------------------------------------------------------------
# obter os chutes dos 7 parâmetros via procedimento gráfico
require(rpanel)  # pacote com funções para manipulação gráfica
start <- list()  # cria uma lista vazia para receber os valores finais
dvg.panel <- function(panel){
  plot(panel$theta~log10(panel$psi))
  curve(dvg(x, ts=panel$ts, ti=panel$ti, tr=panel$tr,
            ae=panel$ae, at=panel$at, ne=panel$ne, nt=panel$nt), add=TRUE)
  start <<- list(ts=panel$ts, ti=panel$ti, tr=panel$tr,
                 ae=panel$ae, at=panel$at, ne=panel$ne, nt=panel$nt)
  panel
}
panel <- rp.control(theta=cra$theta, psi=cra$psi)
rp.slider(panel, ts, 0.7, 0.9, initval=0.8, showvalue=TRUE, action=dvg.panel)
rp.slider(panel, tr, 0, 0.1, initval=0.05, showvalue=TRUE, action=dvg.panel)
rp.slider(panel, ti, 0.1, 0.5, initval=0.2, showvalue=TRUE, action=dvg.panel)
rp.slider(panel, ae, 1, 3, initval=1.5, showvalue=TRUE, action=dvg.panel)
rp.slider(panel, at, 0, 0.0001, initval=0.00005, showvalue=TRUE, action=dvg.panel)
rp.slider(panel, ne, 1, 4, initval=1.5, showvalue=TRUE, action=dvg.panel)
rp.slider(panel, nt, 2, 7, initval=4.5, showvalue=TRUE, action=dvg.panel)
start
                                                                                          #
#-------------------------------------------------------------------------------------------
# usando a manipulate, que só está disponível para o RStudio
require(manipulate)
start <- list()  # cria uma lista vazia para receber os valores finais
manipulate({
             plot(theta~log10(psi), data=cra)
             curve(dvg(x, ts=ts, ti=ti, tr=tr, ae=ae, at=at, ne=ne, nt=nt), add=TRUE)
             start <<- list(ts=ts, ti=ti, tr=tr, ae=ae, at=at, ne=ne, nt=nt)
           },
           ts=slider(0.7, 0.9, initial=0.8),
           ti=slider(0.15, 0.25, initial=0.2),
           tr=slider(0, 0.10, initial=0.05),
           ae=slider(1.01, 3, initial=1.3),
           at=slider(0, 0.0001, initial=0.00005),
           ne=slider(1.01, 3, initial=1.65),
           nt=slider(1.8, 5, initial=4.3)
           )
start
                                                                                          #
#-------------------------------------------------------------------------------------------
# usando a gWidgetsRGtk2
library(gWidgetsRGtk2)
```

```
options("guiToolkit"="RGtk2")
limits <- list(ts=c(0.7,0.9), tr=c(0,0.1), ti=c(0.1,0.5),
               ae=c(1,3), at=c(0,0.0001), ne=c(1,4), nt=c(2,7))
plottest <- function(...){
  plot(theta~log10(psi), data=cra)
  ts <- svalue(ts); tr <- svalue(tr); ti <- svalue(ti)
  ae <- svalue(ae); at <- svalue(at); ne <- svalue(ne); nt <- svalue(nt)
  curve(dvg(x, ts=ts, ti=ti, tr=tr, ae=ae, at=at, ne=ne, nt=nt), add=TRUE)
  start <<- list(ts=ts, ti=ti, tr=tr, ae=ae, at=at, ne=ne, nt=nt)
}
w <- gwindow("Slider and spinbox example")
tbl <- glayout(cont=w)
for(i in 1:length(limits)){
  tbl[i,1] <- paste("Slide to adjuste parameter", names(limits)[i])
  tbl[i,2, expand=TRUE] <- (assign(names(limits)[i],
            gslider(from=limits[[i]][1], to=limits[[i]][2],
                    by=diff(limits[[i]])/20, value=mean(limits[[i]]),
                    container=tbl, handler=plottest)))
}
plottest()
start
                                                                                                    #
#-------------------------------------------------------------------------------------------
.
```

5.16 Procedimento direto para fazer gráfico com valores preditos

```
.
#-------------------------------------------------------------------------------------------
# start <- list(ts=0.772, ti=0.225, tr=0.011, ae=2.5861, at=0.0000788, ne=1.4637, nt=2.786)
start # valores salvos do último movimento
                                                                                                    #
#-------------------------------------------------------------------------------------------
# ajuste do modelo os dados usando os chutes do procedimento gráfico (muito fácil)
n0 <- nls(theta~tr+(ti-tr)/((1+(at*psi)^nt)^(1-1/nt))+(ts-ti)/((1+(ae*psi)^ne)^(1-1/ne)),
          data=cra, start=start)
summary(n0) # quadro de estimativas
confint(n0) # intervalos de confiança perfilhados
                                                                                                    #
#-------------------------------------------------------------------------------------------
# faz a diagnose dos resíduos
qqnorm(residuals(n0))               # gráfico para normalidade
plot(residuals(n0)~log10(cra$psi))  # gráfico para falta de ajuste
plot(abs(residuals(n0))~fitted(n0)) # gráfico para homogeneidade de variância
                                                                                                    #
#-------------------------------------------------------------------------------------------
# gráfico dos dados com a curva estimada
lis <- c(list(x=NULL), as.list(coef(n0)), body(dvg))
plot(theta~log10(psi), data=cra,                                      # faz o gráfico
     ylab=expression("Conteúdo de água no solo"~(theta~","~g~g^{-1})), # rótulo y
     xlab=expression("Tensão matricial"~(Psi~","~"kPa")),               # rótulo x
     xaxt="n")
tmp <- as.function(lis)
curve(tmp, add=TRUE, col=2, lwd=1.5)                                  # adiciona a curva
axis(1, at=-2:5, label=as.character(10^(-2:5)), lwd.ticks=2)          # escala log
s <- log10(sort(sapply(1:9, function(x) x*10^(-3:6))))
axis(1, at=s, label=NA, lwd=0.5)                                      # traços secundários
abline(v=-2:6, h=seq(0,1,0.05), lty=3, col="gray50")                  # grade de referência
                                                                                                    #
#-------------------------------------------------------------------------------------------
.
```

5.17 Bandas de confiança para modelo de regressão não linear

```
.
#-------------------------------------------------------------------------------------------
# dados de postássio liberado em função do tempo
klib <- data.frame(k=c(51.03, 57.76, 26.60, 60.65, 87.07, 64.67,
                       91.28, 105.22, 72.74, 81.88, 97.62, 90.14,
                       89.88, 113.22, 90.91, 115.39, 112.63, 87.51,
                       104.69, 120.58, 114.32, 130.07, 117.65, 111.69,
                       128.54, 126.88, 127.00, 134.17, 149.66, 118.25,
```

```
                    132.67, 154.48, 129.11, 151.83, 147.66, 127.30),
                  t=rep(c(15, 30, 45, 60, 75, 90,
                    120, 150, 180, 210, 240, 270), each=3))
                                                                                          #
#-------------------------------------------------------------------------------------------
# criando função que representa o modelo exponencial reparametrizado, retorna f(x), F e H
expo.der <- deriv3(~A*(1-exp(-log(2)*t/V))+D*t,
                  c("A", "V", "D"),
                  function(t, A, V, D) NULL)
                                                                                          #
#-------------------------------------------------------------------------------------------
# diagnose gráfica e primeiro chute
plot(k~t, data=klib, xlab="Período de incubacão (dias)",
     ylab="Potássio liberado acumulado (mg/kg de solo)")
A <- 90; V <- 20; D <- 0.2
curve(expo.der(x, A, V, D), add=TRUE, col=2)
start <- list(A=A, V=V, D=D)
                                                                                          #
#-------------------------------------------------------------------------------------------
# ajustando o modelo aos dados a partir dos valores iniciais via gráfico
n0 <- nls(k~expo.der(t, A, V, D), data=klib, start=start)
summary(n0)
confint(n0)
                                                                                          #
#-------------------------------------------------------------------------------------------
# valores preditos, gradiente e hessiano avaliado nos valores estimados
str(n0)
str(n0$m$fitted())
c(n0$m$fitted())
attr(n0$m$fitted(), "gradient")
attr(n0$m$fitted(), "hessian")
                                                                                          #
#-------------------------------------------------------------------------------------------
# obtenção dos valores preditos
pred <- data.frame(t=seq(0,300,l=100))
der <- do.call(expo.der, args=c(list(t=pred$t), as.list(coef(n0))))
F <- attr(der, "gradient") # gradiente avaliado no novo grid mais fino de t
U <- chol(vcov(n0))
se <- sqrt(apply(F%*%t(U), 1, function(x) sum(x^2)))  # erro padrão
ttabelado <- qt(c(.5, .025,.975), df=df.residual(n0)) # valor t para IC de 95%
#-------------------------------------------------------------------------------------------
# gráficos dos observados, preditos com IC, legenda e equações
plot(k~t, data=klib, xlab="Período de incubacão (dias)",
     ylab="Potássio liberado acumulado (mg/kg de solo)",
     xlim=c(0,300), ylim=c(0,160))
matlines(pred$t, c(der)+outer(se, ttabelado),
         type="l", col=c(1,2,2), lty=c(1,2,2))
legend("bottomright",
       legend=c("valores observados", "valores preditos",
         "intervalo de confiança (95%)"),
       lty=c(NA,1,2), col=c(1,1,2), pch=c(1,NA,NA), bty="n")
cf <- format(coef(n0), digits=3)
text(par("usr")[1], par("usr")[4], adj=c(-0.05,1.5),
     label=substitute(hat(k)[total]==a%.%(1-e^{-ln(2)%.%t/v})+d%.%t,
       list(a=cf[1], v=cf[2], d=cf[3])))
abline(v=coef(n0)["V"], h=coef(n0)["A"], col="gray70")
curve(expo.der(x, coef(n0)["A"], coef(n0)["V"], 0), add=TRUE, col=3)
curve(expo.der(x, 0, 0, coef(n0)["D"]), add=TRUE, col=3)
text(225, coef(n0)["A"], pos=3,
     label=substitute(hat(k)["fácil"]==a%.%(1-e^{-ln(2)%.%t/v}),
       list(a=cf[1], v=cf[2])))
text(173, 38, pos=3, srt=18,
     label=substitute(hat(k)["difícil"]==d%.%t, list(d=cf[3])))
                                                                                          #
#-------------------------------------------------------------------------------------------
.
```

5.18 Medidas de curvatura do modelo e vício nas estimativas

```
.
#-------------------------------------------------------------------------------------------
# comparar 3 modelos: quociente, exponencial e exponencial reparametrizado (acima)
# precisamos do F e H de todos eles
expo.der <- deriv3(~A*(1-exp(-log(2)*t/V))+D*t, c("A", "V", "D"),
                  function(t, A, V, D) NULL)
exp.der <- deriv3(~A*(1-exp(-B*t))+D*t, c("A", "B", "D"),
```

```
                  function(t, A, B, D) NULL)
quo.der <- deriv3(~A*t/(V+t)+D*t, c("A", "V", "D"),
                  function(t, A, V, D) NULL)
                                                                                          #
#-------------------------------------------------------------------------------------------
# ajustar os 3 modelos aos dados de liberação de potássio
expo <- nls(k~expo.der(t, A, V, D), data=klib, start=list(A=90, V=20, D=0.2))
exp0 <- nls(k~exp.der(t, A, B, D), data=klib, start=list(A=A, B=0.05, D=D))
quo0 <- nls(k~quo.der(t, A, V, D), data=klib, start=list(A=90, V=20, D=0.2))
                                                                                          #
#-------------------------------------------------------------------------------------------
# valores preditos por cada modelo
pred <- data.frame(t=seq(0,300,l=100))
expop <- do.call(expo.der, args=c(list(t=pred$t), as.list(coef(expo))))
exp0p <- do.call(exp.der, args=c(list(t=pred$t), as.list(coef(exp0))))
quo0p <- do.call(quo.der, args=c(list(t=pred$t), as.list(coef(quo0))))
plot(k~t, data=klib, xlim=c(0,300), ylim=c(0,160))
matlines(pred$t, cbind(c(expop),c(exp0p),c(quo0p)),
         type="l", col=c(1,2,3), lty=c(1,2,3))
                                                                                          #
#-------------------------------------------------------------------------------------------
# curvatura média de Bates & Watts, valor de corte é 0.3*sqrt(qf(19/20, p, n-p))
require(MASS)
rms.curv(expo)
rms.curv(exp0) # mesma curvatura intrínseca, maior curvatura devido à parametrização
rms.curv(quo0) # menor curvatura intrínseca, intermediária devido à parametrização
0.3*sqrt(qf(19/20, length(coef(expo)), df.residual(expo)))
                                                                                          #
#-------------------------------------------------------------------------------------------
# vício de Box (função que eu desenvolvi na minha Dissertação de Mestrado)
browseURL(URLencode("http://www.leg.ufpr.br/%7Ewalmes/docs/Walmes%20Marques%20Zeviani%20-%20Dissertacao.pdf"))
biasbox <- function(nls.obj){
  theta <- summary(nls.obj)$coef[,1]
  sd.theta <- summary(nls.obj)$coef[,2]
  F <- attr(nls.obj$m$fitted(), "gradient")
  H <- attr(nls.obj$m$fitted(), "hessian")
  sig <- summary(nls.obj)$sigma
  n <- dim(F)[1]
  FlFi <- t(F)%*%F
  d <- -(sig^2/2)*sapply(1:n, function(x){
    sum(diag(solve(FlFi)%*%H[x, , ]))})
  bias <- as.vector(solve(FlFi)%*%t(F)%*%d)
  names(bias) <- names(coef(nls.obj))
  bias.sd <- 100*bias/sd.theta
  bias.th <- 100*bias/theta
  return(list("viés bruto"=bias,
              "%viés(theta)"=bias.th,
              "%viés(sd(theta))"=bias.sd))
}
                                                                                          #
#-------------------------------------------------------------------------------------------
# calcula o vício de box para as estimativas dos parâmetros
biasbox(expo)
biasbox(exp0)
biasbox(quo0)
                                                                                          #
#-------------------------------------------------------------------------------------------
.
```

5.19 Ajuste de modelo não linear com a função `optim()`

```
.
#-------------------------------------------------------------------------------------------
# é apenas uma outra forma de ajustar modelo, vantagem: obter contornos de verossimilhança
fo <- function(th, x, y){
  ## th: vetor numérico de parâmetros do modelo
  ## x: vetor da covariável
  ## y: vetor da resposta
  mu <- th[1]*(1-exp(-log(2)*x/th[2]))+th[3]*x
  sqr <- sum((y-mu)^2)
  s2 <- sqr/(length(y))
  ll <- sum(dnorm(y, mu, sqrt(s2), log=TRUE))
  ## retorna a log-verossimilhança
  return(ll)
}
                                                                                          #
```

```
#-------------------------------------------------------------------------------------------
# avaliando a ll e a sqr com a função fobj()
fo(coef(expo), x=klib$t, y=klib$k); logLik(expo)
                                                                                          #
#-------------------------------------------------------------------------------------------
# passando função para a optim() otimizar
op <- optim(c(90, 20, 0.2), fo,
            x=klib$t, y=klib$k,
            method="BFGS", hessian=TRUE, control=list(fnscale=-1))
op$par        # estimativas
coef(expo)
logLik(expo)  # log-verossmilhança
op$value
solve(-op$hessian) # matriz de covariância das estimativas (denominador n)
vcov(expo)
                                                                                          #
#-------------------------------------------------------------------------------------------
.
```

5.20 Concetrando a verossmilhança para modelos parcialmente lineares

```
.
#-------------------------------------------------------------------------------------------
# o modelo A*(1-exp(-log(2)*x/V))+D*x é não linear em V apenas, se V for conhecido o
# modelo assume forma linear A*z+D*x, em que z = (1-exp(-log(2)*x/V))
fo.conc <- function(th, x, y){
  ## th: vetor numérico do parâmetro V
  ## x: vetor da covariável
  ## y: vetor da resposta
  y <- matrix(y)
  z <- (1-exp(-log(2)*x/th))
  X <- model.matrix(~-1+z+x)
  ## calcula A e D via mínimos quadrados
  bt <- solve(crossprod(X), crossprod(X,y))
  mu <- X%*%bt
  s2 <- crossprod(y-mu)/nrow(y)
  ll <- sum(dnorm(c(y), c(mu), sqrt(s2), log=TRUE))
  ## retorna a verossimilhança
  attr(ll, "theta") <- c(A=c(bt[1,]), V=th, D=c(bt[2,]))
  return(ll)
}
fo.conc(coef(expo)[2], x=klib$t, y=klib$k)
logLik(expo); coef(expo)
                                                                                          #
#-------------------------------------------------------------------------------------------
# estimação
op <- optim(c(20), fo.conc,
            x=klib$t, y=klib$k,
            method="BFGS", hessian=TRUE, control=list(fnscale=-1))
op$par
                                                                                          #
#-------------------------------------------------------------------------------------------
.
```

5.21 Usando recursos da `libray(nls2)`

```
.
#-------------------------------------------------------------------------------------------
# fazendo o ajuste por busca em um universo discreto de valores (aproximar chute inicial)
# com os procedimentos gráficos a nls2() perde importância em aproximar o chute inicial
require(nls2)
coef(expo)
start.grid <- expand.grid(A=seq(40,120,length=10),
                          V=seq(5,25,length=10),
                          D=seq(0,0.5,length=10)) # cria um grid de 10*10*10=1000 valores
formula(expo)
n0.aprox <- nls2(formula(expo), data=klib, algorithm="grid-search",
                 start=start.grid) # vai procurar o melhor dentro do grid
summary(n0.aprox)
coef(n0.aprox) # são valores iniciais que posso passar numa busca contínua com a nls()
                                                                                          #
#-------------------------------------------------------------------------------------------
```

```
# usando os valores ótimos retornados pelo "grid-search" na nls()
n0 <- nls(formula(expo), data=klib, start=coef(n0.aprox))
summary(n0) # precisou de poucas interações
cbind(coef(n0), coef(n0.aprox))
                                                                                                    #
#-------------------------------------------------------------------------------------------
# qual a real vantagem? passar as estimativas de vero concentrada para nls2() encontrar
# os erros padrões, valor t, pvalor, IC...
coef.op <- fo.conc(op$par, x=klib$t, y=klib$k)
coef.op <- attr(coef.op, "theta")
n0.op <- nls2(formula(expo), data=klib, start=coef.op, algorithm="grid-search")
summary(n0.op)
confint.default(n0.op)
summary(n0)
                                                                                                    #
#-------------------------------------------------------------------------------------------
.
```

5.22 Usando recursos da `libray(nlstools)`

```
#-------------------------------------------------------------------------------------------
# funções para análise de resíduos, gráficos para região de confiança e contornos de SQR
require(nlstools)
help(nlstools, help_type="html")
                                                                                                    #
#-------------------------------------------------------------------------------------------
# modelo ajustado de classe nls
class(expo)
summary(expo)
overview(expo) # summary com informação extra
plotfit(expo)  # observado vs ajustado
points(klib$k, klib$k, pch=19, cex=0.3)
points(klib$k, fitted(expo), pch=4, cex=0.3)
                                                                                                    #
#-------------------------------------------------------------------------------------------
# gráficos resíduos~ajustados, autocorrelação, qqplot...
par(mfrow=c(2,2))
plot(nlsResiduals(expo))
layout(1)
                                                                                                    #
#-------------------------------------------------------------------------------------------
# os 6 gráficos
par(mfrow=c(3,2))
for(i in 1:6) plot(nlsResiduals(expo), which=i)
layout(1)
                                                                                                    #
#-------------------------------------------------------------------------------------------
# contornos da soma de quadrado dos resíduos
plot(nlsContourRSS(expo)) # demora um pouco, observar se os formatos são ellipticos
plot(nlsContourRSS(expo), nlev=6, col=FALSE)
                                                                                                    #
#-------------------------------------------------------------------------------------------
# IC baseado em proposição e aceitação dos valores internos à SQR (Beale1960)
plot(nlsConfRegions(expo), bounds=TRUE)
                                                                                                    #
#-------------------------------------------------------------------------------------------
.
```

5.23 Obtendo intervalo e região de confiança via perfil de verossmilhança

```
#-------------------------------------------------------------------------------------------
# gráficos dos intervalos de confiânça baseados em perfil de verossimilhança
require(MASS)
par(mfrow=c(3,3))
plot(profile(expo))
plot(profile(exp0))
plot(profile(quo0))
layout(1)
                                                                                                    #
#-------------------------------------------------------------------------------------------
# contornos de verossimilhança
```

```
layout(1)
plot(nlsContourRSS(expo, lseq=50), nlev=6, col=FALSE)
x11()
plot(nlsContourRSS(exp0, lseq=50), nlev=6, col=FALSE)
x11()
plot(nlsContourRSS(quo0, lseq=50), nlev=6, col=FALSE)
# note: A e D são lineares -> contornos elípticos, os demais têm desvios de elipcidade
                                                                                                    #
#-------------------------------------------------------------------------------------------------
.
```

5.24 Ajustando modelo parcialmente linear com a `nls()`

```
.
#-------------------------------------------------------------------------------------------------
# ajustando modelo parcialmente linear (procedimento iterativo apenas em V)
n0 <- nls(k~cbind(A=1-exp(-log(2)*t/V), D=t), data=klib,
          algorithm="plinear", start=c(V=20))
summary(n0)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# resumo:
# usando a nls() direto com chutes visuais
# usando a optim() no modelo completo
# usando a optim() no modelo concentrado/parcialmente linear
# usando a nls() com chutes da nls2()
# usando a nls2() com os "chutes" da optim()
# usando a nls() com um modelo concentrado/parcialmente linear
#-------------------------------------------------------------------------------------------------
.
```

5.25 Secagem do solo em micro ondas

```
.
#-------------------------------------------------------------------------------------------------
# dados de umidade em função do tempo (não é medida repetida)
sec <- read.table("../dados/secagem.txt", header=TRUE, sep="\t")
str(sec)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# exploração gráfica
require(lattice)
xyplot(umid~tempo|nome, data=sec, type=c("p","smooth"))
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# os solos foram secos em estufa e adicionados 40% de umidade, será que o micro remove ela?
xyplot(umrel~tempo|nome, data=sec, type=c("p","smooth","g"))
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# fazer um intervalo de confiança para a umidade em 40 minutos, ajustar modelo logístico
require(nlme) # permite um ajuste conjunto com mais facilidade
sec <- groupedData(umrel~tempo|nome, data=sec)
n0 <- nlsList(umrel~SSlogis(tempo, A, x0, S), data=sec)
summary(n0)
coef(n0)
qqnorm(residuals(n0)); qqline(residuals(n0))
plot(residuals(n0)~fitted(n0))
plot(augPred(n0))
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# fazendo a predição, controlando intervalo, etc
pred <- expand.grid(tempo=0:60, nome=levels(sec$nome))
pred$umrel <- predict(n0, newdata=pred)
pred$tipo <- "predito"
sec$tipo <- "observado"
pred <- rbind(pred, sec[,c("nome","tempo","umrel","tipo")])
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# o gráfico
xyplot(umrel~tempo|nome, groups=tipo, data=pred,
       distribute.type=TRUE, type=c("p","l"))
xyplot(umrel~tempo, groups=nome, data=subset(pred, tipo=="predito"), type="l")
```

```
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# H0: 40 minutos é suficiente para secar o solo
lvadbw <- nls(umrel~SSlogis(tempo, A, x0, S), data=subset(sec, nome=="LVAd-Bw"))
summary(lvadbw)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# obter a banda de confiança para a curva
F <- attr(lvadbw$m$fitted(), "gradient")
tc <- qt(0.975, df=df.residual(lvadbw))
vc <- vcov(lvadbw)
se <- diag(sqrt(F%*%vc%*%t(F)))
pred2 <- data.frame(fit=fitted(lvadbw))
pred2$lwr <- pred2$fit-tc*se
pred2$upr <- pred2$fit+tc*se
pred2$tempo <- sec$tempo[sec$nome=="LVAd-Bw"]
pred2$umrel <- sec$umrel[sec$nome=="LVAd-Bw"]
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# gráfico
with(pred2, matplot(tempo, cbind(fit, lwr, upr), type="l", col=c(1,2,2), lty=c(1,2,2)))
with(pred2, points(tempo, umrel))
abline(v=seq(0,45,2.5), h=seq(0,1.1,0.05), col="gray90", lty=2)
abline(h=1, v=40)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# será que os alguns solos possuem o mesmo padrão de secamento?
                                                                                                    #
#-----------------------------------------------------------------------------------------------
.
```

5.26 Modelando a variância

```
.
#-----------------------------------------------------------------------------------------------
# a variância é função de x, f(x)?
par(mfrow=c(1,2))
scatter.smooth(abs(residuals(n0))~fitted(n0), col=2) # decaimento
scatter.smooth(abs(residuals(n0))~sec$tempo, col=2)  # decaimento
layout(1)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# ajustando modelo não linear modelando a variância
help(varClasses, help_type="html") # modelos para a variância
help(gnls, help_type="html")       # função que possui argumento para ajuste da variância
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# ajustando modelando a variância com estrutura varExp()
coef(n0)
sec$nome <- factor(sec$nome, ordered=FALSE) # por padrão é ordenado
n1 <- gnls(umrel~SSlogis(tempo, A, x0, S), data=sec,
           params=A+x0+S~nome, start=list(1,0,0,0, 14,0,0,0, 7,0,0,0))
print(n1, corr=FALSE)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# ajustando varExp()
n2 <- update(n1, weights=varExp(form=~tempo))
summary(n2)
print(n2, corr=FALSE)
anova(n2, n1) # testa a H0: expon=0
logLik(n2)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# ajustando varPower()
n3 <- update(n1, weights=varPower(form=~tempo))
print(n3, corr=FALSE)
anova(n3, n1) # testa a H0: power=0
logLik(n3)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# visualização gráfica das funções de variância
aux <- tapply(residuals(n0), sec$tempo, var)
plot(aux~unique(sec$tempo))
curve(0.1164^2*exp(2*(-0.03753)*x), add=TRUE, col=2)
curve(0.2245^2*x^(-2*0.4998), add=TRUE, col=3)
                                                                                                    #
```

```
#-------------------------------------------------------------------------------------
# impacto nas estimativas e erros padrões dos parâmetros
summary(n1)$tTable
summary(n2)$tTable
                                                                                     #
#-------------------------------------------------------------------------------------
# portanto tem algum impacto nos valores preditos e seus erros padrões
pred <- expand.grid(tempo=0:60, nome=levels(sec$nome))
pred$umrel1 <- predict(n1, newdata=pred)
pred$umrel2 <- predict(n2, newdata=pred)
                                                                                     #
#-------------------------------------------------------------------------------------
# gráfico dos valores preditos
require(latticeExtra)
xyplot(umrel~tempo|nome, data=sec)+
  as.layer(xyplot(umrel1~tempo|nome, data=pred, type="l", col=1))+
  as.layer(xyplot(umrel2~tempo|nome, data=pred, type="l", col=2))
                                                                                     #
#-------------------------------------------------------------------------------------
.
```

5.27 Modelando a variância (2)

```
.
#-------------------------------------------------------------------------------------
# dados de crescimento de goiaba
goi <- read.table("../dados/goiaba.txt", header=TRUE)
str(goi)
                                                                                     #
#-------------------------------------------------------------------------------------
# análise exploratória
require(lattice)
splom(goi[c("long","trans","peso","volume")], groups=goi$daa) # falsa noção de correlação
xyplot(long~trans|daa, goi)                                   # correlação
xyplot(long~trans|daa, goi, scales="free", type=c("p","r"))
                                                                                     #
#-------------------------------------------------------------------------------------
# mais análise exploratória
require(reshape)
aux <- melt(goi, id.vars="daa", measure.vars=c("long","trans","peso","volume"),
            variable="medidas")
str(aux)
xyplot(value~daa|medidas, data=aux, type=c("p","a"), scales="free")
                                                                                     #
#-------------------------------------------------------------------------------------
# ajuste de um modelo para a variável peso em função de daa, sem modelar a variância
# o modelo é um Gompertz reparametrizado para esse estudo
n0 <- nls(peso~A-(A-D)*exp(-exp(C*(daa-B))), data=goi,
          start=c(A=200, C=0.09, B=105, D=7))
summary(n0)
                                                                                     #
#-------------------------------------------------------------------------------------
# análises gráficas do ajuste
require(nlstools)
par(mfrow=c(2,2))
plot(profile(n0))
par(mfrow=c(3,2))
for(i in 1:6) plot(nlstools::nlsResiduals(n0), which=i)
layout(1)
                                                                                     #
#-------------------------------------------------------------------------------------
# modelo com inclusão da função de variâcia: var(e)=sigma^2*|ln{daa}|^{2*delta}
require(nlme)
n1 <- gnls(peso~A-(A-D)*exp(-exp(C*(daa-B))), data=goi,
           start=c(A=225, C=0.05, B=109, D=7),
           weights=varPower(0.1, form=~log(daa)))
print(n1, corr=FALSE)
                                                                                     #
#-------------------------------------------------------------------------------------
# teste para H0: power=0
anova(n1, n0)
intervals(n1) # intervalos assintóticos para os parâmetros
                                                                                     #
#-------------------------------------------------------------------------------------
# compara os valores das estimativas e erros padrões
summary(n0)$coeff
```

```
summary(n1)$tTable # erro padrão de A aumentou...
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# qqplot e residuos vs daa
qqmath(~residuals(n0, type="pearson")+residuals(n1, type="pearson"))
xyplot(residuals(n0, type="pearson")+residuals(n1, type="pearson")~goi$daa,
       jitter.x=TRUE, pch=19)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# gráfico dos valores preditos
require(latticeExtra)
pred <- data.frame(daa=seq(10, 140, l=100))
pred$n0 <- predict(n0, newdata=pred)
pred$n1 <- predict(n1, newdata=pred)
xyplot(peso~daa, data=goi)+
  as.layer(xyplot(n0~daa, data=pred, type="l", col=1))+
  as.layer(xyplot(pred$n1~daa, data=pred, type="l", col=2))
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# bandas de confiança para a média
invgomp <- deriv3(~A-(A-D)*exp(-exp(C*(daa-B))), c("A","C","B","D"), function(daa,A,B,C,D){NULL})
# bandas para o modelo homogêneo
A0 <- do.call(invgomp, args=c(as.list(coef(n0)), list(daa=pred$daa)))
F0 <- attr(A0, "gradient")
t0 <- qt(c(f0=0.5, l0=0.025, u0=0.975), df.residual(n0))
U0 <- chol(vcov(n0))
S0 <- sqrt(apply(F0%*%t(U0), 1, function(x) sum(x^2)))
Y0 <- apply(outer(S0, t0, "*"), 2, function(x) x+c(A0))
# bandas para o modelo heterogêneo
A1 <- do.call(invgomp, args=c(as.list(coef(n1)), list(daa=pred$daa)))
F1 <- attr(A1, "gradient")
t1 <- qt(c(f1=0.5, l1=0.025, u1=0.975), df.residual(n0))
U1 <- chol(vcov(n1))
S1 <- sqrt(apply(F%*%t(U1), 1, function(x) sum(x^2)))
Y1 <- apply(outer(S1, t1, "*"), 2, function(x) x+c(A1))
pred <- cbind(pred, Y0, Y1)
str(pred)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# gráfico com as bandas de confiância para a média
xyplot(peso~daa, data=goi, jitter.x=TRUE)+
  as.layer(xyplot(f0+l0+u0~daa, data=pred, type="l", col=1, lwd=c(2,1,1)))+
  as.layer(xyplot(f1+l1+u1~daa, data=pred, type="l", col=2, lwd=c(2,1,1)))
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# funções gráficas para fazer região preenchida entre as bandas de confiança
prepanel.biH <- function(x, y, ly, uy, subscripts, ...){
  x <- as.numeric(x)
  ly <- as.numeric(ly[subscripts])
  uy <- as.numeric(uy[subscripts])
  list(ylim=range(uy, ly, finite=TRUE), xlim=range(x))
}
panel.biH <- function(x, y, ly, uy, subscripts, colr, alpha=0.1, ...){
  y <- as.numeric(y)
  x <- as.numeric(x)
  or <- order(x)
  ly <- as.numeric(ly[subscripts])
  uy <- as.numeric(uy[subscripts])
  panel.polygon(c(x[or], x[rev(or)]),
                c(ly[or], uy[rev(or)]),
                col=colr, alpha=alpha, border=NA)
  panel.lines(x[or], ly[or], lty=3, lwd=0.5, col=1)
  panel.lines(x[or], uy[or], lty=3, lwd=0.5, col=1)
  panel.xyplot(x, y, subscripts=subscripts, ...)
}
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# região entre as bandas de confiança é preenchida
xyplot(peso~daa, data=goi, type="n")+
  as.layer(with(pred,
                xyplot(f0~daa, type="l", col=4, lwd=2,
                       ly=l0, uy=u0, colr=4, alpha=0.3,
                       prepanel=prepanel.biH,
                       panel=panel.biH))
           )+
  as.layer(with(pred,
                xyplot(f1~daa, type="l", col=3, lwd=2,
                       ly=l1, uy=u1, colr=3, alpha=0.3,
                       prepanel=prepanel.biH,
```

```
                      panel=panel.biH))
          )+
  as.layer(xyplot(peso~daa, data=goi, pch=1, col=1, jitter.x=TRUE))
                                                                                          #
#-------------------------------------------------------------------------------------------
# calculando os intervalos de confiança para a média amostral em cada daa
mic <- aggregate(goi$peso, list(daa=goi$daa),
                 function(x){
                   mean(x)+c(0,-1,1)*qt(0.975, df=length(x)-1)*sd(x)/sqrt(length(x))
                 })
                                                                                          #
#-------------------------------------------------------------------------------------------
# funções gráficas para fazer intervalos (bigodes) de confiança
prepanel.ciH <- function(x, y, ly, uy, subscripts=subscripts, ...){
  x <- as.numeric(x)
  ly <- as.numeric(ly[subscripts])
  uy <- as.numeric(uy[subscripts])
  list(ylim = range(y, uy, ly, finite = TRUE))
}
panel.ciH <- function(x, y, ly, uy, subscripts=subscripts, pch=pch, cola, ...){
  panel.xyplot(x, y, subscripts=subscripts, pch=pch, ...)
  x <- as.numeric(x)
  y <- as.numeric(y)
  ly <- as.numeric(ly[subscripts])
  uy <- as.numeric(uy[subscripts])
  panel.arrows(x, ly, x, uy, col=cola, length=0.05, angle=90, code=3)
}
                                                                                          #
#-------------------------------------------------------------------------------------------
# regiões de confiança e intervalos de confiança para a média amostral
xyplot(peso~daa, data=goi, type="n")+
  as.layer(with(pred,
                xyplot(f0~daa, type="l", col=4, lwd=2,
                       ly=l0, uy=u0, colr=4, alpha=0.3,
                       prepanel=prepanel.biH,
                       panel=panel.biH))
          )+
  as.layer(with(pred,
                xyplot(f1~daa, type="l", col=3, lwd=2,
                       ly=l1, uy=u1, colr=3, alpha=0.3,
                       prepanel=prepanel.biH,
                       panel=panel.biH))
          )+
 as.layer(with(mic,
               xyplot(x[,1]~daa, ly=x[,2], uy=x[,3], pch=5, col=1, cola=1,
                      prepanel=prepanel.ciH,
                      panel=panel.ciH))
         )
                                                                                          #
#-------------------------------------------------------------------------------------------
.
```

5.28 Modelando a correlação entre observações

