Uma primeira sessão com o R

Paulo Justiniano Ribeiro Junior

Última atualização: 5 de maio de 2011

Vamos começar "experimentando o R", para ter uma idéia de seus recursos e a forma de trabalhar com este programa. Para isto vamos rodar e estudar os comandos mostrados no texto e seus resultados para nos familiarizar com aspectos básicos do programa. Ao longo deste curso iremos ver com mais detalhes o uso do programa R.

Siga os seguintes passos:

1. inicie o R em seu computador;
2. voce verá uma janela de comandos com o símbolo >,
   este é o *prompt* do R indicando que o programa está pronto para receber comandos;
3. a seguir digite (ou "recorte e cole") os comandos mostrados abaixo.

No restante deste texto vamos seguir as seguintes convenções.

- comandos do R são sempre mostrados em fontes do tipo *`slanted verbatim como esta`*, e precedidas pelo símbolo `>`,
- saídas do R são sempre mostrados em fontes do tipo `verbatim como esta`,
- linhas iniciadas pelo símbolo `#` são comentários e são ignoradas pelo R.

1 Criando, listando e apagando objetos

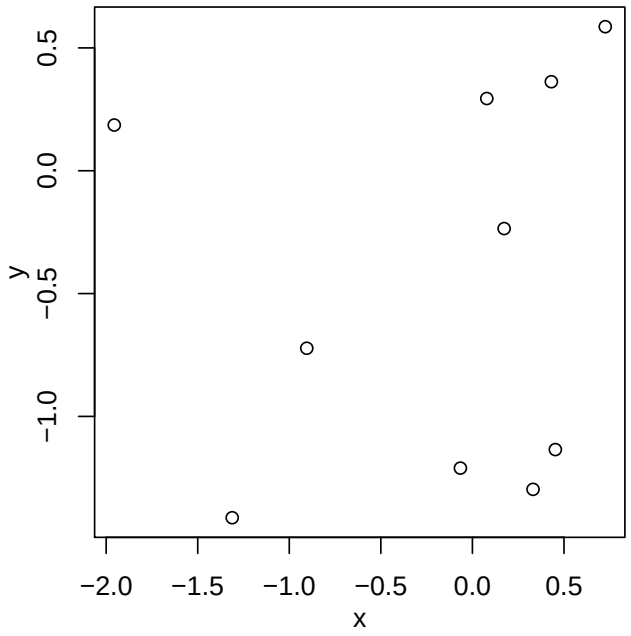
Vamos primeiro gerar dois vetores `x` e `y` de números pseudo-aleatórios e inspecionar os valores gerados

```
> x <- rnorm(10)
> x

 [1] -0.06553367  0.43107470  0.45285227  0.17325535 -1.31158926 -1.95573464
 [7] -0.90443231  0.72535286  0.33099983  0.07852864

> y <- rnorm(x)
> y

 [1] -1.2103902  0.3623927 -1.1348481 -0.2352729 -1.4122960  0.1861627
 [7] -0.7224462  0.5864064 -1.2967893  0.2939657
```

Vamos colocar os pontos em um gráfico. Note que numa sessão interativa a janela gráfica se abrirá automaticamente. O seu gráfico pode ser diferente do meu. Por que?

Verificando os objetos existentes na área de trabalho.

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




```
> ls()
```

```
[1] "x" "y"
```

Removendo objetos que não são mais necessários.

```
> rm(x, y)
> ls()
```

```
character(0)
```

2 Estatística descritiva

O livro *Estatística Básica* de W. Bussab e P. Morettin traz no primeiro capítulo um conjunto de dados hipotético de atributos de 36 funcionários da companhia "Milsa". Os dados estão reproduzidos na tabela **??**. Veja o livro para mais detalhes sobre este dados.

O que queremos aqui é ilustrar alguns comandos para análise descritiva no programa R. Estes são dados no "estilo planilha", com variáveis de diferentes tipos: categóricas e numéricas (qualitativas e quantitativas). Para entrar com estes dados no diretamente no Rpodemos usar o editor que vem com o programa, ou estes dados podem estar digitados em algum outro formato. Para digitar rapidamente estes dados é mais fácil usar códigos para as variáveis categóricas. Desta forma, na coluna de estado civil vamos digitar o código 1 para *solteiro* e 2 para *casado*. Fazemos de maneira similar com as colunas *Grau de Instrução* e *Região de Procedência*.

Vamos usar um arquivo de dados no formato texto chamado `milsa.dat` que já contém estes dados digitados.

```
> file.show("milsa.dat")
```

Para ler os dados usamos o comando de importação de dados de arquivos em formato texto. Em seguida exibimos as 10 primeira linhas.

