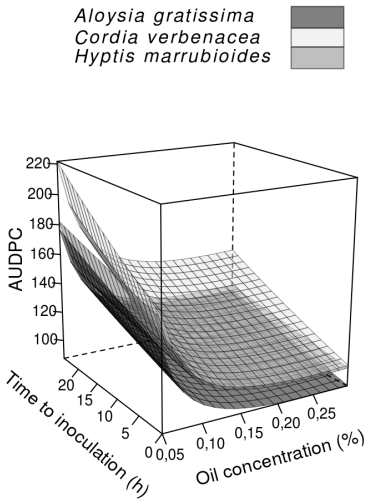

Universidade Federal da Grande Dourados

Curso de capacitação ao ambiente estatístico R

de 25 à 29 de abril de 2011

Walmes Marques Zeviani
Departamento de Estatística
Universidade Federal do Paraná

Documento concluído em 26 de abril de 2011 às 11:15:22 – Centro Politécnico – Universidade Federal do Paraná. Acesse a página do Curso em `http://www.leg.ufpr.br/cursorufgd`.

Sumário

1 Introdução à 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.h
                                                                                          #
#-------------------------------------------------------------------------------------------
# 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

```
.
#-------------------------------------------------------------------------------------------
# quando você só sabe algo sobre
apropos("tukey")
apropos("help")
                                                                                          #
#-------------------------------------------------------------------------------------------
# fazendo a busca do termo
help(TukeyHSD)
help(TukeyHSD, help_type="html")
?TukeyHSD
                                                                                          #
#-------------------------------------------------------------------------------------------
# buscando em pacotes o termo
help.search("Tukey")
??Tukey
                                                                                          #
#-------------------------------------------------------------------------------------------
# fazendo a busca na web
RSiteSearch("Tukey")
                                                                                          #
#-------------------------------------------------------------------------------------------
# procurar implementações de métodos/funções/pacotes no R
browseURL(URLencode("http://www.inside-r.org/"))
                                                                                          #
#-------------------------------------------------------------------------------------------
.
```

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 de passo 1
seq(0, 20, by=2.4)       # uma sequencia de passo 2.4
seq(0, 20, length=4)     # uma sequencia 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
v <- c(a=1, b=2, c=3)
v
length(v)  # dimensão/comprimento
class(v)   # classe
class("R")
names(v)   # nome dos elementos
                                                                                                    #
#-------------------------------------------------------------------------------------------------
m <- matrix(1:3,2,2)
m
dim(m)       # dimensões
class(m)     # classe
colnames(m) # nome das colunas
rownames(m) # nome das linhas
colnames(m) <- c("prod","peso")
rownames(m) <- c("M","F")
colnames(m)
m
                                                                                                    #
#-------------------------------------------------------------------------------------------------
d <- expand.grid(A=1:2, B=c("A","B"))
dim(d)
nrow(d); ncol(d)
names(d)
names(d) <- c("trat", "bloc")
d
                                                                                                    #
#-------------------------------------------------------------------------------------------------
l <- list(A=rnorm(4), B=matrix(1:4,2,2))
length(l)
class(l)
names(l)
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
sqrt(4) # raíz quadrada
```

```
exp(3)  # número neperiano
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
y <- x+rnorm(x,0,1)
plot(x, y)
X <- cbind(1, x)
beta <- solve(t(X)%*%X)%*%t(X)%*%y
beta  # parâmetros estimados pelo método dos mínimos quadrados
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
# operações trigonométricas, útil para transformação de dados(antigamente)
x <- seq(0, 2, 0.5)
pi
sin(x*pi)
cos(x*pi)
tan(x*pi)
asin(1)/pi
acos(-1)/pi
atan(1)/pi
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
.
```

1.6 Operações estatísticas

```
.
#-----------------------------------------------------------------------------------------------------
# em vetores
x <- rnorm(1000, 80, 3)
mean(x)         # média
sum(x)          # soma
var(x)          # variância amostral
sd(x)           # desvio padrão amostral
median(x)       # mediana
max(x)          # máximo
min(x)          # mínimo
range(x)        # extremos
diff(range(x)) # amplitude
summary(x)      # resumo: extremos, quantis e média
plot(x)         # dispersão valores ~ ordem
hist(x)         # histograma
# 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)
var(x)      # matriz de covariância amostral
cor(x)      # matriz de correlação amostral
sd(x)
apply(x, 1, 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"), nitro=c("presente","ausente"), rep=1:10)
x
x$alt <- rnorm(x$A, 1.7)
x
tapply(x$alt, x$produto, mean) # calcula a média de alt separado por níveis de produto
tapply(alt, produto, mean)
ls()                           # lista os objetos criados
with(x, tapply(alt, produto, mean))               # o mesmo de um jeito mais econômico
with(x, tapply(alt, list(produto, nitro), sum))     # fazendo para a combinação dos níveis
with(x, aggregate(alt, 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+y)^2
}
class(f0)
args(f0)
f0(3, 2)     # com escalares
f0(1:3, 0:2) # com vetores
f0(1:3, 2)   # com vetores e escalar
                                                                                                 #
#-------------------------------------------------------------------------------------------------
# exemplo: função para obtenção das raízes de uma função de 2 grau
baskara <- function(a,b,c){ # argumentos da função
  x1 <- (-b-sqrt(b^2-4*a*c))/(2*a)
  x2 <- (-b+sqrt(b^2-4*a*c))/(2*a)
  return(c(x1, x2))         # resultado da função
}
                                                                                                 #
#-------------------------------------------------------------------------------------------------
# aplicando a função criada
baskara(-3,2,1)
baskara(3,2,1)
                                                                                                 #
#-------------------------------------------------------------------------------------------------
# gráfico das funções
curve(3*x^2+2*x+1, -1, 2)             # faz curvas 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
  n3 <- 21-nota1-nota2
  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 Impotando dados

```
.
#-------------------------------------------------------------------------------------------
# como importar/ler dados?
apropos("read")
help(read.table, help_type="html")
                                                                                          #
#-------------------------------------------------------------------------------------------
# onde os dados devem estar?
getwd()                                          # o seu diretório de trabalho atual
setwd("/home/walmes/Documentos/Curso R ufgd")  # alterar o seu diretório de trabalho
                                                                                          #
#-------------------------------------------------------------------------------------------
# importando dados
#soja <- read.table("http://www.leg.ufpr.br/~walmes/cursoR/soja.txt", header=TRUE, sep="\t", dec=",")
soja <- read.table("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
                                                                                          #
#-------------------------------------------------------------------------------------------
# exploração númerica, médias por nível de potássio e potássio:água
with(soja, tapply(rengrao, list(potassio), 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
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()
                                                                                                    #
#-----------------------------------------------------------------------------------------------
.
```

2.3 Explorações gráficas (2)

```
.
#-----------------------------------------------------------------------------------------------
# lendo novos dados
#agr <- read.table("http://www.leg.ufpr.br/~walmes/cursoR/agreg.txt", header=TRUE, sep="\t")
agr <- read.table("agreg.txt", header=TRUE, sep="\t")
names(agr)
str(agr)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# qual a relação marginal entre as variáveis?
pairs(agr)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# qual a distribuição de frequência? (cores, número de classes, tipo de frequência)
hist(agr$roundness)             # Sturges
hist(agr$roundness, col="green4", breaks=seq(0.5,1,length=12), freq=FALSE)
hist(agr$roundness, col=3:4, nclass=7, freq=FALSE)
plot(density(agr$roundness))    # estimação suave a dist de freq
rug(agr$roun)                   # adiciona os traços na margem, usa abreviação
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# os dados têm distribição normal? como checar?
par(mfrow=c(1,2))
qqnorm(agr$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$roundness)), col=3)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# o que fazer? tranformar? qual tranformação? raiz? log?
require(MASS)  # faz a requisição do pacote MASS, suas funções estarão disponíveis
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# faz a estimação do parâmetro lambda para aplicar a transformação boxcox
agr5 <- subset(agr, profundidade==5) # atribui dados à um objeto
boxcox(agr5$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 Recursos gráficos avançados, notas sobre normalidade e correlação

```
.
#-------------------------------------------------------------------------------------------------
# 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$profundidade)
                                                                                                 #
#-------------------------------------------------------------------------------------------------
# nota importante sobre normalidade! os dados abaixo têm distribuição normal?
m <- gl(15,8)
x <- rnorm(m, as.numeric(m), 0.1)
xp <- qqnorm(x); qqline(x)
rug(xp$x)
rug(xp$y, side=2)
m0 <- lm(x~m)
xp <- qqnorm(residuals(m0)); qqline(residuals(m0))
rug(xp$x)
rug(xp$y, side=2)
shapiro.test(x)              # ver sobre esse teste o grau de liberdade
shapiro.test(residuals(m0))
                                                                                                 #
#-------------------------------------------------------------------------------------------------
# gráfico interativo que facilita o entendimento do conceito acima
require(manipulate)
par(mfrow=c(2,1))
manipulate({
           m <- rep(seq(0,by=h1,length.out=nlev), nrep)
           x <- rnorm(m, m, sd)
           xp <- qqnorm(x); qqline(x)
           rug(xp$x); rug(xp$y, side=2)
           legend("topleft", legend=shapiro.test(x)$p, bty="n")
           m0 <- lm(x~factor(m))
           xp <- qqnorm(residuals(m0)); qqline(residuals(m0))
           rug(xp$x);rug(xp$y, side=2)
           legend("topleft", bty="n",
                  legend=shapiro.test(residuals(m0))$p)
         },
```

```
            h1=slider(0.001, 10, initial=1),
            nlev=slider(2, 15, initial=5),
            nrep=slider(2, 25, initial=5),
            sd=slider(0.01, 10, initial=1))
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# o mesmo vale para correlações
dose <- rep(3*1:5, each=20) # doses aplicadas as parcelas, tocar o 3, pelo 6
y1 <- dose+rnorm(100)       # amostra aleatória da variável 1
y2 <- dose+rnorm(100)       # amostra aleatória da variável 2
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# dispersão das duas respostas observadas no experimento
plot(y1~y2)
cor.test(y1, y2) # teste da correlação aplicado ERRADO
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# dispersão das dos resíduos do experimento, retira-se o efeito da dose
plot((y1-dose)~I(y2-dose))
cor.test(y1-dose, y2-dose) # teste da correlação CERTO (ainda requer correção do gl)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# gráfico que ilusta o conceito
require(MASS)
par(mfrow=c(2,1))
manipulate({
             m <- rep(seq(0,by=h1,length.out=nlev), nrep)
             x <- mvrnorm(length(m), mu=c(0,0),
                          Sigma=matrix(c(1,cor,cor,1),2,2))
             x[,1] <- x[,1]+m; x[,2] <- x[,2]+m
             plot(x)
             legend("topleft", legend=cor.test(x[,1], x[,2])$est, bty="n")
             m0 <- aov(x~factor(m))
             r <- residuals(m0)
             plot(r)
             legend("topleft", legend=cor.test(r[,1], r[,2])$est, bty="n")
           },
            h1=slider(0,19.99,initial=0.01),
            nrep=slider(10,300,initial=20),
            nlev=slider(2,15,initial=5),
            cor=slider(-0.99,0.99,initial=0))
                                                                                                    #
#-----------------------------------------------------------------------------------------------
.
```

3 Estatística básica

3.1 Medidas de posição, dispersão, forma e distribuição de frequências

```
.
#-----------------------------------------------------------------------------------------------
# média amostral (notas de alunos)
x <- c(7.5, 6.8, 5.3, 6.1, 6.3, 8.5, 7.3, 5.2, 5.9, 5.2,
       5.2, 7.5, 6.9, 5.8, 5.8, 8.0, 8.7, 7.8, 7.1, 7.1,
       5.9, 7.3, 7.5, 6.6, 6.5)
mean(x)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# média ponderada amostral (notas de um aluno em provas)
y <- c(3,6,8,9)
p <- c(1,2,3,4)
weighted.mean(y, p)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# variância (desvio padrão) amostral
var(x)
sd(x)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# mediana amostral
median(x)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# desvio absoluto da mediana
sum(abs(x-median(x)))/length(x)
```

