

Capítulo 2

A Linguagem R



Gustavo Mello Reis
José Ivo Ribeiro Júnior

Universidade Federal de Viçosa
Departamento de Informática
Setor de Estatística

Viçosa 2007

1. Tela Principal do R

Como pode-se observar (Figura 1), a tela principal do R é bastante simples, cuja utilização é feita através de comandos sobre os objetos (bancos de dados ou resultados gerados por funções, por exemplo), criados pelo usuário quando efetua as análises estatísticas.

No R, os valores numéricos ou não (caracteres) podem ser dispostos em um vetor, matriz ou data frame.

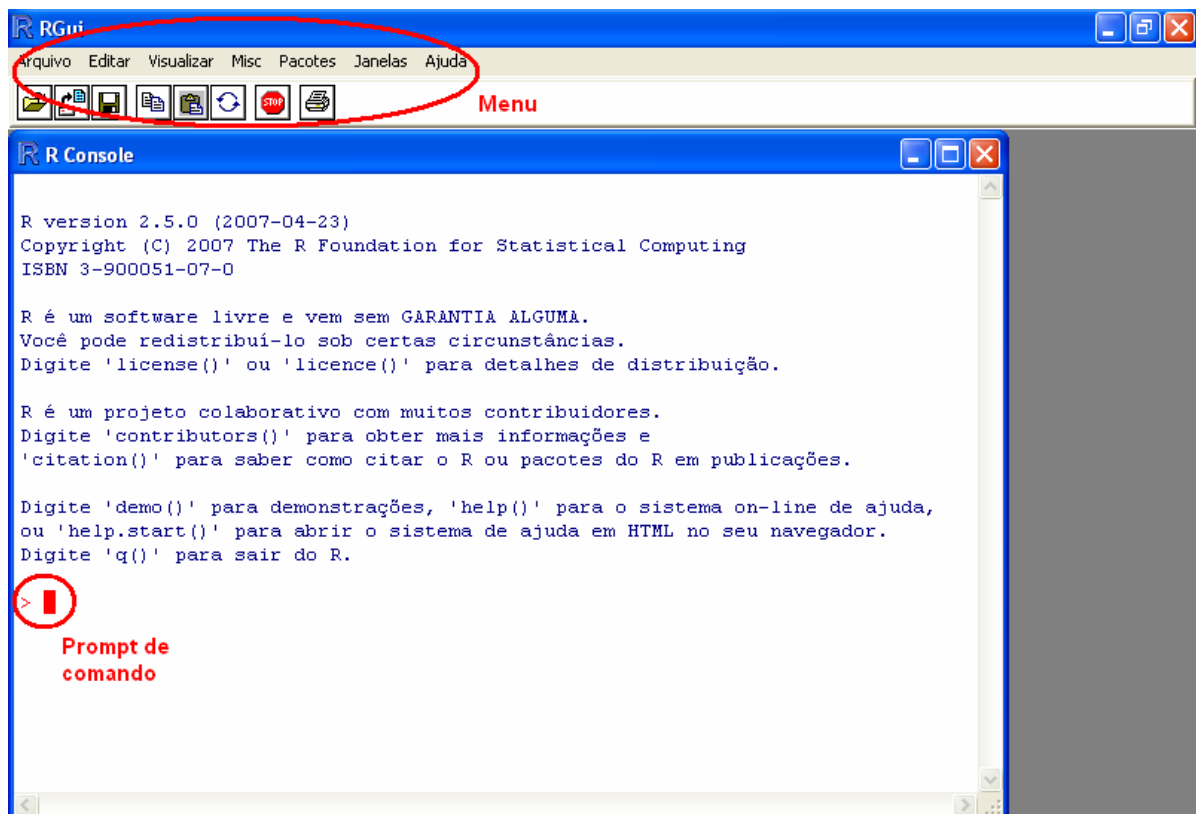
Os comandos são realizados por meio de operadores e de funções vistas da seguinte forma: nome.da.funcao(argumento1, argumento2, ..., argumentoN). É aconselhável não utilizar acentuação nem espaços no nome da função, assim como observado acima, o ponto substitui o espaço. Os argumentos vêm sempre dentro de parênteses e separados por vírgula.

No console, janela onde os comandos são digitados, existe o menu, que é um conjunto de ícones que se encontra na parte superior (Figura 1), e o prompt de comando, que é um sinal indicador de que o programa está pronto para receber o comando (Figura 1).