Tabela 1: Dados de Bussab & Morettin

Funcionário	Est. Civil	Instrução	N^o Filhos	Salário	Ano	Mês	Região
1	solteiro	1o Grau	-	4.00	26	3	interior
2	casado	1o Grau	1	4.56	32	10	capital
3	casado	1o Grau	2	5.25	36	5	capital
4	solteiro	2o Grau	-	5.73	20	10	outro
5	solteiro	1o Grau	-	6.26	40	7	outro
6	casado	1o Grau	0	6.66	28	0	interior
7	solteiro	1o Grau	-	6.86	41	0	interior
8	solteiro	1o Grau	-	7.39	43	4	capital
9	casado	2o Grau	1	7.59	34	10	capital
10	solteiro	2o Grau	-	7.44	23	6	outro
11	casado	2o Grau	2	8.12	33	6	interior
12	solteiro	1o Grau	-	8.46	27	11	capital
13	solteiro	2o Grau	-	8.74	37	5	outro
14	casado	1o Grau	3	8.95	44	2	outro
15	casado	2o Grau	0	9.13	30	5	interior
16	solteiro	2o Grau	-	9.35	38	8	outro
17	casado	2o Grau	1	9.77	31	7	capital
18	casado	1o Grau	2	9.80	39	7	outro
19	solteiro	Superior	-	10.53	25	8	interior
20	solteiro	2o Grau	-	10.76	37	4	interior
21	casado	2o Grau	1	11.06	30	9	outro
22	solteiro	2o Grau	-	11.59	34	2	capital
23	solteiro	1o Grau	-	12.00	41	0	outro
24	casado	Superior	0	12.79	26	1	outro
25	casado	2o Grau	2	13.23	32	5	interior
26	casado	2o Grau	2	13.60	35	0	outro
27	solteiro	1o Grau	-	13.85	46	7	outro
28	casado	2o Grau	0	14.69	29	8	interior
29	casado	2o Grau	5	14.71	40	6	interior
30	casado	2o Grau	2	15.99	35	10	capital
31	solteiro	Superior	-	16.22	31	5	outro
32	casado	2o Grau	1	16.61	36	4	interior
33	casado	Superior	3	17.26	43	7	capital
34	solteiro	Superior	-	18.75	33	7	capital
35	casado	2o Grau	2	19.40	48	11	capital
36	casado	Superior	3	23.30	42	2	interior

```
> milsa <- read.table("dados/milsa.dat", head = T, dec = ",", na.st = ".")
> milsa[1:10, ]
```

```
   funcionario civil instrucao filhos salario ano mes regiao
1            1     1         1     NA    4.00  26   3      1
2            2     2         1      1    4.56  32  10      2
3            3     2         1      2    5.25  36   5      2
4            4     1         2     NA    5.73  20  10      3
5            5     1         1     NA    6.26  40   7      3
6            6     2         1      0    6.66  28   0      1
7            7     1         1     NA    6.86  41   0      1
8            8     1         1     NA    7.39  43   4      2
9            9     2         2      1    7.59  34  10      2
10          10     1         2     NA    7.44  23   6      3
```

A planilha digitada como está ainda não está pronta. Precisamos informar para o programa que as variáveis `civil`, `instrucao` e `regiao`, NÃO são numéricas e sim categóricas. No R variáveis categóricas são definidas usando o comando `factor()`, que vamos usar para redefinir nossas variáveis conforme os comandos a seguir. Primeiro redefinimos a variável `civil` com os *rótulos (labels)* solteiro e casado associados aos *níveis (levels)* 1 e 2. Para variável `intrução` usamos o argumento adicional `ordered = TRUE` para indicar que é uma variável ordinal. Na variável `regiao` codificamos assim: 2=capital, 1=interior, 3=outro. Ao final inspecionamos os dados digitando o nome do objeto.

```
> milsa$civil <- factor(milsa$civil, label = c("solteiro", "casado"),
+     levels = 1:2)
> milsa$instrucao <- factor(milsa$instrucao, label = c("1oGrau",
+     "2oGrau", "Superior"), lev = 1:3, ord = T)
> milsa$regiao <- factor(milsa$regiao, label = c("capital", "interior",
+     "outro"), lev = c(2, 1, 3))
> milsa[1:10, ]
```

```
   funcionario    civil instrucao filhos salario ano mes   regiao
1            1 solteiro    1oGrau     NA    4.00  26   3 interior
2            2   casado    1oGrau      1    4.56  32  10  capital
3            3   casado    1oGrau      2    5.25  36   5  capital
4            4 solteiro    2oGrau     NA    5.73  20  10    outro
5            5 solteiro    1oGrau     NA    6.26  40   7    outro
6            6   casado    1oGrau      0    6.66  28   0 interior
7            7 solteiro    1oGrau     NA    6.86  41   0 interior
8            8 solteiro    1oGrau     NA    7.39  43   4  capital
9            9   casado    2oGrau      1    7.59  34  10  capital
10          10 solteiro    2oGrau     NA    7.44  23   6    outro
```

Além disto precisamos definir uma variável única `idade` a partir das variáveis `ano` e `mes` que foram digitadas. Para gerar a variável idade (em anos) fazemos:

```
> milsa$idade <- milsa$ano + milsa$mes/12
> milsa$idade
```

```
 [1] 26.25000 32.83333 36.41667 20.83333 40.58333 28.00000 41.00000 43.33333
 [9] 34.83333 23.50000 33.50000 27.91667 37.41667 44.16667 30.41667 38.66667
[17] 31.58333 39.58333 25.66667 37.33333 30.75000 34.16667 41.00000 26.08333
[25] 32.41667 35.00000 46.58333 29.66667 40.50000 35.83333 31.41667 36.33333
[33] 43.58333 33.58333 48.91667 42.16667
```

