Curso de Extensão
Planejamento de Experimentos

Adilson dos Anjos
&
Paulo Justiniano Ribeiro Junior

Última atualização: 11 de abril de 2003

1 Uma primeira sessão com o R

Vamos começar "experimentando o R", para ter uma idéia de seus recursos e a forma de trabalhar. Para isto vamos rodar e estudar os comandos abaixo e seus resultados para nos familiarizar com o programa. Nas sessões seguintes iremos ver com mais detalhes o uso do programa R. Siga os seguintes passos.

1. inicie o R em seu computador.
2. voce verá uma janela de comandos com o símbolo >.
   Este é o *prompt* do R indicando que o programa está pronto para receber comandos.
3. a seguir digite (ou "recorte e cole") os comandos mostrados abaixo.
   No restante deste texto vamos seguir as seguintes convenções.
   - comandos do R são sempre mostrados em fontes do tipo `typewriter como esta`,
   - linhas iniciadas pelo símbolo # são comentários e são ignoradas pelo R.

```
# gerando dois vetores de coordenadas x e y de números pseudo-aleatórios
x <- rnorm(50)
y <- rnorm(x)

# colocando os pontos em um gráfico.
# Note que a janela gráfica se abrirá automaticamente
plot(x, y)

# medidas descritivas ...
mean(x)
median(x)
var(x)
sd(x)

quantile(x)
quantile(x, probs = c(0.10, 0.50, 0.90))
quantile(x, probs = seq(0, 1, by=0.1))
```

```
summary(x)
table(cut(x, breaks=seq(-3, 3, by=1)))

# ... e alguns gráficos
hist(x)
boxcox(x)
stem(x)

# verificando os objetos existentes na área de trabalho
ls()

# removendo objetos que não são mais necessários
rm(x, y)

# criando um vetor com uma sequência de números de 1 a 20
x <- 1:20

# um vetor de pesos com os desvios padrões de cada observação
w <- 1 + sqrt(x)/2

# montando um ‘data-frame’ de 2 colunas, x e y, e inspecionando o objeto
dummy <- data.frame(x=x, y= x + rnorm(x)*w)
dummy

# Ajustando uma regressão linear simples de y em x e examinando os resultados
fm <- lm(y ~ x, data=dummy)
summary(fm)

# como nós sabemos os pesos podemos fazer uma regressão ponderada
fm1 <- lm(y ~ x, data=dummy, weight=1/w^2)
summary(fm1)

# tornando visíveis as colunas do data-frame
attach(dummy)

# fazendo uma regressão local não-paramétrica
lrf <- lowess(x, y)

# plotando os pontos
plot(x, y)

# adicionando a linha de regressão local ...
lines(x, lrf$y)

# ... e a linha de regressão verdadeira (intercepto 0 e inclinação 1)
abline(0, 1, lty=3)

# a linha da regressão sem ponderação
abline(coef(fm))
```

```
# e a linha de regressão ponderada.
abline(coef(fm1), col = "red")

# removendo o objeto do caminho de procura
detach()

# O gráfico diagnóstico padrão para checar homocedasticidade.
plot(fitted(fm), resid(fm),
     xlab="Fitted values", ylab="Residuals",
     main="Residuals vs Fitted")

# gráficos de escores normais para checar
# assimetria, curtose e outliers (não muito útil aqui)
qqnorm(resid(fm), main="Residuals Rankit Plot")

# ‘‘limpando’’ novamente (apagando objetos)
rm(fm, fm1, lrf, x, dummy)
```

Vamos agora ver alguns gráficos gerados pelas funções `contour` e `image`.

```
# x é um vetor de 50 valores igualmente espaçados  no intervalo [-pi\, pi]. y idem.
x <- seq(-pi, pi, len=50)
y <- x

# f é uma matrix quadrada com linhas e colunas indexadas por x e y respectivamente
# com os valores da função cos(y)/(1 + x^2).
f <- outer(x, y, function(x, y) cos(y)/(1 + x^2))

# gravando parâmetros gráficos e definindo a região gráfica como quadrada
oldpar <- par(no.readonly = TRUE)
par(pty="s")

# fazendo um mapa de contorno de f e depois adicionando
# mais linhas para visualizar mais detalhes
contour(x, y, f)
contour(x, y, f, nlevels=15, add=TRUE)

#  fa é a ‘‘parte assimétrica’’. (t() é transposição).
fa <- (f-t(f))/2

# fazendo um mapa de contorno
contour(x, y, fa, nlevels=15)

#    ...  e restaurando parâmetros gráficos iniciais
par(oldpar)

# Fazendo um gráfico de imagem
image(x, y, f)
image(x, y, fa)
```

```
# e apagando objetos novamente antes de prosseguir.
objects(); rm(x, y, f, fa)
```

Para encerrar esta sessão vajamos mais algumas funcionalidades do R.

```
# O R pode fazer operação com complexos
th <- seq(-pi, pi, len=100)
#  1i denota o número complexo  i.
z <- exp(1i*th)

# plotando complexos significa parte imaginária versus real
# Isto deve ser um círculo:
par(pty="s")
plot(z, type="l")

# Suponha que desejamos amostrar pontos dentro do círculo de raio unitário.
# uma forma simples de fazer isto é tomar números complexos com parte
# real e imaginária padrão
w <- rnorm(100) + rnorm(100)*1i

#  ...  e para mapear qualquer externo ao círculo no seu recíproco:
w <- ifelse(Mod(w) > 1, 1/w, w)

# todos os pontos estão dentro do círculo unitário, mas a distribuição
# não é uniforme.
plot(w, xlim=c(-1,1), ylim=c(-1,1), pch="+",xlab="x", ylab="y")
lines(z)

# este segundo método usa a distribuição uniforme.
# os pontos devem estar melhor distribuídos sobre o círculo
w <- sqrt(runif(100))*exp(2*pi*runif(100)*1i)
plot(w, xlim=c(-1,1), ylim=c(-1,1), pch="+", xlab="x", ylab="y")
lines(z)

# apagando os objetos
rm(th, w, z)

# saindo do R
q()
```

2 Recursos do R

O projeto R

O programa R é gratuito e de código aberto que propicia excelente ambiente para análises estatísticas e com recursos gráficos de alta qualidade. Detalhes sobre o projeto, colaboradores, documentação e diversas outras informações podem ser encontradas na página oficial do projeto em:
http://www.r-project.org.

O programa pode ser copiado livremente pela internet. Há um espelho (*mirror*) brasileiro da área de *downloads* do programa no *Departamento de Estatística da UFPR*:
http://www.est.ufpr.br/R
ou então via FTP:
ftp://est.ufpr.br/R

Será feita uma apresentação da página do R durante o curso onde os principais recursos serão comentados assim como as idéias principais que governam o projeto e suas direções futuras.