```
.
#-------------------------------------------------------------------------------------------
# dados do livro Regression Analysis and Its Applications - Bates & Watts
# The Chloride data frame has 54 rows and 2 columns representing measurements of the chloride
# ion concentration in blood cells suspended in a salt solution.
# conc: a numeric vector giving the chloride ion concentration (%).
# time: a numeric vector giving the time of the concentration measurement (min).
conc <- c(17.3, 17.6, 17.9, 18.3, 18.5, 18.9, 19, 19.3, 19.8, 19.9, 20.2, 20.5, 20.6,
          21.1, 21.5, 21.9, 22, 22.3, 22.6, 22.8, 23, 23.2, 23.4, 23.7, 24, 24.2,
          24.5, 25, 25.4, 25.5, 25.9, 25.9, 26.3, 26.2, 26.5, 26.5, 26.6, 27, 27,
          27, 27, 27.3, 27.8, 28.1, 28.1, 28.1, 28.4, 28.6, 29, 29.2, 29.3, 29.4,
          29.4, 29.4)
time <- c(2.45, 2.55, 2.65, 2.75, 2.85, 2.95, 3.05, 3.15, 3.25, 3.35, 3.45, 3.55, 3.65,
          3.75, 3.85, 3.95, 4.05, 4.15, 4.25, 4.35, 4.45, 4.55, 4.65, 4.75, 4.85, 4.95,
          5.05, 5.15, 5.25, 5.35, 5.45, 5.55, 5.65, 5.75, 5.85, 5.95, 6.05, 6.15, 6.25,
          6.35, 6.45, 6.55, 6.65, 6.75, 6.85, 6.95, 7.05, 7.15, 7.25, 7.35, 7.45, 7.55,
          7.65, 7.75)
plot(conc~time)
lines(spline(conc~time))
                                                                                          #
#-------------------------------------------------------------------------------------------
# ajustando o modelo de regressão não linear
```

```
fm1 <- nls(conc~Asym*(1-prop*exp(-exp(lrc)*time)),
           start=c(Asym=50, prop=0.6, lrc=log(0.25)), trace=TRUE)
plot(resid(fm1)~fitted(fm1), type="b") # plot shows patterns in the residuals
abline(h=0, lty=2, lwd=1)
r <- residuals(fm1)
plot(r[-1], r[-length(resid(fm1))]) # plot de e_{t} ~ e_{t-1}
abline(h=0, lty=2, lwd=1)
cor(r[-1], r[-length(resid(fm1))])
acf(r)
pacf(r)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# veja os erros padrões
summary(fm1)$coeff
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# ajustando modelo declarando correlação
require(nlme)
fm2 <- gnls(conc~Asym*(1-prop*exp(-exp(lrc)*time)),
            start=c(Asym=50, prop=0.6, lrc=log(0.25)),
            correlation=corExp(0.9, form=~time))
fm2
summary(fm2)
anova(fm2, fm1)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# praticamente o mesmo ajuste
plot(conc~time)
lines(fitted(fm1)~time)
lines(fitted(fm2)~time)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# compare os erros padrões
summary(fm1)$coeff # são menores porque correlação positiva induz menor variância
summary(fm2)$tTable
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# qual é a cara da matriz de correlação?
tm <- seq(0,1,by=0.1)
outer(tm, tm, function(x, y) round(exp(-abs(x-y)/0.2619), 2))
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# fazer o gráfico com IC
time.g <- seq(min(time), max(time), l=200)
mdl <- deriv3(~Asym*(1-prop*exp(-exp(lrc)*time)), c("Asym","prop","lrc"), function(time,Asym,prop,lrc){NULL})
A0 <- do.call(mdl, args=c(as.list(coef(fm1)), list(time=time.g)))
F0 <- attr(A0, "gradient")
t0 <- qt(c(f0=0.5, l0=0.025, u0=0.975), df=length(conc)-length(coef(fm1)))
U0 <- chol(vcov(fm1))
S0 <- sqrt(apply(F0%*%t(U0), 1, function(x) sum(x^2)))
Y0 <- apply(outer(S0, t0, "*"), 2, function(x) x+c(A0))
A1 <- do.call(mdl, args=c(as.list(coef(fm2)), list(time=time.g)))
F1 <- attr(A1, "gradient")
t1 <- qt(c(f1=0.5, l1=0.025, u1=0.975), df=length(conc)-length(coef(fm2))-1)
U1 <- chol(vcov(fm2))
S1 <- sqrt(apply(F1%*%t(U1), 1, function(x) sum(x^2)))
Y1 <- apply(outer(S1, t1, "*"), 2, function(x) x+c(A1))
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# gráfico com bandas
spl <- spline(conc~time)
#xyplot(conc~time)+
xyplot(conc~time, xlim=c(6,7), ylim=c(26.5,28.5))+
  as.layer(xyplot(Y1[,1]~time.g, type="l", col=1, lwd=1,
                  ly=Y1[,2], uy=Y1[,3], colr=2, alpha=0.1,
                  prepanel=prepanel.biH,
                  panel=panel.biH))+
  as.layer(xyplot(Y0[,1]~time.g, type="l", col=4, lwd=1,
                  ly=Y0[,2], uy=Y0[,3], colr=4, alpha=0.4,
                  prepanel=prepanel.biH,
                  panel=panel.biH))+
  as.layer(xyplot(spl$y~spl$x, type="l", col=2, lwd=1))
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# predição com a geoR. (talvez excluir isso)
require(geoR)
da <- as.geodata(data.frame(x=time, y=0, r=r))
dag <- data.frame(x=time.g, y=0)
pars <- c(c(fm2$sigma), exp(c(fm2$modelStruct$corStruct)))
```

```
kg <- krige.conv(da, location=dag, krige=krige.control(cov.pars=pars))
xyplot(conc~time)+
  as.layer(xyplot(Y1[,1]~time.g, type="l", col=1, lwd=1,
                  ly=Y1[,2], uy=Y1[,3], colr=2, alpha=0.1,
                  prepanel=prepanel.biH,
                  panel=panel.biH))+
  as.layer(xyplot(c(Y1[,1]+kg$predict)~time.g, type="l", col=2, lwd=1))
                                                                                                    #
#-------------------------------------------------------------------------------------------
```

5.29 Problemas envolvendo a má especificação do modelo

```
#-------------------------------------------------------------------------------------------
# dados de mineralização de nitrogênio
# tempos de coleta, composto de lixo, esterco de galinha
tp <- c(30, 45, 60, 75, 90, 105, 120, 150, 180, 210, 240, 270)
cl <- c(24.11, 49, 71.01, 93.49, 103.97, 111.48, 118.72, 122.77, 132.42, 149.94, 157.4, 162.14)
eg <- c(33.71, 63.91, 79.79, 104.1, 116.79, 131.17, 145.18, 154.46, 177.5, 185.14, 197.61, 198.95)
                                                                                                    #
#-------------------------------------------------------------------------------------------
# exploração gráfica
require(lattice)
xyplot(cl+eg~tp, type="b")
                                                                                                    #
#-------------------------------------------------------------------------------------------
# ajustando modelo para esterco de galinha - exponencial reparametrizado de 3 parâmetros
require(nlme)
n0 <- gnls(eg~A*(1-exp(-log(2)*(tp)/V))+D*tp, star=list(A=200, V=100, D=1))
summary(n0)                    # parâmetros não significativos!!
plot(eg~tp); lines(fitted(n0)~tp) # estimação plausível
pacf(residuals(n0)) # sem indícios de correlação
acf(residuals(n0))  # sem indícios de correlação
                                                                                                    #
#-------------------------------------------------------------------------------------------
# mas se fosse para declarar a correlação seria assim
n1 <- update(n0, start=as.list(coef(n0)), correlation=corCAR1(0.5, form=~tp))
summary(n1)                    # ainda parâmetros não significativos!!
plot(eg~tp); lines(fitted(n0)~tp) # estimação plausível
lines(fitted(n1)~tp, col=2)       # mesma predição
anova(n0, n1)                  # correlação CAR1 não significativa
                                                                                                    #
#-------------------------------------------------------------------------------------------
# os parâmetros são não significativos, mas o modelo é significativo comparado ao trivial
anova(n0, lm(eg~1))
                                                                                                    #
#-------------------------------------------------------------------------------------------
# desconfiar de má especificação do modelo, deve faltar parâmetro
tp.g <- seq(0, 300, by=2)
plot(eg~tp, xlim=range(tp.g), ylim=c(0,max(eg)))
lines(predict(n0, newdata=data.frame(tp=tp.g))~tp.g) # estimação plausível
                                                                                                    #
#-------------------------------------------------------------------------------------------
# tentar outro modelo não linear, sem restrição y -> 0 quando x -> 0
n2 <- gnls(eg~A*(1-exp(-log(2)*(tp-t0)/V))+D*tp, star=list(A=200, V=100, D=1, t0=0))
summary(n2)                    # parâmetros significativos!!
plot(eg~tp, xlim=range(tp.g), ylim=c(0,max(eg)))
lines(predict(n0, newdata=data.frame(tp=tp.g))~tp.g)        # estimação plausível
lines(predict(n2, newdata=data.frame(tp=tp.g))~tp.g, col=2) # estimação BEM plausível
                                                                                                    #
#-------------------------------------------------------------------------------------------
# veja o que a má especificação do modelo pode fazer!
# a liberação de nitrogênio não começa no instante zero, demora um tempo para iniciar
anova(n2, n0)
                                                                                                    #
#-------------------------------------------------------------------------------------------
# plotar o gráfico com IC
expt0 <- deriv3(~A*(1-exp(-log(2)*(tp-t0)/V))+D*tp, c("A","V","D","t0"), function(tp,A,V,D,t0){NULL})
A0 <- do.call(expt0, args=c(as.list(coef(n2)), list(tp=tp.g)))
F0 <- attr(A0, "gradient")
t0 <- qt(c(f0=0.5, l0=0.025, u0=0.975), df=length(eg)-length(coef(n2)))
U0 <- chol(vcov(n2))
S0 <- sqrt(apply(F0%*%t(U0), 1, function(x) sum(x^2)))
Y0 <- apply(outer(S0, t0, "*"), 2, function(x) x+c(A0))
                                                                                                    #
#-------------------------------------------------------------------------------------------
```

```
# gráfico com bandas
xyplot(eg~tp, xlim=c(0,300), ylim=c(0,250))+
  as.layer(xyplot(Y0[,1]~tp.g, type="l", col=2, lwd=2,
                  ly=Y0[,2], uy=Y0[,3], colr=3, alpha=0.2,
                  prepanel=prepanel.biH,
                  panel=panel.biH))
                                                                                                #
#-------------------------------------------------------------------------------------------
# qual a interpretação dos parâmetros
summary(n2)$tTable
trellis.focus("panel", 1, 1)
panel.abline(v=coef(n2)["t0"], h=coef(n2)["A"], lty=2)
panel.abline(a=0, b=coef(n2)["D"], lty=2)
panel.curve(coef(n2)["A"]*(1-exp(-log(2)*(x-coef(n2)["t0"])/coef(n2)["V"])), lty=2)
trellis.unfocus()
                                                                                                #
#-------------------------------------------------------------------------------------------
.
```

5.30 Pacotes R para análise de regressão não linear

```
# http://www.ncfaculty.net/dogle/fishR/index.html
# http://cran.r-project.org/web/packages/scapeMCMC/vignettes/dsc-vignette.pdf
# http://cran.r-project.org/web/packages/gnm/index.html
# http://cran.r-project.org/web/packages/NISTnls/index.html
# http://cran.r-project.org/web/packages/nlreg/index.html
# http://cran.r-project.org/web/packages/nlrwr/index.html
# http://cran.r-project.org/web/packages/npde/index.html
# http://cran.r-project.org/web/packages/NRAIA/index.html
# http://cran.r-project.org/web/packages/HydroMe/index.html
```

5.31 Curvas de materia seca acumulada

```
.
#-------------------------------------------------------------------------------------------
# dados
msa <- read.table("../dados/msacum.txt", header=TRUE)
str(msa)
names(msa) <- tolower(names(msa))
summary(msa)
#-------------------------------------------------------------------------------------------
# gráficos
require(lattice)
xyplot(mst~dae, groups=cult, data=msa, type="o")
xyplot(mst~gde, groups=cult, data=msa, type="o")
xyplot(mst~dsv, groups=cult, data=msa, type="o")
                                                                                                #
#-------------------------------------------------------------------------------------------
# ajustando curvas separadas, usando a nlsList e a selfStart SSlogis()
require(nlme)
msa <- groupedData(mst~dsv|cult, data=msa)
n0 <- nlsList(mst~SSlogis(dsv,A,x0,S), data=msa)
plot(residuals(n0)~fitted(n0))
qqnorm(residuals(n0)); qqline(residuals(n0))
summary(n0)
plot(augPred(n0))   # gráfico com preditos e observados
plot(intervals(n0)) # gráfico com intervalos assintóticos
                                                                                                #
#-------------------------------------------------------------------------------------------
# ajustar com a nls e obter o mesmo
msa$cult <- factor(msa$cult, ordered=FALSE)
n1 <- nls(mst~A[cult]/(1+exp(-(dsv-x0[cult])/S[cult])), data=msa,
          start=list(A=coef(n0)[,1],x0=coef(n0)[,2],S=coef(n0)[,3]))
summary(n1)
ci <- confint(n1)
ci <- cbind(expand.grid(cult=levels(msa$cult), par=c("A","x0","S")),
            as.data.frame(ci))
names(ci)[3:4] <- c("L","U")
ci
                                                                                                #
#-------------------------------------------------------------------------------------------
# gráfico com os IC perfilhados, são assimétricos
```

```
require(latticeExtra)
segplot(cult~L+U|par, scales=list(x="free"), data=ci, band=FALSE,
        center=coef(n1), layout=c(3,1))
                                                                                                #
#-------------------------------------------------------------------------------------------
# ajustando com a retrição de que P e C tem o mesmo x0
msa$cultx0 <- factor(msa$cult)
levels(msa$cultx0) <- c("D","CP","CP")
n2 <- nls(mst~A[cult]/(1+exp(-(dsv-x0[cultx0])/S[cult])), data=msa,
          start=list(A=coef(n0)[,1],x0=c(0.87,0.97),S=coef(n0)[,3]))
summary(n2)
confint(n2)
anova(n1, n2)
                                                                                                #
#-------------------------------------------------------------------------------------------
# fazer o gráfico com os preditos e observados
range(msa$dsv)
pred <- expand.grid(dsv=seq(0,2,l=30), cult=levels(msa$cult))
pred$cultx0 <- factor(pred$cult)
levels(pred$cultx0) <- c("D","CP","CP")
pred$y <- predict(n2, newdata=pred)
xyplot(mst~dsv, groups=cult, data=msa)+
  as.layer(xyplot(y~dsv, groups=cult, data=pred, type="l"))
                                                                                                #
#-------------------------------------------------------------------------------------------
.
```

6 Análise de experimento com um fator em DIC

6.1 Carregando dados com a função `textConnection()`

```
.
#-------------------------------------------------------------------------------------------
# dados de experimento com plantulas de sorgo
# gen: genótipo; diam: diâmetro das raízes
lines <-
  "gen        comp        arsu        volu        diam
ATF06B        1093.149        127.679        1.190        0.371
ATF06B        841.336        111.200        1.173        0.425
ATF06B        904.775        112.625        1.118        0.394
ATF40B        855.092        117.029        1.275        0.433
ATF40B        1226.227        168.686        1.849        0.429
ATF40B        1271.738        144.250        1.306        0.362
ATF54B        754.458        93.635        0.939        0.420
ATF54B        830.376        104.637        1.056        0.399
ATF54B        611.053        73.885        0.712        0.382
BR001B        1114.234        140.970        1.421        0.399
BR001B        812.591        112.544        1.240        0.441
BR001B        1315.109        163.264        1.614        0.394
BR005B        516.535        58.053        0.520        0.358
BR005B        548.341        62.525        0.567        0.364
BR005B        488.555        50.481        0.416        0.320
BR007B        1179.349        145.470        1.431        0.388
BR007B        1155.819        143.775        1.425        0.398
BR007B        977.735        118.955        1.152        0.387
BR008B        1267.830        148.383        1.398        0.376
BR008B        1094.307        145.242        1.552        0.417
BR008B        1172.689        128.847        1.127        0.347
P9401        637.272        95.581        1.141        0.478
P9401        703.597        95.284        1.032        0.434
P9401        959.356        121.451        1.226        0.408
SC283        869.051        97.603        0.872        0.357
SC283        839.449        107.325        1.093        0.412
SC283        1016.686        120.438        1.136        0.378"
raiz <- read.table(textConnection(lines), header=TRUE); closeAllConnections()
str(raiz)
                                                                                                #
#-------------------------------------------------------------------------------------------
# instalação de pacotes para o teste de Tukey e Scott-Knott
# install.packages("agricolae", dep=TRUE)
# install.packages("ScottKnott", dep=TRUE)
# install.packages("contrast", dep=TRUE)
# install.packages("multcomp", dep=TRUE)
                                                                                                #
#-------------------------------------------------------------------------------------------
```

```
# conferir se temos fatores para fazer a análise de variância
is.factor(raiz$gen)  # é fator?
levels(raiz$gen)     # quais os níveis?
nlevels(raiz$gen)    # quantos níveis?
is.numeric(raiz$gen)
is.character(raiz$gen)
class(raiz$gen)
                                                                                        #
#---------------------------------------------------------------------------------------
.
```

6.2 Modelo estatístico de efeitos e tipos de parametrização

$$Y_{ij} = \mu + \alpha_i + \epsilon_{ij} \qquad \epsilon_{ij} \overset{iid}{\sim} \mathcal{N}(0, \sigma^2_\epsilon) \tag{1}$$

$$\boldsymbol{Y} = [\mathbf{1}_\mu | \boldsymbol{X}_\alpha] \begin{bmatrix} \mu \\ \boldsymbol{\alpha} \end{bmatrix} + \boldsymbol{\epsilon} \tag{2}$$

```
.
#---------------------------------------------------------------------------------------
# representação matricial do modelo estatístico, montar a matriz de incidência de \alpha_i
# a constante \mu sempre incide em todas as observações
# essa é a matriz de deriavadas primeiras de f(x,\beta) em relação à \betas
sapply(levels(raiz$gen),
       function(i){
         ifelse(raiz$gen==i, 1, 0)
       })
                                                                                        #
#---------------------------------------------------------------------------------------
# o espaço vetorial descrito pela matrix X pode estar associado à diversas parametrizações
# (ou restrições) do vetor de efeitos \beta
contrasts(C(raiz$gen, treatment)) # (default) impõe que o primeiro nível tem efeito 0
contrasts(C(raiz$gen, SAS))       # impõe que o último nível tem efeito 0
contrasts(C(raiz$gen, sum))       # impõe a restrição de soma dos efetios igual zero
contrasts(C(raiz$gen, helmert))   # impõe retrição "o próximo contra todos"
contrasts(C(raiz$gen, poly))      # impõe parametriação de polinômios ortogonais (fator ordenado)
# além dessas, o usuário pode declarar qualquer outra parametrização
                                                                                        #
#---------------------------------------------------------------------------------------
# matriz do modelo (será usado só 3 níveis)
ftr <- raiz$gen[raiz$gen%in%levels(raiz$gen)[1:3]]; ftr <- ftr[, drop=TRUE]
X <- model.matrix(~ftr); X
X <- model.matrix(~-1+ftr); X
X <- model.matrix(~ftr, contrast=list(ftr="contr.SAS")); X
X <- model.matrix(~ftr, contrast=list(ftr="contr.sum")); X
X <- model.matrix(~ftr, contrast=list(ftr="contr.helmert")); X
# independente do contrate usado, os valores ajustados serão sempre os mesmos
                                                                                        #
#---------------------------------------------------------------------------------------
# montando minha própria matriz de contrastes e sua matriz de incidência
contr.my <- rbind("c1"=c(-0.5,0.5), "c2"=c(pi,pi), "c3"=c(0,1))
M <- model.matrix(~-1+ftr) # matriz irrestrita
X <- M%*%contr.my          # matriz correspondente à sua parametrização
X
contrasts(ftr) <- contr.my
X <- model.matrix(~ftr); X
X%*%solve(cbind(1, contr.my)) # volta para matriz irrestrita
                                                                                        #
#---------------------------------------------------------------------------------------
```

6.3 Estimação pelo método dos quadrados mínimos

```
.
#---------------------------------------------------------------------------------------
# obter as estimativas de \mu e de I-1 efeitos associados aos níveis do genótipos
# usar o método de mínimos quadrados
contrasts(C(raiz$gen, treatment))
contrasts(raiz$gen) <- "contr.treatment" # declara o tipo de restrição
X <- model.matrix(~gen, data=raiz)
```

```
Y <- as.matrix(raiz$diam)
betas <- solve(t(X)%*%X)%*%t(X)%*%Y # **solução** do sistema de equações lineares
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# estimativas das médias (\mu+\alpha_i) dos tratamentos
cbind(1, contrasts(raiz$gen))%*%betas # médias de mínimos quadrados
tapply(raiz$diam, raiz$gen, mean)     # médias amostrais
#-------------------------------------------------------------------------------------------------
# uso de funções mais apropriadas para o cálculo dos \betas
solve(crossprod(X), crossprod(X, Y)) # computacionalmente mais eficiente
QR <- qr(X)                          # usando a decomposição QR
R <- qr.R(QR); Q <- qr.Q(QR)         # matrizes Q e R da decomposição
solve(R)%*%t(Q)%*%Y                  # ainda mais eficiente
solve(R, crossprod(Q, Y))            # ainda bem mais eficiente
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# valores ajustados, resíduos e variância
Y.fitted <- X%*%betas
plot(Y~Y.fitted)
res <- Y-Y.fitted # resíduos do ajuste
plot(res)
qqnorm(res); qqline(res)
s2 <- sum(res^2)/(length(Y)-length(betas)); s2 # estimativa da variância
mean(tapply(raiz$diam, raiz$gen, var))         # médias das variâncias amostrais
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# erros padrões das estimativas dos efeitos
vcovb <- s2*solve(t(X)%*%X)
diag(vcovb)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# erros padrões das médias (\mu+\alpha_i)
contr <- cbind(1, contrasts(raiz$gen))
contr%*%betas
vcovm <- contr%*%(s2*solve(t(X)%*%X))%*%t(contr)
diag(vcovm)                      # variância da média de mínimos quadrados
tapply(raiz$diam, raiz$gen, var) # variância das médias amostrais
                                                                                                    #
#-------------------------------------------------------------------------------------------------
.
```

6.4 Ajuste do modelo e análise de variância

```
.
#-------------------------------------------------------------------------------------------------
# faz estimação dos parâmetros
a0 <- aov(diam~gen, data=raiz)
a0
str(a0)        # elementos que o objeto do ajuste armazena
class(a0)      # classe do objeto
anova(a0)      # quadro de análise de variância obtido com o método anova
summary(a0)    # também quadro de análise de variância quando a classe é aov
summary.lm(a0) # quadro das estimativas dos efeitos (solução)
a0 <- lm(diam~gen, data=raiz)
class(a0)      # classe do objeto
anova(a0)      # quadro de análise de variância obtido com o método anova
summary(a0)    # quadro das estimativas dos efeitos quando a classe é lm
confint(a0)    # intervalo de confiança para os efeitos
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# análise gráfica dos resíduos
par(mfrow=c(2,2))
plot(a0)
layout(1)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# teste das pressuposições do modelo (normalidade e homocedasticidade)
shapiro.test(residuals(a0))
bartlett.test(raiz$diam~raiz$gen)     # em dic pode-se usar os dados
bartlett.test(residuals(a0)~raiz$gen) # como pode-se usar os resíduos
require(car)
leveneTest(raiz$diam, group=raiz$gen, center=median) # default (mais robusto)
leveneTest(raiz$diam, group=raiz$gen, center=mean)   # Levene original
leveneTest(a0)    # também recebe o objeto do modelo
residualPlots(a0) # teste de Tukey para a aditividade (boxplot 5 números 3 repetições)
                                                                                                    #
```

```
#-------------------------------------------------------------------------------------
# gráfico qqnorm com bandas de confiança
res <- residuals(a0); n <- length(res)
qq <- qqnorm(res); qqline(res, col=2, lwd=2)
res.sample <- replicate(1000, sort(sample(res, n, repl=TRUE)))
mm <- apply(res.sample, 1, quantile, probs=c(0.05, 0.95))
lines(sort(qq$x), mm[1,], col=4)
lines(sort(qq$x), mm[2,], col=4)
nor.sample <- replicate(1000, sort(rnorm(n, 0, sd=sd(res))))
mm <- apply(nor.sample, 1, quantile, probs=c(0.05, 0.95))
lines(sort(qq$x), mm[1,], col=2)
lines(sort(qq$x), mm[2,], col=2)
                                                                                    #
#-------------------------------------------------------------------------------------
# gráficos da car
qqPlot(res, dist="norm", envelope=0.95)                  # norm vs norm
qqPlot(lm(diam~gen, raiz), simulate=TRUE, envelope=0.95) # norm vs t, ic por bootstrap
# distribution=c("t", "norm"), line=c("robust", "quartiles", "none")
                                                                                    #
#-------------------------------------------------------------------------------------
.
```

6.5 Aplicando contrastes entre níveis

```
.
#-------------------------------------------------------------------------------------
# matricialmente um contraste é estimado por, nível 1 vs nível 2
nlevels(raiz$gen)
contr <- cbind(1,-1,0,0,0,0,0,0,0)                     # contraste entre níveis
betas <- coef(a0)                                      # estimativas dos efeitos
restr <- cbind(1, contrasts(raiz$gen))                 # tipo de contraste restrição
contr%*%restr%*%betas                                  # estimativa do contraste
contr%*%restr%*%(s2*solve(t(X)%*%X))%*%t(contr%*%restr) # variância do contraste
                                                                                    #
#-------------------------------------------------------------------------------------
# gmodels::fit.contrast()
require(gmodels)
fit.contrast(a0, varname="gen", contr, conf.int=0.95)              # faz um contraste
contr2 <- rbind(c1=c(-1,1,0,0,0,0,0,0,0), c2=c(-1,0,1,0,0,0,0,0,0)) # 2 ou + contrastes
fit.contrast(a0, varname="gen", contr2, conf.int=0.95)
                                                                                    #
#-------------------------------------------------------------------------------------
# gmodels::glh.test()
glh <- glh.test(a0, contr%*%restr)  # aqui tem que passar o contraste em termos de efeito
summary(glh)                        # é um teste de Wald (F)
glh <- glh.test(a0, contr2%*%restr)
summary(glh)                        # testa simultâneamente 2 ou + contrastes
                                                                                    #
#-------------------------------------------------------------------------------------
# gmodels::estimable()
cont.ef <- contr%*%restr
colnames(cont.ef) <- colnames(coef(a0)) # estimable() checa os nomes
estimable(a0, cont.ef, conf.int=0.95)   # contraste em termos de efeito
cont.ef <- contr2%*%restr
colnames(cont.ef) <- colnames(coef(a0)) # 2 ou + contrastes
estimable(a0, cont.ef, conf.int=0.95)   # faz testes individuais
estimable(a0, cont.ef, joint.test=TRUE) # faz o teste conjunto (Wald, chi²)
                                                                                    #
#-------------------------------------------------------------------------------------
# aod::wald.test(), preciso passar só vcov() e coef()
require(aod)
wald.test(Sigma=vcov(a0), b=coef(a0), L=cont.ef)                    # idem a estimable()
wald.test(Sigma=vcov(a0), b=coef(a0), L=cont.ef, df=df.residual(a0)) # idem a glh.test()
                                                                                    #
#-------------------------------------------------------------------------------------
# car::linearHypothesis()
linearHypothesis(a0, hypothesis.matrix=cont.ef, test=c("F"))     # idem a estimable()
linearHypothesis(a0, hypothesis.matrix=cont.ef, test=c("Chisq")) # idem a glh.test()
                                                                                    #
#-------------------------------------------------------------------------------------
# contrast::contrast(), mais de manipular contrastes em modelos com muitos termos
require(contrast)
levels(raiz$gen)
# média estimativa de mínimos quadrados
contrast(a0, list(gen="ATF40B"))
contrast(a0, list(gen=levels(raiz$gen)))
```

```
# um contraste entre dois níveis
contrast(a0, list(gen="ATF06B"), list(gen="ATF40B"))
# dois contrastes envolvendo um nível contra dois
contrast(a0, list(gen="ATF06B"), list(gen=c("ATF40B","ATF54B")))
# dois contrastes envolvolvendo primeiros e segundos entre si
contrast(a0, list(gen=c("BR001B","BR005B")), list(gen=c("ATF40B","ATF54B")))
# um contraste entre a média dos ATF vs P9401
gen.lev <- levels(raiz$gen)
contrast(a0, type="average", list(gen=gen.lev[1:3]), list(gen="P9401"))
# um contraste entre a média de 3 ATF vs a média de 3 BR00 (ambos mesmo número de níveis)
contrast(a0, type="average",
         list(gen=gen.lev[1:3]), list(gen=gen.lev[4:6]))
                                                                                                #
#-------------------------------------------------------------------------------------------
```

6.6 Aplicando contrastes entre grupos de níveis

```
.
#-------------------------------------------------------------------------------------------
# fazer 3 ATF vs 4 BR00, usando a gmodels::fit.contrast()
gen.lev
contr <- rbind(ATFvsBR00=rep(c(1/3,-1/4,0), c(3,4,2)))
fit.contrast(a0, "gen", contr)
                                                                                                #
#-------------------------------------------------------------------------------------------
# usando a contrast::contrast() e gmodels::estimable()
cATF <- contrast(a0, type="average", list(gen=gen.lev[1:3]))
cBR00 <- contrast(a0, type="average", list(gen=gen.lev[4:7]))
cATF$X # vetor que especifica o contraste
contr <- cATF$X-cBR00$X; rownames(contr) <- "ATRvsBR00"
estimable(a0, contr)
                                                                                                #
#-------------------------------------------------------------------------------------------
# fazer todos os contrastes entre os grupos, serão 4 grupos 2 a 2: 6 contrastes
# vamos que montar a matriz de contrastes, 6 linhas 9 colunas
cP9401 <- contrast(a0, type="average", list(gen="P9401"))
cSC283 <- contrast(a0, type="average", list(gen="SC283"))
# matriz dos coeficientes de cada grupo
contr.gr <- sapply(list(ATF=cATF, BR=cBR00, P=cP9401, SC=cSC283), function(x) x$X)
contr.gr <- t(contr.gr)
colnames(contr.gr) <- names(coef(a0))
contri <- rownames(contr.gr)  # são os 4 grupos
contr.id <- t(combn(contri, 2))
contr.ok <- contr.gr[contr.id[,1],]-contr.gr[contr.id[,2],]
rownames(contr.ok) <- apply(contr.id, 1, paste, collapse="-")
contr.ok
                                                                                                #
#-------------------------------------------------------------------------------------------
# agora é só estimar os contrastes
estimable(a0, contr.ok, conf.int=0.95) # veja que os erros padrões são diferentes
                                                                                                #
#-------------------------------------------------------------------------------------------
# usando a mulcomp::glht()
require(multcomp)
mc <- glht(a0, linfct=contr.ok)
summary(mc)                              # por padrão ajusta os p-valores
summary(mc, test=adjusted(type="none"))   # nenhum método de correção do p-valor
ciglht <- confint(mc, calpha=univariate_calpha(type="none"))
plot(ciglht)
abline(v=0)
ciglht <- confint(mc)
plot(ciglht)
abline(v=0)
                                                                                                #
#-------------------------------------------------------------------------------------------
# gráfico
require(HH)
contr.mmc <- contr.ok%*%cbind(-1, contrasts(raiz$gen)) # deve ser em termos de \mu_\alpha_i
contr.mmc <- t(contr.mmc)
rownames(contr.mmc) <- gen.lev # nomes das linhas *deve* ser o nome dos níveis
mmc <- glht.mmc(a0, linfct=mcp(gen="Tukey"), focus.lmat=contr.mmc)
plot(mmc)
mmc$lmat
                                                                                                #
#-------------------------------------------------------------------------------------------
.
```

6.7 Estudando contrastes dentro da análise de variância

```
.
#-------------------------------------------------------------------------------------------
# é necessário que se especifique a matriz de contrastes
mcaov <- cbind(cATFBRvsPSC=c(1,1,1,1,1,1,1,-3.5,-3.5)/3.5,
               ATFvsBR=c(1/c(3,3,3,-4,-4,-4,-4),0,0),
               ATF1vs23=c(1,-0.5,-0.5,0,0,0,0,0,0),
               ATF2vs3=c(0,1,-1,0,0,0,0,0,0),
               BR12vs34=c(0,0,0,0.5,0.5,-0.5,-0.5,0,0),
               BR1vs2=c(0,0,0,1,-1,0,0,0,0),
               BR3vs4=c(0,0,0,0,0,1,-1,0,0),
               PvsSC=c(0,0,0,0,0,0,0,1,-1))
round(t(mcaov)%*%(mcaov), 3) # contrastes ortogonais
                                                                                          #
#-------------------------------------------------------------------------------------------
# passando os contrastes dentro da aov (importante para fatoriais com adicionais)
a0 <- aov(diam~gen, data=raiz, contrast=list(gen=mcaov))
ctr.list <- as.list(1:ncol(mcaov)); names(ctr.list) <- colnames(mcaov)
ctr.list  # nome do contraste e linha correspondente da matriz
summary(a0, split=list("gen"=ctr.list))  # cada contraste te 1 grau de liberdade
ctr.list2 <- list("c1vs2"=1, "ATFvsBR"=2, "inATF"=3:4, "inBR"=5:7, "PvsSC"=8)
ctr.list2 # nome do contraste e linhas correspondentes da matriz
summary(a0, split=list("gen"=ctr.list2)) # agrupando contrastes
                                                                                          #
#-------------------------------------------------------------------------------------------
# é meio perda de tempo fazer isso dentro da anova porque você vai precisar das estimativas
fit.contrast(a0, "gen", t(mcaov)) # só dão mesmo p-valor são contrastes ortogonais
# na anova, o teste (F) de hipótese para o contraste é sequêncial
# com o teste t, o teste de hipótese para o contraste é marginal
                                                                                          #
#-------------------------------------------------------------------------------------------
# podemos colocar esses contrastes em um gráfico
colnames(mcaov) <- gsub("vs", "-", colnames(mcaov)) # nome *deve* ter traço
rownames(mcaov) <- gen.lev # nomes das linhas *deve* ser o nome dos níveis
mmcaov <- glht.mmc(a0, linfct=mcp(gen="Tukey"), focus.lmat=mcaov, level=0.9)
plot(mmcaov)
mmcaov$lmat
                                                                                          #
#-------------------------------------------------------------------------------------------
.
```

6.8 Aplicando testes de médias: *t*, Bonferroni, Duncan, SNK, Tukey, Scheffe.