```
                                                                                                                    #
#-------------------------------------------------------------------------------------------------------------------
# coeficiente de variação (cv)
100*sd(x)/mean(x)
                                                                                                                    #
#-------------------------------------------------------------------------------------------------------------------
# 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
                                                                                                                    #
#-------------------------------------------------------------------------------------------------------------------
# distribuição de frequências absoluta
hist(x)
                                                                                                                    #
#-------------------------------------------------------------------------------------------------------------------
# distribuição de frequências relativas, controle da amplitude de classe
hist(x, freq=FALSE, breaks=seq(5, 9, length=7))
rug(x)
curve(dnorm(x, mean(x), sd(x)), add=TRUE, col="green4")
lines(density(x), col=2)
                                                                                                                    #
#-------------------------------------------------------------------------------------------------------------------
# distribuição de frequências acumuladas
h <- hist(x, plot=FALSE, breaks=10)
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(x)
                                                                                                                    #
#-------------------------------------------------------------------------------------------------------------------
# distribuição acumulada empírica
plot(ecdf(x))
rug(x)
curve(pnorm(x, mean=mean(x), sd=sd(x)), col=2, add=TRUE)
                                                                                                                    #
#-------------------------------------------------------------------------------------------------------------------
# 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)
                                                                                                                    #
#-------------------------------------------------------------------------------------------------------------------
# separatrizes, quantis e os 5 números de Tukey
fivenum(x)
quantile(x, c(5,50,95)/100) # possui diverso métodos de interpolação
                                                                                                                    #
#-------------------------------------------------------------------------------------------------------------------
# amplitude total e interquartilica
range(x)       # extremos
diff(range(x)) # amplitude total
IQR(x)         # aplitude interquetilica
                                                                                                                    #
#-------------------------------------------------------------------------------------------------------------------
# gráfico de caixas (boxplot)
boxplot(x)
                                                                                                                    #
#-------------------------------------------------------------------------------------------------------------------
# gráfico de caixa entalhado (notched boxplot)
boxplot(x, notch=TRUE)
                                                                                                                    #
```

```
#-------------------------------------------------------------------------------------------
# critério de expulsão dos pontos do boxplot
require(manipulate) # carrega o pacote para manipulação de variáveis gráficas
manipulate({
             x <- rep(1:10, 2)
             x[20] <- extreme
             gr <- gl(2, 10)
             bp <- boxplot(x~gr, outline=outline, range=range,
                           notch=notch, plot=FALSE)
             inf <- bp$stats[4,2]
             sup <- inf+range*diff(bp$stats[c(2,4),2])
             ylim <- extendrange(r=c(min(x), max(c(x,sup))), f=0.05)
             boxplot(x~gr, outline=outline, range=range,
                     notch=notch, ylim=ylim)
             arrows(1.5, inf, 1.5, sup, angle=90, code=3, length=0.1)
           },
           extreme=slider(10, 30, step=0.5, initial=10),
           range=slider(1, 4, step=0.1, initial=1.5),
           outline=checkbox(TRUE, "show.outlier"),
           notch=checkbox(FALSE, "show.interval"))
                                                                                          #
#-------------------------------------------------------------------------------------------
# momento naftalina: diagrama de ramos e folhas
stem(x)
                                                                                          #
#-------------------------------------------------------------------------------------------
.
```

3.2 Teste de hipótese e intervalo de confiança para parâmetros

```
.
#-------------------------------------------------------------------------------------------
# quantos testes o R têm?
apropos("test") # funções que possuem a palavra "test" no nome
                                                                                          #
#-------------------------------------------------------------------------------------------
# para a média de uma população normal
t.test(x, mu=7)
                                                                                          #
#-------------------------------------------------------------------------------------------
# para uma proporção (germinação amostral)
germ <- 82                      # germinação observada na amostra
prop.test(germ, 100, p=0.90)  # testa se a germinação é 0.9 em 100 ensaios
                                                                                          #
#-------------------------------------------------------------------------------------------
# teste para a igualdade de duas médias de dados normais
cA <- c(7.8, 6.7, 8, 7, 6.1, 7.7, 7.1, 6.6, 8.9, 5.3, 6.9, 7.9)
cB <- c(4.6, 5.7, 4.5, 5.5, 5, 3.8, 3.3, 6, 3.5, 5.1, 4.2, 4.3)
t.test(cA, cB, var=TRUE)  # especifica as variâncias são iguais
t.test(cA, cB, var=FALSE) # aproximação de Welch para variâncias diferentes
                                                                                          #
#-------------------------------------------------------------------------------------------
# teste para igualdade de duas variâncias de dados normais
var.test(cA, cB)
help(bartlett.test) # usado após anova
                                                                                          #
#-------------------------------------------------------------------------------------------
# teste para a normalidade de uma amostra (deve ter única média é variância)
shapiro.test(x)
                                                                                          #
#-------------------------------------------------------------------------------------------
# teste para aderência de dados à qualquer distribuição
ks.test(scale(x), "pnorm")
ks.test(x, "pnorm", mean=mean(x), sd=sd(x))
                                                                                          #
#-------------------------------------------------------------------------------------------
# teste para correlação entre duas variáveis normais (assume-se 1 média para cada)
#agr <- read.table("http://www.leg.ufpr.br/~walmes/cursoR/agreg.txt", header=TRUE, sep="\t")
area5 <- agr$area[agr$profundidade==5]
roun5 <- agr$roundness[agr$profundidade==5]
plot(area5, roun5)
cor.test(area5, roun5)
                                                                                          #
#-------------------------------------------------------------------------------------------
# teste de aderência de qui-quadrado (os acidentes de trabalho ocorrem uniforme na semana?)
ac <- c(seg=32, ter=40, qua=20, qui=25, sex=33)
```

```
chisq.test(ac)
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
# outro teste de aderência, será que esses dados tem distribuição normal? beta?
qqnorm(roun5); qqline(roun5)
plot(density(roun5)); rug(roun5)
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
# se forem beta, quais são os parâmetros? (obtive esses parâmetros via maximização da f.v.)
plot(density(roun5)); rug(roun5)
curve(dbeta(x, shape1=18.4, shape2=3.4), col="red", add=TRUE)
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
# teste para a normal e para a beta (ver função MASS::fitdistr())
ks.test(roun5, "pnorm", mean=mean(roun5), sd=sd(roun5))
ks.test(roun5, "pbeta", shape1=18.4, shape2=3.4)
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
# visualização gráfica das distribuições
plot(ecdf(roun5), pch=NULL, cex=0.2)
rug(roun5)
curve(pnorm(x, mean(roun5), sd(roun5)), add=TRUE, col=2)
curve(pbeta(x, shape1=18.4, shape2=3.4), add=TRUE, col=3)
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
# como calcular a estatística do teste
x <- sort(roun5)                                  # ordena a amostra
n <- length(x)                                    # tamanho da amostra
ecdf <- sapply(x, function(i){ 1/n*sum(x<=i) }) # prob acu empírica
p.norm <- pnorm(x, mean=mean(x), sd=sd(x))        # prob acu pela normal
p.beta <- pbeta(x, shape1=18.4, shape2=3.4)       # prob acu pela beta
max(abs(ecdf-p.norm))                             # estatística do teste
max(abs(ecdf-p.beta))                             # estatística do teste
wn <- which.max(abs(ecdf-p.norm))                 # index das maiores distâncias
wb <- which.max(abs(ecdf-p.beta))
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
# gráfico para interpretar a estatística do teste
plot(ecdf(roun5), pch=NULL, cex=0.2)              # gráfico da prob acu empírica
rug(roun5)
curve(pnorm(x, mean(roun5), sd(roun5)), add=TRUE, col=2)
curve(pbeta(x, shape1=18.4, shape2=3.4), add=TRUE, col=3)
segments(x[wn], p.norm[wn], x[wn], ecdf[wn], col=2, lwd=3); abline(v=x[wn], lty=3)
segments(x[wb], p.norm[wb], x[wb], ecdf[wb], col=3, lwd=3); abline(v=x[wb], lty=3)
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
.
```

4 Regressão linear

4.1 Importando e manipulando dados

```
.
#-----------------------------------------------------------------------------------------------------
# importando dados
#dap <- read.table("http://www.leg.ufpr.br/~walmes/cursoR/dap.txt", header=TRUE, sep="\t")
dap <- read.table("dap.txt", header=TRUE, sep="\t")
str(dap)
names(dap) <- c("d","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
pairs(dap)
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
# ordena os dados e deixa apenas os registros completos (linha cheia)
dap <- dap[order(dap$d),]
dapcc <- dap[complete.cases(dap),]
rownames(dapcc) <- NULL
```

```
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
plot(m0)           # apresenta os gráficos de diagnóstico
layout(1)          # volta a janela gráfica para 1 gráfico
                                                                                                    #
#-------------------------------------------------------------------------------------------------
.
```

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)
layout(matrix(c(1,1,2,3,4,5),2,3))
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)
plot(h~d, dapcc)
lines(fitted(m2)~d, dapcc, col=2)
plot(m2)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# modelo recíproco
m3 <- lm(h~d+di, data=dapcc)
summary(m3)
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)
```

```
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 Procedimentos para seleção de modelos/variáveis

```
.
#-----------------------------------------------------------------------------------------------------
# modelo com todas as variáveis
m7 <- lm(h~., data=dapcc)
summary(m7)
plot(h~d, dapcc); lines(fitted(m7)~d, dapcc, col=2); plot(m7)
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
# 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)
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
# modelo m5 foi escolhido pelo critério BIC
summary(m5)
anova(m5)
plot(h~d, dapcc); lines(fitted(m5)~d, dapcc, col=2); plot(m5)
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
.
```

4.5 Estudo e remoção de pontos discrepantes/influentes

```
.
#-----------------------------------------------------------------------------------------------------
# medidas de influencia
inf <- influence.measures(m5)
inf
summary(inf)
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
# sinalizando os pontos influentes
str(inf)         # estrutura do objeto
dfits <- inf$is.inf[,4]  # pontos que são influentes pelo DFITS
layout(1)
plot(h~d, dapcc)
lines(fitted(m5)~d, dapcc, col=2)
with(dapcc, points(d[dfits], h[dfits], col=2, pch=19))
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
# identificar/remover os pontos discrepantes/influentes manualmente
layout(1)
plot(residuals(m5)~d, dapcc)
id <- identify(dapcc$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)
```

```
layout(matrix(c(1,1,2,3,4,5),2,3))
plot(h~d, dapcc2); lines(fitted(m5b)~d, dapcc2, col=2); plot(m5b)
layout(1)
plot(m5b, which=2)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# devemos tranformar? qual transformação usar?
require(MASS)
layout(1)
bc <- boxcox(m5b, lambda=seq(0.5,2,l=100))
str(bc)
bc$x[which.max(bc$y)]
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# qual o resultado dos testes?
shapiro.test(rstudent(m5b))
ks.test(rstudent(m5b), "pnorm")
                                                                                                    #
#-------------------------------------------------------------------------------------------------
.
```

4.6 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(m5, newdata=dap)
str(hpred)
dap$hpred <- hpred
str(dap)
write.table(dap, "dap.xls", sep="\t", quote=FALSE, row.names=FALSE, dec=".")
                                                                                                    #
#-------------------------------------------------------------------------------------------------
.
```

