Universidade Federal de Viçosa
Centro de Ciências Exatas e Tecnológicas
Departamento de Estatística

AS SETE FERRAMENTAS ESTATÍSTICAS DA QUALIDADE

GUIA PRÁTICO DO *SOFTWARE R*

Izabela Salvador Carvalhais
José Ivo Ribeiro Júnior

Prefácio

Este manual se destina a estudantes e profissionais de empresas que possuem o interesse nos procedimentos que são utilizados no *software R*, para as realizações de análises do controle estatístico de processos (CEP).

No primeiro capítulo, nomeado de Informações Gerais, são abordados assuntos gerais a respeito do *software R*, como seu *download* e instalação, telas e botões principais, funções e seus significados, pacotes, entrada de dados, entre outros.

No segundo capítulo são abordadas as medidas de posição e variação, essenciais para a estatística e para o desenvolvimento das ferramentas a serem abordadas posteriormente.

No terceiro capítulo, tem-se como assunto a personalização dos gráficos no *R*, um recurso muito interessante do *software* e que será utilizado posteriormente nas construções de algumas das ferramentas estatísticas abordadas.

Do quarto até o décimo capítulo são abordadas as ferramentas estatísticas escolhidas dentre as sete ferramentas da qualidade. No quarto e quinto capítulos é abordado um passo a passo de como construir o histograma e o *box-plot* padrões do *software R*. Além disso, são mostrados alguns exemplos de personalizações para cada um deles. No sexto capítulo é ensinado um passo a passo para construir o diagrama de dispersão padrão, tal como o personalizado. No sétimo capítulo, a ferramenta abordada é o gráfico de Pareto, seguindo o mesmo procedimento das demais. Primeiro é ensinado um passo a passo para a construção do gráfico padrão e, posteriormente, para a do personalizado. O oitavo capítulo aborda o diagrama de causa e efeito, porém para este não há a possibilidade de personalização. Portanto, apenas um passo a passo para a construção do diagrama padrão é ensinado. Por fim, o nono e o décimo capítulos abordam as cartas (gráficos) de controle por atributos e por variáveis de *Shewhart*, ensinando um passo a passo para construir cada um delas, porém sem as respectivas personalizações.

Introdução

Diante de um cenário altamente globalizado, em que as informações navegam, cada vez mais, com mais facilidade e os avanços tecnológicos são cada dia maiores, o mercado tem se tornado altamente competitivo. Assim, é de extrema importância que a empresa possua diversas estratégias e ferramentas, a fim de garantir a máxima satisfação do cliente e se torne, portanto, mais competitiva.

Uma das estratégias competitivas de uma empresa relacionada à satisfação do cliente é a gestão da qualidade, metodologia de gerenciamento responsável pelo controle de todos os processos de uma empresa. Ela visa diminuição da variabilidade e, consequentemente, padronização, o que promove produtos e serviços conforme especificados.

Para que a visualização, a análise e o controle dos processos sejam possíveis, existe uma série de estatísticas conhecidas como ferramentas da qualidade. Porém, pode-se dizer que delas, apenas sete são essenciais: folhas de verificação, gráfico de Pareto, diagrama de causa e efeito, gráficos de controle, diagrama de dispersão, histograma e estratificação.

Atualmente existe uma série de *softwares* que são capazes de realizar tais análises estatísticas (*Applied Stats*, *InfinityQS*, *Minitab*, *R*, SAS, *Statistica*, *S-Plus*, *Wincep* e, em uma visão mais restrita, o *Excel*). No entanto, apenas o *software R* é livre, ou seja, não cobra pela licença. Sendo assim, optar pela sua utilização é uma escolha interessante do ponto de vista financeiro. Por outro lado, devido a sua interface gráfica ser um pouco mais complexa, muitas vezes ele é esquecido.

O *software R* foi desenvolvido inicialmente em 1996 pelos professores Ross Ihaka e Robert Gentleman, da Universidade de Auckland na Nova Zelândia e está disponível sob a licença GNU GPL (Licença Pública Geral), que confere a ele, ser um *software* livre e de código aberto. Até o momento, ele é oferecido para os seguintes sistemas operacionais: *Linux*, *MacOS X* e *Windows*.

O funcionamento do *R* se dá através da utilização de uma série de pacotes, os quais permitem a realização de específicas e diversas análises estatísticas. Existem três tipos de pacotes: os básicos, que são inclusos ao baixar o programa e carregados automaticamente; os inclusos ao baixar e não carregados automaticamente; e os criados pelos usuários, os quais não são inclusos e nem carregados automaticamente.

Para o CEP, o *R* apresenta quatro pacotes: `qcc` (SCRUCCA, 2004), `CEPpt` (BASTOS; FERREIRA, 2012), `qualityTools` (ROTH, 2016) e `SixSigma` (CANO et al., 2018). Nesse manual o foco foi no pacote `qcc` (SCRUCCA, 2004). Isso se deveu à sua facilidade em gerar os gráficos de controle de *Shewhart* e de analisar a capacidade do processo. Além disso, ele possibilita gerar os gráficos de controle CUSUM e EWMA, as curvas características operacionais, o gráfico de Pareto, o diagrama de causa e efeito e os gráficos de controle multivariado.

Nesse sentido, o objetivo desse manual foi contribuir para a disseminação do *software R*, principalmente entre os estudantes de graduação, ao trazer um passo a passo com uma linguagem clara, objetiva e didática, de como realizar as análises estatísticas do CEP. Vale ressaltar que o fato de o *R* ser um *software* livre não o torna inferior aos outros. Muito opostamente a isso existem *softwares* que geram gráficos e resultados muito inferiores e são pagos.

Apesar da interface gráfica complexa do *R*, ele possui uma diversidade imensa de pacotes e de personalizações de gráficos, o que possibilita os seus resultados serem visualmente excelentes. Isso significa que realmente vale a pena conhecê-lo.

Capítulo 1

Informações Gerais

Download e Instalação

Para realizar o *download* do *software R* no sistema operacional *Windows*, deve-se, primeiramente, acessar o *site www.r-project.org* e clicar em *CRAN* ou *CRAN mirror* (Figura 1.1). Após essa etapa, aparecerá uma série de *links* que são iguais entre si (espelhos). Portanto, pode-se clicar em qualquer um deles. Contudo, indica-se a escolha de um servidor do Brasil e da cidade mais próxima (Figura 1.2).

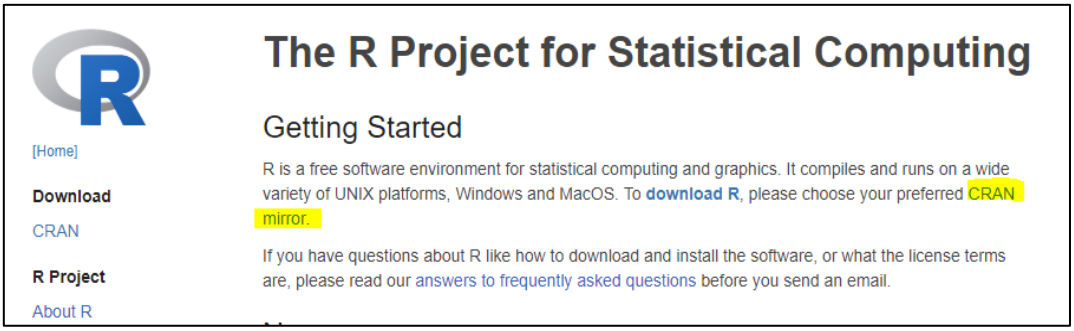


Figura 1.1 - Tela inicial do *site* do *R*.

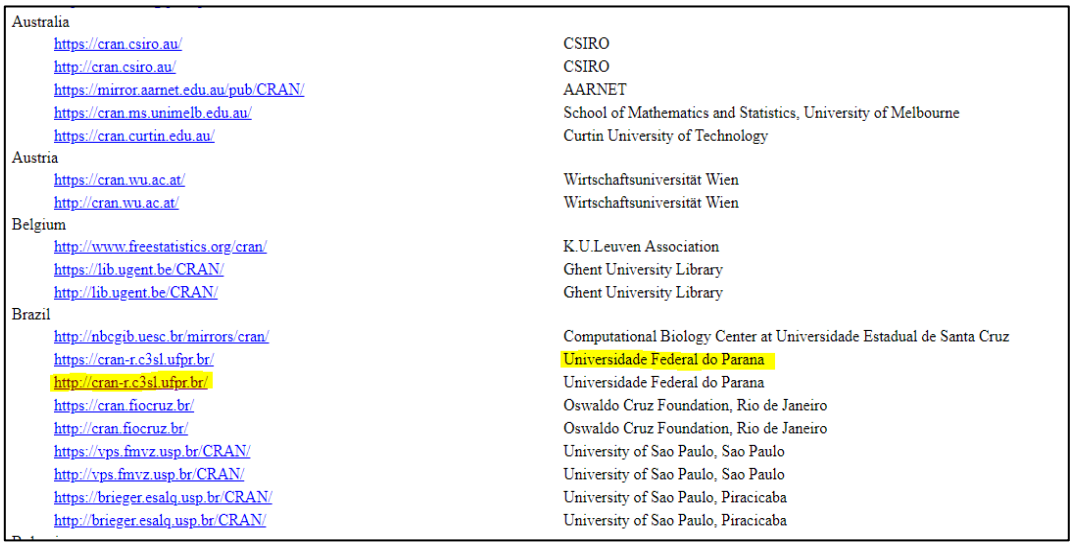


Figura 1.2 - Tela para escolha do servidor.

Para dar continuidade ao *download*, realizam-se as seguintes etapas:

clicar no *link* correspondente ao seu sistema operacional *Linux*, *Mac* ou *Windows* (Figura 1.3);

clicar no *link base* abaixo de *Subdirectories* para ter acesso ao *download* do instalador do *R* (Figura 1.4); e

clicar em *Download R 3.5.1 for Windows*, em que a numeração *3.5.1* representa a versão do *R* disponível naquele momento (Figura 1.5).

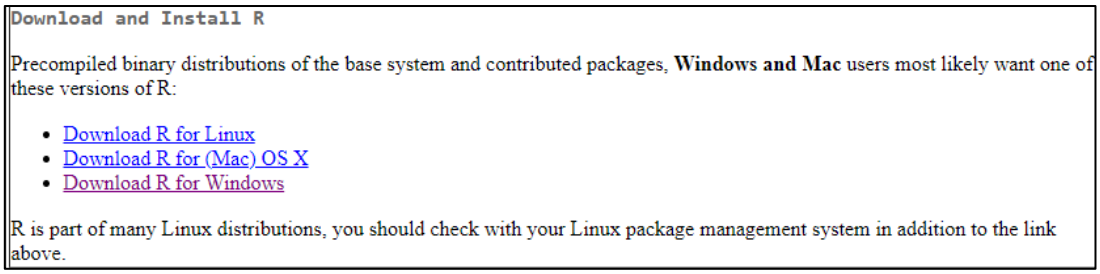


Figura 1.3 - Tela para escolha do sistema operacional.

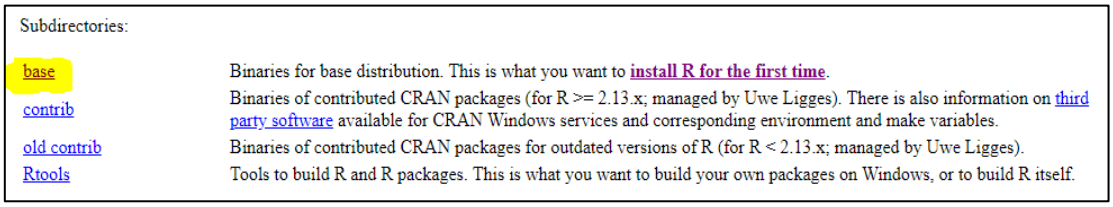


Figura 1.4 - Tela para prosseguir com o *download* do R.

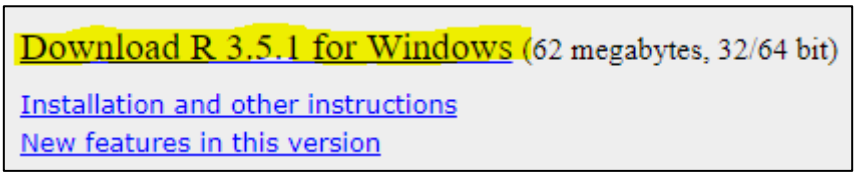


Figura 1.5 - Tela com o *link* para dar início ao *download* do *R*.

Após serem seguidos os três passos anteriores, o *download* do *software R* se iniciará e, ao finalizar, deverá ser realizada a instalação.

Pacotes

Como citado anteriormente, o *R* é um *software* a base de pacotes, compostos por um conjunto de funções que possibilitam ou facilitam a realização das diversas análises estatísticas. Além disso, eles possuem manuais para as suas funções e, até mesmo, demonstrações de execução.

Ao baixar o *software R*, alguns pacotes já vêm instalados, pois são pacotes essenciais e básicos para o funcionamento do programa. Porém, existem muitos pacotes além desses, capazes de realizar as mais diversas análises estatísticas.

Um deles é o `qcc` (SCRUCCA, 2004), ideal para realizar análises estatísticas do CEP. Ele conta com uma série de comandos que possibilitam as construções de ferramentas essenciais, tais como: gráfico de Pareto, diagrama de causa e efeito, gráficos de controle de *Shewhart*, CUSUM e EWMA e análise da capacidade do processo. Devido a isso, o `qcc` (SCRUCCA, 2004) é o pacote mais utilizado no CEP.

Os pacotes adicionais a serem instalados podem ser encontrados no próprio programa. Contudo, é necessário que o usuário esteja conectado à *internet*. Existem duas possibilidades para tal: clicar no botão *Pacotes* ou digitar o respectivo comando no *console*.

Para a primeira opção, são listados os seguintes passos:

clicar no botão *Pacotes* (Figura 1.6);

clicar no botão *Carregar Pacotes* (Figura 1.7);

selecionar na janela com as opções dos pacotes disponíveis, o de interesse, como por exemplo, o `qcc` (SCRUCCA, 2004) (Figura 1.8); e

clicar sobre o pacote selecionado e, após carregado, aparecerá um aviso informando se foi carregado com sucesso ou se houve erro (Figura 1.9).

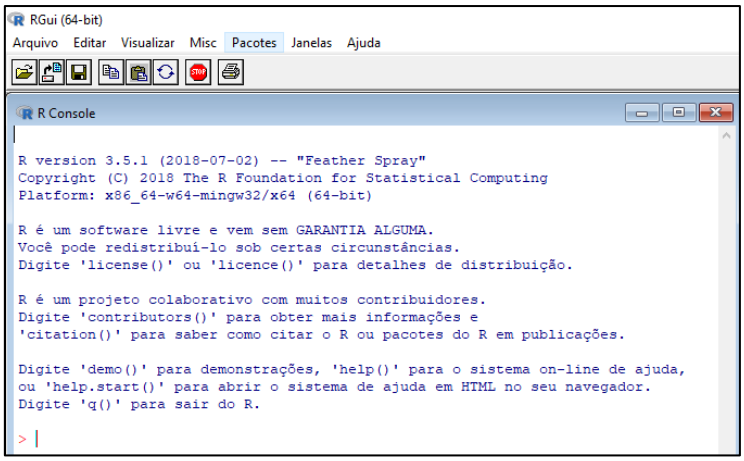


Figura 1.6 - Tela com o botão *Pacotes*.

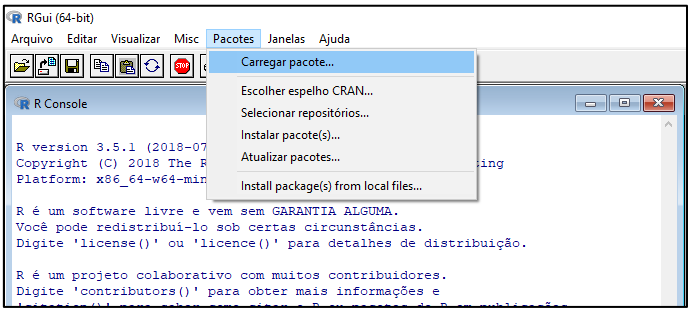


Figura 1.7 - Tela com a opção para carregar os pacotes.

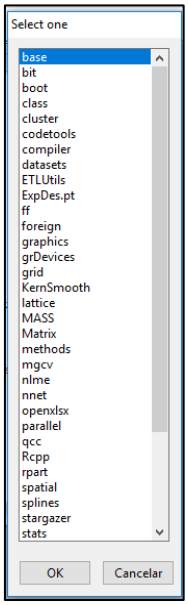


Figura 1.8 - Tela com alguns dos pacotes disponíveis.

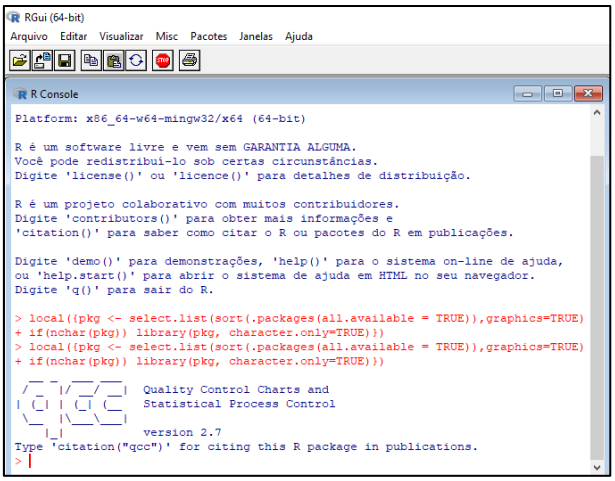


Figura 1.9 - Tela com as informações após a instalação do pacote `qcc`.

E para a segunda opção, é mencionado o seguinte passo: executar a função `install.packages ("pacote")` ou a função `install.packages (c ("pacote1", "pacote2", ...))`, sendo que a última junta e instala todos os pacotes, entre aspas, escolhidos.

Feito isso, aparecerá a janela *Secure CRAN mirrors*, na qual se deve selecionar, de preferência, alguma opção correspondente à localidade mais próxima da sua. Ao selecioná-la e clicar em *OK* o pacote será instalado (Figura 1.10).

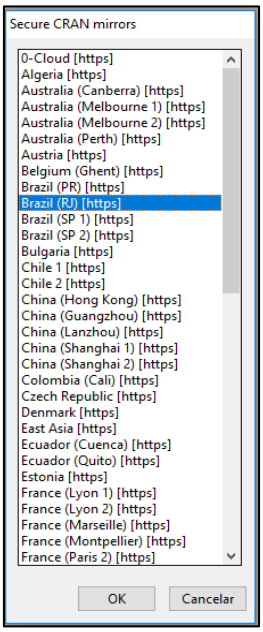


Figura 1.10 - Tela com os *CRAN mirrors* disponíveis.

Telas Principais

A janela inicial do *R* é formada, basicamente, por uma interface de usuário gráfica (GUI), composta por um *menu* contendo ícones e, pelo *console*, que é uma interface de linha de comando (CLI) onde são digitados os comandos a serem executados para gerar as análises estatísticas pretendidas (Figura 1.11).

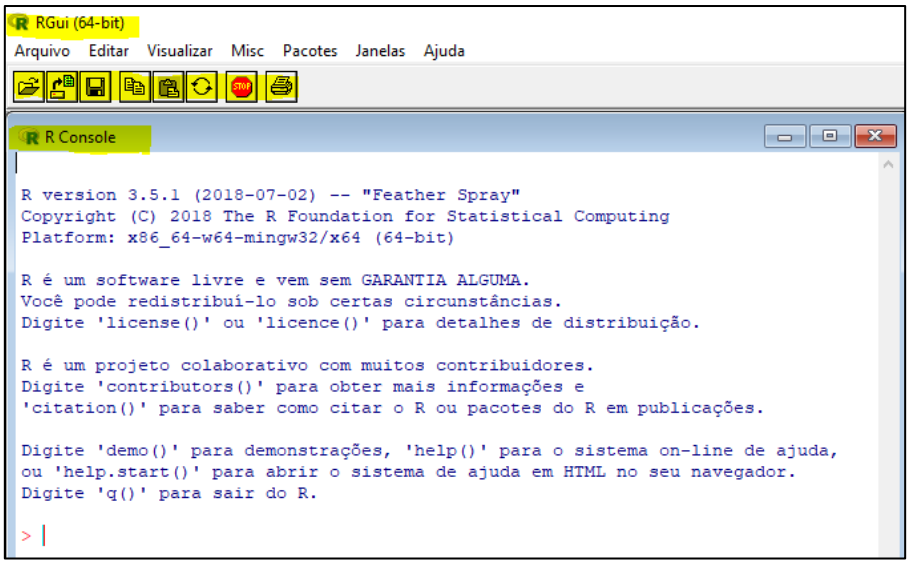


Figura 1.11 - *Console* do *R*.

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

O *software 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 os comandos criados.

Além do *console*, existe no programa *R* um editor de *script*, o qual permite o armazenamento dos comandos para utilização posterior. Ele funciona como uma espécie de rascunho, no qual se pode digitar os comandos e fazer os comentários sem serem executados. Além disso, ele poderá ser salvo ao final, a fim de ser utilizado em situações posteriores.

Para que os comandos digitados no editor de *script* sejam executados no *console*, deve-se clicar no botão direito do mouse e selecionar a opção *run line or selection* ou utilizar os atalhos *Ctrl + R* ou *F5* do teclado. Para fazer comentários, deve-se digitar *hashtag* (#) antes de cada um deles, pois o R não executa textos escritos após este sinal.

Para acessar o editor de *script*, basta clicar em *Arquivo* (Figura 1.12) e, posteriormente, em *Novo script* (Figura 1.13). Caso já se tenha um *script* pronto, existe também a opção de *Abrir script*. Feito isso, aparecerá uma tela em branco nomeada como *Editor R* (Figura 1.14). Para alinhá-la à tela do *R*, deve-se clicar no botão *Janelas* (Figura 1.15) e, posteriormente, em *Dividir Lado a Lado* ou *Dividir Horizontal* (Figura 1.16). Como preferência pessoal, tem-se o resultado das janelas divididas lado a lado (Figura 1.17).

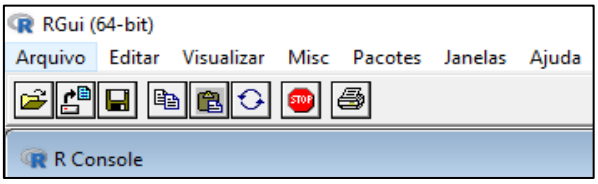


Figura 1.12 - Tela com o botão *Arquivo*.

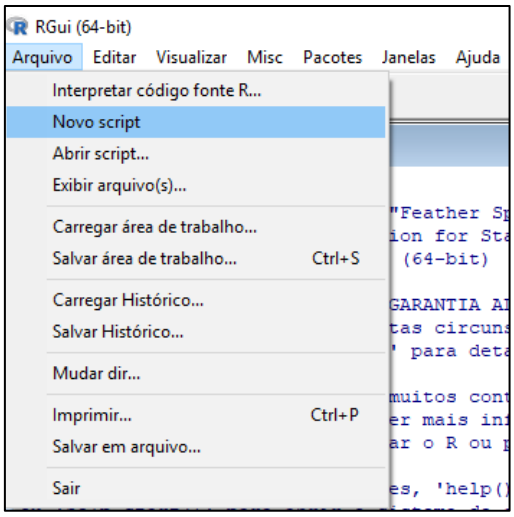


Figura 1.13 - Tela com o botão *Novo script*.

