UNIVERSIDADE ESTADUAL PAULISTA

"Júlio de Mesquita Filho"

Introdução ao Ambiente Computacional R

Paula da Silva Carvalho – Engenharia Ambiental

Monitora de Probabilidade e Estatística – 2006

Profa. Dra. Ma. Da Conceição F. Freitas Tandel

ORIENTADORA: UNESP-IGCE-DEMAC

SUMÁRIO

1. HISTÓRICO E APRESENTAÇÃO DO AMBIENTE COMPUTACIONAL R

O “R” é uma linguagem e ambiente para computação estatística e elaboração de gráficos. Consiste de uma linguagem orientada a objetos mais um ambiente gráfico, disponível como uma biblioteca compartilhada, Pode processar programas armazenados em arquivos. O R pode ser compilado e roda em um grande número de plataformas: Unix , Linux, Macintosh, Windows, etc. É possível a interface com procedimentos escritos em C, C++, ou FORTRAN. Possui um grande número de procedimentos estatísticos convencionais. Entre eles modelos lineares, modelos lineares generalizados, modelos de regressão não linear, análises de séries temporais, testes estatísticos clássicos paramétricos e não paramétricos, métodos da estatística multivariada como análise de cluster, componentes principais, análise fatorial, etc... Possui uma grande quantidade de funções para desenvolvimento de ambiente gráfico e criação de diversos tipos de apresentação de dados. Possui módulos adicionais ou pacotes (“add-on packages”) disponíveis para uma variedade de propósitos específicos. (R Add-On Packages). Os pacotes são complementos ao R. Há 3 tipos de pacotes:

- *Pacotes recomendados* ("recommended packages"): estes pacotes são distribuídos e instalados junto com o R. Entretanto eles não estão imediatamente disponíveis quando voce inicia uma sessão do R. Para utilizar a funcionalidade destes pacotes é necessário *carregar* o pacote.

Por exemplo, para carregar o pacote chamado MASS voce deve digitar: require(MASS)

- *Pacotes contribuídos (oficiais)*: estes pacotes estão disponíveis para download no site do R porém não estão incluídos no programa de instalação do R. Para utilizar estes pacotes voce deve:

a) copiar o pacote,
b) carregar o pacote com o comando require() como acima.

- *Pacotes contribuídos não oficiais*

  São disponibilizados pelos usuários de forma geral em listas de discussões e por contatos pessoais.

Há facilidades para copiar o pacote de dentro de uma sessão do R. Voce pode usar as funções install.packages() e update.packages() para instalar e atualizar pacotes, respectivamente. Além disto, na versão para Windows há atalhos na barra de ferramentas. Há atualmente mais de cem pacotes contribuídos para o R. Para ver s lista de pacotes vá até sessão de pacotes contribuídos ou consulte a documentação do programa.

Nota: o comando library() também pode ser usado para carregar os pacotes no R. Entretanto o comando require() é preferido uma vez que somente carrega o pacote caso ainda não tenha sido carregado, evitando duplicação das funções existentes.

O R é um sistema computacional relativamente novo, início da década de 1990. A primeira versão não-beta, ou seja, já testada e aprovada, foi lançada ao público em Fevereiro de 2000. Tem sido muito utilizado no meio científico devido à modernidade das técnicas implementadas. Possui atualizações diárias e cerca de 2 versões oficialmente disponíveis ao ano, para atualizações completas. Sua divulgação tem tido muito sucesso em Universidades Públicas Brasileiras e vem sendo utilizado como ambiente computacional para o desenvolvimento de projetos de pesquisas de relevância.

Sua importância fica acentuada em virtude de ser um sistema de DOMÍNIO PÚBLICO. Portanto é de acesso livre, ou seja gratuito e de código fonte aberto, ou seja, pode-se acessar as linhas de comandos de cada programa. É disponibilizado sobre os termos da GNU General Public License da FSE: Free Software Foundation , www.gnu.org/ . GNU é o nome da primeira comunidade de compartilhamento de softwares livres, cujo principal patrocinador é a fundação FSE: Free Software Foundation, fundada em 1985. O sítio oficial é http://www.r-project.org e a área para download e espelhos é http://cran.r-project.org. CRAN: Comprehensive R Archive Network, é uma coleção de sites de distribuição do R e do material de documentação. O espelho brasileiro, de onde os arquivos de instalação podem ser copiados, esta disponível no endereço:

http://cran.br.R-project.org/ (Universidade Federal de Paraná, Brazil)

R foi inicialmente escrito por Ross Ihaka e Robert Gentleman no Departamento de Estatística da Universidade de Auckland em Auckland, New Zealand. O sistema R é mantido atualmente, como um projeto colaborativo, com muitos grupos de contribuidores (Development Core Team), formados por pesquisadores de renome internacional, ligados a área acadêmica, em diversos países, inclusive o Brasil. O nome "R" esta baseado na letra inicial dos dois primeiros autores, Robert Gentleman and Ross Ihaka e faz uma referência indireta ao nome da Bell Labs language "S" , versão comercial muito similar ao R (veja What is S?). A versão atual do R (31/08/2006) é a versão 2.3.1.