4.7 Representação gráfica do ajuste

```
.
#-------------------------------------------------------------------------------------------------
# escolhendo o intervalo de predição
range(dapcc2$d)
d.new <- seq(4, 30, length=100)
d.new
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# fazendo predição com intervalo de confiança e predição futura
Yp <- predict(m5b, newdata=data.frame(d=d.new, dr=sqrt(d.new)), interval="confidence")
Yf <- predict(m5b, newdata=data.frame(d=d.new, dr=sqrt(d.new)), interval="prediction")
head(Yp)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# plotando
layout(1)
plot(h~d, dapcc2, xlab="DAP (cm)", ylab="Altura (m)")
matlines(d.new, Yp, col=c(1,2,2), lty=c(1,2,2))
matlines(d.new, Yf, col=c(1,3,3), lty=c(1,3,3))
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# fazendo anotações dentro do gráfico
legend("topleft", c("Predito","ICpredito","ICobsfutura"),
       lty=c(1,2,3), col=c(1,2,3), bty="n")
co <- format(c(coef(m5b), summary(m5)$r.squared), digits=3)
co
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")
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# mais sobre gráficos no R
demo(plotmath)
demo(graphics)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
.
```

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

```
.
#-------------------------------------------------------------------------------------------
# mais sobre resíduos e R2
data(anscombe)
ans1 <- lm(y1~x1, anscombe)
ans2 <- lm(y2~x2, anscombe)
ans3 <- lm(y3~x3, anscombe)
ans4 <- lm(y4~x4, anscombe)
summary(ans1)
summary(ans2)
summary(ans3)
summary(ans4)
                                                                                          #
#-------------------------------------------------------------------------------------------
# gráficos
par(mfrow=c(4,5), oma=c(0,0,0,0), mar=c(2,2,2,2))
plot(y1~x1, anscombe); abline(ans1, col=2); plot(ans1)
plot(y2~x2, anscombe); abline(ans2, col=2); plot(ans2)
plot(y3~x3, anscombe); abline(ans3, col=2); plot(ans3)
plot(y4~x4, anscombe); abline(ans4, col=2); plot(ans4)
                                                                                          #
#-------------------------------------------------------------------------------------------
# o significado dos leverages
hatvalues(ans1)
sapply(list(ans1, ans2, ans3, ans4), hatvalues)
                                                                                          #
#-------------------------------------------------------------------------------------------
# mais sobre medidas de influência (animação)
library(gWidgetsRGtk2)
ans <- anscombe[,c(1,5)]
ans <- ans[order(ans$x1),]
anscombe <- anscombe[order(anscombe$x1),]
rownames(ans) <- NULL
rownames(anscombe) <- NULL
ji <- 8
#anscombe[as.numeric(ji),]
#ans[ji,]
ans$color <- 1; ans$color[ji] <- 2
limits <- list(x=c(-2,2), y=c(-2,2))
plot.dist <- function(...){
  ans[ji,1:2] <- as.numeric(anscombe[ji,c(1,5)])+c(svalue(x), svalue(y))
  plot(y1~x1, data=ans, col=ans$color, pch=19)
  abline(lm(y1~x1, anscombe), lty=3)
  abline(lm(y1~x1, ans), col=2)
}
w <- gwindow("Controle os parÃ¢metros")
tbl <- glayout(cont=w)
for(i in 1:length(limits)){
  tbl[i,1] <- paste("Deslizador para", 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]])/30,
                     value=mean(limits[[i]]),
                     container=tbl, handler=plot.dist)))
}
layout(1)
plot.dist()
                                                                                          #
#-------------------------------------------------------------------------------------------
.
```

4.9 Sobre interpretação de modelos de regressão linear

```
#-------------------------------------------------------------------------------------------
# simulando uma resposta gerada por polinômio com a seguinte lei
b0 <- 1; b1 <- 5; b2 <- -0.2; b3 <- 0.002    # valores paramétricos/populacionais
x <- seq(0,75,by=5)                          # valores de x avaliados
y <- b0+b1*x+b2*x^2+b3*x^3+rnorm(x,0,12)     # 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,])
                                                                                          #
#-------------------------------------------------------------------------------------------
.
```

5 Regressão não linear

5.1 Motivação

```
.
#-------------------------------------------------------------------------------------------
# dados de motivação
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)
                                                                                          #
#-------------------------------------------------------------------------------------------
.
```

5.2 Definição

```
.
#-------------------------------------------------------------------------------------------
# as derivadas de y em relação a theta não são livres de theta
# y = A*x/(B+x) : modelo michaelis mentem
D(expression(A*x/(B+x)),"A")
D(expression(A*x/(B+x)),"B")
                                                                                          #
```

```
#-------------------------------------------------------------------------------------------------
# y = A/(1+exp(B+C*x)) : modelo logístico simples
D(expression(A/(1+exp(B+C*x))),"A")
D(expression(A/(1+exp(B+C*x))),"B")
D(expression(A/(1+exp(B+C*x))),"C")
                                                                                                 #
#-------------------------------------------------------------------------------------------------
.
```

5.3 Exemplo de modelos não lineares

```
.
#-------------------------------------------------------------------------------------------------
# modelo michaelis mentem
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)
                                                                                                 #
#-------------------------------------------------------------------------------------------------
# modelo logístico
A <- 10; B <- 25; C <- 3
curve(A/(1+exp((B-x)/C)), 0, 50, col=2, lwd=3)
abline(h=c(A, A/2), v=B, lty=3)
                                                                                                 #
#-------------------------------------------------------------------------------------------------
# modelo resposta platô
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)
A <- 10; B <- 2; C <- 0.5
curve(x*(A+B*x)^(-1/C), 0, 50, col=2, lwd=3)
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 no solo (van Genuchten, 1980)
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)
curve(eval(D(expression(B+(A-B)/(1+(C*10^x)^D)^(1-1/D)), "x")), -3, 3)
                                                                                                 #
#-------------------------------------------------------------------------------------------------
.
```

5.4 Uso de recursos gráficos para entender o significado dos parâmetros

```
.
#-------------------------------------------------------------------------------------------------
# pacote que permite a construção de interfaces gráficas
library(gWidgetsRGtk2)
                                                                                                 #
#-------------------------------------------------------------------------------------------------
# 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) }
                                                                                                 #
#-------------------------------------------------------------------------------------------------
# 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(10,60), C=c(1,7))
plottest <- function(...){ curve(svalue(A)/(1+exp((svalue(B)-x)/svalue(C))), 0, 50) }
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(...){
  curve(svalue(A)+svalue(B)*x*(x<svalue(x0))+svalue(B)*svalue(x0)*(x>=svalue(x0)), 0, 20)
}
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(...){ curve(x*(svalue(A)+svalue(B)*x)^(-1/svalue(C)), 0, 50) }
source("func.R")
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
# modelo de curva de água no solo (van Genuchten, 1980)
limits <- list(A=c(0.5,0.8), B=c(0.1,0.3), C=c(0.5,1.5), D=c(1,2))
plottest <- function(...){
  curve(svalue(B)+(svalue(A)-svalue(B))/(1+(svalue(C)*10^x)^svalue(D))^(1-1/svalue(D)),
        -3, 4)
}
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 <- 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)
#dap <- read.table("http://www.leg.ufpr.br/~walmes/cursoR/dap.txt", header=TRUE)
dap <- read.table("dap.txt", header=TRUE)
names(dap) <- c("d","h")
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# 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()
limits <- list(b0=c(25,35), b1=c(0,0.5), b2=c(0.7, 1.3))
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 bons chutes)
n0 <- nls(h~b0*(1-exp(-b1*d))^b2, data=dapcc,
          start=list(b0=35, b1=0.1, b2=1.3), trace=TRUE)
n0 <- nls(h~b0*(1-exp(-b1*d))^b2, data=dapcc, start=start, trace=TRUE)
summary(n0)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# ajustar o modelo não linear (com chutes sem noção)
n0 <- nls(h~b0*(1-exp(-b1*d))^b2, data=dapcc,
          start=list(b0=35, b1=0, b2=1.3), trace=TRUE)
n0 <- nls(h~b0*(1-exp(-b1*d))^b2, data=dapcc,
          start=list(b0=35, b1=-1, b2=1.3), trace=TRUE)
n0 <- nls(h~b0*(1-exp(-b1*d))^b2, data=dapcc,
          start=list(b0=35, b1=0.1, b2=-1), trace=TRUE)
```

```
#                                                                                                    #
#----------------------------------------------------------------------------------------------------
# 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)
#                                                                                                    #
#----------------------------------------------------------------------------------------------------
# intervalo de confiança para as estimativas
confint.default(n0)  # observar os valores para entender o que o perfilhamento
confint(n0)          # intervalo assintótico sempre é simétrico
help(confint, help_type="html")
#                                                                                                    #
#----------------------------------------------------------------------------------------------------
# o intervalo de confiança perfilhado para um parâmetro
prof <- profile(n0)
prof$b0
layout(1)
plot(prof$b0[,1]~prof$b0[,2][,1])
abline(h=c(-1.95,1.95), v=c(29.84, 36.96))
#                                                                                                    #
#----------------------------------------------------------------------------------------------------
# o intervalo de confiânça de b2 contém o 1, será que preciso de b2?
n1 <- nls(h~b0*(1-exp(-b1*d)), data=dapcc, start=list(b0=30, b1=0.1))
summary(n1)
#                                                                                                    #
#----------------------------------------------------------------------------------------------------
# como ficou?
layout(1)
plot(h~d, dapcc)
lines(fitted(n0)~d, dapcc, col=2)
lines(fitted(n1)~d, dapcc, col=3)
#                                                                                                    #
#----------------------------------------------------------------------------------------------------
# teste da razão de verossimilhança para exclusão de b2
anova(n1, n0)
#                                                                                                    #
#----------------------------------------------------------------------------------------------------
# comparar o ajuste do modelo não linear com o linear escolhido na aula passada
-2*c(logLik(n1))+2*2
# AIC (k=2 parâmetros)
# -2*c(logLik(m5))+3*2
# 966.5362 (k=3 parâmetros)
-2*c(logLik(n1))+2*log(221)
# BIC (k=2 parâmetros)
-2*c(logLik(m5))+3*log(221)
#                                                                                                    #
#----------------------------------------------------------------------------------------------------
# R2 em modelos não lineares (danger!)
R2 <- function(nls.obj){
  da <- eval(nls.obj$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)
#                                                                                                    #
#----------------------------------------------------------------------------------------------------
.
```

5.7 Comparação de curvas ajustadas

```
.
#-------------------------------------------------------------------------------------------
# dados
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)
                                                                                           #
#-------------------------------------------------------------------------------------------
# as estimativas de A são tão próximas, será que direrem?
confint.default(nAB) # baseado em normalidade assintótica
confint(nAB)         # baseado em perfil de verossimilhança
                                                                                           #
#-------------------------------------------------------------------------------------------
# ajustar um modelo em que A seja comum aos 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 a restrição em A
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
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))
```

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

5.8 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=c(0,1,5,10))
nema$eclod <- c(13, 56.5, 97.5, 168, 246.5, 323, 374, 389, 7, 26, 64.5, 126, 207.5,
                282, 334, 343, 5, 21.5, 45.5, 79, 118.5, 146, 167.5, 174.5, 3.25,
                9.25, 12.5, 20.5, 32.25, 39.25, 40.25, 42.25)
xyplot(eclod~dia, groups=dose, data=nema, type="b", auto.key=TRUE)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# carrega o pacote nlme (do grupo dos recomendados)
require(nlme)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# ajuste das curvas em uma única função (usando função selfstart)
gn0 <- gnls(eclod~SSlogis(dia, Asym, xmid, scal),
            data=nema,
            params=Asym+xmid+scal~factor(dose),
            start=c(500,-100,-200,-400, 4.4,0,0,0, 1.13,0,0,0))
summary(gn0)
anova(gn0, type="marginal")  # é um teste de Wald
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# novos valores de dia para a predição de eclod
new <- expand.grid(dia=seq(0,20,0.2), dose=c(0,1,5,10))
new$eclod <- predict(gn0, newdata=new)
xyplot(eclod~dia, groups=dose, data=new)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# 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 que ir preso!",
       sub="Os gráficos do R são infinitamente melhores!",
       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")
       )
                                                                                                    #
#-----------------------------------------------------------------------------------------------
.
```

5.9 Ajuste do modelo duplo van Genuchten

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