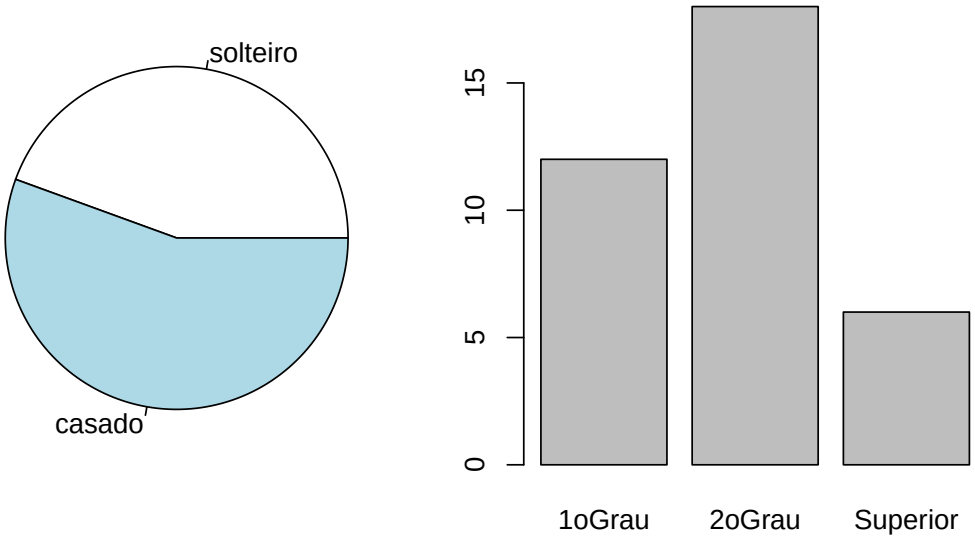
Agora que os dados estão prontos podemos começar a análise descritiva. Inspecionem os comandos a seguir.

```
> is.data.frame(milsa)

[1] TRUE

> names(milsa)

[1] "funcionario" "civil"       "instrucao"   "filhos"      "salario"
[6] "ano"         "mes"         "regiao"      "idade"

> dim(milsa)

[1] 36  9

> attach(milsa)
```

Vejamos inicialmente duas variáveis categóricas (nominal e ordinal) e tabelas, gráficos e medidas descritivas.

```
> is.factor(civil)

[1] TRUE

> civil.tb <- table(civil)
> civil.tb

civil
solteiro   casado
      16       20

> 100 * table(civil)/length(civil)

civil
solteiro   casado
44.44444 55.55556

> prop.table(civil.tb)

civil
 solteiro    casado
0.4444444 0.5555556

> names(civil.tb)[which.max(civil.tb)]

[1] "casado"
```

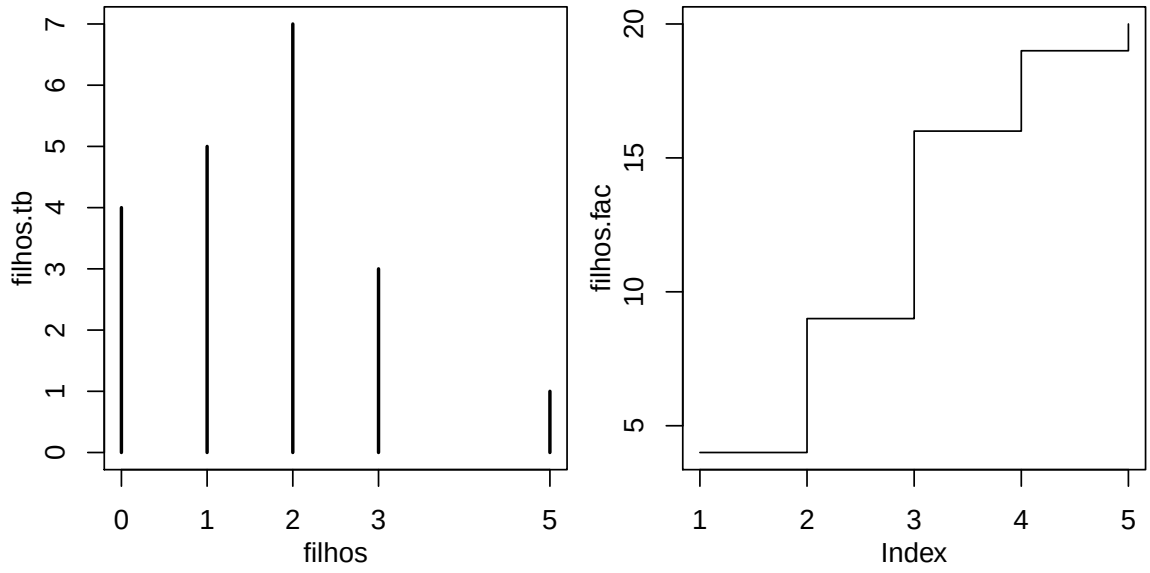
```
> is.factor(instrucao)

[1] TRUE

> instrucao.tb <- table(instrucao)
> instrucao.tb

instrucao
  1oGrau   2oGrau Superior
      12       18        6

> prop.table(instrucao.tb)

instrucao
   1oGrau    2oGrau  Superior
0.3333333 0.5000000 0.1666667

> names(instrucao.tb)[which.max(instrucao.tb)]

[1] "2oGrau"

> median(as.numeric(instrucao))

[1] 2

> levels(milsa$instrucao)[median(as.numeric(milsa$instrucao))]

[1] "2oGrau"


> par(mfrow = c(1, 2))
> pie(table(civil))
> barplot(instrucao.tb)
```



Agora variáveis numéricas.

```
> filhos.tb <- table(filhos)
> filhos.tb

filhos
0 1 2 3 5
4 5 7 3 1

> prop.table(table(filhos))

filhos
   0    1    2    3    5
0.20 0.25 0.35 0.15 0.05

> names(filhos.tb)[which.max(filhos.tb)]

[1] "2"

> median(filhos, na.rm = T)

[1] 2

> filhos.me <- mean(filhos, na.rm = T)
> filhos.me

[1] 1.65

> range(filhos, na.rm = T)

[1] 0 5

> diff(range(filhos, na.rm = T))

[1] 5

> filhos.dp <- sd(filhos, na.rm = T)
> filhos.dp

[1] 1.268028

> var(filhos, na.rm = T)

[1] 1.607895

> 100 * filhos.dp/filhos.me

[1] 76.85018

> filhos.qt <- quantile(filhos, na.rm = T)
> filhos.qt[4] - filhos.qt[2]

75%
  1

> summary(filhos)
```

```
> par(mfrow = c(1, 2))
> plot(filhos.tb)
> filhos.fac <- cumsum(filhos.tb)
> filhos.fac

 0  1  2  3  5
 4  9 16 19 20

> plot(filhos.fac, type = "s")
```