O R reconhece como local de trabalho, a pasta onde foi instalado, podendo a mesma ser mudada, se houver interesse. Assim, cada vez que se abre um novo console, uma nova sessão é criada, devendo ser salva, quando se tem interesse em acessar novamente os objetos e comandos criados.

Com relação aos dados, NA representa a ausência de um dados, Inf ou -Inf informa que o valor tende a mais ou menos infinito e NaN, informa que o valor não é um número válido.

Figura 1. Menu e prompt de comando



Alguns operadores do R são descritos na Tabela 1:

Tabela 1. Operadores do R

<-	atribui um nome a um objeto. Ex: <code>x<-12</code> (para vê-lo, basta digitá-lo no console.)
=	atribui valor a um argumento, como por exemplo, <code>funcao(arg = valor)</code>
+, -, *, /	adição, subtração, multiplicação e divisão
%/%	fornece o maior valor inteiro menor ou igual ao valor da divisão, como por exemplo, <code>5%/%2 = 2</code>
%%	fornece o resto da divisão, como por exemplo, <code>5%%2 = 1</code>
%*%	multiplicação entre matrizes, como por exemplo, $M_{3 \times 2} \%* \% M_{2 \times 4}$, irá fornecer uma matriz do tipo $M_{3 \times 4}$
x^b	x elevado a b, como por exemplo, <code>2^3 = 8</code>
n1:n2	fornece valores entre os números n1 e n2 espaçados por uma unidade
=, <, >, <=, >=, !=	Utilizado em programação lógica: igual, menor, maior, menor ou igual, maior ou igual e diferente, respectivamente
&	Utilizado em programação lógica, significa “e” (adiciona uma condição)
	Utilizado em programação lógica, significa “ou” (adiciona outra possibilidade)
y~x	indica y em função de x
,	separa um argumento de outro dentro da mesma função
;	pode ser utilizado para digitar mais de um comando na mesma linha, porém é mais comum utilizar a tecla ENTER e digitar o comando na outra linha
“	delimita um caractere
‘	utilizada dentro da área de abrangência do operador anterior, ou seja, dentro de um caractere
()	delimita os argumentos de uma função
{ }	indica o início e o fim de uma função
[]	seleciona parte de um objeto
#	adiciona algum comentário na janela de comandos (console)
?	obtem ajuda ou informações sobre alguma função, como por exemplo, <code>?nome.da.funcao</code> irá abrir uma nova janela com informações a respeito da função desejada

Algumas funções básicas do R são descritas na Tabela 2:

Tabela 2. Funções básicas

ls()	lista o nome dos objetos criados na sessão atual
dir()	lista todos os arquivos na pasta de trabalho atual
rm(<i>objetoA</i> ,...)	remove o(s) objeto(s) entre parênteses
library(<i>nome.do.pacote</i>)	ativa o pacote entre parênteses, na sessão em que for usado, caso esteja instalado no computador, como por exemplo, library(qcc);
c(...)	concatena ou junta valores numéricos ou caracteres em um único vetor, sendo que os caracteres devem ser postos entre aspas
factor(<i>objetoA</i>)	transforma o objeto em fator, ou seja, o divide em níveis
attach(<i>DataFrameA</i>)	permite que cada coluna do Data Frame em questão seja reconhecida como um objeto
detach(<i>DataFrameA</i>)	desfaz a função anterior, sendo aconselhável utilizá-lo sempre que o Data Frame em questão não for mais necessário, para evitar possíveis conflitos com outras variáveis com o mesmo nome

Algumas funções relacionadas aos pacotes do R são descritas na Tabela 3:

Tabela 3. Funções relacionadas a pacotes

summary(packageStatus())	mostra um resumo sobre todos os pacotes existentes, dizendo quais estão instalados, quais podem ser atualizados e quais não estão instalados no computador, quando se está conectado à internet
update.packages()	atualiza todos os pacotes instalados
install.packages(c(" <i>pacote1</i> ",...))	instala o(s) pacote(s) entre aspas, quando se está conectado à internet
remove.packages(c(" <i>pacote1</i> ",...))	remove o(s) pacote(s) entre aspas

Algumas funções matemáticas do R são descritas na Tabela 4:

Tabela 4. Funções matemáticas

choose(n, k)	combinação de n tomado de k a k
factorial(x)	fatorial de x
sqrt(x)	raiz quadrada de x
abs(x)	modulo (valor absoluto) de x

