Programação linear
Levantamentos com R
T. Praciano-Pereira

Lista numero 03
tarcisio@member.ams.org
Dep. de Computação

alun@:

29 de outubro de 2012 **Univ. Estadual Vale do Acaraú**

Documento escrito com LaTeX - sis. op. Debian/Gnu/Linux
http://www.otimizacao.sobralmatematica.org

Se entregar em papel, por favor, prenda esta *folha de rosto* na sua solução desta lista, deixando-a em branco. Ela será usada na correção.

Exercícios 1 *Fazendo levantamentos com R objetivo: Aprender a usar algumas funções básicas de R muito poderosas para fazer levantamentos estatísticos. O foco é aprender a usar R.*

palavras chave:

??keyword	*apresenta informação sobre "keyword"*
c()	*concatena objetos*
demo(pacote)	*mostra o uso de pacote, ver também help()*
dir()	*lista o conteúdo do diretório corrente, ver getdwd()*
getwd()	*mostra qual é o diretório corrente, ver setwd()*
help(pacote)	*um auxílio sobre uso de pacote*
length(x)	*mostra o tamanho do vetor x*
ls()	*produz uma listagem dos símbolos*
mean(x)	*calcula o valor médio do vetor x*
objects()	*semelhante a ls()*
rm()	*apaga símbolos da mémoria de* R
rnorm(n)	*cria vetor, n entradas, aleatoriamente*
sample(x)	*seleção aleatoria de uma matriz x*
sd(x)	*standard deviation - desvio padrão de x*
setwd()	*muda diretório*
sink()	*rediciona a saída de dados*
source()	*define origem de input*
sum()	*soma as entrada de uma matriz, ver help(sum)*

fonte:

`http://www.ats.ucla.edu/STAT/r/notes/getting_started.htm`

1. *Amostragem*

   (a) *(V)[ ](F)[ ] A sequência de comandos*

   ```
   n <- 100000; x <- c(0,1); simu <- sample(x, size=n, replace=TRUE);
   ```

   cria um vetor com 100000 coordenadas.

   (b) *(V)[ ](F)[ ] A sequência de comandos*

   ```
   n <- 100000; x <- c(0,1,2,3,4,5); simu <- sample(x, size=30, replace=FALSE);
   ```

   cria uma amostra de elementos tirados de $\{0,1,2,3,4,5\}$.

   (c) *(V)[ ](F)[ ] A sequência de comandos*

   ```
   x <- c(0,1,2,3,4,5); simu <- sample(x, size=30, replace=FALSE);
   ```

   produz um erro porque não é posivel criar uma amostragem de elementos tirados de $\{0,1,2,3,4,5\}$ *com tamanho 30 e sem repetições.*

   (d) *(V)[ ](F)[ ] A sequência de comandos*

   ```
   x <- c(0,1,2,3,4,5); simu <- sample(x, size=30, replace=TRUE);
   ```

   cria uma amostragem de elementos tirados de $\{0,1,2,3,4,5\}$ *com tamanho 30 e com repetições.*

   (e) *(V)[ ](F)[ ] A sequência de comandos*

   ```
   x <- c("a","b","c","d","e","f"); simu <- sample(x, size=30, replace=TRUE);
   ```

   cria uma amostragem de elementos tirados de $\{"a","b","c","d","e","f"\}$ *com tamanho 30 e com repetições.*

2. *Cálculos com R*

   (a) *(V)[ ](F)[ ] Os comandos*

   ```
   x = rnorm(100); y = rnorm(100); plot(x,y)
   ```

   ativam a janela gráfica do R *mostrando um conjunto de pontos sselecinados de forma aleatória no plano.*

   (b) *(V)[ ](F)[ ] Os comandos*

   ```
   x = rnorm(100); y = rnorm(100); plot(x,y)
   ```

   se executados duas vezes, sucessivamente, mostram diferentes conjuntos de pontos e isto mostra que `rnorm(n)` *constrói uma sucessão aleatória de n pontos.*

