

Modelagem e análise de dados experimentais com o programa computacional R

Walmes Marques Zeviani*

de 7 à 11 de fevereiro de 2011

Sumário

1	Introdução à manipulação de objetos e funções	4
1.1	Instalação do R	4
1.2	Pedindo ajuda	4
1.3	Criação, atribuição, acesso e modificação de objetos	5
1.4	Informações sobre objetos (atributos)	6
1.5	Operações matemáticas	7
1.6	Operações estatísticas	8
1.7	Construindo funções	8
2	Importação de dados e análise exploratória	9
2.1	Importando dados	9
2.2	Explorações gráficas (1)	10
2.3	Explorações gráficas (2)	11
2.4	Recursos gráficos avançados	11
3	Estatística básica	12
3.1	Medidas de posição, dispersão e forma	12
3.2	Teste de hipótese e intervalo de confiança para parâmetros	14
4	Regressão linear	14
4.1	Importando e manipulando dados	14
4.2	Regressão linear simples	15
4.3	Regressão linear múltipla	15
4.4	Seleção de modelos/variáveis	16
4.5	Remoção de pontos discrepantes	16
4.6	Predição de valores a partir do modelo escolhido	17
4.7	Representação gráfica do ajuste	17
4.8	Mais sobre análise de resíduos e R^2 (quarteto de Anscombe)	18

*Documento concluído em 8 de fevereiro de 2011 às 17:52:48 – Centro Politécnico – Universidade Federal do Paraná.

5	Regressão não linear	18
5.1	Motivação	18
5.2	Definição	19
5.3	Exemplo de modelos não lineares	19
5.4	Uso de recursos gráficos para entender o significado dos parâmetros	20
5.5	Estimação de parâmetros em modelos não lineares	21
5.6	Ajuste de modelo não linear aos dados de DAP	22
5.7	Comparação de curvas ajustadas	24
5.8	Ajuste de modelos não lineares com a <code>library{nlme}</code>	25
6	Análise de experimento com um fator em DIC	26
6.1	Importando dados	26
6.2	Análise de variância	27
6.3	Aplicando teste de Tukey para comparar médias	27
6.4	Aplicando teste de Scott-Knott para agrupar médias	28
6.5	Aplicando contrastes	28
6.6	Estudo das taxas de erro tipo I dos testes	29
7	Análise de experimentos de um fator em DBC	29
7.1	Entrada de dados	29
7.2	Análise de variância	30
7.3	Teste de médias	30
7.4	Observações perdidas	30
8	Análise de experimento fatorial duplo em DIC	32
8.1	Análise de variância	32
8.2	Testes de médias	33
9	Análise de fatorial duplo em DBC	33
9.1	Entrando com os dados	33
9.2	Análise de variância e desdobramento das somas de quadrados	34
9.3	Desdobramento da interação com testes de médias	34
10	Análise de experimento fatorial com um tratamento adicional	35
11	Análise de covariância	36
11.1	Análise de variância	36
11.2	Contraste entre níveis dos fatores	37
12	Experimento fatorial com fatores qualitativos e quantitativos	38
12.1	Desdobramento da interação	39
12.2	Obtenção das equações de regressão e R^2	39
13	Fatorial com fatores quantitativos - superfície de resposta	40
13.1	Análise de variância e obtenção do modelo empírico	40
13.2	Gráfico do modelo final	41
14	Análise de experimentos em parcela subdividida	41
14.1	Análise de variância	41
14.2	Teste de médias	42

15 Experimentos em parcelas subdivididas	43
15.1 Análise de variância	43
15.2 Testes de médias	43
16 Recursos gráficos	46
16.1 Gráficos do pacote <i>graphics</i>	46
16.2 Gráficos do pacote <i>lattice</i>	48

1 Introdução à manipulação de objetos e funções

1.1 Instalação do R

```
.
#-----
# página do R
browseURL(URLEncode("http://www.r-project.org/"))
#
#-----
# página de download
browseURL(URLEncode("http://cran.stat.ucla.edu/bin/windows/base/"))
#
#-----
# documento com instruções de instalação e primeiros passos
browseURL(URLEncode("http://cran.r-project.org/doc/contrib/Itano-installation.pdf"))
#
#-----
# curiosidades sobre o R
browseURL(URLEncode("http://www.nytimes.com/2009/01/07/technology/business-computing/07program.html"))
browseURL(URLEncode("http://jeromyanglim.blogspot.com/2010/05/abbreviations-of-r-commands-explained.html"))
#
#-----
# quando não souber os links consulte o google
browseURL(URLEncode("http://www.lmgtfy.com/?q=R+download+windows"))
browseURL(URLEncode("http://www.lmgtfy.com/?q=R+reference+card"))
#
#-----
.
```