```
.
#-------------------------------------------------------------------------------------------
# testes para H0: \mu+alpha_i = \mu_\alpha_i' para todo i diferente de i'
# o tipo de teste e o nível nominal de significância devem ser definidos no planejamento
                                                                                          #
#-------------------------------------------------------------------------------------------
# TukeyHSD é do bacote stats (nativo) e funciona para objetos de classe aov apenas
detach(package:HH); detach(package:multcomp) # descarregar pois alteram a TukeyHSD()
tk <- TukeyHSD(aov(diam~gen, data=raiz))     # não tem "letrinhas" (frescura)
plot(tk); abline(v=0)
tk$gen <- tk$gen[order(tk$gen[,1], decreasing=TRUE),]
plot(tk); abline(v=0) # gráfico com IC ordenados
                                                                                          #
#-------------------------------------------------------------------------------------------
# usando a HH
require(HH)
tk <- glht.mmc(a0, linfct=mcp(gen="Tukey"))
tk
plot.matchMMC(tk$mca); abline(v=0) # idem à TukeyHSD()
plot(tk)                           # torna-se complicada quando tense muitas comparações
                                                                                          #
#-------------------------------------------------------------------------------------------
# usando a multcomp
tk <- glht(a0, linfct=mcp(gen="Tukey"))
tks <- summary(tk) # single-step (computacionalmente intensivo, demora para muitas comparações)
tks <- summary(tk, test=adjusted(type="bonferroni"))
tks <- summary(tk, test=adjusted(type="fdr"))
cld(tks)       # é possível obter uma representação com letras também
plot(cld(tks)) # boxpots com letras em cima
                                                                                          #
#-------------------------------------------------------------------------------------------
```

```
# esses testes são baseados nas médias de mínimos quadrados, procedimento vale para dados
# desbalanceados como veremos nas sessões seguintes
                                                                                          #
#-------------------------------------------------------------------------------------------
# para as funções do agricole a seguir são necessários os ingredientes:
glr <- df.residual(a0) # extrai o grau de liberdade do resíduo
sqr <- deviance(a0)    # extrais a soma de quadrados dos resíduos "desviância"
qmr <- sqr/glr         # calcula o quadrado médio do resíduo
                                                                                          #
#-------------------------------------------------------------------------------------------
# aplica o teste t 2 à 2: teste t não controla a fwer (familywise error rate)
require(agricolae)
lsd <- LSD.test(y=raiz$diam, trt=raiz$gen, DFerror=glr, MSerror=qmr, alpha=0.05)
lsd # você pode exportar essa tabela, fazer o gráfico dos IC, etc
                                                                                          #
#-------------------------------------------------------------------------------------------
# aplica o teste t seguido da correção de Bonferroni: significância nominal/ncomparações
lsd <- LSD.test(raiz$diam, raiz$gen, glr, qmr, alpha=0.05, p.adj="bonferroni")
lsd
                                                                                          #
#-------------------------------------------------------------------------------------------
# aplica o teste t com correção pelo critério FDR: false discovery rate
lsd <- LSD.test(raiz$diam, raiz$gen, glr, qmr, alpha=0.05, p.adj="fdr")
lsd
                                                                                          #
#-------------------------------------------------------------------------------------------
# Duncan: é mais conservador que o t mais ainda tem controle do fwer abaixo do nominal
dun <- duncan.test(y=raiz$diam, trt=raiz$gen, DFerror=glr, MSerror=qmr, alpha=0.05)
dun
                                                                                          #
#-------------------------------------------------------------------------------------------
# Student Newman Keuls SNK: controle da fwer próxima a do Tukey
snk <- SNK.test(y=raiz$diam, trt=raiz$gen, DFerror=glr, MSerror=qmr, alpha=0.05)
snk
                                                                                          #
#-------------------------------------------------------------------------------------------
# teste de Scheffe
sch <- scheffe.test(y=raiz$diam, trt=raiz$gen, DFerror=glr, MSerror=qmr,
                    Fc=anova(m0)["gen",4], alpha=0.05)
sch
                                                                                          #
#-------------------------------------------------------------------------------------------
# Tukey (velho conhecido e mais usado), HSD Honestly Significant Difference
tuk <- HSD.test(y=raiz$diam, trt=raiz$gen, DFerror=glr, MSerror=qmr, alpha=0.05)
tuk
                                                                                          #
#-------------------------------------------------------------------------------------------
# os testes acima são todos do pacote agricolae. Em todos, as estimativas de \mu_\alpha_i
# são as médias amostrais que, para esse delineamento, são estimadores admissíveis. Os
# erros padrões calculados são dentro de cada nível. Para representação deve se usar o erro
# padrão das médias de mínimos quadrados.
                                                                                          #
#-------------------------------------------------------------------------------------------
# fazendo os mesmos testes com as funções do pacote laercio
require(laercio)
LDuncan(a0, which="gen")
LTukey(a0, which="gen")
                                                                                          #
#-------------------------------------------------------------------------------------------
# fazendo os mesmos testes com as funções do pacote DTK (usa erros padrões amostrais!!!)
# DTK.test faz o teste de Tukey para níveis com diferente número de repetições
require(DTK)
dtk <- DTK.test(x=raiz$diam, f=raiz$gen, a=0.05) # compara usando erros padrões amostrais!!
dtk
DTK.plot(dtk)
abline(v=0)
dtk[[2]] <- dtk[[2]][order(dtk[[2]][,1]),] # ordenando pelo valor do contraste
DTK.plot(dtk)
abline(v=0)
                                                                                          #
#-------------------------------------------------------------------------------------------
# funções do pacote mutoss
require(mutoss)
snk <- snk(diam~gen, data=raiz, alpha=0.05)
snk
tuk <- tukey.wrapper(aov(diam~gen, data=raiz), alpha=0.05, factorC="gen")
tuk
mul <- multcomp.wrapper(aov(diam~gen, data=raiz), hypotheses="Tukey", alternative="two.sided", alpha=0.05, factorC="gen")
mul
```

```
mul <- multcomp.wrapper(aov(diam~gen, data=raiz), hypotheses="Williams", alternative="two.sided", alpha=0.05, factorC="gen")
mul
                                                                                                    #
#-------------------------------------------------------------------------------------------
# funções da multcomp
glhtcomp <- summary(glht(a0, linfct=mcp(gen="Tukey")), test=adjusted(type="none"))
plot(glhtcomp)
abline(v=0)
glhtcomp <- summary(glht(a0, linfct=mcp(gen="Tukey")), test=adjusted(type="fdr"))
plot(glhtcomp) # é o mesmo plot
abline(v=0)
                                                                                                    #
#-------------------------------------------------------------------------------------------
.
```

6.9 Aplicando o procedimento de Scott-Knott para agrupar médias

```
.
#-------------------------------------------------------------------------------------------
# **não** é teste para H0: \mu+alpha_i = \mu_\alpha_i' para todo i diferente de i'
# é um agrupamento de médias que tem a característica de eliminar sobreposição
                                                                                                    #
#-------------------------------------------------------------------------------------------
# ScottKnoot, também disponível do pacote laercio
require(ScottKnott)
                                                                                                    #
#-------------------------------------------------------------------------------------------
# aplicando o teste de ScootKnott
sk <- SK(x=raiz, y=raiz$diam, model="y~gen", which="gen", sig.level=0.05, id.trim=10)
summary(sk)
plot(sk) # barras ligam o mínimo ao máximo
                                                                                                    #
#-------------------------------------------------------------------------------------------
# usando a do laercio
LScottKnott(a0,"gen")
                                                                                                    #
#-------------------------------------------------------------------------------------------
.
```

6.10 Opções de representação gráfica

```
#-------------------------------------------------------------------------------------------
# gráfico de barras com letras sobre as barras
glr <- df.residual(a0); sqr <- deviance(a0); qmr <- sqr/glr
tuk <- HSD.test(y=raiz$diam, trt=raiz$gen, DFerror=glr, MSerror=qmr, alpha=0.05)
tuk
means <- tuk$means
names(means) <- tuk$trt
let <- gsub(" ", "", tuk$M) # letras da comparação, remove o espaço
barplot(means)
lim <- par()$usr   # limites cartesianos do último gráfico feito
alt <- strheight("teste") # altura de uma palavras em relação ao ultimo gráfico
bp <- barplot(means, ylim=lim[3:4]+c(0,3*alt))
text(bp, means, label=let, pos=3)
box()
                                                                                                    #
#-------------------------------------------------------------------------------------------
# gráfico de barras com letras e médias sobre as barras
let <- paste(format(means, digits=2), let, sep="\n")
let
barplot(means, ylim=c(0.3, lim[4]))
help(barplot, help_type="html")
lim <- par()$usr   # limites cartesianos do último gráfico feito
alt <- strheight("teste") # altura de uma palavras em relação ao ultimo gráfico
bp <- barplot(means, ylim=lim[3:4]+c(0,7*alt), xpd=FALSE)
text(bp, means, label=let, pos=3)
box()
                                                                                                    #
#-------------------------------------------------------------------------------------------
# gráfico com barras e IC
tab <- contrast(a0, list(gen=levels(raiz$gen)))
str(tab)
```

```
tab <- do.call(data.frame, tab[c(1,2,4,5)])
tab <- tab[order(tab$Contrast, decreasing=TRUE),]
means <- tab$Contrast; names(means) <- tab$gen
bp <- barplot(means, ylim=extendrange(tab[,3:4]), xpd=FALSE)
arrows(bp, tab$Lower, bp, tab$Upper, code=3, angle=90, length=0.1)
box()
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# gráfico com barras, médias, letras e ICs (para salvar, decomente uma das funções)
# pdf("myplot.pdf")
# png("myplot.png")
# tiff("myplot.tiff")
# svg("myplot.svg")
bp <- barplot(means, ylim=extendrange(tab[,3:4]), xpd=FALSE,
              xlab="Genótipo de sorgo", ylab="Diâmetro da radícula", xaxt="n")
arrows(bp, means, bp, tab$Upper, code=2, angle=90, length=0.1)
text(bp, means, label=let, pos=1)
mtext(side=1, line=0.5, at=bp, text=names(means), cex=0.9)
mtext("médias seguidas da mesma letra não diferem entre sí pelo teste de Tukey (5%)",
      side=3, line=0.5, font=3, cex=0.9)
mtext("intervalos de confiança de 95%", at=par()$usr[3], adj=0,
      side=4, line=0, font=3, cex=0.8)
box(bty="L")
# dev.off() # também decomente essa linha para salvar a figura
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# gráfico do pacote plotrix::plotCI()
require(plotrix)
with(tab, plotCI(x=as.numeric(gen), y=Contrast, li=Lower, ui=Upper, xaxt="n"))
axis(side=1, at=1:nrow(tab), labels=tab$gen)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# gráfico do pacote gplots::plotmeans(), não faz o que precisamos mas vale conhecer
require(gplots)
plotmeans(diam~gen, data=raiz)
plotmeans(diam~gen, data=raiz, mean.labels=TRUE, connect=FALSE, digits=3, pch=3)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# gráfico de barras
barplot2(means, ylim=extendrange(tab[,3:4]), xpd=FALSE,
         xlab="Genótipo de sorgo", ylab="Diâmetro da radícula",
         plot.ci=TRUE, ci.l=tab$Lower, ci.u=tab$Upper, plot.grid=TRUE)
box()
barplot2(means, xlim=extendrange(tab[,3:4]), xpd=FALSE, horiz=TRUE,
         ylab="Genótipo de sorgo", xlab="Diâmetro da radícula",
         plot.ci=TRUE, ci.l=tab$Lower, ci.u=tab$Upper, plot.grid=TRUE,
         density=5, angle=45)
box()
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# Hmisc::xYplot(), latticeExtra::segplot(), paych::error.bars(), psych::error.bars.by()
                                                                                                    #
#-------------------------------------------------------------------------------------------------

.
```

6.11 Análise usando a função `ExpDes::crd()`

```
#-------------------------------------------------------------------------------------------------
# carrega o pacote (versão em inglês), muda o nome das funções no português
# crd: completely randomized design > dic: delineamento inteiramente ao acaso
require(ExpDes)
help(package="ExpDes")
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# obtem todos os resultados de maneira fácil (só serve para caso balanceado)
# tem todas as opções de comparação múltipla, fazer análise de resíduos separado
with(raiz, crd(treat=gen, resp=diam, quali=TRUE, mcomp="lsd", sigT=0.05, sigF=0.05))
with(raiz, crd(treat=gen, resp=diam, quali=TRUE, mcomp="lsdb", sigT=0.05, sigF=0.05))
with(raiz, crd(treat=gen, resp=diam, quali=TRUE, mcomp="duncan", sigT=0.05, sigF=0.05))
with(raiz, crd(treat=gen, resp=diam, quali=TRUE, mcomp="snk", sigT=0.05, sigF=0.05))
with(raiz, crd(treat=gen, resp=diam, quali=TRUE, mcomp="tukey", sigT=0.05, sigF=0.05))
with(raiz, crd(treat=gen, resp=diam, quali=TRUE, mcomp="sk", sigT=0.05, sigF=0.05))
                                                                                                    #
#-------------------------------------------------------------------------------------------------
.
```

6.12 Análise com desigual número de repetições

```
#-------------------------------------------------------------------------------------
# causas do número desigual:
# *(1)duarante a amostragem um nível ocorre em menor frequência (estudo observacional)
# *(2)ue de um experimento balanceado foram perdidas por causas naturais e aleatórias
# *(2)ue de um experimento balanceado foram perdidas por causas sistemáticas
# em 1 e 2 só causam diferença de precisão nas estimativas, no 3 pode causar viés
                                                                                     #
#-------------------------------------------------------------------------------------
# experimento avaliando o substrato para desenvolvimento de mudas de cajuzeiro
verm <- read.csv("../dados/vermiculita.csv", header=TRUE, sep=";")
str(verm)
                                                                                     #
#-------------------------------------------------------------------------------------
# intensidade de desbalanceamento
table(verm$substr)
nrep <- with(verm, tapply(alt, substr, function(x) sum(!is.na(x))))
nrep
                                                                                     #
#-------------------------------------------------------------------------------------
# ajuste do modelo e quadro de análise de variância
a0 <- lm(alt~substr, data=verm)
par(mfrow=c(2,2)); plot(a0); layout(1)
anova(a0)
summary(a0) # informa os missings, note erros padrões diferentes
                                                                                     #
#-------------------------------------------------------------------------------------
# TukeyHSD usa estimadores de mínimos quadrados no teste de Tukey
detach(package:HH); detach(package:multcomp)
tk <- TukeyHSD(aov(alt~substr, data=verm))
plot(tk); abline(v=0)
str(tk)
tk$substr <- tk$substr[order(tk$substr[,1], decreasing=TRUE),]
plot(tk); abline(v=0) # veja que os IC tem amplitudes diferentes
                                                                                     #
#-------------------------------------------------------------------------------------
# usando a multcomp::glht()
require(multcomp)
tm <- summary(glht(a0, linfct=mcp(substr="Tukey")))
str(tm)
plot(tm); abline(v=0)
                                                                                     #
#-------------------------------------------------------------------------------------
# usando a HH::glht.mmc()
require(HH)
tm <- glht.mmc(a0, linfct=mcp(substr="Tukey"))
plot.matchMMC(tm$mca); abline(v=0)
plot(tm, ry.mmc=c(13.5, 18))
                                                                                     #
#-------------------------------------------------------------------------------------
# usando as funções da agricolae, vão usar a média harmônica do número de repetições
# aplicando o teste de médias usando uma dms *fixa*, não recomendado para desbal intenso
glr <- df.residual(a0) # extrai o grau de liberdade do resíduo
sqr <- deviance(a0)    # extrais a soma de quadrados dos resíduos "desviância"
qmr <- sqr/glr         # calcula o quadrado médio do resíduo
lsd <- LSD.test(verm$alt, verm$substr, glr, qmr, alpha=0.05,  p.adj="bonferroni")
lsd
length(nrep)/sum(1/nrep) # média harmônica
                                                                                     #
#-------------------------------------------------------------------------------------
# Duncan: é mais conservador que o t mais ainda tem controle do fwer abaixo do nominal
dun <- duncan.test(verm$alt, verm$substr, DFerror=glr, MSerror=qmr, alpha=0.05)
dun
                                                                                     #
#-------------------------------------------------------------------------------------
# Student Newman Keuls SNK: controle da fwer próxima a do Tukey
snk <- SNK.test(verm$alt, verm$substr, DFerror=glr, MSerror=qmr, alpha=0.05)
snk
                                                                                     #
#-------------------------------------------------------------------------------------
# Tukey (velho conhecido e mais usado), HSD Honestly Significant Difference
tuk <- HSD.test(verm$alt, verm$substr, DFerror=glr, MSerror=qmr, alpha=0.05)
tuk
                                                                                     #
#-------------------------------------------------------------------------------------
# você pode fazer os gráficos de barras para esse resultado também, mas cuide a relação
# "informação contida/espaço ocupado", ela deve ser máxima
```

```
                                                                                                    #
#-------------------------------------------------------------------------------------------------
.
```

6.13 Modelo de efeitos aleatórios e estimação dos componentes de variância

$$Y_{ij} = \mu + a_i + \epsilon_{ij} \qquad a_i \overset{iid}{\sim} \mathcal{N}(0, \sigma^2_a) \qquad \epsilon_{ij} \overset{iid}{\sim} \mathcal{N}(0, \sigma^2_\epsilon) \tag{3}$$

```
.
#-------------------------------------------------------------------------------------------------
# comum em estudos de melhoramento onde o efeito (decorrente da constituição genética) de
# um indivíduo é uma variável aleatória. É comum nesses casos a estimação da variância dos
# efeitos, ligadas às práticas de seleção, estimação de ganho por seleção, etc.
# hoje em dia pode-se estimar os componentes de variância por diversos métodos. Os mais
# comuns são o método anova (ou dos momentos) e o método da máxima verossimilhança.
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# o componente pelo método da anova consiste em igualar as esperanças dos quadrados médios
# aos valores calculados. Vamos supor, para exemplificação, que os genótipos do experimento
# com plântulas são de efeito aleatório
a0 <- lm(diam~gen, data=raiz)
an0 <- anova(a0)
s2e <- an0[2,"Mean Sq"] # ou deviance(a0)/df.residual(a0)
nrep <- nrow(raiz)/nlevels(raiz$ge); nrep
s2a <- (an0[1,"Mean Sq"]-s2e)/nrep # componente de variância
h2 <- s2a/(an0[1,"Mean Sq"]/nrep)  # herdabilidade
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# usando o pacote GAD (do irmão do meu Professor de guitarra e um Professor da UFPR)
require(GAD)
raiz$gen <- as.random(raiz$gen)
a0 <- lm(diam~gen, data=raiz)
help(gad, help_type="html")
gad(a0)
estimates(a0) # são as esperanças dos quadrados médios
C.test(a0)    # teste de Cochran para homogeneidade de variâncias (p-value>1??)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# método da máxima verossimilhança (ML)
require(nlme)
mm0 <- lme(diam~1, random=~1|gen, data=raiz, method="ML")
summary(mm0)
VarCorr(mm0) # componentes de variâcia
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# método REML, tem a vantagem de ter menor vício e coincidir com anova para balanceados
mm0 <- lme(diam~1, random=~1|gen, data=raiz, method="REML")
summary(mm0)
VarCorr(mm0) # componentes de variâcia
                                                                                                    #
#-------------------------------------------------------------------------------------------------
.
```

6.14 Função para casualização em DIC

```
.
#-------------------------------------------------------------------------------------------------
# fazendo a casualização dos níveis dos tratamentos às unidades experimentais numeradas
trt <- LETTERS[1:5] # 5 níveis
r <- 6 # repetições
crd <- design.crd(trt, r, kinds="Super-Duper")
str(crd)
write.table(crd, file="crd.txt", row.names=FALSE, sep="\t")
write.table(crd, file="crd.xls", row.names=FALSE, sep="\t")
                                                                                                    #
#-------------------------------------------------------------------------------------------------
.
```

7 Análise de experimentos de um fator em DBC

7.1 Entrada de dadoscom `scan()` e `edit()`

```
.
#-------------------------------------------------------------------------------------------
# produção de madeira (m3/ha) em função de procedência de E. grandis e blocos
# entrando com os dados pelo teclado
mad <- expand.grid(proced=c("P1","P2","P3","P4","P5","P6","P7"),
                   bloco=c("I","II","III","IV"))
mad # todos as parcelas do experimento
                                                                                          #
#-------------------------------------------------------------------------------------------
# você pode digitar os dados via scan()
aux <- scan() # digita cada valor no console, dê enter, para sair duplo enter
#mad$prod <- aux
ls()      # lista os objetos criados
rm("aux") # remove o objeto aux
                                                                                          #
#-------------------------------------------------------------------------------------------
# você pode digitar os dados com a edit(), essa depende do seu editor R
aux <- edit(mad)
str(aux)
mad <- aux
ls()      # lista os objetos criados
rm("aux") # remove o objeto aux
                                                                                          #
#-------------------------------------------------------------------------------------------
# você também pode digitar os dados em um vetor assim
mad$prod <- c(358,284,273,284,258,249,318,
              380,249,222,242,263,217,312,
              353,259,236,266,242,267,327,
              360,242,226,252,231,220,319)
str(mad)
                                                                                          #
#-------------------------------------------------------------------------------------------
# qual o melhor jeito? o vetor! porque o script se torna reproduzível/portável
paste(mad$prod, collapse=", ") # cria uma string de valores separados por vírgula
dput(mad) # retorna a estrutura do objeto, copie do console e deixe colado no script
                                                                                          #
#-------------------------------------------------------------------------------------------
# vendo os dados no formato amplo, dupla entrada
with(mad, tapply(prod, list(proced, bloco), identity))
                                                                                          #
#-------------------------------------------------------------------------------------------
# gráficos, vamos verificar a aditividade
require(lattice)
xyplot(prod~proced, groups=bloco, data=mad, type="b")
xyplot(prod~bloco, groups=proced, data=mad, type="b")
                                                                                          #
#-------------------------------------------------------------------------------------------
.
```

7.2 Modelo estatístico e análise de variância

$$Y_{ij} = \mu + \iota_i + \alpha_j + \epsilon_{ij} \qquad \epsilon_{ij} \overset{iid}{\sim} \mathcal{N}(0, \sigma^2_\epsilon) \tag{4}$$

$$\boldsymbol{Y} = [\mathbf{1}_\mu | \boldsymbol{X}_\iota | \boldsymbol{X}_\alpha] \begin{bmatrix} \mu \\ \boldsymbol{\iota} \\ \boldsymbol{\alpha} \end{bmatrix} + \boldsymbol{\epsilon} \tag{5}$$

```
.
#-------------------------------------------------------------------------------------------
# ajuste do modelo
m0 <- lm(prod~bloco+proced, data=mad) # ou aov(...)
class(m0)       # classe do objeto, define os métodos que são aplicaveis
anova(m0)       # análise de variância
summary(m0)     # quadro de estimativas dos efeitos/parâmetros (sob a particular restrição)
contrasts(mad$bloco)
contrasts(mad$proced)
                                                                                          #
```

```
#-------------------------------------------------------------------------------------------
# checagem gráfica das presuposições da análise
par(mfrow=c(2,2))
plot(m0)
layout(1)
                                                                                           #
#-------------------------------------------------------------------------------------------
# teste das pressuposições de normalidade de homocedasticidade
shapiro.test(mad$prod)      # teste de normalidade nos dados ERRADO, contém efeitos
shapiro.test(residuals(m0)) # teste de normalidade nos resíduos CERTO, efeitos removidos
bartlett.test(residuals(m0)~mad$proced)  # homogeneidade em função dos níveis de proced
bartlett.test(residuals(m0)~mad$bloco)   # homogeneidade em função dos níveis de bloco
require(car)
levene.test(residuals(m0)~mad$proced)    # pelo teste de levene
# nestes testes, o grau de liberdade está sendo maior que o real. Nos testes para homog em
# função de proced, o grau de liberdade foi 21 e não 18. Estimamos 1+6+3 parâmetros.
                                                                                           #
#-------------------------------------------------------------------------------------------
# teste da aditividade, é muito particular desse delineamento, mostro por completitividade
require(agricolae)
nonadditivity(y=mad$prod, factor1=mad$bloc, factor2=mad$proced,
              df=df.residual(m0), MSerror=deviance(m0)/df.residual(m0))
                                                                                           #
#-------------------------------------------------------------------------------------------
# nos links abaixo está a função que alguém programou e disponibilizou
tukey.add.test.url <- "http://www.stat.sc.edu/curricula/grad/qualdata/Rmacros.txt"
browseURL(URLencode(tukey.add.test.url))
browseURL(URLencode("http://www.stat.sc.edu/~hitchcock/executive_RCBD_Rexample705.txt"))
                                                                                           #
#-------------------------------------------------------------------------------------------
# podemos carregar qualquer função contida em um script para uso em nossa sessão
source(tukey.add.test.url)
tukeys.add.test(y=mad$prod, A=mad$proced, B=mad$bloco)
                                                                                           #
#-------------------------------------------------------------------------------------------
# essa função já está disponível em um pacote
browseURL(URLencode("http://rgm2.lab.nig.ac.jp/RGM2/func.php?rd_id=asbio:tukey.add.test"))
                                                                                           #
#-------------------------------------------------------------------------------------------
.
```

7.3 Testes de médias

```
.
#-------------------------------------------------------------------------------------------
# podem ser usados todos os testes de médias vistos na sessão anterior
# o teste você definir no planejamento do experimento!
require(agricolae)
glr <- df.residual(m0)
qmr <- deviance(m0)/glr
with(mad, LSD.test(prod, proced, glr, qmr))
with(mad, duncan.test(prod, proced, glr, qmr))
with(mad, SNK.test(prod, proced, glr, qmr))
with(mad, HSD.test(prod, proced, glr, qmr))
with(mad, LSD.test(prod, proced, glr, qmr, p.adj="bonferroni"))
                                                                                           #
#-------------------------------------------------------------------------------------------
# teste de Scott-Knott
require(ScottKnott)
sk <- SK(x=mad, y=mad$prod, model="prod~bloco+proced", which="proced")
summary(sk)
plot(sk)
                                                                                           #
#-------------------------------------------------------------------------------------------
# lembre-se: TukeyHSD() é um método para objetos de classe aov
tk <- TukeyHSD(aov(prod~bloco+proced, data=mad), which="proced")
tk       # estimativas dos contrates acompanhados de IC (de mesmo tamanho)
plot(tk) # gráfico desse resultado tabular
tk$proced <- tk$proced[order(tk$proced[,1]),] # ordena pelo valor do contraste
plot(tk); abline(v=0)
oldpar <- par()  # armazena os parâmetros gráficos
par(las=1)       # altera para rotacionar os rótulos do eixo y
plot(tk); abline(v=0)
par <- oldpar    # devolve as configurações iniciais dos parâmetros gráficos
                                                                                           #
#-------------------------------------------------------------------------------------------
```

```
# usando a multcomp
require(multcomp)
tk2 <- summary(glht(m0, linfct=mcp(proced="Tukey")))
tk2
plot(tk2)
cld(tk2) # compact letter display (cuidado com dados desbalanceados)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# usando a HH
require(HH)
tk3 <- glht.mmc(m0, linfct=mcp(proced="Tukey"))
plot(tk3)
oldpar <- par() # armazena os parâmetros gráficos
par()$mar       # valores atuais para as margens
par(mar=c(5.1,6.1,4.1,5.1)) # aumental as margens esquerda e direita
plot(tk3)
par <- oldpar   # devolve as configurações iniciais dos parâmetros gráficos
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# stats::TukeyHSD(), multcomp::glht() e HH::glht.mmc só darão mesma estimativa para o
# contraste nos casos balanceados. O p-valor será diferente porque envolve critérios de
# correção diferentes.
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# você pode usar os gráficos que vimos na sessão anterior para apresentar os resultados
                                                                                                    #
#-------------------------------------------------------------------------------------------------
.
```

7.4 Análise usando a função `ExpDes::rbd()`

```
.
#-------------------------------------------------------------------------------------------------
# usando a ExpDes, rbd: randomized block design, dbc: delineamentos em blocos casualizados
# análise dos resíduos deve ser feita separado
require(ExpDes)
with(mad, rbd(proced, bloco, prod, quali=TRUE, mcomp="lsd"))
with(mad, rbd(proced, bloco, prod, quali=TRUE, mcomp="lsdb"))
with(mad, rbd(proced, bloco, prod, quali=TRUE, mcomp="duncan"))
with(mad, rbd(proced, bloco, prod, quali=TRUE, mcomp="snk"))
with(mad, rbd(proced, bloco, prod, quali=TRUE, mcomp="tukey"))
with(mad, rbd(proced, bloco, prod, quali=TRUE, mcomp="sk"))
                                                                                                    #
#-------------------------------------------------------------------------------------------------
.
```

7.5 Análise com observações perdidas

```
.
#-------------------------------------------------------------------------------------------------
# causa perda da ortogonalidade e diferente precisão às estimativas
# as médias amostrais não são mais estimadores admissíveis das médias populacionais
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# vamos conderar um exemplo simulado para entender o procedimento
# 5 níveis do fator tratamento, 4 níveis do fator bloco
sim <- expand.grid(blc=gl(4,1, labels="bl"), trt=gl(5,1, labels="tr"))
str(sim)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# abaixo o vetor de parâmetros que vou usar para gerar observações sua matriz de incidência
betas <- c(10, 0.25,0.50,0.75, 1,2,3,4) # mu | iota | alpha
X <- model.matrix(~blc+trt, data=sim)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# o valor esperado a ser observado seria esse se não houvesse fontes de variação não contrl
sim$yesp <- X%*%betas
xyplot(yesp~trt, groups=blc, data=sim, type="o") # parelelismo = não interação = aditividade
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# mas o fenômeno é parte determinístico, parte aleatório, então observo realizações de v.a.
sim$yobs <- sim$yesp+rnorm(nrow(sim), sd=0.1)
```

```
xyplot(yobs~trt, groups=blc, data=sim, type="o")
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# agora suponha que um fenômeno aleatório tenha compometido a observação de 2 unidades exp
missings <- sample(1:nrow(sim), 2); missings # unidades exp não observadas
xyplot(yobs~trt, groups=blc, data=sim[-missings,], type="o")
sim$yobs[missings] <- NA
with(sim, tapply(yobs, list(trt, blc), identity))
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# ajuste do modelo e análise de variância
m0 <- lm(yobs~blc+trt, data=sim)
summary(m0) # todos os efeitos são estimáveis, porém com precisões diferentes
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# em uma anova sequencia, quando não ortogonalidade, ordem dos termos importa
anova(lm(yobs~blc+trt, data=sim)) # SQ(mu), SQ(blc|mu), SQ(trt|blc,mu)
anova(lm(yobs~trt+blc, data=sim)) # SQ(mu), SQ(trt|mu), SQ(blc|trt,mu)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# como estimar as médias (\mu+\alpha_j) dos níveis dos tratamentos?
# o procedimento é matricial mas podemos fazer usando a contrast::contrast()
require(contrast)
levels(sim$trt)
# média do níveis dentro do bl1
c0 <- contrast(m0, list(blc="bl1", trt=c("tr1","tr2","tr3","tr4","tr5")))
c0
c0$X
# é usual obter a média dos níveis ocorrendo com mesma frequência em todos os blocos, assim
c0 <- contrast(m0, type="average", list(blc=c("bl1","bl2","bl3","bl4"), trt=c("tr1")))
str(c0)
c0$X # valor obtido na média dos blocos
allc0 <- lapply(levels(sim$trt),
                function(trt){
                  c0 <- contrast(m0, type="average", list(blc=c("bl1","bl2","bl3","bl4"), trt=trt))
                  c0[1:7]
                })
str(allc0)
mmq <- do.call(rbind, allc0) # erros padrões diferentes
mmq
mm <- unlist(mmq[,1])
mm # médias matriciais
dmm <- outer(mm, mm, "-"); dmm[upper.tri(dmm)] <- NA
dmm
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# veja as médias amostrais
ma <- with(sim, tapply(yobs, trt, mean, na.rm=TRUE))
ma # médias amostrais
dma <- outer(ma, ma, "-"); dma[upper.tri(dma)] <- NA
dma # diferenças entre médias amostrais
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# para comparar médias você não pode usar as agricolae::***.test() por que usam m amostrais
# o mesmo vale para ScottKnott::SK, você terá que usar a contrast e multcomp
detach(package:HH); detach(package:multcomp)
tk <- TukeyHSD(aov(yobs~blc+trt, data=sim), which="trt")
plot(tk)
tk$trt
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# usando a multcomp
require(multcomp)
tk2 <- summary(glht(m0, linfct=mcp(trt="Tukey")))
tk2 # tk2 bate com dmm, é o teste de hipótese para médias do tipo \mu_\alpha_i
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# alguns aplicativos chamam de \mu_\alpha_i de médias ajustadas. O estimador de mínimos
# quadrados é admissível em quase todas situações. O estimador amostral só é valido para o
# caso de efeitos/delineamentos ortogonais.
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# ainda existem na literatura os tipo de somas de quadrados. No R você obtém com
require(car)
anova(m0)                       # sequêncial
Anova(m0, type="II")            # semimarginal
Anova(m0, type="III")           # marginal
drop1(m0, scope=.~., test="F")  # marginal
                                                                                                    #
```

```
#-------------------------------------------------------------------------------------
.
```

7.6 Blocos de efeito aleatório

```
.
#-------------------------------------------------------------------------------------
# às vezes o efeito dos blocos é assumido como aletório, normalmente quando eles
# representam uma amostra de uma população de blocos que poderiam ter sido usados no
# experimento. Podemos estimar os componentes de variância via anova e ML
                                                                                     #
#-------------------------------------------------------------------------------------
# estimação via anova
require(GAD)
mad$bloco <- as.random(mad$bloco)
mad$proced <- as.fixed(mad$proced)
m0 <- lm(prod~bloco+proced, data=mad)
gad(m0)
estimates(m0)
an0 <- anova(m0)
s2b <- (an0["bloco","Mean Sq"]-an0["Residuals","Mean Sq"])/nlevels(mad$proced)
s2b
                                                                                     #
#-------------------------------------------------------------------------------------
# estimação via ML
require(nlme)
mm0 <- lme(prod~proced, random=~1|bloco, data=mad, method="ML")
summary(mm0)
VarCorr(mm0)
                                                                                     #
#-------------------------------------------------------------------------------------
# estimação via REML
mm0 <- lme(prod~proced, random=~1|bloco, data=mad, method="REML")
summary(mm0)
VarCorr(mm0)
                                                                                     #
#-------------------------------------------------------------------------------------
# qual a diferença prática entre as abordagens fixa e aleatória? nível de inferência
diag(vcov(m0))[c(1,5:10)]
diag(vcov(mm0))
                                                                                     #
#-------------------------------------------------------------------------------------
.
```

7.7 Blocos de efeito aleatório quando houve perda de observações

```
.
#-------------------------------------------------------------------------------------
# com a GAD não dá certo porque só opera em experimentos balanceados
sim$blc <- as.random(sim$blc)
sim$trt <- as.fixed(sim$trt)
m0 <- lm(yobs~blc+trt, data=sim)
anova(m0)
gad(m0)
estimates(m0)
                                                                                     #
#-------------------------------------------------------------------------------------
# usando REML (atenção: eu não gerei os dados com efeito de bloco aleatório, mas poderia)
mm0 <- lme(yobs~trt, random=~1|blc, data=sim, na.action=na.omit, method="REML")
summary(mm0)
VarCorr(mm0)
                                                                                     #
#-------------------------------------------------------------------------------------
# como ficam as estimativas das médias \mu+\alpha_i?
contrast(mm0, list(trt=levels(sim$trt))) # diferem um pouco das obtidas com efeitos fixos
ranef(mm0)                               # são as predições dos efeitos dos blocos
                                                                                     #
#-------------------------------------------------------------------------------------
.
```

7.8 Função para casualização em DBC

```
.
#-------------------------------------------------------------------------------------
# fazendo a casualização dos níveis dos tratamentos às unidades experimentais numeradas
require(agricolae)
trt <- LETTERS[1:5] # 5 níveis
b <- 4 # blocos
rcbd <- design.rcbd(trt, r=b, kinds="Super-Duper")
str(rcbd)
matrix(rcbd$trt, ncol=b)
write.table(rcbd, file="rcbd.txt", row.names=FALSE, sep="\t")
write.table(rcbd, file="rcbd.xls", row.names=FALSE, sep="\t")
                                                                                    #
#-------------------------------------------------------------------------------------
.
```

8 Análise de experimentos em blocos incompletos balanceados (BIB)

8.1 Descrição dos dados

