Universidade Estadual Vale do Acaraú
CCET - Centro de Ciências Exatas e Tecnologia
Coordenação de Matemática

Mini Curso de Introdução ao R

Profº. Nilton José Neves Cordeiro

João Paulo Prado Almeida

De 28 de Setembro a 03 de Outubro

Sumário

Capítulo 1

O Software R

O R é um software gratuito, isso é, pode ser obtido sem nenhum custo financeiro de direitos autorais. Sua origem é de um software estatístico, mas com atuações de vários colaboradores, o R passou a ser utilizado para fins mais diversificados.

No website oficial do R (www.r-project.org), define o software como uma "linguagem e ambiente para estatística computacional e gráfica".

Podemos obter o R gratuitamente através de vários "espelhos" existentes em diversos países. No Brasil, por exemplo, podemos conseguir o código fonte em link de instituições como a Fundação Oswaldo Cruz, a Universidade de São Paulo e as Universidades Federais do Paraná e Viçosa.

São disponibilizadas constantemente novas versões do software R para as plataformas Windows, Linux e Macintosh, que torna possível o uso em quase todos os microcomputadores. Um dos grandes diferenciais do software é que podemos obter dicas diretamente com membros da equipe do R Core Team, bastando enviar mensagem por e-mail (vide link "Mailing Lists" no website oficial para receber uma sugestão).

No mesmo website podemos encontrar manuais com vasto material e uma busca por alguns assuntos e dúvidas mais freqüente.Todas as funções e conjuntos de dados do R são armazenados em packages. Isto é feito com muita eficiência e permite a inclusão de novas rotinas e implementações através da contribuição de vários autores de diversas áreas,além da estatística. O uso e a flexibilidade dos packages podem ser visto de maneira acessível em R Developmente Core Team.

Considero como as principais qualidades do R:

- é um software gratuito e de livre distribuição;
- a sua implementação poder ser estendida através de packages adicionais que são continuamente disponibilizados por colaboradores;
- é permitida a criação e modificação de funções;
- é usado nas plataformas Windows, Linux e Macintosh;
- constantemente surgem versões mais atuais e completas;
- engloba várias áreas de conhecimento (Estatística, Matemática, Matemática Financeita, Atuária, Economia, etc.).

Capítulo 2

Alguns comandos básicos do R

O comando help(rotina) ou ?rotina é utilizado para se obter uma ajuda rápida sobre determinada rotina. A documentação do R pode ser aberta no navegador padrão através do comando help.start(). Para procurar em quais rotinas aparece determinado assunto, deve-se digitar help.search("assunto"). Para instalarmos os pacotes usamos o comando *install.packages("nome do pacote")* e para ativar o pacote instalado usamos *library(nome do pacote)*.

O R ignora tudo o que estiver à direita do sinal '#' , interpretando o conteúdo como um comentário. Considerando que > é a indicação que o R está pronto para receber os comandos e '=' o operador de igualdade, ilustra-se a de algumas rotinas com os respectivos comentários contendo as explicações:

```
> 2*3 + 2^3/2          # retorna o valor 10
> sqrt(9)          # retorna como resposta a raiz quadrade de 9 que é o valor 3
> x=2          # armazenando um escalar
[1] 2
> x=c(1,9,-15,8) # constrói um vetor
> x
[1] 1 9 -15 8
> x=c(-3,x,6) # amplia o vetor anterior
> x
[1] -3 1 9 -15 8 6
> c(-3,x,6) # se não armazena num objeto, o resultado já é apresentado
[1] -3 -3 1 9 -15 8 6 6
> 1:6 # seqüência de números
[1] 1 2 3 4 5 6
> 6:1 # ao contrário
[1] 6 5 4 3 2 1
> seq(0,1,0.1) # a seqüência começa em 0 e vai até 1, com espaçamento de 0.1
[1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
> seq(0.1,1,0.2) # não necessariamente termina em 1
[1] 0.1 0.3 0.5 0.7 0.9
> rep(2,5) # repete o 1º argumento o número de vezes indicado no 2º argumento
[1] 2 2 2 2 2
```

```
> rep(c(1,3),4)
[1] 1 3 1 3 1 3 1 3
> c(rep(1,3),rep(0,4))
[1] 1 1 1 0 0 0 0
> rep(c(0,2,5),c(1,3,4)) # pode-se utilizar um vetor em cada argumento
[1] 0 2 2 2 5 5 5 5
> x[4] # extrai o 4º elemento de x
[1] -15
> x[2:4] # extrai do 2º ao 4º elemento de x
[1] 1 9 -15
> x[c(1,3:4)] # extrai os elementos 1, 3 e 4 de x
[1] -3 9 -15
> x[-3] # reproduz x sem o 3º elemento
[1] -3 1 -15 8 6
> x[-c(1,4)] # reproduz x sem o 1º e o 4º elementos
[1] 1 9 8 6
```