1.2 Pedindo 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")
#
#-----
.
```

1.3 Criação, atribuição, acesso e modificação de objetos

```
.
#-----
# vetores, sequências e números aleatórios
c(2,4,7,3,8,9)
1:7
seq(0, 20)
seq(0, 20, length=4)
seq()
help(seq, help_type="html")
rep(1:3, times=3)
rep(1:3, each=3)
rnorm(5, 3, 2)
rnorm(5, sd=2, mean=3)
rnorm(5, mean=3, sd=2)
runif(5)
#-----
# matrizes
matrix(c(1,5,38,400), 2, 2)
matrix(1:6, 2, 3)
matrix(rnorm(9), 3, 3)
matrix(c("a","c","b","j"), 2, 2)
#-----
# 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
#-----
# 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$A
#-----#
# atribuição, acesso e modificação de "planilhas"
x <- list(A=rnorm(4),
          B=matrix(1:4,2,2),
          C=data.frame(a=1:4, b=runif(4), c=letters[1:4]))
x
x[[1]]
x[[3]][,1]
x$B
x[["C"]]
x[["C"]][1,1] <- 0
#-----#
.

```

1.4 Informações sobre objetos (atributos)

```

.
#-----#
# como obter informações sobre um objeto?
v <- c(a=1, b=2, c=3)
v
length(v)
class(v)
class("R")
names(v)
#-----#

m <- matrix(1:3,2,2)
m
dim(m)
class(m)
colnames(m)
rownames(m)
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()

```

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

1.5 Operações matemáticas

```
.  
#-----#  
# as operações fundamentais e mais  
1+100  
3-5  
2*8  
3/4  
2^3  
sqrt(4)  
exp(3)  
2.71^3  
log(10)  
log10(1000)  
log(30, base=2.2)  
#-----#  
# as operações em vetores  
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  
x-4  
x+y  
x*y  
x%*%y  
x+z  
z%*%x  
det(x)  
diag(x)  
solve(x)  
t(z)  
#-----#  
# operações trigonométricas  
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)
sum(x)
var(x)
sd(x)
median(x)
max(x)
min(x)
range(x)
summary(x)
plot(x)
hist(x)
#-----
#
x <- matrix(rnorm(20), 4, 5)
colSums(x)
rowMeans(x)
mean(x)
var(x)
cor(x)
sd(x)
apply(x, 1, var)
apply(x, 1, max)
apply(x, 2, min)
#-----
#
# operações com data.frames
x <- expand.grid(A=c("H","M"), B=c("sim","não"), rep=1:4)
x
x$alt <- rnorm(x$A, 1.7)
x
tapply(x$alt, x$A, mean)
tapply(alt, A, mean)
ls()
with(x, tapply(alt, A, mean))
with(x, tapply(alt, B, var))
with(x, tapply(alt, list(A, B), sum))
with(x, aggregate(alt, list(A, B), mean))
#-----
#
.
```

1.7 Construindo funções

```
.
#-----
# criação e uso de funções
f0 <- function(x, y){
  (x+y)^2
}
class(f0)
args(f0)
f0(3, 2)
#-----
#
# função para obtenção das raízes de uma função de 2 grau
baskara <- function(a,b,c){
```