```
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
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
# procedimento gráfico para obter bons chutes e conhecer a função dos parâmetros
require(manipulate)  # carrega pacote que permite manipulação gráfica
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 <- 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), 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
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
.
```

5.10 Secagem do solo em micro ondas

```
.
#-----------------------------------------------------------------------------------------------------
# dados de umidade em função do tempo (não é medida repetida)
umi <- read.table("emr11.txt", header=TRUE, sep="\t")
str(umi)
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
# exploração gráfica
require(lattice)
xyplot(umid~tempo|nome, data=umi, type=c("p","smooth"))
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
# os solos foram secos em estuda e adicionados 40% de umidade, será que o micro remove ela?
xyplot(umrel~tempo|nome, data=umi, type=c("p","smooth"))
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
# fazer um intervalo de confiança para a umidade em 40 minutos, ajustar modelo logístico
```

```
require(nlme) # permite um ajuste conjunto com mais facilidade
n0 <- nlsList(umrel~SSlogis(tempo, A, x0, S)|nome, data=umi)
summary(n0)
coef(n0)      # qual a interpretaçãs dos parâmetros?
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# prepara os dados preditos
pred <- expand.grid(tempo=0:45, nome=levels(umi$nome))
pred$umrel <- predict(n0, newdata=pred)
pred$tipo <- "predito"
umi$tipo <- "observado"
pred <- rbind(pred, umi[,c("nome","tempo","umrel","tipo")])
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# o gráfico
xyplot(umrel~tempo|nome, groups=tipo, data=pred,
       distribute.type=TRUE, type=c("p","l"))
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# a hipótese 1 era saber se 40 minutos era suficiente para secar o solo
lvadbw <- nls(umrel~SSlogis(tempo, A, x0, S), data=subset(umi, 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$tempo <- umi$tempo[umi$nome=="LVAd-Bw"]
pred2$umrel <- umi$umrel[umi$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)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# será que os alguns solos possuem o mesmo padrão de secamento?
                                                                                                    #
#-------------------------------------------------------------------------------------------------
.
```

6 Análise de experimento com um fator em DIC

6.1 Importando dados

```
.
#-------------------------------------------------------------------------------------------------
# entrada de dados direto no script
Lines <-
  "gen  diam
ATF06B 0.713
ATF06B 0.635
ATF06B 0.757
ATF40B 0.621
ATF40B 0.527
ATF40B 0.640
ATF54B 0.559
ATF54B 0.446
ATF54B 0.616
BR001B 0.734
BR001B 0.635
BR001B 0.763
BR005B 0.597
BR005B 0.415
BR005B 0.460
BR007B 0.601
BR007B 0.506
BR007B 0.623
BR008B 0.724
```

```
BR008B 0.574
BR008B 0.663
P9401  0.686
P9401  0.440
P9401  0.588
SC283  0.632
SC283  0.610
SC283  0.650"
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)
# http://cran.r-project.org/
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
# conferir se temos fatores para fazer a análise de variância
is.factor(raiz$gen)
is.numeric(raiz$gen)
is.character(raiz$gen)
class(raiz$gen)
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
.
```

6.2 Análise de variância

```
.
#-----------------------------------------------------------------------------------------------------
# fazendo a análise de variância
a0 <- aov(diam~gen, data=raiz)
anova(a0)
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
# análise gráfica dos resíduos
par(mfrow=c(2,2))
plot(a0)
layout(1)
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
# teste das pressuposições da análise de variância
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
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
.
```

6.3 Aplicando teste de Tukey para comparar médias

```
.
#-----------------------------------------------------------------------------------------------------
# teste de médias (Tukey), igredientes: QMR e GLR
df.residual(a0)               # grau de liberdade residual (GLR)
deviance(a0)                  # soma de quadrado dos resíduos
deviance(a0)/df.residual(a0)  # quadrado médio do resíduo (QMR)
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
# carregando a biblioteca necessária para fazer o teste de Tukey com letras
require(agricolae)
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
# aplicando o teste de Tukey
with(raiz,
     HSD.test(diam, gen,
              DFerror=df.residual(a0),
              MSerror=deviance(a0)/df.residual(a0), alpha=0.05)
     )
```

```
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# outra função, outra representação
TukeyHSD(a0)
plot(TukeyHSD(a0))
abline(v=0)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
.
```

6.4 Aplicando teste de Scott-Knott para agrupar médias

```
.
#-------------------------------------------------------------------------------------------------
# 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)
summary(sk)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
.
```

6.5 Aplicando contrastes

```
.
#-------------------------------------------------------------------------------------------------
# biblioteca para fazer contrates; o modelo precisa ser classe "lm" e não "aov"
# install.packages("contrast")
require(contrast)
a0 <- lm(diam~gen, data=raiz)
class(a0)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# um nível contra o outro
c0 <- contrast(a0, list(gen="ATF06B"), list(gen="ATF40B"))
c0                  # resultado do contraste
levels(raiz$gen)    # níveis/categorias do fator
coef(a0)            # estimaitvas dos efeitos (sob particular restrição)
c0$X                # matriz que indentifica o contraste
summary(a0)         # quadro de estimativas
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# um grupo de níveis contra outro
c1 <- contrast(a0, type="average", # estima a média desses 4 níveis
               list(gen=c("BR001B","ATF06B","BR008B","SC283")))
c2 <- contrast(a0, type="average", # estima a média desses 5 níveis
               list(gen=c("ATF40B","BR007B","P9401","ATF54B","BR005B")))
c1                      # quadro de estimativa
c2                      # quadro de estimativa
c1$X-c2$X               # vetor do contraste entre grupos
(c1$X-c2$X)%*%coef(a0)  # estimativa do contraste
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# biblioteca para controlar o erro tipo I de comparações multiplas de hipóteses
require(multcomp)
summary(glht(a0, linfct=(c1$X-c2$X)))  # no caso de um único contraste não é necessário
                                                                                                    #
#-------------------------------------------------------------------------------------------------
.
```

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

```
#-------------------------------------------------------------------------------------------------
# carrega o pacote (versão em inglês)
require(ExpDes)
```

```
help(package="ExpDes")
                                                                                                    #
#---------------------------------------------------------------------------------------------------
# obtem todos os resultados de maneira fácil
with(raiz, crd(gen, diam))
                                                                                                    #
#---------------------------------------------------------------------------------------------------
.
```

6.7 Análise com perda de parcelas

```
#---------------------------------------------------------------------------------------------------
# fazendo o descarte de observações que simula a perda aleatória de 3 parcelas
na <- sample(1:nrow(raiz), 3)
raiz$diam[na] <- NA
raiz2 <- raiz[complete.cases(raiz),]
with(raiz2, tapply(diam, gen, length))
                                                                                                    #
#---------------------------------------------------------------------------------------------------
# análise de variância
a0 <- aov(diam~gen, raiz2)
anova(a0)
summary(a0)
                                                                                                    #
#---------------------------------------------------------------------------------------------------
# obtendo as estimativas de todos os contrastes de Tukey
TukeyHSD(a0)
                                                                                                    #
#---------------------------------------------------------------------------------------------------
# usando o teste de Tukey com a média harmônica do número de repetições
with(raiz2,
     HSD.test(diam, gen,
              DFerror=df.residual(a0),
              MSerror=deviance(a0)/df.residual(a0), alpha=0.05)
     )
                                                                                                    #
#---------------------------------------------------------------------------------------------------
.
```

6.8 Estudo das taxas de erro tipo I dos testes

```
.
#---------------------------------------------------------------------------------------------------
# função que gera experimentos em DIC sob H0
trat <- gl(6,4)
geraov <- function(...){
  y <- rnorm(length(trat))
  m0 <- lm(y~trat)
  s0 <- summary(m0)
  t1 <- abs(coef(m0)[2]*sqrt(2))/s0$sigma
  t2 <- diff(range(0, coef(m0)[-1]))*sqrt(2)/s0$sigma
  return(c(t1,t2))
}
geraov()
                                                                                                    #
#---------------------------------------------------------------------------------------------------
# gerando 1000 experimentos aleatórios
exper <- abs(replicate(2000, geraov()))
                                                                                                    #
#---------------------------------------------------------------------------------------------------
# quantil da distribuição t para 95% de área com df graus de liberdade no resíduo
qt(0.975, df=length(trat)-length(levels(trat)))
                                                                                                    #
#---------------------------------------------------------------------------------------------------
# número de experimentos sob H0 que rejeitaram H0, ocorrência do erro tipo I
apply(exper, 1, function(x) sum(x>2.01))/2000
apply(exper, 1, function(x) sum(x>3.17))/2000
                                                                                                    #
#---------------------------------------------------------------------------------------------------
# qual deveria ser a dms para assegurar o nível nominal de significância?
quantile(exper[2,], prob=0.95)
```

```
qtukey(0.95, length(levels(trat)), length(trat)-length(levels(trat)))/sqrt(2)
                                                                                    #
#-------------------------------------------------------------------------------------
# gráfico
layout(1)
plot(density(exper[1,]), lty=1)
lines(density(exper[2,]), lty=2)
abline(v=c(2.1,3.17), lty=1:2)
                                                                                    #
#-------------------------------------------------------------------------------------
.
```

7 Análise de experimentos de um fator em DBC

7.1 Entrada de dados

```
.
#-------------------------------------------------------------------------------------
# produção de madeira (m3/ha) em função de procedência de E. grandis e blocos
dbc <- expand.grid(proced=c("P1","P2","P3","P4","P5","P6","P7"),
                   bloco=c("I","II","III","IV"))
dbc
dbc$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(dbc)
                                                                                    #
#-------------------------------------------------------------------------------------
# gráficos
require(lattice)
xyplot(prod~proced, groups=bloco, data=dbc, type="b")
                                                                                    #
#-------------------------------------------------------------------------------------
.
```

7.2 Análise de variância

```
.
#-------------------------------------------------------------------------------------
# ajuste do modelo
m0 <- aov(prod~bloco+proced, data=dbc)
class(m0)       # classe do objeto, define os métodos que são aplicaveis
anova(m0)       # análise de variância
summary(m0)     # análise de variância
summary.lm(m0)  # quadro de estimativas dos efeitos/parâmetros (sob particular restrição)
                                                                                    #
#-------------------------------------------------------------------------------------
# checagem gráfica das presuposições da análise
par(mfrow=c(2,2))
plot(m0)
layout(1)
                                                                                    #
#-------------------------------------------------------------------------------------
# teste das pressuposições de normalidade de homocedasticidade
shapiro.test(dbc$prod)      # teste de normalidade nos dados ERRADO, contém efeitos
shapiro.test(residuals(m0)) # teste de normalidade nos resíduos CERTO, efeitos removidos
bartlett.test(residuals(m0)~dbc$proced)  # homogeneidade em função dos níveis de proced
bartlett.test(residuals(m0)~dbc$bloco)   # homogeneidade em função dos níveis de bloco
                                                                                    #
#-------------------------------------------------------------------------------------
.
```

7.3 Teste de médias

```
.
#-------------------------------------------------------------------------------------
# teste de Tukey
```

```
with(dbc, HSD.test(prod, proced,
                   DFerror=df.residual(m0),
                   MSerror=deviance(m0)/df.residual(m0)))
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# teste de Scott-Knott
sk <- SK(x=dbc, y=dbc$prod, model="prod~bloco+proced", which="proced")
summary(sk)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
.
```

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

```
.
#-----------------------------------------------------------------------------------------------
# usando a função com teste de Tukey (default)
with(dbc, rbd(proced, bloco, prod))
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# usando a função com teste de Student-Newman-Keuls's (default)
with(dbc, rbd(proced, bloco, prod, mcomp="snk"))
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# usando a função com teste de Skott-Knott
with(dbc, rbd(proced, bloco, prod, mcomp="sk"))
                                                                                                    #
#-----------------------------------------------------------------------------------------------
.
```

7.5 Observações perdidas