Operação com vetores

```
> x=1:10 # cria um vetor x com elementos de 1 a 10
> x+2 # retorna a cada elemento do vetor x adicionado 2
> sqrt(x) # retorna a raiz quadrada de cada elemento do vetor x
> y=1:10 # cria um vetor y com elementos de 1 a 10
> x+y # soma cada elementos do vetor x com os elementos do vetor y
```

* Se os vetores tiverem tamanhos diferentes, os elementos do menor vetor até atingir o do maior vetor.

```
> y=c(1,2,3)
> x; y; x+y
[1] 1 2 3 4 5 6 7 8 9 10
[1] 1 2 3
[1] 2 4 6 5 7 9 8 10 12 11
```

Capítulo 3

Matrizes

Existem várias maneiras disponíveis para se construir matrizes. As mais simples são por meio dos comandos rbind() e cbind() que empilham escalares, vetores ou matrizes em, respectivamente, linhas e colunas. Outra possibilidade é utilizar o comando matrix() que recebe um vetor e pelo menos uma das dimensões da matriz. Veja os exemplos:

```
> x1=rbind(1:3,c(1,3,-1))
> x1
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    1    3   -1

> x2=cbind(c(1,3),c(0,4))
> x2
     [,1] [,2]
[1,]    1    0
[2,]    3    4

> cbind(x2,x1)
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    0    1    2    3
[2,]    3    4    1    3   -1

> matrix(1:6,nrow=3) #este comando utiliza o argumento byrow com F (False) como padrão..
     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6

> matrix(1:6,ncol=3) #que faz com que o vetor seja preenchido por coluna na matriz
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6

> matrix(1:6,ncol=3,byrow=T) #utilizando a opção T (True), preenche-se por linha
```

```
     [,1] [,2] [,3]
[1,] 1 2 3
[2,] 4 5 6

> x1[2,] #extrai a 2a linha de x1
[1] 1 3 -1

> x1[,2] #extrai a 2a coluna de x1
[1] 2 3

> x1[,2:3] #extrai uma submatriz de x1
[,1] [,2]
[1,] 2 3
[2,] 3 -1

> x1[,-1] #outra maneira
[,1] [,2]
[1,] 2 3
[2,] 3 -1

> x1[,c(1,3)] #outra submatriz
   [,1] [,2]
[1,] 1 3
[2,] 1 -1
```

Os sinais $+, -, *, /$ e $\hat{}$ representam as operações básicas de adiçãoo, subtração, multiplicação, divisão e potência.Ao utilizar estes sinais entre vetores ou matrizes, a operação é realizada elemento a elemento. O mesmo acontece com logaritmos neperianos e exponenciais, representadas pelas rotinas *log()* e *exp()*, repectivamente. A operação $A\%*\%B$ realiza a multiplicação matricial, t(A) transpõe e solve(A) inverte. Ao receber como argumento uma matriz quadrada, a rotina *exp()* extrai sua diagonal principal. Se, no entanto, receber como argumento um vetor, a rotina criará uma matriz quadrada com os elementos do vetor na diagonal principal. Por fim, se a rotina receber como argumento um número n, uma matriz identidade de ordem n é retornada.

```
> x1=matrix(1:9,nrow=3)
> x2=matrix(9:1,nrow=3)
> t(x1)
     [,1] [,2] [,3]
[1,] 1 2 3
[2,] 4 5 6
[3,] 7 8 9

> x1
   [,1] [,2] [,3]
[1,] 1 4 7
[2,] 2 5 8
[3,] 3 6 9

> x2
```

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

> x1*x2
     [,1] [,2] [,3]
[1,] 9 24 21
[2,] 16 25 16
[3,] 21 24 9

> x1%*%x2
     [,1] [,2] [,3]
[1,] 90 54 18
[2,] 114 69 24
[3,] 138 84 30

> solve(x1*x2)
     [,1] [,2] [,3]
[1,] -0.7361111 1.333333 -0.6527778
[2,] 0.8888889 -1.666667 0.8888889
[3,] -0.6527778 1.333333 -0.7361111