```

x1 <- (-b-sqrt(b^2-4*a*c))/(2*a)
x2 <- (-b+sqrt(b^2-4*a*c))/(2*a)
c(x1, x2)
}
baskara(-3,2,1)
baskara(3,2,1)
curve(3*x^2+2*x+1, -1, 2)
curve(-3*x^2+2*x+1, -1, 2)
abline(h=0, v=baskara(-3,2,1), lty=2)
#-----#
# função para obtenção da nota necessária para ser aprovado na 3 prova
nota3 <- function(nota1, nota2){
  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)")
  }
}
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()
setwd("/home/walmes/Documentos/Curso R")
apropos("system")
system("ls")
#-----#
# importando dados
#soja <- read.table("http://www.leg.ufpr.br/~walmes/cursosR/soja.txt", header=TRUE, sep="\t", dec=",")
soja <- read.table("soja.txt", header=TRUE, sep="\t", dec=",")
class(soja)
names(soja)
dim(soja)
str(soja)
head(soja)
soja
#-----#
# exploração numérica
with(soja, tapply(reengrao, list(potassio), mean))
with(soja, tapply(reengrao, list(potassio, agua), mean))
#-----#
# selecionando subconjuntos dos dados
subset(soja, potassio==0)

```

```

subset(soja, bloco=="I")
subset(soja, potassio==0 & bloco=="I")
subset(soja, reengrao<15)
subset(soja, reengrao<15 & pesograo<11)
#
#-----
# um pouco sobre perguntas lógicas
1==1
2==1
1!=3
3!=3
1<2
1<1
1<=1
1<=1 & 2>1
1<=1 & 1>1
1<3 | 2<3
1<3 | 4<3
5<3 | 4<3
"joão"=="João"
"joão"=="joao"
#
#-----
.

```

2.2 Explorações gráficas (1)

```

.
#-----
# matriz de diagramas de dispersão
pairs(soja)
#
#-----
# gráficos simples de dispersão
plot(reengrao-potassio, data=subset(soja, agua==50))
plot(reengrao-potassio, data=subset(soja, agua==50),
      xlab="Dose de potássio", ylab="Rendimento de grãos",
      col=2, pch=19, cex=1.2)
#
#-----
# boxplot
boxplot(reengrao-potassio, data=subset(soja, agua==50))
boxplot(reengrao-potassio, data=soja, col="yellow")
#
#-----
# todos níveis de água ao mesmo tempo
par(mfrow=c(1,3))
plot(reengrao-potassio, data=subset(soja, agua==37.5), main="37.5% de água")
plot(reengrao-potassio, data=subset(soja, agua==50), main="50.0% de água")
plot(reengrao-potassio, data=subset(soja, agua==62.5), main="62.5% de água")
#
#-----
# gráficos de barras
par(mfrow=c(1,1))
pot.m <- with(soja, tapply(reengrao, potassio, mean))
pot.m
bp <- barplot(pot.m) # 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)
#-----
# como os dados se distribuem (marginal)?
pairs(agr)
hist(agr$roundness)
plot(agr$roundness, col=agr$prof)
plot(density(agr$roundness))
#-----
# mas as médias não mudam com a profundidade?
with(agr, tapply(roundness, profundidade, mean))
with(agr, tapply(aspecto, profundidade, mean))
#-----
# os dados têm distribuição normal? como checar?
par(mfrow=c(1,2))
qqnorm(agr$roundness); qqline(agr$roundness)
qqnorm(agr$aspecto); qqline(agr$aspecto)
with(subset(agr, profundidade==5), { qqnorm(roundness); qqline(roundness) })
with(subset(agr, profundidade==20), { qqnorm(roundness); qqline(roundness) })
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? transformar? qual transformação?
require(MASS)
agr5 <- subset(agr, profundidade==5)
boxcox(agr5$roundness~1, lambda=seq(-1,6,l=100))
qqnorm(agr5$roundness^4); qqline(agr5$roundness^4)
shapiro.test(agr5$roundness)
shapiro.test(agr5$roundness^4)
#-----
# o que fazer em casos como esse?
boxcox(agr5$aspecto~1, lambda=seq(-1,6,l=100))
qqnorm(agr5$aspecto^3); qqline(agr5$aspecto^3)
#-----
.
```

2.4 Recursos gráficos avançados

```
.
#-----
# 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 e densidade
histogram(~roundness|profundidade, data=agr)
densityplot(~roundness+aspecto|profundidade, data=agr)
#
#-----
# matriz de dispersão
str(agr)
splom(agr[,-1], group=agr$profundidade)
#
#-----
# os dados abaixo têm distribuição normal?
m <- gl(15,8)
x <- rnorm(m, as.numeric(m), 0.1)
xp <- qqnorm(x); qqline(x)
rug(xp$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)
#
#-----
# uma mistura de distribuições
par(mfrow=c(1,1))
curve(dnorm(x,0,1), 0, 20)
for(i in seq(0,20,l=7)) { curve(dnorm(x,i,1), add=TRUE, col=i); abline(v=i, col=i, lty=3) }
#
#-----
.

```

3 Estatística básica

3.1 Medidas de posição, dispersão e forma