Demos

O R vem com algumas demonstrações (*demos*) de seus recursos "embutidas" no programa. Para listar as demos disponíveis digite na linha de comando:

```
demo()
```

E para rodar uma delas basta colocar o nome da escolhida entre os parênteses. Por exemplo, vamos rodar a de recursos gráficos. Note que os comandos vão aparecer na janela de comandos e os gráficos serão automaticamente produzidos na janela gráfica. Voce vai ter que teclar ENTER para ver o próximo gráfico.

- inicie o programaR
- no "prompt" do programa digite:

  ```
  demo(graphics)
  ```

- Voce vai ver a seguinte mensagem na tela:

  ```
      demo(graphics)
      ---- ~~~~~~~~

      Type  <Return>   to start :
  ```

- pressione a tecla ENTER
- a "demo" vai ser iniciada e uma tela gráfica irá se abrir. Na tela de comandos serão mostrados comandos que serão utilizados para gerar um gráfico seguidos da mensagem:

  ```
      Hit <Return> to see next plot:
  ```

- inspecione os comandos e depois pressione novamente a tecla ENTER.
  Agora voce pode visualizar na janela gráfica o gráfico produzido pelos comandos mostrados anteriormente. Inspecione o gráfico cuidadosamente verificando os recursos utilizados (título, legendas dos eixos, tipos de pontos, cores dos pontos, linhas, cores de fundo, etc).

- agora na tela de comandos apareceram novos comandos para produzir um novo gráfico e a mensagem:

  ```
  Hit <Return> to see next plot:
  ```

- inspecione os novos comandos e depois pressione novamente a tecla ENTER.
  Um novo gráfico surgirá ilustrando outros recursos do programa.
  Prossiga inspecionando os gráficos e comandos e pressionando ENTER até terminar a "demo".
  Experimente outras demos como `demo(pers)` e `demo(image)`, por exemplo.

Um tutorial sobre o R

Além dos recursos ilustrados neste curso há também um *Tutorial de Introdução ao* R disponível em http://www.est.ufpr.br/Rtutorial.

Outros materiais podem ser encontrados na página do projeto.

RWeb

Este é um mecanismo que permite rodar o R pela web, sem que voce precise ter o R instalado no seu computador. Para usá-lo basta estar conectado na internet.

Para acessar o **RWeb** vá até a página do Re no menu à esquerda da página siga os links: R GUIs ... R Web
Nesta página selecione primeiro o link **R Web** e examine seu conteúdo.

Há ainda uma diversidade de recursos disponíveis na página do projeto. Os participantes do curso são estimulados a explorar detalhadamente ao final do curso os outros recursos da página.

Cartão de referência

Conforme voce viu na Sessão 1 para utilizar o R é necessário conhecer e digitar comandos. Isto pode trazer alguma dificuldade no inicio até que o usuário de familiarize com os comandos mais comuns. Uma boa forma de aprender e memorizar os comandos básicos é utilizar o Cartão de Referência que contém os comandos mais frequentemente utilizados.

3 Experimentos com delineamento inteiramente casualizados

Nesta sessão iremos usar o R para analisar um experimento em delineamento inteiramente casualizado.
A seguir são apresentados os comandos para a análise do experimento. Inspecione-os cuidadosamente e discuta os resultados e a manipulação do programa R.

Primeiro lemos o arquivo de dados que deve ter sido copiado para o seu diretório de trabalho.

```
ex01 <- read.table("exemplo01.txt", head=T)
ex01
```

Caso o arquivo esteja em outro diretório deve-se colocar o caminho completo deste diretório no argumento de `read.table` acima.
A seguir vamos inspecionar o objeto que armazena os dados e suas componentes.

```
is.data.frame(ex01)
names(ex01)

ex01$resp
ex01$trat

is.factor(ex01$trat)
is.numeric(ex01$resp)
```

Portando concluímos que o objeto é um *data-frame* com duas variáveis, sendo uma delas um fator (a variável *trat*) e a outra uma variável numérica.

Vamos agora fazer uma rápida análise descritiva:

```
summary(ex01)
tapply(ex01$resp, ex01$trat, mean)
```

Há um mecanismo no R de "anexar"objetos ao caminho de procura que permite economizar um pouco de digitação. Veja os comandos abaixo e compara com o comando anterior.

```
search()

attach(ex01)
search()

tapply(resp, trat, mean)
```

Interessante não? Quando "anexamos"um objeto do tipo *list* ou *data.frame* no caminho de procura com o comando `attach()` fazemos com que os componentes deste objeto se tornem imediatamente disponíveis e portanto podemos, por exemplo, digitar somente `trat` ao invés de ex01$trat.

Vamos prosseguir com a análise exploratória, obtendo algumas medidas e gráficos.

```
ex01.m <- tapply(resp, trat, mean)
ex01.m
```

```
ex01.v <-  tapply(resp, trat, var)
ex01.v

plot(ex01)
points(ex01.m, pch="x", col=2, cex=1.5)

boxplot(resp ~ trat)
```

Além dos gráficos acima podemos também verificar a homogeneidade de variâncias com o Teste de Bartlett.

```
bartlett.test(resp, trat)
```

Agora vamos fazer a análise de variância. Vamos "desanexar" o objeto com os dados (embora isto não seja obrigatório).

```
detach(ex01)

ex01.av <- aov(resp ~ trat, data = ex01)
ex01.av

summary(ex01.av)
anova(ex01.av)
```

Portanto o objeto `ex01.av` guarda os resultados da análise. Vamos inspecionar este objeto mais cuidadosamente e fazer também uma análise dos resultados e resíduos:

```
names(ex01.av)
ex01.av$coef

ex01.av$res
residuals(ex01.av)

plot(ex01.av) # pressione a tecla enter para mudar o gráfico

par(mfrow=c(2,2))
plot(ex01.av)
par(mfrow=c(1,1))

plot(ex01.av$fit, ex01.av$res, xlab="valores ajustados", ylab="resíduos")
title("resíduos vs Preditos")

names(anova(ex01.av))
s2 <- anova(ex01.av)$Mean[2]   # estimativa da variância

res <- ex01.av$res         # extraindo resíduos
respad <- (res/sqrt(s2))  # resíduos padronizados
boxplot(respad)
title("Resíduos Padronizados" )

hist(respad, main=NULL)
```

```
title("Histograma dos resíduos padronizados")

stem(respad)
qqnorm(res,ylab="Residuos", main=NULL)
qqline(res)
title("Grafico Normal de Probabilidade dos Resíduos")

shapiro.test(res)
```

E agora um teste Tukey de comparação múltipla

```
ex01.tu <- TukeyHSD(ex01.av)
plot(ex01.tu)
```

4 Experimentos com delineamento em blocos ao acaso

Vamos agora analisar o experimento em blocos ao acaso descrito na apostila do curso. Os dados estão reproduuzidos na tabela abaixo.

Tabela 1: Conteúdo de óleo de *S. linicola*, em percentagem, em vários estágios de crescimento (Steel & Torrie, 1980, p.199).