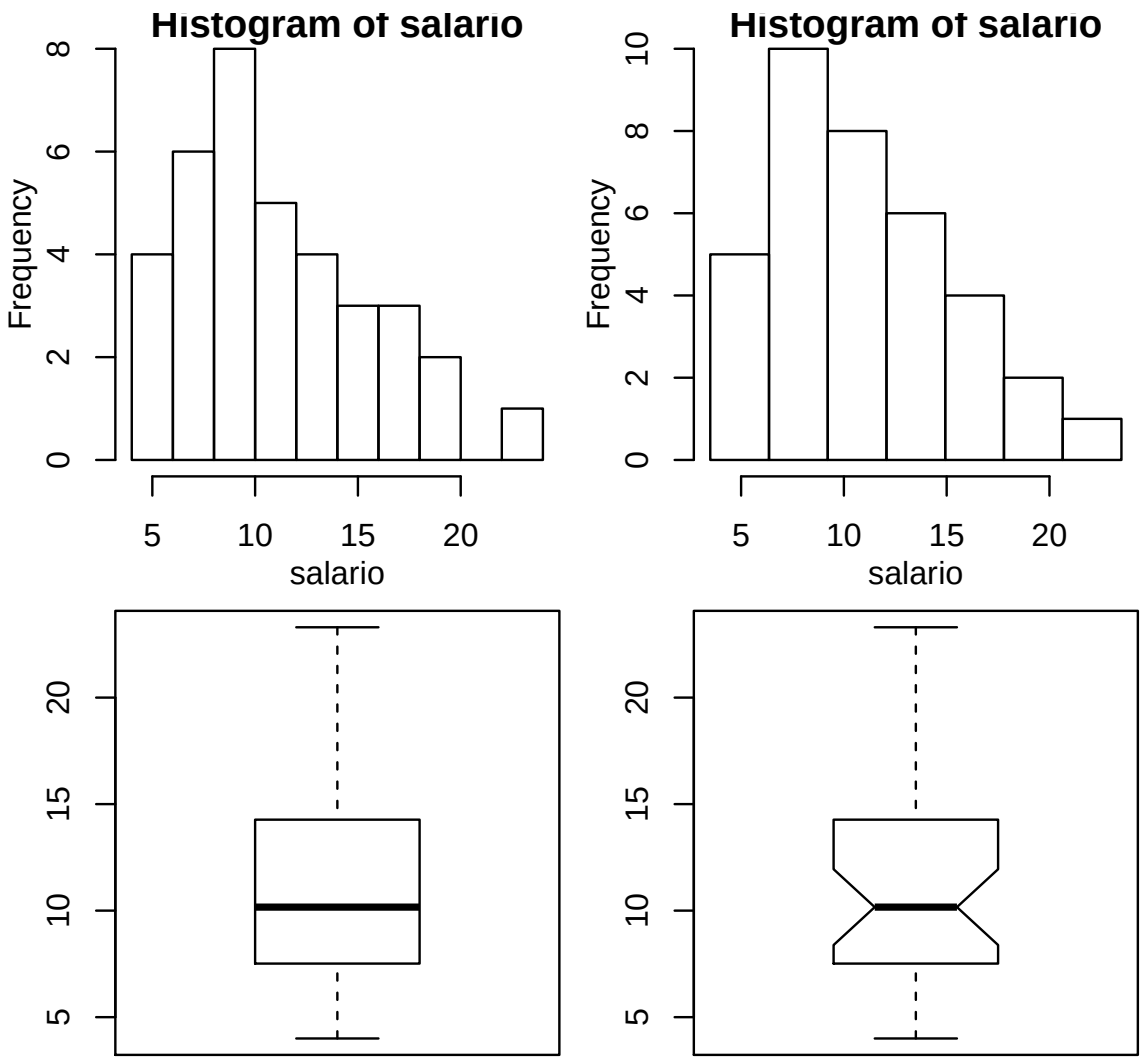

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
   0.00    1.00    2.00    1.65    2.00    5.00   16.00


> range(salario)

[1]  4.0 23.3

> nclass.Sturges(salario)

[1] 7

> table(cut(salario, seq(3.5, 23.5, l = 8)))

 (3.5,6.36] (6.36,9.21] (9.21,12.1] (12.1,14.9] (14.9,17.8] (17.8,20.6]
          5          10           8           6           4           2
(20.6,23.5]
          1

> stem(salario)

  The decimal point is at the |

   4 | 0637
   6 | 379446
```

```
   8 | 15791388
  10 | 5816
  12 | 08268
  14 | 77
  16 | 0263
  18 | 84
  20 |
  22 | 3
```

```
> median(salario, na.rm = T)

[1] 10.165

> mean(salario, na.rm = T)

[1] 11.12222

> range(salario, na.rm = T)

[1]  4.0 23.3

> diff(range(salario, na.rm = T))

[1] 19.3

> sd(salario, na.rm = T)

[1] 4.587458

> var(salario, na.rm = T)

[1] 21.04477

> salario.qt <- quantile(salario, na.rm = T)
> salario.qt

     0%     25%     50%     75%    100%
 4.0000  7.5525 10.1650 14.0600 23.3000

> salario.qt[4] - salario.qt[2]

