INTRODUÇÃO AO SISTEMA LIVRE R EM ANÁLISES ESTATÍSTCAS

Emanuel Fernando Maia de Souza
Luiz Alexandre Peternelli

Introdução

Atualmente o uso de pacotes estatísticos para a análise de dados é de grande importância no que se refere à análise e a interpretação de resultados. Contudo observa-se que estes apresentam um custo de aquisição relativamente elevado. Atualmente é grande a procura, bem como o incentivo ao uso dos chamados softwares livres. Dentre os softwares de domínio público, livres, que podem ser utilizados para análise de dados em geral, encontra-se o Ambiente R, ou simplesmente R, conforme usualmente chamado pelos seus usuários, que além de ser gratuito, apresenta código fonte aberto, podendo ser modificado ou implementado com novos procedimentos desenvolvidos pelo usuário a qualquer momento. O R torna-se, portanto uma importante ferramenta na análise e manipulação de dados, com testes paramétricos e não paramétricos, modelagem linear e não linear, análise de séries temporais, análise de sobrevivência, simulação e estatística espacial, além de apresentar facilidade na elaboração de diversos tipos de gráficos, dentre outras. Pode ser obtido em http://cran.r-project.org, onde é apresentado em versões de acordo com o sistema operacional UNIX, Windows ou Macintosh. Além disso, encontra-se neste site mais informação sobre a sua utilização e uma central de correspondências onde profissionais de vários paises podem contribuir na implementação de novos recursos. Como o R é uma linguagem de programação orientada a objetos o usuário pode criar suas próprias funções, e sua própria rotina na análise de dados. Outro atributo do R é sua capacidade de interagir com outros programas estatísticos, bem como de banco de dados.

O programa

Como já foi dito o R é uma linguagem orientada a objetos criada em 1996 por Ross Ihaka e Robert Gentleman que aliada a um ambiente integrado permite manipulação de dados, realização de cálculos e geração de gráficos, semelhante à linguagem S desenvolvida pela AT&T's Bell Laboratories e que já é utilizada para análise de dados (veja, por exemplo, Venable e Ripley, 1999), mas com a vantagem de ser de livre distribuição.

É importante salientar que o R não é um programa estatístico, mas que devido a suas rotinas permite a manipulação, avaliação e interpretação de procedimentos estatísticos aplicado a dados. O *R Core Team* ("defensores e detentores" do R o classificam como Ambiente R dado a suas características, nós entretanto abordamos como um sistema integrado que permite a execução de nossas tarefas em estatística).

Além dos procedimentos estatísticos o R permite operações matemáticas simples, e manipulação de vetores e matrizes. Assim como confecção de diversos tipos de gráficos.

Como instalar.

Para a instalação do R basta estar conectado a internet conectar-se ao site http://cran.r-project.org , em CRAN "Comprehensive R Archive Network" mais

próximo, no caso de Viçosa: http://www.termix.ufv.br/CRAN . Dar um clique duplo no ícone de acordo com o sistema operacional do seu computador, posteriormente no ícone *base*, e depois no arquivo executável.

E seguir a rotina de instalação.

Após instalado deve-se iniciar o R e clicar na barra de ferramentas em:
Packages → UPDATE PACKEGES FROM CRAN
Para receber as versões atualizadas dos principais pacotes.

Uma primeira sessão

Com o R iniciado você verá o símbolo' > ' em vermelho, que é o *prompt* do R indicando que o sistema está pronto para receber seus comandos.

Acima do *prompt* em cor azul encontram-se algumas informações sobre o sistema e alguns comandos básicos.

O R pode ser trabalhado como uma calculadora, sendo útil para realização dos ais diversos cálculos e operações com matrizes.