Estágios	Blocos			
	I	II	III	IV
Estágio 1	4,4	5,9	6,0	4,1
Estágio 2	3,3	1,9	4,9	7,1
Estágio 3	4,4	4,0	4,5	3,1
Estágio 4	6,8	6,6	7,0	6,4
Estágio 5	6,3	4,9	5,9	7,1
Estágio 6	6,4	7,3	7,7	6,7

Inicialmente vamos entrar com os dados no R. Há várias possíveis maneiras de fazer isto. Vamos aqui usar a função `scan` e entrar com os dados por linha da tabela. Digitamos o comando abaixo e e função `scan` recebe os dados. Depois de digitar o último dado digitamos ENTER em um campo em branco e a função encerra a entrada de daods retornando para o *prompt* do programa.

OBS: Note que, sendo um programa escrito na língua inglesa, os decimais devem ser indicados por '.' e não por vírgulas.

```
> y <- scan()
1: 4.4
2: 5.9
3: 6.0
...
24: 6.7
25:
Read 24 items
```

Agora vamos montar um *data.frame* com os dados e os indicadores de blocos e tratamentos.

```
ex02 <- data.frame(estag = factor(rep(1:6, each=4)), bloco=factor(rep(1:4, 6)), resp=y)
```

Note que usamos a função `factor` para indicar que as variáveis `blocos` e `estag` são níveis de fatores e não valores numéricos.

Vamos agora explorar um pouco os dados.

```
names(ex02)
summary(ex02)

attach(ex02)

plot(resp ~ estag + bloco)

interaction.plot(estag, bloco, resp)
interaction.plot(bloco, estag, resp)
```

```
ex02.mt <- tapply(resp, estag, mean)
ex02.mt
ex02.mb <- tapply(resp, bloco, mean)
ex02.mb

plot.default(estag, resp)
points(ex02.mt, pch="x", col=2, cex=1.5)

plot.default(bloco, resp)
points(ex02.mb, pch="x", col=2, cex=1.5)
```

Nos gráficos e resultados acima procuramos captar os principais aspectos dos dados bem como verificar se não há interação entre blocos e tratamentos, o que não deve acontecer neste tipo de experimento.

A seguir vamos ajustar o modelo e obter outros resultados, incluindo a análise de resíduos e testes para verificar a validades dos pressupostos do modelo.

```
ex02.av <- aov(resp ~ bloco + estag)
anova(ex02.av)

names(ex02.av)

par(mfrow=c(2,2))
plot(ex02.av)

par(mfrow=c(2,1))
residuos <- (ex02.av$residuals)

plot(ex02$bloco,residuos)
title("Resíduos vs Blocos")

plot(ex02$estag,residuos)
title("Resíduos vs Estágios")

par(mfrow=c(2,2))
preditos <- (ex02.av$fitted.values)
plot(residuos,preditos)
title("Resíduos vs Preditos")
respad <- (residuos/sqrt(anova(ex02.av)$"Mean Sq"[2]))
boxplot(respad)
title("Resíduos Padronizados")
qqnorm(residuos,ylab="Residuos", main=NULL)
qqline(residuos)
title("Grafico Normal de \n Probabilidade dos Resíduos")

## teste para normalidade
shapiro.test(residuos)

## Testando a não aditividade
```

```
## primeiro vamos extrair coeficientes de tratamentos e blocos
ex02.av$coeff
bl <- c(0, ex02.av$coeff[2:4])
tr <- c(0, ex02.av$coeff[5:9])
bl
tr

## agora criar um novo termo e testar sua significancia na ANOVA
bltr <- rep(bl, 6) * rep(tr, rep(4,6))

ttna <- update(ex02.av, .~. + bltr)
anova(ttna)
```

Os resultados acima indicam que os pressupostos estão obedecidos para este conjunto de dados e a análise de variância é válida. Como foi detectado efeito de tratamentos vamos proceder fazendo um teste de comparações múltiplas e encerrar as análises desanexando o objeto do caminho de procura.

```
ex02.tk <- TukeyHSD(ex02.av, "estag", ord=T)
ex02.tk
plot(ex02.tk)

detach(ex02)
```

5 Experimentos em esquema fatorial

O experimento fatorial descrito na apostila do curso de Planejamento de Experimentos II comparou o crescimento de mudas de eucalipto considerando diferentes recipientes e espécies.

1. **Lendo os dados**

   Vamos considerar agora que os dados já esteajm digitados em um arquivo texto. Clique aqui para ver e copiar o arquivo com conjunto de dados para o seu diretório de trabalho.

   A seguir vamos ler (importar) os dados para R com o comando `read.table`:

   ```
   > ex04 <- read.table("exemplo04.txt", header=T)
   > ex04
   ```

   Antes de começar as análise vamos inspecionar o objeto que contém os dados para saber quantas observações e variáveis há no arquivo, bem como o nome das variáveis. Vamos tembém pedir o R que exiba um rápido resumo dos dados.

   ```
   > dim(ex04)
   [1] 24  3

   > names(ex04)
   [1] "rec"  "esp"  "resp"

   > attach(ex04)

   > is.factor(rec)
   [1] TRUE
   > is.factor(esp)
   [1] TRUE
   > is.factor(resp)
   [1] FALSE
   > is.numeric(resp)
   [1] TRUE
   ```

   Nos resultados acima vemos que o objeto `ex04` que contém os dados tem 24 linhas (observações) e 3 colunas (variáveis). As variáveis tem nomes `rec`, `esp` e resp, sendo que as duas primeiras são *fatores* enquanto `resp` é uma variável numérica, que neste caso é a variável resposta. O objeto `ex04` foi incluído no caminho de procura usando o comando `attach` para facilitar a digitação.

2. **Análise exploratória**

   Inicialmente vamos obter um resumo de nosso conjunto de dados usando a função `summary`.

   ```
   > summary(ex04)
    rec    esp          resp
    r1:8   e1:12   Min.   :18.60
    r2:8   e2:12   1st Qu.:19.75
    r3:8           Median :23.70
   ```

```
                  Mean   :22.97
                  3rd Qu.:25.48
                  Max.   :26.70
```

Note que para os fatores são exibidos o número de dados em cada nível do fator. Já para a variável numérica são mostrados algumas medidas estatísticas. Vamos explorar um pouco mais os dados

```
> ex04.m <- tapply(resp, list(rec,esp), mean)
> ex04.m
       e1     e2
r1 25.650 25.325
r2 25.875 19.575
r3 20.050 21.325

> ex04.mr <- tapply(resp, rec, mean)
> ex04.mr
     r1      r2      r3
25.4875 22.7250 20.6875

> ex04.me <- tapply(resp, esp, mean)
> ex04.me
      e1       e2
23.85833 22.07500
```

Nos comandos acima calculamos as médias para cada fator, assim como para os cruzamentos entre os fatores. Note que podemos calcular outros resumos além da média. Experimente nos comandos acima substituir `mean` por `var` para calcular a variância de cada grupo, e por `summary` para obter um outro resumo dos dados.