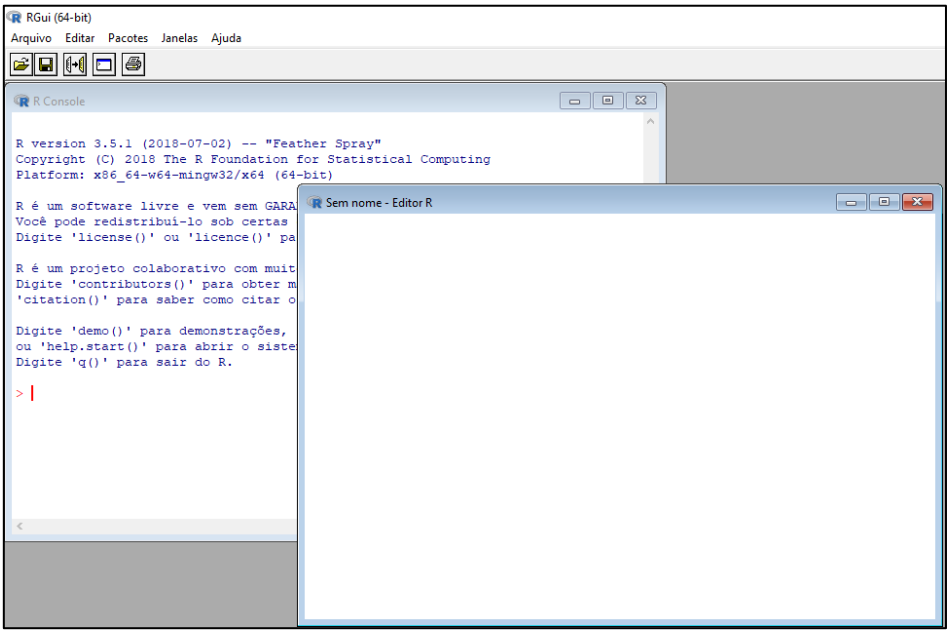


Figura 1.14 - Tela do editor de *script*.

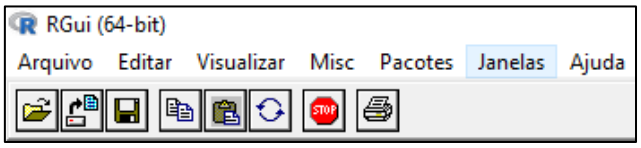


Figura 1.15 - Tela com o botão *Janelas*.

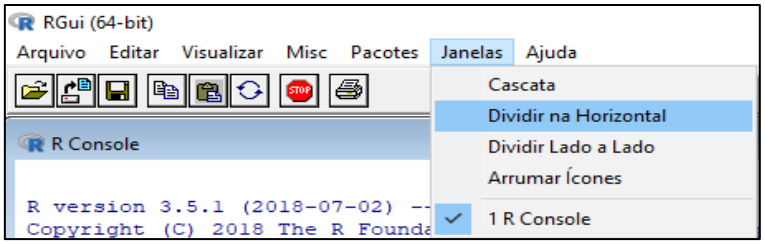


Figura 1.16 - Tela com as opções do botão *Janelas*.

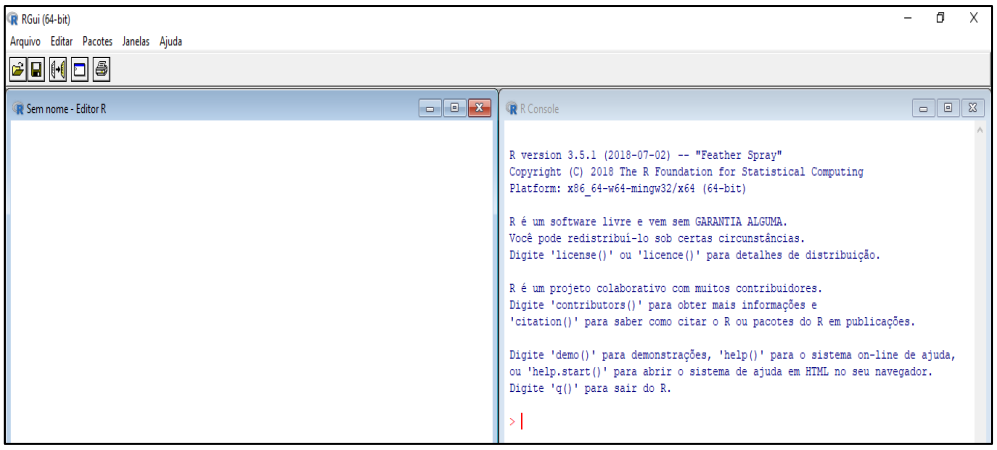


Figura 1.17 - *Console* e editor de *script* divididos lado a lado.

Funções

Como já mencionado, as análises estatísticas são feitas através de pacotes contendo funções. Além das funções relacionadas aos pacotes básicos (Tabela 1.1), existem as relacionadas aos pacotes adicionais do *R* (Tabela 1.2).

Tabela 1.1 - Funções básicas

Função	Comentário
`dir ()`	lista todos os arquivos na pasta de trabalho atual
`library (nome.do.`*`pacote`*`)`	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 (`*`objeto`*`)`	transforma o objeto em fator, ou seja, o divide em níveis
`attach (`*`DataFrame`*`)`	permite que cada coluna do *data frame* seja reconhecida como um objeto
`detach (DataFrame)`	desfaz a função `attach`, sendo aconselhável utilizá-la quando o *data frame* não for mais necessário, para evitar possíveis conflitos de variáveis com o mesmo nome

Tabela 1.2 - Funções relacionadas aos pacotes adicionais

Função	Comentário
`summary (packageStatus())`	mostra um resumo de todos os pacotes 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
`remove.packages (c ("pacote1", "pacote2", ...))`	remove o(s) pacote(s) entre aspas

Ajuda

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 1.3.

Tabela 1.3 - Funções de ajuda

Função	Comentário
`help (package = nome.do.pacote)`	fornece informações sobre os pacotes básicos ou adicionais ativos na sessão
`help.start ()`	ajuda pela *internet*
`help.search ("termo")`	pesquisa o termo, entre aspas, em todos os pacotes ativos
`apropos ("função")` ou `apropos ("objeto")`	pesquisa a função ou o objeto, entre aspas, em todos os pacotes ativos
`example (nome.da.funcao)`	mostra exemplos da função, entre parênteses, quando possuir
`demo (nome.do.pacote)`	exibe demonstrações de algumas funções contidas no pacote ativo, quando possuí-las
`ls ("package: nome.do.pacote")`	exibe todas as funções do pacote
`help (nome.da.funcao)` ou `?nome.da.funcao`	fornecem informações da função

As janelas de ajuda abertas pela função `help` ou pelo operador `?`, podem possuir, de modo geral, até nove tópicos, como seguem:

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 possuir alguma variação, ela também será mostrada;

Arguments: descreve os argumentos possíveis de serem usados, sendo que os obrigatórios vêm primeiro; ao final deste tópico poderão aparecer três pontos, que indicam 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;

Value: descreve qual é o tipo de objeto que a função retorna;

Note: observação final que pode conter dicas para melhorar o funcionamento da função, como informações a respeito de possíveis diferenças entre a utilização da função em diferentes versões do *R*;

References: exibe referências bibliográficas relacionadas à função;

See Also: exibe algumas funções que têm alguma relação com a função em questão; e

Examples: exibe exemplos de como usar a função, que podem ser copiados e colados no *console* para as suas respectivas execuções.

Entrada de Dados

O arquivo de dados ou a matriz que aceita valores quantitativos e qualitativos numa mesma coluna (*data frame*) pode ser criado diretamente dentro do próprio *R* ou pela importação de outro programa, como por exemplo, o *Microsoft Excel.*

No caso da utilização do *Excel,* a planilha contendo os dados deve estar organizada corretamente, com números alinhados à direita e níveis qualitativos alinhados à esquerda. Além disso, na primeira linha (cabeçalho) da planilha, os nomes das variáveis devem ser introduzidos com letras minúsculas, pois há distinção no *R* entre letras maiúsculas e minúsculas. Posteriormente, salva-se cada arquivo de dados, separadamente, com a extensão *xlsx* (arquivo do tipo *Pasta de Trabalho do Excel*). Um dos pacotes responsáveis por abrir os arquivos importados do *Excel* é o `openxlsx` (WALKER, 2019). Sendo assim, ele deverá ser baixado e instalado anteriormente ao processo de importação dos dados.

Para fazer a importação dos dados do *Excel* para o *R*, deve-se, no *console* do *software R*:

clicar em *Arquivo*; e

clicar em *Mudar dir* (Figura 1.18) para abrir uma janela com as pastas de trabalho do seu computador (Figura 1.19); e

após selecionar a pasta de trabalho a ser salvo o arquivo de dados (*.xlsx*), clicar em *OK*.

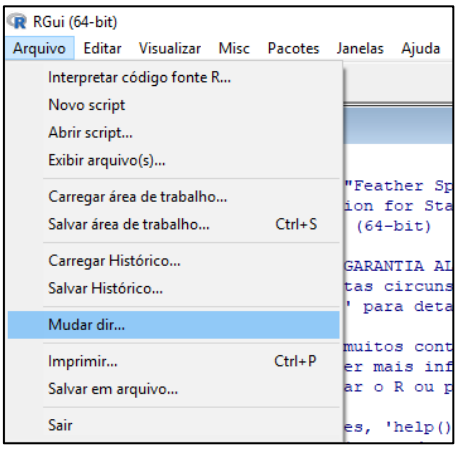


Figura 1.18 - Tela com a opção *Arquivo* → *Mudar dir*.

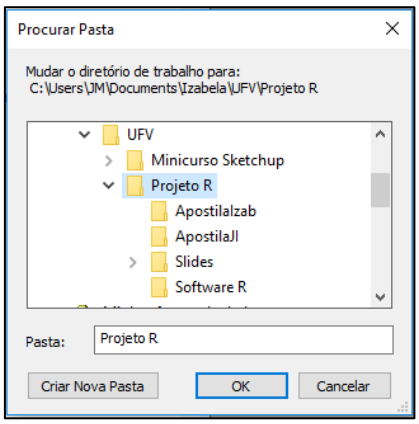


Figura 1.19 - Janela com a pasta de trabalho a ser selecionada.

Posteriormente, para a entrada de dados em si, devem ser executados, sequencialmente, os seguintes comandos no *console* do *software R*:

```
> library (openxlsx)
> dNomeDoArquivo = read.xlsx ("NomeDoArquivo.xlsx")
> attach (dNomeDoArquivo)
> dNomeDoArquivo
```

Nesse manual, todo objeto criado no *software R* e que armazena um *data frame* (conjunto de dados) importado do *Excel*, apresentará, antes do nome do respectivo arquivo de dados, a letra d de dados.

Capítulo 2

Medidas de Posição e Variação

Introdução

A estatística descritiva trata da descrição da amostra, isto é, da sua interpretação da amostra por meio de medidas e, ou, gráficos que resumem, em termos de posição, de variação e de formato, a distribuição dos n dados amostrais. Depois e com base nessa interpretação, é feita a inferência estatística para os dados populacionais.

Para calcular as medidas de posição e variação, considera-se como exemplo os valores observados de uma variável-resposta (Y) avaliada em uma amostra aleatória com 25 repetições (RIBEIRO JÚNIOR, 2012) (Tabela 2.1).

Tabela 2.1 - Arquivo de dados *cap2e1.xlsx*

	A	B	C
1	x	rept	y
2	1	1	68,44
3	1	2	68,81
4	1	3	68,98
5	1	4	68,99
6	1	5	69,01
7	1	6	69,14
8	1	7	69,16
9	1	8	69,19
10	1	9	69,31
11	1	10	69,32
12	1	11	69,75
13	1	12	69,82
14	1	13	69,96
15	1	14	69,96
16	1	15	70,31
17	1	16	70,62
18	1	17	70,65
19	1	18	70,7
20	1	19	70,96
21	1	20	70,98
22	1	21	71,1
23	1	22	71,12
24	1	23	71,19
25	1	24	71,27
26	1	25	71,72

Entrada de Dados

Para calcular as medidas, deve-se, primeiro, realizar a entrada de dados. Para isso, deve-se mudar o diretório para a pasta de trabalho onde se encontra o arquivo de dados *cap2e1.xlsx* e digitar os seguintes comandos no *R*:

```
> library (openxlsx)
> dcap2e1 = read.xlsx ("cap2e1.xlsx")
> attach (dcap2e1)
> dcap2e1
```

Funções

Como se sabe, existem diversas medidas de posição e variação. Portanto, existe uma função referente a cada uma delas. Para a variável-resposta Y, têm-se:

`min (y)`: menor valor;

`max (y)`: maior valor;

`range (y)`: menor e maior valor;

`mean (y)`: média;

`median (y)`: mediana;

`var (y)`: variância;

`sd (y)`: desvio-padrão; e

`summary (y)`: retorna todos os valores listados anteriormente de uma só vez.

Considerando o exemplo de aplicação (Tabela 2.1), foram obtidos os seguintes resultados:

```
> summary (y)
 Min.  1st Qu.  Median   Mean  3rd Qu.   Max.
68.44    69.16   69.96  70.02    70.96  71.72

> min (y) # menor valor
[1] 68.44

> max (y) # maior valor
[1] 71.72

> range (y)
[1] 68.44 71.72

> mean (y) # média
[1] 70.0184

> median (y) # mediana
[1] 69.96

> var (y) # variância
[1] 0.9061057

> sd (y) # desvio-padrão
[1] 0.9518958

> 100 * sd (y) / mean (y) # coeficiente de variação
[1] 1.359494

> sd (y) / sqrt (length (y)) # erro-padrão da média
[1] 0.1903792
```

As telas contendo as execuções e os resultados do exemplo da Tabela 2.1, encontram-se nas Figuras 2.1 e 2.2, respectivamente. À direita das imagens, tem-se o editor de *script* e, à esquerda, o *console* do *R*, onde os comandos (em vermelho) foram executados e os resultados mostrados em azul.

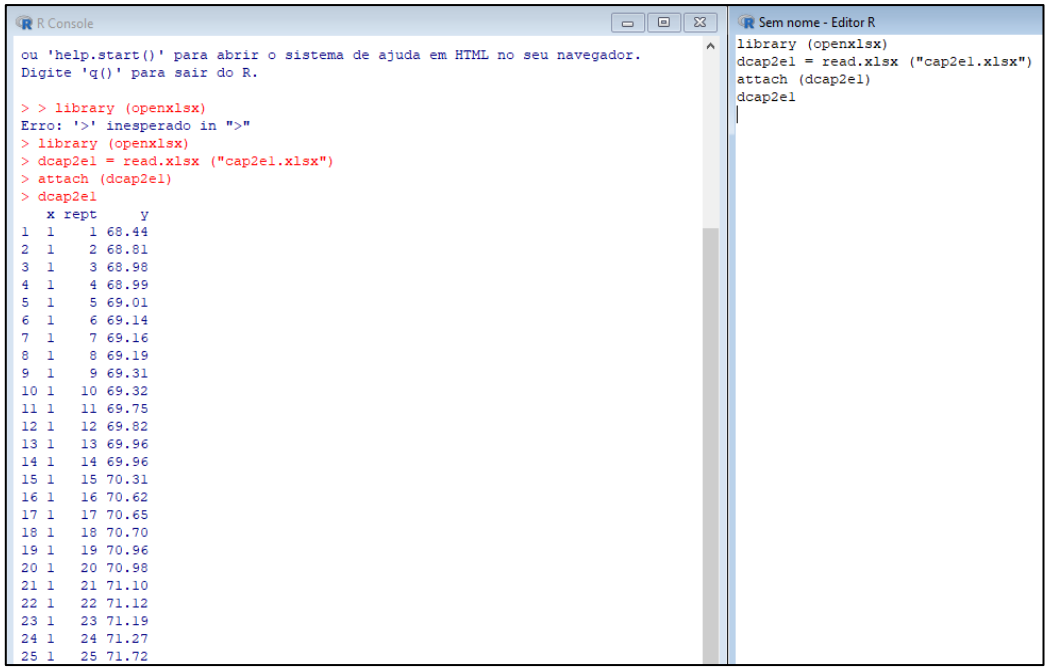


Figura 2.1 - Telas da entrada de dados e do editor de *script*.

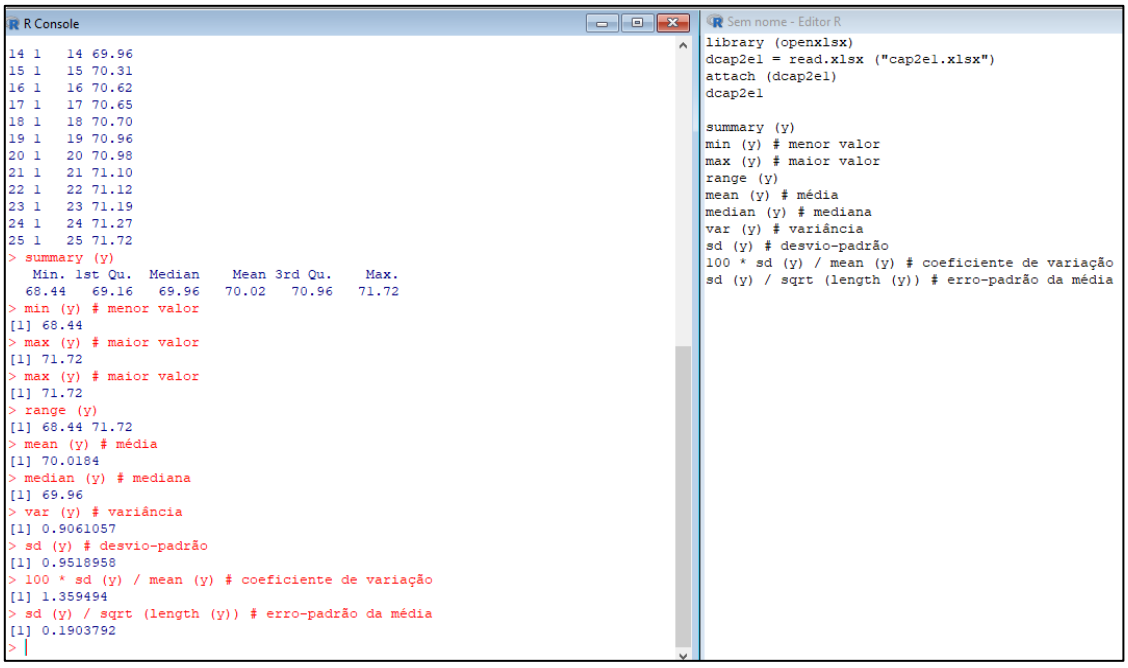


Figura 2.2 - Telas dos resultados e do editor de *script*.

Capítulo 3

Personalização dos Gráficos

Introdução

Uma das vantagens do *software R* é a oportunidade de construir e personalizar uma série de gráficos, tais como: gráfico de colunas ou de barras (`barplot`), histograma (`hist`), *box-plot* (`boxplot`) e diagrama de dispersão (`plot`). A personalização da aparência possibilita mudar a cor de alguns elementos gráficos, incluir títulos, ajustar a cor, o tamanho e a fonte dos textos, alterar a cor e o tipo das linhas e dos símbolos e adicionar outros itens.

A maioria das personalizações de um gráfico é feita com argumentos adicionais para a função `plot`. Entretanto, muitos desses argumentos também podem ser usados nas funções `barplot`, `hist` e `boxplot`.

Como exemplo, considere a construção e a personalização de um diagrama de dispersão pela função `plot` (Tabela 3.1).

Tabela 3.1 - Arquivo de dados *cap3e1.xlsx*

	A	B
1	x	y
2	1	55
3	2	65,5
4	3	75
5	4	85,5
6	5	95

Entrada de Dados

O primeiro passo, antes de tudo, é realizar a entrada desses dados no *software R*. Para isso, deve-se mudar o diretório para a pasta de trabalho onde se encontra o arquivo de dados *cap3e1.xlsx* e digitar os seguintes comandos:

```
> library (openxlsx)
> dcap3e1 = read.xlsx ("cap3e1.xlsx")
> attach (dcap3e1)
> dcap3e1
```

Função *plot*

No *software R*, as variáveis dos eixos x e y do diagrama de dispersão podem ser adicionadas de duas maneiras:

`plot (x, y)` (cartesiana); ou

`plot (y ~ x)` (fórmula).

Neste *software*, não há a necessidade de inserir os argumentos nas suas definições padrões.

```
> help (plot)
plot (x, y, type = "p", xlim = NULL, ylim = NULL, main = NULL,
sub = NULL, xlab = NULL, ylab = NULL, axes = TRUE, ...)
plot (y ~ x, type = "p", xlim = NULL, ylim = NULL, main = NULL,
sub = NULL, xlab = NULL, ylab = NULL, axes = TRUE, ...)
```

Na função `plot`, os argumentos têm os seguintes significados:

`type = "p"`: plota pontos associados aos pares de valores das variáveis dos eixos x e y;

`xlim = NULL`: adiciona limites inferior e superior não especificados aos valores do eixo x;

`ylim = NULL`: adiciona limites inferior e superior não especificados aos valores do eixo y;

`main = NULL` ou `main = ""`: não mostra o título do gráfico;

`sub = NULL` ou `sub = ""`: não mostra o subtítulo do gráfico;

`xlab = NULL`: mostra o rótulo do eixo x com o próprio nome designado à sua variável;

`ylab = NULL`: mostra o rótulo do eixo y com o próprio nome designado à sua variável; e

`axes = TRUE` ou `axes = T`: plota os eixos x e y e mostra os seus valores.

Executando-se a função com os seus argumentos padrões, tem-se como resultado o diagrama de dispersão padrão do *R* (Figura 3.1):

```
> plot (x, y)
```

```
> plot (y ~ x)
```

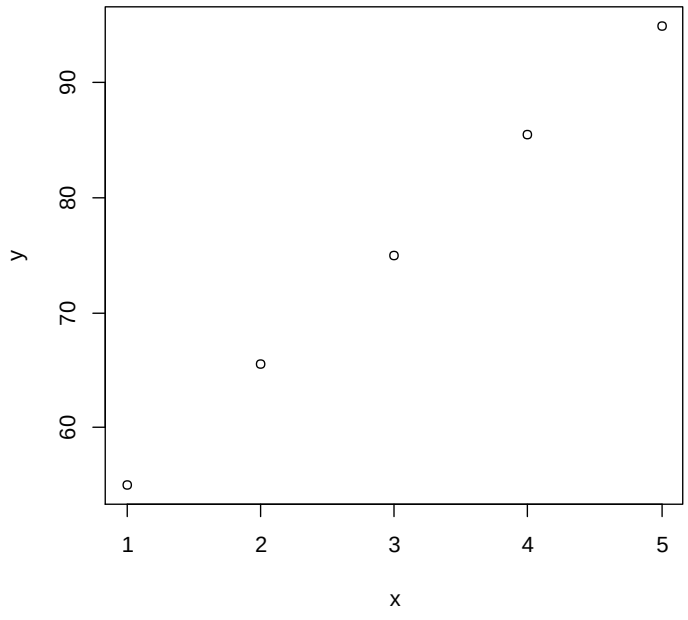


Figura 3.1 - Diagrama de dispersão padrão do *R*.