   (c) *(V)[ ](F)[ ] é possível calcular a soma de uma amostra aleatória de n números com dois comandos sucessivos de* R*, usando de forma adequada:*

   ```
   x = rnorm(n); sum(x);
   ```

   e o resultado da soma pode ser visto com `print S`

(d) (V)[ ](F)[ ] *é possível calcular a soma de uma amostra aleatória de n números com dois comandos sucessivos de* R*, com a sequência de comandos:*

```
x = rnorm(n); S = sum(x); print(S)
```

o resultado da soma ficando registrada no objeto S *que pode ser exibido com o método* `print()`.

(e) (V)[ ](F)[ ] *com o comando* `mean(x)`*, se anteriormente tiver sido criado um vetor de números associado ao símbolo* x*, é possível calcular o valor médio de* x.

3. *Desvio padrão e média*

O `desvio padrão` *é uma* `medida de dispersão`*, para o seu cálculo primeiro precisamos saber o valor médio de uma amostra para depois calcular o* afastamento *que esta amostra tem de sua média. O método* `sd()` *do* R *faz tudo isto.*

(a) (V)[ ](F)[ ] *Se executarmos* `D = sd(x)` *teremos guardado no símbolo* D *o desvio padrão da amostra* x.

(b) (V)[ ](F)[ ] *A sequência de comandos*

```
x = rnorm(100); D = sd(x); print(D)
```

exibe o desvio padrão e o comando `plot(x,D)` *exibe o gráfico de* x *e da média* D.

(c) (V)[ ](F)[ ] *Podemos substituir todos os valores de um vetor* x *pelo seu valor médio, com dois comandos de* R

```
x = rnorm(100); M = mean(x); y = x;
y[y<=M] <- M; y[y>=M] <- M; plot(x,y);
```

(d) (V)[ ](F)[ ] *A linguagem* R *tem o* objeto vetor *como um* objeto básico *da linguagem, quer dizer, grande parte dos métodos da linguagem foram produzidos para serem aplicados em vetores.*

(e) (V)[ ](F)[ ] *Os comandos*

```
x =  sample(c(0,1,2,3,4,5,6,7,8,9), 100, replace = TRUE);
y = x; y[y>5] <- 5; plot(x,y);
```

produzem o gráfico dos valores de x *que sejam menores do que 5.*

4. *Comunicação com a shell*

(a) (V)[ ](F)[ ] *O comando*

```
getwd()
```

exibe o diretório corrente.

(b) (V)[ ](F)[ ] *O comando*


```
setwd(/home/usuario/)
```

altera o ponteiro de diretório do R *para o diretório básico de* `usuario`*. O caminho deve ser* absoluto*!*

(c) (V)[ ](F)[ ] *O comando* `ls()` *faz uma listagem dos arquivos no diretório corrente.*

(d) (V)[ ](F)[ ] *O comando* `ls()` *faz uma listagem dos símbolos ativos na memória de* R.

(e) (V)[ ](F)[ ] *O comando* `dir()` *faz uma listagem dos arquivos no diretório corrente.*

5. *Funções de auxílio do* R *definindo uma função*

(a) (V)[ ](F)[ ] *Num terminal do* R *executando* `demo()` *faz com que* R *mostre uma lista de pacotes* `pkg`*, e executando* `demo(pkg)`*, com um dos elementos desta lista, você pode avaliar a capacidade de* R.

(b) (V)[ ](F)[ ] *Executando num terminal do* R `help(simbolo)` *produz algumas páginas de informação sobre* `simbolo` *terminando com exemplos.*

(c) (V)[ ](F)[ ] *Executando num terminal do* R `help(demo)` *você pode obter mais informações sobre o pacote* `demo`.

(d) (V)[ ](F)[ ] *A expressão*

```
f <- function(x){ 4 + x*(5 + x*(3 + x))}
```

associa ao símbolo `f` *um objeto do* `tipo função` *correspondente a um polinômio do terceiro grau com termo constante* 4.