   75%
6.5075

> summary(salario)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  4.000   7.552  10.160  11.120  14.060  23.300

> fivenum(salario)

[1]  4.000  7.515 10.165 14.270 23.300
```

Análises bivariadas

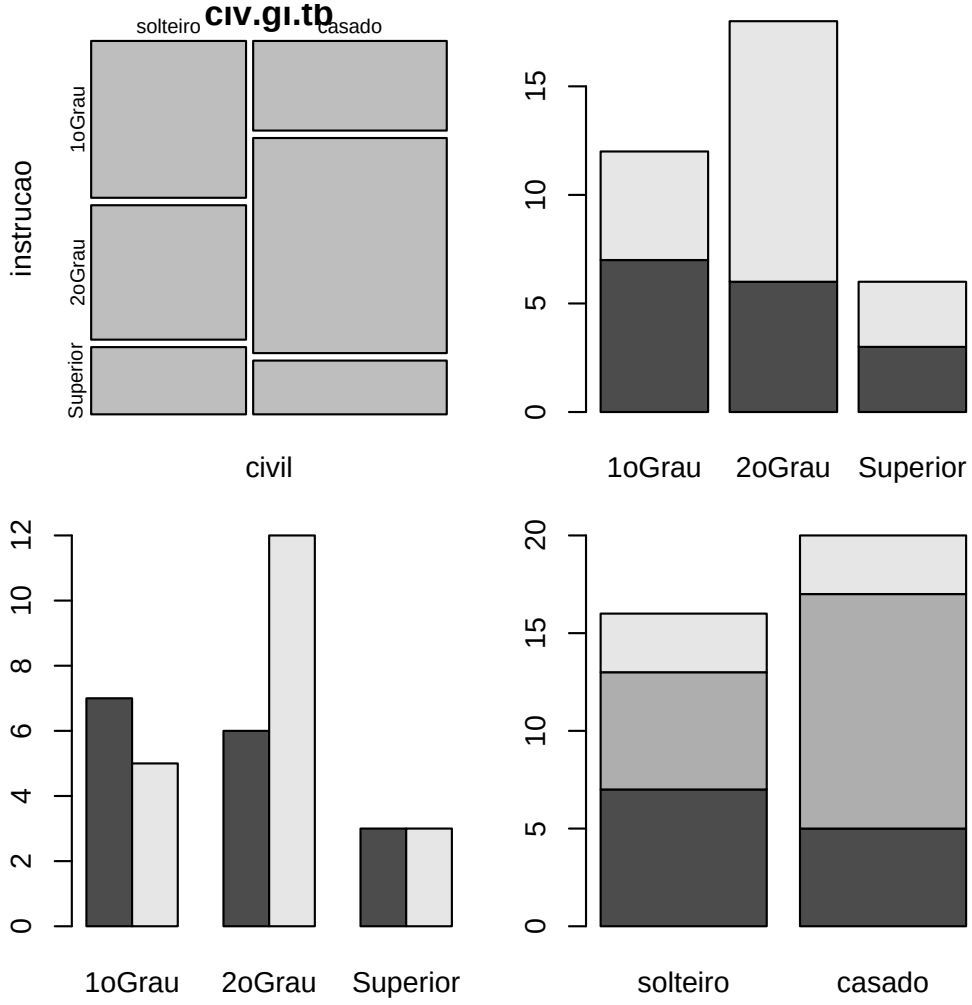
```
> par(mfrow = c(2, 2))
> hist(salario)
> hist(salario, br = seq(3.5, 23.5, 1 = 8))
> boxplot(salario)
> boxplot(salario, notch = T)
```




```
> civ.gi.tb <- table(civil, instrucao)
> civ.gi.tb

          instrucao
civil      1oGrau 2oGrau Superior
  solteiro      7      6        3
  casado        5     12        3

> civ.gi.tb/as.vector(table(civil))

          instrucao
civil      1oGrau 2oGrau Superior
  solteiro 0.4375 0.3750   0.1875
  casado   0.2500 0.6000   0.1500
```

```
> summary(civ.gi.tb)

Number of cases in table: 36
Number of factors: 2
Test for independence of all factors:
        Chisq = 1.9125, df = 2, p-value = 0.3843
        Chi-squared approximation may be incorrect

> instrucao1 <- ifelse(instrucao == "1oGrau", "1oGrau", "2 ou mais")
> table(instrucao)

instrucao
  1oGrau   2oGrau Superior
      12       18        6

> table(instrucao1)

instrucao1
   1oGrau 2 ou mais
       12        24

> table(civil, instrucao1)

          instrucao1
civil      1oGrau 2 ou mais
  solteiro      7         9
  casado        5        15

> summary(table(civil, instrucao1))

Number of cases in table: 36
Number of factors: 2
Test for independence of all factors:
        Chisq = 1.4062, df = 1, p-value = 0.2357


> quantile(salario)

     0%     25%     50%     75%    100%
 4.0000  7.5525 10.1650 14.0600 23.3000

> ins.sal.tb <- table(instrucao, cut(salario, quantile(salario)))
> ins.sal.tb

instrucao  (4,7.55] (7.55,10.2] (10.2,14.1] (14.1,23.3]
  1oGrau          6           3           2           0
  2oGrau          2           6           5           5
  Superior        0           0           2           4

> tapply(salario, instrucao, mean)