Desse modo, com o objetivo de personalizar o diagrama de dispersão para que ele possua uma melhor qualidade, é preciso adicionar à função `plot`, os seguintes argumentos:

`type = "n"`: não plota os pares de valores das variáveis dos eixos x e y;

`xlim = c (LIx, LSx)`: adiciona limites inferior e superior especificados aos valores do eixo x;

`ylim = c (LIy, LSy)`: adiciona limites inferior e superior especificados aos valores do eixo y;

`xlab = ""`: não mostra o rótulo do eixo x;

`ylab = ""`: não mostra o rótulo do eixo y; e

`axes = FALSE` ou `axes = F`: não plota os eixos x e y e nem mostra os seus valores.

Desse modo e sem as adições dos argumentos `type` e `axes` nas suas formas padrões, têm-se:

```
> plot (x, y, type = "n", axes = F)
```

O resultado encontra-se na Figura 3.2.

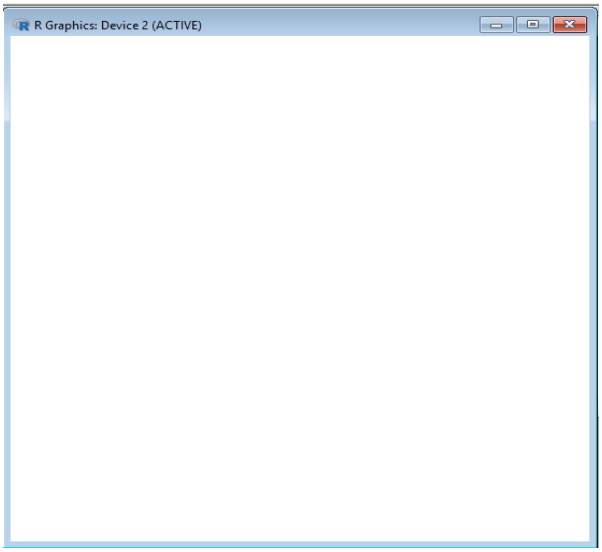


Figura 3.2 - Resultado inicial da personalização do diagrama de dispersão.

Como se pode perceber, o resultado é uma página em branco. Isso possibilita, portanto, adicionar os argumentos nas formas de interesse, a fim de personalizar o gráfico de acordo com o que se quer. Isso significa que, posteriormente, novas funções devem ser adicionadas ao *script*, para que as personalizações desejadas em relação à função `plot`, possam ser configuradas e executadas.

Função *title*

A função `title` auxilia na adição e na personalização de títulos, subtítulos, rótulos dos eixos x e y, tipo, cor e tamanho da fonte.

```
> help (title)
title (main = NULL, sub = NULL, xlab = NULL, ylab = NULL, ...)
```

Dentro dela, os argumentos necessários para a personalização do gráfico apresentam os seguintes significados:

`main`: adiciona título do gráfico;

`sub`: adiciona subtítulo do gráfico;

`xlab`: adiciona rótulo do eixo x;

`ylab`: adiciona rótulo do eixo y;

`font` (tipo da fonte): `1` (normal e padrão), `2` (negrito), `3` (itálico) e `4` (negrito e itálico);

`cex` (tamanho da fonte): `cex = 1` (padrão), `0 < cex < 1` (diminui o tamanho em relação ao padrão) e `cex > 1` (aumenta o tamanho em relação ao padrão); e

`col` (cor da fonte): determinada por um valor numérico entre `1` e `8` ou por um nome entre aspas, sendo `col = 1` ou `col = "black"`, o padrão.

O tipo da fonte pode ser dividido em três categorias:

`font.main`: tipo da fonte do título do gráfico;

`font.sub`: tipo da fonte do subtítulo do gráfico; e

`font.lab`: tipo da fonte do rótulo do eixo.

O tamanho da fonte pode ser dividido em três categorias:

`cex.main`: tamanho da fonte do título do gráfico;

`cex.sub`: tamanho da fonte do subtítulo do gráfico; e

`cex.lab`: tamanho da fonte do rótulo do eixo.

E a cor da fonte também pode ser dividida em três categorias:

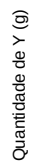
`col.main`: cor da fonte do título do gráfico;

`col.sub`: cor da fonte do subtítulo do gráfico; e

`col.lab`: cor da fonte do rótulo do eixo.

Para o exemplo (Tabela 3.1), foram adicionados os seguintes rótulos aos eixos x e y (Figura 3.3):

```
> title (xlab = "Quantidade de X (g)", ylab = "Quantidade de Y (g)")
```



Quantidade de X (g)

Figura 3.3 - Resultado das adições dos rótulos dos eixos x e y.

Função *axis*

A função `axis` auxilia na adição dos eixos x e y com os seus respectivos valores situados dentro dos limites definidos. Os valores dos eixos podem apresentar-se com diferentes cores (`col`), tamanhos (`cex`) e alinhamentos (`las`), mas da mesma fonte definida pela função `title`. Primeiramente, deve-se verificar a ajuda da função `axis`:

```
> help (axis)
axis (side, at = NULL, labels = TRUE, tick = TRUE, line = NA,
pos = NA, outer = FALSE, font = NA, lty = "solid", lwd = 1,
lwd.ticks = lwd, col = NULL, col.ticks = NULL, ...)
```

Para o exemplo (Tabela 3.1), foram adicionados os seguintes intervalos de valores aos eixos x (Figura 3.4) e y (Figura 3.5):

```
> axis (side = 1, at = seq (0, 5, by = 1), pos = 50, cex.axis =
1, col.axis = 1)
```

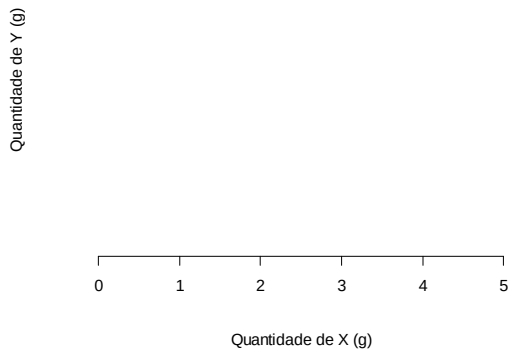


Figura 3.4 - Resultado da adição do intervalo de valores do eixo x.

```
> axis (side = 2, at = seq (50, 100, by = 10), pos = 0, las = 1,
cex.axis = 1, col.axis = 1)
```

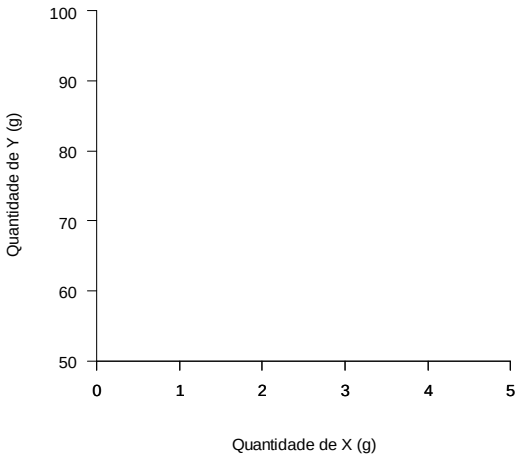


Figura 3.5 - Resultado da adição do intervalo de valores do eixo y.

Os significados dos argumentos contidos na função `axis` são:

`side = 1`: adiciona o eixo x com valores entre 0 e 5 (`seq`), com intervalos de 1 unidade (`by`), que corta o eixo y na posição y = 50 (`pos`) e que possui valores com alinhamento paralelo em relação a ele (`las = 0`) (padrão);

`side = 2`: adiciona o eixo y com valores entre 50 e 100 (`seq`), com intervalos de 10 unidades (`by`), que corta o eixo x na posição x = 0 (`pos`) e que possui valores com alinhamento horizontal em relação a ele (`las = 1`);

`at`: permite determinar os valores que serão exibidos nos eixos x (0, 1, 2, 3, 4 e 5) e y (50, 60, 70, 80, 90 e 100);

`pos = 50` do `axis 1`: indica que o eixo x corta o eixo y em y = 50;

`pos = 0` do `axis 2`: indica que o eixo y corta o eixo x em x = 0;

las: alinhamento dos valores em relação ao respectivo eixo, de forma paralela (`0`) (padrão), horizontal (`1`), perpendicular (`2`) e vertical (`3`);

`cex.axis` (tamanho da fonte dos valores do eixo): `cex.axis = 1` (padrão), `0 < cex.axis < 1` (diminui o tamanho em relação ao padrão) e `cex.axis > 1` (aumenta o tamanho em relação ao padrão); e

`col.axis` (cor da fonte dos valores do eixo): determinada por um valor numérico entre `1` e `8` ou por um nome entre aspas, sendo o padrão, `col.axis = 1` ou `col.axis = "black"`.

Os valores dos eixos x e y também podem ser substituídos por respectivos nomes (*labels*), sendo que a letra `F` representa a anulação do respectivo argumento (Figuras 3.6, 3.7 e 3.8).

```
> plot (x, y, type = "n", xlim = c (0, 5), ylim = c (50, 100),
xlab = "Quantidade de X (g)", ylab = "Quantidade de Y (g)", axes
= F)
```

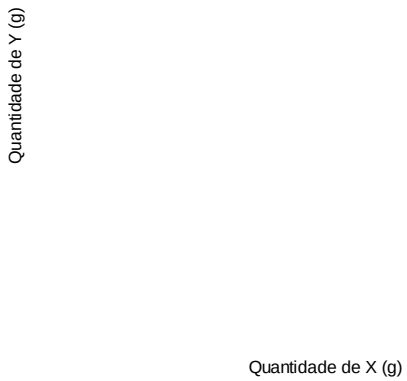


Figura 3.6 - Resultado das exclusões dos eixos x e y.

```
> axis (side = 1, at = seq (0, 5, by = 1), pos = 50, las = 0,
labels = c ("x0", "x1", "x2", "x3", "x4", "x5"), cex.axis = 1,
col.axis = 1)
```

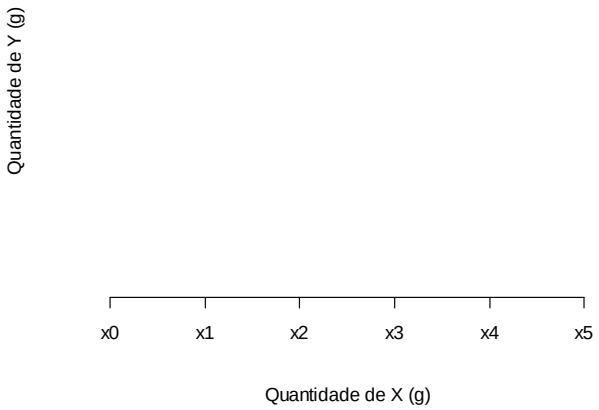


Figura 3.7 - Resultado da adição do intervalo de nomes do eixo x.

```
> axis (side = 2, at = seq (50, 100, by = 10), pos = 0, las = 1,
labels = c ("y50", "y60", "y70", "y80", "y90", "y100"), cex.axis
= 1, col.axis = 1)
```

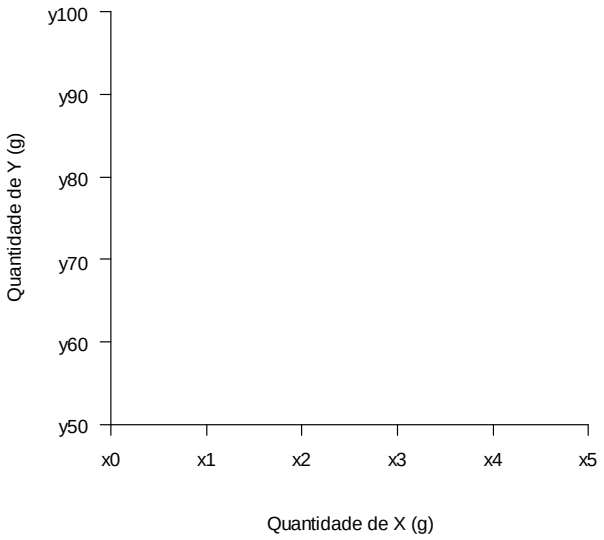


Figura 3.8 - Resultado da adição do intervalo de nomes do eixo y.

Função *points*

A função `points` é a responsável pela adição dos pontos e/ou da linha que liga os pares de valores das variáveis dos eixos x e y. Ao acessar a ajuda da função, tem-se:

```
> help (points)
points (x, y = NULL, type = "p", ...)
```

Dentro da função `points`, foram listados os seguintes argumentos:

`type`: `"p"` para pontos (padrão), `"l"` para linhas, `"b"` para pontos e linhas, `"c"` para pontos vazios unidos por linhas, `"o"` para pontos e linhas sobrepostos, `"s"` ou `"S"` para degraus e `"h"` para linhas verticais;

`pch` (`type = "p"`): tipo do ponto (símbolo) plotado, sendo os 26 tipos definidos entre `0`, `1` (padrão) e `25`;

`cex` (`type = "p"`): tamanho do ponto (símbolo), sendo o padrão `cex = 1`;

`col` (`type = "p"`): cor do ponto (símbolo) determinada por um valor numérico entre `1` e `8` ou por um nome entre aspas, sendo o padrão, `col = 1` ou `col = "black"`;

`lty` (`type = "l"`): tipo da linha plotada, sendo `lty = 1` ou `lty = "solid"` para linha sólida (padrão), `lty = 2` ou `lty = "dashed"` para linha tracejada, `lty = 3` ou `lty = "dotted"` para linha pontilhada, `lty = 4` ou `lty = "dotdash"` para linha tracejada e pontilhada, `lty = 5` ou `lty = "longdash"` para linha tracejada mais espessa e `lty = 6` ou `lty = "twodash"` para linha com dois tracejados paralelos;

`lwd` (`type = "l"`): espessura da linha, sendo o padrão `lwd = 1`.

Os 26 tipos de pontos são definidos, respectivamente, por:

.

Sendo assim, para fazer um gráfico com pontos no modelo padrão, devem ser digitados os seguintes comandos:

```
> plot (x, y, type = "n", xlim = c (0, 5), ylim = c (50, 100),
xlab = "Quantidade de X (g)", ylab = "Quantidade de Y (g)", axes
= F)
> axis (1, at = seq (0, 5, by = 1), pos = 50)
> axis (2, at = seq (50, 100, by = 10), pos = 0, las = 1)
> points (x, y, type = "p", pch = 1, cex = 1, col = 1)
```

E para fazer um gráfico de linha no modelo padrão, têm-se:

```
> plot (x, y, type = "n", xlim = c (0, 5), ylim = c (50, 100),
xlab = "Quantidade de X (g)", ylab = "Quantidade de Y (g)", axes
= F)
> axis (1, at = seq (0, 5, by = 1), pos = 50)
> axis (2, at = seq (50, 100, by = 10), pos = 0, las = 1)
> points (x, y, type = "l", lty = 1, ltw = 1, col = 1)
```

No exemplo de aplicação (Tabela 3.1), para construção do diagrama de dispersão personalizado somente com os pontos, foram utilizados os seguintes comandos no *software R*:

```
# Adicionar os títulos dos eixos x e y (Figura 3.9)
> plot (x, y, type = "n", xlim = c (0, 5), ylim = c (50, 100),
xlab = "Quantidade de X (g)", ylab = "Quantidade de Y (g)", axes
= F)
```

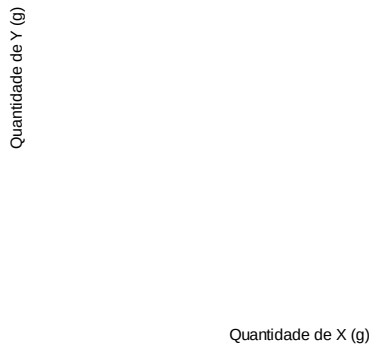


Figura 3.9 - Resultado da adição dos títulos dos eixos x e y.

```
# Adicionar a linha e o intervalo do eixo x (Figura 3.10)
> axis (1, at = seq (0, 5, by = 1), pos = 50)
```

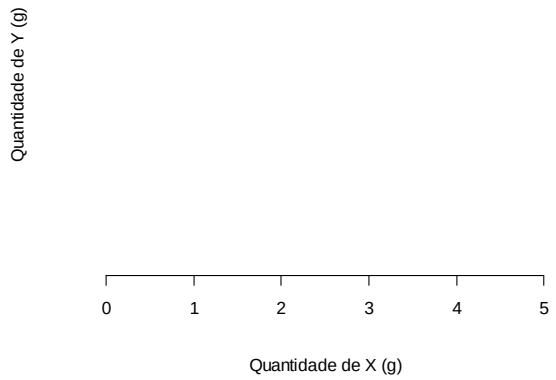


Figura 3.10 - Resultado da adição do intervalo de valores do eixo x.

```
# Adicionar a linha e o intervalo do eixo y (Figura 3.11)
> axis (2, at = seq (50, 100, by = 10), pos = 0, las = 1)
```

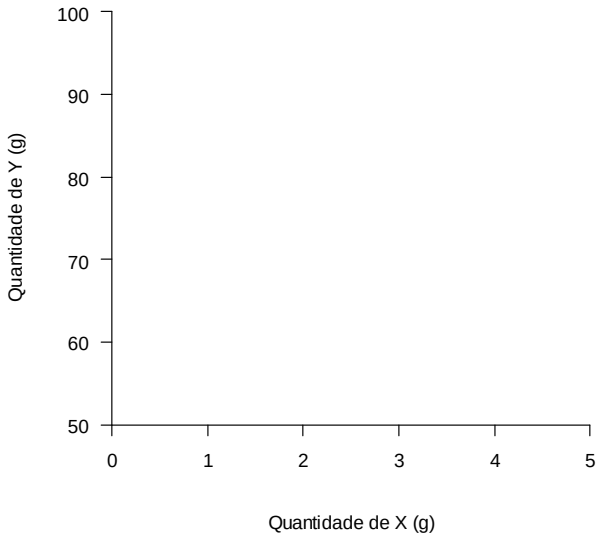


Figura 3.11 - Resultado da adição do *intervalo de valores do* eixo y.

```
# Adicionar os pontos (Figura 3.12)
> points (x, y, pch = 16)
```

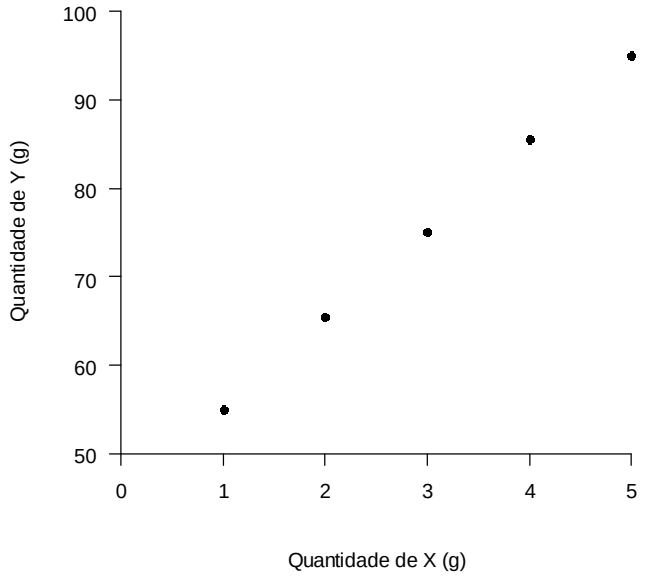


Figura 3.12 - Resultado da adição dos pontos.

Caso seja de interesse, a cor de fundo do gráfico também poderá ser alterada por meio da função `par (bg = "cor especificada")`. Assim, foram utilizados os seguintes comandos no *software R* para construir o diagrama de dispersão com o fundo representado por uma tonalidade clara de cinza (Figura 3.13):

```
> par (bg = "gray90")
> plot (x, y, type = "p", pch = 16, xlim = c (0, 5), ylim = c (50, 100), xlab = "Quantidade de X (g)", ylab = "Quantidade de Y (g)", axes = F)
> axis (1, at = seq (0, 5, by = 1), pos = 50)
> axis (2, at = seq (50, 100, by = 10), pos = 0, las = 1)
```

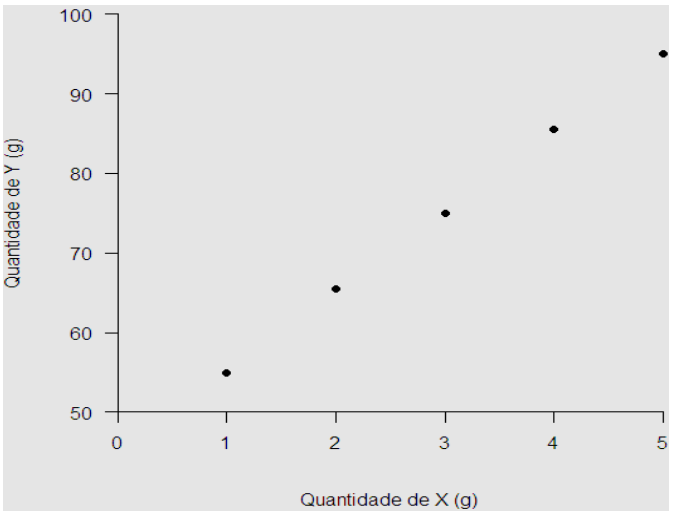


Figura 3.13 - Resultado da alteração da cor de fundo.

Cores

No *software R* existem oito cores básicas e um total de 657 cores. Para ver qual número corresponde a qual cor das oito cores básicas, basta digitar o seguinte comando:

```
> palette ()
[1] "black"  [2] "red"    [3] "green3" [4] "blue"   [5] "cyan"
[6] "magenta" [7] "yellow" [8] "gray"
```

E para ver os nomes das 657 cores, use o seguinte comando:

```
> colours ()
[1] "white"                "aliceblue"            "antiquewhite"
[4] "antiquewhite1"       "antiquewhite2"        "antiquewhite3"
[7] "antiquewhite4"         "aquamarine"           "aquamarine1"
[10] "aquamarine2"          "aquamarine3"          "aquamarine4"
...
[652] "yellow"              "yellow1"              "yellow2"
[655] "yellow3"             "yellow4"              "yellowgreen"
```

Fontes

Nos gráficos do *software R*, as fontes básicas são:

arial: `sans`;

times new roman: `serif`; e

courier: `mono`.

Capítulo 4

Histograma

Introdução

O histograma é uma representação gráfica de uma distribuição de frequências por meio de colunas justapostas. Entre as diversas recomendações para determinar aproximadamente o número de classes (k) de n valores observados de Y, é citada a regra de Sturges:

$k = 1 + \log_2 (n)$.