(e) (V)[ ](F)[ ] *A sequência de comandos*

```
x <- x = runif(500, min=-15, max=15);
x <- sort(x);  ## ver help(sort)
y <- f(x);  ## aplica uma função, ver help(apply)
```

cria um vetor de números racionais `(float)` *com 500 coordenadas, com valores no intervalo* $[-15, 15]$*, distribuição uniforme, guarda em* y *as imagens de* x *pela função* f *defina no item 5d. Finalmente é produzido o gráfico de* $(x, f(x))$.

6. *Matrizes e sistemas de equações*

(a) (V)[ ](F)[ ] *Com os seguintes comandos podemos resolver um sistema de equações* 10×10 *fornecendo as entradas das matriz* A *pelo teclado e definindo* B *como um vetor-coluna aleatório com dimensão 10 de dados para o sistema.*

```
> A = matrix(c(1,2,3,4,5,6,7,8,9,10,
+              0,-1,0,0,0,0,0,0,0,0 ,
+              0, 0,2,0,0,0,0,0,0,0,
+              0, 0,3,0,0,0,0,0,0,0,
+              0, 0,0,4,0,0,0,0,0,0,
+              0, 0,0,0,5,0,0,0,0,0,
+              0, 0,0,0,0,6,0,0,0,0,
+              0, 0,0,0,0,0,7,0,0,0,
+              0,0,0,0,0, 0,0,1,0,0,
+              0, 0,0,0,0,0,0,0,-1,0,
+              0, 0,0,0,0,0,0,0,0,1),  nrow = 10, ncol=10)
B = matrix(rnorm(10, mean = 5, sd = 1),  nrow = 10, ncol=1);
x = solve(A,B)
```

(b) (V)[ ](F)[ ] *Com os seguintes comandos podemos resolver um sistema de equações* 10×10 *fornecendo as entradas das matriz A pelo teclado e definindo B como um vetor-coluna aleatório com dimensão 10 de dados para o sistema.*

```
A = matrix(c(1,2,3,4,5,6,7,8,9,10,
+            0,-1,0,0,0,0,0,0,0,0 ,
+            0, 0,2,0,0,0,0,0,0,0,
+            0, 0,0,3,0,0,0,0,0,0,
+            0, 0,0,0,4,0,0,0,0,0,
+            0, 0,0,0,0,5,0,0,0,0,
+            0, 0,0,0,0,0,6,0,0,0,
+            0, 0,0,0,0,0,0,7,0,0,
+            0,0,0,0,0, 0,0,0,1,0,
+            0, 0,0,0,0,0,0,0,0,-1,  nrow = 10, ncol=10)
B = matrix(rnorm(10, mean = 5, sd = 1),  nrow = 10, ncol=1);
x = solve(A,B)
```

e como x é um resultado aletório, é interessante ter um sumário dos seus dados o que pode ser obtido com:

```
summary(x)
```

(c) (V)[ ](F)[ ] *Antes de resolver um sistema de equações podemos verificar se ele tem solução única calculando* $det(A)$

(d) (V)[ ](F)[ ] *Antes de resolver um sistema de equações podemos verificar se ele tem solução única calculando* $det(A)$ *que se for zero sugere indicarmos um parâmetro para alter a rotina de* `solve`*, como* `QR`*,* `help(solve)` *o que pode bem ser o caso com*

```
A = matrix(rnorm(400, mean=0, sd = 0.2), nrow=20, ncol =20)
```

a probabilidade de que `det(A)` *seja zero é grande.*

(e) (V)[ ](F)[ ] *Um comando pode ser de dois tipos,* uma expressão *ou* uma atribuição*, quando for uma expressão* `R` *a avalia e imprime o resultado, entretanto, economicamente, não imprime o resultado de* uma atribuição*. No caso*

```
A = matrix(rnorm(400, mean=0, sd = 0.2), nrow=20, ncol =20)
```