2. Ajuda no R

Ao clicar no ícone “Help”, pode-se ter acesso a alguns manuais do R, em inglês, no formato *.pdf* e também em *Html*. De outro modo, pode-se ter acesso a esse conteúdo através de algumas funções descritas na Tabela 5:

Tabela 5. Funções de ajuda

<code>help(package=nome.do.pacote)</code>	informações sobre o pacote que deve estar ativo na sessão, por meio da função <code>library(nome.do.pacote)</code> . Alguns pacotes são ativados automaticamente quando o R é aberto, como por exemplo, <code>help(package=base)</code>
<code>help.start()</code>	ajuda pela internet
<code>help.search("termo")</code>	pesquisa o termo entre aspas em todos os pacotes ativos
<code>apropos("funcao")</code> ou <code>apropos("objeto")</code>	pesquisa a função ou o objeto entre aspas, em todos os pacotes ativos
<code>example(nome.da.funcao)</code>	mostra exemplo(s) da função entre parênteses, quando ela possuir
<code>demo(nome.do.pacote)</code>	exibe demonstrações de algumas funções contidas no pacote em questão, que deve estar ativo e possuir as demonstrações, como por exemplo, <code>demo(graphics)</code>
<code>ls("package:nome.do.pacote")</code>	exibe todas as funções contidas no pacote em questão, como por exemplo, <code>ls("package:graphics")</code>
<code>help(nome.da.funcao)</code> ou <code>?nome.da.funcao</code>	fornece informações da função entre parênteses

As janelas de ajuda, abertas pela função **help()** ou pelo operador **?**, de forma geral, podem possuir até nove tópicos, como segue:

- Description:** descreve o que a função realiza.
- Usage:** exibe a forma de usar a função com os argumentos na ordem correta entre parênteses e, quando a função possui alguma variação, esta também será mostrada.
- Arguments:** descreve os argumentos possíveis de serem usados, sendo que os argumentos obrigatórios vêm primeiro e a ordem é a mesma mostrada no tópico **Usage**. No final deste tópico poderá aparecer três pontos, o que indica que a função em questão poderá aceitar outros argumentos não descritos e que fazem parte de alguma função relacionada a ela.
- Details:** exibe alguns detalhes sobre como usar os argumentos, os valores que podem ser atribuídos a eles e outros assuntos relacionados.

- e) **Value:** descreve que tipo de objeto a função retorna.
- f) **Note:** é um tipo de observação final, que pode conter dicas para melhorar o funcionamento da função, informações a respeito de possíveis diferenças entre a utilização da função em versões diferentes do R e outras.
- g) **References:** exibe referências bibliográficas relacionadas à função.
- h) **See Also:** exibe algumas funções que têm alguma relação com a função em questão.
- i) **Examples:** exibe exemplos de como usar a função, que podem ser copiados e colados no console para executá-los.

3. Salvar as Sessões do R

Para trabalhar no R, é aconselhável criar uma nova **pasta de trabalho** diferente daquela padrão do Windows, como por exemplo, Rdados. Para escolher essa pasta, deve-se clicar em “Arquivo”, no menu do R, e em “Mudar dir...”. Depois deve-se selecioná-la, clicando em “Browse” e em “OK” e “OK” mais uma vez.

O formato que o R salva suas sessões é *.RData*. Este arquivo irá conter apenas os objetos armazenados. O arquivo *.RData* pode ser criado de duas formas:

- a) menu do R: clicar em “Arquivo”, em “Salvar área de trabalho...”, digitar um nome para a sessão em questão e clicar em “Salvar”.
- b) console do R: função **save.image()** (“nome da sessão.Rdata”), salva o arquivo na pasta de trabalho escolhida anteriormente.

Os comandos utilizados são salvos em outro arquivo do R, que tem a extensão do tipo *.Rhistory*. Para criar este arquivo, deve-se clicar em “Arquivo”, em “Salvar Histórico...”, digitar um nome para o arquivo, preferencialmente o mesmo utilizado para a sessão, e clicar em “Salvar”.

3.1. Abrir as Sessões Salvas

Depois que a sessão estiver salva, pode-se fechar o R e abrí-lo novamente na mesma sessão sem perder nada. Para isso, deve-se ir até a pasta onde a sessão foi salva e dar dois cliques no arquivo “nome da sessão.RData”. O R será aberto e, os objetos criados nesta sessão, estarão prontos para serem usados, sendo que os comandos utilizados ainda não estarão disponíveis. Para ativá-los, deve-se clicar em “Arquivo” e em “Carregar Histórico...” e selecionar o arquivo “nome da sessão.Rhistory” e clicar em “Abrir”. E para acessá-los, basta apertar a seta para cima do teclado que eles irão aparecendo.