```
#-------------------------------------------------------------------------------------
# delineamento comum em melhoramento e análise sensorial de alimentos
# dados de um experimento em blocos incompletos tipo I (Pimentel Gomes p.185)
# -- em cada rept cada trat ocorre 1 vez --
# rept: conjunto de unidades experimentais com mesmo número repetições de cada nível
# bloc: possuem menos ue que os níveis de tratamento, logo um bloco não contém todos níveis
# trat: são o níveis do fator de interesse
# resp: resposta observada
bib1 <- c(20,1,18,2,15,3,16,4,14,5,15,6,16,7,18,8,
          24,1,18,3,25,2,19,8,13,4,16,5,12,6,16,7,
          23,1,17,4,26,2,18,7,15,3,17,6,13,5,16,8,
          21,1,13,5,23,2,16,3,10,4,12,7,13,6,11,8,
          28,1,14,6,27,2,18,4,18,3,15,8,16,5,17,7,
          22,1,17,7,24,2,16,6,18,3,14,5,15,4,17,8,
          23,1,15,8,21,2,13,5,15,3,12,7,13,4,16,6)
bib1 <- matrix(bib1, ncol=2, byrow=TRUE)
bib1 <- as.data.frame(bib1)
bib1 <- cbind(rept=gl(7,8), bib1)
bib1$bloc <- gl(4,2)
names(bib1) <- c("rept","resp","trat","bloc")
bib1$trat <- factor(bib1$trat)
str(bib1)
with(bib1, table(trat))             # número de parcelas com cada tratamento
with(bib1, table(rept, bloc))       # número de blocos por repetição
with(bib1, table(bloc, trat, rept)) # sempre 2 tratamentos ocorrem juntos
with(bib1, table(rept:bloc, trat))  # cada tratamento ocorre uma vez com outro
# o necessário para haver estimabilidade é conectividade
                                                                                    #
#-------------------------------------------------------------------------------------
# análise gráfica
require(lattice)
xyplot(resp~trat|rept, groups=bloc, data=bib1, type="b")
                                                                                    #
#-------------------------------------------------------------------------------------
.
```

8.2 Análise intra bloco de BIB tipo I

$$Y_{ij} = \mu + \kappa_i + \iota_{i(j)} + \alpha_k + \epsilon_{ijk} \qquad \epsilon_{ijk} \overset{iid}{\sim} \mathcal{N}(0, \sigma^2_\epsilon) \tag{6}$$

$$\boldsymbol{Y} = [\mathbf{1}_\mu | \boldsymbol{X}_\kappa | \boldsymbol{X}_\iota | \boldsymbol{X}_\alpha] \begin{bmatrix} \mu \\ \boldsymbol{\kappa} \\ \boldsymbol{\iota} \\ \boldsymbol{\alpha} \end{bmatrix} + \boldsymbol{\epsilon} \tag{7}$$

```
#-------------------------------------------------------------------------------------
# análise intrabloco considera que o efeito de bloco dentro de repetição é fixo
# ajuste do modelo, anova sequencial e marginal, só o teste F para trat é válido
m0 <- lm(terms(resp~rept/bloc+trat, keep.order=TRUE), data=bib1,
         contrast=list(rept=contr.sum, bloc=contr.sum, trat=contr.sum))
par(mfrow=c(2,2)); plot(m0); layout(1) # checagem dos pressupostos
anova(m0)
car::Anova(m0, type="III")
drop1(m0, scope=.~., test="F")
                                                                                     #
#-------------------------------------------------------------------------------------
# usando o terms() declaramos a sequência das fontes de variação no quadro de anova
# usamos contraste soma zero para fazer com que os feitos de rept e rept/bloc se anulem
# isso facilita o cálculo das médias \mu+\alpha_i
# não tem como usar a contrast() por causa do operador "/" que declara o aninhamento
                                                                                     #
#-------------------------------------------------------------------------------------
# médias \mu+\alpha_i pelo procedimento é matricial
Xtrat <- m0$contrast$trat # matriz de contraste (soma zero)
assi <- m0$assign         # matriz com índices dos termos [0=\mu, 1=\, 2=\iota, 3=\alpha]
mg <- coef(m0)[1]         # contraste soma zero e balanceamento, (Intercept) é média geral
mmq <- c(Xtrat%*%coef(m0)[assi==3]+mg)
mmq # são as médias de mínimos quadrados (médias ajustadas/lsmeans)
                                                                                     #
#-------------------------------------------------------------------------------------
# contraste entre dois tratamentos: tr1 vs tr2
contr <- rep(0, length(coef(m0)))
contr[m0$assign==3] <- c(1,-1,0,0,0,0,0) # t1 vs t2
contr%*%coef(m0) # estimativa do contraste
#-------------------------------------------------------------------------------------
# variância do contraste, como é balanceado, todos os contrastes terão mesma variância
v.dif <- contr%*%vcov(m0)%*%contr; v.dif
#-------------------------------------------------------------------------------------
# diferença mínima significativa pelo teste de Tukey
delta <- qtukey(0.95, nlevels(bib1$trat), df=df.residual(m0))*sqrt(0.5*v.dif); delta
#-------------------------------------------------------------------------------------
# aplicação do teste de Tukey (usando a order::stat da agricolae)
tk <- agricolae::order.stat(levels(bib1$trat), mmq, delta)
tk
#-------------------------------------------------------------------------------------
# a SK não usa as médias de mínimos quadrados, usa as amostrais que são viesadas (não usar)
require(ScottKnott)
sk <- SK(x=bib1, y=bib1$resp, model="y~rept/bloc+trat", which="trat")
summary(sk)
sort(with(bib1, tapply(resp, trat, mean)))
                                                                                     #
#-------------------------------------------------------------------------------------
# gráfico de barras com as médias
tk2 <- tk$means
names(tk2) <- tk$trt
bp <- barplot(tk2, xlab="Tratamentos", ylab="Variável resposta",
              ylim=c(0, 1.2*max(tk2)))
text(bp, tk2,
     labels=paste(format(tk$means, dig=3), gsub(" ","",tk$M), sep="\n"), pos=3)
mtext(3, line=2,
      text="Comparação múltipla de médias", cex=1.5)
mtext(3, line=0.5,
      text="Médias seguidas de mesma letra não diferem entre si pelo teste de Tukey (5%)")
box()
                                                                                     #
#-------------------------------------------------------------------------------------
# calculando as médias \mu_\alpha_i com a gmodels::estimable() (valido para contr.sum)
require(gmodels)
mc <- matrix(0, ncol=sum(m0$assign<3), nrow=nlevels(bib1$trat)) # matriz com zeros
mc <- cbind(mc, m0$contrast$trat) # cola ao lado a porção dos tratamentos
mc[,1] <- 1                       # adiciona 1 na coluna no (Intercept)
estimable(m0, mc)                 # coloca a estimable para trabalhar, fácil
                                                                                     #
#-------------------------------------------------------------------------------------
# todos os contrastes possíveis entre os níveis de tratamento
nl <- nlevels(bib1$trat) # número de níveis
nc <- t(combn(8, 2))     # comparações
mc <- sapply(1:nrow(nc), function(x){ aux <- rep(0, nl); aux[nc[x,]] <- c(1,-1); aux })
mc <- t(mc)              # matriz de todos os contrastes 2 à 2
restr <- cbind(1, m0$contrast$trat)
contr <- mc%*%restr
contr[,-1]               # são as colunas dos contrastes entre os níveis de tratamentos
sum(m0$assign<3)
contr.all <- cbind(matrix(0, nrow=nrow(mc), ncol=sum(m0$assign<3)), contr[,-1])
```

```
rownames(contr.all) <- apply(nc, 1, paste, collapse="-")
str(contr.all)
                                                                                                    #
#-------------------------------------------------------------------------------------------
# calculando os contrastes com a gmodels::estimable() (válido para contr.sum)
estimable(m0, rbind("1-2"=contr.all[1,]))
estimable(m0, contr.all) # erros padrões iguais devido ao balanceamento/equilíbrio
fit.contrast(m0, "trat", mc[1,])
apply(mc, 1, FUN=fit.contrast, model=m0, varname="trat") # usando a apply()+fit.contrast()
                                                                                                    #
#-------------------------------------------------------------------------------------------
# usando a multcomp
require(multcomp)
glht0 <- summary(glht(m0, linfct=contr.all), test=adjusted(type="fdr"))
glht0
plot(glht0)
                                                                                                    #
#-------------------------------------------------------------------------------------------
# usando a HH
require(HH)
mmc <- glht.mmc(m0, linfct=contr.all, focus="trat")
plot(mmc)
                                                                                                    #
#-------------------------------------------------------------------------------------------
.
```

8.3 Análise inter bloco de BIB tipo I

$$Y_{ij} = \mu + \kappa_i + \iota_{i(j)} + \alpha_k + \epsilon_{ijk} \qquad \iota_{i(j)} \overset{iid}{\sim} \mathcal{N}(0, \sigma^2_\iota) \qquad \epsilon_{ijk} \overset{iid}{\sim} \mathcal{N}(0, \sigma^2_\epsilon) \tag{8}$$

$$\boldsymbol{Y} = [\mathbf{1}_\mu | \boldsymbol{X}_\kappa | \boldsymbol{X}_\iota | \boldsymbol{X}_\alpha] \begin{bmatrix} \mu \\ \boldsymbol{\kappa} \\ \boldsymbol{\iota} \\ \boldsymbol{\alpha} \end{bmatrix} + \boldsymbol{\epsilon} \tag{9}$$

```
#-------------------------------------------------------------------------------------------
# criar o fator bloc dentro de rept para associar efeito aleatório
str(bib1)
bib1$bloc.rept <- with(bib1, interaction(bloc, rept))
str(bib1)
                                                                                                    #
#-------------------------------------------------------------------------------------------
# ajuste do modelo e teste de Wald sequencial para os efeitos fixos
require(nlme)
mm0 <- lme(resp~rept+trat, random=~1|bloc.rept, data=bib1,
           contrast=list(rept=contr.sum, trat=contr.sum))
anova(mm0) # é um teste de Wald, o Fc não vem da divisão de QM
anova(mm0, type="marginal")
                                                                                                    #
#-------------------------------------------------------------------------------------------
# checagem da normalidade efeitos/erros aleatórios
par(mfrow=c(1,2))
qqnorm(residuals(mm0), main="Resíduos"); qqline(residuals(mm0))
qqnorm(unlist(ranef(mm0)), main="Blocos"); qqline(unlist(ranef(mm0)))
layout(1)
                                                                                                    #
#-------------------------------------------------------------------------------------------
# estimativas dos componentes de variância
VarCorr(mm0)
                                                                                                    #
#-------------------------------------------------------------------------------------------
# médias ajustadas
Xtrat <- cbind(1, mm0$contrast$trat)
assign <- lapply(list(rept="rept", trat="trat"), function(x){ grep(x, names(fixef(mm0))) })
assign
mmq <- c(Xtrat%*%fixef(mm0)[c(1,assi$trat)])
mmq # médias de mínimos quadrados
                                                                                                    #
#-------------------------------------------------------------------------------------------
# vetor de coeficientes do contraste tr1 vs tr2 e sua estimativa
contr <- rep(0, length(fixef(mm0))); contr
```

```
contr[assign$trat] <- c(1,-1,0,0,0,0,0)
contr%*%fixef(mm0)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# variância do contraste, como tem equilíbrio, todos os contrastes tem mesma variância
v.dif <- contr%*%vcov(mm0)%*%contr; v.dif
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# diferença mínima significativa pelo teste de Tukey
delta <- qtukey(0.95, nlevels(bib1$trat), df=anova(mm0)["trat","denDF"])*sqrt(0.5*v.dif)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# aplicação do teste de Tukey
agricolae::order.stat(levels(bib1$trat), mmq, delta) # veja, inverteu-se os 2 melhores
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# usando a gmodels::estimable()
mc <- mm0$contrast$trat
mc <- cbind(matrix(0, ncol=length(assign$rept)+1, nrow=nrow(mc)), mc)
mc[,1] <- 1
mc
estimable(mm0, mc) # médias de mínimos quadrados
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# todos os contrastes possíveis entre os níveis de tratamento
nl <- nlevels(bib1$trat) # número de níveis
nc <- t(combn(nl, 2))    # comparações
mc <- sapply(1:nrow(nc), function(x){ aux <- rep(0, nl); aux[nc[x,]] <- c(1,-1); aux })
mc <- t(mc)              # matriz de todos os contrastes 2 à 2
restr <- cbind(1, mm0$contrast$trat)
contr <- mc%*%restr
contr[,-1]               # são as colunas dos contrastes entre os níveis de tratamentos
contr.all <- cbind(matrix(0, nrow=nrow(mc), ncol=length(assign$rept)+1), contr[,-1])
rownames(contr.all) <- apply(nc, 1, paste, collapse="-")
str(contr.all)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# usando a multcomp
require(multcomp)
glht0 <- summary(glht(mm0, linfct=contr.all), test=adjusted(type="fdr"))
glht0
plot(glht0)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# usando a HH
require(HH)
mmc <- glht.mmc(mm0, linfct=contr.all, focus="trat") # método não aplicável para lme
plot(mmc)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
.
```

8.4 Descrição dos dados

```
#-----------------------------------------------------------------------------------------------
# dados de um experimento em blocos incompletos tipo II (Pimentel Gomes p.188)
# -- em cada grup cada trat ocorre 2 vezes (mero detalhe) --
# grup: grupo de unidades experimentais em que 1 trat ocorre duas vezes
# bloc: agrupa 2 níveis de tratamento
# trat: níveis dos tratamentos
# resp: resposta observada
bib2 <- c(35,1, 28,2, 32,2, 37,3, 35,3, 25,4, 28,4, 27,5, 30,5, 32,6, 24,6, 26,7,
          31,7, 27,1, 38,1, 40,3, 36,3, 27,5, 23,5, 30,7, 28,7, 25,2, 26,2, 28,4,
          23,4, 24,6, 28,6, 33,1, 30,1, 22,4, 27,4, 34,7, 32,7, 39,3, 33,3, 24,6,
          28,6, 34,2, 29,2, 26,5, 23,5, 33,1)
bib2 <- matrix(bib2, ncol=2, byrow=TRUE)
bib2 <- as.data.frame(bib2)
bib2$grup <- gl(3,14)
bib2$bloc <- gl(7,2)
names(bib2)[1:2] <- c("resp","trat")
bib2 <- transform(bib2, trat=factor(trat), resp=resp/10)
str(bib2)
with(bib2, table(trat))             # número de parcelas com cada tratamento
with(bib2, table(grup, bloc))       # número de blocos por repetição
with(bib2, table(bloc, trat, grup)) # sempre 2 tratamentos ocorrem juntos
```

```
with(bib2, table(grup:bloc, trat))  # cada tratamento ocorre uma vez com outro
                                                                                    #
#-------------------------------------------------------------------------------------
# gráficos
xyplot(resp~trat, data=bib2)
xyplot(resp~trat|grup, groups=bloc, data=bib2, type="o")
                                                                                    #
#-------------------------------------------------------------------------------------
.
```

8.5 Análise intra bloco de BIB tipo II

$$Y_{ij} = \mu + \kappa_i + \iota_{i(j)} + \alpha_k + \epsilon_{ijk} \qquad \epsilon_{ijk} \overset{iid}{\sim} \mathcal{N}(0, \sigma^2_\epsilon) \tag{10}$$

$$\boldsymbol{Y} = [\mathbf{1}_\mu | \boldsymbol{X}_\kappa | \boldsymbol{X}_\iota | \boldsymbol{X}_\alpha] \begin{bmatrix} \mu \\ \boldsymbol{\kappa} \\ \boldsymbol{\iota} \\ \boldsymbol{\alpha} \end{bmatrix} + \boldsymbol{\epsilon} \tag{11}$$

```
.
#-------------------------------------------------------------------------------------
# ajuste do modelo, anova sequencial e marginal
m0 <- lm(terms(resp~grup/bloc+trat, keep.order=TRUE), data=bib2,
         contrast=list(grup=contr.sum, bloc=contr.sum, trat=contr.sum))
par(mfrow=c(2,2)); plot(m0); layout(1)
anova(m0)
car::Anova(m0, type="III")
drop1(m0, scope=.~., test="F")
                                                                                    #
#-------------------------------------------------------------------------------------
# médias \mu_\alpha_i
Xtrat <- m0$contrast$trat
assi <- m0$assign
mmq <- c(Xtrat%*%coef(m0)[assi==3]+coef(m0)[1])
mmq
                                                                                    #
#-------------------------------------------------------------------------------------
# vetor de coeficientes do contraste tr1 vs tr2 e sua estimativa
contr <- rep(0, length(coef(m0)))
contr[assi==3] <- c(1,-1,0,0,0,0)
contr%*%coef(m0)
                                                                                    #
#-------------------------------------------------------------------------------------
# variância do contraste, como tem equilíbrio, todos os contrastes tem mesma variância
v.dif <- contr%*%vcov(m0)%*%contr; v.dif
                                                                                    #
#-------------------------------------------------------------------------------------
# diferença mínima significativa pelo teste de Tukey
delta <- qtukey(0.95, nlevels(bib2$trat), df=df.residual(m0))*sqrt(0.5*v.dif); delta
                                                                                    #
#-------------------------------------------------------------------------------------
# aplicação do teste de Tukey
tk <- agricolae::order.stat(levels(bib2$trat), mmq, delta)
tk
                                                                                    #
#-------------------------------------------------------------------------------------
# obtendo as médias com a gmodels::estimable()
mc <- matrix(0, ncol=sum(m0$assign<3), nrow=nlevels(bib2$trat)) # matriz com zeros
mc <- cbind(mc, m0$contrast$trat) # cola ao lado a porção dos tratamentos
mc[,1] <- 1                       # adiciona 1 na coluna no (Intercept)
estimable(m0, mc)                 # coloca a estimable para trabalhar, fácil
                                                                                    #
#-------------------------------------------------------------------------------------
# todos os contrastes possíveis entre os níveis de tratamento
nl <- nlevels(bib2$trat) # número de níveis
nc <- t(combn(nl, 2))    # comparações
mc <- sapply(1:nrow(nc), function(x){ aux <- rep(0, nl); aux[nc[x,]] <- c(1,-1); aux })
mc <- t(mc)              # matriz de todos os contrastes 2 à 2
restr <- cbind(1, m0$contrast$trat)
contr <- mc%*%restr
contr[,-1]               # são as colunas dos contrastes entre os níveis de tratamentos
sum(m0$assign<3)
```

```
contr.all <- cbind(matrix(0, nrow=nrow(mc), ncol=sum(m0$assign<3)), contr[,-1])
rownames(contr.all) <- apply(nc, 1, paste, collapse="-")
str(contr.all)
                                                                                                    #
#-------------------------------------------------------------------------------------------
# calculando os contrastes com a gmodels::estimable() (válido para contr.sum)
estimable(m0, rbind("1-2"=contr.all[1,]))
estimable(m0, contr.all) # erros padrões iguais devido ao balanceamento/equilíbrio
fit.contrast(m0, "trat", mc[1,])
apply(mc, 1, FUN=fit.contrast, model=m0, varname="trat") # usando a apply()+fit.contrast()
                                                                                                    #
#-------------------------------------------------------------------------------------------
# usando a multcomp
require(multcomp)
glht0 <- summary(glht(m0, linfct=contr.all), test=adjusted(type="fdr"))
glht0
plot(glht0)
                                                                                                    #
#-------------------------------------------------------------------------------------------
# usando a HH
require(HH)
mmc <- glht.mmc(m0, linfct=contr.all, focus="trat")
plot(mmc)
                                                                                                    #
#-------------------------------------------------------------------------------------------
.
```

8.6 Análise inter bloco de BIB tipo II

$$Y_{ij} = \mu + \kappa_i + \iota_{i(j)} + \alpha_k + \epsilon_{ijk} \qquad \iota_{i(j)} \overset{iid}{\sim} \mathcal{N}(0, \sigma^2_\iota) \qquad \epsilon_{ijk} \overset{iid}{\sim} \mathcal{N}(0, \sigma^2_\epsilon) \tag{12}$$

$$\boldsymbol{Y} = [\mathbf{1}_\mu | \boldsymbol{X}_\kappa | \boldsymbol{X}_\iota | \boldsymbol{X}_\alpha] \begin{bmatrix} \mu \\ \boldsymbol{\kappa} \\ \boldsymbol{\iota} \\ \boldsymbol{\alpha} \end{bmatrix} + \boldsymbol{\epsilon} \tag{13}$$

```
#-------------------------------------------------------------------------------------------
# blocos dentro dos grupos são de efeito aleatório
# criar o fator bloco dentro de grupo para associar efeito aleatório
bib2$bloc.grup <- with(bib2, interaction(bloc, grup))
str(bib2)
                                                                                                    #
#-------------------------------------------------------------------------------------------
# ajuste do modelo e teste de Wald sequencial para os efeitos fixos
mm0 <- lme(resp~grup+trat, random=~1|bloc.grup, data=bib2,
           contrast=list(grup=contr.sum, trat=contr.sum))
par(mfrow=c(2,2)) # checagem
qqnorm(residuals(mm0)); qqline(residuals(mm0))
qqnorm(unlist(ranef(mm0))); qqline(unlist(ranef(mm0)))
scatter.smooth(residuals(mm0)~fitted(mm0))
scatter.smooth(abs(residuals(mm0))~fitted(mm0))
layout(1)
anova(mm0)   # teste de Wald
summary(mm0) # estimativas
                                                                                                    #
#-------------------------------------------------------------------------------------------
# estimativas dos componentes de variância
VarCorr(mm0)
#-------------------------------------------------------------------------------------------
# médias ajustadas
Xtrat <- cbind(1, mm0$contrast$trat)
assi <- lapply(list(grup="grup",trat="trat"), function(x){ grep(x, names(fixef(mm0))) })
mmq <- c(Xtrat%*%fixef(mm0)[c(1, assi$trat)])
mmq
                                                                                                    #
#-------------------------------------------------------------------------------------------
# vetor de coeficientes do contraste tr1 vs tr2 e sua estimativa
contr <- rep(c(0,0,1,-1,0), c(1,length(assi$grup),1,1,length(assi$trat)-2)); contr
sum(contr*fixef(mm0))
                                                                                                    #
#-------------------------------------------------------------------------------------------
```

```
# variância do contraste, como tem equilíbrio, todos os contrastes tem mesma variância
v.dif <- contr%*%vcov(mm0)%*%contr; v.dif
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# diferença mínima significativa pelo teste de Tukey
delta <- qtukey(0.95, nlevels(bib2$trat), df=anova(mm0)["trat","denDF"])*sqrt(0.5*v.dif)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# aplicação do teste de Tukey
agricolae::order.stat(levels(bib2$trat), mmq, delta)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# usando a gmodels::estimable()
mc <- mm0$contrast$trat
mc <- cbind(matrix(0, ncol=length(assi$grup)+1, nrow=nrow(mc)), mc)
mc[,1] <- 1
mc
estimable(mm0, mc) # médias de mínimos quadrados
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# todos os contrastes possíveis entre os níveis de tratamento
nl <- nlevels(bib2$trat) # número de níveis
nc <- t(combn(nl, 2))    # comparações
mc <- sapply(1:nrow(nc), function(x){ aux <- rep(0, nl); aux[nc[x,]] <- c(1,-1); aux })
mc <- t(mc)              # matriz de todos os contrastes 2 à 2
restr <- cbind(1, mm0$contrast$trat)
contr <- mc%*%restr
contr[,-1]               # são as colunas dos contrastes entre os níveis de tratamentos
contr.all <- cbind(matrix(0, nrow=nrow(mc), ncol=length(assi$grup)+1), contr[,-1])
rownames(contr.all) <- apply(nc, 1, paste, collapse="-")
str(contr.all)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# usando a multcomp
require(multcomp)
glht0 <- summary(glht(mm0, linfct=contr.all), test=adjusted(type="fdr"))
glht0
plot(glht0)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
.
```

8.7 Descrição dos dados

```
.
#-------------------------------------------------------------------------------------------------
# dados de um experimento em blocos incompletos tipo III (Pimentel Gomes, p.179)
# é o tipo mais comum de delineamento bib (melhoramento e análise sensorial)
# -- não existe agrupamento em repetições --
# bloc: é o bloco
# trat: são os tratamentos
# resp: é a resposta observada
bib3 <- c(1, 1, 35, 1, 2, 28, 1, 3, 27, 2, 1, 30, 2, 2, 20, 2, 4, 22,
          3, 1, 28, 3, 2, 16, 3, 5, 18, 4, 1, 36, 4, 3, 29, 4, 4, 30,
          5, 1, 29, 5, 3, 19, 5, 5, 22, 6, 1, 25, 6, 4, 16, 6, 5, 19,
          7, 2, 26, 7, 3, 30, 7, 4, 28, 8, 2, 27, 8, 3, 29, 8, 5, 27,
          9, 2, 29, 9, 4, 29, 9, 5, 27, 10, 3, 27, 10, 4, 26, 10, 5, 29)
bib3 <- as.data.frame(matrix(bib3, ncol=3, byrow=TRUE))
names(bib3) <- c("bloc","trat","resp")
bib3 <- transform(bib3, bloc=factor(bloc), trat=factor(trat))
str(bib3)
with(bib3, table(trat))             # número de parcelas com cada tratamento
with(bib3, table(bloc))             # número de parcelas por bloco
with(bib3, table(trat, bloc))       # relação bloc trat
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# gráficos
xyplot(resp~trat, groups=bloc, data=bib3, type="o")
                                                                                                    #
#-------------------------------------------------------------------------------------------------
.
```

8.8 Análise intra bloco de BIB tipo III

$$Y_{ij} = \mu + \iota_i + \alpha_j + \epsilon_{ij} \qquad \epsilon_{ij} \overset{iid}{\sim} \mathcal{N}(0, \sigma^2_\epsilon) \tag{14}$$

$$\boldsymbol{Y} = [\mathbf{1}_\mu | \boldsymbol{X}_\iota | \boldsymbol{X}_\alpha] \begin{bmatrix} \mu \\ \boldsymbol{\iota} \\ \boldsymbol{\alpha} \end{bmatrix} + \boldsymbol{\epsilon} \tag{15}$$

```
#-------------------------------------------------------------------------------------------
# soma de quadrados sequencial com tratamentos ajustados aos blocos (intrabloco)
m0 <- lm(resp~bloc+trat, data=bib3,
         contrast=list(bloc=contr.sum, trat=contr.sum))
par(mfrow=c(2,2)); plot(m0); layout(1) # checagem
anova(m0) # quadro de análise de variância
summary(m0) # quadro de estimativas
                                                                                          #
#-------------------------------------------------------------------------------------------
# soma de quadrados marginal, ambos fatores ajustados um ao outro
drop1(m0, scope=.~., test="F")
car::Anova(m0, type="III")
                                                                                          #
#-------------------------------------------------------------------------------------------
# tipo de contraste
m0$contrast$trat
m0$assign
                                                                                          #
#-------------------------------------------------------------------------------------------
# obtenção das médias ajustadas é o produto matricial abaixo somado à média geral
contr <- cbind(1, m0$contrast$trat)
mmq <- contr%*%coef(m0)[m0$assign%in%c(0,2)]
mmq
                                                                                          #
#-------------------------------------------------------------------------------------------
# vetor de coeficientes do contraste tr1 vs tr2 e sua estimativa
contr <- rep(c(0,0,1,-1,0), c(1, nlevels(bib3$bloc)-1, 1, 1, nlevels(bib3$trat)-3)); contr
sum(contr*coef(m0))
                                                                                          #
#-------------------------------------------------------------------------------------------
# variância do contraste, como tem equilibrio, todos os contrastes tem mesma variância
v.dif <- contr%*%vcov(m0)%*%contr; v.dif
                                                                                          #
#-------------------------------------------------------------------------------------------
# diferença mínima significativa pelo teste de Tukey
delta <- qtukey(0.95, nlevels(bib3$trat), df=df.residual(m0))*sqrt(0.5*v.dif); delta
                                                                                          #
#-------------------------------------------------------------------------------------------
# aplicação do teste de Tukey
tk <- agricolae::order.stat(levels(bib3$trat), mmq, delta)
tk
                                                                                          #
#-------------------------------------------------------------------------------------------
# usando a contrast::contrast() para obter as médias \mu+\alpha_i
require(contrast)
contrast(m0, type="average", list(bloc=levels(bib3$bloc), trat="1")) # colocar na sapply()
                                                                                          #
#-------------------------------------------------------------------------------------------
# usando a gmodels::estimable (só válido quando usar contr.sum)
mc <- m0$contrast$trat
mc <- cbind(matrix(0, ncol=sum(m0$assign<2), nrow=nrow(mc)), mc)
mc[,1] <- 1
mc
estimable(m0, mc) # médias de mínimos quadrados
                                                                                          #
#-------------------------------------------------------------------------------------------
# todos os contrastes possíveis entre os níveis de tratamento
nl <- nlevels(bib3$trat) # número de níveis
nc <- t(combn(nl, 2))    # comparações
mc <- sapply(1:nrow(nc), function(x){ aux <- rep(0, nl); aux[nc[x,]] <- c(1,-1); aux })
mc <- t(mc)              # matriz de todos os contrastes 2 à 2
restr <- cbind(1, m0$contrast$trat)
contr <- mc%*%restr
contr[,-1]               # são as colunas dos contrastes entre os níveis de tratamentos
contr.all <- cbind(matrix(0, nrow=nrow(mc), ncol=sum(m0$assign<2)), contr[,-1])
rownames(contr.all) <- apply(nc, 1, paste, collapse="-")
str(contr.all)
```

```
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# usando a multcomp
require(multcomp)
glht0 <- summary(glht(m0, linfct=contr.all), test=adjusted(type="fdr"))
glht0
plot(glht0)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# usando a HH
require(HH)
mmc <- glht.mmc(m0, linfct=contr.all, focus="trat")
plot(mmc)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# o pacote agricolae tem a função bib.test() para está análise intra bloco
require(agricolae)
apropos("bib")
str(bib3)
BIB.test(block=bib3$bloc, trt=bib3$trat, y=bib3$resp, method="lsd", alpha=0.05)
BIB.test(block=bib3$bloc, trt=bib3$trat, y=bib3$resp, method="duncan", alpha=0.05)
BIB.test(block=bib3$bloc, trt=bib3$trat, y=bib3$resp, method="snk", alpha=0.05)
BIB.test(block=bib3$bloc, trt=bib3$trat, y=bib3$resp, method="tukey", alpha=0.05)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
.
```

8.9 Análise inter bloco de BIB tipo III

$$Y_{ij} = \mu + \iota_i + \alpha_j + \epsilon_{ij} \qquad \iota_i \overset{iid}{\sim} \mathcal{N}(0, \sigma^2_\iota) \qquad \epsilon_{ij} \overset{iid}{\sim} \mathcal{N}(0, \sigma^2_\epsilon) \tag{16}$$

$$\boldsymbol{Y} = [\mathbf{1}_\mu | \boldsymbol{X}_\iota | \boldsymbol{X}_\alpha] \begin{bmatrix} \mu \\ \boldsymbol{\iota} \\ \boldsymbol{\alpha} \end{bmatrix} + \boldsymbol{\epsilon} \tag{17}$$

```
#-----------------------------------------------------------------------------------------------
# ajuste do modelo, anova() não faz anova e sim teste de Wald sequencial para efeitos fixos
require(nlme)
mm0 <- lme(resp~trat, random=~1|bloc, data=bib3, contrast=list(trat=contr.sum))
par(mfrow=c(1,2))
qqnorm(residuals(mm0)); qqline(residuals(mm0))
qqnorm(unlist(ranef(mm0))); qqline(unlist(ranef(mm0)))
layout(1)
anova(mm0)
summary(mm0)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# estimativa dos componetes de variância
VarCorr(mm0)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# matriz de contraste dos tratamentos
restr <- cbind(1, mm0$contrasts$trat)
fixef(mm0)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# médias ajustadas dos tratamentos
mmq <- restr%*%fixef(mm0)
mmq
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# vetor de coeficientes do contraste tr1 vs tr2 e sua estimativa
contr <- c(1,-1,0,0,0); contr
contr%*%restr%*%fixef(mm0)
contr%*%mmq
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# variância do contraste, como tem equilibrio, todos os contrastes tem mesma variância
v.dif <- (contr%*%restr)%*%vcov(mm0)%*%t(contr%*%restr); sqrt(v.dif)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# diferença mínima significativa pelo teste de Tukey
```

```
delta <- c(qtukey(0.95, nlevels(bib3$trat), anova(mm0)["trat","denDF"])*sqrt(0.5*v.dif))
delta
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# aplicação do teste de Tukey
agricolae::order.stat(levels(bib3$trat), mmq, delta)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# usando a gmodels::estimable()
mc <- cbind(1, mm0$contrast$trat)
mc
estimable(mm0, mc) # médias de mínimos quadrados
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# todos os contrastes possíveis entre os níveis de tratamento
nl <- nlevels(bib3$trat) # número de níveis
nc <- t(combn(nl, 2))    # comparações
mc <- sapply(1:nrow(nc), function(x){ aux <- rep(0, nl); aux[nc[x,]] <- c(1,-1); aux })
mc <- t(mc)              # matriz de todos os contrastes 2 à 2
restr <- cbind(1, mm0$contrast$trat)
contr <- mc%*%restr
contr[,-1]               # são as colunas dos contrastes entre os níveis de tratamentos
contr.all <- contr
rownames(contr.all) <- apply(nc, 1, paste, collapse="-")
str(contr.all)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# usando a multcomp
require(multcomp)
glht0 <- summary(glht(mm0, linfct=contr.all), test=adjusted(type="fdr"))
glht0
plot(glht0)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
.
```

8.10 Função para casualização em BIB tipo III

```
.
#-------------------------------------------------------------------------------------------------
# função da agricolae, gerar um bib para 7 tratamentos, bloco de tamanho 3
trt <- LETTERS[1:7]
k <- 3
bib <- design.bib(trt, k, kinds="Super-Duper")
str(bib)
bib
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# arrumar o resultado no formato amplo, melhor para vizualizar
field <- as.character(bib[,3])
t(matrix(field,c(3,4)))
write.table(bib, file="bib.txt", row.names=FALSE, sep="\t")
write.table(bib, file="bib.xls", row.names=FALSE, sep="\t")
                                                                                                    #
#-------------------------------------------------------------------------------------------------
.
```

9 Análise de experimento em quadrado latino

9.1 Modelo de efeitos fixos

$$Y_{ijk} = \mu + \kappa_i + \iota_j + \alpha_k + \epsilon_{ijk} \qquad \epsilon_{ijk} \overset{iid}{\sim} \mathcal{N}(0, \sigma^2_\epsilon) \tag{18}$$

$$\boldsymbol{Y} = [\mathbf{1}_\mu | \boldsymbol{X}_\kappa | \boldsymbol{X}_\iota | \boldsymbol{X}_\alpha] \begin{bmatrix} \mu \\ \boldsymbol{\iota} \\ \boldsymbol{\iota} \\ \boldsymbol{\alpha} \end{bmatrix} + \boldsymbol{\epsilon} \tag{19}$$

```
.
#-------------------------------------------------------------------------------------------
# o delineamento quadrado latico faz controle local em duas direções. Normalmente está
# associado à experimentos de campo onde controla a variabilidade nas linhas (diferencial
# de atributos de solo) e nas colunas (diferencial de manejo: máquinas, irrigação, manejo)
# o número de linhas e colunas é o número de níveis do tratamento, por isso o número de
# é sempre o quadrado do número de níveis. Pode ser entendido como um caso particular do
# delineamento crossover. Quando possível, fazer o controle local em uma direção apenas.
                                                                                          #
#-------------------------------------------------------------------------------------------
# dados 'melon' do pacote agricolae (preciso de exemplos reais desse delineamento)
require(agricolae)
help(melon, help_type="html")
data(melon)
str(melon)
melon <- transform(melon, row=factor(row), col=factor(col))
                                                                                          #
#-------------------------------------------------------------------------------------------
# análise
m0 <- lm(yield~row+col+variety, data=melon)
par(mfrow=c(2,2)); plot(m0); layout(1)
anova(m0)
                                                                                          #
#-------------------------------------------------------------------------------------------
# no caso de balanceamento as médias amostrais são estimadores admissíveis das médias
# podemos usar as funções da agricolae para fazer testes de médias
glr <- df.residual(m0)
qmr <- deviance(m0)/glr
with(melon, LSD.test(yield, variety, glr, qmr))
with(melon, SNK.test(yield, variety, glr, qmr))
with(melon, HSD.test(yield, variety, glr, qmr))
                                                                                          #
#-------------------------------------------------------------------------------------------
# podemos aplicar o procedimento de Scott-Knott (porque médias amostrais são admissíveis)
require(ScottKnott)
sk <- SK(x=melon, y=melon$yield, model=formula(m0), which="variety")
summary(sk)
                                                                                          #
#-------------------------------------------------------------------------------------------
# obtendo as médias de mínimos quadrados (vão coincidir com as médias amostrais nesse caso)
# fazer isso diretamente com a HH::glht.mmc()
require(HH)
glht0 <- glht.mmc(m0, linfct=mcp(variety="Tukey"))
glht0
plot(glht0)
                                                                                          #
#-------------------------------------------------------------------------------------------
.
```

9.2 Modelo de efeitos fixos e aleatórios

$$Y_{ijk} = \mu + \kappa_i + \iota_j + \alpha_k + \epsilon_{ijk} \qquad \iota_j \overset{iid}{\sim} \mathcal{N}(0, \sigma^2_\iota) \qquad \epsilon_{ijk} \overset{iid}{\sim} \mathcal{N}(0, \sigma^2_\epsilon) \tag{20}$$

$$\boldsymbol{Y} = [\mathbf{1}_\mu | \boldsymbol{X}_\kappa | \boldsymbol{X}_\iota | \boldsymbol{X}_\alpha] \begin{bmatrix} \mu \\ \boldsymbol{\iota} \\ \boldsymbol{\iota} \\ \boldsymbol{\alpha} \end{bmatrix} + \boldsymbol{\epsilon} \tag{21}$$