Para fazer a citação do R em trabalhos científicos siga as instruções abaixo ou com o programa aberto digite o comando **'citation()'.**

Título: R: A Language and Environment for Statistical Computing
Autor: R Development Core Team
Organização: R Foundation for Statistical Computing
Endereço: Viena, Áustria
Ano: 2006
ISBN: 3-900051-07-0
URL: http://www.R-project.org

2. INTRODUÇÃO AO AMBIENTE R

Comandos de ajuda do R:

- help.start() inicia documentação na forma de arquivos html visualizados no browser.
- help (tópico) inicia uma janela de ajuda sobre tópico.

Inicie o R e defina o diretório de trabalho clicando em:

Arquivo → mudar diretório

O programa será inicializado mostrando a seguinte mensagem:

```
R : Copyright 2006, The R Foundation for Statistical Computing
Version 2.3.0 (2006-04-24) ISBN 3-900051-07-0
R é um software livre e vem sem GARANTIA ALGUMA.
Você pode redistribuí-lo sob certas circunstâncias.
Digite 'license()' ou 'licence()' para detalhes de distribuição.
R é um projeto colaborativo com muitos contribuidores.
Digite 'contributors()' para obter mais informações e
'citation()' para saber como citar o R ou pacotes do R em
publicações.
Digite 'demo()' para demonstrações, 'help()' para o sistema on-line
de ajuda,
ou 'help.start()' para abrir o sistema de ajuda em HTML no seu
navegador.
Digite 'q()' para sair do R.
>
```

O símbolo > indica a linha de comando (“prompt”) na qual serão digitados os comando para execução das análises.

2.1 Armazenamento de Área de Trabalho e linhas de comando

Entende-se por área de trabalho, todos os objetos criados numa seção, como por exemplo, variáveis, vetores, funções, matrizes,etc...(excetua-se gráficos)

Há duas maneiras de salvar a área de trabalho.

a) para recarregar automaticamente na futura seção do R: é necessário salvar através da opção sair do menu principal ou do atalho “x”, caso contrario ele recarregará a última área salva desta maneira e perderá a área trabalhada não salva.

b) Salvar em arquivo para recarregar via o comando do menu principal **“arquivo → “carregar área de trabalho”**: É necessário salvar via o comando do menu principal “arquivo -> salvar área de trabalho” e especificar o nome desejado.

Para salvar o arquivo de comandos utilizados na seção em funcionamento do R, seleciona opção do menu principal **“arquivo → salvar histórico**”. Para

acessar em outra seção é necessário recarregar o mesmo histórico através do comando **"arquivo carregar histórico"**. Desta maneira acessa-se os comandos um a um, teclando a tecla "seta para cima" sucessivas vezes. Ou abre-se o histórico salvo num editor de texto, o que permite visualizar o programa completo, facilitando a edição.

3. OPERAÇÕES ARITMÉTICAS

Você pode usar o R para avaliar algumas operações aritméticas simples. Por exemplo:

```
> 1+2+3  # somando estes números
[1] 6
> 2+3*4 # prioridade de operações(multiplicação primeiro)
[1] 14
> 3/2   # assim como a divisão
[1] 1.5
> 4*3**3  # potências são indicadas por ** ou ^
[1] 108
```

Aqui está uma lista de algumas das funções aritméticas no R:

NOME	**OPERAÇÃO**
sqrt	***Raiz quadrada***
abs	***Valor absoluto***
sin, cós, tan	***Funções trigonométricas***
asin, acos, atan	***Funções trigonométricas inversas***
sinh, cosh, tanh	***Funções hiperbólicas***
asinh, acosh, atanh	***Funções hiperbólicas inversas***
exp, log	***Exponencial e logaritmo natural***
log10	***Logaritmo – base 10***
Integrate(f,a,b)	***Integral de f nos limites 'a' e 'b'***

As expressões podem ser agrupadas e combinadas em expressões mais complexas:

```
> sqrt(45*pi/180)
[1] 0.886227
```

Pode-se criar funções específicas de acordo com a necessidade do usuário. Exemplos:

```
> funcao<-function(x){3*x^2}
> funcao(4)


> conc<-function(x){x^2}
> final<-function(x,y){integrate(conc,x,y)}
```

```
> final(0,1)
```

4. OBJETOS

R é uma linguagem orientada a objetos: variáveis, dados, matrizes, funções,etc são armazenados na memória ativa do computador na forma de objetos.