Em experimentos fatoriais é importante verificar se existe interação entre os fatores. Inicialmente vamos fazer isto graficamente e mais a frente faremos um teste formal para presença de interação. Os comandos a seguir são usados para produzir os gráficos exibidos na Figura 1.

```
> par(mfrow=c(1,2))
> interaction.plot(rec, esp, resp)
> interaction.plot(esp, rec, resp)
```

3. **Análise de variância**

   Seguindo o modelo adequado, o análise de variância para este experimento inteiramente casualizado em esquema fatorial pode ser obtida com o comando:

```
> ex04.av <- aov(resp ~ rec + esp + rec * esp)
```

   Entretanto o comando acima pode ser simplificado produzindo os mesmos resultados com o comando

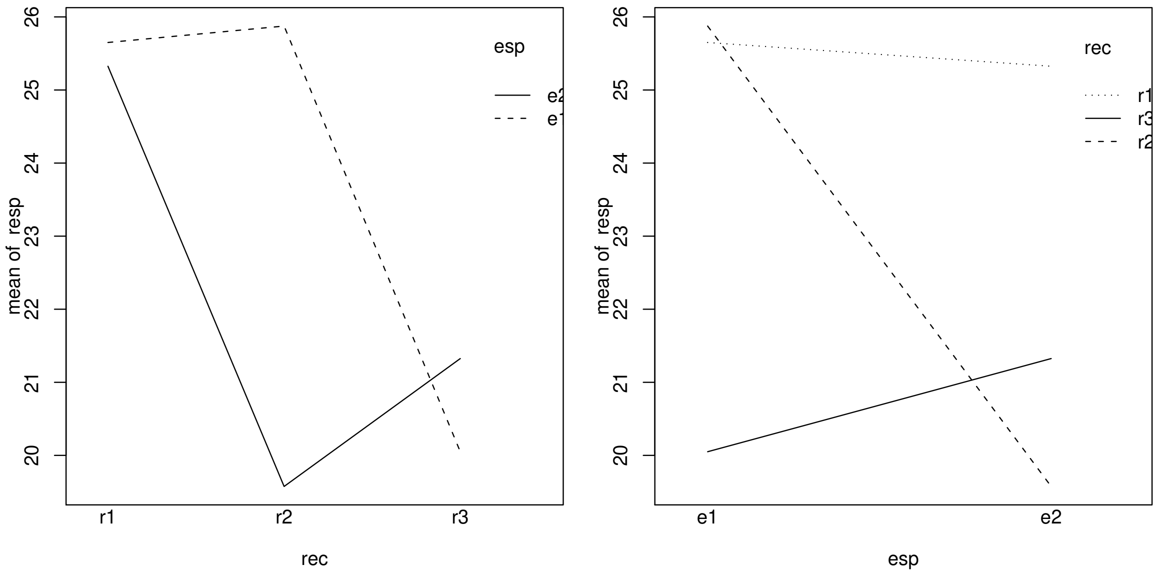

Figura 1: Gráficos de interação entre os fatores.

```
> ex04.av <- aov(resp ~ rec * esp)
> summary(ex04.av)
            Df Sum Sq Mean Sq F value    Pr(>F)
rec          2 92.861  46.430  36.195 4.924e-07 ***
esp          1 19.082  19.082  14.875  0.001155 **
rec:esp      2 63.761  31.880  24.853 6.635e-06 ***
Residuals   18 23.090   1.283
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Isto significa que no R, ao colocar uma interação no modelo, os efeitos principais são incluídos automaticamente. Note no quadro de análise de variância que a interação é denotada por `rec:esp`. A análise acima ostra que este efeito é significativo, confirmando o que verificamos nos gráficos de interação vistos anteriormente.

O objeto `ex04.av` guarda todos os resultados da análise e pode ser explorado por diversos comandos. Por exemplo a função `model.tables` aplicada a este objeto produz tabelas das médias definidas pelo modelo. O resultado mostra a média geral, médias de cada nível fatores e das combinações dos níveis dos fatores. Note que no resultado está incluído também o número de dados que gerou cada média.

```
> ex04.mt <- model.tables(ex04.av, ty="means")
> ex04.mt
Tables of means
Grand mean

22.96667

 rec
       r1    r2    r3
    25.49 22.73 20.69
rep  8.00  8.00  8.00
```

```
 esp
       e1    e2
    23.86 22.07
rep 12.00 12.00

 rec:esp
     esp
rec   e1     e2
  r1  25.650 25.325
  rep  4.000  4.000
  r2  25.875 19.575
  rep  4.000  4.000
  r3  20.050 21.325
  rep  4.000  4.000
```

Mas isto não é tudo! O objeto `ex04.av` possui vários elementos que guardam informações sobre o ajuste.

```
> names(ex04.av)
 [1] "coefficients"  "residuals"     "effects"       "rank"
 [5] "fitted.values" "assign"        "qr"            "df.residual"
 [9] "contrasts"     "xlevels"       "call"          "terms"
[13] "model"

> class(ex04.av)
[1] "aov" "lm"
```

O comando `class` mostra que o objeto `ex04.av` pertence às classes `aov` e `lm`. Isto significa que devem haver *métodos* associados a este objeto que tornam a exploração do resultado mais fácil. Na verdade já usamos este fato acima quando digitamos o comando `summary(ex04.av)`. Existe uma função chamada `summary.aov` que foi utilizada já que o objeto é da classe `aov`. Iremos usar mais este mecanismo no próximo passo da análise.

4. **Análise de resíduos**

   Após ajustar o modelo devemos proceder a análise dos resíduos para verificar os pressupostos. O R produz automaticamente 4 gráficos básicos de resíduos conforme a Figura 2 com o comando `plot`.

```
> par(mfrow=c(2,2))
> plot(ex04.av)
```

   Os gráficos permitem uma análise dos resíduos que auxiliam no julgamento da adequacidade do modelo. Evidentemente voce não precisa se limitar os gráficos produzidos automaticamente pelo R – voce pode criar os seus próprios gráficos muito facilmente. Neste gráficos voce pode usar outras variáveis, mudar texto de eixos e títulos, etc, etc, etc. Examine os comandos abaixo e os gráficos por eles produzidos.