```
.
#-----------------------------------------------------------------------------------------------
# simulando observações perdidas aleatóriamente no conjunto dados
id <- sample(1:nrow(dbc), 3)
id
dbc2 <- dbc[-id,]
with(dbc2, tapply(prod, list(bloco, proced), length))
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# os efeitos dos fatores são aditivos, portando são estimaveis mas não ortogonais
m1 <- aov(prod~proced+bloco, data=dbc2)
summary(m1)
m1 <- aov(prod~bloco+proced, data=dbc2)
summary(m1)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# o que acontece com os estimadores amostrais das médias? são viesados pela não ortogonali
with(dbc, tapply(prod, proced, mean, na.rm=TRUE))
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# o que acontece com os estimadores de mínimos quadrados das médias?
ajus <- predict(m1, newdata=dbc)
with(dbc, tapply(ajus, proced, mean))
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# como comparar médias? Tukey usando a média harmônica do número de repetições?
# isso não remove o viés causado pela não ortogonalidade EVITAR
with(dbc2,
     HSD.test(prod, proced,
              DFerror=df.residual(m1),
              MSerror=deviance(m1)/df.residual(m1)
              ))
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# procedimentos diretos que precisam melhor descrição metodológica (usar com cuidado!)
TukeyHSD(m1)
layout(1)
plot(TukeyHSD(m1))
abline(v=0)
                                                                                                    #
```

```
#-------------------------------------------------------------------------------------------------
# usando a função glht()
summary(glht(m1, linfct=mcp(proced="Tukey")))
                                                                                                 #
#-------------------------------------------------------------------------------------------------
# contraste de médias populacionais marginais, montando os vetores de comparações
comp <- outer(levels(dbc$proced), levels(dbc$proced),
              function(x, y) paste(x, y, sep="-"))
comp
comp <- comp[upper.tri(comp)]
comp2 <- do.call(rbind, strsplit(comp, "-"))
comp2
                                                                                                 #
#-------------------------------------------------------------------------------------------------
# montando a matriz de contrastes
cX <- sapply(1:nrow(comp2),
             function(i){
               c.contr <- contrast(m1, type="average",
                                   list(proced=comp2[i,1], bloco=levels(dbc$bloco)),
                                   list(proced=comp2[i,2], bloco=levels(dbc$bloco)))
               c.contr$X
             })
cX
colnames(cX) <- comp
                                                                                                 #
#-------------------------------------------------------------------------------------------------
# fornecendo a matriz para a glht para manutenção do erro tipo I
comP <- glht(m1, linfct=t(cX))
summary(comP)          # mesmo resultado de summary(glht(m1, linfct=mcp(proced="Tukey")))
                                                                                                 #
#-------------------------------------------------------------------------------------------------
# as médias marginais populacionais
do.call(c, sapply(levels(dbc$proced),
                  function(i){
                    contrast(m1, type="average",
                             list(proced=i, bloco=levels(dbc$bloco)))[1]
                  }))
                                                                                                 #
#-------------------------------------------------------------------------------------------------
# obtendo as médias populacionais de maneira mais simples
m1 <- aov(prod~bloco+proced, data=dbc2,
          contrast=list(bloco=contr.sum, proced=contr.sum))
summary.lm(m1)
                                                                                                 #
#-------------------------------------------------------------------------------------------------
# cálculo matricial das médias ajustadas
Xproc <- cbind(1, m1$contrast$proced)  # matriz de contrastes para o fator proced
Iproc <- m1$assign                     # posições dos efeitos no vetor
Eproc <- coef(m1)[Iproc%in%c(0,2)]     # subvetor dos efeitos de proced
maju <- Xproc%*%Eproc                  # médias ajustadas
outer(c(maju), c(maju), function(x, y) abs(x-y))
                                                                                                 #
#-------------------------------------------------------------------------------------------------
.
```

8 Análise de experimento fatorial duplo em DIC

8.1 Análise de variância

```
.
#-------------------------------------------------------------------------------------------------
#vol <- read.table("http://www.leg.ufpr.br/~walmes/cursoR/volume.txt", header=TRUE)
vol <- read.table("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)
```

```
m0 <- aov(volu~gen*dose, data=vol)
summary(m0)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# verificando tipo das variáveis
class(vol$gen)
class(vol$dose)
vol$dose <- factor(vol$dose)
class(vol$dose)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# análise de variância com a especificação correta
m0 <- aov(volu~gen*dose, data=vol)
summary(m0)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# checagem
par(mfrow=c(2,2))
plot(m0)
layout(1)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# testes
shapiro.test(residuals(m0))
bartlett.test(residuals(m0)~vol$dose)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# precisa-se de tranformação para normalidade e homocedasticidade
require(MASS)
boxcox(m0)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# usando a tranformação indicada
m1 <- aov(volu^0.33~gen*dose, data=vol)
par(mfrow=c(2,2))
plot(m1)
layout(1)
shapiro.test(residuals(m1))
bartlett.test(residuals(m1)~vol$dose)
bartlett.test(residuals(m1)~vol$gen)
summary(m1)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# falta de normalidade é causa da mistura de 3 normais com variância distinta o que resulta
# numa distribuição mistura com mais curtose
                                                                                                    #
#-------------------------------------------------------------------------------------------------
.
```

8.2 Testes de médias

```
.
#-------------------------------------------------------------------------------------------------
# teste de Tukey para gen (com dados transformados)
require(agricolae)
Tu <- with(vol, HSD.test(volu^0.33, gen,
                         DFerror=df.residual(m1),
                         MSerror=deviance(m1)/df.residual(m1)
                         ))
Tu
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# aplicando a transformação inversa
Tu$means <- Tu$means^(1/0.33)
Tu
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# médias na escala original
with(vol, tapply(volu, gen, mean))
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# cuidados ao combinar funções estatísticas com funções não lineares
mean(sqrt(1:3))
sqrt(mean(1:3))
                                                                                                    #
#-------------------------------------------------------------------------------------------------
```

```
# teste de ScottKnott (com dados tranformados)
require(ScottKnott)
sk <- SK(x=vol, y=vol$volu^0.33, model="y~gen*dose", which="gen")
sk <- summary(sk)
sk$Means <- sk$Means^(1/0.33)
sk
cv.model(m1)
sk <- SK(x=vol, y=vol$volu^0.33, model="y~gen*dose", which="dose")
sk <- summary(sk)
sk$Means <- sk$Means^(1/0.33)
sk
                                                                                                    #
#-----------------------------------------------------------------------------------------------
.
```

8.3 Usando a função `ExpDes::fat2.crd()`

```
.
#-----------------------------------------------------------------------------------------------
# usando a variável original
with(vol, fat2.crd(gen, dose, volu, mcomp=c("sk","snk")))
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# usando a trasformação recomendada pela boxcox
with(vol, fat2.crd(gen, dose, volu^0.33, mcomp=c("sk","snk")))
                                                                                                    #
#-----------------------------------------------------------------------------------------------
.
```

9 Análise de fatorial duplo em DBC

9.1 Entrando com os dados

```
.
#-----------------------------------------------------------------------------------------------
#rend <- read.table("http://www.leg.ufpr.br/~walmes/cursoR/rendimento.txt", header=TRUE)
rend <- read.table("rendimento.txt", header=TRUE)
str(rend)
rend <- transform(rend, K=factor(K), A=factor(A), bloc=factor(bloc))
str(rend)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# análise gráfica
xyplot(rg~K|A, groups=bloc, data=rend, type="b", auto.key=TRUE)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
.
```

9.2 Análise de variância e desdobramento das somas de quadrados

```
.
#-----------------------------------------------------------------------------------------------
m0 <- aov(rg~bloc+A*K, data=rend)
summary(m0)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# checagem
par(mfrow=c(2,2))
plot(m0)
layout(1)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# desdobrando somas de quadrados para a variação de K dentro de A
m1 <- aov(rg~bloc+A/K, data=rend)
summary(m1)
coef(m1)
summary(m1, split=list("A:K"=list(
                          "A-37.5"=c(1,4,7,10),
```

```
                          "A-50.0"=c(2,5,8,11),
                          "A-62.5"=c(3,6,9,12)
                          )))
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# para facilitar encontrar as posições pode-se fazer a busca por expessões regulares
words <- c("O","R","é","um","programa","livre")
grep("r", words)
names(coef(m1))
names(coef(m1))[8:19]
grep("A37.5", names(coef(m1))[8:19])
grep("A50", names(coef(m1))[8:19])
grep("A62.5", names(coef(m1))[8:19])
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# usando as expressões regulares vamos desdobrar A dentro de K
m2 <- aov(rg~bloc+K/A, data=rend)
summary(m2)
names(coef(m2))
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# buscando pela expressão regular
grep("K0", names(coef(m2))[10:19])
grep("K30", names(coef(m2))[10:19])
grep("K60", names(coef(m2))[10:19])
grep("K120", names(coef(m2))[10:19])
grep("K180", names(coef(m2))[10:19])
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# decomposição das somas de quadrados
summary(m2, split=list("K:A"=list(
                          "K-0"=c(1,6),
                          "K-30"=c(2,7),
                          "K-60"=c(3,8),
                          "K-120"=c(4,9),
                          "K-180"=c(5,10)
                          )))
#-------------------------------------------------------------------------------------------------
# usando o pacote do Eric
require(ExpDes)
help(package="ExpDes")
help(fat2.rbd, help_type="html")
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# aplicando a função do Eric fat2.rbd
str(rend)
with(rend, fat2.rbd(A, K, bloc, rg, mcomp="sk", quali=c(TRUE, TRUE)))
                                                                                                    #
#-------------------------------------------------------------------------------------------------
.
```

9.3 Desdobramento da interação com testes de médias

```
.
#-------------------------------------------------------------------------------------------------
# desdobrando a interação em testes de médias para níveis de K fixando os níveis de A
with(subset(rend, A=="37.5"),
     HSD.test(rg, K, DFerror=df.residual(m0), MSerror=deviance(m0)/df.residual(m0)))
with(subset(rend, A=="50"),
     HSD.test(rg, K, DFerror=df.residual(m0), MSerror=deviance(m0)/df.residual(m0)))
with(subset(rend, A=="62.5"),
     HSD.test(rg, K, DFerror=df.residual(m0), MSerror=deviance(m0)/df.residual(m0)))
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# usando funções para fazer o desdobramento (lapply)
levels(rend$A)
lapply(levels(rend$A),
       function(a){
         with(subset(rend, A==a),
              HSD.test(rg, K,
                       DFerror=df.residual(m0),
                       MSerror=deviance(m0)/df.residual(m0)))
       })
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# fazendo o mesmo para o teste ScottKnott (a ordem A*K e K*A é importante!)
```

```
sk <- SK.nest(x=rend, y=rend$rg, model="y~bloc+A*K", which="A:K", fl2=1)
summary(sk)
sk <- SK.nest(x=rend, y=rend$rg, model="y~bloc+K*A", which="K:A", fl2=1)
summary(sk)
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
# fazer o teste de ScottKnott com um comando apenas (lapply)
length(levels(rend$A))
lapply(1:3,
       function(a){
         sk <- SK.nest(x=rend, y=rend$rg, model="y~bloc+K*A", which="K:A", fl2=a)
         summary(sk)
       })
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
lapply(1:5,
       function(a){
         sk <- SK.nest(x=rend, y=rend$rg, model="y~bloc+A*K", which="A:K", fl2=a)
         summary(sk)
       })
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
.
```

10 Análise de experimento fatorial com um tratamento adicional