```

.
#-----
# 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
hist(x, freq=FALSE)
curve(dnorm(x, mean(x), sd(x)), add=TRUE)
lines(density(x), col=2)
#-----#
# todas essas funções estão disponíveis no pacote fBasics
install.packages("fBasics", dependencies=TRUE)
require(fBasics)
basicStats(x)
basicStats
#-----#
# separatrizes, quantis e os 5 números de Tukey
fivenum(x)
quantile(x, c(5,50,95)/100)
#-----#
# amplitude total e interquartilica
range(x)
diff(range(x))
IQR(x)
#-----#
# box plot
boxplot(x)
#-----#
# momento naftalina: diagrama de ramos
stem(x)
#-----#
.

```

3.2 Teste de hipótese e intervalo de confiança para parâmetros

```
.
#-----
# para a média de uma população normal
t.test(x, mu=7)
#
#-----
# para uma proporção (germinação amostral)
germ <- 82.4
prop.test(germ, 100, p=0.90)
#
#-----
# 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)
#
#-----
# teste para igualdade de duas variâncias de dados normais
var.test(cA, cB)
bartlett.test()
#
#-----
# teste para a normalidade de uma amostra
shapiro.test(x)
ks.test(scale(x), "pnorm")
ks.test(x, "pnorm", mean=mean(x), sd=sd(x))
#
#-----
# teste para correlação entre duas variáveis normais
area5 <- agr$area[agr$profundidade==5]
roun5 <- agr$roundness[agr$profundidade==5]
plot(area5, roun5)
cor.test(area5, roun5)
#
#-----
.
```

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 regressoras
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)
pairs(dap)
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)
str(m0)
#
#-----
# verificando o ajuste
plot(h~d, dapcc) # xlab=, ylab=
lines(fitted(m0)~d, dapcc, col="red")
abline(m0, col=3, lty=2)
#
#-----
# análise de resíduos
par(mfrow=c(2,2))
plot(m0)
#
#-----
.

```

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)
summary(m1)
layout(matrix(c(1,1,2,3,4,5),2,3))
plot(h~d, dapcc)
lines(fitted(m1)~d, dapcc)
plot(m1)
#
#-----
# modelo cúbico
m2 <- lm(h~d+d2+d3, data=dapcc) # ou lm(h~d+I(d^2)+I(d^3), data=dapcc)
summary(m2)
plot(h~d, dapcc)
lines(fitted(m2)~d, dapcc)
plot(m2)
#
#-----
# modelo recíproco
m3 <- lm(h~d+di, data=dapcc)
summary(m3)
plot(h~d, dapcc); lines(fitted(m3)~d, dapcc); plot(m3)
#
#-----
# modelo quadrado do recíproco
m4 <- lm(h~d+di2, data=dapcc)
summary(m4)
plot(h~d, dapcc); lines(fitted(m4)~d, dapcc); plot(m4)

```

```

#-----#
# modelo raiz quadrada
m5 <- lm(h~d+dr, data=dapcc)
summary(m5)
plot(h~d, dapcc); lines(fitted(m5)~d, dapcc); plot(m5)
#-----#
# modelo logarítimo
m6 <- lm(h~d+dl, data=dapcc)
summary(m6)
plot(h~d, dapcc); lines(fitted(m6)~d, dapcc); plot(m6)
#-----#
.

```

4.4 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); plot(m7)
#-----#
# seleção de modelos/variáveis
step(m7, direction="both")
step(m7, direction="both", k=log(nrow(dapcc)))
#-----#
# modelo m5 foi escolhido pelo critério BIC
summary(m5)
anova(m5)
plot(h~d, dapcc); lines(fitted(m5)~d, dapcc); plot(m5)
#-----#
.