```
#-------------------------------------------------------------------------------------------
# vamos considerar que nas colunas está sendo feito o controle para condições de manejo
# (pessoas, máquinas, irrigação) que podem ser pensados como de origem aleatória, assim
# nosso modelo terá um termo de efeito aleatório. Vamos usar o procedimento REML.
                                                                                          #
#-------------------------------------------------------------------------------------------
# ajustando modelo com col de efeito aleatório
require(nlme)
mm0 <- lme(yield~row+variety, random=~1|col, data=melon,
           contrast=list(row=contr.sum, variety=contr.sum))
summary(mm0)
VarCorr(mm0)
                                                                                          #
```

```
#-------------------------------------------------------------------------------------
# médias
mc <- cbind(1, mm0$contrast$variety)
assign <- lapply(list(row="row", variety="variety"), grep, x=names(fixef(mm0)))
assign
mmq <- mc%*%fixef(mm0)[c(1,assign$variety)]
mmq # médias de mínimos quadrados
                                                                                    #
#-------------------------------------------------------------------------------------
# todos os contrastes possíveis entre os níveis de tratamento
nl <- nlevels(melon$variety) # número de níveis
nc <- t(combn(nl, 2))        # comparações
mc <- sapply(1:nrow(nc), function(x){ aux <- rep(0, nl); aux[nc[x,]] <- c(1,-1); aux })
mc <- t(mc)                  # matriz de todos os contrastes 2 à 2
restr <- cbind(1, mm0$contrast$variety)
contr <- mc%*%restr
contr[,-1]               # são as colunas dos contrastes entre os níveis de tratamentos
contr.all <- cbind(matrix(0, nrow=nrow(mc), ncol=length(assign$row)+1), contr[,-1])
rownames(contr.all) <- apply(nc, 1, paste, collapse="-")
str(contr.all)
                                                                                    #
#-------------------------------------------------------------------------------------
# usando a multcomp
require(multcomp)
glht0 <- summary(glht(mm0, linfct=contr.all), test=adjusted(type="fdr"))
glht0
plot(glht0)
                                                                                    #
#-------------------------------------------------------------------------------------
.
```

9.3 Função para casualização em quadrados latinos

```
.
#-------------------------------------------------------------------------------------
# função da agricolae, gerar um bib para 7 tratamentos, bloco de tamanho 3
trt <- LETTERS[1:7]
lsd <- design.lsd(trt, kinds="Super-Duper")
str(lsd)
matrix(lsd$trt, ncol=length(trt), nrow=length(trt))
                                                                                    #
#-------------------------------------------------------------------------------------
# arrumar o resultado no formato amplo, melhor para vizualizar
write.table(lsd, file="lsd.txt", row.names=FALSE, sep="\t")
write.table(lsd, file="lsd.xls", row.names=FALSE, sep="\t")
                                                                                    #
#-------------------------------------------------------------------------------------
.
```

10 Análise de experimento fatorial duplo em DIC

10.1 Ajuste do modelo

$$Y_{ijk} = \mu + \alpha_i + \gamma_j + \delta_{ij} + \epsilon_{ijk} \qquad \epsilon_{ijk} \overset{iid}{\sim} \mathcal{N}(0, \sigma^2_\epsilon) \tag{22}$$

$$\boldsymbol{Y} = [\mathbf{1}_\mu | \boldsymbol{X}_\alpha | \boldsymbol{X}_\gamma | \boldsymbol{X}_\delta] \begin{bmatrix} \mu \\ \boldsymbol{\alpha} \\ \boldsymbol{\gamma} \\ \boldsymbol{\delta} \end{bmatrix} + \boldsymbol{\epsilon} \tag{23}$$

```
.
#-------------------------------------------------------------------------------------
# dados de volume de raízes de variedades de sorgo submetidas à doses de indutor de
# enraizamento
#vol <- read.table("http://www.leg.ufpr.br/~walmes/cursoR/volume.txt", header=TRUE)
vol <- read.table("../dados/volume.txt", header=TRUE)
str(vol)
```

```
unique(vol$dose)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# análise gráfica
require(lattice)
xyplot(volu~dose, groups=gen, data=vol, type=c("p","smooth"))
xyplot(volu~gen|dose, data=vol)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# análise de variância
m0 <- aov(volu~gen+dose+gen:dose, data=vol) # modelo decrarado por extenso
m0 <- aov(volu~gen*dose, data=vol)          # modelo declarado por abre
summary(m0)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# verificando tipo das variáveis (para vetores: numeric, integer, factor, character)
class(vol$gen)
class(vol$dose)
vol$dose <- factor(vol$dose) # converte para fator (níveis categóricos)
levels(vol$dose)             # apresenta os níveis/categorias do fator
class(vol$dose)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# análise de variância com a especificação correta
m0 <- aov(volu~gen*dose, data=vol)
summary(m0) # uma quase interação
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# checagem das pressuposições
par(mfrow=c(2,2)); plot(m0); layout(1)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# testes
shapiro.test(residuals(m0))
bartlett.test(residuals(m0)~interaction(vol$gen,vol$dose)) # sensível à fuga de normalidade
car::leveneTest(m0)               # um pouco mais robusto (mediana)
car::leveneTest(m0, center="mean") # (média)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
.
```

10.2 Aplicando transformação Box-Cox

```
.
#-------------------------------------------------------------------------------------------------
# precisa-se de tranformação para normalidade e homocedasticidade
require(MASS)
boxcox(m0)
#locator(n=1) # clicar no gráfico. Seria coincidência dar 1/3 visto que volume é mm^3?
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# usando a tranformação indicada
vol$volu0.33 <- vol$volu^0.33
m1 <- lm(volu0.33~gen*dose, data=vol)
par(mfrow=c(2,2)); plot(m1); layout(1)
shapiro.test(residuals(m1))
bartlett.test(residuals(m1)~paste(vol$gen,vol$dose))
car::leveneTest(m1)               # um pouco mais robusto (mediana)
car::leveneTest(m1, center="mean") # (média)
anova(m1)   # ausência de interação
summary(m1)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# falta de normalidade é causa da mistura de 3 normais com variância distinta o que resulta
# numa distribuição mistura com mais curtose
                                                                                                    #
#-------------------------------------------------------------------------------------------------
.
```

10.3 Teste de médias para dados transformados

```
.
#-------------------------------------------------------------------------------------------------
```

```
# dado que não houve interação, significa que diferenças entre níveis de dose são
# constantes em relação aos níveis de genótipo, e da mesma forma, diferenças entre
# genótipos são constantes com relação aos níveis de dose. Podemos não declarar a interação
# no modelo?
anova(lm(volu0.33~gen+dose, data=vol))
anova(m1)
                                                                                                #
#-------------------------------------------------------------------------------------------
# teste de média para o efeito principal dos fatores (a média de um nível de A é a média do
# seu desempenho em todos os níveis de B, o que importa são as diferenças e o raking)
require(agricolae)
glr <- df.residual(m1)
qmr <- deviance(m1)/glr
tk.g <- with(vol, HSD.test(volu0.33, gen, glr, qmr))
tk.g
tk.d <- with(vol, HSD.test(volu0.33, dose, glr, qmr))
tk.d
                                                                                                #
#-------------------------------------------------------------------------------------------
# colocar os resultados em um gráfico de barras, dessa vez, usar a lattice::barchart()
tk.g$meansl <- paste(format(tk.g$means, dig=3),      # formata as médias com 3 digitos
                     gsub(" ","", tk.g$M), sep="\n") # remove os espaços e concatena
barchart(means~reorder(trt, means, mean), data=tk.g, horiz=FALSE,
         ylab="Raíz cúbica do volume de raízes", ylim=c(0.9,1.45),
         xlab="Genótipo de sorgo",
         sub=list("*médias seguidas de mesma letra não diferem entre sí pelo teste de Tukey (5%).",
           cex=0.8, font=3),
         strip=strip.custom(bg="gray75"), col=colors()[46],
         panel=function(x, y, subscripts, ...){
           panel.barchart(x, y, subscripts=subscripts, ...)
           panel.text(x, y, label=tk.g[subscripts,"meansl"], pos=3, cex=1)
         })
                                                                                                #
#-------------------------------------------------------------------------------------------
# aplicando Scott-Knott
require(ScottKnott)
sk.g <- SK(x=vol, y=vol$volu0.33, model=formula(m1), which="gen", id.trim=6)
sk.g <- summary(sk.g)
                                                                                                #
#-------------------------------------------------------------------------------------------
# gráfico
sk.g$meansl <- paste(format(sk.g$Means, dig=3), sk.g$SK, sep="\n")
barchart(Means~reorder(Levels, Means, mean), data=sk.g, horiz=FALSE,
         ylab="Raíz cúbica do volume de raízes", ylim=c(0.9,1.45),
         xlab="Genótipo de sorgo",
         sub=list("*médias seguidas de mesma letra não diferem entre sí pelo teste de Skott-Knott (5%).",
           cex=0.8, font=3),
         strip=strip.custom(bg="gray75"), col=colors()[88],
         panel=function(x, y, subscripts, ...){
           panel.barchart(x, y, subscripts=subscripts, ...)
           panel.text(x, y, label=sk.g[subscripts,"meansl"], pos=3, cex=1)
         })
                                                                                                #
#-------------------------------------------------------------------------------------------
.
```

10.4 Transformação para correção da heterocedasticidade

```
#-------------------------------------------------------------------------------------------
# parte do pressuposto que \sigma = a*\mu^b (o desvio padrão é função potência da média)
# tomando o log dos dois lados temos ln(\sigma) = ln(a)+b*ln(\mu), se b for 0, \sigma será
# constante, temos que encontrar o valor de b e fazer sigma constante
musd <- with(vol, aggregate(volu, list(gen, dose), function(x) c(mean(x), sd(x))))
str(musd)
scatter.smooth(musd$x)
scatter.smooth(log(musd$x))
                                                                                                #
#-------------------------------------------------------------------------------------------
# estimando a e b
ab <- lm(log(musd$x[,2])~log(musd$x[,1]))
coef(ab)
abline(ab, col=2)
                                                                                                #
#-------------------------------------------------------------------------------------------
# aplicando a transformação: elevar a resposta a 1-b para estabilizar a variância
```

```
lambda <- 1-coef(ab)[2]; lambda
vol$volulamb <- vol$volu^lambda
                                                                                                #
#-------------------------------------------------------------------------------------------
# rodando a análise
m2 <- aov(volulamb~gen*dose, data=vol)
par(mfrow=c(2,2)); plot(m2); layout(1)
shapiro.test(residuals(m2))        # piorou
bartlett.test(residuals(m2)~paste(vol$gen,vol$dose))
car::leveneTest(m2)                # um pouco mais robusto (mediana)
car::leveneTest(m2, center="mean") # (média)
summary(m2)
                                                                                                #
#-------------------------------------------------------------------------------------------
# essa transformação visa apenas tornar a variância constante partindo do pressuposto que
# esta é uma função potência da média. Nesse caso ela não foi melhor que a BoxCox que ao
# buscar uma tranformação que maximize a verossilhança, está considerando tanto normalidade
# como homocestasticidade.
                                                                                                #
#-------------------------------------------------------------------------------------------
.
```

10.5 Análise usando a função `ExpDes::fat2.crd()`

```
.
#-------------------------------------------------------------------------------------------
# usando a variável original
require(ExpDes)
with(vol, fat2.crd(factor1=gen, factor2=dose, resp=volu0.33,
                   quali=c(TRUE,TRUE), mcomp="sk"))
with(vol, fat2.crd(gen, dose, volu0.33, mcomp="lsd"))
                                                                                                #
#-------------------------------------------------------------------------------------------
.
```

10.6 Estimação ponderada pela variância amostral

```
.
#-------------------------------------------------------------------------------------------
# o padrão de heterogeneidade em função das doses
xyplot(volu~gen|dose, data=vol)
                                                                                                #
#-------------------------------------------------------------------------------------------
# ajustar modelo de dic dentro de cada dose e obter a variância (qmr)
m00 <- lapply(split(vol, f=vol$dose),
              function(split){
                m0 <- lm(volu~gen, data=split); m0
              })
str(m00)
lapply(m00, anova)
w <- unlist(lapply(m00, deviance))/unlist(lapply(m00, df.residual)) # são os qmr
w <- data.frame(w=w, id=levels(vol$dose))
vol <- merge(vol, w, by.x="dose", by.y="id")
str(vol)
                                                                                                #
#-------------------------------------------------------------------------------------------
# fazendo a análise de variância passando pesos para corrigir a heterogeneidade
m3 <- lm(volu~gen*dose, data=vol, weights=1/w) # passando os pesos
par(mfrow=c(2,2)); plot(m3); layout(1)      # resíduos ok!
shapiro.test(residuals(m3, type="pearson")) # resíduos de pearson são corrigidos pelos pesos
bartlett.test(residuals(m3, type="pearson")~paste(vol$gen,vol$dose))
anova(m3)   # interação presente
summary(m3)
                                                                                                #
#-------------------------------------------------------------------------------------------
# como os pesos são usados? e estimação envolve a matriz de pesos W assim
X <- model.matrix(m3)
Y <- matrix(vol$volu)
W <- diag(1/vol$w)
betas <- solve(t(X)%*%W%*%X)%*%(t(X)%*%W%*%Y) # lembre-se, crossprod() e QR() +efeicientes
cbind(betas, coef(m3))
                                                                                                #
```

```
#-------------------------------------------------------------------------------------
# vantagem: não transforma a resposta, não perde interpretação
# desvantagem: estimativas com erros padrões difererentes, tratar isso no teste de médias
                                                                                     #
#-------------------------------------------------------------------------------------
.
```

10.7 Desdobramento da interação dentro da anova

```
.
#-------------------------------------------------------------------------------------
# vamos desdobrar as somas de quadrados dentro da análise de variância, usar aov()
# desdobrar dose dentro de genótipo, gen/dose = gen+gen:dose
m3 <- aov(volu~gen/dose, data=vol, weights=1/w)
summary(m3)
coef(m3) # temos que saber as posições das doses para cada genótipo, usar grep()
coef.name <- names(coef(m3))[m3$assign==2]
desd.dose <- sapply(levels(vol$gen), grep, x=coef.name, simplify=FALSE)
desd.dose
summary(m3, split=list("gen:dose"=desd.dose))
                                                                                     #
#-------------------------------------------------------------------------------------
# podemos desdobrar os dois graus de liberdade que equivalem aos contrastes 0vs05 e 0vs25
coef.name <- gsub("^gen(.*):dose(.*)$", "\\1-0vs\\2", coef.name) # trata os nomes
coef.name <- gsub("vs5", "vs05", coef.name)                      # troca 5 por 05
desd.dose.2 <- as.list(order(coef.name))
names(desd.dose.2) <- sort(coef.name)
summary(m3, split=list("gen:dose"=desd.dose.2))
                                                                                     #
#-------------------------------------------------------------------------------------
.
```

10.8 Contrastes com a testemunha para ajuste com pesos

```
.
#-------------------------------------------------------------------------------------
# estimativas das médias
pred <- expand.grid(gen=levels(vol$gen), dose=levels(vol$dose))
pred$means <- predict(m3, newdata=pred)
barchart(means~dose|gen, data=pred)
                                                                                     #
#-------------------------------------------------------------------------------------
# vamos estimar os contrastes 0 vs 5 e 0 vs 25
require(gmodels)
mc <- model.matrix(~gen/dose, pred)
mc[,m3$assign<2] <- 0 # recebe 0 porque esse coeficientes se anulam no contraste
mc.split <- split(as.data.frame(mc), f=pred$gen)
mc.split <- lapply(mc.split, as.matrix)
mc.split <- lapply(mc.split, function(x){ rownames(x) <- c("0","05-0","25-0"); x} )
mmq <- lapply(mc.split,
              function(split){
                e0 <- estimable(m3, cm=split[-1,])
              })
c0 <- do.call(rbind, mmq) # conferir o p-valor com os do teste F da anova desdobrada
c0
                                                                                     #
#-------------------------------------------------------------------------------------
# vamos colocar um * e ** ao lados das médias de 5 e 25 que diferiram de 0
ast <- ifelse(c0[,5]<0.05, "*", "")
pred$ast <- ""
pred <- pred[order(pred$gen, pred$dose),]
pred$ast[pred$dose!="0"] <- ast
pred$let <- with(pred, paste(format(means, dig=1), ast, sep="0"))
                                                                                     #
#-------------------------------------------------------------------------------------
# gráfico final
barchart(means~dose|gen, data=pred, horiz=FALSE, layout=c(3,3),
         ylab="Volume de raízes", ylim=c(0.40,4.5),
         xlab="Dose de indutor de enraizamento",
         key=list(text=list("Genótipos de sorgo")),
         sub=list("* ao lado das médias indica contraste signicativo em relação à dose 0 pelo teste t (5%).",
           cex=0.8, font=3),
```

```
         strip=strip.custom(bg="gray75"), col=colors()[133],
         panel=function(x, y, subscripts, ...){
           panel.barchart(x, y, subscripts=subscripts, ...)
           panel.text(x, y, label=pred[subscripts,"let"], pos=3, cex=0.9)
         })
                                                                                                    #
#-------------------------------------------------------------------------------------------
# as cores da paleta colors()
colors()         # as cores pelo nome
length(colors()) # número de cores
x <- expand.grid(1:25, 1:25)
x$col <- 1:nrow(x)
plot(Var1~Var2, x, pch=19, col=colors()[x$col], cex=4)
with(x, text(Var2, Var1, text=col, cex=0.5))
#with(x, identify(Var2, Var1))
                                                                                                    #
#-------------------------------------------------------------------------------------------
# porque algumas doses 5 foram maiores que as 25? Porque uma única solução em cada dose é
# feita. Logo, as 3 repetições recebem a mesma concentração, que se sujeita à erros de
# titulação(deve ter sido o caso) compromete o resultado das 3 repetições. Isso deve ser
# evitado no planejamento. As unidades experimentais devem ser independentes!
                                                                                                    #
#-------------------------------------------------------------------------------------------
# como fica a análise com uma distribuição gamma?
                                                                                                    #
#-------------------------------------------------------------------------------------------
.
```

10.9 Modelando a variância simultaneamente

```
.
#-------------------------------------------------------------------------------------------
# a análise com weigths tornou a análise mais coerente porém ainda não é um procedimento
# "ótimo". Os pesos usados são estimativas que possuem um grau de incerteza e não valores
# paramétricos. Ao fazer a modelagem conjunta, ou seja, estimando os pesos junto com os
# efeitos essa incerteza entra no modelo. Usaremos a nlme::gls()
require(nlme)
help(varClasses, help_type="html")
g0 <- gls(volu~gen*dose, data=vol, weights=varIdent(form=~1|dose)) # weights recebe função
g0         # os pesos foram estimados junto com os efeitos
sqrt(w$w)  # mas são os mesmos pesos do procedimento duas etapas
                                                                                                    #
#-------------------------------------------------------------------------------------------
# veja as log-verossimilhanças
logLik(m3) # gl 28
logLik(g0) # gl 28 + 2 (pesos que foram estimados)
                                                                                                    #
#-------------------------------------------------------------------------------------------
# não é um teste de Wald, que para modelos lineares dá o mesmo F da anova
anova(g0)
anova(m3)
                                                                                                    #
#-------------------------------------------------------------------------------------------
# veja que os erros padrões dos efeitos são muito semelhantes
summary(m3)$coeff
summary(g0)$tTable
                                                                                                    #
#-------------------------------------------------------------------------------------------
# usando a contrast::contrast() para desdobrar a interação (não usar com lm(... weigths))
gen.lev <- levels(vol$gen)
require(contrast)
contrast(g0, list(gen=gen.lev[1], dose=c("5","25")), list(gen=gen.lev[1], dose="0"))
contrast(g0, list(gen=gen.lev[9], dose=c("5","25")), list(gen=gen.lev[9], dose="0"))
# fazendo para todos os níveis de gen
c00 <- sapply(gen.lev, simplify=FALSE,
              function(g){
                c0 <- contrast(g0, list(gen=g, dose=c("5","25")), list(gen=g, dose="0"))
                print(c0)
                return(c0[1:8])
              })
do.call(rbind, lapply(c00, as.data.frame))
                                                                                                    #
#-------------------------------------------------------------------------------------------
# podemos usar uma função de variância que seja contínua na dose
vol$d <- as.numeric(as.character(vol$dose))
str(vol)
```

```
g1 <- gls(volu~gen*dose, data=vol, weights=varExp(form=~d))
g1         # os pesos foram estimados junto com os efeitos
logLik(g0) # gl 28 + 2 (pesos que foram estimados)
logLik(g1) # gl 28 + 1 (pesos que foram estimados)
AIC(g0) # menor é melhor!
AIC(g1)
                                                                                        #
#---------------------------------------------------------------------------------------
.
```

10.10 Função para casualização de experimento fatorial

```
.
#---------------------------------------------------------------------------------------
# fazendo a casualização dos níveis dos tratamentos às unidades experimentais numeradas
# usar a mesma função do DIC, já que a combinação dos níveis pode ser vista como um fator
A <- gl(5, 1, labels="A")      # 5 níveis
B <- gl(3, 1, labels="B")      # 3 níveis
AB <- c(outer(A, B, paste))    # 5*3 = 15 níveis
r <- 6                         # repetições
fat2 <- design.crd(AB, r, kinds="Super-Duper")
str(fat2)
write.table(fat2, file="fat2.txt", row.names=FALSE, sep="\t")
write.table(fat2, file="fat2.xls", row.names=FALSE, sep="\t")
                                                                                        #
#---------------------------------------------------------------------------------------
.
```

11 Análise de experimento fatorial duplo em DBC

11.1 Entrando com os dados

```
.
#---------------------------------------------------------------------------------------
# dados de experimento em casa de vegetação com soja para verificar o efeito da adubação
# potássica em relação ao nível de água do solo na componentes de produção da cultura
# interesse é saber se o potássio ameniza os efeitos do défict hídrico
#soja <- read.table("http://www.leg.ufpr.br/~walmes/cursoR/cnpaf/soja.txt", header=TRUE)
soja <- read.table("../dados/soja.txt", header=TRUE)
str(soja) # ops!!
soja <- read.table("../dados/soja.txt", header=TRUE, dec=",")
str(soja) # ok!!
soja <- transform(soja, K=factor(potassio), A=factor(agua), bloco=factor(bloco)) # passando para fator
str(soja)
# em outra sessão análisaremos como fatores contínuos
                                                                                        #
#---------------------------------------------------------------------------------------
# análise gráfica
require(lattice)
xyplot(rengrao~K|A, groups=bloco, data=soja, type="b", auto.key=TRUE)
xyplot(pesograo~K|A, groups=bloco, data=soja, type="b", auto.key=TRUE)
                                                                                        #
#---------------------------------------------------------------------------------------
.
```

11.2 Ajuste do modelo e desdobramento da SQ

$$Y_{ijk} = \mu + \iota_i + \alpha_j + \gamma_k + \delta_{jk} + \epsilon_{ijk} \qquad \epsilon_{ijk} \overset{iid}{\sim} \mathcal{N}(0, \sigma^2_\epsilon) \tag{24}$$

$$\boldsymbol{Y} = [\mathbf{1}_\mu | \boldsymbol{X}_\iota | \boldsymbol{X}_\alpha | \boldsymbol{X}_\gamma | \boldsymbol{X}_\delta] \begin{bmatrix} \mu \\ \boldsymbol{\iota} \\ \boldsymbol{\alpha} \\ \boldsymbol{\gamma} \\ \boldsymbol{\delta} \end{bmatrix} + \boldsymbol{\epsilon} \tag{25}$$

```
.
#-------------------------------------------------------------------------------------------
# ajuste do modelo e análise de variância
m0 <- lm(rengrao~bloco+A*K, data=soja)
par(mfrow=c(2,2)); plot(m0); layout(1) # perfeito
summary(m0) # estimativas dos efeitos
anova(m0)   # interação significativa
                                                                                          #
#-------------------------------------------------------------------------------------------
# desdobrando somas de quadrados para a variação de K dentro de A
m1 <- aov(rengrao~bloco+A/K, data=soja) # ajustar modelo aov com "fora/dentro"
summary(m1) # GL e SQ agrupados
coef(m1)    # o padrão dos nomes corresponde aos indices usados no desdobramento
m1$assign   # efeitos da interação correspondem aos índices 3
coef.name <- names(coef(m1))[m1$assign==3]
coef.name # separar os índices de cada nível de A, usando grep isso fica fácil
grep("37.5", coef.name)
list.KinA <- sapply(paste("A",levels(soja$K),sep=""), grep, x=coef.name, simplify=FALSE)
summary(m1, split=list("A:K"=list.KinA))
                                                                                          #
#-------------------------------------------------------------------------------------------
# podemos ainda desdobrar os 4 graus em contrastes de 30, 60, 120 e 180 vs 0
list.KinA <- lapply(list.KinA,
                    function(x){names(x) <- c("0vs30","0vs60","0vs120","0vs180");x})
list.KinA2 <- as.list(unlist(list.KinA))
summary(m1, split=list("A:K"=list.KinA2))
                                                                                          #
#-------------------------------------------------------------------------------------------
# desdobrar A dentro de K
m2 <- aov(rengrao~bloco+K/A, data=soja)
summary(m2)
names(coef(m2))
coef.name <- names(coef(m2))[m2$assign==3]
coef.name # separar os índices de cada nível de A, usando grep isso fica fácil
list.AinK <- sapply(paste("K",levels(soja$K),sep=""), grep, x=coef.name, simplify=FALSE)
summary(m2, split=list("K:A"=list.AinK))
                                                                                          #
#-------------------------------------------------------------------------------------------
# desdobrando os 2 graus de liberdades em contrastes contra o nível mínimo de água
list.AinK <- lapply(list.AinK,
                    function(x){names(x) <- c("37.5vs50.0","37.5vs62.5");x})
list.AinK2 <- as.list(unlist(list.AinK))
summary(m2, split=list("K:A"=list.AinK2))
                                                                                          #
#-------------------------------------------------------------------------------------------
.
```

11.3 Desdobrando a interação em testes de médias

```
.
#-------------------------------------------------------------------------------------------
# vamos aplicar testes de médias usuais, usando as funções da agricolae e SkottKnott
# teste de Tukey (pratique os demais). Comparar níveis de K dentro de A=37.5
require(agricolae)
glr <- df.residual(m0)
qmr <- deviance(m0)/glr
with(subset(soja, A=="37.5"), HSD.test(rengrao, K, glr, qmr))
                                                                                          #
#-------------------------------------------------------------------------------------------
# vamos fazer para todos os níveis de A usando by(), split()+lappply(), plyr::ddply()
desKinA.by <- by(soja, INDICES=soja$A,
                 FUN=function(A){
                   with(A, HSD.test(rengrao, K, glr, qmr))
                 })
desKinA.by
str(desKinA.by)            # formato lista
do.call(rbind, desKinA.by) # coloca um abaixo do outro
                                                                                          #
#-------------------------------------------------------------------------------------------
# usando split()+lapply()
desKinA.sl <- lapply(split(soja, f=soja$A),
                     function(A){
                       with(A, HSD.test(rengrao, K, glr, qmr))
                     })
desKinA.sl
do.call(rbind, desKinA.sl)
```

```
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# usando a plyr::ddply()
require(plyr)
desKinA.dd <- ddply(soja, .(A),
                    .fun=function(A){
                      with(A, HSD.test(rengrao, K, glr, qmr))
                    })
desKinA.dd # resultado já sai no capricho!
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# desdobrando A dentro de K, já vou direto na ddply()
desAinK.dd <- ddply(soja, .(K),
                    .fun=function(K){
                      with(K, HSD.test(rengrao, A, glr, qmr))
                    })
desAinK.dd # resultado já sai no capricho!
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# como juntar tudo em uma tabela de dupla entrada? criar uma id comum aos data.frames
ds <- function(x){gsub(" ", "", as.character(x))} # deleta os espaços (delete spaces)
desKinA.dd <- transform(desKinA.dd, id=paste(ds(A), ds(trt)))
desAinK.dd <- transform(desAinK.dd, id=paste(ds(trt), ds(K)))
desAinK.dd$id
desKinA.dd$id
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# agora é fazer um merge()
desd <- merge(desAinK.dd[c("id","K","means","M")], desKinA.dd[c("id","A","M")], by="id")
desd$M.x <- toupper(ds(desd$M.x))
desd$meansl <- paste(format(desd$means, dig=4)," ", desd$M.x, ds(desd$M.y), sep="")
desd[c("A","K","means","meansl")]
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# fazendo a tabela de dupla entrada
require(reshape)
cast(desd[c("A","K","meansl")], formula=K~A) # minusculas-linhas, maiusculas-colunas
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# gráfico de barras do resultado
barchart(means~K|A, data=desd, horiz=FALSE,
         col=colors()[100], strip=strip.custom(bg="gray90"),
         panel=function(x, y, subscripts, ...){
           panel.barchart(x, y, subscripts=subscripts, ...)
           panel.text(x, y, label=desd$meansl[subscripts], pos=3)
         })
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# fazendo gráfico de barras com barras agrupadas
desd$meansl2 <- paste(format(desd$means, dig=4),"\n", desd$M.x, ds(desd$M.y), sep="")
barchart(means~K, groups=A, data=desd, ylim=c(10, 40), col=cm.colors(6)[1:3],
         panel=function(x, y, subscripts, groups, box.ratio, ...){
           panel.barchart(x, y, subscripts=subscripts, groups=groups, box.ratio=box.ratio, ...)
           d <- 1/(nlevels(groups)+nlevels(groups)/box.ratio)
           g <- (as.numeric(groups[subscripts])-1); g <- (g-median(g))*d
           panel.text(as.numeric(x)+g, y, label=desd$meansl2[subscripts], pos=3)
         })
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# fazer o teste de SkottKnott é quase a mesma coisa, usar lapply()
require(ScottKnott)
sk0 <- sapply(seq(levels(soja$A)), simplify=FALSE,
              function(A){
                sk <- SK.nest(x=soja, y=soja$rengrao, id.trim=10,
                              model="y~bloco+K/A", which="K:A", fl2=A)
                summary(sk)
              })
sk0
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# usando plyr::ldply(), nas formulas o factor ficado vem atrás da "/" (K/A) e do ":" (K:A)
sk0 <- ldply(seq(levels(soja$A)),
              function(A){
                sk <- SK.nest(x=soja, y=soja$rengrao, id.trim=10,
                              model="y~bloco+K/A", which="K:A", fl2=A)
                summary(sk)
              })
sk0
                                                                                                    #
```

```
#-------------------------------------------------------------------------------------
.
```

11.4 Análise usando a função `ExpDes::fat2.rbd()`

```
.
#-------------------------------------------------------------------------------------
# usando ExpDes
require(ExpDes)
with(soja, fat2.rbd(A, K, bloco, rengrao, mcomp="tukey", quali=c(TRUE, TRUE)))
                                                                                    #
#-------------------------------------------------------------------------------------
.
```

11.5 Função para casualização de experimento fatorial em DBC

```
.
#-------------------------------------------------------------------------------------
# fazendo a casualização dos níveis dos tratamentos às unidades experimentais numeradas
# usar a mesma função do DIC, já que a combinação dos níveis pode ser vista como um fator
A <- gl(5, 1, labels="A")      # 5 níveis
B <- gl(3, 1, labels="B")      # 3 níveis
AB <- c(outer(A, B, paste))    # 5*3 = 15 níveis
r <- 6                         # blocos
fat2 <- design.rcbd(AB, r, kinds="Super-Duper")
str(fat2)
write.table(fat2, file="fat2.txt", row.names=FALSE, sep="\t")
write.table(fat2, file="fat2.xls", row.names=FALSE, sep="\t")
                                                                                    #
#-------------------------------------------------------------------------------------
.
```

12 Análise de experimento fatorial com tratamentos adicionais

12.1 Ajuste do modelo para um nível adicional

$$Y_{ijk} = \mu + \iota_i + \alpha_j + \gamma_k + \delta_{jk} + \tau_{j+1} + \epsilon_{ijk} \qquad \epsilon_{ijk} \overset{iid}{\sim} \mathcal{N}(0, \sigma^2_\epsilon) \tag{26}$$

$$\boldsymbol{Y} = [\mathbf{1}_\mu | \boldsymbol{X}_\iota | \boldsymbol{X}_\alpha | \boldsymbol{X}_\gamma | \boldsymbol{X}_\delta | \boldsymbol{X}_\tau] \begin{bmatrix} \mu \\ \boldsymbol{\iota} \\ \boldsymbol{\alpha} \\ \boldsymbol{\gamma} \\ \boldsymbol{\delta} \\ \boldsymbol{\tau} \end{bmatrix} + \boldsymbol{\epsilon} \tag{27}$$

```
.
#-------------------------------------------------------------------------------------
# experimento em fitopatologia testanto óleos essenciais e concetrações de aplicação,
# foi incluída uma testemunha que serve para comparar o efeito dos óleos, até então
# desconhecido sobre a ?severidade da doença.
# dados (segredo está em como montar a planilha, para provocar o confundimento correto)
#fa <- read.table("http://www.leg.ufpr.br/~walmes/cursoR/fat-adi.txt", header=TRUE)
fa <- read.table("../dados/fat-adi.txt", header=TRUE)
str(fa)
names(fa) <- c("tr","or","cc","bl","m") # nomes curtos
fa <- transform(fa, cc=factor(cc), bl=factor(bl))
str(fa)
fa
subset(fa, or=="T" | cc==0) # veja que T e 0 ocorrem nas mesmas linhas, estão confundidos
                                                                                    #
#-------------------------------------------------------------------------------------
# análise de variância para os tratamentos que despreza estrutura fatorial-adicional, nessa
# fazer vamos chegar os resíduos
```

```
m0 <- lm(m~bl+tr, fa)
par(mfrow=c(2,2)); plot(m0); layout(1)
anova(m0)
                                                                                #
#-------------------------------------------------------------------------------
# é comum nesses experimentos o uso de notas/escalar de medida. Deve-se tomar o cuidado de
# garantir a independencia entre unidades experimentais. As notas, por mais que se evite,
# são comparativas. Outro ponto seria o experimento ser conduzído "as cegas", ou seja, sem
# o avaliador saber qual tratamento foi aplicado a unidade experimental avaliada. Usar a
# média como resposta é importante visto que as notas são discretas (ex 1 à 5) e as médias,
# a medida que se tem muitas plantas por tratamento, tende a ser contínua. Pode-se adotar
# uma escala contínua fazendo um risco em um segmento não graduado em uma folha de papel.
# É importante aleatorizar a ordem de avaliação das plantas a cada coleta.
                                                                                #
#-------------------------------------------------------------------------------
.
```

12.2 Desdobramento das SQ