```
> par(mfrow=c(2,1))
> residuos <- resid(ex04.av)
```

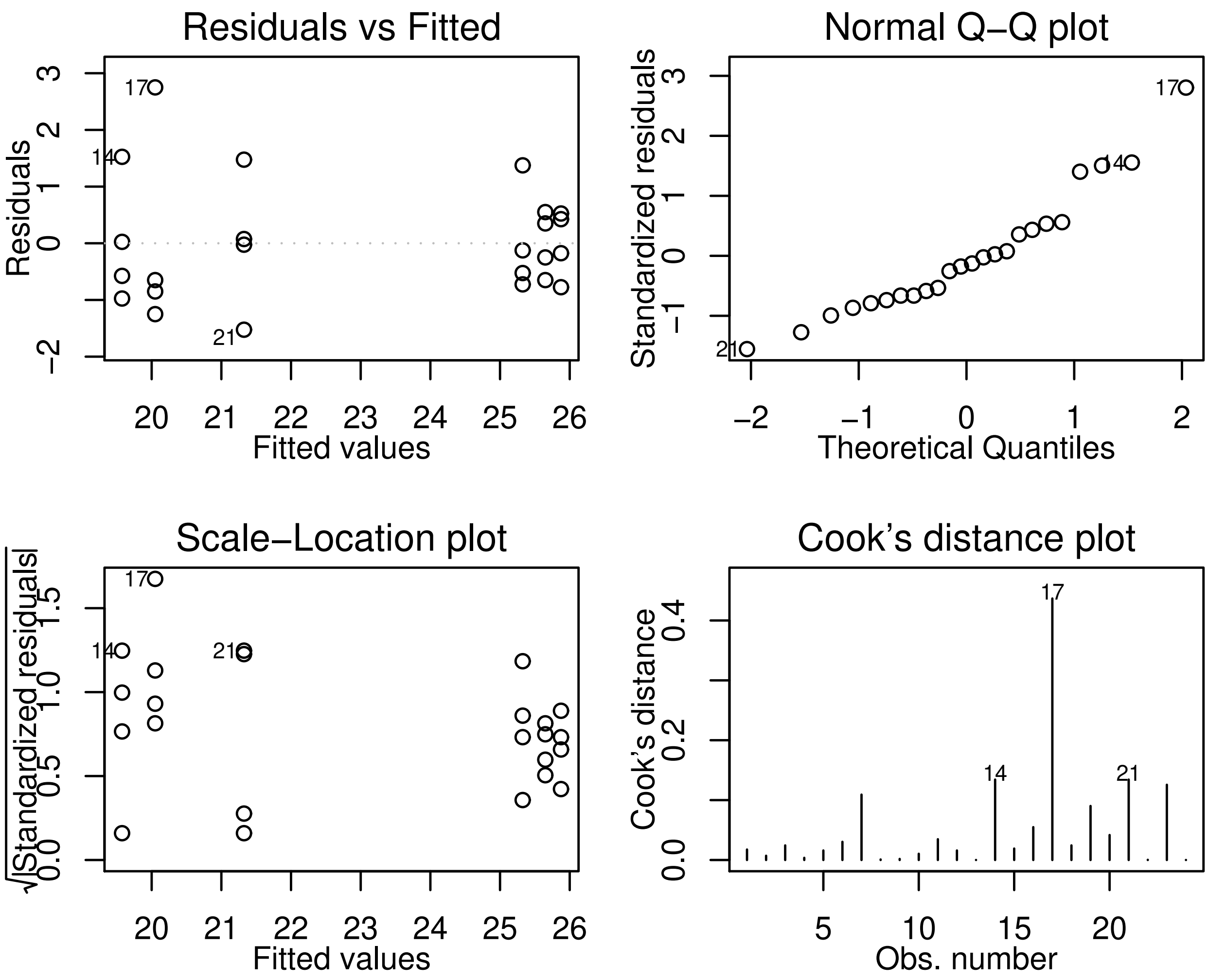

Figura 2: Gráficos de resíduos produzidos automaticamente pelo R.

```
> plot(ex04$rec, residuos)
> title("Resíduos vs Recipientes")

> plot(ex04$esp, residuos)
> title("Resíduos vs Espécies")

> par(mfrow=c(2,2))
> preditos <- (ex04.av$fitted.values)
> plot(residuos, preditos)
> title("Resíduos vs Preditos")
> s2 <- sum(resid(ex04.av)^2)/ex04.av$df.res
> respad <- residuos/sqrt(s2)
> boxplot(respad)
> title("Resíduos Padronizados")
> qqnorm(residuos,ylab="Residuos", main=NULL)
> qqline(residuos)
> title("Grafico Normal de \n Probabilidade dos Resíduos")
```

Além disto há alguns testes já programados. Como exemplo vejamos e teste de Shapiro-

Wilk para testar a normalidade dos resíduos.

```
> shapiro.test(residuos)

        Shapiro-Wilk normality test

data:  residuos
W = 0.9293, p-value = 0.09402
```

5. **Desdobrando interações**

   Conforma visto na apostila do curso, quando a interação entre os fatores é significativa podemos desdobrar os graus de liberdade de um fator dentro de cada nível do outro. A forma de fazer isto no R é reajustar o modelo utilizando a notação / que indica efeitos aninhados. Desta forma podemos desdobrar os efeitos de espécie dentro de cada recipiente e vice versa conforme mostrado a seguir.

```
> ex04.avr <- aov(resp ~ rec/esp)
> summary(ex04.avr, split=list("rec:esp"=list(r1=1, r2=2, r3=3)))
              Df Sum Sq Mean Sq F value    Pr(>F)
rec            2 92.861  46.430 36.1952 4.924e-07 ***
rec:esp        3 82.842  27.614 21.5269 3.509e-06 ***
  rec:esp: r1  1  0.211   0.211  0.1647    0.6897
  rec:esp: r2  1 79.380  79.380 61.8813 3.112e-07 ***
  rec:esp: r3  1  3.251   3.251  2.5345    0.1288
Residuals     18 23.090   1.283
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1


> ex04.ave <- aov(resp ~ esp/rec)
> summary(ex04.ave, split=list("esp:rec"=list(e1=c(1,3), e2=c(2,4))))
              Df  Sum Sq Mean Sq F value    Pr(>F)
esp            1  19.082  19.082  14.875  0.001155 **
esp:rec        4 156.622  39.155  30.524 8.438e-08 ***
  esp:rec: e1  2  87.122  43.561  33.958 7.776e-07 ***
  esp:rec: e2  2  69.500  34.750  27.090 3.730e-06 ***
Residuals     18  23.090   1.283
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

6. **Teste de Tukey para comparações múltiplas**

   Há vários testes de comparações múltiplas disponíveis na literatura, e muitos deles implementados no R. Os que não estão implementados podem ser facilmente calculados utilizando os recursos do R.

   Vejamos por exemplo duas formas de usar o *Teste de Tukey*, a primeira usando uma implementação com a função `TukeyHSD` e uma segunda fazendo ops cálculos necessários com o R.

   Poderíamos simplesmente digitar:

```
> ex04.tk <- TukeyHSD(ex04.av)
> plot(ex04.tk)
> ex04.tk
```

e obter diversos resultados. Entretanto nem todos nos interessam. Como a interação foi significativa na análise deste experimento a comparação dos níveis fatores principais não nos interessa.