O segundo passo é determinar aproximadamente o intervalo de classe (h):

$h = LS_i - LI_i = \frac{at_Y}{k}$, em que:

$at_Y = y_{(n)} - y_{(1)}$ = estimativa da amplitude total de Y;

$y_{(n)}$ = maior valor observado de Y;

$y_{(1)}$ = menor valor observado de Y;

LI_i = limite inferior da classe i; e

LS_i = limite superior da classe i (i = 1, 2, ..., k).

Desse modo, o ponto médio da classe i é calculado por:

$PM_i = \frac{LI_i + LS_i}{2}$, para i = 1, 2, ..., k.

A construção de um histograma com os limites inferior (LIE) e superior (LSE) de especificação para uma variável-resposta Y permite verificar se o processo é capaz de atender ou não à especificação, sendo o valor-alvo ou nominal (VN) definido por:

$VN = \frac{LIE + LSE}{2}$.

Como exemplo, considere os valores, também, ordenados, de uma variável-resposta Y (LIE = 68, VN = 80 e LSE = 92) coletados em uma amostra aleatória de 80 itens (RIBEIRO JÚNIOR, 2012) (Tabela 4.1).

Entrada de Dados

Novamente, o primeiro passo é realizar a entrada de dados (Figura 4.1). Após mudar o diretório para a pasta onde se encontra o arquivo de dados do *Excel* (*cap4e1.xlsx*), devem ser escritos os seguintes comandos:

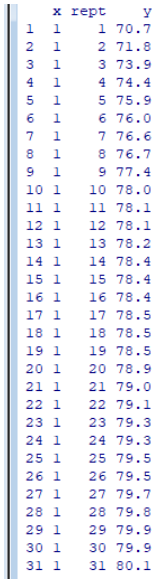
```
> library (openxlsx)
> dcap4e1 = read.xlsx ("cap4e1.xlsx")
> attach (dcap4e1)
> dcap4e1
```

Tabela 4.1 - Arquivo de dados *cap4e1.xlsx*

	A	B	C
1	x	rept	y
2	1	1	70,7
3	1	2	71,8
4	1	3	73,9
5	1	4	74,4
6	1	5	75,9
7	1	6	76
8	1	7	76,6
9	1	8	76,7
10	1	9	77,4
11	1	10	78
12	1	11	78,1
13	1	12	78,1
14	1	13	78,2
15	1	14	78,4
16	1	15	78,4
17	1	16	78,4
18	1	17	78,5
19	1	18	78,5
20	1	19	78,5
21	1	20	78,9
22	1	21	79
23	1	22	79,1
24	1	23	79,3
25	1	24	79,3
26	1	25	79,5
27	1	26	79,5
28	1	27	79,7
29	1	28	79,8
30	1	29	79,9
31	1	30	79,9
32	1	31	80,1
33	1	32	80,2
34	1	33	80,4
35	1	34	80,4
36	1	35	80,5
37	1	36	80,7
38	1	37	80,7
39	1	38	80,7
40	1	39	80,9
41	1	40	81,3

Continua...

42	1	41	81,4
43	1	42	81,6
44	1	43	81,8
45	1	44	81,9
46	1	45	82
47	1	46	82
48	1	47	82,1
49	1	48	82,3
50	1	49	82,5
51	1	50	82,7
52	1	51	82,9
53	1	52	83
54	1	53	83
55	1	54	83,2
56	1	55	83,4
57	1	56	83,5
58	1	57	83,6
59	1	58	83,6
60	1	59	83,7
61	1	60	83,8
62	1	61	84,3
63	1	62	84,5
64	1	63	84,5
65	1	64	84,5
66	1	65	84,6
67	1	66	85,2
68	1	67	85,5
69	1	68	85,5
70	1	69	85,7
71	1	70	86,4
72	1	71	86,5
73	1	72	86,8
74	1	73	86,8
75	1	74	86,8
76	1	75	87,1
77	1	76	87,1
78	1	77	87,1
79	1	78	87,3
80	1	79	88,5
81	1	80	90



```
   x rept    y
1  1    1 70.7
2  1    2 71.8
3  1    3 73.9
4  1    4 74.4
5  1    5 75.9
6  1    6 76.0
7  1    7 76.6
8  1    8 76.7
9  1    9 77.4
10 1   10 78.0
11 1   11 78.1
12 1   12 78.1
13 1   13 78.2
14 1   14 78.4
15 1   15 78.4
16 1   16 78.4
17 1   17 78.5
18 1   18 78.5
19 1   19 78.5
20 1   20 78.9
21 1   21 79.0
22 1   22 79.1
23 1   23 79.3
24 1   24 79.3
25 1   25 79.5
26 1   26 79.5
27 1   27 79.7
28 1   28 79.8
29 1   29 79.9
30 1   30 79.9
31 1   31 80.1
```

Figura 4.1 - Parte da tela da entrada de dados no *R*.

Após realizar a entrada de dados, analisa-se a função responsável por auxiliar na construção do histograma:

```
> help (hist)
hist (x, breaks = "Sturges", freq = NULL, include.lowest = TRUE,
density = NULL, angle = 45, col = NULL, border = NULL, main =
paste ("Histogram of", xname), xlim = range (breaks), ylim =
NULL, xlab = xname, ylab, axes = TRUE, plot = TRUE, ...)
```

Histograma Padrão

Para construir o histograma no formato padrão (Figura 4.2), digita-se o seguinte comando no *R*:

```
> hist (y)
```

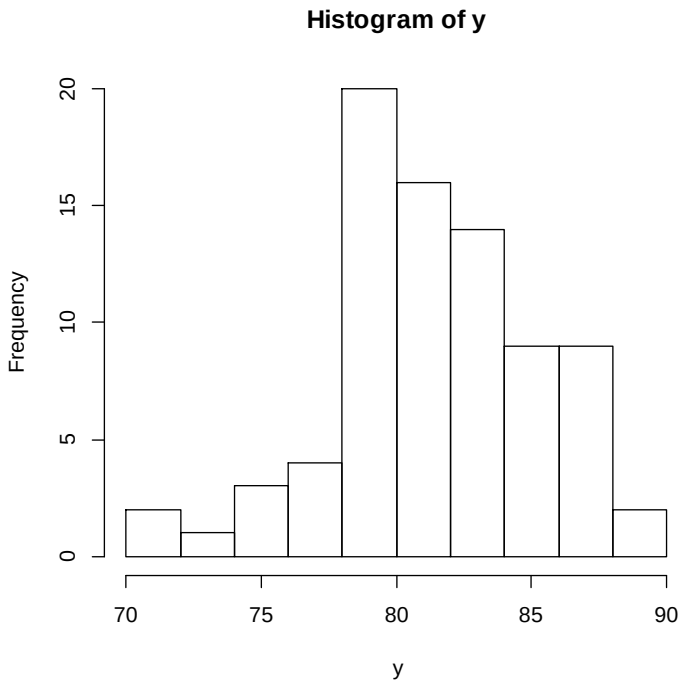


Figura 4.2 - Histograma padrão gerado pelo *R*.

Da função `hist`, foram mencionados e definidos os seguintes argumentos:

`x`: vetor com os valores para a construção do histograma;

`breaks`: conjunto de valores que representa os limites inferior e superior de cada classe estabelecida pela regra de Sturges;

`freq`: quantidade de valores que ocorre dentro de cada classe estabelecida pela regra de Sturges; se `TRUE`, as colunas são representadas por frequências, sendo o padrão apenas quando as classes apresentarem o mesmo intervalo e quando a probabilidade não for especificada; e se `FALSE`, as colunas são representadas por densidades de probabilidades, de modo que o histograma tenha uma área total igual a 1;

`include.lowest`: se `TRUE` (padrão), indica intervalo fechado à esquerda e aberto à direita; e se `FALS`E, indica intervalo aberto à esquerda e fechado à direita; e

`axes`: se `TRUE` (padrão), plota os eixos x e y e mostra os seus valores; e se `FALSE`, não.

Histograma Personalizado

Com o objetivo de personalizar o histograma, deve-se, primeiro, conhecer o menor e o maior valor de Y da amostra, para que todos sejam inseridos, e obter os valores aproximados de k e h, conforme os seguintes comandos:

```
> min (y)
[1] 70.7

> max (y)
[1] 90

> 1 + log2 (length (y)) # número de classes (k)
[1] 7.321928

> (max (y) – min (y)) / 7 # intervalo de classe (h)
[1] 2.757143
```

Assim, para facilitar a contagem das frequências, tomou-se $LI_1 = 70$ e, consequentemente, $LS_1 = 73$ ($h \cong 3$). A continuação desse procedimento gerou as demais classes de intervalo ($k \cong 7$): 70, 73, 76, 79, 82, 85, 88 e 91.

Com isso, dá-se início à personalização (Figura 4.3). Nesse primeiro momento, serão definidas as classes dos intervalos, os limites dos eixos x e y, a densidade e os ângulos das linhas que preenchem as barras e os títulos dos eixos x e y, conforme o seguinte comando:

```
> hist (y, breaks = c (70, 73, 76, 79, 82, 85, 88, 91), density
= 10, xlim = c (66, 93), ylim = c (0, 30), main = "", xlab =
"Y", ylab = "Frequência", axes = F)
```

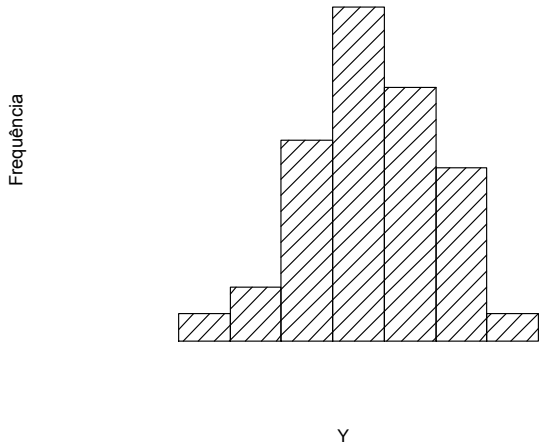


Figura 4.3 - Gráfico inicial da personalização do histograma.

Nesse exemplo e de acordo com o argumento `breaks`, foram criadas sete classes com intervalos fechados à esquerda e abertos à direita (`include.lowest = T`) e com os seguintes pontos médios: 71,5 (70 a 73), 74,5 (73 a 76), 77,5 (76 a 79), 80,5 (79 a 82), 83,5 (82 a 85), 86,5 (85 a 88) e 89,5 (88 a 91).

No *software R*, a personalização de um gráfico se dá, basicamente, evitando-se que os padrões não desejados sejam executados e, posteriormente, configurando-se e executando-se as respectivas personalizações desejadas por meio de novas funções adicionadas ao *script*. Nesse exemplo, dado `axes = F`, considere a função `axis` para as adições dos eixos x e y.

```
> help (axis)
axis (side, at = NULL, labels = TRUE, pos = NA, lty = "solid",
lwd = 1, col = 1, cex = 1, font = 1, las = 0, ...)
```

Da função `axis`, foram definidos os seguintes argumentos:

`side = 1`: adiciona o eixo x com valores (`at`) que serão exibidos entre LI_X e LS_X (`seq`), com intervalos equidistantes (`by`) e que corta o eixo y na posição `pos`;

`side = 2`: adiciona o eixo y com valores (`at`) que serão exibidos entre LI_Y e LS_Y (`seq`), com intervalos equidistantes (`by`) e que corta o eixo x na posição `pos`;

`at`: permite determinar os valores que serão exibidos nos eixos x e y;

`labels`: substitui os valores dos eixos x e y pelos seus respectivos nomes;

`pos`: coordenada de corte dos eixos x e y;

`lty`: tipo das linhas dos eixos x e y determinada por um valor numérico entre `1` e `6` ou por um nome entre aspas, sendo: `1` ou `"solid"` (sólida) (padrão), `2` ou `"dashed"` (tracejada), `3` ou `"dotted"` (pontilhada), `4` ou `"dotdash"` (tracejada e pontilhada), `5` ou `"longdash"` (sólida e tracejada) e `6` ou `"twodash"` (tracejada dupla); e

`lwd`: espessura das linhas dos eixos x e y, sendo: `1` (padrão), `0 < lwd < 1` (diminui a espessura em relação ao padrão) e `lwd > 1` (aumenta a espessura em relação ao padrão).

Continuando a construção do histograma personalizado, têm-se:

```
# Adicionar os pontos médios do eixo x (Figura 4.4)
> axis (1, at = seq (71.5, 89.5, by = 3), pos = 0, labels = c
("71,5", "74,5", "77,5", "80,5", "83,5", "86,5", "89,5"))
```

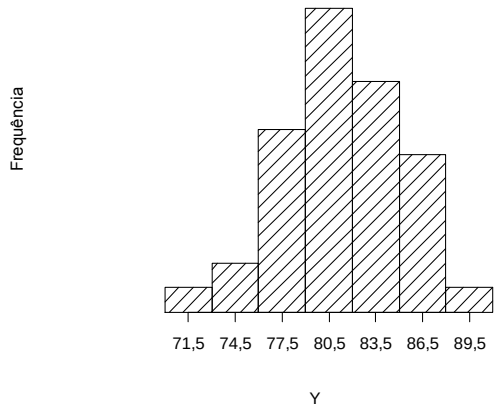


Figura 4.4 - Histograma resultante da adição do eixo x.

```
# Adicionar os valores do eixo y (Figura 4.5)
> axis (2, at = seq (0, 30, by = 5), pos = 66, las = 1)
```

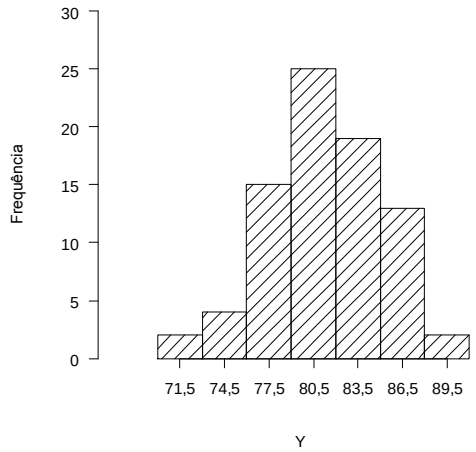


Figura 4.5 - Histograma resultante da adição do eixo y.

```
# Adicionar a linha do eixo x (Figura 4.6)
> abline (h = 0) # adiciona linha horizontal em y = 0
```

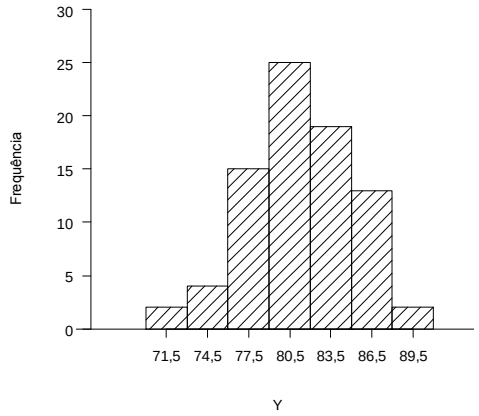


Figura 4.6 - Histograma resultante da adição da linha horizontal do eixo x.

```
# Adicionar a linha tracejada do LIE (Figura 4.7)
> segments (68, 0, 68, 28, lty = 2) # insere uma reta da coordenada (x = 68, y = 0) para a coordenada (x = 68, y = 27) com estilo de linha tracejada
```

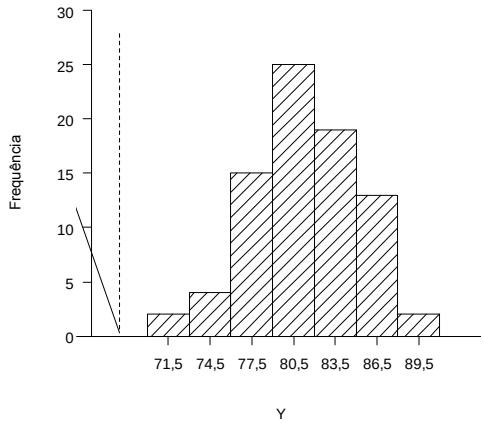


Figura 4.7 - Histograma resultante da adição da linha tracejada do LIE.

```
# Adicionar a linha tracejada do VN (Figura 4.8)
> segments (80, 0, 80, 28, lty = 3) # insere uma reta da coordenada (x = 80, y = 0) para a coordenada (x = 80, y = 27) com estilo de linha pontilhada
```

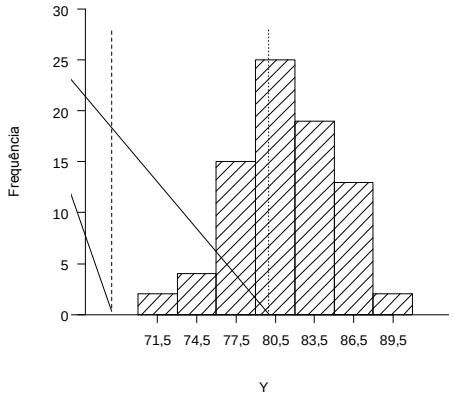


Figura 4.8 - Histograma resultante da adição da linha tracejada do VN.

```
# Adicionar a linha tracejada do LSE (Figura 4.9)
> segments (92, 0, 92, 28, lty = 2) # insere uma reta da coordenada (x = 92, y = 0) para a coordenada (x = 92, y = 27) com estilo de linha tracejada
```

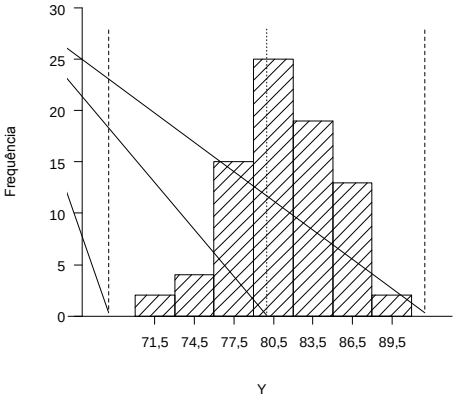


Figura 4.9 - Histograma resultante da adição da linha tracejada do LSE.

```
# Inserir a palavra LIE (Figura 4.10)
> text (68, 29, "LIE") # adiciona LIE ao gráfico centrado na coordenada (x = 68, y = 29)
```

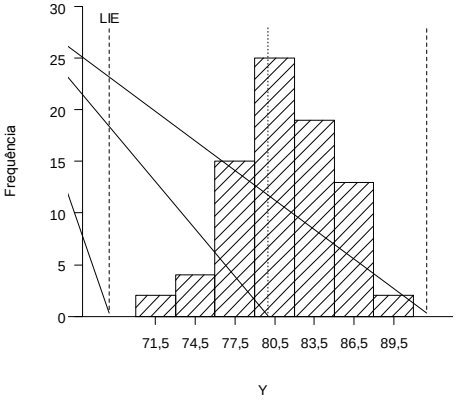


Figura 4.10 - Histograma resultante da adição da palavra LIE.

```
# Inserir a palavra VN (Figura 4.11)
> text (80, 29, "VN") # adiciona VN ao gráfico centrado na coordenada (x = 80, y = 29)
```

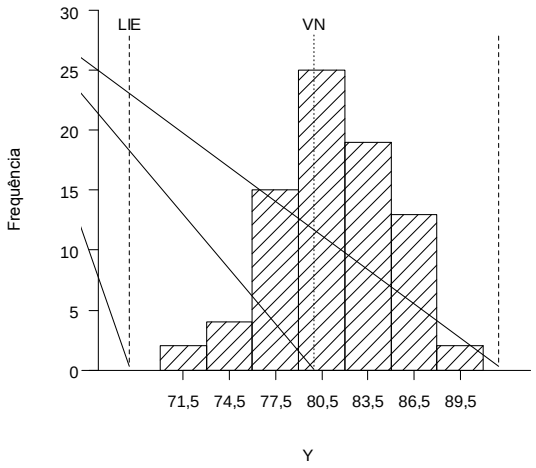


Figura 4.11 - Histograma resultante da adição da palavra VN.

```
# Inserir a palavra LSE (Figura 4.12)
> text (92, 29, "LSE") # adiciona LSE ao gráfico centrado na coordenada (x = 92, y = 29)
```

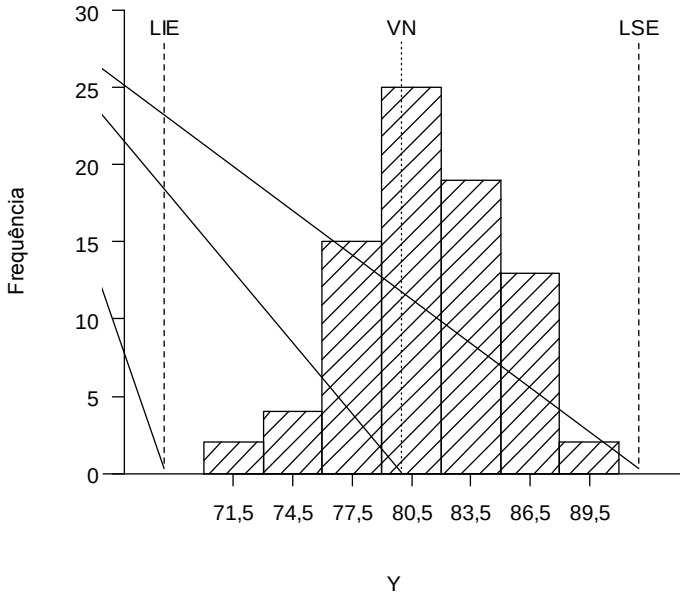


Figura 4.12 - Histograma resultante da adição da palavra LSE.

Por fim, tem-se o histograma personalizado e com a adição do limite de especificação e, quando comparado ao histograma padrão, de melhor qualidade gráfica. Assim, é evidente que o histograma personalizado torna a análise muito mais fácil e assertiva do que o histograma padrão gerado anteriormente.

Além dessa personalização, podem-se personalizar outros detalhes, como por exemplo: definir se a barra será sólida ou hachurada, as cores, alterar ou tirar os limites de especificação, entre outros.

Capítulo 5

Box-plot

Introdução

O *box-plot* é um gráfico que apresenta simultaneamente várias características de um conjunto de dados: locação, dispersão, simetria ou assimetria e presença de observações discrepantes (*outliers*).

Para construção do *box-plot*, considere o mesmo exemplo de aplicação com os 80 valores observados de uma variável-resposta Y (RIBEIRO JÚNIOR, 2012) (Tabela 4.1).

Entrada de Dados

Para construção do *box-plot*, deve-se, primeiro realizar a mudança do diretório para a pasta onde se encontra o arquivo de dados (*.xlsx*) e, posteriormente, fazer a entrada de dados, como segue:

```
> library (openxlsx)
> dcap4e1 = read.xlsx ("cap4e1.xlsx")
> attach (dcap4e1)
> dcap4e1
```

Após realizar a entrada de dados, analisa-se a função responsável por auxiliar na construção do *box-plot*:

```
> help (boxplot)
boxplot (x, ..., range = 1.5, width = NULL, varwidth = FALSE, notch = FALSE, outline = TRUE, names, plot = TRUE, border = par ("fg"), col = NULL, log = "", pars = list (boxwex = 0.8, staplewex = 0.5, outwex = 0.5), horizontal = FALSE, add = FALSE, at = NULL)
```

Da função `boxplot`, foram definidos os seguintes argumentos:

`x`: vetor com os valores para a construção do *box-plot*;

`horizontal`: se `FALSE` (padrão), indica a construção do *box-plot* na vertical; e se `TRUE`, indica a construção do *box-plot* na horizontal;

`axes`: se `TRUE` (padrão), plota o eixo y e mostra os seus valores; e se `FALSE`, não;

`lty`: tipo da linha plotada determinada por um valor numérico entre `1` e `6` ou por um nome entre aspas, sendo `2` ou `"dashed"` o padrão;

`col`: cor da caixa determinada por um valor numérico ou por um nome entre aspas, sendo a ausência de cor o padrão; e

`pch`: tipo do(s) ponto(s) plotado(s) para representar(em) o(s) *outlier(s)*, sendo os 26 tipos definidos entre `0`, `1` (padrão) e `25`.