Você pode armazenar um valor em um objeto com certo nome usando o símbolo <- ou ->

```
> x <-sqrt(2)
> x
[1] 1.414214
> y <-sqrt(5)  # outra variável
> y+x          # soma das variáveis
[1] 3.650282
```

Há uma função para listar todos os objetos criados na seção: `ls()`

Há uma função para remover objetos: `remove ()`. Para usar esta função basta fornecer o objeto a ser removido:

```
> remove(x)
```

O nome das variáveis deve começar com uma letra e podem conter letras, números e pontos. Maiúsculas e minúsculas são consideradas diferentes.

Dica: Tente atribuir nomes que tenham um significado lógico. Isto facilita lidar com um grande número de objetos.

Os tipos de objetos do R são:

- Vetores
- Matrizes
- Data-frames
- Listas
- Arrays
- Funções

4.1 Vetores

Os vetores armazenam mais de um valor. È usada à função `c()` para criar um vetor a partir de seus argumentos. Por exemplo:

```
> x <- c(2,3,4,5)  # são esses os valores do vetor
> x
[1] 2 3 4 5
> y <- c(x,6,7,8)
> y
[1] 2 3 4 5 6 7 8
```

```
> x[1]  # aparece apenas o valor correspondente à posição 1
[1] 2
> x[2]
[1] 3
> z <- 1:10
> z
 [1]  1  2  3  4  5  6  7  8  9 10
> w <- seq(0,1, by=0.1)
> w
 [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
```

4.2 Matrizes

Há várias formas de criar uma matriz. A função **`matrix()`** recebe um vetor como argumento e o transforma em uma matriz de acordo com as dimensões especificadas:

```
> m1<- 1:12
> x<-1:12
> x
 [1]  1  2  3  4  5  6  7  8  9 10 11 12
> matx <- matrix (x,ncol=3)
> matx
     [,1] [,2] [,3]
[1,]    1    5    9
[2,]    2    6   10
[3,]    3    7   11
[4,]    4    8   12
```

Note que a matriz foi preenchida ao longo das linhas. A função **`cbind()`** também pode ser usada para a construção de matrizes.

```
> m2 <- cbind(1:5, 6:10)
> m2
     [,1] [,2]
[1,]    1    6
[2,]    2    7
[3,]    3    8
[4,]    4    9
[5,]    5   10
```

4.3 Data-frames

Data-frames é uma estrutura dom linhas e colunas, semelhante a uma planilha eletrônica, portanto tem duas dimensões. Entretanto, diferentemente de matrizes, cada coluna pode armazenar elementos de diferentes tipos, desde que seja mantido o tipo dentro da mesma coluna. Por exemplo: a primeira coluna pode ser numérica enquanto a segunda pode ser constituída de caracteres alfa-numéricos. Observe que neste formato, a primeira coluna sempre é um identificador dos dados, ex. 1,2,...,n).

```
> d1 <- data.frame (X= 1:10, Y=c(51,54,61,67,68,75,77,75,80,82))
> d1
    X  Y
1   1 51
2   2 54
3   3 61
4   4 67
5   5 68
6   6 75
7   7 77
8   8 75
9   9 80
10 10 82
```

O comando `edit(data.frame())` abre uma planilha para a digitação de dados que são armazenados como data.frame.

```
> planilha <- edit(data.frame())
```

4.4 Listas

É O FORMATO MAIS GERAL DO R. Listas são estruturas genéricas e flexíveis que permitem armazenar diversos formatos em um único objeto. Pode ter tamanho diferentes de linhas e colunas, contendo tipos diferentes de dados.

```
> list1 <- list(A=1:10, B="this is a message", C=matrix(1:9,
ncol=3))
> list1
$A
 [1]  1  2  3  4  5  6  7  8  9 10

$B
[1] "this is a message"

$C
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
```

4.5 Arrays

O conceito de array generaliza a idéia de matriz, no sentido de que numa matriz tem-se 2 dimensões e num array tem-se um número arbitrário de dimensões.

```
varray1 <-array(1:24,dim=c(3,4,2))
```

4.6 Funções

Os conteúdos das funções (ex. lm, plot) podem ser vistos digitando o nome das funções sem os (). Se isto não acontecer é porque não foi escrita na linguagem R , em geral quando não o são, são escritas em C(ex. min, max, rnorm). Neste caso é necessário examinar o código fonte do R para vizualizar o conteúdo da função.

As funções do R são documentadas e pode-se acessar a ajuda digitando **help ("nome da função").**

5. GERADOR DE NÚMEROS ALEATÓRIOS

O R pode gerar números aleatórios (pseudo-aleatórios - estritamente falando) de um grande número de distribuições: uniforme, normal, binomial, poisson, gamma, chi-quadrado, etc.

Os nomes das funções para gerar números aleatórios iniciam-se com a letra r e são bastante intuitivos: **runif, rnorm, rbinom, rpois, rgamma, rchisq** etc.