```
.
#-------------------------------------------------------------------------------
# vamos desdobrar a SQ de tr em "efeitos dos níveis fatoriais" e "níveis fatoriais vs nível
# adicional". Porque? Só porque é um contraste ortogonal. A hipótese com relação a
# testemunha pode não ser essa, ou seja, de contrastar com a *média* dos níveis fatoriais
                                                                                #
#-------------------------------------------------------------------------------
# tipos de testemunha:
# * negativa: aquela que se sabe antemão que dará resultado ruim, queremos saber se os
#   níveis avaliados são melhores (hipótese unilateral), é o caso da testemunha sem capina
#   em experimentos com herbicida e da testemunha em aplicação na fitopatologia;
# * posititva: aquela que possivelmente dará resultado melhor ou igual aos níveis estudados
#   é o caso da testemunha capinada à mão em experimentos com herbicida (não sofre o efeito
#   negativo da toxidade);
# * padrão: é o tipo de manejo padrão, o experimento é feito para encontrar alguém superior
#   ao padrão, é o caso das cultivares já usadas em uma região, ou da dose já usada de um
#   produto;
                                                                                #
#-------------------------------------------------------------------------------
# as matrizes de contrastes envolvidas
contrasts(fa$tr)
contrasts(fa$or)
contrasts(fa$cc)
                                                                                #
#-------------------------------------------------------------------------------
# vamos usar o contraste de Helmet que já tá pronto
contrasts(C(factor(1:5), helmert)) # último nível vs os anteriores
                                                                                #
#-------------------------------------------------------------------------------
# passar os níveis que represetam a testemunha para a última posição
fa$cc <- factor(fa$cc, levels=c("25","50","75","0"))
contrasts(C(fa$cc, helmert)) # nível 0 vs demais
levels(fa$cc)                # basta fazer em um dos fatores
fa$cc <- factor(fa$cc, levels=c("25","50","75","0"))
levels(fa$or)                # testemunha já é o último nível
                                                                                #
#-------------------------------------------------------------------------------
# anova só da parte fatorial, isso para vermos as SQ e conferir nosso desdobramento
anova(m0)
m1 <- aov(m~bl+or*cc, data=subset(fa, tr!="TEST")) # exlui a TEST
summary(m1)
                                                                                #
#-------------------------------------------------------------------------------
# anova com fornecimento dos contrates e "arrancando" a SQ do contraste com o adicional
# da SQ do fator origem
m2 <- aov(m~bl+or*cc, data=fa,
          contrast=list(or=contr.helmert, cc=contr.helmert))
summary(m2) # or tem 3 gl mas deveria ter 2, isso porque test tá junto dessa SQ
summary(m2, expand.split=FALSE,
        split=list("or"=list("fatorial"=c(1:2), "adicional"=3))) # separa as SQs
                                                                                #
#-------------------------------------------------------------------------------
# nessa análise a anova diz tudo: concentação não depende do óleo, as concetrações e óleos
# não diferem entre sí, mas na média de seus efeitos, são diferentes da testemunha
                                                                                #
#-------------------------------------------------------------------------------
# teste de média da test contra as origens na menor concentração (testemunha negativa)
```

```
require(agricolae)
glr <- df.residual(m2)
qmr <- deviance(m2)/glr
with(subset(fa, cc%in%c("0","25")), HSD.test(m, or, glr, qmr))
with(subset(fa, cc%in%c("0","50")), HSD.test(m, or, glr, qmr))
with(subset(fa, cc%in%c("0","75")), HSD.test(m, or, glr, qmr))
                                                                                    #
#-----------------------------------------------------------------------------------
# uma hipótese a ser testada é saber se a testemunha difere das cc na média dos or
fa$cc <- factor(fa$cc, levels=c("0","25","50","75"))
m3 <- lm(m~bl+cc, data=fa)
summary(m3) # test difere das concentrações, mas estas não diferem entre sí
par(mfrow=c(2,2)); plot(m3); layout(1)
anova(m3, m2)  # não tem falta de ajuste, posso fazer o abandono de termos
                                                                                    #
#-----------------------------------------------------------------------------------
# ou testar se a test difere da média dos fatorais (só porque não teve efeito de nada)
require(contrast)
c0 <- contrast(m3, type="average", list(bl=levels(fa$bl), cc="0")) # média da testemunha
print(c0, X=TRUE)
c1 <- contrast(m3, type="average", list(bl=levels(fa$bl), cc=c("25","50","75"))) # média cc
print(c1, X=TRUE)
c2 <- contrast(m3, type="average",
               list(bl=levels(fa$bl)[1], cc="0"),
               list(bl=levels(fa$bl)[1], cc=c("25","50","75")))
print(c2, X=TRUE)
                                                                                    #
#-----------------------------------------------------------------------------------
# o intervalo de confiança da diferença é muito útil, da noção do ganho em controle da
# doença com uso de qualquer concentração (entre 25 e 75) de qualquer óleo.
                                                                                    #
#-----------------------------------------------------------------------------------
# o que fazer nos próximos experimentos? Usar doses menores, tentar encontrar a dose
# limitante. Se for para estudar fatores de níveis contínuos, use mais níveis, explore a
# região experimental usando mais níveis para melhor entender/modelar a relação entre a
# resposta e a covariável. Teste de médias para fatores contínuos é gastar dinheiro e tempo
# com algo e não usufruir dos benefícios. Deve-se planejar.
                                                                                    #
#-----------------------------------------------------------------------------------
.
```

12.3 Análise usando a função `ExpDes::fat2.ad.rbd()`

```
.
#-----------------------------------------------------------------------------------
# carrega o pacote
require(ExpDes)
fa.fat <- fa[fa$tr!="TEST",] # tem que separar nas porções fatorial
fa.adi <- fa[fa$tr=="TEST",] # e adicional
fat2.ad.rbd(fa.fat$or, fa.fat$cc, fa.fat$bl, fa.fat$m, fa.adi$m)
# essa função compara a média da testemunha com a média do fatorial. Na maioria dos casos
# será uma hipótese sem significado, como por exemplo no caso em que houver interação entre
# os fatores.
                                                                                    #
#-----------------------------------------------------------------------------------
.
```

12.4 Ajuste do modelo para dois níveis adicionais

```
.
#-----------------------------------------------------------------------------------
# experimento com óleos essenciais aplicados sob duas formas onde observou a área sobre a
# curve de progresso da doença. Existem dois níveis adicionais (test - e test +). Colunas
# começadas em "a" são as notas datas em cada avaliação. Teremos que calculas a aacpd.
ol <- read.table("../dados/oleos.txt", header=TRUE, sep="\t")
str(ol)
ol$bloco <- factor(ol$bloco)
                                                                                    #
#-----------------------------------------------------------------------------------
# análise gráfica, passar dados para o formato longo
require(reshape)
aux <- melt(ol, id.vars=1:3, variable_name="avaliacao")
```

```
str(aux)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# gráfico
require(lattice)
xyplot(value~avaliacao|oleo, groups=forma, data=aux, type=c("p","a"))
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# obter a área abaixo da curva de progresso da doença
daval <- c("2010-05-13","2010-05-20","2010-05-27","2010-06-03","2010-06-10") # datas aval
daval <- as.Date(daval)
t <- cumsum(c(0, diff(daval))) # dias após primeira avaliação
t # foi semanal
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# fazendo o calculo da audpc com a agricolae::audpc() e plyr::ddply()
require(agricolae)
ol$audpc <- 0
for(i in 1:nrow(ol)){ ol$audpc[i] <- audpc(evaluation=ol[i,4:8], dates=t) }
str(ol)
xyplot(audpc~oleo, groups=forma, data=ol, type=c("p","a"))
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# ajuste do modelo
m0 <- lm(audpc~bloco+oleo*forma, data=ol)
par(mfrow=c(2,2)); plot(m0); layout(1)
anova(m0) # se deu interação não preciso nem desdobrar para ver os efeitos principais
                                                                                                    #
#-------------------------------------------------------------------------------------------------
.
```

12.5 Desdobrando a SQ

```
.
#-------------------------------------------------------------------------------------------------
# farei o desdobramento da SQ para apresentar o procedimento
# ordem dos níveis deve ser mudada, deixar assim A, C, H, fung(positiva), test(negativa)
levels(ol$oleo)
ol$oleo <- factor(ol$oleo, levels=c("A","C","H","fung","test"))
levels(ol$oleo)
levels(ol$forma)
ol$forma <- factor(ol$forma, levels=c("extrato","oleo","fung","test"))
levels(ol$forma)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# fazer a análise usando contr.helmert porque faz com que último confronte com os demais
m1 <- aov(audpc~bloco+oleo*forma, data=ol,
          contrast=list(oleo=contr.helmert, forma=contr.helmert))
summary(m1, split=list(oleo=list("fat"=1:2, "fung"=3, "test"=4)))
summary(m1, expand.split=FALSE, split=list(oleo=list("fat"=1:2, "fung"=3, "test"=4)))
# dado que existe interação, nem olhe para as SQ de efeitos principais, não tem significado
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# desdobrar a interação dentro da anova, forma dentro de oleo
m2 <- aov(audpc~bloco+oleo/forma, data=ol,
          contrast=list(oleo=contr.helmert, forma=contr.helmert))
summary(m2)
coef(m2)
summary(m2, expand.split=FALSE,
        split=list(
          "oleo"=list("fung"=3, "test"=4, "fat"=1:2),
          "oleo:forma"=list(A=1, B=2, C=3)))
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# desdobrar a interação dentro da anova, oleo dentro de forma
m3 <- aov(audpc~bloco+forma/oleo, data=ol,
          contrast=list(oleo=contr.helmert, forma=contr.helmert))
summary(m3)
coef(m3)
summary(m3, expand.split=FALSE,
        split=list(
          "forma"=list("fung"=2, "test"=3, "fat"=1),
          "forma:oleo"=list(extrato=c(1,3), oleo=c(2,4))))
                                                                                                    #
#-------------------------------------------------------------------------------------------------
.
```

12.6 Testes de média

```
.
#-------------------------------------------------------------------------------------
# aplicando testes de médias ignorando a estrutura fatorial, visto que houve interação
require(agricolae)
ol$tr <- with(ol, interaction(oleo, forma))
glr <- df.residual(m0)
qmr <- deviance(m0)/glr
with(ol, HSD.test(audpc, tr, glr, qmr))
#-------------------------------------------------------------------------------------
# fazendo os contrastes
m4 <- aov(audpc~bloco+oleo*forma, data=ol,
          contrast=list(oleo=contr.helmert, forma=contr.helmert))
summary.lm(m4) # omite os NA e a gmodels::estimable() pode ser usada
# como escrever os contrastes nessa parametrização????
require(gmodels)
estimable(m4, rep(c(1,0), c(1,9)))
                                                                                    #
#-------------------------------------------------------------------------------------
# mudar para uma opção mais fácil: ajustar o modelo de efeito de médias
levels(ol$tr)
ol$tr <- factor(ol$tr, levels=rev(levels(ol$tr))) # faz a testemunha ser o primeiro nível
m5 <- lm(audpc~bloco+tr, data=ol)
coef(m5) # aqui é mais fácil estabelecer os contrastes, podemos usar a contrast()
                                                                                    #
#-------------------------------------------------------------------------------------
# fazendo contrastes contra a testemunha negativa e positiva
tr.lev <- names(coef(m5))[m5$assign==2]
tr.lev <- gsub("^tr", "", tr.lev)
tr.lev
contrast(m5, list(bloco="1", tr="test.test"), list(bloco="1", tr=tr.lev))
contrast(m5, list(bloco="1", tr=tr.lev[1]), list(bloco="1", tr=tr.lev[-1]))
                                                                                    #
#-------------------------------------------------------------------------------------
.
```

12.7 Níveis adicionais e um fator contínuo

```
.
#-------------------------------------------------------------------------------------
# dados de mistura de duas componentes: restrição que a soma dá um, predição é para x
# no intervalo unitário, reparemetrização torna mais clara a interpretação
mis <- read.table("../dados/biomassa.txt", header=TRUE, sep="\t")
str(mis)
names(mis) <- tolower(names(mis)) # passando nomes das colunas para minúsculo
                                                                                    #
#-------------------------------------------------------------------------------------
# por enquanto vamos análisar apenas uma espécie
mis <- subset(mis, pla=="E")
str(mis)
                                                                                    #
#-------------------------------------------------------------------------------------
# fazendo o gráficos de todas as respostas
require(reshape)
aux <- melt(mis, id.vars="k", measure.vars=8:20, variable_name="resposta") # empilhamento
str(aux)
                                                                                    #
#-------------------------------------------------------------------------------------
# gráfico
require(lattice)
xyplot(value~k|resposta, data=aux, type=c("p","a","smooth","r"), scales="free")
                                                                                    #
#-------------------------------------------------------------------------------------
# análise do modelo de regressão quadrático em K com parametrização usual e adicionais
str(mis)
m0 <- lm(dc~bj+naplus+k+I(k^2), data=mis)
summary(m0)
par(mfrow=c(2,2)); plot(m0); layout(1)
                                                                                    #
#-------------------------------------------------------------------------------------
# análise do modelo de regressão (quadrático) mas com a parametrização de mistura
m1 <- lm(dc~-1+k+na+k:na+bj+naplus, data=mis)
m1 <- lm(dc~-1+k+I(1-k)+k:I(1-k)+bj+naplus, data=mis)
summary(m1) # vantagem é a interpretação dos parâmetros
```

```
par(mfrow=c(2,2)); plot(m1); layout(1)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# gráfico do ajuste do modelo
pred <- data.frame(k=seq(0,1,l=25), bj=0, naplus=0)
pred <- rbind(pred, c(0,1,0), c(0,0,1))
pred$y <- predict(m1, newdata=pred, interval="confidence")
mis$kaux <- mis$k
mis$kaux[mis$bj==1] <- 1.3
mis$kaux[mis$naplus==1] <- 1.6
pred$kaux <- pred$k
pred$kaux[pred$bj==1] <- 1.3
pred$kaux[pred$naplus==1] <- 1.6
plot(dc~kaux, data=mis)
with(subset(pred, bj<1 & naplus<1), matlines(kaux, y, type="l", col=c(1,2,2), lty=c(1,2,2)))
with(subset(pred, bj==1), arrows(kaux, y[,2], kaux, y[,3], code=3, angle=90, length=0.1))
with(subset(pred, naplus==1), arrows(kaux, y[,2], kaux, y[,3], code=3, angle=90, length=0.1))
                                                                                                    #
#-----------------------------------------------------------------------------------------------
.
```

12.8 Contrastes da resposta no máximo contra as testemunhas

```
.
#-----------------------------------------------------------------------------------------------
# a dose de melhor mistura é diferente dos componentes puros?
coef(m1)
kmax <- (2*m1$coef[5]/(m1$coef[5]+m1$coef[1]-m1$coef[2]))^(-1)
names(kmax) <- NULL
abline(v=kmax)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# usando a gmodels::estimable(), detalhe: o vetor passado não pode ter atributo names
require(gmodels)
ymax <- rbind("max"=c(kmax,1-kmax,0,0,kmax*(1-kmax)))
ytes <- rbind("bj"=c(0,1,1,0,0), "naplus"=c(0,1,0,1,0))
ytes-rbind(ymax, ymax)
estimable(m1, cm=rbind(ymax, ytes))      # estimativas
estimable(m1, cm=ytes-rbind(ymax, ymax)) # contrastes contra o máximo
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# usando a contrast::contrast(), vantagem: simples de usar
require(contrast)
contrast(m1,
         list("k"=kmax, "I(1 - k)"=1-kmax, bj=0, naplus=0),
         list("k"=0, "I(1 - k)"=1-0, bj=0, naplus=1))
contrast(m1,
         list("k"=kmax, "I(1 - k)"=1-kmax, bj=0, naplus=0),
         list("k"=0, "I(1 - k)"=1-0, bj=1, naplus=0))
                                                                                                    #
#-----------------------------------------------------------------------------------------------
.
```

12.9 Função para casualização de experimento fatorial com adicionais

```
.
#-----------------------------------------------------------------------------------------------
# fazendo a casualização dos níveis dos tratamentos às unidades experimentais numeradas
# usar a mesma função do DIC, já que a combinação dos níveis pode ser vista como um fator
A <- gl(5, 1, labels="A")           # 5 níveis        |
B <- gl(3, 1, labels="B")           # 3 níveis        | porção factorial
AB <- c(outer(A, B, paste))         # 5*3 = 15 níveis |
tr <- c(AB, "test-neg", "test-pos") # adiciona os níveis de testemunha
r <- 6                          # blocos
fatadi <- design.rcbd(tr, r, kinds="Super-Duper") # se for em blocos
fatadi <- design.crd(tr, r, kinds="Super-Duper")  # se não for em blocos
str(fatadi)
write.table(fatadi, file="fatadi.txt", row.names=FALSE, sep="\t")
write.table(fatadi, file="fatadi.xls", row.names=FALSE, sep="\t")
                                                                                                    #
#-----------------------------------------------------------------------------------------------
.
```

13 Análise de covariância

13.1 Ajuste do modelo

$$Y_{ijk} = \mu + \beta \times z_{ijk} + \alpha_i + \gamma_j + \delta_{ij} + \epsilon_{ijk} \qquad \epsilon_{ijk} \overset{iid}{\sim} \mathcal{N}(0, \sigma^2_\epsilon) \tag{28}$$

$$\boldsymbol{Y} = [\mathbf{1}_\mu | \boldsymbol{Z}_\beta | \boldsymbol{X}_\alpha | \boldsymbol{X}_\gamma | \boldsymbol{X}_\delta] \begin{bmatrix} \mu \\ \boldsymbol{\beta} \\ \boldsymbol{\alpha} \\ \boldsymbol{\gamma} \\ \boldsymbol{\delta} \end{bmatrix} + \boldsymbol{\epsilon} \tag{29}$$

```
.
#-------------------------------------------------------------------------------------------
# existe 2 tipos de análise de covariância:
# * a covariável funciona como bloco, entra no modelo com o objetivo de explicar parte da
# variação, é uma variável não controlada pelo pesquisador, assume-se não interagir com os
# efeitos de interesse, normalmente entra apenas com termo linear. Trataremos isso agora.
# * a covariável é controlada (ou parcialmente) pelo pesquisar, há o interesse de saber sua
# interação com outros termos, pode-se estudar diferentes especificações de modelo.
                                                                                          #
#-------------------------------------------------------------------------------------------
# dados de experimento com nutrição de suínos. Animais foram pesados antes do experimento e
# tinham idade conhecida. Essas variáveis contínuas foram usadas para explicar/corrigir
# parte da variação presente e melhor comparar os níveis de energia na ração fornecidos
#ac <- read.table("http://www.leg.ufpr.br/~walmes/cursoR/ancova.txt", header=TRUE)
sui <- read.table("../dados/ancova.txt", header=TRUE)
str(sui)
                                                                                          #
#-------------------------------------------------------------------------------------------
# número de animais para cada combinação de níveis de sexo e energia
with(sui, tapply(peso28, list(sexo, energia), length))
                                                                                          #
#-------------------------------------------------------------------------------------------
# gráficos, distribuição das id e pi nos grupos formados pelos fatores categóricos
require(lattice)
xyplot(pi~id|sexo, groups=energia, data=sui, cex=2, pch=19, auto.key=TRUE)
xyplot(pi~id|energia, groups=sexo, data=ac, cex=2, pch=19, auto.key=TRUE)
                                                                                          #
#-------------------------------------------------------------------------------------------
# pode-se fazer uma anova dessas covariáveis para verificar se elas são separadas ou
# confundidas com os fatores experimentais
anova(aov(pi~sexo*energia, data=sui)) # testa se pi é separada por sexo*energia
anova(aov(id~sexo*energia, data=sui)) # testa se id é separada por sexo*energia
# não significativo é o resultado esperado se os tratamentos foram casualizados aos animais
                                                                                          #
#-------------------------------------------------------------------------------------------
# análise de variância (em experimentos não ortogonais a ordem dos termos é importante!)
m0 <- aov(peso28~sexo*energia, data=sui)        # modelo só com os fatores categoricos
summary(m0)
m1 <- aov(peso28~pi+id+sexo*energia, data=sui)  # modelo com os categóricos e contínuos
summary(m1) # veja redução na SQ, parte da varição é explicada por pi e id
m1 <- aov(peso28~id+pi+sexo*energia, data=sui)
summary(m1)
anova(m0, m1) # testa o poder de explicação dos "blocos" contínuos
                                                                                          #
#-------------------------------------------------------------------------------------------
# checagem
par(mfrow=c(2,2)); plot(m1); layout(1)
                                                                                          #
#-------------------------------------------------------------------------------------------
# estimativas
summary.lm(m1)
#-------------------------------------------------------------------------------------------
# dado que há efeito de sexo após correção da variação para pi e id, fazer teste de médias
# deve-se escolher o valor das covariáveis a ser fixado para comparar os níveis de sexo
mean(sui$pi) # média amostral de peso inicial dos animais do experimento (menor erro padrão)
mean(sui$id) # média amostral de idade dos animais do experimento (menor erro padrão)
                                                                                          #
#-------------------------------------------------------------------------------------------
# ajuste do modelo com a função lm
m0 <- lm(peso28~pi+id+sexo*energia, data=ac)
anova(m0)
```

```
                                                                                                    #
#-----------------------------------------------------------------------------------------------
.
```

13.2 Contrastes entre níveis fixando as covariáveis

```
.
#-----------------------------------------------------------------------------------------------
# para fazer os contrastes entre níveis de sexo, para um animal com peso e idade médios
require(contrast)
levels(sui$sexo)    # níveis do fator
levels(sui$energia) # níveis do fator
pim <- mean(sui$pi)
idm <- mean(sui$id)
m1 <- lm(peso28~id+pi+sexo*energia, data=sui) # modelo deve ser de classe lm para contrast
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# femea vs macho castrado (observe que os erros padrões dos contrastes são diferentes)
c0 <- contrast(m1, type="average",
               list(sexo="F", energia=levels(sui$energia), pi=pim, id=idm),
               list(sexo="MC", energia=levels(sui$energia), pi=pim, id=idm))
c0
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# femêa vs macho imunocatrado
c1 <- contrast(m1, type="average",
               list(sexo="F", energia=levels(sui$energia), pi=pim, id=idm),
               list(sexo="MI", energia=levels(sui$energia), pi=pim, id=idm))
c1
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# macho castrado vs macho imunocastrado
c2 <- contrast(m1, type="average",
               list(sexo="MI", energia=levels(sui$energia), pi=pim, id=idm),
               list(sexo="MC", energia=levels(sui$energia), pi=pim, id=idm))
c2
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# passando os contrastes para a multcomp corrigir os p-valores
cm <- rbind(c0$X, c1$X, c2$X)
rownames(cm) <- c("FvsMC","FvsMI","MIvsMC")
require(multcomp)
summary(glht(m1, linfct=cm))
#-----------------------------------------------------------------------------------------------
# as médias marginais populacionais de sexo na média das 3 rações com pi e id médios
med <- sapply(levels(sui$sexo), simplify=FALSE,
              function(s){
                contrast(m1, type="average",
                         list(sexo=s, energia=levels(sui$energia), pi=pim, id=idm))[1:7]
              })
str(med)
med <- do.call(rbind, lapply(med, data.frame))
med <- med[order(med$Contrast),]
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# gráfico de barras com IC para a média
require(gplots)
bp <- with(med, barplot2(Contrast, ylim=c(120, 130), xpd=FALSE, plot.ci=TRUE,
                         ci.l=Lower, ci.u=Upper, ylab="Peso aos 28 dias"))
axis(1, at=bp, labels=rownames(med), tick=FALSE)
box()
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# gráfico de barras com as médias e resultado da comparação
bp <- barplot(med$Contrast, ylim=c(120, 130), xpd=FALSE, ylab="Peso aos 28 dias")
text(bp, med$Contrast,
     label=paste(format(med$Contrast, dig=5), c("ab","b","a")), pos=3)
axis(1, at=bp, labels=rownames(med), tick=FALSE)
box()
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# gráfico com IC e médias
bp <- with(med, barplot2(Contrast, ylim=c(120, 130), xpd=FALSE, plot.ci=TRUE,
                         ci.l=Contrast, ci.u=Upper, ylab="Peso aos 28 dias"))
axis(1, at=bp, labels=rownames(med), tick=FALSE)
text(bp, med$Contrast,
```

```
     label=paste(format(med$Contrast, dig=5), c("b","ab","a")), pos=1)
box()
                                                                                                    #
#-------------------------------------------------------------------------------------------
.
```

14 Regressão polinomial na análise de variância

14.1 Ajuste do modelo

$$Y_{ijk} = \mu + \iota_i + \alpha_j + \beta \times z_{ijk} + \beta_j \times z_{ijk} + \epsilon_{ijk} \qquad \epsilon_{ijk} \overset{iid}{\sim} \mathcal{N}(0, \sigma^2_\epsilon) \tag{30}$$

```
.
#-------------------------------------------------------------------------------------------
# esse é segundo caso de análise de covariância. Devido ao fato de que os níveis do fator
# contínuo serem os mesmos, pode também ser entendido como um experimento fatorial.
                                                                                                    #
#-------------------------------------------------------------------------------------------
# dados de índice agronomico de cultivares de sorgo em função da adubação
# sorgo <- read.table("http://www.leg.ufpr.br/~walmes/cursoR/cnpaf/anovareg.txt", header=TRUE)
sorgo <- read.table("../dados/anovareg.txt", header=TRUE)
sorgo <- transform(sorgo, bloco=factor(bloco), cultivar=factor(cultivar))
str(sorgo)
                                                                                                    #
#-------------------------------------------------------------------------------------------
# gráficos exploratórios
require(lattice)
xyplot(indice~dose|cultivar, groups=bloco, data=sorgo,
       jitter.x=TRUE, type=c("p","l"), layout=c(3,1))
xyplot(indice~dose, groups=cultivar, data=sorgo, jitter.x=TRUE, type=c("p","a"))
                                                                                                    #
#-------------------------------------------------------------------------------------------
# análise de variância do modelo de fatores (desconsidera que dose é métrica)
# os níveis são igualmente espaçados, podemos usar contr.poly, default para ordered()
sorgo$ds <- ordered(sorgo$dose)
m0 <- aov(indice~bloco+cultivar*ds, data=sorgo)
par(mfrow=c(2,2)); plot(m0); layout(1)
summary(m0)
coef(m0) # note os sufixos .L, .Q, .C, ^4 e ^5, são os efeitos das tendências
                                                                                                    #
#-------------------------------------------------------------------------------------------
# veja que o fator contínuo tem que ser igualmente espaçado, se não for...
x1 <- c(1,2,3,4,5) # regular
x2 <- c(1,2,4,5,7) # irregular
contrasts(ordered(x2))       # mesma matriz? polinômios ortogonais é para o caso regular
cm <- contrasts(ordered(x1)) # o que quer dizer ortogonais?
round(t(cm)%*%cm, 5)         # que o conjunto de colunas gerados é ortogonal
# qual a vantagem? estabilidade numérica, mas a interpretação das estimativas é difícil
                                                                                                    #
#-------------------------------------------------------------------------------------------

.
```

14.2 Desdobramento da interação em polinômios

```
.
#-------------------------------------------------------------------------------------------
# desdobrar os efeitos polinomiais dentro dos níveis de cultivar
m1 <- aov(indice~bloco+cultivar/ds, data=sorgo)
summary(m1)
m1$assign
coef.name <- names(coef(m1)[m1$assign==3]); coef.name
DinC.list <- sapply(levels(sorgo$cultivar), grep, x=coef.name, simplify=FALSE)
DinC.list
summary(m1, split=list("cultivar:ds"=DinC.list))
                                                                                                    #
#-------------------------------------------------------------------------------------------
# podemos desdobrar mais, em efeitos L, Q e falta de ajuste (C, ^4 e ^5)
DinC.list2 <- sapply(DinC.list, simplify=FALSE,
```

```
                    function(x){
                      list(L=x[1], Q=x[2], lof=x[3:5])
                    })
DinC.list2 <- unlist(DinC.list2, recursive=FALSE)
summary(m1, split=list("cultivar:ds"=DinC.list2)) # ops! Pioner pede mais
                                                                                    #
#-------------------------------------------------------------------------------------
# vamos desdobrar até o 3
DinC.list3 <- sapply(DinC.list, simplify=FALSE,
                    function(x){
                      list(L=x[1], Q=x[2], C=x[3], lof=x[4:5])
                    })
DinC.list3 <- unlist(DinC.list3, recursive=FALSE)
summary(m1, split=list("cultivar:ds"=DinC.list3))
                                                                                    #
#-------------------------------------------------------------------------------------
# temos 2 polinômios quadráticos e 1 cúbico, o que fazer?
                                                                                    #
#-------------------------------------------------------------------------------------
.
```

14.3 Predição sem efeito dos blocos

```
.
#-------------------------------------------------------------------------------------
# vamos ajustar o polinômio com 2 graus para todos os cultivares
m1 <- lm(indice~bloco+cultivar*(dose+I(dose^2)), data=sorgo,
         contrast=list(bloco=contr.sum))
coef(m1)
                                                                                    #
#-------------------------------------------------------------------------------------
# iremos considerar o desempenho de todos os cultivares no bloco I (poderia ser outro)
# depois subtraímos o efeito desse bloco que é
coef(m1)[2]
pred <- expand.grid(bloco="I", cultivar=levels(sorgo$cultivar), dose=seq(0,300,by=5))
aux <- predict(m1, newdata=pred, interval="confidence")-coef(m1)[2]
pred <- cbind(pred, as.data.frame(aux))
                                                                                    #
#-------------------------------------------------------------------------------------
# fazendo o gráfico da predição
require(latticeExtra) # para usar as.layer()
xyplot(indice~dose|cultivar, data=sorgo, jitter.x=TRUE, layout=c(3,1))+
  as.layer(xyplot(fit+lwr+upr~dose|cultivar, data=pred,
           type="l", col=c(1,2,2), lty=c(1,2,2)))
                                                                                    #
#-------------------------------------------------------------------------------------
.
```

14.4 Ajuste com polinômios de diferentes graus

```
.
#-------------------------------------------------------------------------------------
# ajustar dois qudráticos e um cúbico, usar apenas as colunas de X relativas aos efeitos
m1 <- aov(indice~bloco+cultivar/ds, data=sorgo)
X <- model.matrix(m1)
colnames(X)
Xred <- X[,c(2:12,15)]
m2 <- lm(indice~Xred, data=sorgo) # sim, a lm rece formula ou matriz
anova(m2)                         # todas as colunas agrupadas
summary(m2)
                                                                                    #
#-------------------------------------------------------------------------------------
# não há falta de ajuste
anova(m2, m0) # 3 de ^5, 3 de ^4 e 2 de .C, somando dá 8 graus de liberdade
                                                                                    #
#-------------------------------------------------------------------------------------
# mas os coeficientes são não interpretáveis porque a matriz usada é de poli orotgonais
                                                                                    #
#-------------------------------------------------------------------------------------
.
```

14.5 Obtenção das equações de regressão

```
.
#-------------------------------------------------------------------------------------
# ajustar sem ser polinômios ortogonais, usar a dose que é numérica
m3 <- aov(indice~-1+cultivar/(dose+I(dose^2)+I(dose^3))+bloco, data=sorgo)
X <- model.matrix(m3)
colnames(X)
Xred <- X[,-c(13:14)]
m4 <- lm(indice~-1+Xred, data=sorgo)
anova(m4, m0) # mesmo ajuste, só que sem polinômios ortogonais
summary(m4)   # apesar de ser o coeficiente do polinômio, não tem interpretação biológica
              # basta olhar os sinais e o intercepto
                                                                                    #
#-------------------------------------------------------------------------------------
# separando os coeficientes para cada cultivar
aux <- sapply(levels(sorgo$cultivar), grep, names(coef(m4)))
sapply(aux, function(x) summary(m4)$coeff[x,]) # tabela de estimativas
sapply(aux, function(x) round(coef(m4)[x],7))  # só as estimativas
                                                                                    #
#-------------------------------------------------------------------------------------

.
```

14.6 Predição com diferentes graus

```
.
#-------------------------------------------------------------------------------------
# para fazer a predição temos que remover o efeito dos blocos, fácil usando contr.sum
m3 <- aov(indice~bloco+cultivar/(dose+I(dose^2)+I(dose^3)), data=sorgo,
          contrast=list(bloco=contr.sum))
X <- model.matrix(m3)
colnames(X)
Xred <- X[,-c(13:14)]
m4 <- lm(indice~-1+Xred, data=sorgo)
anova(m4, m0) # mesmo ajuste, só que sem polinômios ortogonais
summary(m4)   # apesar de ser o coeficiente do polinômio, não tem interpretação biológica
              # basta olhar os sinais e o intercepto
                                                                                    #
#-------------------------------------------------------------------------------------
# para usar a predict com precição o modelo deve ser ajustado com fórmula
Xdata <- as.data.frame(Xred)
str(Xdata)
m4 <- lm(sorgo$indice~-1+., data=Xdata)
                                                                                    #
#-------------------------------------------------------------------------------------
# agora vamos preparar um data.frame com grid fino para fazermos a predição
levels(sorgo$bloco) # o efeito de blocos não vai entrar, mas deve estar no data.frame
range(sorgo$dose)
pred <- expand.grid(bloco="I", cultivar=levels(sorgo$cultivar), dose=seq(0,300,by=5))
str(pred)
Xpred <- model.matrix(~cultivar/(dose+I(dose^2)+I(dose^3)), pred)
Xpred <- Xpred[,-c(10:11)] # retira os efeitos cúbicos
Xpred <- cbind(Xpred[,1], 0, 0, 0, Xpred[,-1])
colnames(Xpred) <- colnames(Xred)
Xpred <- as.data.frame(Xpred)
                                                                                    #
#-------------------------------------------------------------------------------------
# fazendo a predição com IC
aux <- predict(m4, newdata=Xpred, interval="confidence")
pred <- cbind(pred, as.data.frame(aux))
xyplot(fit+lwr+upr~dose|cultivar, data=pred)
                                                                                    #
#-------------------------------------------------------------------------------------
# fazendo o gráfico da predição
require(latticeExtra) # para usar as.layer()
xyplot(indice~dose|cultivar, data=sorgo, jitter.x=TRUE, layout=c(3,1))+
  as.layer(xyplot(fit+lwr+upr~dose|cultivar, data=pred,
           type="l", col=c(1,2,2), lty=c(1,2,2)))
                                                                                    #
#-------------------------------------------------------------------------------------
.
```

14.7 A questão do R^2

```
.
#-------------------------------------------------------------------------------------------
# o R² do nosso ajuste, que é conjunto, é obtido por
m4 <- lm(indice~Xred[,-1], data=sorgo) # modelo deve colocar o seu intercepto
coef(m4)
summary(m4)
summary(m4)$r.squared     # R² ordinário
summary(m4)$adj.r.squared # R² ajustado
                                                                                          #
#-------------------------------------------------------------------------------------------
# como obter o R² para cada curva (se é que isso é informativo)
# R² = 1-SQR/SQTc = 1-(desvios da curva)^2/(desvios da media)^2
str(sorgo)
sorgo$indice.aj <- fitted(m4)
by(sorgo, INDICES=sorgo$cultivar,
   function(x){
     1-sum(c(x$indice-x$indice.aj)^2)/sum(c(x$indice-mean(x$indice))^2)
   })
                                                                                          #
#-------------------------------------------------------------------------------------------
# tem aplicativo que solta um R² nos desvios da média (que pra mim está ERRADO)
by(sorgo, INDICES=sorgo$cultivar,
   function(x){
     ma <- tapply(x$indice, x$dose, mean)
     aj <- tapply(x$indice.aj, x$dose, mean)
     1-sum(c(ma-aj)^2)/sum(c(ma-mean(ma))^2)
   })
# esse R² não representa a proporção que o modelo explica da variabilidade dos dados
# e sim da variabilidade da média. Nossa inferência é para a população de onde retiramos
# os dados e não para a população da média dos dados. Acredito que esse foi algum artifício
# usado para ter maiores valores de R². Além do mais, esse procedimento só é possível se
# houver repetição de níveis, coisa que não ocorre em estudos observacionais.
                                                                                          #
#-------------------------------------------------------------------------------------------
.
```

14.8 Análise usando a função `ExpDes::fat2.crb()`

```
.
#-------------------------------------------------------------------------------------------
# tudo o que eu fiz em uma única linha de comando
require(ExpDes)
with(sorgo, fat2.rbd(cultivar, dose, bloco, indice, quali=c(TRUE, FALSE)))
                                                                                          #
#-------------------------------------------------------------------------------------------
.
```