```
.
#-----------------------------------------------------------------------------------------------------
# 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("fat-adi.txt", header=TRUE)
str(fa)
fa <- transform(fa, concentração=factor(concentração), bloc=factor(bloc))
str(fa)
fa
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
# análise de variância para os tratamentos (despreza estrutura fatorial-adicional)
m0 <- lm(media~bloc+trat, fa)
anova(m0)
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
# checagem
par(mfrow=c(2,2))
plot(m0)
layout(1)
shapiro.test(residuals(m0))
bartlett.test(residuals(m0)~fa$trat)
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
# as matrizes de contrastes envolvidas
contrasts(fa$trat)
contrasts(fa$origem)
contrasts(fa$concentração)
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
# para definir os contrastes a testemunha deve ser o último nível
levels(fa$origem)
levels(fa$concentração)
fa$concentração <- factor(fa$concentração, levels=c("25","50","75","0"))
contrasts(fa$concentração)
levels(fa$concentração)
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
# usar contrastes em que a testemunha contraste com os tratamentos (helmert)
contrasts(C(fa$origem, treatment))
contrasts(C(fa$origem, SAS))
contrasts(C(fa$origem, sum))
contrasts(C(fa$origem, poly))
contrasts(C(fa$origem, helmert))
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
# anova só da parte fatorial
anova(m0)
m1 <- aov(media~bloc+origem*concentração, data=subset(fa, trat!="TEST"))
```

```
summary(m1)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# anova com fornecimento dos contrates e "arrancando" a SQ do contraste com o adicional
# da SQ do fator origem
m2 <- aov(media~bloc+origem*concentração, data=fa,
          contrast=list(origem=contr.helmert, concentração=contr.helmert))
summary(m2)
summary(m2, expand.split=FALSE,
        split=list("origem"=list("fatorial"=c(1:2), "adicional"=3)))
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# teste de média da testemunha contra as origens na menor concentração
with(subset(fa, concentração %in% c("0","25")),
     HSD.test(media, origem,
              DFerror=df.residual(m2),
              MSerror=deviance(m2)/df.residual(m2)))
                                                                                                    #
#-------------------------------------------------------------------------------------------------
.
```

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

```
.
#-------------------------------------------------------------------------------------------------
require(ExpDes)
help(fat2.ad.rbd, help_type="html")
fa.fat <- fa[fa$trat!="TEST",]
fa.adi <- fa[fa$trat=="TEST",]
fat2.ad.rbd(fa.fat$origem, fa.fat$concentração, fa.fat$bloc, fa.fat$media, fa.adi$media)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
.
```

11 Análise de experimento com mistura de ingredientes

```
.
#-------------------------------------------------------------------------------------------------
# dados de diâmetro do caule em função da proporção de mistrura de K com Na na nutrição
mis <- read.table("mistura.txt", header=TRUE, sep="\t")
str(mis)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# exploração
xyplot(dc~trat, dat=mis, type=c("p","a"))
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# análise como um DIC, pouco informativo e não permite estrapolação
m0 <- lm(dc~trat, data=mis)
anova(m0)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# modelo de mistura de ingredientes
m1 <- lm(dc~K+K:Na+Naplus+BJ, data=mis)
anova(m1)
anova(m0, m1) # não há falta de ajuste
summary(m1)
model.matrix(m1)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# ajustar um novo modelo que permite melhor interpretação da solução
m2 <- lm(dc~-1+K+Na+K:Na+Naplus+BJ, data=mis)
summary(m2)
confint(m2)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# qual a mistura que confere o máximo?
Kmax <- (2*m2$coef["K:Na"]/(m2$coef["K:Na"]+m2$coef["K"]-m2$coef["Na"]))^(-1)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# comparar as médias dos tratamentos puros contra as testemunhas
```

```
require(contrast)
contrast(m2, list(K=0, Na=0, Naplus=1, BJ=0), list(K=0, Na=0, Naplus=0, BJ=1))
contrast(m2, list(K=0, Na=0, Naplus=1, BJ=-1))
contrast(m2, list(K=1, Na=0, Naplus=0, BJ=0), list(K=0, Na=1, Naplus=0, BJ=0))
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# predição
pred <- data.frame(K=seq(0,1,l=15), Na=0, Naplus=0, BJ=0); pred$Na <- 1-pred$K
pred <- rbind(pred, c(0,0,1,0), c(0,0,0,1))
pred$dc <- predict(m2, newdata=pred, interval="confidence")
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# gráfico
with(subset(pred, K!=0 | Na!=0),
     matplot(K, dc, type="l", col=c(1,2,2), lty=c(1,2,2),
             xlim=c(0,1.6), ylim=range(c(pred$dc,mis$dc)), xaxt="n"))
arrows(1.5, min(pred[16,"dc"]), 1.5, max(pred[16,"dc"]), code=3, angle=90)
arrows(1.25, min(pred[17,"dc"]), 1.25, max(pred[17,"dc"]), code=3, angle=90)
axis(1, at=seq(0,1,l=5))
axis(1, at=c(1.25,1.50), labels=c("BJ","Na plus"))
abline(v=Kmax, col="gray90", lty=3)
with(subset(mis, K!=0 | Na!=0), points(K, dc))
with(subset(mis, BJ==1), points(rep(1.25, length(dc)), dc))
with(subset(mis, Naplus==1), points(rep(1.50, length(dc)), dc))
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# fazendo o teste de médias
require(multcomp)
aux <- data.frame(trat=gl(4,2,la=names(coef(m2))[1:4]), y=rnorm(8))
contr <- glht(lm(y~-1+trat,aux), linfct=mcp(trat="Tukey"))
contr <- contr$linfct
colnames(contr) <- levels(aux$trat)
contr <- do.call(c, apply(contr, 1, function(x) list(x) ))
Xc <- lapply(contr, function(x){ contrast(m2, x) })
Xc <- do.call(rbind, lapply(Xc, function(x){ x$X }))
rownames(Xc) <- names(contr)
summary(glht(m2, linfct=Xc))
                                                                                                    #
#-------------------------------------------------------------------------------------------------
.
```

12 Análise de covariância

12.1 Análise de variância

```
.
#-------------------------------------------------------------------------------------------------
# dados
#ac <- read.table("http://www.leg.ufpr.br/~walmes/cursoR/ancova.txt", header=TRUE)
ac <- read.table("ancova.txt", header=TRUE)
str(ac)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# número de animais para cada combinação de níveis de sexo e energia
with(ac, tapply(peso28, list(sexo, energia), length))
                                                                                                    #
#-------------------------------------------------------------------------------------------------
#
xyplot(pi~id|sexo, groups=energia, data=ac, cex=2, pch=19, auto.key=TRUE)
xyplot(pi~id|energia, groups=sexo, data=ac, cex=2, pch=19, auto.key=TRUE)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# análise de variância (em experimentos não ortogonais a ordem dos termos é importante!)
m0 <- aov(peso28~sexo*energia, data=ac)        # modelo com os fatores categoricos
summary(m0)
m1 <- aov(peso28~pi+id+sexo*energia, data=ac)  # modelo com os categóricos e contínuos
summary(m1)
m1 <- aov(peso28~id+pi+sexo*energia, data=ac)
summary(m1)
anova(m0, m1) # testa o poder de explicação dos "blocos" contínuos
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# ajustes trocando a ordem dos termos
m2 <- aov(peso28~energia*sexo+pi+id, data=ac) # muda a ordem dos termos na fórmula
summary(m2)
```

```
m2 <- aov(peso28~sexo*energia+pi+id, data=ac) # muda a ordem dos termos na fórmula
summary(m2)
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
# 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 tratamentos
mean(ac$pi)   # média amostral de peso inicial dos animais do experimento (mais preciso)
mean(ac$id)   # média amostral de idade dos animais do experimento (mais preciso)
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
# para fazer os contrastes
require(contrast)
levels(ac$sexo)    # níveis do fator
levels(ac$energia) # níveis do fator
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
# ajuste do modelo com a função lm
m0 <- lm(peso28~pi+id+sexo*energia, data=ac)
anova(m0)
par(mfrow=c(2,2))
plot(m0)
layout(1)
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
.
```

12.2 Constraste entre níveis dos fatores

```
.
#-----------------------------------------------------------------------------------------------------
# femea vs macho castrado (observe que os erros padrões dos contrastes são diferentes)
contrast(m0, type="average",
         list(sexo="F", energia=c("baixo","medio","alto"), pi=92, id=138),
         list(sexo="MC", energia=levels(ac$energia), pi=92, id=138))
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
# femêa vs macho imunocatrado
contrast(m0, type="average",
         list(sexo="F", energia=levels(ac$energia), pi=92, id=138),
         list(sexo="MI", energia=levels(ac$energia), pi=92, id=138))
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
# macho castrado vs macho imunocastrado
contrast(m0, type="average",
         list(sexo="MI", energia=levels(ac$energia), pi=92, id=138),
         list(sexo="MC", energia=levels(ac$energia), pi=92, id=138))
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
# as médias marginais populacionais de sexo nas 3 rações
med <- sapply(levels(ac$sexo),
              function(s){
                contrast(m0, type="average",
                         list(sexo=s, energia=levels(ac$energia), pi=92, id=138))[1:7]
              })
str(med)
med
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
# gráfico de barras com IC para a média
require(gplots)
barplot2(unlist(med[1,]), ylim=c(120, 130), xpd=FALSE, plot.ci=TRUE,
         ci.l=unlist(med[3,]), ci.u=unlist(med[4,]),
         ylab="Peso aos 28 dias")
box()
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
# gráfico de barras com as médias e resultado da comparação
bp <- barplot(unlist(med[1,]), ylim=c(120, 130), xpd=FALSE, ylab="Peso aos 28 dias")
text(bp, unlist(med[1,]),
     label=paste(round(unlist(med[1,]), 2), c("b","b","a")), pos=3) # letras na mão
box()
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
.
```

13 Experimento fatorial com fatores qualitativos e quantitativos

```
.
#-------------------------------------------------------------------------------------------
# dados
# sorgo <- read.table("http://www.leg.ufpr.br/~walmes/cursoR/anovareg.txt", header=TRUE)
sorgo <- read.table("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
m0 <- aov(indice~bloco+cultivar*ordered(dose), data=sorgo)
summary(m0)
                                                                                          #
#-------------------------------------------------------------------------------------------
# checagem
par(mfrow=c(2,2))
plot(m0)
layout(1)
                                                                                          #
#-------------------------------------------------------------------------------------------
.
```

13.1 Desdobramento da interação

```
.
#-------------------------------------------------------------------------------------------
# desdobrando as somas de quadrados de doses dentro de cultivar
# dicas: forneça para 'by' o número de níveis de cultivar (3)
# forneça para 'length.out' os graus de liberdade de dose (6-1)
m1 <- aov(indice~bloco+cultivar/ordered(dose), data=sorgo)
summary(m1)
coef(m1)
summary(m1, split=list("cultivar:ordered(dose)"=list(
                         "Ag-1002"=seq(1, by=3, length.out=5),
                         "BR-300"=seq(2, by=3, length.out=5),
                         "Pioneer-B815"=seq(3, by=3, length.out=5)
                         )))
                                                                                          #
#-------------------------------------------------------------------------------------------
# desdobrando somas de quadrados de cultivar dentro das doses
# dicas: forneça para 'by' o número de níveis de dose (6)
# forneça para 'length.out' os graus de liberdade de cultivar (3-1)
m2 <- aov(indice~bloco+ordered(dose)/cultivar, data=sorgo)
coef(m2)
summary(m2, split=list("ordered(dose):cultivar"=list(
                         "N.0"=seq(1, by=6, length.out=2),
                         "N.60"=seq(2, by=6, length.out=2),
                         "N.120"=seq(3, by=6, length.out=2),
                         "N.180"=seq(4, by=6, length.out=2),
                         "N.240"=seq(5, by=6, length.out=2),
                         "N.300"=seq(6, by=6, length.out=2)
                         )))
                                                                                          #
#-------------------------------------------------------------------------------------------
# desdobrando efeitos dos graus polinômio dentro de dose dentro de cultivar
# lof é falta de ajuste (lack of fit)
summary(m1, split=list("cultivar:ordered(dose)"=list(
                         "Ag-1002.L"=1,
                         "Ag-1002.Q"=4,
                         "Ag-1002.C"=7,
                         "Ag-1002.lof"=c(10,13),
                         "BR-300.L"=2,
                         "BR-300.Q"=5,
                         "BR-300.C"=8,
                         "BR-300.lof"=c(11,14),
```