Box-plot Padrão

Para gerar o *box-plot* padrão basta digitar o seguinte comando no *R* (Figura 5.1):

```
> boxplot (y)
```

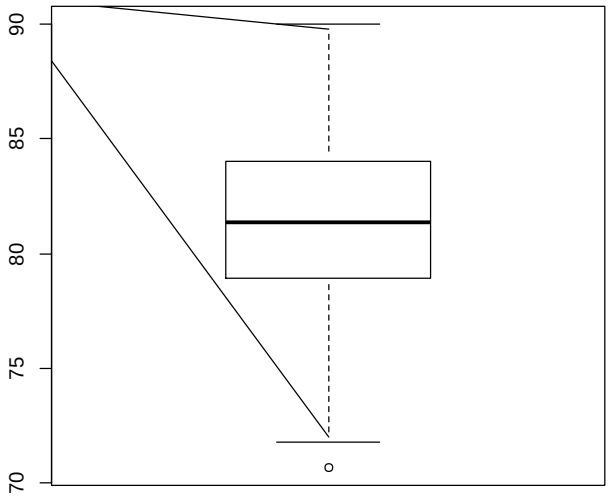


Figura 5.1 - *Box-plot* padrão gerado pelo *R*.

Box-plot Personalizado

Para personalização do *box-plot*, segue-se a mesma metodologia utilizada para a do histograma.

```
# Não adicionar os textos, definir os limites do eixo vertical (y), definir a cor da caixa e o tipo de ponto do outlier (Figura 5.2)
> boxplot (y, ylab = "Y", ylim = c (70, 90), axes = F, lty = 1, col = 8, pch = 8)
```

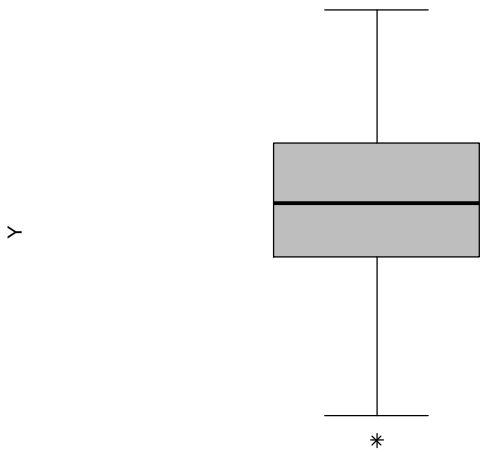


Figura 5.2 - *Box-plot* resultante do primeiro comando.

```
# Adicionar a linha e o intervalo de valores do eixo y (Figura 5.3)
> axis (2, at = seq (70, 90, by = 5), las = 1)
```

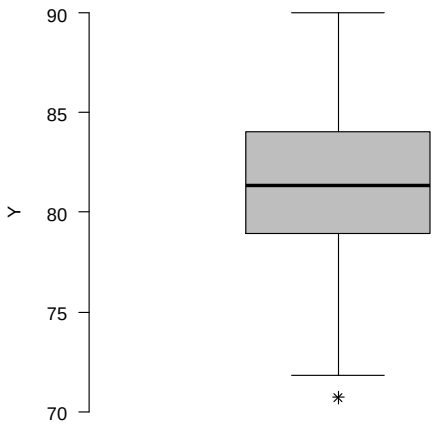


Figura 5.3 - *Box-plot* resultante da adição do eixo y.

```
# Adicionar a linha do eixo x (Figura 5.4)
> abline (h = 70) # adiciona linha horizontal em y = 70
```

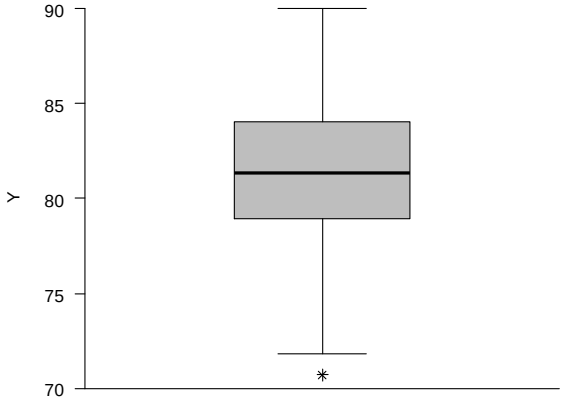


Figura 5.4 - *Box-plot* resultante da adição do eixo x.

```
# Adicionar a palavra outlier (Figura 5.5)
> text (0.7, 71, "outlier") # adiciona outlier na coordenada (0,7; 71)
```

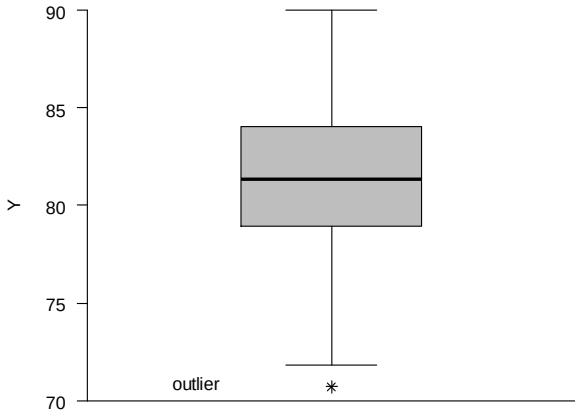


Figura 5.5 - *Box-plot* resultante da adição da palavra *outlier*.

```
# Adicionar a palavra quartil 1 (Figura 5.6)
> text (0.7, 78.975, "quartil 1") # adiciona quartil 1 na coordenada (0,7; 78,975)
```

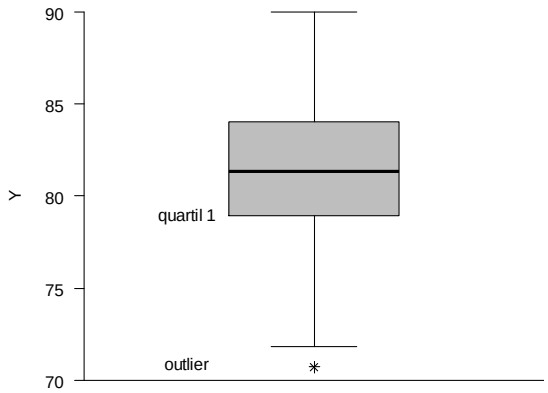


Figura 5.6 - *Box-plot* resultante da adição da palavra *quartil 1*.

```
# Adicionar a palavra mediana (Figura 5.7)
> text (0.7, 81.35, "mediana") # adiciona mediana na coordenada (0,7; 81,35)
```

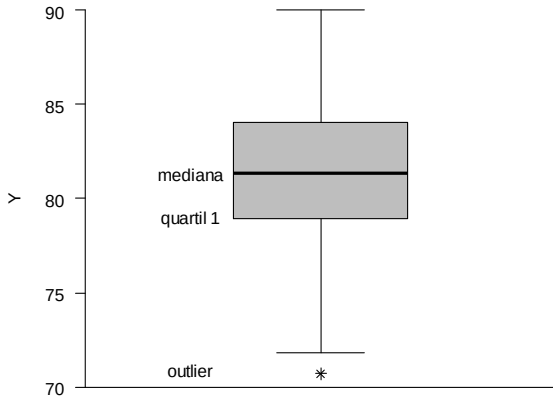


Figura 5.7 - *Box-plot* resultante da adição da palavra *mediana*.

```
# Adicionar a palavra quartil 3 (Figura 5.8)
> text (0.7, 83.925, "quartil 3") # adiciona quartil 3 na coordenada (0,7; 83,925)
```

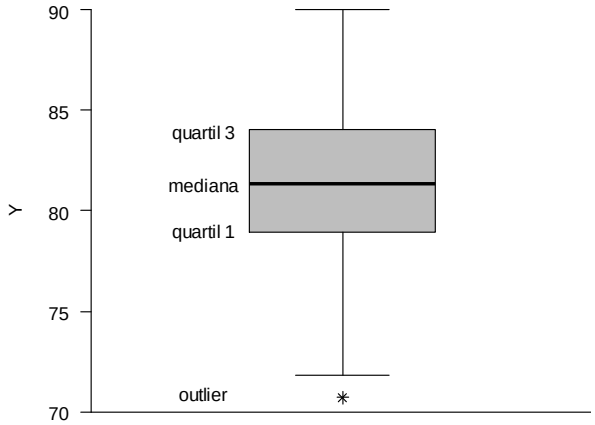


Figura 5.8 - *Box-plot* resultante da adição da palavra *quartil 3*.

Por fim, tem-se a construção de um *box-plot* de melhor qualidade, com o intervalo de valores, o tipo de ponto e a cor, desejados.

Cor da Caixa

Para alterar a cor da caixa, é preciso substituir no primeiro comando que foi mostrado no passo a passo da personalização, o número que representa a cor atual para o número que representa a nova cor. Nesse exemplo, a cor cinza foi substituída pela cor verde. Sendo assim, o novo comando foi digitado como segue:

```
> boxplot (y, ylab = "Y", ylim = c (70, 90), axes = F, lty = 1,
col = 3, pch = 8)
```

Ao comparar o novo comando com o inicial, pode-se perceber que o número da cor foi trocado de `8` para `3`, o que resultou em um *box-plot* com caixa verde. Contudo, é importante saber que o novo *box-plot* contendo todas as personalizações, foi obtido através dos comandos adicionais descritos anteriormente e, não somente, através do último comando (Figura 5.9).

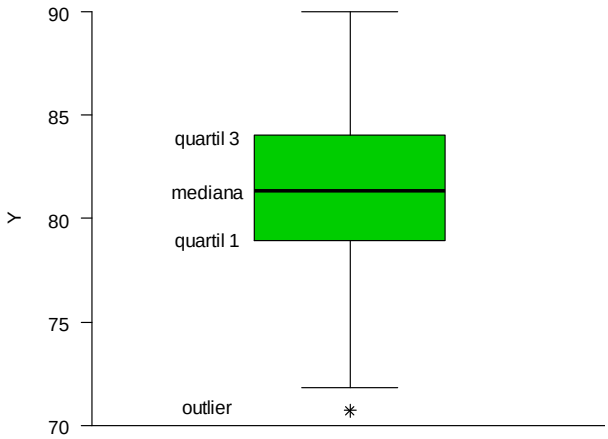


Figura 5.9 - *Box-plot* resultante da alteração da cor da caixa.

Tipo de Ponto do *Outlier*

Para alterar o tipo de ponto do *outlier*, é necessário mudar o número do argumento *pch* para o número correspondente ao ponto que se deseja ter no novo *box-plot*. Assim, será necessário alterar, novamente, o primeiro comando descrito no passo a passo da personalização. No exemplo, foi trocado o ponto atual (*) por um x, símbolo equivalente a `pch = 4`. Desse modo, o comando foi definido da seguinte forma:

```
> boxplot (y, ylab = "Y", ylim = c (70, 90), axes = F, lty = 1,
col = 3, pch = 4)
```

O *box-plot* resultante encontra-se na Figura 5.10.

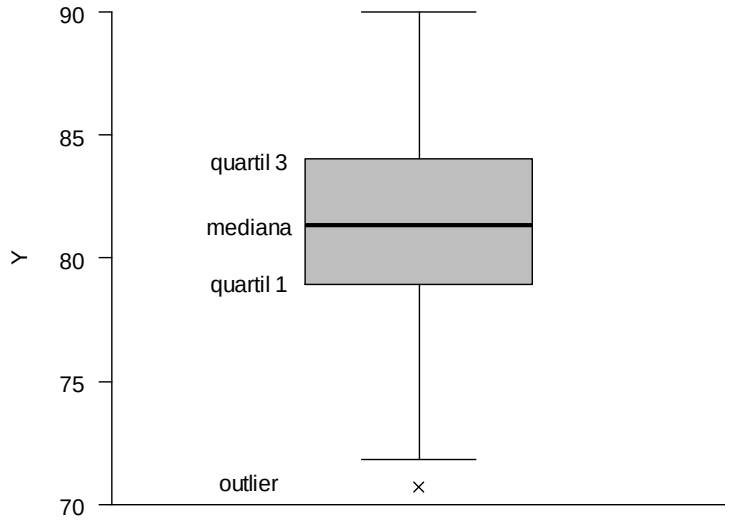


Figura 5.10 - *Box-plot* resultante da alteração do tipo de ponto do *outlier*.

Intervalo do Eixo y

Como exemplo, o intervalo de valores do eixo y será alterado para o intervalo de 70 a 100 com variação de 10 em 10 (Figura 5.11):

```
> boxplot (y, ylab = "Y", ylim = c (70, 100), axes = F, lty = 1,
col = 8, pch = 8)
> axis (2, at = seq (70, 100, by = 10), las = 1)
```

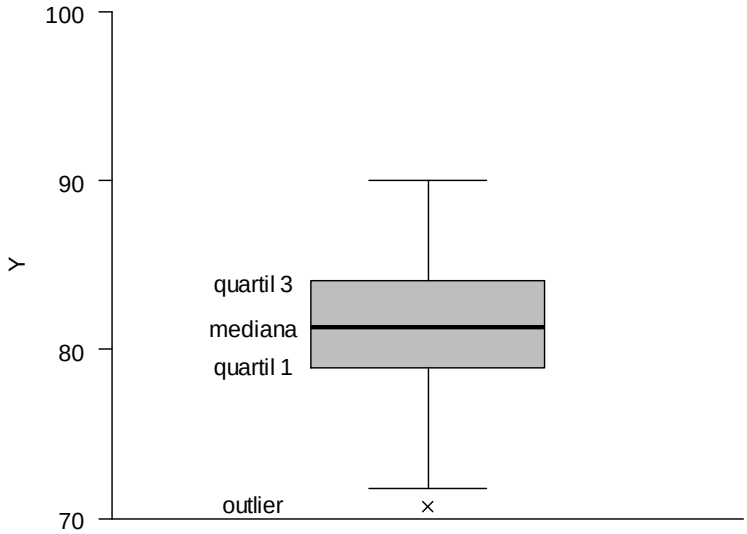


Figura 5.11 - *Box-plot* resultante da alteração do intervalo de valores do eixo y.

Capítulo 6

Diagrama de Dispersão

Introdução

O diagrama de dispersão é um gráfico utilizado para a visualização do tipo de relacionamento existente entre duas variáveis. Para construção do diagrama de dispersão, considere como exemplo uma amostra aleatória de 35 pares de valores observados das variáveis-resposta Y_1 e Y_2 (RIBEIRO JÚNIOR, 2012) (Tabela 6.1).

Tabela 6.1 - Arquivo de dados *cap6e1.xlsx*

	A	B	C
1	item	y1	y2
2	1	213	19,5
3	2	213,6	19,3
4	3	213,9	19,1
5	4	214,2	18,7
6	5	214,4	18,6
7	6	214,9	18,6
8	7	215,3	18,5
9	8	215,5	18,3
10	9	215,8	18,2
11	10	216	18
12	11	216,5	17,8
13	12	216,7	17,6
14	13	217	17,4
15	14	217,3	17,3
16	15	217,7	17
17	16	218,1	17
18	17	218,4	16,8
19	18	218,6	16,7
20	19	219,2	16,5
21	20	219,4	16,3
22	21	219,9	16,2
23	22	220	16,3
24	23	220,6	16,1
25	24	220,9	16,1
26	25	221,2	16,2
27	26	221,5	16,1
28	27	221,7	16
29	28	222	16
30	29	222,2	15,9
31	30	222,7	15,7
32	31	223,3	15,6
33	32	223,5	15,7
34	33	223,8	15,5
35	34	224,1	15,4
36	35	224,7	15,3

Entrada de Dados

Como usual, o primeiro passo será realizar a entrada de dados, após mudar o diretório para a pasta onde se encontra o arquivo de dados do *Excel* (Figura 6.1):

```
> library (openxlsx)
> dcap6e1 = read.xlsx ("cap6e1.xlsx")
> attach (dcap6e1)
> dcap6e1
```

	item	y1	y2
1	1	213.0	19.5
2	2	213.6	19.3
3	3	213.9	19.1
4	4	214.2	18.7
5	5	214.4	18.6
6	6	214.9	18.6
7	7	215.3	18.5
8	8	215.5	18.3
9	9	215.8	18.2
10	10	216.0	18.0
11	11	216.5	17.8
12	12	216.7	17.6
13	13	217.0	17.4
14	14	217.3	17.3
15	15	217.7	17.0
16	16	218.1	17.0
17	17	218.4	16.8
18	18	218.6	16.7
19	19	219.2	16.5
20	20	219.4	16.3
21	21	219.9	16.2
22	22	220.0	16.3
23	23	220.6	16.1
24	24	220.9	16.1
25	25	221.2	16.2
26	26	221.5	16.1
27	27	221.7	16.0
28	28	222.0	16.0
29	29	222.2	15.9
30	30	222.7	15.7

Figura 6.1 - Parte da tela da entrada de dados no *R*.

Após realizar a entrada de dados, analisa-se a função (cartesiana ou fórmula) responsável por auxiliar na construção do diagrama de dispersão:

```
> help (plot)
plot (x, y, type = "p", pch = 1, cex = 1, col = 1, ...)
plot (y ~ x, type = "p", pch = 1, cex = 1, col = 1, ...)
```

Da função `plot`, foram definidos os seguintes argumentos:

`x`: vetor com os valores do eixo x;

`y`: vetor com os valores do eixo y;

`type`: `"p"` para plotar pontos associados aos pares de valores das variáveis dos eixos x e y (padrão), `"l"` para linhas, `"b"` para pontos e linhas, `"c"` para pontos vazios unidos por linhas, `"o"` para pontos e linhas sobrepostos, `"s"` ou `"S"` para degraus, `"h"` para linhas verticais e `"n"` para não plotar; e

`pch`, `cex` e `col` (`type = "p"`): tipo, tamanho e cor do ponto (símbolo) plotado respectivamente, sendo o padrão igual a `1`.

Diagrama de Dispersão Padrão

Para construção do diagrama de dispersão padrão (Figura 6.2), podem ser utilizados os seguintes comandos:

```
> plot (y1, y2)
```

```
> plot (y2 ~ y1)
```

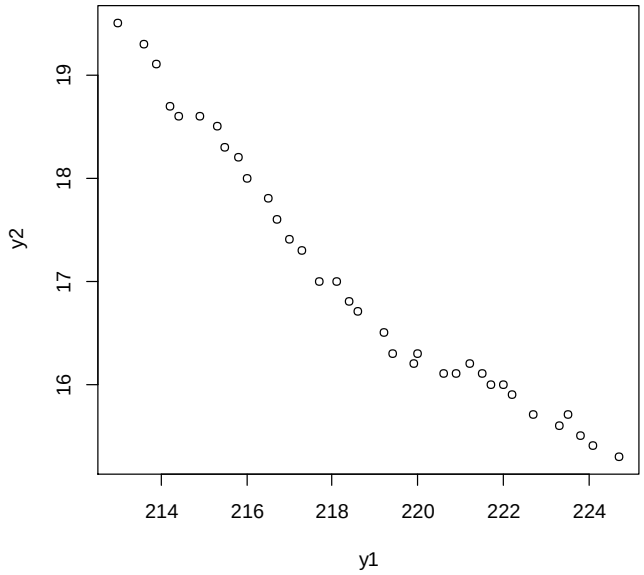


Figura 6.2 - Diagrama de dispersão padrão gerado pelo *R*.

Diagrama de Dispersão Personalizado

Para personalização do diagrama de dispersão, seguem-se as mesmas metodologias utilizadas para as do histograma e do *box-plot*.

```
# Definir tipo de ponto e nomes dos eixos x e y e excluir as linhas dos eixos x e y (Figura 6.3)
> plot (y1, y2, pch = 16, xlab = "Y1", ylab = "Y2", xlim = c (212, 226), ylim = c (15, 20), axes = F)
```

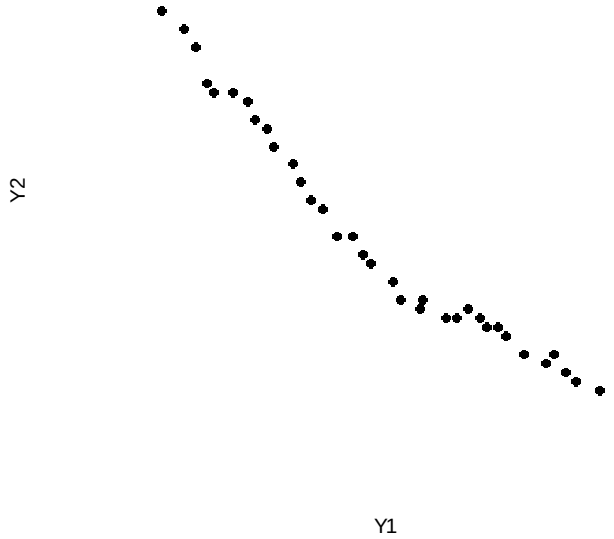


Figura 6.3 - Diagrama de dispersão resultante da alteração do tipo de ponto e dos nomes dos eixos x e y.

```
# Adicionar o intervalo de valores do eixo x (Figura 6.4)
> axis (1, at = seq (212, 226, by = 2), pos = 15)
```

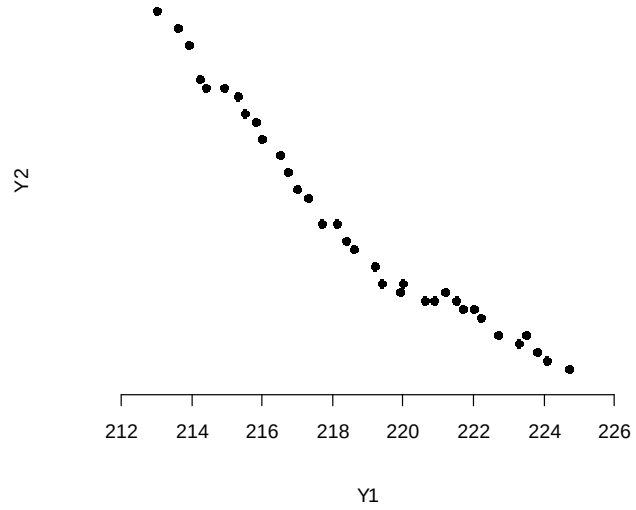


Figura 6.4 - Diagrama de dispersão resultante da adição dos valores do eixo x.

```
# Adicionar o intervalo de valores do eixo y (Figura 6.5)
> axis (2, at = seq (15, 20, by = 1), pos = 212, las = 1)
```

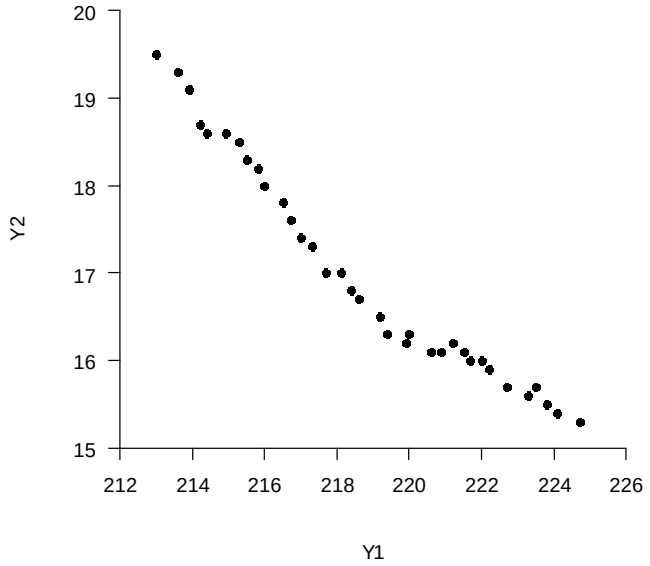


Figura 6.5 - Diagrama de dispersão resultante da adição dos valores do eixo y.