Junto a cada função geradora de cada distribuição há funções para calcular probabilidade, densidade e quantis. Por exemplo, para distribuição normal temos:

```
rnorm(n,mean=0,sd=1) Gera n números
dnorm(q,mean=0,sd=1) Densidade para o q-ésimo quantil
pnorm(q,mean=0,sd=1) Probabilidade para o q-ésimo quantil
qnorm(p,mean=0,sd=1) Quantil para a probabilidade p
```

```
> rnorm(5)
[1]  1.36765720  1.48876652  0.22262049  0.21998024 -0.05312521
> rnorm(10)
 [1]  -0.3790560    0.3323459    0.1792362  -0.3124366    0.8829086
1.4238435 -0.1085266 -0.8900406 -0.2273283  0.0793036
```

6. GERADOR DE AMOSTRAS ALEATÓRIAS

```
x<-1:41
sample(x,5)
vetor<-sample(x,5)
nomes<-c("Maria", "Fábio", "João", "Antonio", "Ricardo", "Ângela",
"Carla", "José", "Luiz", "Tiago", "Tamires", "Tatiana", "Fátima",
"Ana", "Tânia", "Roque", "Rogério", "Soraia", "Patrícia",
```

```
"Gilberto", "Sílvio", "Carlos", "Danilo", "Milton", "Adolfo",
"Amilton", "Janete", "Marlene", "Solange", "Renata")
sample(nomes,5)
vetor<-sample(nomes,5)
```

7. ENTRANDO COM DADOS

Pode-se entrar com dados no R de diferentes formas, o formato mais adequado vai depender do tamanho do conjunto de dados, e se os dados já existem em outro formato para serem importados ou se serão diretamente no R.

7.1 Função Scan

Esta função coloca o R em modo *prompt onde* o usuário deve digitar cada dado seguido da tecla ENTER. Para encerrar a entrada de dados basta digitar ENTER duas vezes.

```
> y <- scan()
1: 20
2: 22
3: 21
4: 22
5: 19
6: 20
7: 22
 # leu 7 itens
> y
[1] 20 22 21 22 19 20 22
```

7.2 Função Edit

O comando `edit(data.frame())` abre uma planilha para a digitação de dados que são armazenados como data-frame. Data-frames são análogos no R a uma planilha.

Portanto digitando:

```
> planilha <- edit(data.frame())
```

7.3 Arquivo texto

Se os dados já estão disponíveis em formato eletrônico, isto é, já foram digitados em outro programa, você pode importar os dados para o R sem a necessidade de digita-los novamente.

A forma mais fácil de fazer isto é usar um dado em formato texto (arquivo do tipo ASCII). Por exemplo, se seus dados estão disponíveis em uma planilha eletrônica como EXCEL ou similar, você pode na planilha escolher a opção SALVAR COMO e gravar os dados em um arquivo em formato texto.

No R usa-se a função **`read.table`** para ler os dados de um arquivo texto e armazenar no formato de data-frame. Neste formato o arquivo deve possuir na primeira coluna a identificação de cada linha do arquivo, exemplo: 1,2,3,4,

8. ANÁLISE DESCRITIVA UNIVARIADA

Nesta sessão vamos ver alguns comandos do R para fazer uma análise univariada de um conjunto de dados. Para isso usamos um conjunto de dados de precipitação mensal da cidade de Rio Claro entre os ano de 1936 e 2004.

```
>precipitação<-read.table("preciprclaro.txt",row.names=1,quote="",
head=T)
> precipitação
```

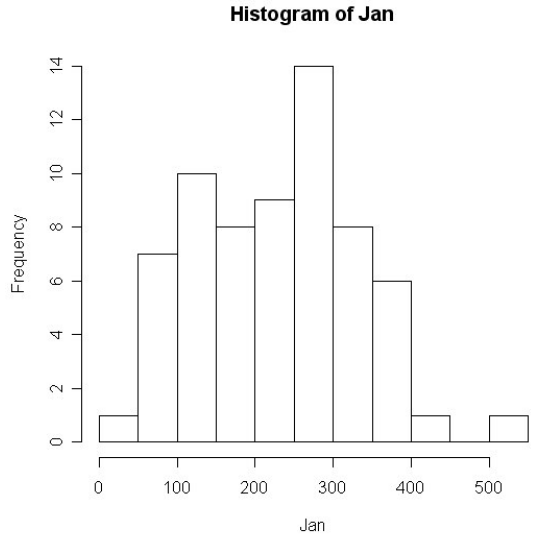
Para podermos trabalhar com a tabela, usando isoladamente as suas variáveis precisamos deixá-la no caminho de procura do sistema. Para isso usamos a seguinte função:

```
> attach(precipitação)

# detach(precipitação) para retirar o objeto precipitação do
# caminho de procura
```