   1oGrau    2oGrau  Superior
 7.836667 11.528333 16.475000
```

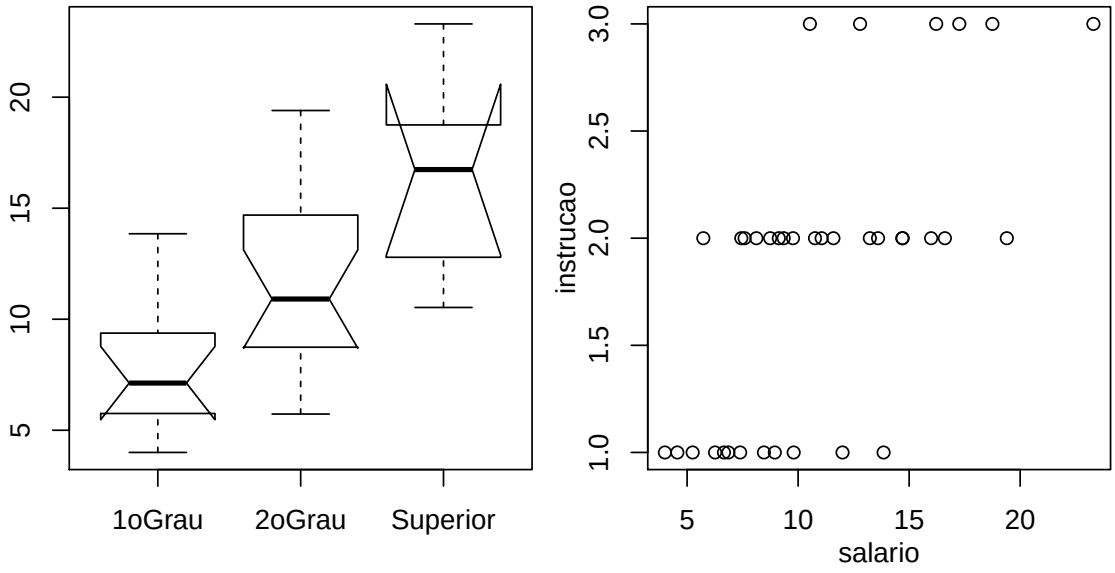
```
> par(mfrow = c(2, 2))
> plot(civ.gi.tb)
> barplot(civ.gi.tb)
> barplot(civ.gi.tb, bes = T)
> barplot(t(civ.gi.tb))
```




```
> tapply(salario, instrucao, var)

   1oGrau    2oGrau  Superior
 8.740679 13.802297 20.271950

> tapply(salario, instrucao, quantile)

$`1oGrau`
     0%     25%     50%     75%    100%
 4.0000  6.0075  7.1250  9.1625 13.8500

$`2oGrau`
     0%     25%     50%     75%    100%
 5.7300  8.8375 10.9100 14.4175 19.4000

$Superior
```

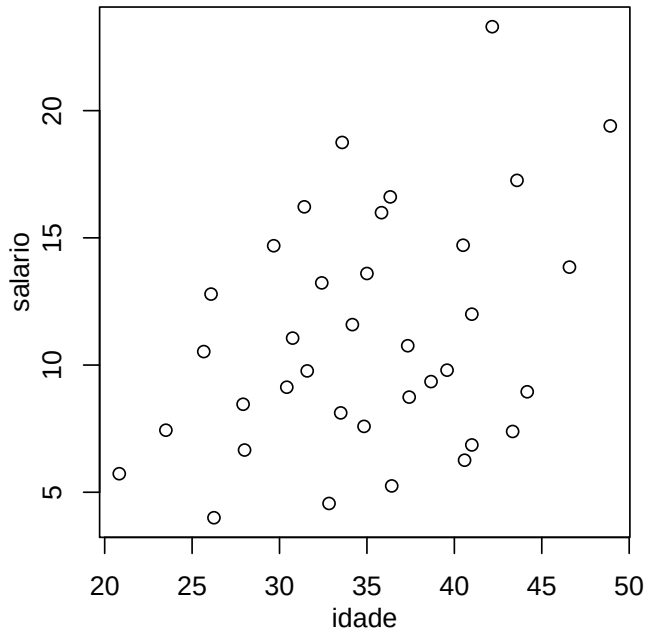
```
     0%      25%      50%      75%     100%
10.5300 13.6475 16.7400 18.3775 23.3000
```

```
> par(mfrow = c(1, 2))
> plot(instrucao, salario, notch = T)
> plot(salario, instrucao)
```




```
> table(cut(idade, quantile(idade)), cut(salario, quantile(salario)))

             (4,7.55] (7.55,10.2] (10.2,14.1] (14.1,23.3]
  (20.8,30.7]        2           2           2           1
  (30.7,34.9]        1           3           3           2
  (34.9,40.5]        1           3           2           3
  (40.5,48.9]        3           1           2           3

> table(cut(idade, quantile(idade, seq(0, 1, len = 4))), cut(salario,
+     quantile(salario, seq(0, 1, len = 4))))

             (4,8.65] (8.65,12.9] (12.9,23.3]
  (20.8,32.1]        3           5           2
  (32.1,37.8]        4           3           5
  (37.8,48.9]        3           4           5

> cor(idade, salario)

[1] 0.3651397

> cor(idade, salario, method = "spearman")

[1] 0.2895939


> detach(milsa)
```

```
> plot(idade, salario)
```



3 Um exemplo de regressão

Agora um exemplo de regressão linear simples, usual e ponderada. Vamos criar um novo vetor `x` com uma sequência de números de 1 a 20 e um vetor `w` de pesos com os desvios padrões de cada observação. Na sequencia montamos um *data-frame* de 2 colunas, x e y, e inspecionamos o objeto criado.