Tipo de Ponto

Para alterar o tipo de ponto, é preciso mudar no primeiro comando apresentado durante a personalização do diagrama de dispersão, o tipo de ponto definido (`pch = 16`). Nesse exemplo, o tipo do ponto foi alterado para o símbolo de triângulo (`pch = 2`) (Figura 6.6). Assim, os novos comandos foram digitados por:

```
> plot (y1, y2, pch = 3, xlab = "Y1", ylab = "Y2", xlim = c
(212, 226), ylim = c (15, 20), axes = F)
> axis (1, at = seq (212, 226, by = 2), pos = 15)
> axis (2, at = seq (15, 20, by = 1), pos = 212, las = 1)
```

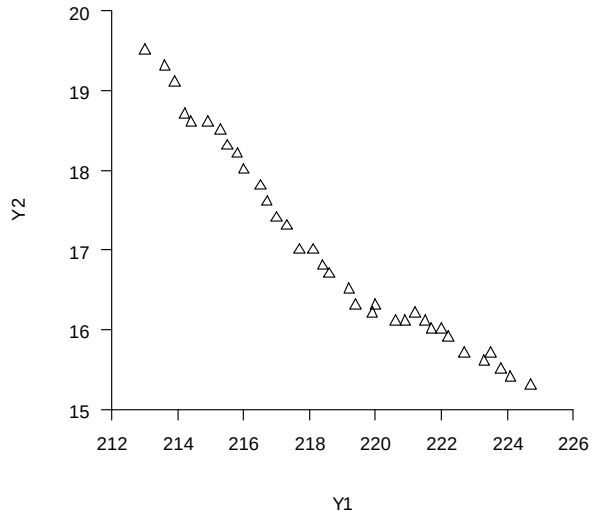


Figura 6.6 - Diagrama de dispersão resultante da alteração do tipo de ponto.

Nomes dos Eixos x e y

Agora, os nomes dos eixos x (Y1) e y (Y2) serão alterados para Altura e Peso, respectivamente. Desse modo, será necessário substituir os argumentos `xlab = Y1` e `ylab = Y2` para `xlab = Altura` e `ylab = Peso` (Figura 6.7), como segue:

```
> plot (y1, y2, pch = 16, xlab = "Altura", ylab = "Peso", xlim =
c (212, 226), ylim = c (15, 20), axes = F)
> axis (1, at = seq (212, 226, by = 2), pos = 15)
> axis (2, at = seq (15, 20, by = 1), pos = 212, las = 1)
```

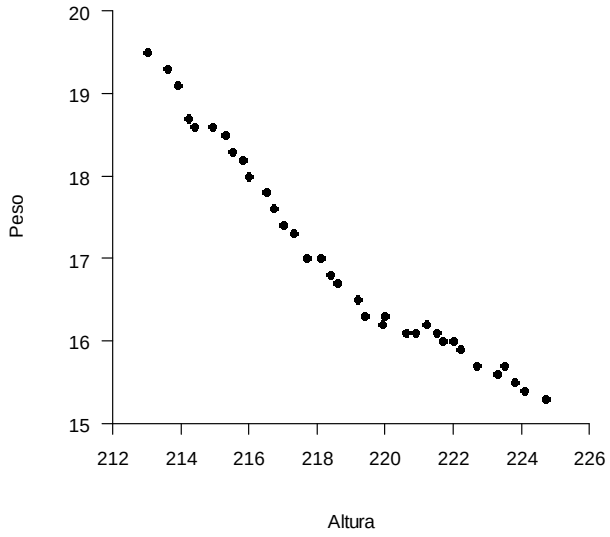


Figura 6.7 - Diagrama de dispersão após as alterações dos nomes dos eixos x e y.

Capítulo 7

Gráfico de Pareto

Introdução

O gráfico de Pareto é um gráfico de colunas verticais ordenadas de forma decrescente, que divide os efeitos (variáveis-resposta Ys) em classes vitais e triviais. Por fim e pertencente às triviais, a classe “Outros” deverá conter, no máximo, 10% dos defeitos e ser colocada por último.

Como exemplo, considere que em uma empresa há 130 itens classificados de acordo com o tipo de defeito: Y_1 (12), Y_2 (41), Y_3 (55), Y_4 (11), Y_5 (3), Y_6 (2), Y_7 (1), Y_8 (2), Y_9 (2) e Y_{10} (1). Cada Y representa um tipo de defeito e os números nos parênteses representam a quantidade de itens que possuem tal defeito. Nesse exemplo, a classe “Outros” (O), com 6,15% das frequências de ocorrências, foi composta pelos últimos cinco tipos de defeitos. Para que o gráfico seja feito corretamente, os tipos de defeitos devem ser digitados na planilha em ordem decrescente de frequências e, além disso, a classe ‘’Outros’’ (O) deve vir por último (RIBEIRO JÚNIOR, 2012) (Tabela 7.1).

Tabela 7.1 - Arquivo de dados *cap7e1.xlsx*

	A	B
1	defeito	freq
2	Y3	55
3	Y2	41
4	Y1	12
5	Y4	11
6	Y5	3
7	O	8

Entrada de Dados

O primeiro passo é fazer a entrada de dados. Primeiro, muda-se o diretório para a pasta onde se encontra o arquivo de dados na extensão *xlsx* (Figura 7.1).

```
> library (openxlsx)
> dcap7e1 = read.xlsx ("cap7e1.xlsx")
> attach (dcap7e1)
> dcap7e1
```

```
  defeito freq
1      Y3   55
2      Y2   41
3      Y1   12
4      Y4   11
5      Y5    3
6       O    8
>
```

Figura 7.1 - Tela da entrada de dados no *R*.

Para construção do gráfico de Pareto, foi utilizada a função `barplot`.

```
> help (barplot)
barplot (height, space = NULL, names.arg = NULL, beside = FALSE,
horiz = FALSE, density = NULL, angle = 45, col = NULL, border =
1, main = NULL, sub = NULL, xlab = NULL, ylab = NULL, xlim =
NULL, ylim = NULL, plot = TRUE, cex = 1, font = 1, las = 0, ...)
```

Os argumentos mencionados da função `barplot` são definidos por:

`height`: conjunto de valores que representam as alturas (frequências) das barras ou das colunas separadas;

`space`: espaço entre as barras, sendo o padrão igual a 0,2;

`names.arg`: vetor com os nomes a serem plotados como rótulos das barras ou das colunas, respectivamente;

`horiz`: se `FALSE` (padrão), insere colunas verticais com a primeira à esquerda (gráfico de colunas); e se `TRUE`, insere barras horizontais com a primeira na parte inferior (gráfico de barras);

`density`: densidade de linhas de sombreamento de cada barra ou coluna; se omitida (padrão), significa que nenhuma linha de sombreamento é desenhada;

`angle`: ângulo no sentido anti-horário, em graus, de cada linha de sombreamento disposta dentro de cada barra, sendo o padrão igual a 45°, se `density > 0`;

`col`: cores das barras ou das colunas determinadas por um ou mais valores numéricos ou por um ou mais nomes entre aspas

`border`: cor da borda das barras ou das colunas determinada por um valor numérico ou por um nome entre aspas, sendo `1` ou `"black"` o padrão e `FALSE` para omitir a borda;

`main` e `sub`: se igual a `NULL` ou `""` (padrão), não são mostrados o título e o subtítulo do gráfico; e se especificados entre aspas, são mostrados o título e o subtítulo do gráfico de acordo com o que foi especificado;

`xlab` e `ylab`: se igual a `NULL` ou `""` (padrão), não são mostrados os rótulos dos eixos x e y; e se especificados entre aspas, são mostrados os rótulos dos eixos x e y de acordo com o que foi especificado;

`xlim` e `ylim`: se igual a `NULL` (padrão), são adicionados os limites inferior (LI) e superior (LS) não especificados aos valores dos eixos x (LI_X e LS_X) e y (LI_Y e LS_Y); e se igual a `c (LI, LS)`, são adicionados os limites (LI e LS) especificados;

`plot`: se `TRUE` (padrão), o gráfico é plotado; e se `FALSE`, o gráfico não é plotado;

`cex`: tamanho da fonte, sendo `1` (padrão), `0 < cex < 1` (diminui o tamanho em relação ao padrão) e `cex > 1` (aumenta o tamanho em relação ao padrão);

`font`: tipo da fonte, sendo `1` (normal e padrão), `2` (negrito), `3` (itálico) e `4` (negrito e itálico); e

`las`: alinhamento dos valores em relação ao respectivo eixo, sendo `0` (alinha os valores de forma paralela ao eixo e padrão), `1` (alinha os valores na horizontal do eixo), `2` (alinha os valores na perpendicular do eixo) e `3` (alinha os valores na vertical do eixo).

Gráfico de Pareto Padrão

Para construção do gráfico de Pareto na forma padrão da função `barplot` e, dado que não há necessidade de inserir os argumentos nas definições padrões dentro de cada função do *software R*, utiliza-se o seguinte comando (Figura 7.2):

```
> barplot (freq)
```

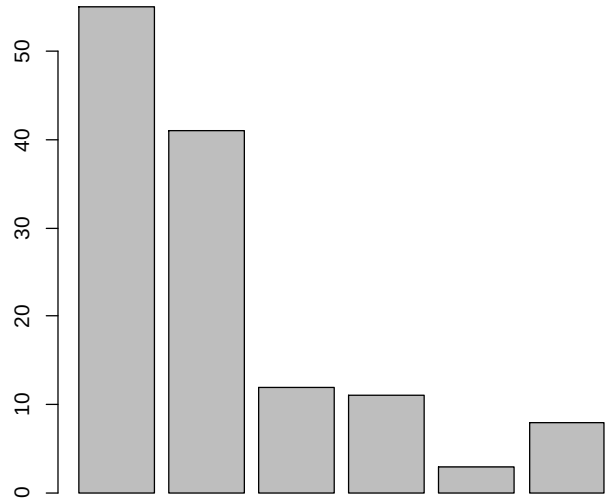


Figura 7.2 - Gráfico de Pareto padrão gerado pelo *R*.

Como se pode perceber, o gráfico de Pareto padrão que é gerado não possui uma qualidade muito boa. Afinal, não possui o eixo x e nem o seu intervalo. Além disso, as classes não são separadas entre vitais e triviais.

Gráfico de Pareto Personalizado

A personalização visa mudar a cor de alguns elementos gráficos, incluir títulos e rótulos, ajustar a cor, o tamanho e o tipo da fonte dos textos, alterar a cor e o tipo das linhas e dos símbolos e adicionar outros itens. Nesse caso, a maioria das personalizações é feita com argumentos não padrões que devem ser adicionados, para que elas possam ser configuradas e executadas.

```
# Definir os intervalos e os nomes dos eixos x e y, alterar a cor para branca e hachurar as barras e alinhar os valores na horizontal do eixo y (Figura 7.3)
> barplot (freq, names.arg = defeito, density = 10, col = 1, xlab = "Tipo de defeito", ylab = "Frequência", ylim = c (0, 60), las = 1)
```

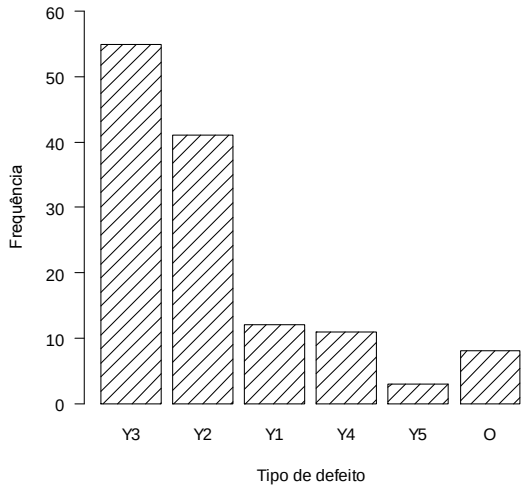


Figura 7.3 - Gráfico de Pareto resultante após a primeira personalização.

```
# Adicionar a linha do eixo x (Figura 7.4)
> abline (h = 0) # adiciona linha horizontal em y = 0
```

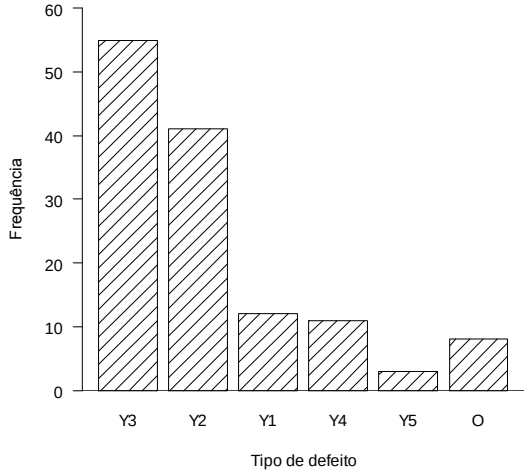


Figura 7.4 - Gráfico de Pareto resultante após a segunda personalização.

```
# Adicionar Classes vitais (Figura 7.5)
> text (1.3, 58, "Classes vitais") # adiciona o texto, entre
aspas, ao gráfico centrado na coordenada (1,3; 58)
```

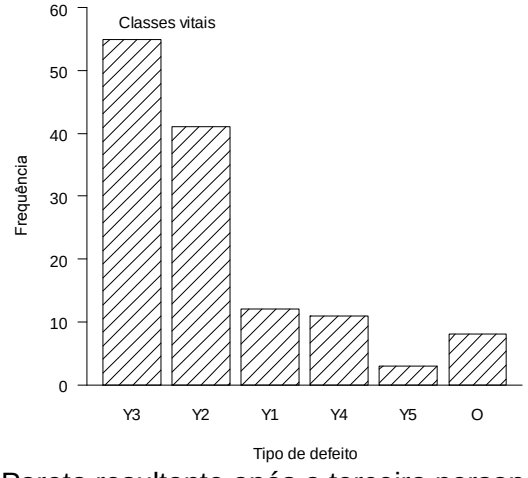


Figura 7.5 - Gráfico de Pareto resultante após a terceira personalização.

```
# Adicionar Classes triviais (Figura 7.6)
> text (5, 15, "Classes triviais") # adiciona o texto, entre aspas, ao gráfico centrado na coordenada (5, 15)
```

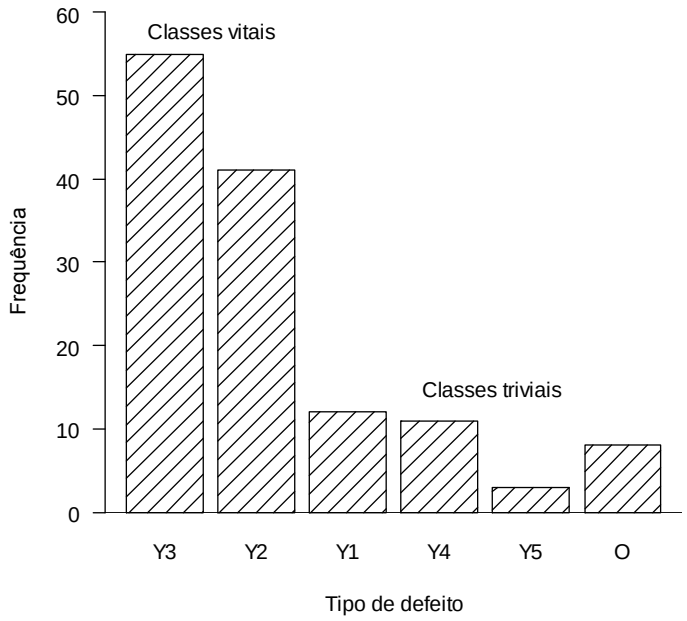


Figura 7.6 - Gráfico de Pareto resultante após a última personalização.

Na sequência, serão mostradas outras opções de personalização que existem e podem ser feitas de acordo com a preferência e a finalidade.

Cores dos Hachurados e das Barras

Nesse exemplo, a cor branca (`col = NULL`) da barra foi alterada para amarela (`col = 7`) e sem a presença dos hachurados (`density = NULL`) (Figura 7.7):

```
> barplot (freq, names.arg = defeito, col = 7, xlab = "Tipo de defeito", ylab = "Frequência", ylim = c (0, 60), las = 1)
```

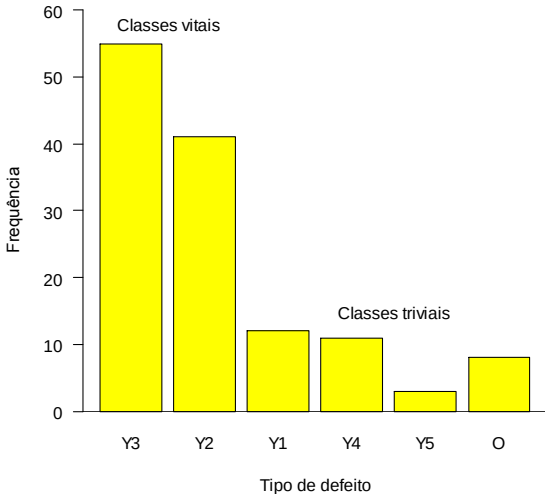


Figura 7.7 - Gráfico de Pareto resultante da alteração da cor das barras.

Por outro lado, a cor do hachurado da barra foi alterada de preta (`col = 1`) para vermelha (`col = 2`) (Figura 7.8):

```
> barplot (y, names.arg = defeito, density = 10, col = 2, xlab = "Tipo de defeito", ylab = "Frequência", ylim = c (0, 60), las = 1)
```

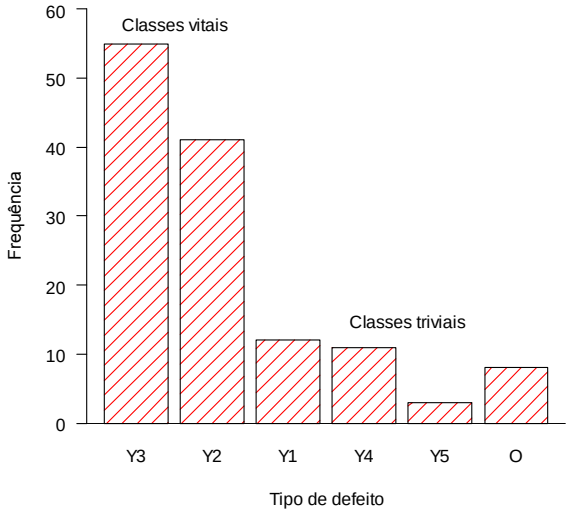


Figura 7.8 - Gráfico de Pareto resultante da alteração da cor do hachurado da barra.

Espaçamento das Barras

Para alterar o espaçamento entre as barras, precisa-se mudar o valor do argumento `"space"`. A sua ausência indica que o espaçamento se encontra no valor padrão, que equivale a 0,2. Sendo assim, foi alterado o espaçamento de 0,2 para 0,5. Para isso, adicionou-se o argumento `space = 0.5`, conforme o seguinte comando (Figura 7.9):

```
> barplot (y, names.arg = defeito, density = 10, col = 1, space = 0.5, xlab = "Tipo de defeito", ylab = "Frequência", ylim = c (0, 60), las = 1)
```

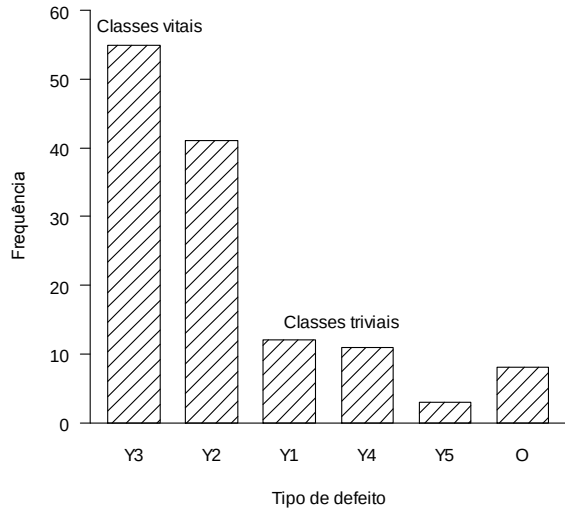


Figura 7.9 - Gráfico de Pareto resultante da alteração do espaçamento das barras.

Capítulo 8

Diagrama de Causa e Efeito

Introdução

O diagrama de causa e efeito (DCE) é uma ferramenta utilizada para apresentar a relação existente entre um resultado do processo (efeito) e os fatores (causas) do mesmo processo que, por razões técnicas, possam afetar o resultado considerado.

O exemplo a ser utilizado para construção do DCE foi o mesmo utilizado para construção do gráfico de Pareto. Nesse caso, a construção do DCE tem como objetivo verificar se pelos menos uma das combinações entre os diferentes níveis (causas secundárias) dos fatores controláveis A e B (causas primárias) podem resultar na solução do problema (efeito) mais importante, o qual foi definido pelo gráfico de Pareto como a variável-resposta Y_3 (RIBEIRO JÚNIOR, 2012, 2013) (Tabela 7.1).

Entrada de Dados

Para construção do DCE, deve-se, primeiro realizar a mudança do diretório para a pasta onde se encontra o arquivo de dados (*.xlsx*) e, posteriormente, fazer a entrada de dados, como segue:

```
> library (openxlsx)
> dcap7e1 = read.xlsx ("cap7e1.xlsx")
> attach (dcap7e1)
> dcap7e1
```

Após realizar a entrada de dados, analisa-se a função `cause.and.effect` do pacote `qcc` (SCRUCCA, 2004), responsável por auxiliar na construção do DCE:

```
> library (qcc)
> help (cause.and.effect)
cause.and.effect (cause, effect, title = "Cause-and-Effect
diagram", cex = c (1, 0.9, 1), font = c (1, 3, 2))
```

Os argumentos da função `cause.and.effect` são definidos por:

`cause`: lista das causas primárias e das causas secundárias entre aspas;

`effect`: título do efeito entre aspas;

`title`: título do gráfico entre aspas, sendo que `title = ""` indica sem título;

`cex`: vetor com os tamanhos das fontes dos textos na ordem especificada (causa primária, causa secundária e efeito), sendo o tamanho normal igual a `1`; e

`font`: vetor com os tipos das fontes dos textos na ordem especificada (causa primária, causa secundária e efeito), podendo ser normal (`1`), negrito (`2`), itálico (`3`) e negrito e itálico (`4`).