> diag(x1)
[1] 1 5 9

> diag(c(1,3,4))
     [,1] [,2] [,3]
[1,] 1 0 0
[2,] 0 3 0
[3,] 0 0 4

> diag(2.5)
     [,1] [,2]
[1,] 1 0
[2,] 0 1
```

Nomear linhas e colunas de uma matriz

```
> x=matrix(1:10,5)
> dimnames(x)=list(c("a1","a2","a3","a4","a5"),c("b1","b2")) # nomeia linhas e colunas
```

Capítulo 4

Arquivo de Dados

Para ilustrar as pontencialidades do programa R, será usado o arquivo de dados *tratamento1.txt*. Inicialmente será carregado o pacote *foreign*, que permitirá importar arquivos de diversos outros programas.

Importando arquivo de dados

> require(foreign) # carregar o pacote foreign (não necessariamente para arquivos tipo textos).

> d=read.table("/home/aluno/r/tratamento1.txt",hearder=TRUE) # Ler o arquivo tratamento1.txt

> d # Mostrar na tela o arquivo

Criando Arquivo de Dados

Há mais de uma maneira de criar um arquivo de dados no R. Apresentaremos duas delas

Função: scan()

Procedimento: Digite a função e aperte a tecla ENTER. O programa esperará pela entrada do "valor". A cada valor aperte a tecla ENTER; para finalizar aperte a ENTER duas vez. Abaixo é mostrado um exemplo.

```
> m=scan()
1:10
2:20
3:30
4:40
5:50
6:
Read 5 items
> m
[1] 10 20 30 40 50
> is.vector(m)
[1] TRUE
> p=scan()
1: 100
```

```
2: 200
3: 300
4: 400
5: 500
6:
Read 8 items
> p
[1] 100 200 300 400 500
> is.vector(p)
[1] TRUE
```

Podemos agora, fazer uso da função *data.frame* para criar o arquivo de dados.

```
> dados=data.frame(m,p)
> dados
   m   p
1  10   100
2  20   200
3  30   300
4  40   400
5  50   500
```

Vejamos uma forma mais conviniete:

```
> dados=edit(data.frame()) # Criar um data.frame em branco para edição.
> dados=edit(dados) # para editarmos dados
```

Salvando Arquivo de dados

Função: write.table()

```
> write.table(d,"/home/aluno/r/teste.txt")
> ?write.table # Obtem ajuda para a função.
```

Mais sobre arquivos de dados

Para importa arquivos da **planilha EXCEL** salve a planilha com formato do tipo texto formatado (separado por espaços) (a extensão será .prn) ou CSV(separado por vírgula) ou CSV(MS-DOS). A leitura (importação) do arquivo pelo R pode ser feita pela função *read.table*. Veja exemplos abaixo:

```
> dados1=read.table("/home/aluno/r/teste.prn",dec=",");
```

dados1 # Se o arquivo tiver cabeçalho, acrescente a opção header=TRUE.

```
> dados2=read.table("/home/aluno/r/teste.csv",dec=",",sep=";"); dados2
```

No caso dos formatos CSV, o arquivo é salvo (no Excel) usando como separador de registro o ponto-e-vírgula. Por isso, na leitura pelo R, usa-se a opção *sep* para importar de forma conveniente. Se o arquivo tiver cabeçalho, acrescente a opção header=TRUE.

Para salvar arquivos de dados do R no formato de **planilha EXCEL**, faça:

```
> write.table(d,file="/home/aluno/r/teste.csv", sep=";", row.names=FALSE)
```

Capítulo 5

Estatística Descritiva

fazer a leitura do arquivo, caso não tenha sido feita anteriormente.

> d=read.table("/home/aluno/R/tratamento1.txt", header=TRUE, dec=",")

> d=edit(d) # Apresenta os dados na tela(planilha)

> attach(d) # Anexa o objeto d; permite acessar o arquivo de dado pelo nome das variáveis.

> names(d) # Mostrar os objetos (nomes das variáveis) associadas a d.

> str(d) # mostra a estrutura do arquivo

> summary(rede) # Quartis, mínimo e mázimo # Se o arquivo não estiver anexado (attach(d)) é necessário usar o nome do arquivo seguido do símbolo $ antes do nome da variável: summary(d$rede).

> range(rede) # mínimo e máximo

> min(rede);max(rede)

> quantile(rede) # Quartis

> quantile(rede,probs=seq(0,1,0.10)) Decis; a opção probs permite obter os percentis.

> mean(rede) # Média aritmética de cada variável rede

> mean(rede, na.rm=TRUE) # Média aritmética da variável rede, excluíndo os possíveis "dados perdidos"

> mean(rede, trim=0.05) # Média aritmética aparada(5%)

> mean(d) # Média aritmética de todas as variáveis do arquivo d.

> mean(d[,2:4]) # Média aritmética das variáveis (colunas) 2 até 4.

Um exemplo como Média ponderada.

> notas=c(5,6,9)

> pesos=c(1,2,3)

> weighted.mean(notas,pesos)

> sd(rede); var(rede) # Desvio-padrão e variância

Tabela de frequêcia

Adicione ao arquivo d a variável setor com calores 1 1 1 1 2 2 2 3 3 3 3 3 3 3 4 4 4 4 4. feche a planilha. Para isso, use o código abaixo

> d=edit(d)

> str(d) # ver a estrutura do arquivo

> d$setor=as.factor(d$setor) # Setor é uma variável qualitativa ("factor")

> levels(setor)=c("r1","r2","r3","r3")

> str(d)
> attach(d)
> table(setor) # Tabela de frequência
> prop.table(table(setor)) # obetendo precentuais para a tabela acima.
> t=as.data.frame(table(setor))
No exemplo acaima, o objeto resultante da table(setor) (tipo array) é lido como um data.frame e armazenado em tab.

> t1= cbind(t,"perc"=t[,2]/sum(t[,2])*100) # acrescentando percentuais.
> t1=cbind(t,"perc"=round(t[,2]/sum(t[,2])*100,2)) # Função round: controla o nume de casas decimais. O primeiro argumento é o valor a ser arrendondado e o segundo é o número de casas decimais.
> str(t1) # Verifica a estrutura do objeto t1
> print(t1,digits=2)
> y=tapply(pea, setor, mean) # retorna a média da variável pea para cada valor(categoria) da variável setor.
> barplot(y) # Gráfico de colunas
Adicione ao arquivo d a variável subsetor com valores 1 1 2 2 1 1 2 1 1 1 1 2 2 2 1 1 1 2 2.
> d=edit(d)
> d$subsetor=as.factor(d$subsetor)
> levels(d$subsetor)=c("sub1","sub2")
> attach(d)

> tapply(pea,list(setor,subsetor),mean)
No exemplo acima, a média da variável pea é apresentada para cada combinação dos valores(categorias) das variáveis setor e subsetor.

> tapply(passo,list("setor"=setor,"subsetor"),length) # Tabela de dupla entrada
No exemplo acima, a função length retorna o número de casos da variável pea para cada combinação dos valores(categorias) das variáveis setor e subsetor.
> addmargins(table (setor,subsetor)) # substituímos ftable por table
> ftable=(setor,subsetor) # Tabela de Contingência
> ftable=(Titanic, row.vars=1:2, col.vars="Survived") # Outro exemplo com ftable.
> help(Titanic) # Informações sobre o arquivo de dados Titanic
> require(nlme)
> data(Orthodont)
> help(Orthodont)
> x=ftable(Orthodont[c("age","Sex")])
> d1=edit(Orthodont) # Faça alteralções no arquivo
> x=table(d1[("age","Sex")])
> addmargins(x)

Função: aggregate()

Esta função estabelece subgrupos de dados e calcula estatísticas para cada um deles. Use o R para reproduzir(digitar) o arquivo abaixo:

	reg	ind	set	pres	carb	temp
1	240	10	1	91	100	105
2	240	10	2	90	95	120
3	240	10	3	88	110	99
4	240	20	1	87	88	120
5	240	20	2	91	94	130
6	240	20	3	94	99	132
7	300	10	1	87	97	105
8	300	10	2	86	96	106
9	300	20	1	88	110	99
10	300	20	2	91	105	102
11	300	20	3	90	100	105
12	320	10	1	89	98	110
13	320	10	2	86	99	108
14	320	20	1	79	100	105
15	320	20	2	91	102	101

As variáveis do arquivo mostrado acima são: região(reg), indústria(ind), setor(set), pressão(pres), nível de carbono(carb) e temperatura(temp)

> dados=edit(data.frame())
> dados
> write.table(dados,"/home/aluno/aggr.txt",sep=,row.names=FALSE)
> attach(dados)

> y=aggregate(dados[,3:5], list("reg"=reg,"ind"=ind), FUN=mean) # para combinação das variáveis reg e ind é calculada a média aritmética das variáveis(colunas) 3,4 e 5 do data.frame dados.

> print(agrregate(dados[,3:5], list("reg"=reg,"ind"=ind), FUN=mean), digits=3) # Uma opção para "controlar"o número mínimo de dígitos significativos a serem impressos para a função especificada.

> options(digits=6) # muda o número mínimo de dígitos significativos a serem impressos para os resultados posteriores. O default é digits=7.