```
> x <- 1:20
> w <- 1 + sqrt(x)/2
> dummy <- data.frame(x = x, y = x + rnorm(x) * w)
> dummy
```

```
    x         y
1   1  2.124247
2   2  1.254262
3   3  5.029933
4   4  5.976538
5   5  1.037631
6   6  3.978979
7   7  5.227829
8   8  7.975761
9   9 16.934580
10 10 11.109367
11 11 10.685461
12 12  9.132284
13 13 14.735796
14 14 15.980588
15 15 16.094292
16 16 13.735926
17 17 22.218401
18 18 16.101075
```

```
19 19 15.489320
20 20 17.739231
```

Ajustando uma regressão linear simples de `y` em `x` e examinando os resultados.

```
> fm <- lm(y ~ x, data = dummy)
> summary(fm)

Call:
lm(formula = y ~ x, data = dummy)

Residuals:
    Min      1Q  Median      3Q     Max
-4.4219 -2.0733 -0.3577  1.4516  7.7161

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.7608     1.4025   0.542    0.594
x             0.9397     0.1171   8.027 2.34e-07 ***
---
Signif. codes:  0
```

Agora vamos utilizar os pesos para fazer uma regressão ponderada, e comparar os resultados.

```
> fm1 <- lm(y ~ x, data = dummy, weight = 1/w^2)
> summary(fm1)

Call:
lm(formula = y ~ x, data = dummy, weights = 1/w^2)

Residuals:
    Min      1Q  Median      3Q     Max
-2.0413 -0.7630 -0.1322  0.6337  3.0977

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.5748     1.0735   0.535    0.599
x             0.9573     0.1095   8.745 6.74e-08 ***
---
Signif. codes:  0
```

Entendendo o comando `attach` para anexar objetos, incluindo suas variáveis no caminho de procura.

```
> y

> x

 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20

> attach(dummy)
```

```
The following object(s) are masked _by_ '.GlobalEnv':

    x
```

```
> search()

 [1] ".GlobalEnv"        "dummy"             "package:stats"
 [4] "package:graphics"  "package:grDevices" "package:utils"
 [7] "package:datasets"  "package:methods"   "Autoloads"
[10] "package:base"
```

```
> x

 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
```

```
> y

 [1]  2.124247  1.254262  5.029933  5.976538  1.037631  3.978979  5.227829
 [8]  7.975761 16.934580 11.109367 10.685461  9.132284 14.735796 15.980588
[15] 16.094292 13.735926 22.218401 16.101075 15.489320 17.739231
```

```
> detach(dummy)
> search()

[1] ".GlobalEnv"        "package:stats"     "package:graphics"
[4] "package:grDevices" "package:utils"     "package:datasets"
[7] "package:methods"   "Autoloads"         "package:base"
```

```
> x

 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
```

```
> y
```

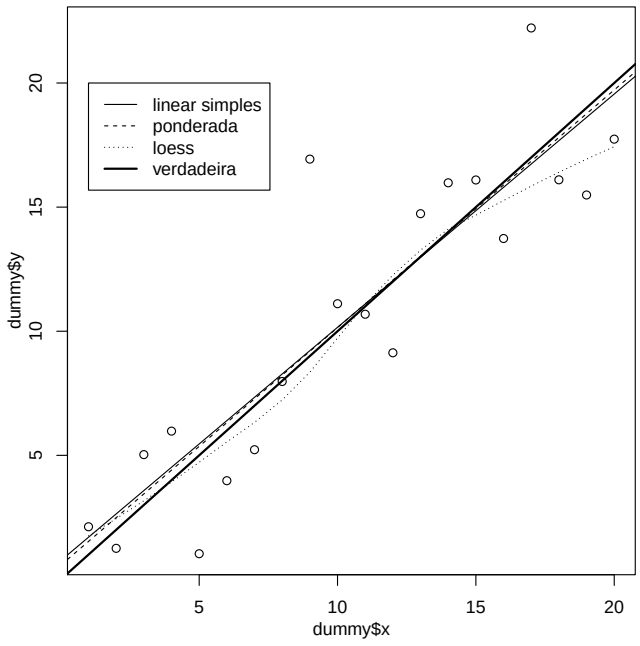
Perceba a ordem de prioridade de procura e o perigo de confundir objetos com nomes diferentes em ambientes diferentes! O `attach` é uma excelente ferramenta, mas pode fazer com que o usuário descuidade cometa erros sérios, confundindo objetos diferentes com mesmo nome.

Agora fazendo uma regressão local não-paramétrica.

```
> attach(dummy)

The following object(s) are masked _by_ '.GlobalEnv':

    x
```

```
> lrf <- lowess(x, y)
> lrf

$x
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20

$y
 [1]  1.758141  2.450245  3.167184  3.922882  4.721830  5.540565  6.327915
 [8]  7.230968  8.352182  9.724113 11.083325 12.261133 13.265009 14.115844
[15] 14.675916 15.264231 15.845831 16.404497 16.934200 17.436393
```

```
> plot(dummy$x, dummy$y)
> abline(fm$coef)
> abline(fm1$coef, lty = 2)
> lines(lrf, lty = 3)
> abline(0, 1, lwd = 2)
> legend(1, 20, c("linear simples", "ponderada", "loess", "verdadeira"),
+     lty = c(1, 2, 3, 1), lwd = c(1, 1, 1, 2))
```




```
> detach()
```

Plotando os pontos e as curvas das três regressões e a linha de regressão verdadeira (intercepto 0 e inclinação 1), e adicionando uma legenda. Ao final desanexa o objeto `dummy`.

Vamos agora entender melhor como o R guarda e exibe resultados. Para isto vamos examinar o objeto `fm` em mais detalhes. Primeiro vamos ver um resumo e um resumo expandido das análises.