Para se trabalhar com o R são necessários alguns conceitos. Dentre os mais importantes para desenvolvimento de funções mais elaboradas pelo usuário. Como foi dito anteriormente o R é uma linguagem orientada a objetos (variáveis, dados , matrizes, arrays, data-frames, listas, funções etc) que são armazenados na memória ativa do computador.
Exemplo:

```
x<-sqrt(4) #objeto x irá receber o valor da operação indicada
x
```

Objetos

Vetores

```
x <- c(2,3,5,7,11)      # os 5 primeiros números primos
x

y <- c(x,13,17,19)        # adicionando  mais três números primos
```

Sequências

```
1:10

xx <- 100:1     # seqüência decrescente de 100 a 1
xx

seq(1,10,1)              # o mesmo que 1:10

seq(1,10,2)              # de 2 em 2
seq(10,1,3)              # tentando ordem inversa...

seq(10,1,-3)             # a forma correta é usando passo negativo...

rep(1,10)
rep(c(1,2),10)
c(rep(0,10),rep(1,5))

paste("tr",1:5,sep="")
```

Listas

```
person <- list(age=21,name='Fred',score=c(65,78,55))
person

person$name          # nome

person$score[2]   # segundo elemento de $score

tt <- t.test(rnorm(1000,mean=1),rnorm(1000,mean=1.2),var.equal=T)

is.list(tt)

names(tt)
```

Matrizes

Criando Matrizes

```
x <- 1:12
xmat <- matrix(x,ncol=3)

matrix(x,ncol=3,byrow=T)
```

Informações sobre a matriz

```
x1 <- matrix(1:12,ncol=4)

x <- matrix(10:1,ncol=2)

y <- cbind(x,1:5)

y <- rbind(y,c(99,99,99))   # adicionando uma nova linha...

z <- cbind(y,rep(88,6),y)
```

Índices das matrizes

```
z[2,5]
z[,4]                # extraindo a quarta coluna
z[3,]                # extraindo a terceira linha

z[c(1,3,5),]    # extraindo três colunas

z[,5:7]   # extraindo três colunas...

 z[c(2,3),c(4,6)]       # tomando uma sub-matrix 2x2...
```

Adicionar nomes as linhas e colunas matrizes

```
colnames(y)<-c("C1","C2","C3")  #  se  for  colunas,  para  linhas  use
rownames
```

Data Frames

```
L3 <- LETTERS[1:3]
d <- data.frame(cbind(x=1, y=1:10), fac=sample(L3, 10, repl=TRUE))
```

```
teste<- read.table("a:\dataframe.txt",h=T)

teste[,3]
teste$fac
```

Entrada de dados

```
teste<- read.table("a:\dataframe.txt",h=T)
```

Criação de vetores, matrizes, data frames, listas etc
Exemplo: excel
B&K.dados.p179.xls
Usar a função scan() e read.table()

Salvar o ler arquivos *.R

Para salvar o arquivo de trabalho, deve-se clicar na barra de ferramentas em:
File -> Save Workspace... -> escolher o local para salvar e nomear o arquivo.
Para ler um arquivo deve-se:
File -> Load Workspace... -> escolher o local e o arquivo desejado.

A utilização desta operação quando se trabalha com arquivos no R é a possibilidade de ter armazenado todos os objetos criados em análises, facilitando assim o trabalho de revisão.

Operações aritméticas

Você pode utilizar o R como uma calculadora:

```
 2+3  # somando estes números
2+3*2  # observando prioridades
2**4  # potências utilizado ** ou ^

sqrt(9)  # raiz quadrada
sin(3.14159)   # seno de Pi radianos é zero
sin(pi)   # bem mais próximo.
factorial(4) # 4!
```

Algumas funções disponíveis do R

Função	Significado
log(x)	Log de base e de x
exp(x)	Antilog de x (e^x)
log(x,n)	Log de base n de x
Log10(x)	Log de base 10 de x
sqrt(x)	Raiz quadrada de x
choose(n,x)	n!/(x!(n-z)!)
cos(x), sin(x), tan(x)	Funções trigonométricas de x em radianos
acos(x), asin(x), atan(x)	Funções trig. Inversas de x em radianos

Existem inúmeras funções no R que estão disponíveis em manuais de indrodução.