Capítulo 6

Construíndo Gráficos no R

Funções gráficas de alto nível

Uma das funções mais usadas no R é a função plot(), que é uma função genérica e o tipo de gráfico que é criado depende do tipo ou da classe do primeiro argumento dado à função.

Se x e y são vetores, plot(x,y) cria um gráfico de pontos ou diagrama de dispersão de y em função de x. O mesmo efeito é obtido dando apenas um argumento ou uma lista contendo dois elementos x e y ou uma matriz de duas colunas.

plot(x,y)

Se x é uma série de observação ao longo do tempo, este comando produz um gráfico de série temporal; se x é um vetor numérico, o comando cria um gráfico dos valores do vetor sobre os respectivos índices; se x é um vetor complexo, é produzido um gráfico da parte imaginária versus a parte real dos elementos.

plot(x)

Argumentos das funções gráficas de alto nível

É possível definir uma série de argumentos para as funções gráficas de alto nível, entre os quais:

add= TRUE Obriga a portar-se como se tratasse de uma função de baixo nível, de modo que o gráfico criado será sobreposto ao gráfico atual, em vez de o apagar previamente, ressaltando que só está disponível para algumas funções.

axes=False Elimina os eixos. Esta opção é útil para quem a defina, e personalize os eixos com a função axes(). Por definição a opção é axes=TRUE que define automaticamente os eixos.

log="x"

log="y"

log="xy" Transforma o eixo x, o eixo y ou ambos, em escala logarítmica. Não funciona em alguns tipos de gráficos.

type= Este argumento controla o tipo de gráfico produzido, de acordo com as seguinte especificações:

type= "p" Representa os pontos individualmente (por definição)

type= "l" Gráfico de linhas

type= "b" Pontos unidos por linha

type= "o" Pontos e linhas, com estas sobrepostas aos pontos

type= “h” Representa linhas verticais desde os pontos ao eixo x=0, assim teremos um gráficos conhecido por “hastes”.

type=”n” Não se produz qualquer gráfico; são apenas desenhados os eixos (por definição) e são representadas as coordenadas de acordo com os dados.

xlab = string

ylab = string Definem os nomes para os eixos x e y, respectivamente, para substituição dos nomes definidos por definição, que normalmente são os nomes dos objetos utilizados para a criação do gráfico.

main=string Define o título do gráfico, colocando-o no topo, em letras de tamanho grande.

sub=string Define o subtítulo do gráfico, colocando-o abaixo do eixo x em letras de tamanho pequeno.

Funções gráficas de baixo nível

Pode acontecer que algumas funções gráficas de alto nível não produzam exatamente o tipo de gráfico pretendido. Nesse caso, os comandos de baixo nível podem ser usados para adicionar informações, tal como pontos, linhas ou texto ao gráfico atual.

Algumas das funções de baixo nível mais usadas são:

points(x,y)

lines(x,y) Acrescenta pontos ou linhas no gráfico atual.

A opção type da função plot() pode ser usada nesta função (os valores pré-definidos são “p” para points() e “l” para lines()).

text(x,y,. . .) Acrescenta texto ao ponto (x,y)

legend(x,y,. . .) Aplica a legenda ao gráfico atual, na posição especificada. As fontes a usar, estilo de linhas, cores, etc., são definidos no vetor legenda. Especificando algumas características, tais como se segue:

legend(. . . ,fill= “”) Cores de preenchimento;

legend(. . . ,col= “”) Cores para linhas ou pontos;

legend(. . . ,lty= “”) Tipo de linha;

legend(. . . ,lwd= “”) Espessura da linha;

Ressaltando que a escolha da cor deve ser definida em inglês.

title(main,sub) Aplica o título principal, main, na parte superior do gráfico, em caracteres grandes e o subtítulo, sub, na parte inferior, em fontes menores.

Produzindo Gráficos no ambiente R

De posse dos conteúdos apresentado neste trabalho, produziremos alguns gráficos no software R.

Gráficos de pontos individuais

Quando definimos um objeto x, como vetor numérico, a função plot(x) fará por definição, pontos do tipo "∘" para cada valor. Para especificar o tipo de ponto, utilizaremos o argumento pch, que utiliza 25 (vinte e cinco) caracteres pré-estabelecido, conforme mostra o quadro abaixo, ou qualquer caractere entre aspas.

pch=1	○	pch=10	⊕	pch=19	●
pch=2	Δ	pch=11	✡	pch=20	•
pch=3	+	pch=12	⊞	pch=21	○
pch=4	X	pch=13	⊗	pch=22	□
pch=5	◊	pch=14	⊠	pch=23	◊
pch=6	∇	pch=15	■	pch=24	Δ
pch=7	▯	pch=16	●	pch=25	∇
pch=8	*	pch=17	▲	pch="1"	1
pch=9	✥	pch=18	♦	pch="p"	p

Gráficos em linha

Para transformar o gráfico de pontos individuais para gráfico de linha, basta acrescentar o argumento type= "l" e para melhorarmos o visual do gráfico, faremos uso do argumento col para definir a cor da linha do gráfico, sabendo que por definição do próprio programa é a cor preta (black). Segue o exemplo de comandos para um gráfico de linha azul.