Podemos então pedir a função que somente mostre a comparação de médias entre as combinações dos nívies dos fatores.

```
> ex04.tk <- TukeyHSD(ex04.ave, "esp:rec")
> plot(ex04.tk)
> ex04.tk
  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = resp ~ esp/rec)

$"esp:rec"
        diff        lwr       upr
 [1,] -0.325 -2.8701851  2.220185
 [2,]  0.225 -2.3201851  2.770185
 [3,] -6.075 -8.6201851 -3.529815
 [4,] -5.600 -8.1451851 -3.054815
 [5,] -4.325 -6.8701851 -1.779815
 [6,]  0.550 -1.9951851  3.095185
 [7,] -5.750 -8.2951851 -3.204815
 [8,] -5.275 -7.8201851 -2.729815
 [9,] -4.000 -6.5451851 -1.454815
[10,] -6.300 -8.8451851 -3.754815
[11,] -5.825 -8.3701851 -3.279815
[12,] -4.550 -7.0951851 -2.004815
[13,]  0.475 -2.0701851  3.020185
[14,]  1.750 -0.7951851  4.295185
[15,]  1.275 -1.2701851  3.820185
```

Mas ainda assim temos resultados que não interessam. Mais especificamente estamos intessados nas comparações dos níveis de um fator dentro dos nívies de outro. Por exemplo, vamos fazer as comparações dos recipientes para cada uma das espécies.

Primeiro vamos obter

```
> s2 <- sum(resid(ex04.av)^2)/ex04.av$df.res
> dt <- qtukey(0.95, 3, 18) * sqrt(s2/4)
> dt
[1] 2.043945
>
> ex04.m
       e1     e2
r1 25.650 25.325
```

```
r2 25.875 19.575
r3 20.050 21.325
>
> m1 <- ex04.m[,1]
> m1
    r1     r2     r3
25.650 25.875 20.050
> m1d <- outer(m1,m1,"-")
> m1d
       r1     r2    r3
r1  0.000 -0.225 5.600
r2  0.225  0.000 5.825
r3 -5.600 -5.825 0.000
> m1d <- m1d[lower.tri(m1d)]
> m1d
    r2     r3   <NA>
 0.225 -5.600 -5.825
>
> m1n <- outer(names(m1),names(m1),paste, sep="-")
> names(m1d) <- m1n[lower.tri(m1n)]
> m1d
 r2-r1  r3-r1  r3-r2
 0.225 -5.600 -5.825
>
> data.frame(dif = m1d, sig = ifelse(abs(m1d) > dt, "*", "ns"))
         dif sig
r2-r1  0.225  ns
r3-r1 -5.600   *
r3-r2 -5.825   *
>
> m2 <- ex04.m[,2]
> m2d <- outer(m2,m2,"-")
> m2d <- m2d[lower.tri(m2d)]
> m2n <- outer(names(m2),names(m2),paste, sep="-")
> names(m2d) <- m2n[lower.tri(m2n)]
> data.frame(dif = m2d, sig = ifelse(abs(m2d) > dt, "*", "ns"))
        dif sig
r2-r1 -5.75   *
r3-r1 -4.00   *
r3-r2  1.75  ns
```

6 Transformação de dados

Tranformação de dados é uma das possíveis formas de contarnar o problema de dados que não obedecem os pressupostos da análise de variância. Vamos ver como isto poder ser feito com o programa R.

Considere o seguinte exemplo da apostila do curso.

Tabela 2: Número de reclamações em diferentes sistemas de atendimento

Trat	Repetições					
	1	2	3	4	5	6
1	2370	1687	2592	2283	2910	3020
2	1282	1527	871	1025	825	920
3	562	321	636	317	485	842
4	173	127	132	150	129	227
5	193	71	82	62	96	44

Inicialmente vamos entrar com os dados usando a função `scan` e montar um *data-frame*.

```
> y <- scan()
1: 2370
2: 1687
3: 2592
...
30: 44
31:
Read 30 items

> tr <- data.frame(trat = rep(1:5, each=6), resp = y)
> tr
```

A seguir vamos fazer ajustar o modelo e inspecionar os resíduos.

```
tr.av <- aov(resp ~ trat, data=tr)
plot(tr.av)
```

O gráfico de resíduos *vs* valores preditos mostra claramente uma heterogeneidade de variâncias e o $QQ-plot$ mostra um comportamento dos dados que se afasta muito da normal. A menssagem é clara mas podemos ainda fazer testes para verificar o desvio dos pressupostos.

```
> bartlett.test(tr$resp, tr$trat)

        Bartlett test for homogeneity of variances

data:  tr$resp and tr$trat
Bartlett's K-squared = 29.586, df = 4, p-value = 5.942e-06

> shapiro.test(tr.av$res)

        Shapiro-Wilk normality test

data:  tr.av$res
W = 0.939, p-value = 0.08535
```

Nos resultados acima vemos que a homogeneidade de variâncias foi rejeitada.

Para tentar contornar o problema vamos usar a transformação Box-Cox, que consiste em transformar os dados de acordo com a expressão

$$y' = \frac{y^\lambda - 1}{\lambda},$$

onde λ é um parâmeto a ser estimado dos dados. Se $\lambda = 0$ a equação acima se reduz a

$$y' = \log(y),$$

onde log é o logarítmo neperiano. Uma vez obtido o valor de λ encontramos os valores dos dados transformados conforme a equação acima e utilizamos estes dados transformados para efetuar as análises.

A função `boxcox` do pacote MASS calcula a verossimilhança perfilhada do parâmetro λ. Devemos escolher o valor que maximiza esta função. Nos comandos a seguir começamos carregando o pacote MASS e depois obtemos o gráfico da verossimilhança perfilhada. Como estamos interessados no máximo fazermos um novo gráfico com um *zoom* na região de interesse.

```
require(MASS)
boxcox(resp ~ trat, data=tr, plotit=T)
boxcox(resp ~ trat, data=tr, lam=seq(-1, 1, 1/10))
```

O gráfico mostra que o valor que maximiza a função é aproximadamente $\hat{\lambda} = 0.1$. Desta forma o próximo passo é obter os dados transformados e depois fazer as análise utilizando estes novos dados.

```
tr$respt <- (tr$resp^(0.1) - 1)/0.1
tr.avt <- aov(respt ~ trat, data=tr)
plot(tr.avt)
```

Note que os resíduos tem um comportamento bem melhor do que o observado para os dados originais. A análise deve prosseguir usando então os dados transformados.

NOTA: No gráfico da verossimilhança perfilhada notamos que é mostrado um intervalo de confiança para λ e que o valor 0 está contido neste intervalo. Isto indica que podemos utilizar a transformação logarítimica dos dados e os resultados serão bom próximos dos obtidos com a transformação préviamente adotada.

```
tr.avl <- aov(log(resp) ~ trat, data=tr)
plot(tr.avl)
```

7 Experimentos com fatores hierárquicos

Vamos considerar o exemplo da apostila retirado do livro de Montgomery. Clique aqui para ver e copiar o arquivo com conjunto de dados para sua área de trabalho.