Operações com vetores e matrizes

Veja o exemplo onde já estou criando diretamente o vetor no R:

```
coluna1<-c(2,1,0)
coluna2<-c(1,3,1)
coluna3<-c(1,1,2)

A<-cbind(coluna1,coluna2,coluna3)
A
t(A)
```

Dificultando um pouco

```
t(A)%*%A    # A'A (A transposta "vezes" A)

coluna1%*%coluna1  # uma multiplicação de vetores

coluna1*coluna1
```

Finalmente, a inversa de uma matriz não singular quadrada:
Exemplo:

```
solve(A)   # inversa da matriz A
```

Ficou com duvidas:

```
 solve(A)%*%A  # verificação

 round(solve(A)%*%A,5)
```

Outro exemplo: inversa da matriz A'A:

```
solve(t(A)%*%A)
```

Operações lógicas e outras funções

Símbolo	Significado
!=	Diferente
%%	Modulo
%/%	Divisão inteira
+, - , *, /	Adição, subtração, multiplicação, divisão
** ou ^	Potência
<, >	Menor, maior que
<=, >=	Menor ou igual, maior ou igual que
= =	Igual
seq(5,-5,-1)	Seqüência de 5 a -5 de -1 em -1.
rep(2,10)	Repetição de 2 dez vezes
max(), min(), range()	Maximo, mínimo e amplitude
sum(x)	Soma total de x
mean(x), var(x)	Média aritmética , variância amostral de x
cor(x,y)	Correlação entre os vetores x e y
median(x)	Mediana de x
oder(x)	Vetor contendo as posições ordenadas crecentes de x
sort(x)	Versão ordenada de x
rank(x)	Retorna vetor com a classificação crescente de x
ColSums(A)	Retorna a soma das colunas da matriz A

Por exemplo, vamos criar um vetor com uma seqüência de 0 a 10:

```
x<-0:10
```

```
sum(x)    # soma desta seqüência
sum(x<5)    # uma forma de descobrir quantos valores são menores que 5
sum(x[x<5])   # soma dos valores menores que 5
x<5    # obtendo a reposta lógica de x<5
```

Você pode imaginar se os valores falsos fosse zero e os valores verdadeiros fossem um. Multiplicando valores lógicos por valores numéricos obtem-se valores numéricos.

```
1*(x<5)
```

Agora imagine a multiplicação dos valores do vetor x pelos valores lógicos do valor x na condição discriminada:

```
x*(x<5)
```

Quando aplicamos a função sum(x), temos a respostas da soma dos valores dos números 0+1+2+3+4+5 = 10.

```
sum(x*(x<5))
```

Agora você percebe a diferença entre utilizar parênteses ou colchetes.

Para saber qual é a posição de um valor ou de determinados valores pode-se utilizar:

```
which(x<=5)
```

Além destas opções pode-se contar com os operadores de conjuntos como: intersect(x,y), union(x,y), setdiff(x,y), setequal(x,y), unique(x,y). Para saber mais consulte a ajuda do R.

Criando funções

Vamos criar inicialmente uma função que realize a média aritmética. Que é a seguinte função

```
med<-function(x)
{
Soma<-sum(x)
N<-length(x)       # "tamanho de x"
Soma/N
}

x<-0:10
med(x)
```

Agora vamos melhorar nossa função através do comando edit()

```
medv2<-edit(med)
```

Ao aparecer a tela do bloco de notas é só deixar como o que está abaixo:

```
function(x) sum(x)/length(x)
```

Salvar e fechar o arquivo.