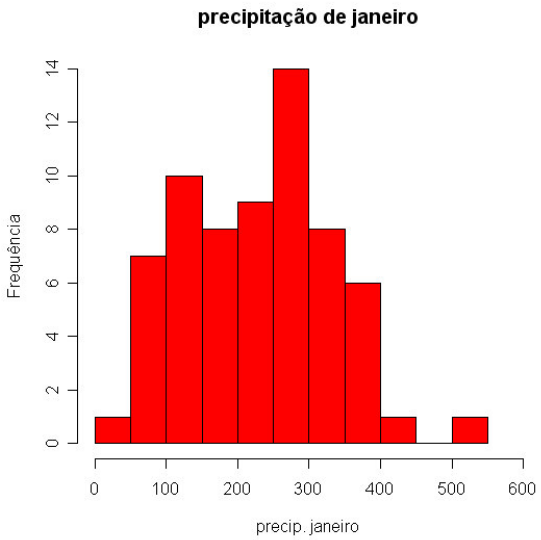
Após isso podemos fazer os diferentes tipos de gráficos que o sistema R disponibiliza para cada uma das variáveis.

- Histograma

```
> hist(Jan)
```

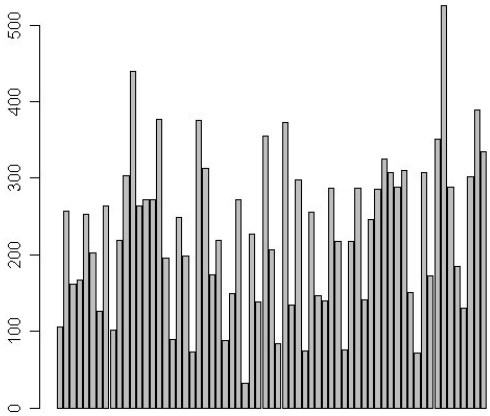


```
>hist(Jan,breaks=10,xlim=(range(0,600)),col=2,main="precipitação de janeiro", xlab="precip. janeiro",ylab="Frequência")
```



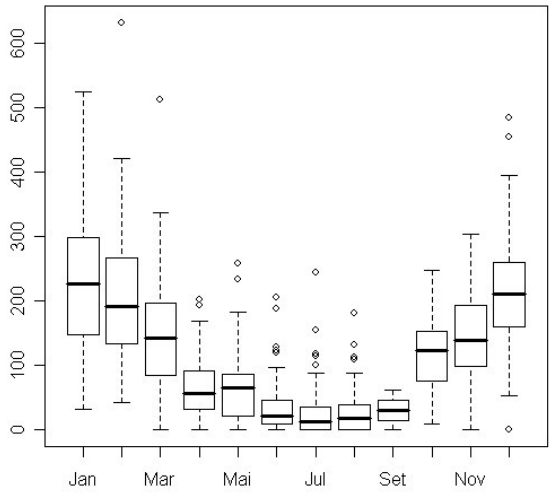
- Gráfico de barras

```
> barplot(Jan)
```



- Boxplot

```
> boxplot(precipitacao)
```



Para se ter a MÉDIA, MEDIANA, PRIMEIRO QUARTIL, TERCEIRO QUARTIL, MÁXIMO e MÍNIMO usamos o **summary**:

```
> summary(Jan)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   32.0   147.5   226.8   228.4   298.5   524.9
> summary(precipitação)
```

Ao invés de usarmos **summary(),** podemos usar isoladamente as funções de média, máximo e mínimo usando **mean(), max()e min()** respectivamente.

Para achar o desvio padrão usamos a seguinte função:

```
> sd (Jan)
[1] 101.6433
```

Para achar a variância usamos a função **var()**:

```
> var(Jan)
[1] 10331.36
```

9. ANÁLISE DESCRITIVA BIVARIADA

Agora veremos alguns comandos do R para fazer uma análise bivariada de um conjunto de dados. Utilizaremos um conjunto de dados que já vem disponível com o R – O conjunto AIRQUALITY.

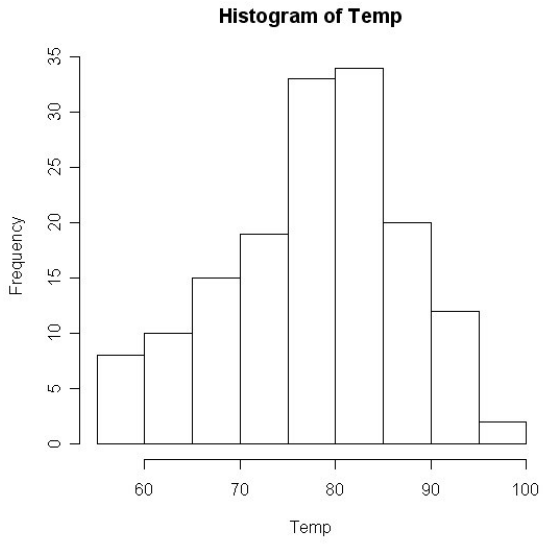
Estes dados são medidas de concentração de ozônio (Ozone), radiação solar (Solar. R), velocidade de vento (Wind) e temperatura (Temp) coletados diariamente por cinco meses.