O experimento estuda a variabilidade de lotes e fornecedores na puraza da matéria prima. A análise assume que os fornecedores são um efeito fixo enquanto que lotes são efeitos aleatórios.

Inicialmente vamos ler (importar) os dados para R com o comando `read.table` (certifique-se que o arquivo `exemplo06.txt` está na sua área de trabalho ou coloque o caminho do arquivo no comando abaixo). A seguir vamos examinar o objeto que contém os dados.

```
> ex06 <- read.table("exemplo06.txt", header=T)
> ex06

> dim(ex06)
[1] 36  3

> names(ex06)
[1] "forn" "lot"  "resp"

> is.factor(ex06$forn)
[1] FALSE
> is.factor(ex06$lot)
[1] FALSE

> ex06$forn <- as.factor(ex06$forn)
> ex06$lot <- as.factor(ex06$lot)

> is.factor(ex06$resp)
[1] FALSE
> is.numeric(ex06$resp)
[1] TRUE

> summary(ex06)
 forn   lot        resp
 1:12   1:9   Min.   :-4.0000
 2:12   2:9   1st Qu.:-1.0000
 3:12   3:9   Median : 0.0000
        4:9   Mean   : 0.3611
              3rd Qu.: 2.0000
              Max.   : 4.0000
```

Nos comandos acima verificamos que o objeto `ex06` possui 36 linhas correspondentes às observações e 3 colunas que correspondem às variáveis `forn` (fornecedor), `lot` (lote) e `resp` (a variável resposta).

A seguir verificamos que `forn` e `lot` não foram lidas como fatores. **NÃO** podemos seguir as análise desta forma pois o R leria os valores destas variáveis como quantidades numéricas e não como indicadores dos nívies dos fatores. Para corrigir isto usamos o comando `as.factor` para indicar ao R que estas variáveis são fatores.

Finalmente verificamos que a variável resposta é numérica e produzimos um rápido resumo dos dados.

Na sequência deveríamos fazer uma análise exploratória, alguns gráficos descritivos etc, como na análise dos experimentos mostrados anteriormente. Vamos deixar isto por conta do leitor e passar direto para a análise de variância.

A notação para indicar efeitos aninhados no modelo é `/`. Desta forma poderíamos ajustar o modelo da seguinte forma:

```
> ex06.av <- aov(resp ~ forn/lot, data=ex06)
> summary(ex06.av)
            Df Sum Sq Mean Sq F value  Pr(>F)
forn         2 15.056   7.528  2.8526 0.07736 .
forn:lot     9 69.917   7.769  2.9439 0.01667 *
Residuals   24 63.333   2.639
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Embora os elementos do quadro de análise de variância estejam corretos o teste F para efeito dos fornecedores está **ERRADO**. A análise acima considerou todos os efeitos como fixos e portanto dividiu os quadrados médios dos efeitos pelo quadrado médio do resíduo. Como `lotes` é um efeito aleatório deveríamos dividir o quadrado médio de to termo `lot` pelo quadrado médio de `forn:lot`

Uma forma de indicar a estrutura hierárquica ao R é especificar o modelo de forma que o termo de resíduo seja dividido de maneira adequada. Veja o resultado abaixo.

```
> ex06.av1 <- aov(resp ~ forn/lot + Error(forn) , data=ex06)
> summary(ex06.av1)

Error: forn
     Df  Sum Sq Mean Sq
forn  2 15.0556  7.5278

Error: Within
          Df Sum Sq Mean Sq F value  Pr(>F)
forn:lot   9 69.917   7.769  2.9439 0.01667 *
Residuals 24 63.333   2.639
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Agora o teste F errado não é mais mostrado, mas o teste correto também não foi feito! Isto não é problema porque podemos extrair os elementos que nos interessam e fazer o teste desejado. Primeiro vamos guardar verificamos que o comando `anova` produz uma lista que tem entre seus elementos os graus de liberdade `Df` e os quadrados médios (`Mean Sq`. A partir destes elementos podemos obtemos o valor da estatística F e o valor P associado.

```
> ex06.anova <- anova(ex06.av)
> is.list(ex06.anova)
[1] TRUE

> names(ex06.anova)
[1] "Df"      "Sum Sq"  "Mean Sq" "F value" "Pr(>F)"

> ex06.anova$Df
 1  2
```

```
 2  9 24
> ex06.anova$Mean
       1        2
7.527778 7.768519 2.638889

> Fcalc <- ex06.anova$Mean[1]/ex06.anova$Mean[2]
> Fcalc
        1
0.9690107
> pvalor <- 1 - pf(Fcalc, ex06.anova$Df[1], ex06.anova$Df[2])
> pvalor
        1
0.4157831
```

USANDO O PACOTE NLME

Uma outra possível e elegante solução no R para este problema é utilizar a função `lme` do pacote `nlme`. Note que a abordagem do problema por este pacote é um pouco diferente da forma apresentada no curso por se tratar de uma ferramente geral para modelos com efeitos aleatórios. Entretanto os todos os elementos relevantes da análise estão incluídos nos resultados. Vamos a seguir ver os comandos necessários comentar os resultados.

Inicialmente temos que carregar o pacote `nlme` com o comando `require`.
A seguir criamos uma variável para indicar o efeito aleatório que neste exemplo chamamos de `ex06$fa` utilizando a função `getGroups`.
Feito isto podemos rodar a função `lme` que faz o ajuste do modelo.

```
> require(nlme)
[1] TRUE
> ex06$fa <- getGroups(ex06, ~ 1|forn/lot, level=2)
> ex06.av <- lme(resp ~ forn, random=~1|fa, data=ex06)
> ex06.av
Linear mixed-effects model fit by REML
  Data: ex06
  Log-restricted-likelihood: -71.42198
  Fixed: resp ~ forn
(Intercept)       forn2       forn3
 -0.4166667   0.7500000   1.5833333

Random effects:
 Formula: ~1 | fa
        (Intercept) Residual
StdDev:    1.307561 1.624483

Number of Observations: 36
Number of Groups: 12
```

Este modelo tem a variável `forn` como efeito fixo e a variável `lot` como efeito aleatório com o componente de variância σ^2_{lote}. Além disto temos a variância residual σ^2. A saída acima mostra as estimativas destes componentes da variância como sendo $\hat{\sigma}^2_{lote} = (1.307)^2 = 1.71$ e $\hat{\sigma}^2 = (1.624)^2 = 2.64$.