Função para plotar símbolos

```
simbol<-function()
{
k<- -1
plot(c(0,1),c(0,1),lab=c(0,0,0),xlab="",ylab="",type="n")
      for(i in 1:9)
       {
            for(j in 1:9)
            {
            k<- k+1
            points(j/10,i/10,pch=k)
            }
      }
```

```
}
simbol()
```

Manipulação inicial dos dados

Para se ter uma abordagem estatística adequada é importante que o interessado "namore" seu dados. E uma forma inicial seria uma abordagem gráfica.

Gráficos

Um gráfico simples

```
x <- 1:20
 y <- x**3
 plot(x,y)
plot(x,y,type="l")
```

Há várias outras opções para os gráficos. Consulte ?plot
Você gostaria de mudar os pontos

```
plot(x,y)
 points(rev(x),y,pch=3)    # adiciona cruzes
 points(x,8000-y,pch="$")   # usando o cifrão
```

Linhas ?

```
plot(x,y)
lines(x,y,lwd=4)         # linha grossa
lines(rev(x),y,lty=2)    # linha interrompida
```

Cores e textos ?

```
plot(x,y,xlab="Eixo X aqui",ylab="Eixo Y aqui",col="blue")
 title("Título vai aqui \n (e note a acentuação!!!)")
 text(6,4000,"Texto em qualquer lugar")
```

Mais sobre gráficos

```
demo(graphics)
```

Uso do R como sistema estatístico

Estatística descritiva

max(x), min(x),mean(x), median(x), quantile(x), var(x),sd(), cov(x,y), cor(x,y) calculam respectivamente máximo, mínimo, média, mediana, quantis, variância e desvio padrão amostrais de x; covariância e correlação entre x e y.

Probabilidades

Algumas distribuições existentes no R são:
beta (beta), binomial (binom), cauchy (cauchy), Qui-quadrado (chisq), exponencial (exp), F (f), Gamma (gamma), geométrica (geom), hypergeométrica (hyper), log-normal (lnorm), logistica (logis), binomial negativa (nbinom), normal (norm), Poisson (pois), t de Student (t), uniforme (unif), Weibull (weibull), soma de ranks de Wilcoxon (wilcox).
Para conhecer os parâmetros necessários em cada distribuição, use na linha de comando, por exemplo, ?rbinom. Aparecerá um menu de ajuda completo.

```
x<-seq(-3,3)
curve(dnorm(x),-3,3)
curve(dnorm(x,),-5,5,col="red")
```

```
text(-2,0.3,expression(list(bar(x)==0, sigma==1)),col="red")
curve(dnorm(x,0,2),-5,5,add=T)
text(2,0.15,expression(list(bar(x)==0, sigma==2)))
curve(dnorm(x,2,1),-5,5,add=T,col=”blue”)
text(3,0.4,expression(list(bar(x)==2, sigma==1)),col="blue")

x<-rbinom(100,10,.4)
hist(x,probability=T,col=gray(.95), main="Distribuição Binomial",
sub="100 valores aleatórios")
vax<-0:10
points(vax,dbinom(vax,10,.4),type="h",lwd=3,col=”red”)
```

Testes de comparações

Teste F

Máquina A	145	127	136	142	141	137
Máquina B	143	128	132	138	142	132

```
ma<- c (145,127,136,142,141,137)
na<- length(ma)

mb<-c(143,128,132,138,142,132)
nb<-length(mb)
```

Usando o R como uma calculadora
Já que iremos usar o teste F, temos:

$$Fcal = \frac{\sigma_{\rangle}^2}{\sigma_{\langle}^2}$$

```
vma<-var(ma)

vmb<-var(mb)

fcal<- vma/vmb
fcal   # observando o valor da estatística do teste

pval <- pf(fcal,na-1,nb-1, lower=F) # calculando o valor o P-valor.
pval
```