Primeiramente vamos carregar e visualizar os dados com os comandos:

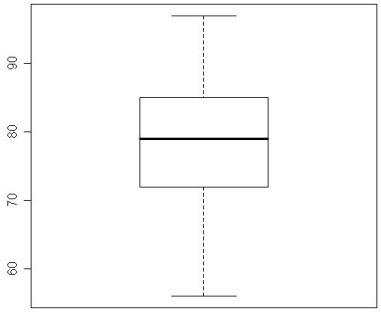
```
> data(airquality) # carrega os dados
> airquality
```

Vamos agora usar alguns comandos para “conhecer melhor” os dados:

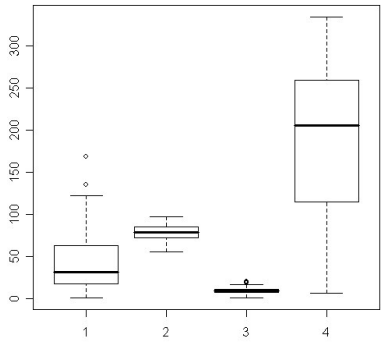
```
> names(airquality)  # nomes das colunas (variáveis)
[1] "Ozone"   "Solar.R" "Wind"    "Temp"    "Month"   "Day"
> dim(airquality) # dimensões do data-frame
[1] 153   6
> help(airquality)  3 mostra o “help” que explica os dados

> summary(airquality)
     Ozone           Solar.R           Wind             Temp
 Min.   :  1.00   Min.   :  7.0   Min.   : 1.700   Min.   :56.00
 1st Qu.: 18.00   1st Qu.:115.8   1st Qu.: 7.400   1st Qu.:72.00
 Median : 31.50   Median :205.0   Median : 9.700   Median :79.00
 Mean   : 42.13   Mean   :185.9   Mean   : 9.958   Mean   :77.88
 3rd Qu.: 63.25   3rd Qu.:258.8   3rd Qu.:11.500   3rd Qu.:85.00
 Max.   :168.00   Max.   :334.0   Max.   :20.700   Max.   :97.00
 NA's   : 37.00   NA's   :  7.0
     Month            Day
 Min.   :5.000   Min.   : 1.00
 1st Qu.:6.000   1st Qu.: 8.00
Median :7.000   Median :16.00
 Mean   :6.993   Mean   :15.80

 3rd Qu.:8.000   3rd Qu.:23.00
 Max.   :9.000   Max.   :31.00

> attach(airquality)
> summary(Ozone)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
   1.00   18.00   31.50   42.13   63.25  168.00   37.00
> mean(Ozone, na.rm=T)
[1] 42.12931
> hist(Temp)
```



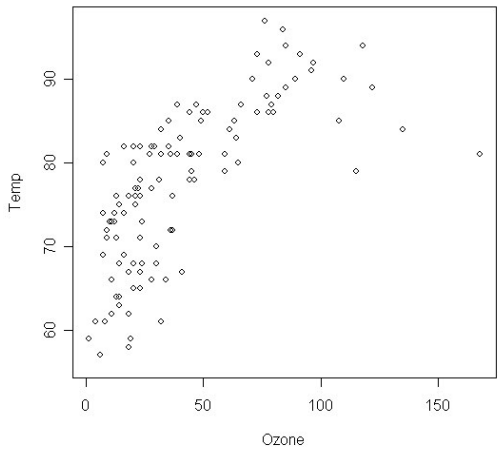
```
> boxplot(Temp)
```



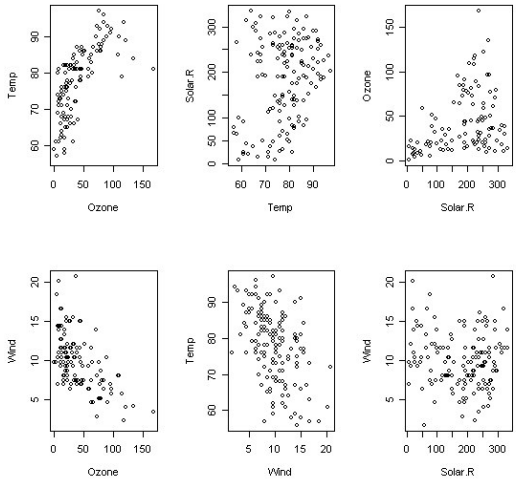

```
> boxplot(Ozone, Temp, Wind, Solar.R)
```




```
> plot(Ozone, Temp)
```



```
> par(mfrow=c(2,3)) # possibilita que vários gráficos sejam colocados juntos
> plot(Ozone,Temp)
> plot(Temp,Solar.R)
> plot(Solar.R,Ozone)
> plot(Ozone,Wind)
> plot(Wind,Temp)
> plot(Solar.R,Wind)
```