```
                          "Pioneer-B815.L"=3,
                          "Pioneer-B815.Q"=6,
                          "Pioneer-B815.C"=9,
                          "Pioneer-B815.lof"=c(12,15)
                          )))
                                                                                                    #
#-------------------------------------------------------------------------------------------------
.
```

13.2 Obtenção das equações de regressão e R^2

```
.
#-------------------------------------------------------------------------------------------------
# obter as equações de regressão e R^2 para os modelos linear, quadrático e cúbico
# dica: usar contraste tipo soma zero para blocos para se anularem na fórmula
# e remover o intercepto especificando o '-1', trocar a ordem dos termos no modelo
# linear (estimativas corretas mas erros padrões e p-valores precisam de correção)
m3 <- aov(indice~-1+cultivar/dose+bloco, data=sorgo,
          contrast=list(bloco=contr.sum))
summary.lm(m3)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# quadrático (estimativas corretas mas erros padrões e p-valores precisam de correção)
m4 <- aov(indice~-1+cultivar/(dose+I(dose^2))+bloco, data=sorgo,
          contrast=list(bloco=contr.sum))
summary.lm(m4)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# cúbico (estimativas corretas mas erros padrões e p-valores precisam de correção)
m5 <- aov(indice~-1+cultivar/(dose+I(dose^2)+I(dose^3))+bloco, data=sorgo,
          contrast=list(bloco=contr.sum))
summary.lm(m5)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# calcular os R^2
sapply(c(linear=1, quadrático=2, cúbico=3),
       function(degree){
         sapply(levels(sorgo$cultivar),
                function(i){
                  da <- with(subset(sorgo, cultivar==i),
                             aggregate(indice, list(dose=dose), mean))
                  summary(lm(x~poly(dose, degree, raw=TRUE), da))$r.squared
                })})
                                                                                                    #
#-------------------------------------------------------------------------------------------------
.
```

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

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

14 Fatorial com fatores quantitativos - parece superfície de resposta

14.1 Análise de variância e obtenção do modelo empírico

```
.
#-------------------------------------------------------------------------------------------------
# vamos usar os dados de rendimento de grãos de soja em função de K e A
#rend <- read.table("http://www.leg.ufpr.br/~walmes/cursoR/rendimento.txt", header=TRUE)
rend <- read.table("rendimento.txt", header=TRUE)
rend <- transform(rend, bloc=factor(bloc))
```

```
str(rend)
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
# ajustar um modelo quadrático completo
m0 <- lm(ts~bloc+poly(A, 2, raw=TRUE)*poly(K, 4, raw=TRUE), data=rend) # modelo saturado
m1 <- lm(ts~bloc+poly(A, 2, raw=TRUE)*poly(K, 2, raw=TRUE), data=rend) # modelo desejado
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
# testar a falta de ajuste e checagem dos resíduos
anova(m1, m0)
par(mfrow=c(2,2))
plot(m1)
layout(1)
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
# ajustando o modelo de segundo grau (dica, usar contr.sum para blocos)
levels(rend$bloc)
contrasts(rend$bloc) <- contr.sum(5)
contrasts(rend$bloc)
m2 <- lm(ts~bloc+(A+I(A^2))*(K+I(K^2)), data=rend)
anova(m2)
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
.
```

14.2 Gráfico do modelo final

```
.
#-----------------------------------------------------------------------------------------------------
# ajustar modelo menor e testar a falta de ajuste
m3 <- lm(ts~bloc+A+K+A:K+I(A^2)+I(K^2), data=rend)
anova(m3)
anova(m2, m3)
anova(m3, m0)
summary(m3)
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
# fazer o gráfico tridimensional dos valores preditos (dica, ajustar um modelo sem blocos
# apenas para fazer a predição, certificar-se de que as estimativa são as mesmas)
m4 <- lm(ts~A+K+A:K+I(A^2)+I(K^2), data=rend)
summary(m4)
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
# fazer a predição da resposta
p0 <- expand.grid(A=seq(35,65,l=20), K=seq(0,200,l=20))
p0$ts <- predict(m4, newdata=p0)
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
# usar a wireframe() da lattice (ver persp(), contour(), contourplot())
require(lattice)
wireframe(ts~A*K, data=p0, scales=list(arrows=FALSE))
levelplot(ts~A*K, data=p0, scales=list(arrows=FALSE), col.regions=heat.colors)
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
# outros gráficos
A <- seq(35,65,l=20); K <- seq(0,200,l=20)
p0 <- expand.grid(A=A, K=K)
p0$ts <- predict(m4, newdata=p0)
filled.contour(A, K, matrix(p0$ts,20,20))
contour(A, K, matrix(p0$ts,20,20))
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
# aprimoramento do gráfico, função traça as curvas de nível no gráfico
panel.3d.contour <- function(x, y, z, rot.mat, distance,
                             nlevels=20, zlim.scaled, col.contour=1, ...){
  panel.3dwire(x, y, z, rot.mat, distance, zlim.scaled=zlim.scaled, ...)
  z.grid <- seq(zlim.scaled[1], zlim.scaled[2], length=nlevels)
  clines <- contourLines(x, y, matrix(z, nrow=length(x), byrow=TRUE), nlevels=nlevels)
  for(ll in clines){
    n <- ltransform3dto3d(rbind(ll$x, ll$y, ll$level), rot.mat, distance)
    panel.lines(n[1,], n[2,], col=col.contour, lty=add.line$lty, lwd=add.line$lwd)
  }
}
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
# 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, 240), col="gray30", col.contour="gray30",
          panel.3d.wireframe="panel.3d.contour",
          col.regions=colr(100),  drape=TRUE)
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
# escolher o ângulo de observação e o esquema de cores
require(manipulate)
manipulate({
            ## faz o gráfico tridimensional
            colr <- colorRampPalette(c(c1, c2, c3), space="rgb")
            wireframe(ts~A*K, data=p0, scales=list(arrows=FALSE),
                      zlim=c(100, 240), col="gray30", col.contour="gray30",
                      panel.3d.wireframe="panel.3d.contour",
                      col.regions=colr(100),  drape=TRUE,
                      screen=list(z=z.angle, x=x.angle)
                      )},
            ## controla o valor dos angulos e das cores
            z.angle=slider(0, 360, step=10, initial=40),
            x.angle=slider(-180, 0, step=5, initial=-60),
            c1=picker("red","yellow","orange","green","blue","pink","violet"),
            c2=picker("red","yellow","orange","green","blue","pink","violet"),
            c3=picker("red","yellow","orange","green","blue","pink","violet")
            )
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
.
```

15 Análise de experimentos em parcela subdividida

15.1 Análise de variância

```
.
#-----------------------------------------------------------------------------------------------------
# dados
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)
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
# 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
summary(m0)
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
# checagem dos resíduos
class(m0)
m1 <- lm(alt~BL+AD*(BL+ES), data=ps) # mesmo modelo mas Erro A agora e efeito fixo
anova(m1)
par(mfrow=c(2,2))
plot(m1)
shapiro.test(residuals(m1))
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
.
```

15.2 Teste de médias

```
.
#-----------------------------------------------------------------------------------------------------
# dedobrar ES em cada nível de AD
require(agricolae)
df.residual(m0) # não extraem
deviance(m0)    # não extraem
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
# temos que retirar os valores de GL e QM da anova
```

```
str(summary(m0))
str(summary(m0)[[1]])
str(summary(m0)[[1]][[1]])
summary(m0)[[1]][[1]]
summary(m0)[[2]][[1]]
glP <- summary(m0)[[1]][[1]][3,1]
qmP <- summary(m0)[[1]][[1]][3,3]
glS <- summary(m0)[[2]][[1]][3,1]
qmS <- summary(m0)[[2]][[1]][3,3]
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# teste de Tukey
lapply(levels(ps$AD),
       function(a){
         with(subset(ps, AD==a),
              HSD.test(alt, ES, DFerror=glS, MSerror=qmS))
       })
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# teste de ScottKnott
require(ScottKnott)
lapply(1:3,
       function(a){
         sk <- SK.nest(x=ps, y=ps$alt, model="y~BL+ES*AD+Error(BL:AD)",
                       which="ES:AD", fl2=a, error="Within")
         summary(sk)
       })
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# 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])
  ngl <- (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])
  return(c(qmr=qmr, ngl=ngl))
}
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# 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
lapply(levels(ps$ES),
       function(a){
         with(subset(ps, ES==a),
              HSD.test(alt, AD, DFerror=aux["ngl"], MSerror=aux["qmr"]))
       })
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# desdobrar com o teste de ScottKnott
lapply(1:2,
       function(a){
         sk <- SK.nest(x=ps, y=ps$alt, model="y~BL+AD*ES+Error(BL:AD)",
                       which="AD:ES", fl2=a, error="BL:AD")
         summary(sk)
       })
                                                                                                    #
#-------------------------------------------------------------------------------------------------
.
```

15.3 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=c("tukey","tukey")))
                                                                                                    #
#-------------------------------------------------------------------------------------------------
.
```

16 Experimentos em parcelas subsubdivididas

16.1 Análise de variância

```
.
#-------------------------------------------------------------------------------------------
# dados
#pss <- read.table("http://www.leg.ufpr.br/~walmes/cursoR/pss.txt", header=TRUE)
pss <- read.table("pss.txt", header=TRUE)
pss <- transform(pss, dorg=factor(dorg), dnpk=factor(dnpk), bloco=factor(bloco))
str(pss)
                                                                                          #
#-------------------------------------------------------------------------------------------
# análise de variância, ErroA=bloco:parcela, ErroB=bloco:parcela:subparcela
m0 <- aov(AF~bloco+fonte*dorg*dnpk+Error(bloco:fonte/dorg), data=pss)
summary(m0)
                                                                                          #
#-------------------------------------------------------------------------------------------
# checagem não é possível por padrão
class(m0)
m1 <- aov(AF~bloco/fonte/dorg+fonte*dorg*dnpk, data=pss)
anova(m1)
par(mfrow=c(2,2))
plot(m1)
layout(1)
                                                                                          #
#-------------------------------------------------------------------------------------------
# aplicar uma transformação aos dados
require(MASS)
boxcox(m1) # ponta a transformação log
m2 <- aov(log(AF)~bloco/fonte/dorg+fonte*dorg*dnpk, data=pss)
par(mfrow=c(2,2))
plot(m2)
layout(1)
                                                                                          #
#-------------------------------------------------------------------------------------------
# análise de variância com dados tranformados
m0 <- aov(log(AF)~bloco+fonte*dorg*dnpk+Error(bloco:fonte/dorg), data=pss)
summary(m0)
                                                                                          #
#-------------------------------------------------------------------------------------------
.
```