```

4.5 Remoção de pontos discrepantes

```

.
#-----#
# identificar/remover os pontos discrepantes/influentes
layout(1)
plot(residuals(m5)~d, dapcc)
id <- identify(dapcc$d, residuals(m5))
id
#-----#
# análise com os pontos removidos
dapcc2 <- dapcc[-c(15,41,209),]
str(dapcc2)
dapcc2 <- dapcc[-id,]
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); plot(m5b)
#-----#

```



```

#-----
# e se tentarmos tranformar?
require(MASS)
layout(1)
bc <- boxcox(m5b, lambda=seq(0.5,2,l=100))
bc
str(bc)
bc$x[which.max(bc$y)]
#

#-----
# usando a resposta transformada
m5c <- lm(h^(1.2)~d+dr, data=dapcc2)
summary(m5c)
layout(matrix(c(1,1,2,3,4,5),2,3))
plot(h~d, dapcc2); lines(fitted(m5c)^(1/1.2)~d, dapcc2); plot(m5c)
shapiro.test(rstudent(m5c))
ks.test(rstudent(m5c), "pnorm")
shapiro.test(rstudent(m5))
ks.test(rstudent(m5), "pnorm")
#

#-----
.

```

4.6 Predição de valores a partir do modelo escolhido

```

.
#-----
# tudo para encontrar o modelo, vamos preder a artura das arvores 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")
summary(m5b)
text(20, 15, expression(hat(h)==-16.8-1.12*d+14.8*sqrt(d)~~~(R^2==84.6)), 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)
#-----
#
browseURL(URLEncode("http://animation.yihui.name/lm:least_squares"))
#-----
#
.

```

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 contrói 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 fertilizante)
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$x, da$y)
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)
  }
}
#-----#
# 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  $h = b_0 \cdot (1 - \exp(b_1 \cdot d))^{b_2}$ 
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)
}

```

```

source("func.R")
#
#-----
# 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)
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)
#
#-----
# fazemos 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)
confint(n0)
#
#-----
# o intervalo de confiança 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 eclodidos em função dos dias e dose de nematocida
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")

```

```

#-----#
# 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 nematocidas (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")
       )
#-----#
.

```

6 Análise de experimento com um fator em DIC

6.1 Importando dados

```

.
#-----#
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"
diam <- read.table(textConnection(Lines), header=TRUE); closeAllConnections()

```

```

str(diam)
#-----#
# 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(diam$gen)
is.numeric(diam$gen)
is.character(diam$gen)
class(diam$gen)
#-----#
.

```

6.2 Análise de variância

```

.
#-----#
# fazendo a análise de variância
a0 <- aov(diam~gen, data=diam)
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(residuals(a0)~diam$gen)
#-----#
.

```

6.3 Aplicando teste de Tukey para comparar médias

```

.
#-----#
# teste de médias (Tukey), ingredientes: 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
require(agricolae)
#-----#
# aplicando o teste de Tukey
with(diam,

```

```

HSD.test(diam, gen,
         DFerror=df.residual(a0),
         MSerror=deviance(a0)/df.residual(a0), alpha=0.05)
)
#-----#
.

```

6.4 Aplicando teste de Scott-Knott para agrupar médias

```

.
#-----#
# ScottKnott
require(ScottKnott)
#-----#
# aplicando o teste de ScottKnott
sk <- SK(x=diam, y=diam$diam, model="y~gen", which="gen", sig.level=0.05)
summary(sk)
#-----#
.

```

6.5 Aplicando contrastes

```

.
#-----#
# biblioteca para fazer contrastes; o modelo precisa ser classe "lm" e não "aov"
require(contrast)
a0 <- lm(diam~gen, data=diam)
class(a0)
#-----#
# um nível contra o outro
c0 <- contrast(a0, list(gen="ATF06B"), list(gen="ATF40B"))
c0
levels(diam$gen)
coef(a0)
c0$X
summary(a0)
#-----#
# um grupo de níveis contra outro
c1 <- contrast(a0, type="average", list(gen=c("BR001B", "ATF06B", "BR008B", "SC283")))
c2 <- contrast(a0, type="average", list(gen=c("ATF40B", "BR007B", "P9401", "ATF54B", "BR005B")))
c1
c2
c1$X-c2$X
(c1$X-c2$X)%*%coef(a0)
#-----#
# biblioteca para coduzir comparações multiplas de hipóteses
require(multcomp)
summary(glht(a0, linfct=(c1$X-c2$X)))
#-----#
.

```

6.6 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