```
> x=c(10,15,30,20,25)
> plot(x,type="l",col="blue")
```

Gráficos de colunas

O R utiliza a função barplot() para a construção de gráficos de colunas. São adicionadas título e subtítulo com o comando title(main,sub) e texto ou etiqueta no gráfico com o comando text() (este deveremos informar as coordenadas para que o caractere central do texto ou etiqueta deve aparecer).

Com os argumentos density e angle pode-se preencher com textura as colunas do gráfico. O argumento density define quantas retas "cabem" em uma parte da coluna,

caso esse número seja 0 (zero), não terá nenhuma reta e conforme o número seja maior, mais retas "caberão". O argumento angle define o ângulo em que as retas serão traçadas.

```
>x=c(10,15,30,20,25)
>barplot(x, density=c(15,10,5,25,3),angle=c(10,30,15,60,90))
>title(main="Titulo do Gráfico",sub="Subtítulo do Gráfico")
>text(2,20,"Texto para exemplificar")
```

Gráficos de barras

Assim como o gráfico de colunas é parecido com o gráfico de barras, os comandos no R também o são, a diferença que o argumento horiz será defino como True, pois por definição é False.

Os textos que devem aparecer nos eixos podem ser definidos pelos os argumentos xlab e ylab, respectivamente, eixo x e eixo y.

Caso seja necessário o uso de gráficos de colunas conjugadas, o uso do argumento beside=TRUE, assim as colunas ficaram ao lado, de acordo com os dados atribuídos no vetor. Com os mesmos comandos do exemplo anterior e acrescentando os argumentos horiz, xlab e ylab, construiremos o gráfico gerado pelos comandos:

```
>x=c(10,15,30,20,25)
>barplot(x,horiz=TRUE,density=c(15,10,5,25,3),angle=c(10,30,15,60,90),xlab="Eixo
+do X", ylab ="Eixo do Y")
>title(main="Título do Gráfico","Subtítulo do Gráfico")
>text(2,20,"Texto para exemplificar")
```

Gráficos de setor

A função pie() produz o gráfico de setor ou "pizza" e assim como algumas outras funções, ela aceita o argumento main e sub, para adicionarmos o título e o subtítulo, o argumento col para definir as cores de cada setor e o argumento label para nomear os setores da circunferência, quando for label=NA, não adicionará nenhum nome aos setores.

Para aplicar uma legenda será necessário a função legend(x,y,. . .), onde o par ordenado (x,y) indica onde o quadro da legenda irá aparecer e o argumento fill preenche com as cores esses quadrados. Teremos um gráfico de setor dos comandos abaixo:

```
> x=c(10,15,30,20,25)
>pie(x,col=c("green","red","light blue","yellow","orange"),main="Título do
+Gráfico",sub="Subtítulo do Gráfico", label=c("Brasil","Bolivia","Argentina",
+"México","Chile"))
>legend(0.8,1, c("Brasil","Bolivia","Argentina","México","Chile"),
```

+fill=c(”green”,”red”,”light blue”,”yellow”,”orange”))

Histograma

A função hist() produz o histograma, que é uma distribuição de freqüência, como já vimos. O histograma produzido conforme o comando abaixo:

> x=c(rep(5,10),rep(6.5,4),rep(7.5,8),rep(8.5,3),rep(9.5,1))

> hist(x, main= ”Histograma das média de outubro de 2008”,xlab=”Médias”,ylab= +”Quantidade de Alunos”)