16.2 Testes de médias

```
.
#-------------------------------------------------------------------------------------------
# desdobrar níveis da subsub dentro de níveis da sub com parcela (usa erro C)
require(agricolae)
gl3 <- summary(m0)[[3]][[1]][5,1]
qm3 <- summary(m0)[[3]][[1]][5,3]
                                                                                          #
#-------------------------------------------------------------------------------------------
lapply(levels(pss$fonte),
       function(f){
         lapply(levels(pss$dorg),
                function(o){
                  invisible(capture.output(
                  tes <- with(subset(pss, fonte==f & dorg==o),
                              HSD.test(log(AF), dnpk, DFerror=gl3, MSerror=qm3))))
                  tes$means <- exp(tes$means)
                  tes
                })
       })
                                                                                          #
#-------------------------------------------------------------------------------------------
# desdobrar níveis de dorg em níveis de fonte com dnpk (usa variância combinada)
gl1 <- summary(m0)[[1]][[1]][3,1]
qm1 <- summary(m0)[[1]][[1]][3,3]
gl2 <- summary(m0)[[2]][[1]][3,1]
qm2 <- summary(m0)[[2]][[1]][3,3]
vcBemCA <- satter(c(qm2, gl2, 4),
                  c(qm3, gl3, 5))
vcBemCA
```

```
                                                                                                                    #
#-------------------------------------------------------------------------------------------------------------------
lapply(levels(pss$fonte),
       function(f){
         lapply(levels(pss$dnpk),
                function(npk){
                  invisible(capture.output(
                  tes <- with(subset(pss, fonte==f & dnpk==npk),
                              HSD.test(log(AF), dorg,
                                       DFerror=vcBemCA["ngl"], MSerror=vcBemCA["qmr"]))))
                  tes$means <- exp(tes$means)
                  tes
                })
       })
                                                                                                                    #
#-------------------------------------------------------------------------------------------------------------------
# desdobrar níveis de fonte dentro de níveis de dorg com dnpk
vcAemBC <- satter(c(qm1, gl1, 3),
                  c(qm2, gl2, 4),
                  c(qm3, gl3, 5))
vcAemBC
                                                                                                                    #
#-------------------------------------------------------------------------------------------------------------------
lapply(levels(pss$dorg),
       function(o){
         lapply(levels(pss$dnpk),
                function(npk){
                  invisible(capture.output(
                  tes <- with(subset(pss, dorg==o & dnpk==npk),
                              HSD.test(log(AF), fonte,
                                       DFerror=vcAemBC["ngl"], MSerror=vcAemBC["qmr"]))))
                  tes$means <- exp(tes$means)
                  tes
                })
       })
                                                                                                                    #
#-------------------------------------------------------------------------------------------------------------------
# usando o teste de ScottKnott para dnpk em fonte com dorg
require(ScottKnott)
tes <- SK.nest(x=pss, y=log(pss$AF),
               model="y~bloco+dnpk*fonte*dorg+Error(bloco:fonte/dorg)",
               which="dnpk:fonte:dorg", error="Within", fl2=1, fl3=1)
summary(tes)
                                                                                                                    #
#-------------------------------------------------------------------------------------------------------------------
lapply(1:3,
       function(f){
         lapply(1:4,
                function(o){
                  tes <- SK.nest(x=pss, y=log(pss$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$Means <- exp(tes$Means)
                  tes
                })
       })
                                                                                                                    #
#-------------------------------------------------------------------------------------------------------------------
# desdobrar dorg em fonte com dnpk
tes <- SK.nest(x=pss, y=log(pss$AF),
               model="y~bloco+dorg*fonte*dnpk+Error(bloco:fonte/dorg)",
               which="dorg:fonte:dnpk", error="bloco:fonte:dorg", fl2=1, fl3=1)
summary(tes)
                                                                                                                    #
#-------------------------------------------------------------------------------------------------------------------
lapply(1:3,
       function(f){
         lapply(1:5,
                function(npk){
                  tes <- SK.nest(x=pss, y=log(pss$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$Means <- exp(tes$Means)
                  tes
                })
```

```
       })
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# desdobrar fonte em dorg com dnpk
tes <- SK.nest(x=pss, y=log(pss$AF),
               model="y~bloco+fonte*dorg*dnpk+Error(bloco:fonte/dorg)",
               which="fonte:dorg:dnpk", error="bloco:fonte", fl2=1, fl3=1)
summary(tes)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
lapply(1:4,
       function(o){
         lapply(1:5,
                function(npk){
                  tes <- SK.nest(x=pss, y=log(pss$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$Means <- exp(tes$Means)
                  tes
                })
       })
                                                                                                    #
#-----------------------------------------------------------------------------------------------
.
```

17 Procedimentos para análise de dados de proporção

17.1 Latência em pêssego

```
.
#-----------------------------------------------------------------------------------------------
# dados de latência em pessêgo
pes <- read.table("latencia.txt", header=TRUE, sep="\t")
str(pes)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# assumindo normalidade e ajustando o modelo total
m0 <- lm(lat48~., data=pes)
summary(m0)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# atualizando com as variáveis significativas
m1 <- lm(lat48~(ms0+b0)^2, data=pes)
summary(m1)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# remove a interação
m1 <- lm(lat48~ms0+b0, data=pes)
summary(m1)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# 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)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# 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))
                                                                                                    #
#-----------------------------------------------------------------------------------------------
.
```

17.2 Número de sementes viáveis

```
.
#-----------------------------------------------------------------------------------------------
# dados de número de sementes viáveis de soja
rend <- read.table("rendimento.txt", header=TRUE)
rend <- transform(rend, k=factor(K), a=factor(A), bloc=factor(bloc))
str(rend)
                                                                                                    #
#-----------------------------------------------------------------------------------------------
# ajuste modelo de caselas aos dados assumindo distribuição binomial (link=logit)
g0 <- glm(cbind(nv, nvi)~bloc+k*a, data=rend, 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=rend, 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(rend,
             expand.grid(A=seq(min(A),max(A),l=20),
                         K=seq(min(K),max(K),l=20),
                         bloc="1"))
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)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
.
```

18 Análise de dados de contagem

```
.
#-------------------------------------------------------------------------------------------------
# dados de mortalidade de aves em galpões com diferentes sistemas de arrefecimento
mor <- read.table("mortes.txt", header=TRUE, sep="\t")
str(mor)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# 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)
anova(g0, test="Chisq")
g0 <- update(g0, family=quasipoisson, contrast=list(galpao=contr.sum))
anova(g0, test="F")
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# check
par(mfrow=c(2,2))
plot(g0)
layout(1)
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# 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
                                                                                                    #
#-------------------------------------------------------------------------------------------------
# artifício 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=expression(Idade~das~aves~(dias)),
       ylab=expression(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)
       })
                                                                                                    #
#-------------------------------------------------------------------------------------------------
.
```

19 Recursos gráficos

19.1 Gráficos do pacote *graphics*

```
.
#-------------------------------------------------------------------------------------------
# conhecendo os recursos gráficos
layout(1)
demo(graphics)
                                                                                          #
#-------------------------------------------------------------------------------------------
# carregando dados disponível no R
data(anscombe)
str(anscombe)
                                                                                          #
#-------------------------------------------------------------------------------------------
# gráficos de dispersão e identificação de pontos
plot(y1~x1, data=anscombe,
     col="red", pch=3, type="p", cex=1.2)
with(anscombe, identify(x1, y1))
                                                                                          #
#-------------------------------------------------------------------------------------------
# gráficos de funções e inserção de legenda
curve((2*pi*1)^-0.5*exp(-0.5*(x-0)^2/1), from=-3, to=3)
curve(dnorm(x, 0.5, 1.1), col="green", lty=2, add=TRUE)
legend(x=-3, y=0.4, legend=c("N(0,1)","N(0.5,1.1)"),
       col=c(1,3), lty=c(1,2))
                                                                                          #
#-------------------------------------------------------------------------------------------
# visualizando a distribuição dos dados
hist(anscombe$y1)
with(anscombe, plot(density(y1)))
qqnorm(anscombe$y1); qqline(anscombe$y1)
with(anscombe, plot(ecdf(y1)))
                                                                                          #
#-------------------------------------------------------------------------------------------
# boxplot e adição de retas
x <- matrix(rep(1:10, 10), ncol=10)
x[10,] <- 10:19
boxplot(x)
fivenum(1:10)
abline(h=fivenum(1:10), col="orange", lty=5)
abline(h=8+(8-3)*1.5, col="cyan", lty=4)
abline(v=6.5)
abline(a=9, b=1, col="red")
                                                                                          #
#-------------------------------------------------------------------------------------------
# combinando recursos gráficos (1)
hist(anscombe$y1, freq=FALSE)
lines(density(anscombe$y1))
mean(anscombe); sd(anscombe)
curve(dnorm(x, 7.5, 2.03), col="green", lty=2, add=TRUE)
                                                                                          #
#-------------------------------------------------------------------------------------------
# combinando recursos gráficos (2)
plot(y1~x1, data=anscombe)
m0 <- lm(y1~x1, data=anscombe)
abline(m0, col="red")
with(anscombe, segments(x1, y1, x1, fitted(m0)))
with(anscombe, points(x1, fitted(m0), pch=3))
                                                                                          #
#-------------------------------------------------------------------------------------------
# combinando recursos gráficos (3)
plot(y1~x1, data=anscombe,
     xlab="Valores de x", ylab="Valores de y")
new.x1 <- seq(4, 14, length=20)
p0 <- predict(m0, newdata=data.frame(x1=new.x1),
              interval="confidence")
str(p0)
lines(new.x1, p0[,"fit"], lwd=2)
lines(new.x1, p0[,"lwr"], lty=2)
lines(new.x1, p0[,"upr"], lty=2)
coef(m0)
legend("topleft", legend="y=3+0.5*x",
       col=1, lwd=2, bty="n")
                                                                                          #
#-------------------------------------------------------------------------------------------
# gráficos de barras com texto
```

```
mads <- apply(anscombe[,5:8], 2, mad)
tt <- barplot(mads, ylim=c(0,2.5))
text(tt, mads, label=mads, pos=3)
title("Desvios absolutos da mediana")
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
# gráficos de setores (pizza)
str(HairEyeColor)
x <- apply(HairEyeColor, 2, sum)
x <- apply(HairEyeColor, 1, sum)
pie(x)
pie(mads, main="DAM")
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
# interpretando o qqplot
n <- 1000
x <- rnorm(n, 2, 1.2) #x <- rbeta(n, 2, 1.2) #x <- rgamma(n, 2, 1.2)
qnorm(x); qqline(x, col="red")
op <- par(fig=c(.02,.5,.5,.98), new=TRUE)
hist(x, freq=FALSE, axes=FALSE, main="", xlab="", ylab="")
lines(density(x), col="red", lwd=2)
box()
par(op)
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
# gráficos de contornos de níveis
str(volcano)
x <- 10*(1:nrow(volcano))
y <- 10*(1:ncol(volcano))
image(x, y, volcano)
contour(x, y, volcano, add=TRUE)
image(matrix(rnorm(100),10,10))
contour(matrix(rnorm(100),10,10))
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
# funções paramétricas de representação 3D
x <- seq(-10, 10, length=50)
y <- x
z <- outer(x, y, function(x,y) 0.5*sin(x)+0.8*sin(y))
filled.contour(x, y, z)
persp(x, y, z, theta=30, phi=30, expand=0.5, col="lightgreen")
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
# matriz de gráficos de dispersão
pairs(~mpg+disp+drat+wt, data=mtcars,
      main="Matriz gráfica de dispersão")
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
.
```

19.2 Gráficos do pacote *lattice*

```
.
#-----------------------------------------------------------------------------------------------------
# carregando a biblioteca gráfica (vem com o R por padrão)
require(lattice)
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
# distribuição
histogram(~height|voice.part, data=singer)
densityplot(~height|voice.part, data=singer)
qqmath(~height|voice.part, data=singer)
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
# dispersão
xyplot(Petal.Length~Sepal.Length|Species, data=iris,
type=c("p","smooth"))
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
# box and whiskers (caixa e bigode)
bwplot(depth~factor(mag)|cut(stations,2), data=quakes, pch="|")
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
# representação 3D
wireframe(volcano, shade=TRUE)
g <- expand.grid(x=1:10, y=5:15, gr=1:2)
```

```
g$z <- log((g$x^g$g+g$y^2)*g$gr)
wireframe(z~x*y, data=g, groups=gr)
cloud(Sepal.Length~Petal.Length*Petal.Width|Species, data=iris)
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
# matriz de gráficos de dispersão
splom(~iris[1:4], groups=Species, data=iris)
                                                                                                    #
#-----------------------------------------------------------------------------------------------------
.
```