O comando `anova` vai mostrar a análise de variância com apenas os efeitos principais. O fato do programa não incluir o efeito aleatório de lotes na saída não causa problema algum. O comando `intervals` mostra os intervalos de confiança para os componentes de variância. Portanto para verificar a significância do efeito de lotes basta ver se o intervalo para este componente de variância exclui o valor 0, o que é o caso neste exemplo conforme vamos abaixo.

```
> anova(ex06.av)
            numDF denDF   F-value p-value
(Intercept)     1    24 0.6043242  0.4445
forn            2     9 0.9690643  0.4158
> intervals(ex06.av)
Approximate 95% confidence intervals

 Fixed effects:
                 lower       est.    upper
(Intercept) -2.0772277 -0.4166667 1.243894
forn2       -1.8239746  0.7500000 3.323975
forn3       -0.9906412  1.5833333 4.157308

 Random Effects:
  Level: fa
                    lower     est.    upper
sd((Intercept)) 0.6397003 1.307561 2.672682

 Within-group standard error:
   lower     est.    upper
1.224202 1.624483 2.155644
```

Finalmente uma versão mais detalhada dos resultados pode ser obtida com o comando `summary`.

```
> summary(ex06.av)
Linear mixed-effects model fit by REML
 Data: ex06
      AIC      BIC    logLik
  152.8440 160.3265 -71.42198

Random effects:
 Formula: ~1 | fa
        (Intercept) Residual
StdDev:    1.307561 1.624483

Fixed effects: resp ~ forn
                 Value Std.Error DF    t-value p-value
(Intercept) -0.4166667 0.8045749 24 -0.5178718  0.6093
forn2        0.7500000 1.1378407  9  0.6591432  0.5263
forn3        1.5833333 1.1378407  9  1.3915246  0.1975
 Correlation:
      (Intr) forn2
forn2 -0.707
forn3 -0.707  0.500
```

```
Standardized Within-Group Residuals:
       Min         Q1        Med         Q3        Max
-1.4751376 -0.7500844  0.0812409  0.7060895  1.8720268

Number of Observations: 36
Number of Groups: 12
```

O próximo passo da seria fazer uma análise dos resíduos para verificar os pressupostos, semelhante ao que foi feito nos experimentos anteriormente analisados. Vamos deixar isto por conta do leitor.

8 Análise De Covariância

Vejamos agora a análise de covariância do exemplo da apostila do curso. Clique aqui para ver e copiar o arquivo com conjunto de dados para sua área de trabalho.

Começamos com a leitura e organização dos dados. Note que neste caso temos 2 variáveis numéricas, a resposta (`resp`) e a covariável (`cov`).

```
> ex12 <- read.table("exemplo12.txt", header=T)
> ex12

> dim(ex12)
[1] 15  3
> names(ex12)
[1] "maq"  "cov"  "resp"
>
> ex12$maq <- as.factor(ex12$maq)
> is.numeric(ex12$cov)
[1] TRUE
> is.numeric(ex12$resp)
[1] TRUE
>
> summary(ex12)
 maq        cov              resp
 1:5   Min.   :15.00   Min.   :32.0
 2:5   1st Qu.:21.50   1st Qu.:36.5
 3:5   Median :24.00   Median :40.0
       Mean   :24.13   Mean   :40.2
       3rd Qu.:27.00   3rd Qu.:43.0
       Max.   :32.00   Max.   :49.0
```

Na análise de covariância os testes de significância tem que ser obtidos em ajustes separados. Isto é porque não temos ortogonalidade entre os fatores.

Primeiro vamos testar o intercepto (coeficiente β) da reta de regressão. Na análise de variância abaixo devemos considerar apenas o teste referente à variável `cov` que neste caso está corrigida para o efeito de `maq`. Note que para isto a variável `cov` tem que ser a última na especificação do modelo.

```
> ex12.av <- aov(resp ~ maq + cov, data=ex12)
> summary(ex12.av)
            Df  Sum Sq Mean Sq F value    Pr(>F)
maq          2 140.400  70.200  27.593 5.170e-05 ***
cov          1 178.014 178.014  69.969 4.264e-06 ***
Residuals   11  27.986   2.544
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

A seguir testamos o efeito do fator `maq` corrigindo para o efeito da covariável. Para isto basta inverter a ordem dos termos na especificação do modelo.

```
> ex12.av <- aov(resp ~ cov + maq, data=ex12)
> summary(ex12.av)
            Df  Sum Sq Mean Sq  F value   Pr(>F)
```

```
cov          1 305.130 305.130 119.9330 2.96e-07 ***
maq          2  13.284   6.642   2.6106   0.1181
Residuals   11  27.986   2.544
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

9 Experimentos Em Parcelas Subdivididas

Vamos mostrar aqui como especificar o modelo para análise de experimentos em parcelas subdivididas. Os comandos abaixo mostram a leitura e preparação dos dados e a obtenção da análise de variância. Deixamos por conta do leitor a análise exploratória e de resíduos, desdobramento das interações e testes de comparações múltiplas.

Considere o experimento em parcelas subdivididas de dados de produção de aveia descrito na apostila do curso. Clique aqui para ver e copiar o arquivo com conjunto de dados. A obtenção da análise de variância é ilustrada nos comandos e saídas abaixo.

```
> ex09 <- read.table("exemplo09.txt", header=T)
> ex09

> dim(ex09)
[1] 64  4
> names(ex09)
[1] "a"     "b"     "bloco" "resp"
>
> ex09$a <- as.factor(ex09$a)
> ex09$b <- as.factor(ex09$b)
> ex09$bloco <- as.factor(ex09$bloco)
>
> summary(ex09)
 a      b      bloco       resp
 1:16   1:16   1:16   Min.   :28.30
 2:16   2:16   2:16   1st Qu.:44.90
 3:16   3:16   3:16   Median :52.30
 4:16   4:16   4:16   Mean   :52.81
                      3rd Qu.:62.38
                      Max.   :75.40
>
> ex09.av <- aov(resp ~ bloco + a*b + Error(bloco/a), data=ex09)
> summary(ex09.av)

Error: bloco
      Df  Sum Sq Mean Sq
bloco  3 2842.87  947.62

Error: bloco:a
          Df  Sum Sq Mean Sq F value   Pr(>F)
a          3 2848.02  949.34  13.819 0.001022 **
Residuals  9  618.29   68.70
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Error: Within
          Df Sum Sq Mean Sq F value   Pr(>F)
b          3 170.54   56.85  2.7987 0.053859 .
a:b        9 586.47   65.16  3.2082 0.005945 **
Residuals 36 731.20   20.31
```

```
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Sobre os ministrantes do curso

Adilson dos Anjos é Eng. Agrônomo pela Universidade Federal do Santa Catarina, Mestre em Agronomia com área de concentração em estatística e experimentação agronômica pela ESALQ/USP.

Adilson é professor do Departamento de Estatística da Universidade Federal do Paraná desde 1998.

Paulo Justiniano Ribeiro Junior é Eng. Agrônomo pela ESAL, Lavras (atual UFLA), Mestre em Agronomia com área de concentração em estatística e experimentação agronômica pela ESALQ/USP. PhD em Estatística pela Lancaster University, UK.

PJRJr é professor do Departamento de Estatística da Universidade Federal do Paraná desde 1992 e tem usado o programa R em suas pesquisas desde 1999. É autor dos pacotes geoR e geoRglm contribuídos ao **CRAN** (*Compreheensive R Arquives Network*).