Outras análises que podem ser feita no R é o teste de correlação e o teste de independência (Qui-quadrado):

```
> cor.test(Ozone,Temp)

  Pearson's product-moment correlation

data:  Ozone and Temp
t = 10.4177, df = 114, p-value < 2.2e-16
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.591334 0.781211
sample estimates:
      cor 0.6983603

> chisq.test(Ozone,Temp)

  Pearson's Chi-squared test

data:  Ozone and Temp
X-squared = 2699.624, df = 2508, p-value = 0.004036

Warning message:
Chi-squared approximation may be incorrect in: chisq.test(Ozone, Temp)
```

10. DISTRIBUIÇÃO BINOMIAL

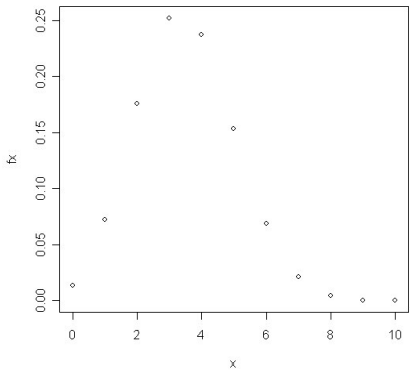
Para os cálculos de distribuição binomial usamos a função **dbinom**. Antes de começarmos é importante darmos uma olhada no **help** dessa função.

```
> help(binom)
```

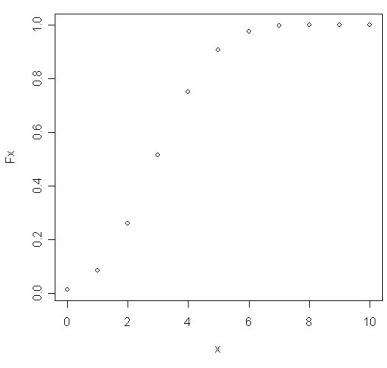
Seja X uma v. a. com distribuição binomial com n=10 e p=0.35. Vamos ver os comandos do R para:

- Fazer os gráficos das funções de densidade
- Idem para a função de probabilidade
- Calcular P|X=7|
- Calcular P|X<8| = P|X<-7|
- Calcular P|X>-8| = P|X>7|
- Calcular P[3<X<-6] = P[4<-X<-7]

```
> x <-0:10
> x
 [1]  0  1  2  3  4  5  6  7  8  9 10
> fx <- dbinom(x,10,0.35)
> fx
 [1] 1.346274e-02 7.249169e-02 1.756530e-01 2.522196e-01 2.376685e-
01
 [6] 1.535704e-01 6.890980e-02 2.120302e-02 4.281378e-03 5.123017e-
04
[11] 2.758547e-05
> plot(x,fx)
```




```
> Fx <- pbinom(x,10,0.35)
> Fx
 [1]    0.01346274                    0.08595444 0.26160739
0.51382702                    0.75149551 0.90506592
 [7]    0.97397572                    0.99517873 0.99946011
0.99997241                    1.00000000
> plot(x,Fx)
```

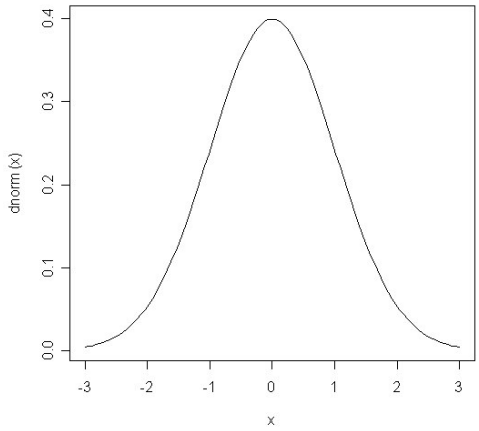


```
> dbinom(7,10,0.35)
[1] 0.02120302
> pbinom(7,10,0.35)
[1] 0.9951787
> 1-pbinom(7,10,0.35)
[1] 0.004821265
>
> pbinom(7,10,0.35,lower=F)
[1] 0.004821265
> pbinom(6,10,0.35)
[1] 0.9739757
> pbinom(6,10,0.35)-pbinom(3,10,0.35)
[1] 0.4601487
```