`R` *não produz nenhuma saída de dados, se você quiser ver a matriz* `A` *é preciso executar*

```
A = matrix(rnorm(400, mean=0, sd = 0.2), nrow=20, ncol =20);
A;
```

porque agora `A` *é uma expressão da linguagem.*

`R` *é uma* expression language *e isto significa que um* `programa` *é uma concatenação de expressões cujos átomos são* `funções da linguagem R` *ou símbolos definidos pelo usuário obedecendo a sintaxe das* expression languages*.*

fonte: *Este texto se encontra no link "textos" da página.*

```
 An Introduction to R
 Notes on R: A Programming Environment for Data Analysis
and Graphics
                                Version 2.15.1
(2012-06-22)
W. N. Venables, D. M. Smith
and the R Core Team
```

7. *Edição e sessão*

É possível guardar *ou* recuperar *os dados de uma sessão de* `R`*, verifique como com* `help(save)` *e observe que é preciso identificar o objeto* `arquivo` *com um símbolo definido usando* `file` *e que os símbolos que identicam objetos devem seguir as regras para os antigos nomes de variáveis.*

(a) (V)[ ](F)[ ] *Se você tiver guardado* `sessao` *usando* `save(file = "sessao")` *será possível ler com um editor de textos os dados da sessão.*

(b) (V)[ ](F)[ ] *Se você tiver guardado* `sessao` *usando* `save(file = "sessao", ascii=TRUE)` *será possível ler com um editor de textos os dados da sessão.*

(c) (V)[ ](F)[ ] *Você pode acessar* expressões *anteriores usando a tecla* $\uparrow$ *e pode saltar entre os átomos da* expressão *usando CTR_L* $\leftarrow$ *ou CTR_L* $\rightarrow$*.*

(d) (V)[ ](F)[ ] *Ao editar uma* expressão*, você pode mover para o final ou o começo da* expressão *usando as teclas* `home, end` *do teclado.*

(e) (V)[ ](F)[ ] *Se no arquivo* `mat.txt` *estiver*

```
A = matrix(c(1,2,3,4,5,6,7,8,9,10,
             0,-1,0,0,0,0,0,0,0,0 ,
             0, 0,2,0,0,0,0,0,0,0,
             0, 0,3,0,0,0,0,0,0,0,
             0, 0,0,4,0,0,0,0,0,0,
             0, 0,0,0,5,0,0,0,0,0,
             0, 0,0,0,0,6,0,0,0,0,
             0, 0,0,0,0,0,7,0,0,0,
             0,0,0,0,0, 0,0,1,0,0,
             0, 0,0,0,0,0,0,0,-1,0,  nrow = 10, ncol=10)
```

a expresão `source(file="exercicios/Levantamento/mat.txt")` *irá associar ao símbolo* `A` *a matriz* 10×10 *que está gravada no arquivo* `mat.txt`. *Diferentes expressões podem ser assim gravadas em arquivos e lidas posteriormente numa sessão do* `R`. *Se estiver*

```
matrix(c(1,2,3,4,5,6,7,8,9,10,
             0,-1,0,0,0,0,0,0,0,0 ,
             0, 0,2,0,0,0,0,0,0,0,
             0, 0,3,0,0,0,0,0,0,0,
             0, 0,0,4,0,0,0,0,0,0,
             0, 0,0,0,5,0,0,0,0,0,
             0, 0,0,0,0,6,0,0,0,0,
             0, 0,0,0,0,0,7,0,0,0,
             0,0,0,0,0, 0,0,1,0,0,
             0, 0,0,0,0,0,0,0,-1,0,  nrow = 10, ncol=10)
```

nada irá acontecer (visível) ao executar

```
source(file="exercicios/Levantamento/mat.txt")
```

porque o resultado se perderá. Observe que o caminho *deve ser* absoluto *a partir do valor guardado por* `setwd()` *no símbolo* `DIR`.