Mais fácil

```
var.test(ma,mb,alternative = c("greater"))
```

Teste t.

```
x<-c(30.5,35.3,33.2,40.8,42.3,41.5,36.3,43.2,34.6,38.5)
y<-c(28.2,35.1,33.2,35.6,40.2,37.4,34.2,42.1,30.5,38.4)

t.test(x, mu=35, alternative=c(“greater"))
t.test(x,y, conf.level = 0.99) # supondo α=1%
t.test(x, y, alternative=c(“less"),paired=T)
```

Outros testes ver a biblioteca Classical Tests

```
help.search("Ctest")
```

Análise de variância

Fórmulas de modelos:

Modelo	Fórmula
DIC	y ~ t onde t é uma variável categórica
DBC	y ~ t + b
DQL	y ~ t + l + c
Fatorial/ DIC	y ~ N*P igual a N + P + N:P
Fatorial/ DBC	y ~ b + N*P igual a b + N + P + N:P
Regressão linear simples	y ~ x onde x é uma variável exploratória
Regressão quadrática	y ~ x + x2 onde x2 é um objeto x2<-x^2

lm() para regressão linear (linear models)
aov() para ANOVA, com erros NID.
glm() para ANOVA, com estrutura de erros especificada. (generalised linear models)
lme() para modelos mistos
nls() para modelos não lineares.

DIC

Entrada dos dados da resposta do experimento.

```
res<-scan()
25 31 22 33 26 25 26 29 20 28 28 31 23 27 25 34 21 24 29 28
```

Criando os nomes dos tratamentos na ordem correspondente.

```
trat<-factor(rep(paste("tr",1:4,sep=""),5))
```

Fazendo a ANOVA

```
dic.aov<-aov(res~trat)
```

Quadro da ANOVA

```
anova(dic.aov)

summary(dic.aov) # para ver outros detalhes
plot(dic.aov)
```

Testes para comparações múltiplas

Há vários testes de comparações múltiplas disponíveis na literatura, e muitos deles disponíveis no R, e os que não estão são um convite para os novos usuários a estarem implementando com os recursos do R.

Vejamos duas formas de se usar o teste de Tukey, a primeira usando a função TukeyHSD e a segunda fazendo os cálculos necessários com o R.

Primeira:

```
expl.tk<-TukeyHSD(dic.aov,"trat")
plot(expl.tk)
expl.tk
```

Segunda

Trabalha a partir da resolução mais conhecida pelo usuário, com certas adaptações para a operação via computador.

Obter o QMRes

```
s2<- sum(resid(dic.aov)^2)/dic.aov$df.res #conferir com a Anova
```

Valor tabelado

```
dt <- qtukey(0.95,3,18)*sqrt(s2/5)

tra.m<-tapply(res,trat,mean)


md1<-outer(tra.m,tra.m,"-")   # matriz que terá as diferenças entre as
médias dos tratamentos
md1
```

Só a triangular inferior

```
md1<- md1[lower.tri(md1)]
md1

m1n<-outer(names(tra.m),names(tra.m),paste, sep="-")
m1n

names(md1)<-m1n[lower.tri(m1n)]
md1
```

Agora a mágica!!

```
dic.tkv2<-data.frame(dif = md1, sig = ifelse(abs(md1)>dt,"*","ns"))
dic.tkv2
```

Bem mais parecido com o conhecido, não é? Agora basta ordenar as médias e acrescentar as letras comparativas de praxe. Desenvolver uma função para resolver em qualquer caso fica como um desafio.

DBC

```
DBC<-read.table("A/DBC.prn",h=T)

attach(DBC)
```

OBS: ao finalizar a análise devemos dar o comando `detach(DBC)`

```
saida<-aov(y~trat+rep,DBC) # temos um modelo em DBC

anova(saida)

saida<-aov(y~factor(trat)+factor(rep),DBC) # temos um modelo em DBC
```

Para obter os resíduos podemos fazer

```
res<-resid(saida)
tapply(y,trat,mean)
tapply(y,rep,sum)
```

IMPORTANTE

Para alterar certos valores no conjunto de DBC originais, podemos fazer

```
DBC2<-edit(DBC)
```