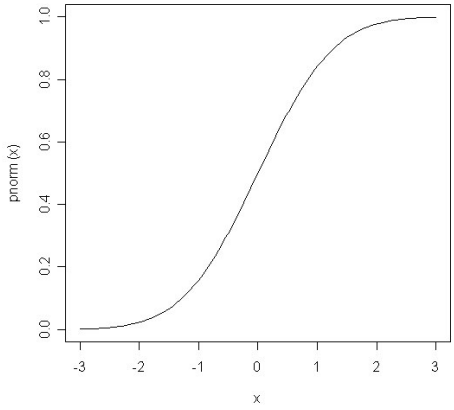
11. DISTRIBUIÇÃO NORMAL

A funcionalidade para a distribuição normal é implementada por argumentos que combinam as seguintes letras (d – densidade de probabilidade f(x) no ponto; p – função de probabilidade acumulada F(x) no ponto; q – quantil correspondente a uma dada probabilidade; r – retira uma amostra da distribuição) com o termo **norm**. Por defaut as funções assumem a distribuição normal padrão N(u=0, sigma2=1).

```
> dnorm(-1)
[1] 0.2419707
> pnorm(-1)  # P(X<- -1)
[1] 0.1586553
> qnorm(0.975)  # P (X<- a)=0.975
[1] 1.959964
> rnorm(10)   # amostra de 10 elementos da normal padrão
[1]   -0.64510490       0.60990359   -0.02129819   -0.88023803
0.76878430 -0.58685400
[7]  1.59534874 -0.89408665 -0.09678214 -1.25377940
> args(rnorm)
function (n, mean = 0, sd = 1)
NULL
> qnorm(0.975, mean=100, sd=8)
[1] 115.6797

> plot(dnorm, -3,3)
```



> plot(pnorm, -3,3)



12. INTERVALOS DE CONFIANÇA

Veremos agora como utilizar o R para obter intervalos de confiança para parâmetros de distribuições de probabilidade.

12.1 t – student

Sabemos que o intervalo de confiança para a média de uma distribuição normal com variância desconhecida, para uma amostra de tamanho n é dado por:

Formula do t-student

```
> chuva<- c(105,158.4,0,70,4,71.7,31.4, 13.7, 17.2, 48, 149, 83.1,
132.2)
> chuva
[1] 105.0 158.4   0.0  70.0   4.0  71.7  31.4  13.7  17.2  48.0
149.0  83.1 132.2
> t.test(chuva)

    One Sample t-test

data:  chuva
t = 4.4466, df = 12, p-value = 0.0007976
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
  34.66842 101.28543
sample estimates:
mean of x
 67.97692
```

13. TESTES DE HIPÓTESES

13.1 Comparação de variâncias de uma distribuição normal

Vamos verificar se as duas cidades possuem a mesma distribuição de chuvas. Para isso, pegamos dados pluviométricos de Rio Claro e de Limeira no ano de 2004.

Rio Claro:

323,9	321,9	134,8	163,9	110	46,1	83	0	7	113,7	221,6	354,3

Limeira:

259,4	249	70,5	88,5	66	84	61	1,1	15,4	82,3	148	182

Verificaremos se:

Ho: $\sigma_A^2 = \sigma_B^2$

Ha: $\sigma_A^2 \neq \sigma_B^2$

Calcula-se o teste:

$$F = S_A^2 / S_B^2$$

E em seguida comparando-se este valor com um valor da tabela de F e/ou calculando-se o p-valor associado com na – 1 e nb – 1 graus de liberdade. Devemos também fixar o nível de significância do teste, que neste caso vamos definir como sendo 5%.

```
> RC <- c(323.9, 321.9, 134.8, 163.9, 110, 46.1, 83, 0, 7, 113.7,
221.6, 354.3)
```

```
> LM <- c(259.4, 249, 70.5, 88.5, 66, 84, 61, 1.1, 15.4, 82.3,
148,182)
> nRC <- length(RC)
> nRC
[1] 12
> nLM <- length(LM)
> nLM
[1] 12
```

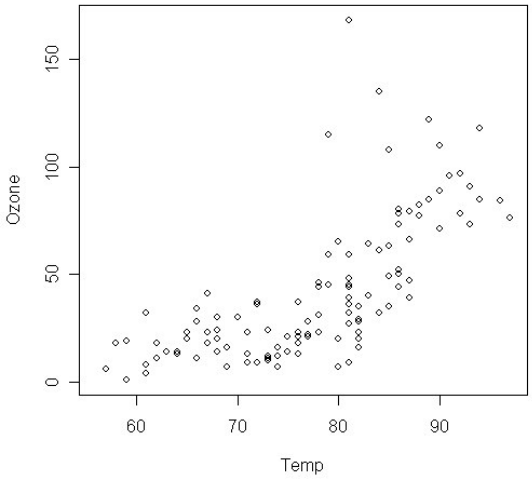
O R já possui uma função para esse teste, o `var.test`, o qual possui mais de uma função associada.

```
> var.test(RC,LM)

var.test(RC,LM,alternative="less",conf.level=0.90)

        F test to compare two variances

data:  RC and LM
F = 2.1816, num df = 11, denom df = 11, p-value = 0.2115
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
 0.6280421 7.5783240
sample estimates:
ratio of variances
          2.181629
```

14. REGRESSÃO LINEAR SIMPLES

Para fazermos uma regressão linear utilizaremos novamente o airquality.

```
> airquality
> attach(airquality)
> airfit<-lm(Ozone~Temp)
> airfit

Call:
lm(formula = Ozone ~ Temp)

Coefficients:
(Intercept)         Temp
   -146.995        2.429

> plot(Temp,Ozone)
```



> abline(airfit)