4. Entrada de Dados Externos

Se os dados estiverem em uma planilha de dados, deve-se fazer com que o R leia esta planilha e a transforme em um objeto. Mas para isso, deve ser informado o que separa uma coluna da outra. As planilhas do tipo *.xls* e *.ods*, que são os formatos padrões do Excel e do Calc (editor de planilhas do pacote OpenOffice), respectivamente, não possuem nenhum sinal separando as colunas. Portanto, o R não conseguirá separá-las e irá exibir uma mensagem de erro. Um modo fácil de resolver este problema é salvar a planilha de dados com um outro formato (*.csv*), que utiliza o sinal “;” para separar as colunas. Isto pode ser feito no próprio Excel ou no Calc.

Um outro possível problema na entrada de dados é a configuração do sinal utilizado para separar as casas decimais. No Brasil, o padrão é utilizar a vírgula. No entanto, o R utiliza o ponto, que é o padrão da língua inglesa. Para resolver isto, têm-se duas opções: informar ao R que o arquivo a ser lido utiliza a vírgula como separador decimal e ele irá convertê-la para ponto ou alterar a configuração do editor de planilha para que ele aceite o ponto como separador decimal.

De acordo com a Figura 2, pode-se verificar que os dados estão com o ponto como separador decimal. Uma forma fácil de saber se o valor digitado foi reconhecido como numérico ou não, é observar o seu posicionamento na coluna. Os caracteres são alinhados à esquerda, enquanto os números são alinhados à direita. Além disso, deve-se evitar fazer formatações na planilha de dados e, todas as fórmulas executadas, devem ser convertidas a valores. Como o programa R irá reconhecer células em branco que já tenham sido trabalhadas como sendo um valor perdido, para evitar possíveis problemas, é aconselhável copiar os valores de entrada para uma nova planilha para depois convertê-la para o formato *.csv*.

Figura 2. Dados externos

	A	B	
1	x	y	
2	A	12.3	
3	A	11.8	
4	A	13.6	
5	A	13	
6	A	12.5	
7	B	11.4	
8	B	10.9	
9	B	12.1	
10	B	11.2	
11	B	10.5	
12			

Para salvar a planilha de dados no formato *.csv*, deve-se clicar em “Arquivo”, e em “Salvar como...”, e selecionar a pasta onde o arquivo será salvo, que no presente trabalho é C:\Rdados. Em seguida, deve-se escolher o nome do arquivo que, no exemplo será entrada, e o formato a ser

salvo. No Excel, será “CSV (separado por vírgulas) (*.csv)” e clicar em “Salvar”, aparecerão duas mensagens. Clicar em “OK” na primeira e em “Sim” na segunda. No Calc, será “Texto CSV (.csv)” e após clicar em “Salvar”, serão exibidas três mensagens. Clicar em “Sim”, alterar a vírgula para ponto e vírgula no item “Delimitador de campo” e clicar em “OK”. A terceira será um aviso, em que se deve clicar em “OK”.

Antes de iniciar a entrada dos dados para o R, deve-se alterar a pasta de trabalho padrão para a pasta de trabalho C:\Rdados (Figura 3 e 4), utilizada neste material.

Figura 3

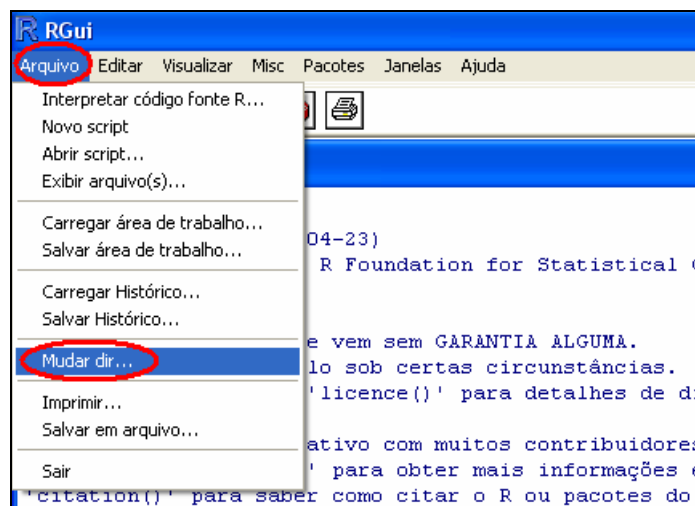
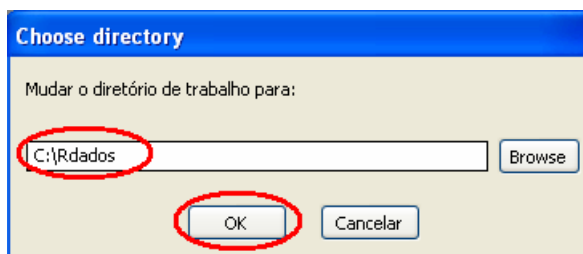