DCE Padrão

Para elaborar o DCE, considerando as causas primárias como A e B, as causas secundárias de A como a_1, a_2, a_3 e a_4, as causas secundárias de B como b_1, b_2, b_3 e b_4 e o efeito como Y_3 (Figura 8.1), foi digitado no *R* o seguinte comando:

```
> cause.and.effect (cause = list (A = c ("a1", "a2", "a3", "a4"), B = c ("b1", "b2", "b3", "b4")), effect = "Y3", title = "", cex = c (1, 1, 1), font = c (1, 1, 1))
```

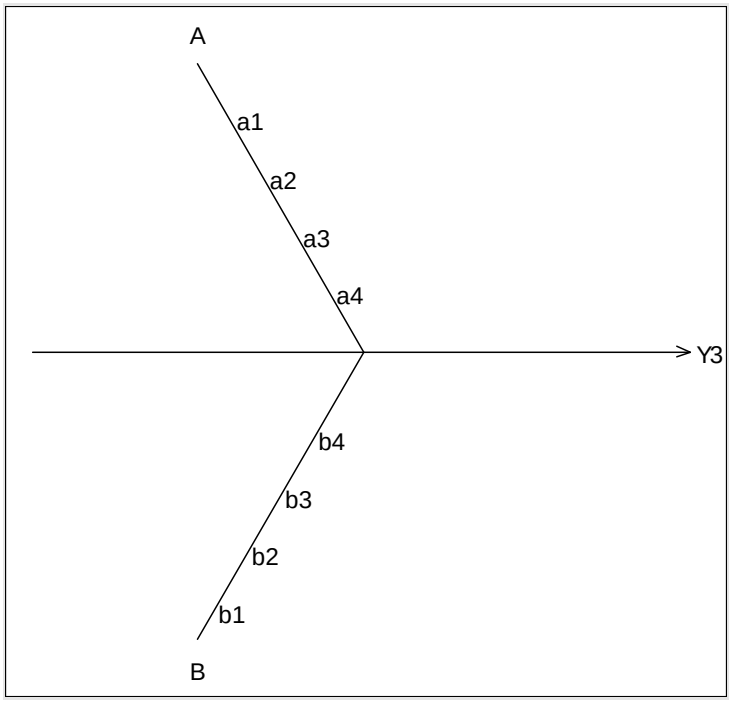


Figura 8.1 - Diagrama de causa e efeito resultante do *R*.

Capítulo 9

Gráficos de Controle por Atributos

Introdução

Os gráficos de controle por atributos de *Shewhart* são divididos em quatro tipos: np, p, c e u. E para construção deles, foi utilizado o exemplo de aplicação (RIBEIRO JÚNIOR, 2013) apresentado na Tabela 9.1.

Tabela 9.1 - Arquivo de dados *cap9e1.xlsx*

	A	B	C
1	sr	n	y
2	1	100	5
3	2	100	2
4	3	100	7
5	4	100	3
6	5	100	6
7	6	100	2

Entrada de Dados

Para a entrada dos dados, deve-se, primeiro realizar a mudança do diretório para a pasta onde se encontra o arquivo de dados (*.xlsx*) e, posteriormente, digitar o que se segue:

```
> library (openxlsx)
> dcap9e1 = read.xlsx ("cap9e1.xlsx")
> attach (dcap9e1)
> dcap9e1
```

Após realizar a entrada de dados, analisa-se a função `qcc` do próprio pacote `qcc` (SCRUCCA, 2004):

```
> library (qcc)
> help (qcc)
qcc (data, type, sizes, center, std.dev, limits, data.name,
labels, newdata, newsizes, newdata.name, newlabels, nsigmas = 3,
confidence.level, rules = shewhart.rules, plot = TRUE, ...)
```

Alguns dos argumentos da função `qcc` são definidos por:

`data`: uma matriz ou um vetor contendo os valores observados da variável-resposta Y, sendo que cada linha de valores da matriz ou de valor do vetor se refere a uma amostra ou um subgrupo racional;

`type`: tipos de gráficos de controle, sendo divididos em `Xbar`, `R`, `S`, `xbar.one`, `p`, `np`, `c` e `u`;

`size`: um valor ou um vetor de valores especificando os respectivos tamanhos dos subgrupos racionais;

`center`: média ou valor-alvo de Y e, quando não especificada, calcula-se a média amostral;

`stdv.dev`: desvio-padrão de Y e, quando não especificado, calcula-se o desvio-padrão amostral que ocorre dentro dos subgrupos racionais; e

`nsigmas`: número (k) de desvios-padrão utilizado para calcular os limites de controle, sendo o padrão igual a `3`.

Gráfico de Controle np

Ele monitora o número de unidades não conformes (Y) em subgrupos racionais de tamanhos constantes. Para sua construção, existem duas opções: sem especificações ou com especificações da média de Y, do desvio-padrão de Y e/ou do número de desvios-padrão.

Para construir o gráfico de controle np sem especificações no *R*, basta digitar o seguinte comando (Figura 9.1):

```
> qcc (y, type = "np", sizes = n)
```

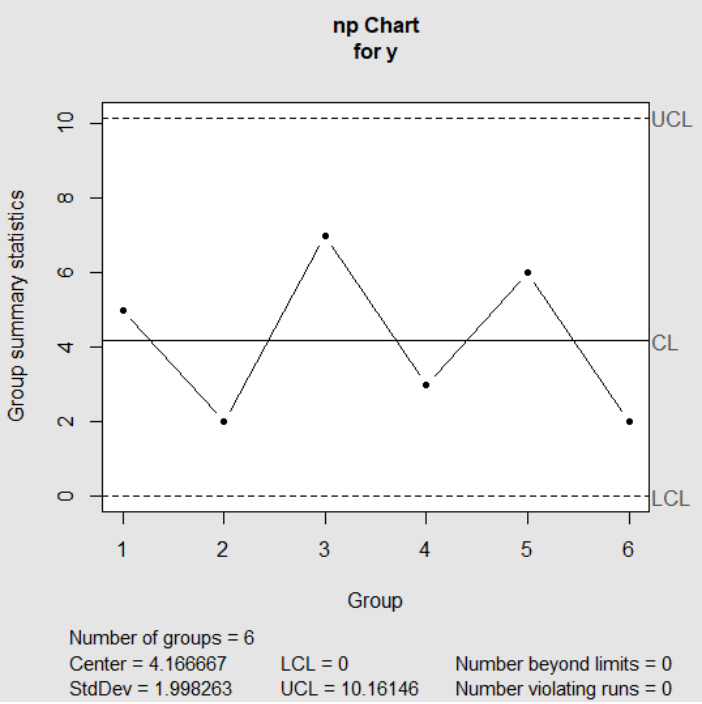


Figura 9.1 - Gráfico de controle np sem especificações.

E para construir o gráfico de controle np com especificações no *R*, basta definilas adequada e respectivamente para os argumentos `center` (np), `std.dev` [np(1 – p)] e/ou `nsigmas` (k) (Figura 9.2):

```
> qcc (y, type = "np", sizes = n, nsigmas = 3, center = 4,
std.dev = 1)
```

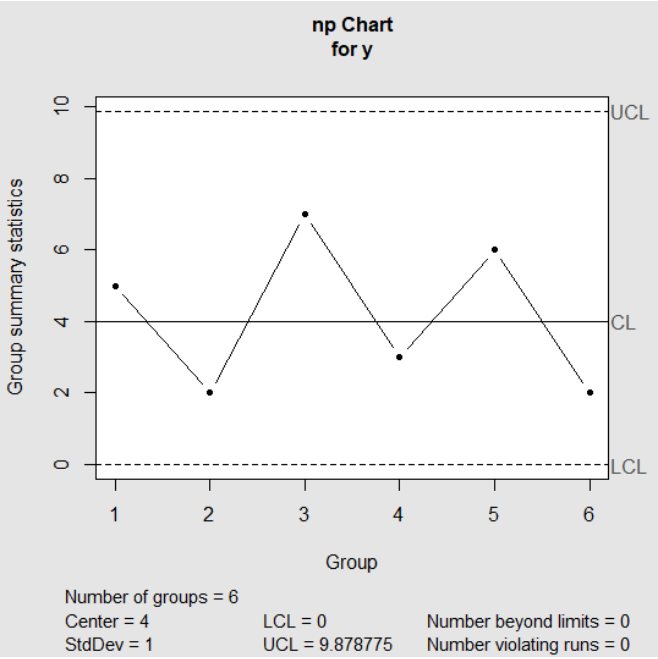


Figura 9.2 - Gráfico de controle np com especificações.

Gráfico de Controle p

Ele monitora a proporção de unidades não conformes (p) em subgrupos racionais de tamanhos constantes ou variáveis. Para sua construção, existem, também, duas opções: sem especificações ou com especificações da média de p, do desvio-padrão de p e/ou do número de desvios-padrão.

Para construir o gráfico de controle p sem especificações no *R*, basta digitar o seguinte comando (Figura 9.3):

```
> qcc (y, type = "p", sizes = n)
```

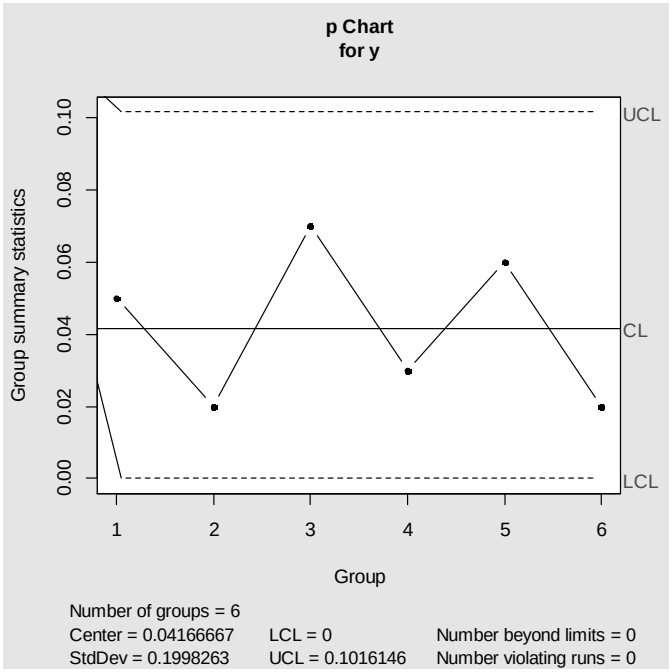


Figura 9.3 - Gráfico de controle p sem especificações.

E para construir o gráfico de controle p com especificações no *R*, basta definilas adequada e respectivamente para os argumentos `center` (p), `std.dev` [p(1 – p)/n] e/ou `nsigmas` (k) (Figura 9.4):

```
> qcc (y, type = "p", sizes = n, nsigmas = 3, center = 0.05,
std.dev = 0.2)
```

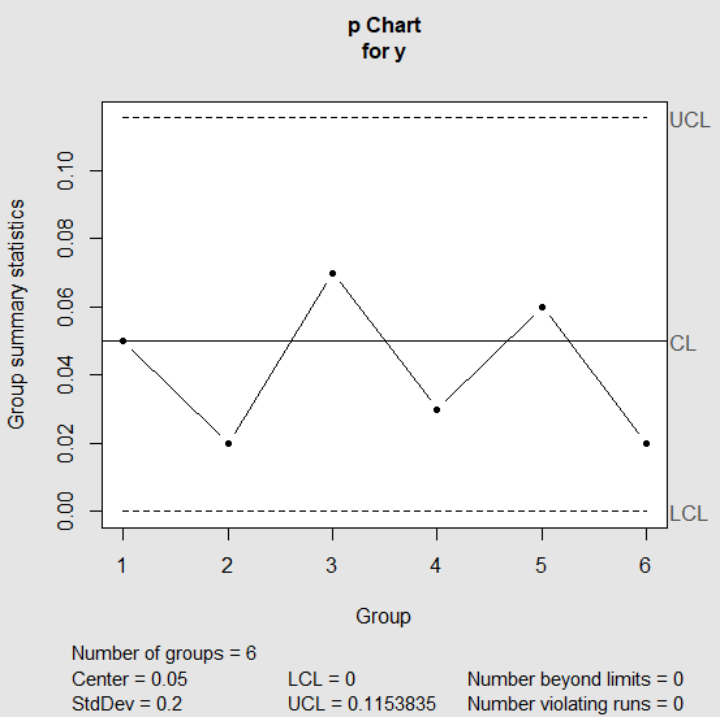


Figura 9.4 - Gráfico de controle p com especificações.

Gráfico de Controle c

Ele monitora o número de não conformidades (c) por subgrupo racional de tamanho constante. Para sua construção, existem duas opções: sem especificações ou com especificações da média de c, do desvio-padrão de c e/ou do número de desvios-padrão.

Para construir o gráfico de controle c sem especificações no *R*, basta digitar o seguinte comando (Figura 9.5):

```
> qcc (y, type = "c", sizes = 1)
```

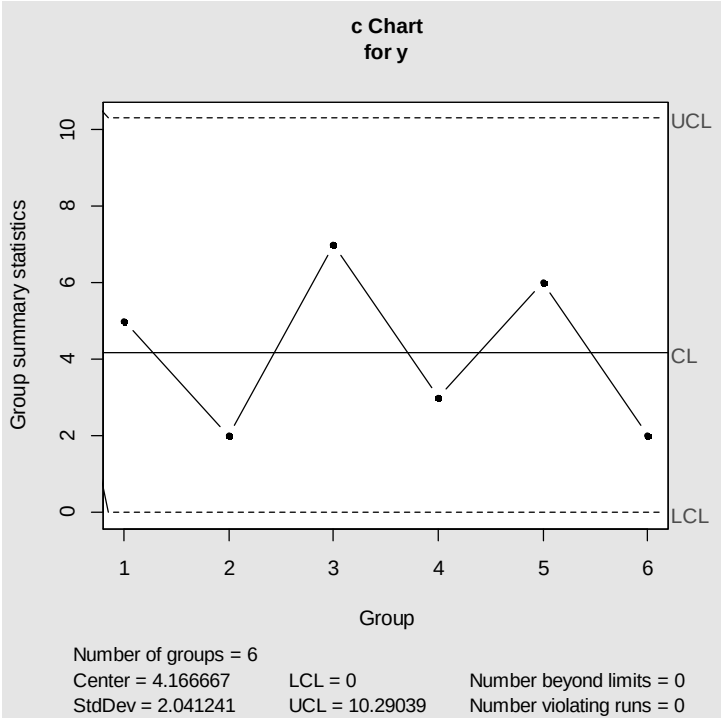


Figura 9.5 - Gráfico de controle c sem especificações.

E para construir o gráfico de controle c com especificações no *R*, basta definilas adequada e respectivamente para os argumentos `center` (média do número de não conformidades por subgrupo racional), `std.dev` (desvio-padrão do número de não conformidades por subgrupo racional) e/ou `nsigmas` (k) (Figura 9.6):

```
> qcc (y, type = "c", sizes = 1, nsigmas = 3, center = 5)
```

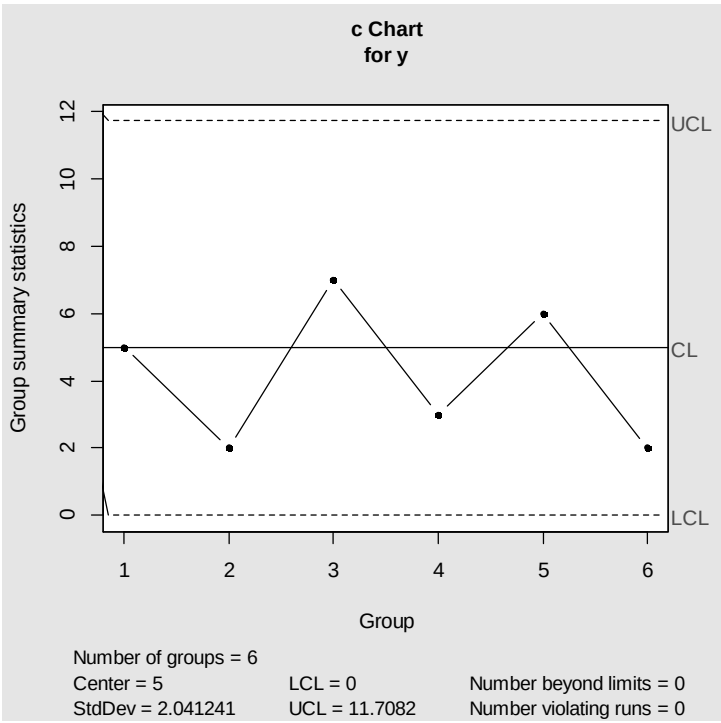


Figura 9.6 - Gráfico de controle c com especificações.

Gráfico de Controle u

Ele monitora o número médio de não conformidades por unidade (u) de subgrupos racionais de tamanhos constantes ou variáveis. Para sua construção, existem duas opções: sem especificações ou com especificações da média de u, do desvio-padrão de u e/ou do número de desvios-padrão.

Para construir o gráfico de controle u sem especificações no *R*, basta digitar o seguinte comando (Figura 9.7):

```
> qcc (y, type = "u", sizes = n)
```

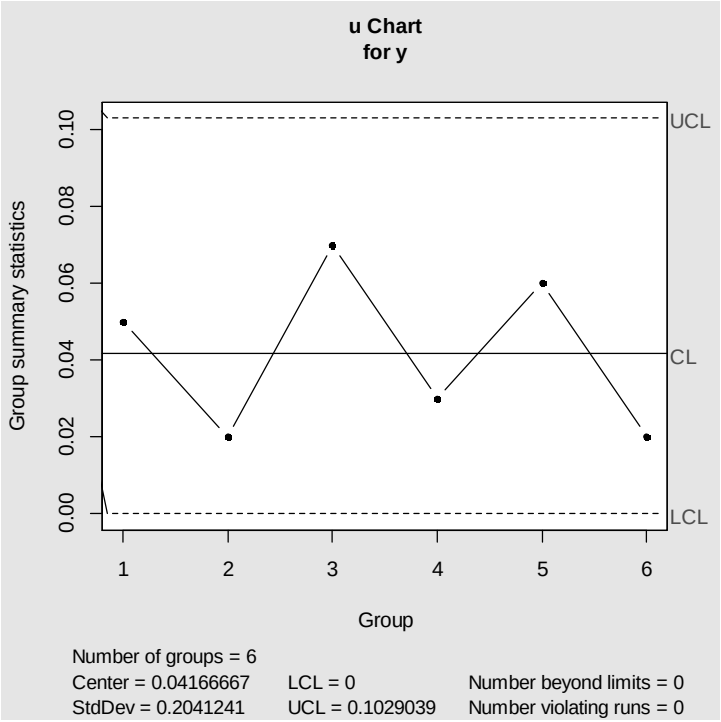


Figura 9.7 - Gráfico de controle u sem especificações.

E para construir o gráfico de controle u com especificações no *R*, basta definilas adequada e respectivamente para os argumentos `center` (média do número médio de não conformidades por unidade), `std.dev` (desvio-padrão do número médio de não conformidades por unidade) e/ou `nsigmas` (k) (Figura 9.8):

```
> qcc (y, type = "u", sizes = 1, nsigmas = 3, center = 0.05)
```

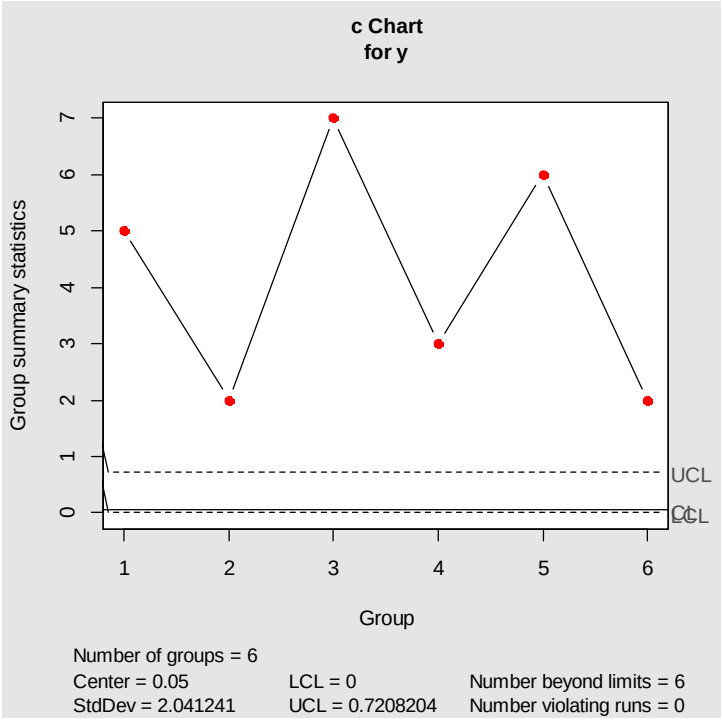


Figura 9.8 - Gráfico de controle u com especificações.

Capítulo 10

Gráficos de Controle por Variáveis

Introdução

Os gráficos de controle por variáveis de *Shewhart* para o monitoramento da média e da variabilidade são divididos em dois tipos: sem (observações individuais) e com repetições por subgrupo racional.

Observações Individuais

Para construção dos gráficos de controle da amplitude móvel (AM) para o monitoramento da variabilidade e da medida individual (Y) para o monitoramento da média, foi utilizado o exemplo de aplicação (RIBEIRO JÚNIOR, 2013) apresentado na Tabela 10.1.

Tabela 10.1 - Arquivo de dados *cap10e1.xlsx*

	A	B
1	sr	y
2	1	33,8
3	2	33,1
4	3	34
5	4	33,8
6	5	33,5
7	6	34
8	7	33,7
9	8	33,3
10	9	33,5
11	10	33,2
12	11	33,6
13	12	33
14	13	33,5
15	14	33,1
16	15	33,8
17	16	33,5
18	17	33,3
19	18	33,4
20	19	33,3
21	20	34,7
22	21	34,8
23	22	34,6
24	23	35
25	24	34,8
26	25	34,5
27	26	34,7
28	27	34,3
29	28	34,6
30	29	34,5
31	30	35

Entrada de Dados

Antes, deve-se realizar a mudança do diretório para a pasta onde se encontra o arquivo de dados (*.xlsx*) e, posteriormente, ativar o pacote `qcc` e proceder a entrada dos dados, como segue:

```
> library (qcc)
> library (openxlsx)
> dcap10e1 = read.xlsx ("cap10e1.xlsx")
> attach (dcap10e1)
> dcap10e1
```

Gráfico de Controle AM

O gráfico de controle da amplitude móvel recebe o nome de `gcam` no *R*. Sendo assim, deve-se utilizar a função `gcam` responsável por criar uma nova coluna que contém as amplitudes móveis. Dentro dela, o argumento `cbind` combina as colunas, colocando os valores adjacentes lado a lado. Já o argumento `ncol` determina o número de colunas. Desse modo, o primeiro passo após a entrada de dados será de digitar no *R*, o seguinte comando:

```
> gcam = matrix (cbind (y [1: length (y) - 1], y [2: length (y)]), ncol = 2)
```

Para sua construção, existem duas opções: sem especificações ou com especificações da média de AM, do desvio-padrão de AM e/ou do número de desvios-padrão.