```

.
#-----
m0 <- aov(prod~bloco+proced, data=dbc)
class(m0)
anova(m0)
summary(m0)
summary.lm(m0)
#-----
# checagem gráfica
par(mfrow=c(2,2))
plot(m0)
layout(1)
#-----
# teste das pressuposições de normalidade de homocedasticidade
shapiro.test(residuals(m0))
bartlett.test(residuals(m0)~dbc$proced)
bartlett.test(residuals(m0)~dbc$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 Observações perdidas

```

.
#-----
# simulando observações perdidas aleatoriamente no conjunto dados
id <- sample(1:nrow(dbc), 3)
id

```

```

dbc$prod[id] <- NA
dbc
#-----#
m1 <- aov(prod~bloco+proced, data=dbc)
summary(m1)
#-----#
# o que acontece com os estimadores amostrais das médias?
dbc$ajus <- predict(m1, newdata=dbc)
with(dbc, tapply(prod, proced, mean, na.rm=TRUE))
#-----#
# o que acontece com os estimadores de mínimos quadrados das médias?
with(dbc, tapply(ajus, proced, mean))
#-----#
# como comparar médias? Tukey usando a média harmônica do número de repetições (danger)
dbc.cc <- dbc[complete.cases(dbc),]
with(dbc.cc,
      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)
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)]
comp <- do.call(rbind, strsplit(comp, "-"))
comp
#-----#
# montando a matriz de contrastes
cX <- sapply(1:nrow(comp),
             function(i){
               c.contr <- contrast(m1, type="average",
                                   list(proced=comp[i,1], bloco=levels(dbc$bloco)),
                                   list(proced=comp[i,2], bloco=levels(dbc$bloco)))
               c.contr$X
             })
cX
#-----#
# fornecendo a matriz para a glht para manutenção do erro tipo I
compP <- glht(m1, linfct=t(cX))
summary(compP)
#-----#
# as médias marginais populacionais
do.call(c, sapply(levels(dbc$proced),
                  function(i){
                    contrast(m1, type="average", list(proced=i, bloco=levels(dbc$bloco)))[1]
                  }))
#-----#
.

```

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$dose)
#
#-----
# precisa-se de transformação para normalidade e homocedasticidade
require(MASS)
boxcox(m0)
#
#-----
# usando a transformação indicada
m1 <- aov(volu^0.33~gen*dose, data=vol)
par(mfrow=c(2,2))
plot(m1)
layout(1)
shapiro.test(residuals(m1))
bartlett.test(residuals(m1)~vol$dose)
bartlett.test(residuals(m1)~vol$gen)
summary(m1)
```



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

8.2 Testes de médias

```
.
#-----#
# teste de Tukey para gen (com dados transformados)
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 SkottKnott (com dados transformados)
sk <- SK(x=vol, y=volu$volu^0.33, model="y-gen*dose", which="gen")
sk <- summary(sk)
sk$Means <- sk$Means^(1/0.33)
sk
#-----#
.
```

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 expressões regulares
words <- c("0","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)
)))
#

.
```

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 contrastes 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)))

#-----#
.

```

11 Análise de covariância

11.1 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)
#-----
# análise de variância (em experimentos não ortogonais a ordem dos termos é importante!)
m0 <- aov(peso28~sexo*energia, data=ac)
summary(m0)
m1 <- aov(peso28~pi+id+sexo*energia, data=ac)
summary(m1)
m1 <- aov(peso28~id+pi+sexo*energia, data=ac)
summary(m1)
anova(m0, m1)
#-----
#-----
m2 <- aov(peso28~energia*sexo+pi+id, data=ac)
summary(m2)
m2 <- aov(peso28~sexo*energia+pi+id, data=ac)
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$pi)
mean(ac$id)
#-----
#-----
# para fazer os contrastes (objeto deve ser da classe lm)
require(contrast)
levels(ac$sexo)
levels(ac$energia)
m0 <- lm(peso28~pi+id+sexo*energia, data=ac)
anova(m0)
par(mfrow=c(2,2))
plot(m0)
layout(1)
#-----
#-----
.