Figura 4



Após a alteração da pasta de trabalho, os passos para verificar se o arquivo entrada.csv está na pasta de trabalho e para ler os seus dados são:

`dir()`

```
[1] "entrada.csv"
```

```
dados1<-read.table(file="entrada.csv",sep=";",header=T,dec=".")
```

Os termos técnicos utilizados anteriormente são definidos a seguir:

- a) **dados1**: objeto pelo qual os dados lidos serão reconhecidos pelo R, cujo nome poderia ser outro;
- b) **<-**: sinal que atribui os dados lidos ao objeto dados1;
- c) **read.table**: função que lê o arquivo do tipo .csv;

d) **file**: informa o endereço e nome do arquivo a ser lido.

No exemplo, foi escrito apenas o nome do arquivo, pois ele está localizado na pasta de trabalho escolhida. Caso isto não ocorra, deve-se informar o endereço completo de onde o arquivo está, como por exemplo: **file="C:/windows/desktop/entrada.csv"**. Note que as barras utilizadas são invertidas em relação ao padrão do Windows.

Além do endereço e do nome, outros atributos do arquivo de dados devem ser especificados:

a) **sep**: informa qual é o separador de colunas e, embora o arquivo .csv indique separação por vírgula, ele será separado por ponto e vírgula que é o padrão utilizado pelo Windows em português;

b) **header**: informa se os nomes das colunas estão na primeira linha (TRUE) ou não (FALSE);

c) **dec**: informa qual é o separador decimal, para que o R converta para ponto.

Ao final, para ver se os dados foram lidos corretamente, deve-se digitar o nome **dados1** no console, cujos dados irão aparecer no formato Data Frame, que é o valor retornado pela função utilizada.

No R, existem algumas variações da função **read.table** e, para visualizá-las, basta pedir ajuda: ?read.table. No tópico **Usage** da janela de ajuda, podem ser vistas as outras formas de entrada de dados, cada uma delas com padrões diferentes para os valores dos argumentos. Quando se utiliza os argumentos na ordem correta, não é necessário informar seu nome. No exemplo acima, os dados poderiam ser lidos da seguinte forma:

```
dados1<-read.csv("entrada.csv",sep=";")
```

ou

```
dados1<-read.csv2("entrada.csv",dec=".")
```

Esta última será a utilizada durante as realizações das análises estatísticas no R.

Uma observação a ser feita, é que o R reconhece de forma diferenciada, as letras maiúsculas e minúsculas, tanto para os argumentos como para os nomes das funções ou objetos.

5. Manipulação de Dados

5.1. Vetor

As formas de se criar um vetor no R estão apresentadas na Tabela 6:

Tabela 6. Formas de criar um vetor

<code>c(1,2,3,4,5,6)</code>	concatena valores
<code>n1:n2</code>	seqüência de valores, de uma em uma unidade, do primeiro valor (n1) até o último (n2)
<code>rep(n1,r)</code>	cria um vetor contendo r vezes o valor n1
<code>seq(a,b,c)</code>	cria uma seqüência de números, no intervalo ab, separados por intervalos de c unidades
<code>gl(i,r)</code>	cria um vetor contendo i níveis representados pelos números de 1 a i, sendo que cada nível contém r repetições

Algumas manipulações de vetores estão apresentadas Tabela 7:

Tabela 7. Manipulação de vetores

<code>rounded(v)</code>	arredonda o número ou vetor de números v para o valor inteiro mais próximo, inferior ou superior
<code>length(v)</code>	retorna o tamanho do vetor v
<code>rank(v)</code>	exibe a classificação de cada valor do vetor v
<code>sort(v)</code>	exibe os valores do vetor v em ordem crescente