Para construir o gráfico de controle AM sem especificações no *R*, basta digitar o seguinte comando (Figura 10.1):

```
> qcc (gcam, type = "R")
```

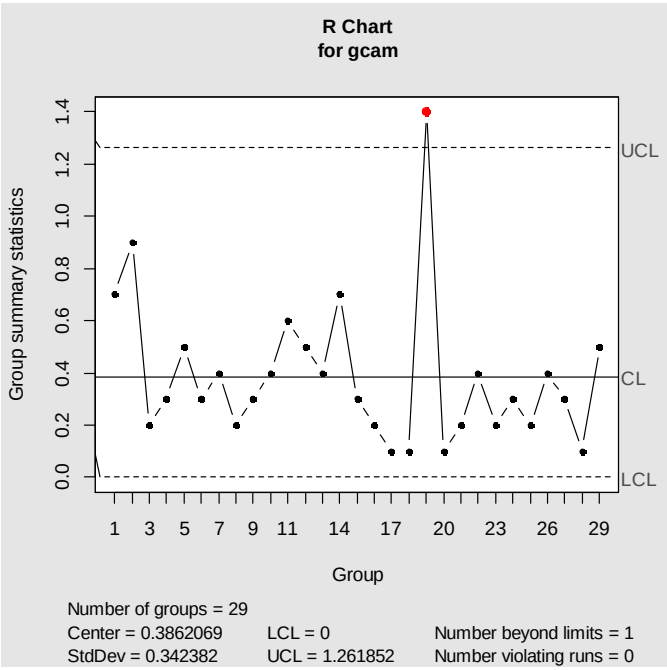


Figura 10.1 - Gráfico de controle AM sem especificações.

E para construir o gráfico de controle AM com especificações no *R*, basta defini-las adequada e respectivamente para os argumentos `center` (μ_{AM}), `std.dev` (σ_{AM}) e/ou `nsigmas` (k) (Figura 10.2):

```
> qcc (gcam, type = "R", nsigmas = 3, center = 0.4, std.dev = 0.35)
```

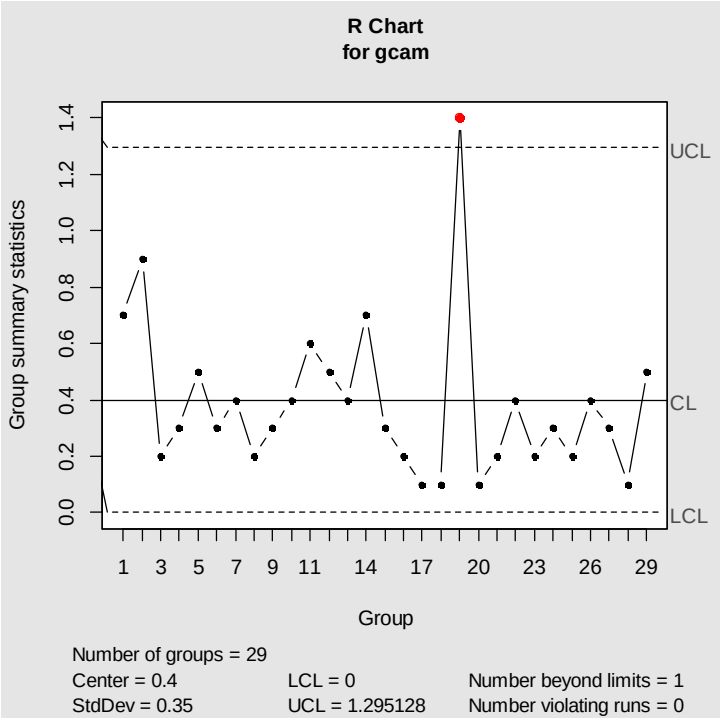


Figura 10.2 - Gráfico de controle AM com especificações.

Gráfico de Controle Y

Para construção do gráfico de controle da medida individual, existem duas opções: sem especificações ou com especificações da média de Y, do desvio-padrão de Y e/ou do número de desvios-padrão.

Para construir o gráfico de controle Y sem especificações no *R*, basta digitar o seguinte comando (Figura 10.3):

```
> qcc (y, type = "xbar.one")
```

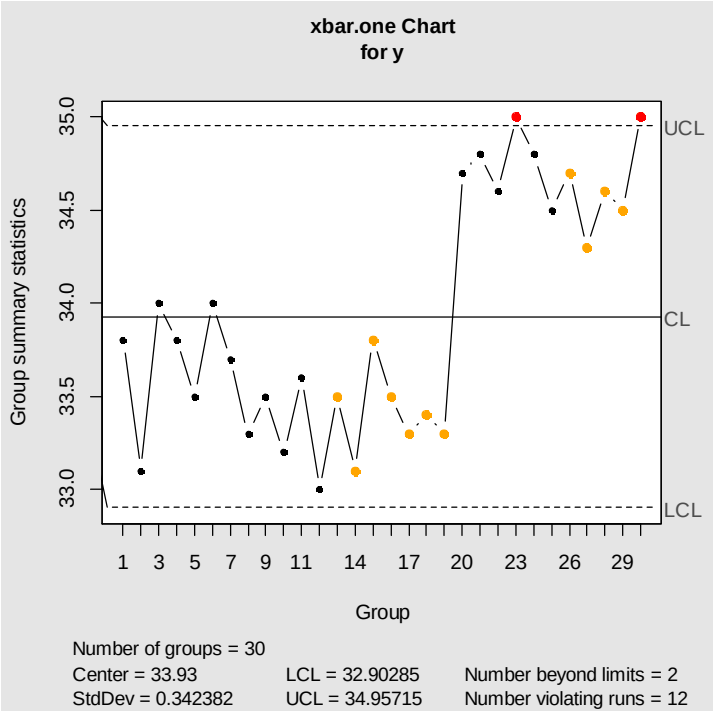


Figura 10.3 - Gráfico de controle Y sem especificações.

E para construir o gráfico de controle Y com especificações no *R*, basta definilas adequada e respectivamente para os argumentos `center` (μ_Y), `std.dev` (σ_Y) e/ou `nsigmas` (k) (Figura 10.4):

```
> qcc (gcam, type = "xbar.one", nsigmas = 3, center = 35,
std.dev = 0.35)
```

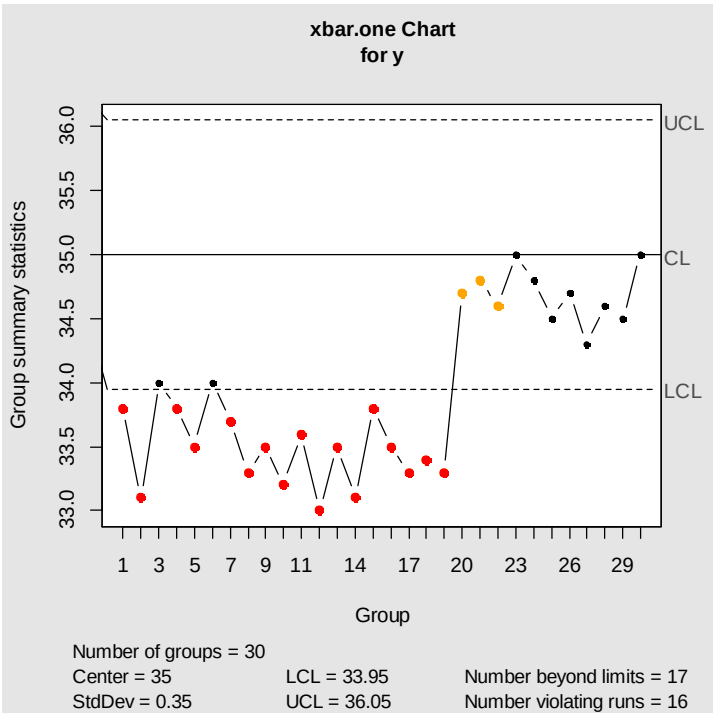


Figura 10.4 - Gráfico de controle Y com especificações.

Repetições

Para construção dos gráficos de controle da amplitude (R) e do desvio-padrão (S) para o monitoramento da variabilidade e da média ($\overline{Y}$) para o monitoramento da média, foi utilizado o exemplo de aplicação (RIBEIRO JÚNIOR, 2013) apresentado na Tabela 10.2.

Tabela 10.2 - Arquivo de dados *cap10e2.xlsx*

	A	B
1	sr	y
2	1	7,1
3	1	7,09
4	1	7,1
5	1	7,11
6	2	7,12
7	2	7,11
8	2	7,1
9	2	7,12
10	3	7,09
11	3	7,1
12	3	7,11
13	3	7,12
14	4	7,11
15	4	7,1
16	4	7,1
17	4	7,11
18	5	7,07
19	5	7,09
20	5	7,11
21	5	7,13
22	6	7,1
23	6	7,11
24	6	7,09
25	6	7,08

Entrada de Dados

Antes, deve-se realizar a mudança do diretório para a pasta onde se encontra o arquivo de dados (*.xlsx*) e, posteriormente, ativar o pacote `qcc` e proceder a entrada dos dados, como segue (Figura 10.5):

```
> library (qcc)
> library (openxlsx)
> dcap10e2 = read.xlsx ("cap10e2.xlsx")
> attach (dcap10e2)
> dcap10e2
```

Para construção dos gráficos de controle por variáveis de *Shewhart* no *R*, as repetições de cada subgrupo racional devem estar dispostas em colunas e não em linhas (Tabela 10.2). Sendo assim, é preciso criar um novo objeto que armazena os dados, cujas repetições estejam dispostas em colunas (Figura 10.6):

```
> dvargc = qcc.groups (y, sr)
```

```
   sr    y
1   1 7.10
2   1 7.09
3   1 7.10
4   1 7.11
5   2 7.12
6   2 7.11
7   2 7.10
8   2 7.12
9   3 7.09
10  3 7.10
11  3 7.11
12  3 7.12
13  4 7.11
14  4 7.10
15  4 7.10
16  4 7.11
17  5 7.07
18  5 7.09
19  5 7.11
20  5 7.13
21  6 7.10
22  6 7.11
23  6 7.09
24  6 7.08
> 
```

Figura 10.5 - Repetições por subgrupo racional em diferentes linhas.

```
  [,1] [,2] [,3] [,4]
1 7.10 7.09 7.10 7.11
2 7.12 7.11 7.10 7.12
3 7.09 7.10 7.11 7.12
4 7.11 7.10 7.10 7.11
5 7.07 7.09 7.11 7.13
6 7.10 7.11 7.09 7.08
> 
```

Figura 10.6 - Repetições por subgrupo racional em diferentes colunas.

Gráfico de Controle R

Para construção do gráfico de controle da amplitude, existem duas opções: sem especificações ou com especificações da média de R, do desvio-padrão de R e/ou do número de desvios-padrão.

Para construir o gráfico de controle R sem especificações no *R*, basta digitar o seguinte comando (Figura 10.7):

```
> qcc (dvargc, type = "R")
```

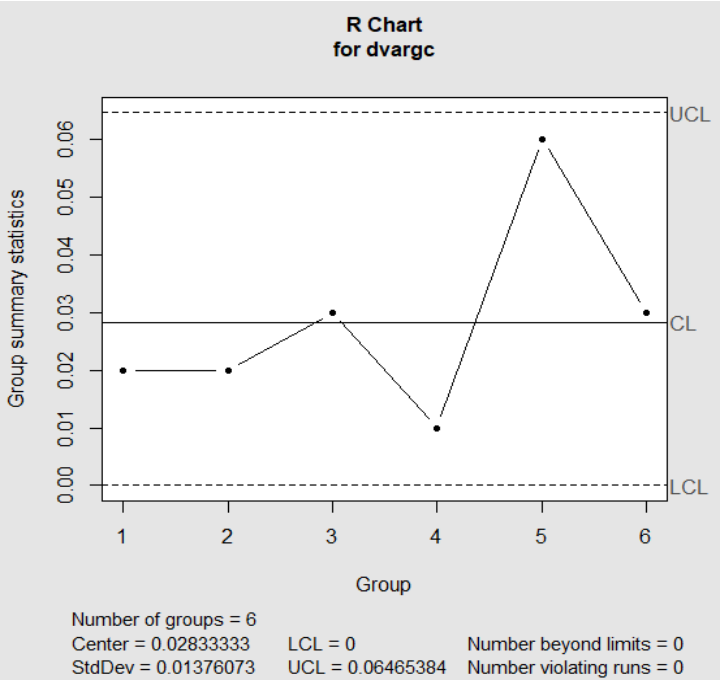


Figura 10.7 - Gráfico de controle R sem especificações.

E para construir o gráfico de controle R com especificações no *R*, basta definilas adequada e respectivamente para os argumentos `center` (μ_R), `std.dev` (σ_R) e/ou `nsigmas` (k) (Figura 10.8):

```
> qcc (dvargc, type = "R", nsigmas = 3, center = 0.05, std.dev = 0.02)
```

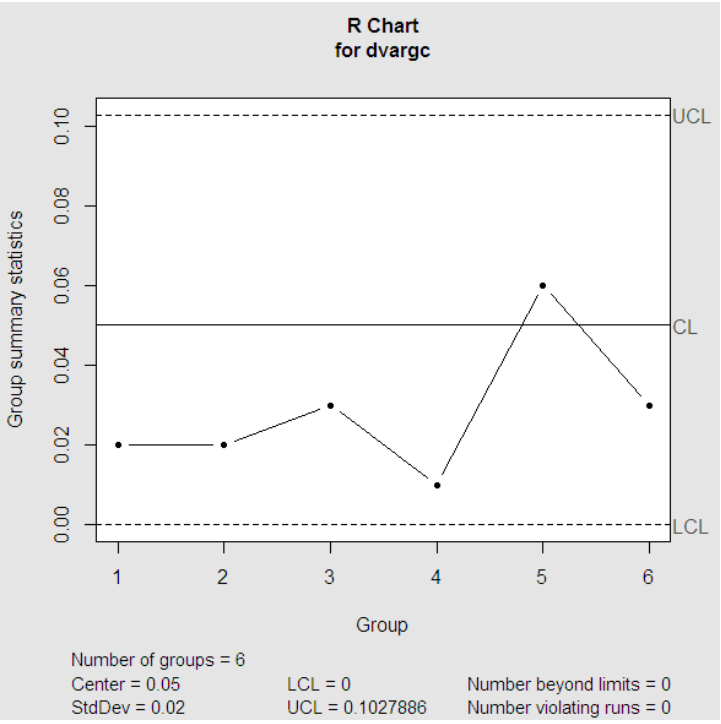


Figura 10.8 - Gráfico de controle R com especificações.

Gráfico de Controle S

Para construção do gráfico de controle do desvio-padrão, existem duas opções: sem especificações ou com especificações da média de S, do desvio-padrão de S e/ou do número de desvios-padrão.

Para construir o gráfico de controle S sem especificações no *R*, basta digitar o seguinte comando (Figura 10.9):

```
> qcc (dvargc, type = "S")
```

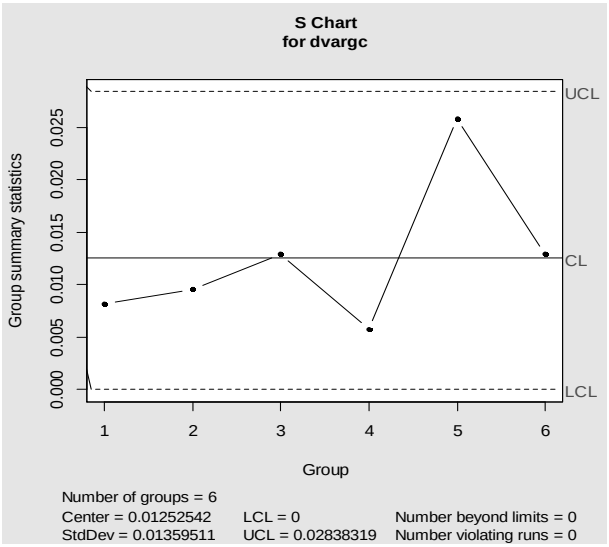


Figura 10.9 - Gráfico de controle S sem especificações.

E para construir o gráfico de controle S com especificações no *R*, basta defini-las adequada e respectivamente para os argumentos `center` (μ_S), `std.dev` (σ_S) e/ou `nsigmas` (k) (Figura 10.10):

```
> qcc (dvargc, type = "S", nsigmas = 3, center = 0.013, std.dev
= 0.014)
```

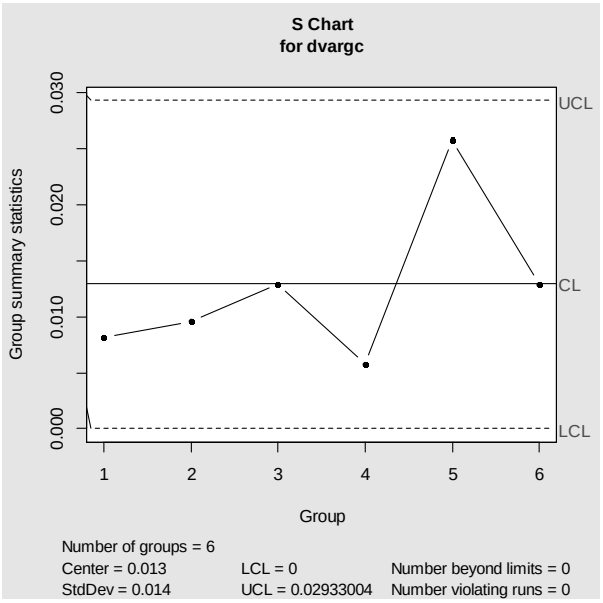


Figura 10.10 - Gráfico de controle S com especificações.

Gráfico de Controle $\overline{Y}$

Para construção do gráfico de controle da média, existem duas opções: sem especificações ou com especificações da média de $\overline{Y}$, do desvio-padrão de $\overline{Y}$ e/ou do número de desvios-padrão.

Para construir o gráfico de controle $\overline{Y}$ sem especificações no *R*, basta digitar o seguinte comando (Figura 10.11):

```
> qcc (dvargc, type = "xbar")
```

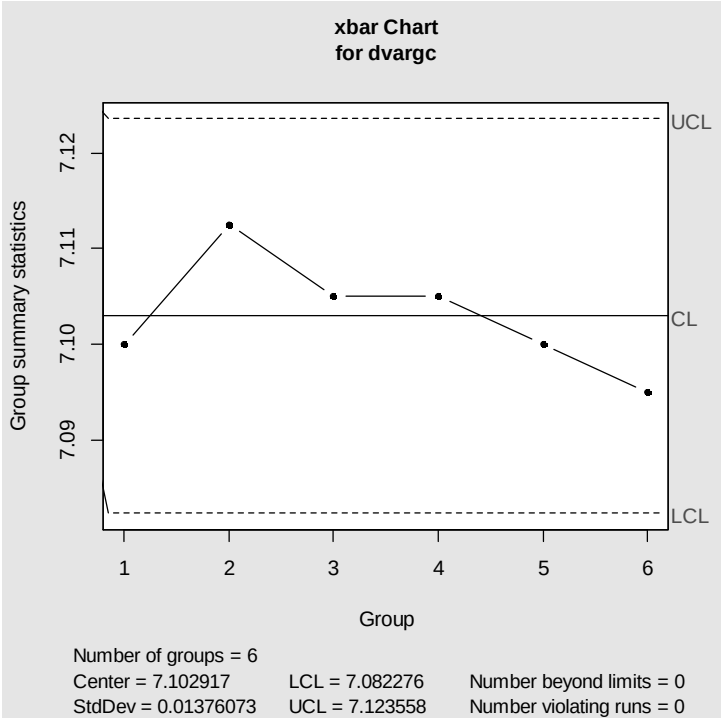


Figura 10.11 - Gráfico de controle $\overline{Y}$ sem especificações.

E para construir o gráfico de controle $\overline{Y}$ com especificações no *R*, basta definilas adequada e respectivamente para os argumentos `center` ($\mu_{\overline{Y}}$), `std.dev` ($\sigma_{\overline{Y}}$) e/ou `nsigmas` (k) (Figura 10.12):

```
> qcc (dvargc, type = "xbar", nsigmas = 3, center = 7, std.dev =
0.015)
```

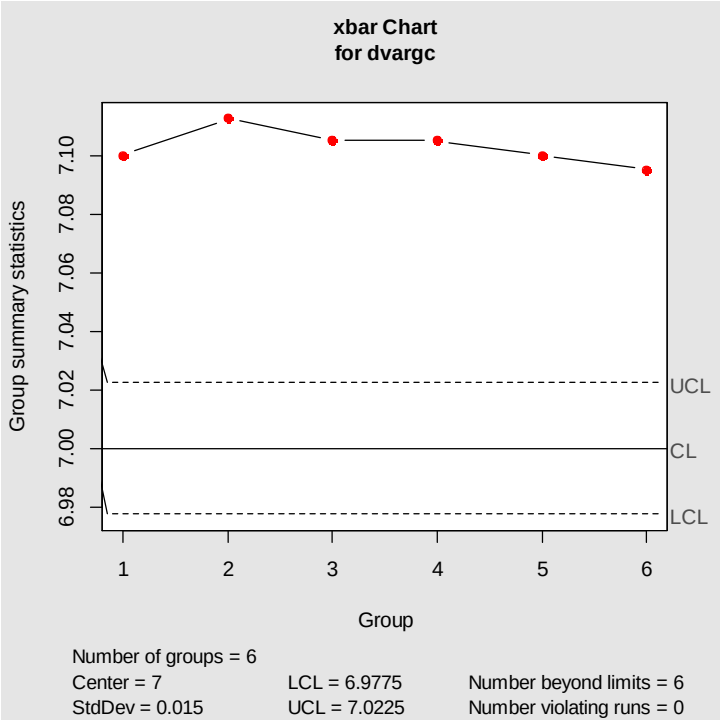


Figura 10.12 - Gráfico de controle $\overline{Y}$ com especificações.

Referências Bibliográficas

BASTOS, R. L.; FERREIRA, E. B. CEPpt: um pacote R para o controle estatístico de processo. **Sigmae**, v. 1, p. 1-17, 2012.

CANO, E. L.; MOGUERZA, J. M.; PRIETO, M.; REDCHUCK, A. **Six sigma tools for quality control and improvement**. R package version 0.9-52, 2018. 47 p.

RIBEIRO JÚNIOR, J. I. **Métodos estatísticos aplicados à melhoria da qualidade**. Viçosa: Editora UFV, 2012. 385 p.

RIBEIRO JÚNIOR, J. I. **Métodos estatísticos aplicados ao controle da qualidade**. Viçosa: Editora UFV, 2013. 274 p.

ROTH, T. **Statistical methods for quality science**. R package version 1.55, 2016. 147 p.

SCRUCCA, L. qcc: an R package for quality control charting and statistical process control. **R News**, v. 4, p. 11-17, 2004.

WALKER, A. **Read, write and edit xlsx files**. R package version 4.1.0.1, 2019. 77 p.