```

11.2 Contraste entre níveis dos fatores

```

.
#-----
# fema vs macho castrado
contrast(m0, type="average",
         list(sexo="F", energia=c("baixo","medio","alto"), pi=92, id=138),
         list(sexo="MC", energia=levels(ac$energia), pi=92, id=138))
#-----
#-----
# fema vs macho imunocastrado
contrast(m0, type="average",
         list(sexo="F", energia=levels(ac$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
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)
box()
#-----
.

```

12 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)
#-----
.

```

12.1 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)
)))
#-----
.
```

12.2 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 Fatorial com fatores quantitativos - superfície de resposta

13.1 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)
#-----#
.

```

13.2 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 estimativas são as mesmas)
m4 <- lm(ts~A+K+A:K+I(A^2)+I(K^2), data=rend)
summary(m4)
p0 <- expand.grid(A=seq(35,65,l=80), K=seq(0,200,l=80))
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))
#-----
#-----
.
```

14 Análise de experimentos em parcela subdividida

14.1 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)
m0 <- aov(alt~BL+AD*ES+Error(BL:AD), data=ps)
summary(m0)
#-----
# checagem
class(m0)
m1 <- lm(alt~BL+AD*ES, data=ps)
par(mfrow=c(2,2))
```

```

plot(m1)
shapiro.test(residuals(m1))
#-----
#

```

14.2 Teste de médias

```

#-----
# dedobrar ES em cada nível de AD
require(agricolae)
df.residual(m0)
deviance(m0)
#-----
#
# 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]
qmS
#-----
#
# teste de Tukey
lapply(levels(ps$AD),
  function(a){
    with(subset(ps, AD==a),
      HSD.test(alt, ES, DFerror=glS, MSerror=qmS))
  })
#-----
#
# teste de ScottKnott
lapply(1:3,
  function(a){
    sk <- SK.nest(x=ps, y=ps$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 fornecido em casos de parcela subdividida
  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)
satter(A=c(qmP, glP, 3), B=c(qmS, glS, 2))
lapply(levels(ps$ES),
  function(a){
    with(subset(ps, ES==a),
      HSD.test(alt, AD, DFerror=7.43, MSerror=29.83))
  })

```

```

    })
#-----#
# 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 Experimentos em parcelas subdivididas

15.1 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)
str(pss)
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*dnpk, data=pss)
par(mfrow=c(2,2))
plot(m1)
layout(1)
#-----#
require(MASS)
boxcox(m1)
m2 <- aov(log(AF)~bloco+fonte*dorg*dnpk, data=pss)
par(mfrow=c(2,2))
plot(m2)
#-----#
# análise de variância com dados transformados
m0 <- aov(log(AF)~bloco+fonte*dorg*dnpk+Error(bloco:fonte/dorg), data=pss)
summary(m0)
#-----#
.

```

15.2 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){
        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){
        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){
        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$AF),
          model="y~bloco+dnpk*fonte*dorg+Error(bloco:fonte/dorg)",
          which="dnpk:fonte:dorg", error="Within",
          fl2=f, fl3=0)
        tes <- summary(tes)
        tes$Means <- exp(tes$Means)
        tes
      })
    })
#-----#
# desdobrar dorg em fonte com dnkp
tes <- SK.nest(x=pss, y=log(pss$AF),
  model="y~bloco+dorg*fonte*dnkp+Error(bloco:fonte/dorg)",
  which="dorg:fonte:dnkp", 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$AF),
          model="y~bloco+dorg*fonte*dnkp+Error(bloco:fonte/dorg)",
          which="dorg:fonte:dnkp", error="bloco:fonte:dorg",
          fl2=f, fl3=npk)
        tes <- summary(tes)
        tes$Means <- exp(tes$Means)
        tes
      })
    })
#-----#
# desdobrar fonte em dorg com dnkp
tes <- SK.nest(x=pss, y=log(pss$AF),
  model="y~bloco+fonte*dorg*dnkp+Error(bloco:fonte/dorg)",
  which="fonte:dorg:dnkp", 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$AF),
          model="y~bloco+fonte*dorg*dnkp+Error(bloco:fonte/dorg)",
          which="fonte:dorg:dnkp", error="bloco:fonte",
          fl2=o, fl3=npk)
        tes <- summary(tes)
        tes$Means <- exp(tes$Means)
        tes
      })
    })
#-----#
.

```

16 Recursos gráficos

16.1 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")
#-----#
.

```

16.2 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)
#
#-----
.
```