Como exemplo, considere as seguintes manipulações:

a) criar um vetor com os seis primeiros valores nulos e os seis últimos iguais a um:

```
v<-c(rep(0,6),rep(1,6)) # Concatenar dois vetores em um
```

```
v # Ver o vetor v
```

```
[1] 0 0 0 0 0 0 1 1 1 1 1 1
```

b) criar um vetor contendo uma seqüência de números de 1 a 15 espaçados por duas unidades:

```
v<-seq(1,15,2)
```

```
v # Ver o vetor v
```

```
[1] 1 3 5 7 9 11 13 15
```

c) criar um vetor de números divididos em cinco níveis, com cada nível contendo três repetições:

```
v<-gl(5,3)
```

```
v # Ver o vetor v
```

```
[1] 1 1 1 2 2 2 3 3 3 4 4 4 5 5 5
Levels: 1 2 3 4 5
```

5.2. Matriz

No R, deve-se criar, primeiramente, um vetor com os valores a serem inseridos na matriz. Depois, utiliza-se um comando para que os valores sejam dispostos em n linhas e m colunas. Como exemplo, será criada uma matriz com os valores de 1 a 6, com duas linhas e três colunas, como segue:

```
# Criar um vetor com os números 1, 2, 3, 4, 5 e 6
v<-c(1:6)
# Criar uma matriz com duas linhas e três colunas
matriz1<-matrix(data=v,nrow=2,ncol=3)
# Ver a matriz1
matriz1
```

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

Como pode-se observar, o R tem como padrão ler os dados seguindo a ordem por colunas. Caso os dados tivessem que ser ordenados por linha, tem-se a opção a seguir:

```
matriz2<-matrix(v,2,3,byrow=T)
```

Neste caso, a **matriz2** é criada com os dados armazenados no vetor **v**, ordenando-os por linha (byrow=TRUE). Além disso, os nomes dos três primeiros argumentos não foram utilizados, pois os seus valores estão na ordem correta.

```
# Ver a matriz2
matriz2
```

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

Além disso, podem ser acessadas partes da matriz, utilizando o colchete:

```
matriz2[2,3] # Ver o valor contido na linha 2 e coluna 3
```

```
[1] 6
```

```
matriz2[1,] # Ver a linha 1
```

```
[1] 1 2 3
```

```
matriz2[,2] # Ver a coluna 2
```

```
[1] 2 5
```

Também, é possível adicionar uma coluna ou uma linha à matriz já existente no R. Isso pode ser feito das seguintes formas:

a) utilizar a função **cbind()**, para coluna ou **rbind()**, para linha:

```
matriz3<-matrix(c(1:9),3,3)      # Criar a matriz
matriz3<-cbind(matriz3,c(10,11,12)) # Adicionar coluna como sendo a última
matriz3<-rbind(c(4,3,2,1),matriz3) # Adicionar linha como sendo a primeira
matriz3                          # Ver a matriz3
```

b) utilizar a função **fix()** (Figuras 5, 6 e 7):

```
matriz3<-matrix(c(1:9),3,3)      # Criar a matriz
fix(matriz3)                      # Abrir uma janela para editar a matriz
```

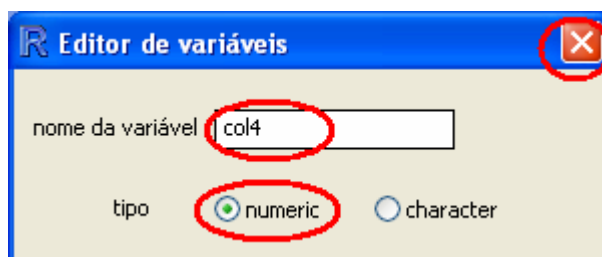
Na Figura 5, deve-se clicar sobre o nome da quarta coluna, onde será aberta uma nova janela (Figura 6). Nesta, deve-se clicar em “numeric” e alterar o nome para “col4”, com o objetivo de padronizar, e depois fechar a janela. Algumas restrições na criação de novos dados nesta janela são que as novas linhas são inseridas apenas para baixo e, as novas colunas, apenas para a direita, sendo que não é possível selecionar mais do que uma célula ao mesmo tempo. A coluna e a linha serão criadas digitando número por número (Figura 7). Após a digitação de todos os números, pode-se fechar a janela e ver a nova matriz.