```
.
#-------------------------------------------------------------------------------------------
browseURL(URLencode("http://ridiculas.wordpress.com/2011/07/11/como-adicionar-legendas-em-graficos-da-lattice/"))
browseURL(URLencode("http://ridiculas.wordpress.com/2011/07/14/predicao-em-modelos-de-regressao-com-blocos/"))
browseURL(URLencode("http://ridiculas.wordpress.com/2011/07/18/como-fazer-regressao-polinomial-com-diferentes-graus/"))
browseURL(URLencode("http://ridiculas.wordpress.com/2011/08/19/uma-alternativa-para-apresentar-bandas-de-confianca-em-regressa
#-------------------------------------------------------------------------------------------
.
```

14.9 Comparação de curvas

```
.
#-------------------------------------------------------------------------------------------
# carregando dados de número de folhas no colmo principal (nfcp) e função ?graus dias (gde)
# separado por cultivar de ?arroz. Verificar se a taxa de emissão é a mesma.
temf <- read.table("../dados/temf.txt", header=TRUE, sep="\t")
str(temf)
#-------------------------------------------------------------------------------------------
# gráficos
require(lattice)
```

```
xyplot(nfcp~gde, groups=cult, data=temf)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# ajuste do modelo, usar um termo que dê curvatura (^2, sqrt, log)
m0 <- lm(nfcp~cult*(gde+I(gde^2)), data=temf)
par(mfrow=c(2,2)); plot(m0); layout()
anova(m0) # dispensar termo quadrático
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# ajustando modelo removendo termo não necessário, usando a update()
m1 <- update(m0, .~.-cult:I(gde^2))
anova(m1)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# ajustar remoendo intercepto para obter os coeficientes livres da parametrização
m2 <- lm(nfcp~-1+cult/gde+I(gde^2), data=temf)
summary(m2)
confint(m2)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# fazer gráficos com preditos, observados e bandas
range(temf$gde)
pred <- expand.grid(cult=levels(temf$cult), gde=seq(300,930,l=50))
aux <- predict(m1, newdata=pred, interval="confidence")
pred <- cbind(pred, as.data.frame(aux))
str(pred)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# gráfico
require(latticeExtra)
xyplot(nfcp~gde|cult, data=temf, layout=c(3,1), pch=19, col=1)+
  as.layer(xyplot(fit+lwr+upr~gde|cult, type="l", col=c(1,3,3), , lty=c(1,2,2), data=pred))
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# um gráfico (mais bagunçado)
xyplot(nfcp~gde, groups=cult, data=temf, pch=19, col=c(1,2,3))+
  as.layer(xyplot(fit+lwr+upr~gde, type="l", col=1, lty=c(1,3,3), data=subset(pred, cult=="C")))+
  as.layer(xyplot(fit+lwr+upr~gde, type="l", col=2, lty=c(1,3,3), data=subset(pred, cult=="D")))+
  as.layer(xyplot(fit+lwr+upr~gde, type="l", col=3, lty=c(1,3,3), data=subset(pred, cult=="P")))
                                                                                                    #
#-------------------------------------------------------------------------------------------------
.
```

15 Fatorial duplo com fatores quantitativos

15.1 Obtenção do modelo empírico

```
.
#-------------------------------------------------------------------------------------------------
# vamos usar os dados de rendimento de grãos de soja em função de K e A
#soja <- read.table("http://www.leg.ufpr.br/~walmes/cursoR/cnpaf/soja.txt", header=TRUE)
soja <- read.table("../dados/soja.txt", header=TRUE, dec=",")
soja <- transform(soja, bloco=factor(bloco))
names(soja)[1:2] <- c("K","A")
str(soja)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# ajustar um modelo polinômio saturado (usa todos os graus de polinômio permitidos)
m0 <- lm(ts~bloco+poly(A, 2, raw=TRUE)*poly(K, 4, raw=TRUE), data=soja) # modelo saturado
par(mfrow=c(2,2)); plot(m0); layout(1)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# começar a busca por um modelo quadrático completo (inclui termo de 2 grau e interação)
soja <- transform(soja, A2=A^2, K2=K^2)
m1 <- lm(ts~bloco+(A+K)^2+A2+K2, data=soja) # (A+B)^2=A+B+A:B, (A+B+C)^2=A+B+C+A:B+A:C+B:C
par(mfrow=c(2,2)); plot(m1); layout(1)
summary(m1)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# testar a falta de ajuste
anova(m1, m0) # o modelo menor representa
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# ajustando o modelo de segundo grau (dica, usar contr.sum para blocos)
levels(soja$bloco)
```

```
contrasts(soja$bloco)
contrasts(soja$bloco) <- contr.sum(5) # outra forma de trocar o tipo de contraste
m1 <- lm(ts~bloco+(A+K)^2+A2+K2, data=soja)
anova(m1)
                                                                                                    #
#-------------------------------------------------------------------------------------------
.
```

15.2 Predição e gráfico

```
.
#-------------------------------------------------------------------------------------------
# fazer a predição da resposta
A <- seq(35,65,l=20); K <- seq(0,200,l=20)
p0 <- expand.grid(bloco="I", A=A, K=K)
p0 <- transform(p0, A2=A^2, K2=K^2)
p0$ts <- predict(m1, newdata=p0)
                                                                                                    #
#-------------------------------------------------------------------------------------------
# usar a wireframe() da lattice (ver persp(), contour(), contourplot())
require(lattice)
wireframe(ts~A*K, data=p0, scales=list(arrows=FALSE))
levelplot(ts~A*K, data=p0, scales=list(arrows=FALSE), col.regions=heat.colors)
                                                                                                    #
#-------------------------------------------------------------------------------------------
# outros gráficos
filled.contour(A, K, matrix(p0$ts,length(A),length(K)))
contour(A, K, matrix(p0$ts,20,20))
                                                                                                    #
#-------------------------------------------------------------------------------------------
# aprimoramento do gráfico, função traça as curvas de nível no gráfico
panel.3d.contour <- function(x, y, z, rot.mat, distance, type="on",
                             nlevels=20, zlim.scaled, col.contour=1, ...){
  clines <- contourLines(x, y, matrix(z, nrow=length(x), byrow=TRUE), nlevels=nlevels)
  if(any(type%in%c("bottom"))){
    for(ll in clines){
      n <- ltransform3dto3d(rbind(ll$x, ll$y, zlim.scaled[1]), rot.mat, distance)
      panel.lines(n[1,], n[2,], col=col.contour, lty=1, lwd=1)
    }}
  panel.3dwire(x, y, z, rot.mat, distance, zlim.scaled=zlim.scaled, ...)
  if(any(type%in%c("on"))){
    for(ll in clines){
      n <- ltransform3dto3d(rbind(ll$x, ll$y, ll$level), rot.mat, distance)
      panel.lines(n[1,], n[2,], col=col.contour, lty=1, lwd=1)
    }}
}
                                                                                                    #
#-------------------------------------------------------------------------------------------
# define a escala de cores para a superfície
colr <- colorRampPalette(c("yellow", "orange", "red"), space="rgb")
                                                                                                    #
#-------------------------------------------------------------------------------------------
# faz o gráfico aprimorado
wireframe(ts~A*K, data=p0, scales=list(arrows=FALSE),
          zlim=c(100, 250), col="gray30", col.contour="gray30",
          panel.3d.wireframe="panel.3d.contour", type=c("bottom","on"),
          col.regions=colr(100),  drape=TRUE)
                                                                                                    #
#-------------------------------------------------------------------------------------------
# escolher o ângulo de observação e o esquema de cores
require(rpanel)
wire.panel <- function(panel){
  print(wireframe(ts~A*K, data=panel$data, scales=list(arrows=FALSE),
            zlim=c(100, 250), col="gray30", col.contour="gray30",
            panel.3d.wireframe="panel.3d.contour", , type=c("bottom","on"),
            col.regions=panel$col.regions,  drape=TRUE,
            screen=list(z=panel$z.angle, x=panel$x.angle)))
  panel
}
                                                                                                    #
#-------------------------------------------------------------------------------------------
# usando
panel <- rp.control(data=p0, col.regions=colr(100))
rp.slider(panel, z.angle, 0, 360, initval=40, showvalue=TRUE, action=wire.panel)
rp.slider(panel, x.angle, -180, 0, initval=-60, showvalue=TRUE, action=wire.panel)
                                                                                                    #
```

```
#-------------------------------------------------------------------------------------
# encontrando o ponto de máximo algebricamente
coef(m1)
b <- coef(m1)[c("A","K")]
B <- matrix(coef(m1)[c(8,10,10,9)]/c(1,2,2,1), 2, 2)
-0.5*solve(B)%*%b
                                                                                     #
#-------------------------------------------------------------------------------------
# encontrando o ponto de máximo numericamente
coef(m1)[-c(2:5)]
quad.surf <- function(AK, betas){
  A <- AK[1]; K <- AK[2]
  sum(betas*c(1,A,K,A^2,K^2,A*K))
}
op <- optim(c(60, 150), quad.surf, betas=coef(m1)[-c(2:5)], control=list(fnscale=-1))
levelplot(ts~A*K, data=p0, scales=list(arrows=FALSE), col.regions=heat.colors)
trellis.focus("panel", 1, 1)
panel.abline(v=op$par[1], h=op$par[2], col=2)
panel.points(op$par[1], op$par[2], cex=2, col=2)
trellis.unfocus()
                                                                                     #
#-------------------------------------------------------------------------------------
# gráfico aprimorado
wireframe(ts~A*K, data=p0, scales=list(arrows=FALSE),
          zlim=c(100, 250), col="gray30", col.contour="gray30",
          panel.3d.wireframe="panel.3d.contour", type=c("bottom","on"),
          col.regions=colr(100),  drape=TRUE,
          screen=list(z=58, x=-80),
          panel=function(...){
            L <- list(...)
            L$x <- c(op$par[1],op$par[1]); L$y <- c(0,200); L$z <- c(100,100)
            L$type <- "l"; L$col <- "red"; L$lty <- 2; L$par.box <- list(lty=0)
            do.call(panel.cloud, L)
            L$y <- c(op$par[2],op$par[2]); L$x <- c(35,65); L$z <- c(100,100)
            do.call(panel.cloud, L)
            panel.abline(v=50)
            panel.wireframe(...)
          })
                                                                                     #
#-------------------------------------------------------------------------------------
.
```

16 Análise de experimentos em parcela subdividida

16.1 Ajuste do modelo e anova

```
.
#-------------------------------------------------------------------------------------
# parcela são níveis de adubação, na subparcela são níveis de espaçamento
ps <- expand.grid(BL=c("I","II","III","IV"), ES=c("e1","e2"), AD=c("a1","a2","a3"))
ps$alt <- c(58,77,38,52,44,59,30,34,85,90,73,77,59,68,45,55,66,93,67,64,54,75,53,48)
str(ps)
                                                                                     #
#-------------------------------------------------------------------------------------
# gráficos
require(lattice)
xyplot(alt~ES, groups=AD, data=ps, type=c("p","a"), jitter.x=TRUE)
xyplot(alt~AD, groups=ES, data=ps, type=c("p","a"), jitter.x=TRUE)
                                                                                     #
#-------------------------------------------------------------------------------------
# análise de variância (erro A = BL x AD)
m0 <- aov(alt~BL+AD+Error(BL:AD)+ES+AD:ES, data=ps)  # termo Error para declarar erro A
m0 <- aov(alt~BL+AD*ES+Error(BL:AD), data=ps)        # o mesmo com fórmula comprimida
par(mfrow=c(2,2)); plot(m0); layout(1)               # ops! método não aplica, qual classe?
class(m0)
                                                                                     #
#-------------------------------------------------------------------------------------
# pára análise dos resíduos temos que rodar um lm() sem Error
m1 <- lm(alt~BL+AD*(BL+ES), data=ps) # mesmo modelo mas Erro A agora e efeito fixo
par(mfrow=c(2,2)); plot(m1); layout(1)
                                                                                     #
#-------------------------------------------------------------------------------------
# ok, a anova
summary(m0)
                                                                                     #
```

```
#-------------------------------------------------------------------------------------------
# desdobrando a SQ
m2 <- aov(alt~BL+AD/ES+Error(BL:AD), data=ps) # ajusta com aninhamento
summary(m2)
coef(m2)
summary(m2, split=list("AD:ES"=list("AD1"=1, "AD2"=2, "AD3"=3)))
                                                                                          #
#-------------------------------------------------------------------------------------------
# usando o pacote GAD
require(GAD)
ps$BL <- as.random(ps$BL)
ps$AD <- as.fixed(ps$AD)
ps$ES <- as.fixed(ps$ES)
m3 <- lm(alt~BL+BL%in%AD+AD*ES, data=ps) # modelo lm
gad(m3)       # quadro de anova para parcela subdividida
estimates(m3) # mostra quais são os QM envolvidos no F
m3 <- lm(terms(alt~BL+AD+BL%in%AD+ES+AD:ES, keep.order=TRUE), data=ps) # impõe a ordem
gad(m3)       # quadro de anova para parcela subdividida na ordem usual dos termos
                                                                                          #
#-------------------------------------------------------------------------------------------
.
```

16.2 Teste de médias para os níveis da subparcela

```
.
#-------------------------------------------------------------------------------------------
# dedobrar ES em cada nível de AD
require(agricolae)
df.residual(m0) # não extrai, mesmo porque agora existem 2 resíduos
deviance(m0)    # não extrai, existem 2 resíduos
                                                                                          #
#-------------------------------------------------------------------------------------------
# temos que retirar os valores de GL e QM da anova
str(summary(m0)) # uma lista com 2 data.frames
str(summary(m0)[[1]])
str(summary(m0)[[1]][[1]])
summary(m0)[[1]][[1]] # anova da parte da parcela
summary(m0)[[2]][[1]] # anova da parte da subparcela
glp <- summary(m0)[[1]][[1]]["Residuals","Df"]
qmp <- summary(m0)[[1]][[1]]["Residuals","Mean Sq"]
gls <- summary(m0)[[2]][[1]]["Residuals","Df"]
qms <- summary(m0)[[2]][[1]]["Residuals","Mean Sq"]
c(glp, qmp, gls, qms)
                                                                                          #
#-------------------------------------------------------------------------------------------
# teste de Tukey, usando plyr::ddply()
require(agricolae)
require(plyr)
ddply(ps, .(AD),
      .fun=function(a){
        with(a, HSD.test(alt, ES, DFerror=gls, MSerror=qms))
      })
                                                                                          #
#-------------------------------------------------------------------------------------------
# teste de ScottKnott
require(ScottKnott)
sapply(1:nlevels(ps$AD), simplify=FALSE,
       function(a){
         sk <- SK.nest(x=ps, y=ps$alt, model="y~BL+ES*AD+Error(BL:AD)",
                       which="ES:AD", fl2=a, error="Within")
         summary(sk)
       })
                                                                                          #
#-------------------------------------------------------------------------------------------
.
```

16.3 Teste de médias para os níveis da parcela

$$\text{QMr} = \frac{\text{QMa} + (n_b - 1) \cdot \text{QMb}}{n_b} \tag{31}$$

$$\text{GLr} = \frac{(\text{QMa} + (n_b - 1) \cdot \text{QMb})^2}{\text{QMa}^2/\text{GLa} + ((n_b - 1) \cdot \text{QMb})^2/\text{GLb}} \tag{32}$$

```
#-------------------------------------------------------------------------------------------
# desdobrar AD dentro de ES (requer variância complexa, expressão de Satterthwaite)
# função criada para calcular o QM e GL aproximados baseados na função linear de QMs
satter <- function(A, B, C=c(0,1,1)){
  ## cada termo é um vetor cujos elementos são QM, GL e número de níveis de cada estrato/fator
  ## o vetor em C só precisa ser fonecido em casos de parcela subsubdivida
  qmr <- (A[1]+(B[3]-1)*B[1]+B[3]*(C[3]-1)*C[1])/(B[3]*C[3])
  glr <- (A[1]+(B[3]-1)*B[1]+B[3]*(C[3]-1)*C[1])^2/
    ((A[1]^2/A[2])+((B[3]-1)*B[1])^2/B[2]+(B[3]*(C[3]-1)*C[1])^2/C[2])
  ## retorna um vetor com o QMr e o GLr obtidos pela aproximação
  return(c(qmr=qmr, glr=glr))
}
                                                                                          #
#-------------------------------------------------------------------------------------------
# obtendo o QM e GL (QM do resíduo, GL do resíduo e número de níveis do fator do estrato)
aux <- satter(A=c(qmp, glp, 3), B=c(qms, gls, 2)); aux
ddply(ps, .(ES),
      .fun=function(e){
        with(e, HSD.test(alt, AD, DFerror=aux["glr"], MSerror=aux["qmr"]))
      })
                                                                                          #
#-------------------------------------------------------------------------------------------
# desdobrar com o teste de ScottKnott
sapply(1:nlevels(ps$ES), simplify=FALSE,
       function(e){
         sk <- SK.nest(x=ps, y=ps$alt, model="y~BL+AD*ES+Error(BL:AD)",
                       which="AD:ES", fl2=e, error="BL:AD")
         summary(sk)
       })
                                                                                          #
#-------------------------------------------------------------------------------------------
.
```

16.4 Análise usando a função `ExpDes::split2.rbd()`

```
.
#-------------------------------------------------------------------------------------------
# usando o pacote ExpDes
require(ExpDes)
with(ps, split2.rbd(AD, ES, BL, alt, quali=c(TRUE,TRUE), mcomp="tukey"))
                                                                                          #
#-------------------------------------------------------------------------------------------
.
```

16.5 Ajuste pelo método REML

```
.
#-------------------------------------------------------------------------------------------
# a interação BL:AD é de efeito aleatório e podemos estimar sua variância por REML
# a desvantagem do método anova é que para o caso desbalanceado surgem problemas na
# decomposição das SQ, veja
ps2 <- ps
ps2[1,"alt"] <- NA                              # substitui o primeiro valor por um NA
m3 <- aov(alt~BL+AD*ES+Error(BL:AD), data=ps2)
summary(m3) # NA causa perda de ortogonalidade
coef(m3)
                                                                                          #
#-------------------------------------------------------------------------------------------
# a estimação REML (ML) não depende de ortogonalidade
require(nlme)
mm0 <- lme(alt~AD*ES, random=~1|BL/AD, data=ps2, na.action=na.omit)
anova(mm0)
VarCorr(mm0)
print(mm0, corr=FALSE)
summary(mm0)$tTable
                                                                                          #
```

```
#-------------------------------------------------------------------------------------
# se quer o efeito de bloco seja fixo, deve-se criar um novo fator BL:AD
ps2$parcela <- interaction(ps2$BL, ps2$AD)
mm1 <- lme(alt~BL+AD*ES, random=~1|parcela, data=ps2, na.action=na.omit)
anova(mm1)
VarCorr(mm1)
print(mm1, corr=FALSE)
summary(mm1)$tTable # o desbalanceamento não comprometeu a estimabilidade
                                                                                    #
#-------------------------------------------------------------------------------------
# fazendo o desdobramento da interação (modelo sem o NA)
ps$parcela <- interaction(ps$BL, ps$AD)
mm1 <- lme(alt~BL+AD*ES, random=~1|parcela, data=ps, na.action=na.omit)
anova(mm1)
require(contrast)
contrast(mm1, type="average",
         list(BL=levels(ps$BL), AD="a1", ES="e1"),
         list(BL=levels(ps$BL), AD="a1", ES="e2"))
                                                                                    #
#-------------------------------------------------------------------------------------
# níveis de espaçamento dentro de níveis de adubação
contrast(mm1,
         list(BL=levels(ps$BL)[1], AD=levels(ps$AD), ES="e1"),
         list(BL=levels(ps$BL)[1], AD=levels(ps$AD), ES="e2"))
                                                                                    #
#-------------------------------------------------------------------------------------
# níveis de adubação dentro de níveis de espaçamento (não envolve variância complexa)
contrast(mm1,
         list(BL=levels(ps$BL)[1], AD=levels(ps$AD)[1], ES=levels(ps$ES)),
         list(BL=levels(ps$BL)[1], AD=levels(ps$AD)[2], ES=levels(ps$ES)))
                                                                                    #
#-------------------------------------------------------------------------------------
.
```

16.6 Função para casualização em parcelas subdivididas

```
.
#-------------------------------------------------------------------------------------
# 3 níveis da parcela, 4 níveis na subparcela
corr <- c("gesso","calcario","gesso&calcario")
cult <- paste("cultivar", 1:4)
r <- 5
parsub <- design.split(corr, cult, r=r, design="rcbd", kinds="Super-Duper")
parsub
write.table(parsub, "parsub.xls", row.names=FALSE, sep="\t")
                                                                                    #
#-------------------------------------------------------------------------------------
```

17 Análise de experimentos em parcelas sub-subdivididas

17.1 Ajuste do modelo e anova

```
.
#-------------------------------------------------------------------------------------
# experimento em parcela sub-subdividida com fontes alternativas de fertilizante orgânico
# em diferentes doses e adubação complementar de npk na cultura da soja.
# fonte: cama de frango (CF), esterco de curral (EC) e póde carvão (PC) [parcela]
# dorg: dose do fertilizante orgânico (ton/ha)                          [subparcela]
# dnpk: dose do fertilizante mineral (kg/ha)                            [subsubparcela]
# AF: área foliar
# prod: produção de grãos
                                                                                    #
#-------------------------------------------------------------------------------------
#pss <- read.table("http://www.leg.ufpr.br/~walmes/cursoR/fao.txt", header=TRUE)
#fao <- read.table("../dados/fao.txt", header=TRUE)
fao <- read.table("/media/SAMSUNG/Consultorias/Alexandre Abdao Passos/fao.txt", header=TRUE)
fao <- transform(fao, dorg=factor(dorg), dnpk=factor(dnpk), bloco=factor(bloco))
str(fao)
                                                                                    #
#-------------------------------------------------------------------------------------
# gráficos
```

```
require(lattice)
xyplot(log(AF)~dorg|dnpk, groups=fonte, data=fao, type=c("p","a"))
xyplot(prod~dorg|dnpk, groups=fonte, data=fao, type=c("p","a"))
xyplot(PCS~dorg|dnpk, groups=fonte, data=fao, type=c("p","a"))
                                                                                #
#-------------------------------------------------------------------------------
# ajusta primeiro modelo com a lm para poder fazer a checagem dos resíduos
m0 <- aov(AF~bloco/fonte/dorg+fonte*dorg*dnpk, data=fao)
par(mfrow=c(2,2)); plot(m0); layout(1)
MASS::boxcox(m0) # indica log()
m0 <- aov(log(AF)~bloco/fonte/dorg+fonte*dorg*dnpk, data=fao)
par(mfrow=c(2,2)); plot(m0); layout(1) # ok
                                                                                #
#-------------------------------------------------------------------------------
# análise de variância, ErroA=bloco:fonte, ErroB=bloco:fonte:dorg
m0 <- aov(log(AF)~bloco+fonte*dorg*dnpk+Error(bloco:fonte/dorg), data=fao)
summary(m0) # interação
                                                                                #
#-------------------------------------------------------------------------------
.
```

17.2 Testes de médias para os níveis da subsubparcela

```
.
#-------------------------------------------------------------------------------
# desdobrar níveis da subsub dentro de níveis da sub com parcela (usa erro C)
class(m0)
str(summary(m0))  # df.residual() e deviance() não funcionam para aovlist
gl3 <- summary(m0)[["Error: Within"]][[1]]["Residuals","Df"]
qm3 <- summary(m0)[["Error: Within"]][[1]]["Residuals","Mean Sq"]
                                                                                #
#-------------------------------------------------------------------------------
# usar a agricole::HSD.test() junto com a plyr::ddply() para desdobrar a interação
require(plyr)
require(agricolae)
desCinAB <- ddply(fao, .(fonte, dorg),
                  .fun=function(AB){
                    with(AB, HSD.test(log(AF), dnpk, gl3, qm3))
                  })
desCinAB
split(desCinAB, f=interaction(desCinAB$fonte, desCinAB$dorg)) # dividir para conquistar
                                                                                #
#-------------------------------------------------------------------------------
# gráfico
desCinAB$meansl <- with(desCinAB, paste(format(means, dig=3), gsub(" ","",M)))
barchart(means~trt|fonte*dorg, data=desCinAB, horiz=FALSE, ylim=c(2.8,5.5),
          col=colors()[400], strip=strip.custom(bg="gray90"),
          panel=function(x, y, subscripts, ...){
            panel.barchart(x, y, subscripts=subscripts, ...)
            panel.text(x, y, label=desCinAB$meansl[subscripts], pos=3)
          })
                                                                                #
#-------------------------------------------------------------------------------
# gráfico aprimorado
levels(desCinAB$fonte) <- c("Cama de frango","Esterco de curral","Pó de carvão")
levels(desCinAB$dorg) <- paste(levels(desCinAB$dorg), "ton/ha")
bct <-
  barchart(means~trt|fonte*dorg, data=desCinAB, horiz=FALSE, ylim=c(2.8,5.5),
           col=colors()[399],
           strip=strip.custom(bg="gray90"),
           panel=function(x, y, subscripts, ...){
             panel.barchart(x, y, subscripts=subscripts, ...)
             panel.text(x, y, label=desCinAB$meansl[subscripts], pos=3)
           })
require(latticeExtra)
useOuterStrips(bct, strip=strip.custom(bg="gray90"),
               strip.left=strip.custom(bg="gray50"))
                                                                                #
#-------------------------------------------------------------------------------
# fazendo o teste de Scott-Knott
require(ScottKnott)
skCinAB <- sapply(1:nlevels(fao$fonte), simplify=FALSE,
                  function(f){
                    sapply(1:nlevels(fao$dorg), simplify=FALSE,
                           function(o){
                             tes <- SK.nest(x=fao, y=log(fao$AF),
```

```
                                        model="y~bloco+dnpk*fonte*dorg+Error(bloco:fonte/dorg)",
                                        which="dnpk:fonte:dorg", error="Within",
                                        fl2=f, fl3=o)
                            tes <- summary(tes)
                            tes
                           })
                  })
skCinAB <- ldply(unlist(skCinAB, recursive=FALSE))
split(skCinAB, f=rep(1:12, e=5))
                                                                                                    #
#-------------------------------------------------------------------------------------------
.
```

17.3 Testes de médias para os níveis da subparcela

```
#-------------------------------------------------------------------------------------------
# função criada para calcular o QM e GL aproximados baseados na função linear de QMs
satter <- function(A, B, C=c(0,1,1)){
  ## cada termo é um vetor cujos elementos são QM, GL e número de níveis de cada estrato/fator
  ## o vetor em C só precisa ser fonecido em casos de parcela subsubdivida
  qmr <- (A[1]+(B[3]-1)*B[1]+B[3]*(C[3]-1)*C[1])/(B[3]*C[3])
  glr <- (A[1]+(B[3]-1)*B[1]+B[3]*(C[3]-1)*C[1])^2/
    ((A[1]^2/A[2])+((B[3]-1)*B[1])^2/B[2]+(B[3]*(C[3]-1)*C[1])^2/C[2])
  ## retorna um vetor com o QMr e o GLr obtidos pela aproximação
  return(c(qmr=qmr, glr=glr))
}
                                                                                                    #
#-------------------------------------------------------------------------------------------
# desdobrar dorg em fonte com dnpk
str(summary(m0))
gl2 <- summary(m0)[["Error: bloco:fonte:dorg"]][[1]]["Residuals","Df"]
qm2 <- summary(m0)[["Error: bloco:fonte:dorg"]][[1]]["Residuals","Mean Sq"]
vcBinAC <- satter(c(qm2, gl2, nlevels(fao$dorg)), c(qm3, gl3, nlevels(fao$dnpk)))
vcBinAC
c(qm2,qm3)
c(gl2, gl3)
                                                                                                    #
#-------------------------------------------------------------------------------------------
# teste de médias
desBinAC <- ddply(fao, .(fonte, dnpk),
                  .fun=function(AC){
                    with(AC, HSD.test(log(AF), dorg, vcBinAC["glr"], vcBinAC["qmr"]))
                  })
desBinAC
split(desBinAC, f=interaction(desBinAC$fonte, desBinAC$dnpk)) # dividir para conquistar
                                                                                                    #
#-------------------------------------------------------------------------------------------
# desdobrar com a ScottKnott
skBinAC <- sapply(1:nlevels(fao$fonte), simplify=FALSE,
                  function(f){
                    sapply(1:nlevels(fao$dnpk), simplify=FALSE,
                           function(npk){
                             tes <- SK.nest(x=fao, y=log(fao$AF),
                                            model="y~bloco+dorg*fonte*dnpk+Error(bloco:fonte/dorg)",
                                            which="dorg:fonte:dnpk", error="bloco:fonte:dorg",
                                            fl2=f, fl3=npk)
                             tes <- summary(tes)
                             tes
                           })
                  })
skBinAC <- ldply(unlist(skBinAC, recursive=FALSE))
split(skBinAC, f=rep(1:15, e=4))
                                                                                                    #
#-------------------------------------------------------------------------------------------
.
```

17.4 Testes de médias para os níveis da parcela

```
#-------------------------------------------------------------------------------------------
# desdobrar fonte em dorg com dnpk
gl1 <- summary(m0)[["Error: bloco:fonte"]][[1]]["Residuals","Df"]
```

```
qm1 <- summary(m0)[["Error: bloco:fonte"]][[1]]["Residuals","Mean Sq"]
vcAinBC <- satter(c(qm1, gl1, nlevels(fao$fonte)),
                  c(qm2, gl2, nlevels(fao$dorg)),
                  c(qm3, gl3, nlevels(fao$dnpk)))
vcAinBC
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# teste de médias
desAinBC <- ddply(fao, .(dorg, dnpk),
                  .fun=function(BC){
                    invisible(capture.output( # silencia as saidas da HSD.test()
                    aux <- with(BC, HSD.test(log(AF), fonte, vcAinBC["glr"], vcAinBC["qmr"]))
                                                ))
                    aux
                  })
desAinBC
split(desAinBC, f=interaction(desAinBC$dorg, desAinBC$dnpk)) # dividir para conquistar
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# usando o teste de ScottKnott
skAinBC <- sapply(1:nlevels(fao$dorg), simplify=FALSE,
                  function(o){
                    sapply(1:nlevels(fao$dnpk), simplify=FALSE,
                           function(npk){
                             tes <- SK.nest(x=fao, y=log(fao$AF),
                                            model="y~bloco+fonte*dorg*dnpk+Error(bloco:fonte/dorg)",
                                            which="fonte:dorg:dnpk", error="bloco:fonte",
                                            fl2=o, fl3=npk)
                             tes <- summary(tes)
                             tes
                           })
                  })
skAinBC <- ldply(unlist(skAinBC, recursive=FALSE))
split(skAinBC, f=rep(1:20, e=3))
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# em geral, eu rezo para não ter interação tripla
                                                                                                    #
#-------------------------------------------------------------------------------------------------
.
```

17.5 Análise com fatores contínuos

```
.
#-------------------------------------------------------------------------------------------------
# análise da produção
xyplot(prod~dorg|dnpk, groups=fonte, data=fao, type=c("p","a")) # padrão claro nos dados
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# ajusta primeiro modelo com a lm para poder fazer a checagem dos resíduos
m0 <- aov(prod~bloco/fonte/dorg+fonte*dorg*dnpk, data=fao)
par(mfrow=c(2,2)); plot(m0); layout(1)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# análise de variância, ErroA=bloco:fonte, ErroB=bloco:fonte:dorg
m0 <- aov(prod~bloco+fonte*dorg*dnpk+Error(bloco:fonte/dorg), data=fao)
summary(m0)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# análise de variância para efeitos até o segundo grau do polinômio, criar nomes curtos
# e converter para métricas dorg e dnpk, dnpk foi divido por 1000
fao <- transform(fao, bl=bloco, ft=fonte,
                 o1=as.numeric(as.character(dorg)), m1=as.numeric(as.character(dnpk))/1000)
fao <- transform(fao, o2=o1^2, m2=m1^2)
str(fao)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# ajuste com termos lineares e quadráticos (modelo maior)
m1 <- aov(prod~bl+ft*(o1+o2)*(m1+m2)+Error(bloco:fonte/dorg), data=fao)
summary(m1)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# retirar os termos quadráticos e demais não importantes
m2 <- aov(prod~bl+ft*o1+m1+Error(bloco:fonte/dorg), data=fao)
summary(m2)
                                                                                                    #
```

```
#-------------------------------------------------------------------------------------
# fazendo o gráfico final, predição para o bloco 1, aovlist não tem método predict()
# ajustar modelo lm() e usar a predict()
m3 <- lm(prod~-1+ft/o1+m1, data=fao)
summary(m3)
par(mfrow=c(2,2)); plot(m3); layout(1)
pred <- with(fao, expand.grid(ft=levels(ft),
                              o1=seq(0,9,l=12),
                              m1=seq(0,0.4,l=4)))
str(pred)
pred$y <- predict(m3, newdata=pred)
require(latticeExtra)
xyplot(prod~dorg|fonte, groups=dnpk, data=fao)+
  as.layer(xyplot(y~o1|ft, groups=m1, data=pred, type="l"))
                                                                                     #
#-------------------------------------------------------------------------------------
# gráfico 3d
pred2 <- with(fao, expand.grid(ft=levels(ft),
                               o1=seq(0,9,l=12),
                               m1=seq(0,0.4,l=12)))
str(pred2)
pred2$y <- predict(m3, newdata=pred2)
wireframe(y~m1+o1|ft, data=pred2, drape=TRUE, layout=c(3,1))
wireframe(y~m1+o1, groups=ft, data=pred2)
                                                                                     #
#-------------------------------------------------------------------------------------
# gráfico aprimorado
cols <- c("lightblue", "green4", 1)
wireframe(y~m1+o1, groups=ft, data=pred2,
          col.groups=cols,
          scales=list(arrows=FALSE, col="black"),
          xlab=list("Adubação mineral (ton/ha)", rot=30),
          ylab=list("Adubação orgânica (ton/ha)",rot=-40),
          zlab=list("Produtividade (kg/ha)", rot=90),
          auto.key=list(space="top", columns=3,
            text=c("Cama de frango","Esterco de curral","Pó de carvão"),
            columns=1, points=FALSE, font=6,
            rectangles=TRUE, cex=1, border=FALSE),
          par.settings=simpleTheme(col=cols, alpha=0.5))
                                                                                     #
#-------------------------------------------------------------------------------------
.
```

17.6 Ajuste pelo método REML (ML)

```
.
#-------------------------------------------------------------------------------------
# declarar bloco e suas interações como aleatórias (é mais fácil de trabalhar)
require(nlme)
mm0 <- lme(prod~fonte*dorg*dnpk, random=~1|bloco/fonte/dorg, data=fao, method="REML")
anova(mm0)
VarCorr(mm0)
print(mm0, corr=FALSE)
summary(mm0)$tTable
                                                                                     #
#-------------------------------------------------------------------------------------
# ajustar o modelo com efeitos quadráticos, usar ML para fazer LRT
mm1 <- lme(prod~ft*(o1+o2)*(m1+m2), random=~1|bl/ft/dorg, data=fao, method="ML")
mm2 <- lme(prod~ft*o1+m1, random=~1|bl/ft/dorg, data=fao, method="ML")
anova(mm1, mm2)
anova(mm2)
summary(mm2)$tTable
                                                                                     #
#-------------------------------------------------------------------------------------
# fazer a predição
pred3 <- with(fao, expand.grid(bl=levels(bl)[1],
                               ft=levels(ft),
                               o1=seq(0,9,l=12),
                               m1=seq(0,0.4,l=12)))
pred3$y <- predict(mm2, newdata=pred3, level=0)
                                                                                     #
#-------------------------------------------------------------------------------------
# gráfico
cols <- c("lightblue", "green4", 1)
wireframe(y~m1+o1, groups=ft, data=pred3,
          col.groups=cols,
```

```
          scales=list(arrows=FALSE, col="black"),
          xlab=list("Adubação mineral (ton/ha)", rot=30),
          ylab=list("Adubação orgânica (ton/ha)",rot=-40),
          zlab=list("Produtividade (kg/ha)", rot=90),
          auto.key=list(space="top", columns=3,
            text=c("Cama de frango","Esterco de curral","Pó de carvão"),
            columns=1, points=FALSE, font=6,
            rectangles=TRUE, cex=1, border=FALSE),
          par.settings=simpleTheme(col=cols, alpha=0.5))
                                                                                          #
#-------------------------------------------------------------------------------------
.
```

18 Análise de experimento em faixas

18.1 Ajuste do modelo e anova