ANOVA de experimentos com dois fatores

Para realizar a ANOVA de um fatorial

Exemplo do SAEG

```
fat<-read.table("a:/fatsubd.txt",header=T)

attach(fat)
fat.aov<-aov(PROD~BLOCO+VAR*SEM,data=fat)
anova(fat.aov)

bloco<-factor(BLOCO)
var<-factor(VAR)
sem<-factor(SEM)

fat1.aov<-aov(PROD~bloco+var*sem)
anova(fat1.aov)

interaction.plot(sem,var,PROD)
detach(fat)
```

Realizando a anova para parcela sub-dividida.
Supondo dois fatores A e B e no DBC. tomando como exemplo o exercício da página 179 do Banzatto e Kronka. Veja arquivo anexo B&K.dados.p179.txt

```
dados<-read.table("a:/B&K.dados.p179.txt",h=T)
attach(dados)
Af<-factor(A)
Bf<-factor(B)
blocof<-factor(bloco)

saida<-aov(prod~blocof+Af+Bf+Af:Bf+Error(blocof/Af),data=dados)

summary(saida)
```

Modelos mistos

```
library(nlme)

data(Orthodont)
Orthodont   # para visualizar os dados.

fm1 <- lme(distance ~ age, data = Orthodont) # Quando o efeito
aleatório é age
fm1

anova(fm1)

VarCorr(fm1)

ranef(fm1)    # para efeitos aleatórios
fixef(fm1)     # para efeitos fixos
```

```
AIC(fm1)
```

Regressão

```
x<-c(10,15,20,25,30)
y<-c(1003,1005,1010,1011,1014)
```

Fazendo a regressão linear simples.

```
reg1<-lm(y~x)
predict(reg1) #obtendo os valores estimados

coefficients(reg1) #obtendo as estimativas dos coeficientes

summary(reg1) #obtendo alguns resultados úteis: teste t e ANOVA
anova(reg1)
```

Para construir o gráfico

```
plot(x,y) #construindo o diagrama de dispersão

abline(reg1) #para obter a reta
pred<-predict(reg1) #usando os valores estimados
for(i in 1:5) lines(c(x[i],x[i]),c(y[i],pred[i])) # para caprichar
```

Polinomial

Exemplo:

```
x<-scan()
1 2.5 4 5.5 7 8.5

y<-scan()
20.3 31.3 34.6 35.1 30.2 19.7

reg2<-lm(y~poly(x,2))

anova(reg2) # visualizar o quadro da anova

reg3<-lm(y~x+I(x^2))

anova(reg3)

coefficients(reg3)   #ou

summary(reg3)
```

Para plotar o gráfico

```
plot(x,y,ylim=c(20,37))
curve((11.242222+10.467698*x-1.113492*x*x),1,8.5,add=T)
pred<-predict(reg3)
for(i in 1:6) lines(c(x[i],x[i]),c(y[i],pred[i]))
```

Para obter ajuda nessa função, por exemplo, use

```
?curve
```

Teste para normalidade

Fazendo os testes de Kolmogorov-Smirnov e de Shapiro-Wilk para normalidade Será usado o exemplo apresentado na página 564 do livro do Steel, Torrie e Dickey (1997)

```
y<-c(73.9,74.2,74.6,74.7,75.4,76,76,76,76.5,76.6,76.9,77.3,77.4,77.7)

ks.test(y,"pnorm",mean(y),sd(y))


shapiro.test(y)

qqnorm(y) #Obtendo o normal probability plot só para comparação

qqline(y) #colocando uma linha auxiliar
```

Não paramétrica

Teste de Wilcoxon

```
? wilcox.test
```

Teste de Kruskal-Wallis

```
data(airquality)
 boxplot(Ozone ~ Month, data = airquality)
 kruskal.test(Ozone ~ Month, data = airquality)
```