Figura 5



	col1	col2	col3	var4	var5
1	1	4	7		
2	2	5	8		
3	3	6	9		
4					

Figura 6



R Editor de variáveis

nome da variável:

tipo: ☒ numeric ☐ character

Figura 7



	col1	col2	col3	col4
1	1	4	7	10
2	2	5	8	11
3	3	6	9	12
4	4	3	2	1
5	5	2	1	0

No R, existem algumas funções de operações com matrizes, apresentadas na Tabela 8:

Tabela 8. Operações com matrizes

solve(m)	calcula a inversa de uma matriz m
t(m)	calcula a transposta da matriz m
det(m)	calcula o determinante da matriz m
eigen(m)	calcula os autovalores e autovetores da matriz m
sum(diag(m))	calcula o traço da matriz m
M1 %*% M2	multiplicação entre as matrizes M1 e M2
kronecker(M1,M2)	produto de Kronecker: $M1 \otimes M2$

5.3. Data Frame

A entrada de dados externos não é a única forma de se ter um “Data Frame” no R. Eles podem também serem criados dentro do próprio programa. Primeiro, é preciso armazenar cada coluna em um vetor diferente no R, da seguinte forma:

a) utilizar a função `c()` :

```
X<-c("A","A","A","A","A","B","B","B","B","B") # usar aspas em caracteres
```

```
Y<-c(12.3,11.8,13.6,13,12.5,11.4,10.9,12.1,11.2,10.5)
```

b) utilizar a função `scan()` :

```
X<-scan(what=" ") # ao criar um vetor de caracteres, deve-se usar what=" "
```

```
1: A # pressionar ENTER e digitar o próximo valor
```

```
2: A # seguir até o último valor
```

```
Y<-scan()
```

```
1: 12.3 # pressionar ENTER e digitar o próximo número
```

```
2: 11.8 # seguir até o último número
```

Uma outra forma, é copiar uma coluna de dados de uma planilha e colar no elemento "1:" da função scan(), cuja coluna inteira será lida, automaticamente.

Agora que os vetores já foram criados em (a) ou (b), basta juntá-los em um "Data Frame" e remover as variáveis X e Y da memória, pois elas já estarão armazenadas no "Data Frame" criado, como segue:

```
dados<-data.frame(X,Y)    # criar o data frame e associá-lo ao nome dados
rm(X,Y)                   # remover os vetores X e Y
```

A forma de manipular um "Data Frame" é parecida com a de uma matriz. A diferença entre essas duas estruturas de dados é que o "Data Frame" permite trabalhar com valores numéricos e caracteres ao mesmo tempo e também é possível mudar os nomes das colunas e de ter acesso a cada uma delas através dos seus nomes. Como exemplo, tem-se:

```
dados          # Ver o data frame criado anteriormente
```

X	Y
A	12.3
A	11.8
A	13.6
A	13.0
A	12.5
B	11.4
B	10.9
B	12.1
B	11.2
B	10.5

```
names(dados)<-c("amostras","operador") # Alterar os nomes das colunas do data frame
```

```
attach(dados) # Função que permite utilizar as colunas pelos nomes delas
```

```
operador      # Ver os dados contidos na coluna "operador"
```

[1] 12.3 11.8 13.6 13.0 12.5 11.4 10.9 12.1 11.2 10.5

6. Características de Qualidade

No controle de qualidade, muitas são as características estudadas que, de modo geral, podem ser divididas em variáveis aleatórias discretas ou contínuas. No texto, elas serão denominadas de variáveis Ys, respostas, dependentes, explicadas ou, simplesmente, efeitos. Quando uma variável Y é avaliada, observa-se uma variabilidade em torno da média, que pode ser "pequena", devida à variância aleatória provocada por fatores ou causas não controláveis, ou "grande", devida, além da variância aleatória, à variância não aleatória provocada por fatores ou causas controláveis, que serão denominados de variáveis Xs, independentes, classificatórias, explicativas ou preditoras. No controle estatístico, é muito comum a denominação dos fatores