```
.
#-------------------------------------------------------------------------------------
# experimento em faixa com 4 blocos e 3 fatores (sistema, gesso, adubação). A casualização
# dos tratamentos foi em faixa, sendo os 2*2=4 níveis de sistema:gesso no sentido vertical
# e os 5 níveis de adubação no sentido horizontal, na cultura do milho.
# sis: plantio convencional (2); plantio direto (1);
# ges: sem gesso (1); com 2 ton/ha de gesso (2);
# dos: dose de npk (30, 60, 90, 120 e 150 kg/ha);
# prod: produtividade;
# cem: peso de 100 sementes;
# alt: altura de plantas;
                                                                                          #
#-------------------------------------------------------------------------------------
# lendo os dados
fai <- read.table("../dados/faixas.txt", header=TRUE, sep="\t")
str(fai)
names(fai) <- tolower(names(fai))
#-------------------------------------------------------------------------------------
# gráficos
require(lattice)
xyplot(prod~dos|sis, groups=ges, data=fai, type=c("p","a"))
xyplot(cem~dos|sis, groups=ges, data=fai, type=c("p","a"))
xyplot(alt~dos|sis, groups=ges, data=fai, type=c("p","a"))
                                                                                          #
#-------------------------------------------------------------------------------------
# converter para fatores, cria o fator que combina sis e ges (é a faixa vertical)
fai <- transform(fai, rep=factor(rep), sis=factor(sis), ges=factor(ges), dos=ordered(dos))
fai <- transform(fai, sisges=interaction(sis, ges))
levels(fai$sis) <- c("PD","PC")
levels(fai$ges) <- c("0t","2t")
str(fai)
                                                                                          #
#-------------------------------------------------------------------------------------
# análise da produtividade, ajuste do modelo com um erro para checar os resíduos
m0 <- lm(prod~rep/(sisges+dos)+sis*ges*dos, data=fai)
par(mfrow=c(2,2)); plot(m0); layout(1)
                                                                                          #
#-------------------------------------------------------------------------------------
# ajuste do modelo de experimentos em faixas
m1 <- aov(prod~rep+sis*ges*dos+Error(rep:(sisges+dos)), data=fai)
summary(m1)
                                                                                          #
#-------------------------------------------------------------------------------------
# desdobramento da soma de quadrados
m2 <- aov(prod~rep+(sis/ges)*dos+Error(rep:(sisges+dos)), data=fai)
coef(m2)
summary(m2, expand.split=FALSE,
        split=list("sis:ges"=list(PD=1,PC=2), dos=list(L=1,Q=2,C=3,QQ=4)))
                                                                                          #
#-------------------------------------------------------------------------------------
# obtenção das médias
with(fai, tapply(prod, list(sis, ges), mean)) # isso porque é balanceado
                                                                                          #
#-------------------------------------------------------------------------------------
# tabelas de efeitos e de médias (amostrais)
model.tables(m1)               # tabela dos efeitos
model.tables(m1, type="means") # tabela das médias (são amostrais, só caso balanceado)
                                                                                          #
```

```
#-------------------------------------------------------------------------------------
# em caso de interações, usar a aproximação de Satterthwaite conforme visto na sessão de
# parcelas subdivididas
                                                                                      #
#-------------------------------------------------------------------------------------
.
```

18.2 Estimação pelo método REML

```
.
#-------------------------------------------------------------------------------------
# declarar no argumento random os termos do modelo que são de efeito aleatório
# não é possível fazer com a nlme::lme() pois esta só admite efeitos aleatórios aninhados
# e não cruzados como é o caso do experimento em faixas. Vamos usar a lme4::lmer()
require(lme4)
mm0 <- lmer(prod~sis*ges*dos+(1|rep)+(1|rep:sisges)+(1|rep:dos), data=fai)
anova(mm0)
print(mm0, corr=FALSE)
                                                                                      #
#-------------------------------------------------------------------------------------
.
```

19 Análise de experimento com medidas repetidas

19.1 Ajuste do modelo e teste de hipótese

```
.
#-------------------------------------------------------------------------------------
# experimento em campo avaliando presença de gesso, sistemas de cultivo e efeito da
# profundidade em componentes da fertilidade do solo. Medidas repetidas são feitas no
# perfil do solo o que pode induzir correlação entre as observações.
# sis: sistema de plantio convencional (PC), sistema de plantio direto (PD);
# ges: G_0 (correção gesso ausente), G_1 (correção com gesso 2 ton/ha);
# prof: profundidade de amostragem do solo (cm);
# demais variáveis de análises químicas de rotina feitas em solo;
                                                                                      #
#-------------------------------------------------------------------------------------
# carregando os dados
sgp <- read.table("../dados/gesso.txt", header=TRUE, sep="\t")
str(sgp)
names(sgp) <- gsub("_","",tolower(names(sgp)))
sgp$pro <- ordered(sgp$prof)
                                                                                      #
#-------------------------------------------------------------------------------------
# gráficos
require(lattice)
xyplot(al~prof|sis, groups=ges, data=sgp, type=c("p","a"))
xyplot(al~prof|sis, groups=interaction(blo,ges), data=sgp, type=c("p","a"))
                                                                                      #
#-------------------------------------------------------------------------------------
# vendo todos os gráficos, um após o outro
for(i in names(sgp)[5:19]){
  aux <- sgp[,i]
  print(xyplot(aux~prof|sis, groups=interaction(blo,ges),
         data=sgp, type=c("p","a"), main=i))
  Sys.sleep(1.5)
}
                                                                                      #
#-------------------------------------------------------------------------------------
# usar a lm para checar resíduos
m0 <- lm(phkcl~sis*ges*pro+blo/sis/ges, data=sgp)
par(mfrow=c(2,2)); plot(m0); layout(1)
anova(m0) # essa anova é só para ver se declaramos todos os termos, não fazer inferência
                                                                                      #
#-------------------------------------------------------------------------------------
# fazer o gráfico dos resíduos e_{i} ~ e_{i-1}
id <- with(sgp, interaction(blo, sis, ges))
res <- residuals(m0)
aux <- tapply(res, id,
              function(x){
                n <- length(x)
```

```
              cbind(x[1:(n-1)], x[2:n])
            })
aux <- do.call(rbind, aux)
str(aux)
plot(aux)
cor(aux)
cor.test(aux[,1], aux[,2])
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# adicionando elipses
require(ellipse)
plot(aux)
for(i in c(0.5,0.75,0.9,0.95,0.99)) lines(ellipse(cor(aux), scale=diag(cov(aux)), level=i))
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# usando a car::scatterplotMatrix()
require(car)
scatterplotMatrix(aux, diagonal="qqplot", ellipse=TRUE)
help(scatterplotMatrix, help_type="html")
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# ajuste do modelo de parcelas sub-subdivididas (usar ML para fazer LRT)
require(nlme)
mm0 <- lme(phkcl~sis*ges*pro, random=~1|blo/sis/ges, data=sgp, method="ML")
anova(mm0)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# ajuste do modelo com estrutura de correlação na profundidade
# -- modelo de correlação exponencial
mm1 <- update(mm0, correlation=corExp(0.9, form=~prof|blo/sis/ges)) # Exponencial
print(mm1, cor=FALSE)
phi <- exp(c(mm1$modelStruct$corStruct))
round(exp(-abs(outer(1:5, 1:5, "-"))/phi), 3)
curve(exp(-x/phi),0,20)
logLik(mm1) # -43.49
# -- modelo de correlação Gaussiana
mm2 <- update(mm0, correlation=corGaus(3, form=~prof|blo/sis/ges))
print(mm2, cor=FALSE)
phi <- exp(c(mm2$modelStruct$corStruct))
round(exp(-(abs(outer(1:5, 1:5, "-"))/phi)^2), 3)
curve(exp(-(x/phi)^2),0,20)
logLik(mm2) # -41.17
# -- modelo de correlação linear
mm3 <- update(mm0, correlation=corLin(10, form=~prof|blo/sis/ges))
print(mm3, cor=FALSE)
phi <- exp(c(mm3$modelStruct$corStruct))
round(1-(abs(outer(1:5, 1:5, "-"))/phi), 3)
curve((1-(x/phi))*(x<phi)+0,0,20)
logLik(mm3) # -42.06
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# o modelo mm2 alcançou maior verossimilhança portanto é mais compatível
anova(mm0, mm2) # testa H0: phi=0
anova(mm0)
anova(mm2) # menor sigficância para os termos que envolvem profundidade
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# ajustar modelo contínuo para a prof, modelo aditivo
xyplot(phkcl~prof|sis, groups=ges, data=sgp, type=c("p","a")) # vai pedir 3 grau
mm4 <- lme(phkcl~(prof+I(prof^2)+I(prof^3)), random=~1|blo/sis/ges, data=sgp,
           correlation=corGaus(3, form=~prof|blo/sis/ges), method="ML")
print(mm4, cor=FALSE)
anova(mm4, mm2) # modelo apenas com profundidade ok
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# gráfico
pred <- data.frame(prof=seq(5,25,by=0.1))
pred$y <- predict(mm4, newdata=pred, level=0)
require(latticeExtra)
xyplot(phkcl~(prof-2.5), data=sgp, jitter.x=TRUE)+
  as.layer(xyplot(y~(prof-2.5), data=pred, type="l"))
                                                                                                    #
#-----------------------------------------------------------------------------------------------
.
```

19.2 Transformação na resposta e dependência

```
.
#-------------------------------------------------------------------------------------------
# estudar o S
xyplot(s~prof|sis, groups=ges, data=sgp, type=c("p","a"))
xyplot(s~prof|sis, groups=interaction(blo,ges), data=sgp, type=c("p","a"))
m0 <- lm(s~sis*ges*pro+blo/sis/ges, data=sgp)
par(mfrow=c(2,2)); plot(m0); layout(1)
                                                                                          #
#-------------------------------------------------------------------------------------------
# gráficos dos resíduos defasados
res <- residuals(m0)
aux <- tapply(res, id,
              function(x){
                n <- length(x)
                cbind(x[1:(n-1)], x[2:n])
              })
aux <- do.call(rbind, aux)
str(aux)
plot(aux) # pontos distantes são resíduos extremos devido a heterogeneidade de variância
cor(aux)
cor.test(aux[,1], aux[,2])
                                                                                          #
#-------------------------------------------------------------------------------------------
# vamos aplicar uma transformação boxcox
MASS::boxcox(m0) # atenção: essa tranformação tá supondo observações independentes!
xyplot(log(s)~prof|sis, groups=interaction(blo,ges), data=sgp, type=c("p","a"))
m1 <- lm(log(s)~sis*ges*pro+blo/sis/ges, data=sgp)
par(mfrow=c(2,2)); plot(m1); layout(1)
                                                                                          #
#-------------------------------------------------------------------------------------------
# gráficos dos resíduos defasados da variável trasnformada
res <- residuals(m1)
aux <- tapply(res, id,
              function(x){
                n <- length(x)
                cbind(x[1:(n-1)], x[2:n])
              })
aux <- do.call(rbind, aux)
str(aux)
plot(aux) # sumiu a correlação
cor(aux)
cor.test(aux[,1], aux[,2])
                                                                                          #
#-------------------------------------------------------------------------------------------
# ajustar modelo para a variável original com e sem correlação e fazer um LRT
mm0 <- lme(s~sis*ges*pro, random=~1|blo/sis/ges, data=sgp, method="ML")
print(mm0, corr=FALSE)
mm1 <- update(mm0, correlation=corGaus(3, form=~prof|blo/sis/ges))
print(mm1, corr=FALSE)
anova(mm0, mm1) # com a variável original há dependencia
phi <- exp(c(mm1$modelStruct$corStruct))
round(exp(-(abs(outer(1:5, 1:5, "-"))/phi)^2), 3)
curve(exp(-(x/phi)^2),0,20)
                                                                                          #
#-------------------------------------------------------------------------------------------
# ajustar modelo para a variável tranformada com e sem correlação e fazer um LRT
mm0 <- lme(log(s)~sis*ges*pro, random=~1|blo/sis/ges, data=sgp, method="ML")
print(mm0, corr=FALSE)
mm1 <- update(mm0, correlation=corGaus(3, form=~prof|blo/sis/ges))
print(mm1, corr=FALSE)
anova(mm0, mm1) # com a variável transformada não há dependencia!!!
phi <- exp(c(mm1$modelStruct$corStruct))
round(exp(-(abs(outer(1:5, 1:5, "-"))/phi)^2), 3)
curve(exp(-(x/phi)^2),0,20)
                                                                                          #
#-------------------------------------------------------------------------------------------
# quadro de testes de hipótese
anova(mm0)
xyplot(log(s)~prof|sis, groups=ges, data=sgp, type=c("p","a"))
                                                                                          #
#-------------------------------------------------------------------------------------------
.
```

20 Análise de experimentos em vários locais

20.1 Análise exploratória

```
#-------------------------------------------------------------------------------------------
# experimento com soja da Embrapa Agropecuária Oeste em 4 municípios, delineamento em
# blocos com materiais em ensaio de competição (efeitos fixos). Muitos NA.
# local: município onde foram feitos os experimentos;
# gen: material genético avaliado;
# trat: outra coluna (numérica) que identifica o gen;
# bl: indice dos blocos em cada local;
# df: dias para o ?florescimento (dias);
# dm: dias para ?maturação (dias);
# aca: nota para grau de acamamento (nota);
# ap: altura de plantas (cm);
# avag: altura de inserção da primeira vagem (cm);
# rend: rendimento de grãoes (kg/ha)
# p100: peso de 100 sementes (g);
                                                                                          #
#-------------------------------------------------------------------------------------------
# lendo os dados
da <- read.table("../dados/grupoexperimentos.txt", header=TRUE, sep="\t", na.string=".")
str(da)               # estrutura dos dados importados
da$bl <- factor(da$bl) # passando bloco para fator
#-------------------------------------------------------------------------------------------
# análise exploratória
sapply(as.list(da[c("local","gen","bl")]), levels)
with(da, table(local, trat)) # todos os locais tem todos os tratamentos (trat = gen)
with(da, table(local, bl))   # mesma quantidade de blocos em cada experimento
                                                                                          #
#-------------------------------------------------------------------------------------------
# análise gráfica (para a variável rendimento)
require(lattice)
xyplot(rend~trat|local, data=da) # na cpao é mais controlado? ou é efeito de bloco?
xyplot(rend~local|gen, data=da)
                                                                                          #
#-------------------------------------------------------------------------------------------
.
```

20.2 Análise de variância separadas por local

```
.
#-------------------------------------------------------------------------------------------
# análise de variância separada por local, modelo estatístico
# y_ij = mu + bl_i + gen_j + e_ij
#  bl: efeito fixo dos i blocos, I=3
#  gen: efeito fixo dos j genótipos, J=25
#  e: erro idd normal
# fazendo análise usando subset()
levels(da$local)
cpao <- subset(da, local=="CPAO-DDOS")
with(cpao, tapply(rend, list(bl, gen), identity))
                                                                                          #
#-------------------------------------------------------------------------------------------
# algumas combinações foram perdidas isso faz com que, no modelo de efeito fixo, as
# estimativas de médias tenham diferentes precisões
m.cpao <- lm(rend~bl+gen, data=cpao) # modelo de efeito fixos
par(mfrow=c(2,2)); plot(m.cpao); layout(1)
summary(m.cpao)
                                                                                          #
#-------------------------------------------------------------------------------------------
# médias ajustadas de cada tratamento
require(contrast)
mmq <- sapply(levels(da$gen),
              function(tr){
                c0 <- contrast(m.cpao, type="average", list(bl=levels(da$bl), gen=tr))
                do.call(c, c(c0[1:7]))
              })
mmq <- t(mmq)
mmq
                                                                                          #
#-------------------------------------------------------------------------------------------
# gráfico
```

```
mmq <- as.data.frame(mmq)
mmq$gen <- rownames(mmq)
require(latticeExtra)
segplot(reorder(gen, Contrast)~Lower+Upper, data=mmq, draw.bands=FALSE, centers=Contrast)
segplot(reorder(gen, Contrast)~Lower+Upper, data=mmq, draw.bands=TRUE, centers=Contrast)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# fazendo a análise para todos os locais de uma vez só
das <- split(da, da$local) # cria uma lista em que cada slot é o data.frame de um local
str(das)
lapply(das, function(loc) with(loc, tapply(rend, list(bl, gen), identity))) # ver os NA
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# ajusta os modelos e mostra a anova
m0 <- lapply(das, FUN=aov, formula=rend~bl+gen) # ajusta para todos os locais
lapply(m0, summary)              # as análises de variância
glrs <- sapply(m0, df.residual)   # graus de liberdade
qmrs <- sapply(m0, deviance)/glrs # quadrados médios
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# pode-se obter as médias ajustadas de cada tratamento de uma vez só
mmqs <- sapply(m0, simplify=FALSE,
               function(m){
                 mmq <- sapply(levels(da$gen),
                               function(tr){
                                 c0 <- contrast(m, type="average", list(bl=levels(da$bl), gen=tr))
                                 do.call(c, c(c0[1:7]))
                               })
                 mmq <- t(mmq); mmq <- as.data.frame(mmq);
                 mmq$gen <- rownames(mmq); rownames(mmq) <- NULL; mmq[,c(6,1:5)]
               })
mmqs
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# gráfico, vamos passar tudo par um data.frame, melhor usar a ldply para list->data.frame
require(plyr)
mmqs2 <- ldply(mmqs)
mmqs2
segplot(reorder(gen, Contrast)~Lower+Upper|.id, data=mmqs2, draw.bands=FALSE, centers=Contrast)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# fazendo gráficos separados ordenados na mesma janela
mmqs2$local <- factor(mmqs2$.id)
levels(mmqs2$local) <- c("Dourados","Maracaju","Naviraí","Sidrolândia")
p1 <- segplot(reorder(gen, Contrast)~Lower+Upper|local,
              data=subset(mmqs2, .id=="CPAO-DDOS"), draw.bands=FALSE, centers=Contrast)
p2 <- segplot(reorder(gen, Contrast)~Lower+Upper|local,
              data=subset(mmqs2, .id=="MARAC"), draw.bands=FALSE, centers=Contrast)
p3 <- segplot(reorder(gen, Contrast)~Lower+Upper|local,
              data=subset(mmqs2, .id=="NAV"), draw.bands=FALSE, centers=Contrast)
p4 <- segplot(reorder(gen, Contrast)~Lower+Upper|local,
              data=subset(mmqs2, .id=="SIDROL"), draw.bands=FALSE, centers=Contrast)
print(p1, split=c(1,1,2,2), more=TRUE)
print(p2, split=c(1,2,2,2), more=TRUE)
print(p3, split=c(2,1,2,2), more=TRUE)
print(p4, split=c(2,2,2,2), more=FALSE)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
.
```

20.3 Análise conjunta dos experimentos

```
.
#-------------------------------------------------------------------------------------------------
# ajuste do modelo para análise dos resíduos
mc0 <- lm(rend~local/bl+local*gen, data=da)
par(mfrow=c(2,2)); plot(mc0); layout(1)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# ajuste do modelo para teste de hipótese, mas devido ao NA, aov() não recomendada
mc0 <- aov(rend~local/bl+gen+Error(local:gen), data=da)
summary(mc0)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# ajustar com a lm e corrir os F se for necessário
mc0 <- lm(terms(rend~local+local:bl+gen+local:gen, keep.order=TRUE), data=da)
```

```
anova(mc0) # só o F de local:gen é válido, mas se a interação é sig, não ver outros termos
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# obter as médias dos tratamentos em cada local na média dos blocos
# para usar a contrast, não pode ter "/" na fórmula, então usar "*"
mc1 <- lm(terms(rend~local+bl+local:bl+gen+local:gen, keep.order=TRUE), data=da)
c0 <- contrast(mc1, type="average",
               list(local="NAV", bl=c("1","2","3"), gen="Vmax"))
c0$X
cbind(mc1$assign, c(c0$X))
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# para todos os locais e tratamentos
mmq <- sapply(levels(da$local), simplify=FALSE,
              function(loc){
                 mmq <- sapply(levels(da$gen),
                               function(tr){
                                 c0 <- contrast(mc1, type="average",
                                                list(local=loc, bl=levels(da$bl), gen=tr))
                                 do.call(c, c(c0[1:7]))
                               })
                 mmq <- t(mmq); mmq <- as.data.frame(mmq);
                 mmq$gen <- rownames(mmq); rownames(mmq) <- NULL; mmq[,c(6,1:5)]
              })
mmq
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# fazendo o gráfico
mmq2 <- ldply(mmq)
mmq2
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# fazendo gráficos separados ordenados na mesma janela
mmq2$local <- factor(mmq2$.id)
levels(mmq2$local) <- c("Dourados","Maracaju","Naviraí","Sidrolândia")
p1 <- segplot(reorder(gen, Contrast)~Lower+Upper|local,
              data=subset(mmq2, .id=="CPAO-DDOS"), draw.bands=FALSE, centers=Contrast)
p2 <- segplot(reorder(gen, Contrast)~Lower+Upper|local,
              data=subset(mmq2, .id=="MARAC"), draw.bands=FALSE, centers=Contrast)
p3 <- segplot(reorder(gen, Contrast)~Lower+Upper|local,
              data=subset(mmq2, .id=="NAV"), draw.bands=FALSE, centers=Contrast)
p4 <- segplot(reorder(gen, Contrast)~Lower+Upper|local,
              data=subset(mmq2, .id=="SIDROL"), draw.bands=FALSE, centers=Contrast)
print(p1, split=c(1,1,2,2), more=TRUE)
print(p2, split=c(1,2,2,2), more=TRUE)
print(p3, split=c(2,1,2,2), more=TRUE)
print(p4, split=c(2,2,2,2), more=FALSE)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
.
```

20.4 Análise usando método REML

```
.
#-------------------------------------------------------------------------------------------------
# ajuste do modelo, vamos usar a lme4
require(lme4)
mm0 <- lmer(rend~gen+(1|local)+(1|local:bl)+(1|local:gen), data=da, REML=FALSE)
print(mm0, corr=FALSE)
anova(mm0)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# existe interação genotipo ambiente?
mm1 <- lmer(rend~gen+(1|local)+(1|local:bl), data=da, REML=FALSE) # sem o termo local:gen
anova(mm0, mm1) # sim!!
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# colocar o termo como fixo (o que isso significa em termos de inferênicia?)
mm2 <- lmer(rend~gen+local:gen+(1|local)+(1|local:bl), data=da, REML=FALSE)
anova(mm2)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# nessa estrutura de efeito aleatório, podemos usar a lme() para qual funciona a contrast()
# local vai entrar como fixo, criar fator local:bl
detach(package:lme4)
require(nlme)
```

```
da$local.bl <- with(da, interaction(local, bl))
mm3 <- lme(rend~local+gen+local:gen, random=~1|local.bl, data=da, na.action=na.omit)
print(mm3, corr=FALSE)
anova(mm3)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# médias para todas combinações de locais e tratamentos
mmq <- sapply(levels(da$local), simplify=FALSE,
              function(loc){
                 mmq <- sapply(levels(da$gen),
                               function(tr){
                                 c0 <- contrast(mm3, list(local=loc, gen=tr))
                                 do.call(c, c(c0[1:7]))
                               })
                 mmq <- t(mmq); mmq <- as.data.frame(mmq);
                 mmq$gen <- rownames(mmq)#; rownames(mmq) <- NULL; mmq[,c(6,1:5)]
                 mmq
              })
mmq
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# fazendo o gráfico
mmq2 <- ldply(mmq)
mmq2
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# fazendo gráficos separados ordenados na mesma janela
mmq2$local <- factor(mmq2$.id)
levels(mmq2$local) <- c("Dourados","Maracaju","Naviraí","Sidrolândia")
str(mmq2)
names(mmq2) <- gsub(".1","",names(mmq2))
p1 <- segplot(reorder(gen, Contrast)~Lower+Upper|local,
              data=subset(mmq2, .id=="CPAO-DDOS"), draw.bands=FALSE, centers=Contrast)
p2 <- segplot(reorder(gen, Contrast)~Lower+Upper|local,
              data=subset(mmq2, .id=="MARAC"), draw.bands=FALSE, centers=Contrast)
p3 <- segplot(reorder(gen, Contrast)~Lower+Upper|local,
              data=subset(mmq2, .id=="NAV"), draw.bands=FALSE, centers=Contrast)
p4 <- segplot(reorder(gen, Contrast)~Lower+Upper|local,
              data=subset(mmq2, .id=="SIDROL"), draw.bands=FALSE, centers=Contrast)
print(p1, split=c(1,1,2,2), more=TRUE)
print(p2, split=c(1,2,2,2), more=TRUE)
print(p3, split=c(2,1,2,2), more=TRUE)
print(p4, split=c(2,2,2,2), more=FALSE)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
.
```

21 Análise de dados de proporção

21.1 Latência em pêssego

```
.
#-------------------------------------------------------------------------------------------------
# dados de latência em pessêgo, nos análisados o tamanho da lesão
pes <- read.table("../dados/latencia.txt", header=TRUE, sep="\t")
str(pes)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# assumindo normalidade e ajustando o modelo total (latência 0 e 1)
m0 <- lm(lat48~., data=pes)
par(mfrow=c(2,2)); plot(m0); layout(1)
summary(m0)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# atualizando com as variáveis significativas, permitido interações
m1 <- lm(lat48~(ms0+b0)^2, data=pes)
summary(m1)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# remove a interação
m1 <- lm(lat48~ms0+b0, data=pes)
summary(m1) # veja como a interação mascara os efeitos principais!!
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# como estão as pressuposições?
```

```
par(mfrow=c(2,2)); plot(m0); layout(1)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# usar distribuição Bernoulli, ou binomial com n=1 pois a resposta é 0 ou 1
g0 <- glm(lat48~., data=pes, family=binomial)
summary(g0)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# deixar as significatvas
g1 <- glm(lat48~(ms0+b0)^2, data=pes, family=binomial)
summary(g1)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# remover a interação
g1 <- glm(lat48~ms0+b0, data=pes, family=binomial)
summary(g1)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# como estão as pressuposições?
par(mfrow=c(2,2)); plot(g1); layout(1) # veja a curvatura (glm existem vários tipos de res)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# na pior das situações, onde estou ganhando? na predição.
pred <- with(pes[complete.cases(pes),],
             expand.grid(ms0=seq(min(ms0),max(ms0),l=20),
                         b0=seq(min(b0),max(b0),l=20)))
str(pred)
pred$lat48a <- predict(g1, newdata=pred, type="response") # predição do risco
pred$lat48b <- predict(m1, newdata=pred)                  # predição de ?
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# gráfico dos valores preditos
require(lattice)
wireframe(lat48a~ms0+b0, data=pred, drape=TRUE, scales=list(arrows=FALSE),
          screen=list(z=60, x=-60))
wireframe(lat48b~ms0+b0, data=pred, drape=TRUE, scales=list(arrows=FALSE),
          screen=list(z=60, x=-60)) # valores fora do intervalo (0,1)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
.
```

21.2 Número de mudas viáveis

```
.
#-----------------------------------------------------------------------------------------------
# carregando os dados
mud <- read.table("../dados/mudas.txt", sep=";", header=TRUE)
str(mud)
mud$bloc <- factor(mud$bloc)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# gráfico
require(lattice)
xyplot(viab~trat|bloc, data=mud)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# fazendo outro data.frame com o número de viáveis
require(plyr)
mud2 <- ddply(mud, .(bloc,trat), summarise, nvia=sum(viab))
str(mud2)
xyplot(nvia~trat, groups=bloc, data=mud2, type="o")
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# análise usando glm()
g0 <- glm(cbind(nvia,20-nvia)~bloc+trat, data=mud2, family=binomial)
par(mfrow=c(2,2)); plot(g0); layout(1)
anova(g0, test="Chisq")
pchisq(deviance(g0), df=df.residual(g0), lower.tail=FALSE) # não tem superdispersão
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# obtendo o intervalo de confiança para \mu_\alpha_i, reajustar modelo com contr.sum
g1 <- glm(cbind(nvia,20-nvia)~-1+trat+bloc, data=mud2, family=binomial,
          contrast=list(bloc=contr.sum))
summary(g1)
IC.nu <- confint(g1) # perfilhamento, IC assimétrico
IC.nu # intervalo de confiança do parâmetro, temos que passar o inverso do link
```

```
IC.nu <- cbind(fit=coef(g1), IC.nu)
IC.mu <- 1/(1+exp(-IC.nu))
IC.mu # intervalo de confiança para proporção de viáveis
IC.mu <- cbind(trat=rownames(IC.mu), as.data.frame(IC.mu))
names(IC.mu)[3:4] <- c("L","U")
IC.mu <- IC.mu[grep("trat", IC.mu$trat),]
str(IC.mu)
                                                                                        #
#---------------------------------------------------------------------------------------
# gráfico, note os intervalos de confiança assimétricos a medida que afastam de 0.5
# quando mais perto de 0.5 mais amplo também
require(latticeExtra)
segplot(reorder(trat, fit)~L+U, center=fit, data=IC.mu, draw.bands=FALSE,
        segments.fun=panel.arrows, ends="both", angle=90, length=0.1, col=1)
                                                                                        #
#---------------------------------------------------------------------------------------
# podemos comparar as médias dos substratos
require(multcomp)
glht0 <- summary(glht(g1, linfct=mcp(trat="Tukey")), test=adjusted(type="bonferroni"))
glht0
let <- cld(glht0) # se for para associar as letras!
let <- let$mcletters$Letters
                                                                                        #
#---------------------------------------------------------------------------------------
# apromirando o gráfico
IC.mu <- IC.mu[order(IC.mu$fit),]       # ordena pelas médias
IC.mu$let <- sort(let, decreasing=TRUE) # poe coluna com as letras
IC.mu$tratlet <- paste(gsub("^trat","", as.character(IC.mu$trat)), # rótulo com letras
                       " (", IC.mu$let,")", sep="")
segplot(reorder(tratlet, fit)~L+U, center=fit, data=IC.mu, draw.bands=FALSE,
        segments.fun=panel.arrows, ends="both", angle=90, length=0.1, col=1)
                                                                                        #
#---------------------------------------------------------------------------------------
.
```

21.3 Número de sementes viáveis

```
.
#---------------------------------------------------------------------------------------
# dados de número de sementes viáveis de soja
soja <- read.table("../dados/soja.txt", header=TRUE, dec=",")
soja <- transform(soja, k=factor(potassio), a=factor(agua))
names(soja)[1:3] <- c("K","A","bloc")
str(soja)
                                                                                        #
#---------------------------------------------------------------------------------------
# ajuste modelo de caselas aos dados assumindo distribuição binomial (link=logit)
g0 <- glm(cbind(nv, nvi)~bloc+k*a, data=soja, family=binomial)
summary(g0)
                                                                                        #
#---------------------------------------------------------------------------------------
# quadro de análise de deviance, faz a vez da anova
anova(g0, test="Chisq")
                                                                                        #
#---------------------------------------------------------------------------------------
# obter modelo mais parcimonioso, usar fatores na forma contínua
g1 <- glm(cbind(nv, nvi)~bloc+K+A+I(K^2)+I(A^2)+K:A, data=soja, family=binomial)
summary(g1)
g1 <- update(g1, formula=.~.-K:A, family=quasibinomial)
summary(g1)
anova(g1, test="F")
                                                                                        #
#---------------------------------------------------------------------------------------
# faz a predição dos valores
pred <- with(soja,
             expand.grid(A=seq(min(A),max(A),l=20),
                         K=seq(min(K),max(K),l=20),
                         bloc="I"))
pred$prob <- predict(g1, newdata=pred, type="response")
                                                                                        #
#---------------------------------------------------------------------------------------
# função que aprimora o gráfico com as projeções das curvas no piso
panel.3d.contour <- function(x, y, z, rot.mat, distance,
                             nlevels=20, zlim.scaled, col.contour=1, ...){
  add.line <- trellis.par.get("add.line")
  panel.3dwire(x, y, z, rot.mat, distance, zlim.scaled=zlim.scaled, ...)
```

```
  clines <- contourLines(x, y, matrix(z, nrow=length(x), byrow=TRUE), nlevels=nlevels)
  for(ll in clines){
    n <- ltransform3dto3d(rbind(ll$x, ll$y, zlim.scaled[1]), rot.mat, distance)
    panel.lines(n[1,], n[2,], col=col.contour, lty=add.line$lty, lwd=add.line$lwd)
  }
}
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# gráfico
wireframe(prob~A+K, data=pred, scales=list(arrows=FALSE),
          screen=list(z=-50, x=-60), nlevels=60,
          panel.3d.wireframe="panel.3d.contour",
          par.settings=list(regions=list(alpha=0.75)), drape=TRUE)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
.
```

22 Análise de dados de contagem

22.1 Número de aves mortas

```
.
#-------------------------------------------------------------------------------------------------
# dados de mortalidade de aves em galpões com diferentes sistemas de arrefecimento
mor <- read.table("../dados/mortes.txt", header=TRUE, sep="\t")
str(mor)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# gráficos
require(lattice)
xyplot(mortes~idade|asper, groups=galpao, data=mor, type="o")
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# usando normal
m0 <- lm(mortes~asper/galpao+idade+asper:idade+asper:h, data=mor)
par(mfrow=c(2,2)); plot(m0); layout(1) # variância aumenta com a média, comum em contagens
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# modelar mortes (poisson) como função de sistema de aspersão, galpão, idade, e entalpia
g0 <- glm(mortes~asper/galpao+idade+asper:idade+asper:h,
          data=mor, family=poisson)
par(mfrow=c(2,2)); plot(g0); layout(1) # ok!
anova(g0, test="Chisq")
g0 <- update(g0, family=quasipoisson, contrast=list(galpao=contr.sum))
anova(g0, test="F")
summary(g0)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# fazer a predição, não usar os desvios devido a bloco
X <- model.matrix(g0)          # matriz de incidência completa
ass <- attr(X, "assign")       # identifica os níveis de cada fator
Xt <- X[,-which(ass==3)]       # matriz de incidência sem colunas de galpão
bt <- coef(g0)[-which(ass==3)] # vetor de estimativas sem estimativas de efeito de galpão
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# os efeitos de bloco dentro de asper somam zero devido à restrição usada
unique(X[,which(ass==3)]%*%coef(g0)[which(ass==3)])
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# fazer a predição acompanhada do intervalo de confiança
eta <- Xt%*%bt  # valores preditos na escala linear
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# artificío de algebra para obter os erro padrão mais rápido
U <- chol(vcov(g0)[-which(ass==3),-which(ass==3)])    # com isso a conta é mais eficiente
se <- sqrt(apply(Xt%*%t(U), 1, function(x) sum(x^2))) # erro padrão das estimativas em eta
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# valores preditos
tc <- qt(0.975, df.residual(g0))
pred <- cbind(mor,
              fit=c(exp(eta), exp(eta-tc*se), exp(eta+tc*se)),
              tipo=rep(c("fit","lwr","upr"), each=length(eta)))
pred$tipo.asper <- paste(pred$tipo, pred$asper)
                                                                                                    #
```

```
#-------------------------------------------------------------------------------------------
# gráfico com os valores preditos
xyplot(fit~idade, groups=tipo.asper, data=pred, type="a",
       ylim=extendrange(range(mor$mortes), f=0.05),
       xlab="Idade~das~aves~(dias)",
       ylab="Número~diário~de~aves~mortas",
       distribute.type=TRUE, lty=c(1,1,2,2,2,2),
       col=c(1,2,1,2,1,2), lwd=c(2,2,1,1,1,1),
       scales=list(x=list(at=seq(21,39,2)), y=list(at=seq(0,150,20))),
       key=list(x=0.025, y=0.9, lines=list(lty=1, lwd=2, col=2:1),
         text=list(c("Sistema convencional","Sistema com aspersão no telhado")),
         align=TRUE, transparent=TRUE),
       panel=function(...){
         panel.xyplot(...)
         panel.points((mor$idade), mor$mortes, col=mor$asper)
         panel.abline(v=seq(21,39,2), h=seq(0,150,20), col="gray90", lty=3)
       })
                                                                                          #
#-------------------------------------------------------------------------------------------
.
```