8. `sink()`

Guardando e recuperando dados

```
source(file="exercicios/Levantamento/mat.txt")
sink(file="exercicios/Levantamento/mat.txt")
```

(a) (V)[ ](F)[ ] *A expressão* `sink(file=exercicios/Levantamento/mat.txt)` *permite escrever resultado no arquivo* `exercicios/Levantamento/mat.txt`.

(b) (V)[ ](F)[ ] *A expressão* `sink(file="exercicios/Levantamento/mat.txt")` *rediciona a saída de dados para o arquivo* `exercicios/Levantamento/mat.txt` *de formas que todas as saídas de dados a partir deste momento irão para o arquivo mencionado.*

(c) (V)[ ](F)[ ] *A expressão* `system("comando_da_shell")` *executa um comando da shell do UNIX, por exemplo, se* `joe` *for o nome de um editor de textos,* `system("joe exercicios/Levantamento/mat.txt")` *lhe permitira editar os dados enviados para o arquivo mencionado ao executar* `sink(file="exercicios/Levantamento/mat.txt")`

(d) (V)[ ](F)[ ] *A expressão*

```
A = matrix(rnorm(900, mean=0, sd = 0.1), nrow=30, ncol =30)
```

depois de `sink(file="exercicios/Levantamento/mat.txt")` *permite criar uma matriz com grande número de entradas que pode depois ser editada de forma conveniente usando*

```
system("joe exercicios/Levantamento/mat.txt")
```

em que `joe` *é o nome de um editor de textos do sistema.*

(e) (V)[ ](F)[ ] *É relativamente fácil adquirir dados usando as funções* `sink(), source(), system()`;

9. *O* `espaço de nomes`

Três funções servem para gerenciar o `espaço de nomes` *de* `R`

```
objects()
ls()
rm()
```

Analise a sintaxe destas funções com `help`.

(a) (V)[ ](F)[ ] *As funções* `objects(), ls(), dir()` *têm o mesmo efeito.*

(b) (V)[ ](F)[ ] *As funções* `objects(), ls()` *têm o mesmo efeito, ao passo que* `dir()` *faz uma listagem dos arquivos do diretório apontado por* `setwd()`.

(c) (V)[ ](F)[ ] *Asignar um valor é uma forma de criar novas expressões isto é possível com*

```
A <- expressão;
expressão -> A;
assign("A", expressão)
assign("x", c(10.4, 5.6, 3.1, 6.4, 21.7))
```

no último caso criamos um vetor numérico com 5 coordenadas associado ao símbolo `x`.

(d) (V)[ ](F)[ ] *A função* `c()` *concatena objetos mas se não houver* assignação *o resultado se perde.*

(e) (V)[ ](F)[ ] *As expressões*

```
x <- c(1,2,3,4,5);
y <- c(-x,0,x);
y <- sort(y);
```

produzem um vetor com 11 coordenadas ordenadas, aritmeticamente.

10. *Aritmética sobre vetores*

`R` *aplica as funções predefinidas ou aquelas que a usuária tiver definido às coordenadas de um vetor, porisso o dado básico da linguagem* `R` *é vetor.*

   (a) *(V)[ ](F)[ ] Se x for um vetor numérico e f for a função polinomial definida na questão 5d então $f(x)$ produz um erro.*

   (b) *(V)[ ](F)[ ] Se* `A` *for uma matriz como definida na questão 7e e f for a função polinomial definida na questão 5d então* `B <- f(A)` *produz uma nova matriz gravada no símbolo* `B` *com a mesma dimensão de* `A`.

   (c) *(V)[ ](F)[ ] E possível ver a dimensão de um objeto com a função* `length()`.

   (d) *(V)[ ](F)[ ] Se* `length(x) = 10` *e* `length(y) = 10` *então*

   ```
   v <- 2*x + y + 1
   ```

   `length(v) = 10`.

   (e) *(V)[ ](F)[ ] Se* `length(x) = 10` *e* `length(y) = 10` *então*

   ```
   v <- 2*x + y + 1
   ```

   produz um erro.