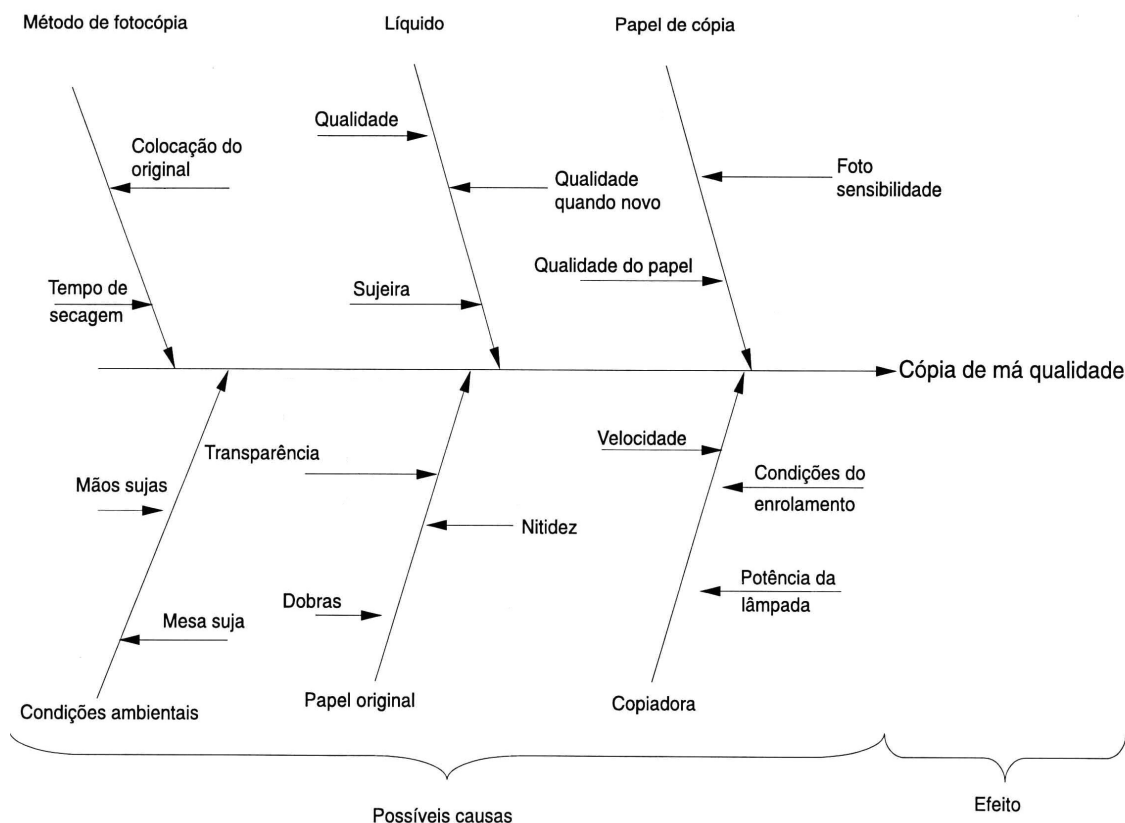
6M's (mão-de-obra, matéria-prima, máquina, método, meio-ambiente e medida), além de outros, às variáveis Xs.

As variáveis Xs podem apresentar níveis qualitativos ou quantitativos. No primeiro caso, pode-se citar como exemplo, a variável mão-de-obra (X) com quatro níveis ($x_1 = 1 = \text{operador 1}$, $x_2 = 2 = \text{operador 2}$, $x_3 = 3 = \text{operador 3}$ e $x_4 = 4 = \text{operador 4}$) e, no segundo, a variável temperatura (XX) com quatro níveis ($xx_1 = 10$, $xx_2 = 15$, $xx_3 = 20$ e $xx_4 = 25^\circ \text{C}$).

Então, quais serão as variáveis Xs e Ys estudadas no experimento? Uma forma de planejamento pode ser feita por meio do diagrama de causa e efeito, também conhecido como diagrama de espinha de peixe ou de Ishikawa.

Como exemplo do diagrama de Ishikawa, considere uma pequena empresa do ramo de cópias e impressões. Depois de ter passado por um período muito bom de crescimento, devido ao seu preço diferenciado em relação ao dos concorrentes, surgiu uma fase de muitas dificuldades. O dono da empresa, percebendo que estava perdendo a sua clientela, tentou identificar o problema que vinha acontecendo. Preparou alguns questionários e os aplicou diretamente aos seus clientes, na maioria estudantes. Depois de respondidos e analisados por ele próprio, a conclusão foi que a qualidade das cópias das outras empresas eram bastante superiores. Sem perder tempo, tratou de convocar uma reunião com todos os funcionários, de forma a buscar as causas do problema observado, ou seja, a cópia de má qualidade (Figura 8)

Figura 8. Diagrama de causas primárias e secundárias



A construção do diagrama de Ishikawa no R será vista no próximo capítulo.