```
> fm

Call:
lm(formula = y ~ x, data = dummy)

Coefficients:
(Intercept)            x
     0.7608       0.9397

> summary(fm)

Call:
lm(formula = y ~ x, data = dummy)

Residuals:
    Min      1Q  Median      3Q     Max
```

```
-4.4219 -2.0733 -0.3577  1.4516  7.7161

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.7608     1.4025   0.542    0.594
x             0.9397     0.1171   8.027 2.34e-07 ***
---
Signif. codes:  0
```

Mas há muito mais guardado . . . , e note que o resultado é um objeto da *classe* `lm` Cada elemento pode ser extraído usando o símbolo `$`

```
> class(fm)

[1] "lm"

> names(fm)

 [1] "coefficients"  "residuals"     "effects"       "rank"
 [5] "fitted.values" "assign"        "qr"            "df.residual"
 [9] "xlevels"       "call"          "terms"         "model"

> fm$coef

(Intercept)           x
  0.7608399   0.9397367
```

Para *classes* podem existir *métodos* associados. `print` e `summary` são dois deles. Outro comum é um método para `plot` que no caso de regressão produz quatro gráficos diagnósticos. Para mostrá-los todos de uma só vez dividimos a tela gráfica em quatro partes.

Mas o usuário pode definir e montar os gráficos que deseja. Por exemplo um gráfico diagnóstico padrão para checar homocedasticidade e um gráficos de escores normais para checar assimetria, curtose e 'outliers' personalizados pelo usuário.

Apagando o que criamos até aqui e ao final voltando para a tela única.

```
> rm(fm, fm1, lrf, x, dummy)
> par(mfrow = c(1, 1))
```

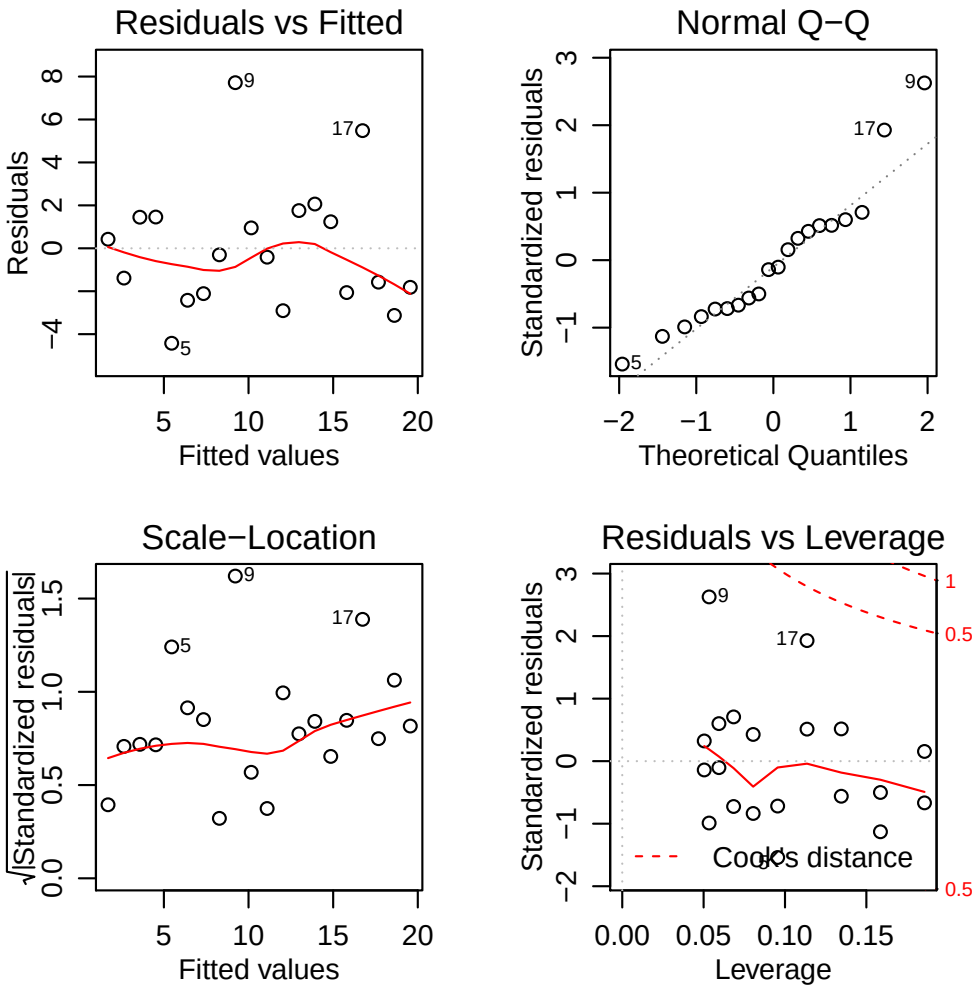
4 Uma análise de experimento

Agora vamos inspecionar dados do experimento clássico de Michaelson e Morley para medir a velocidade da luz. Os dados estão disponíveis no arquivo de dados em formato texto `morley.tab`. Grave este arquivo no seu diretório de trabalho e para inspecionar o arquivo de dentro do R voce pode usar:

```
> file.show("dados/morley.tab")
```

Vamos importar os dados para dentro do R usado a função `read.table` e inspecionar o objeto que contém os dados, um *data-frame*. Há 5 experimentos (coluna `Expt`) e cada um com 20 "rodadas" (coluna `Run`) e `Speed` é o valor medido da velocidade da luz numa escala apropriada.

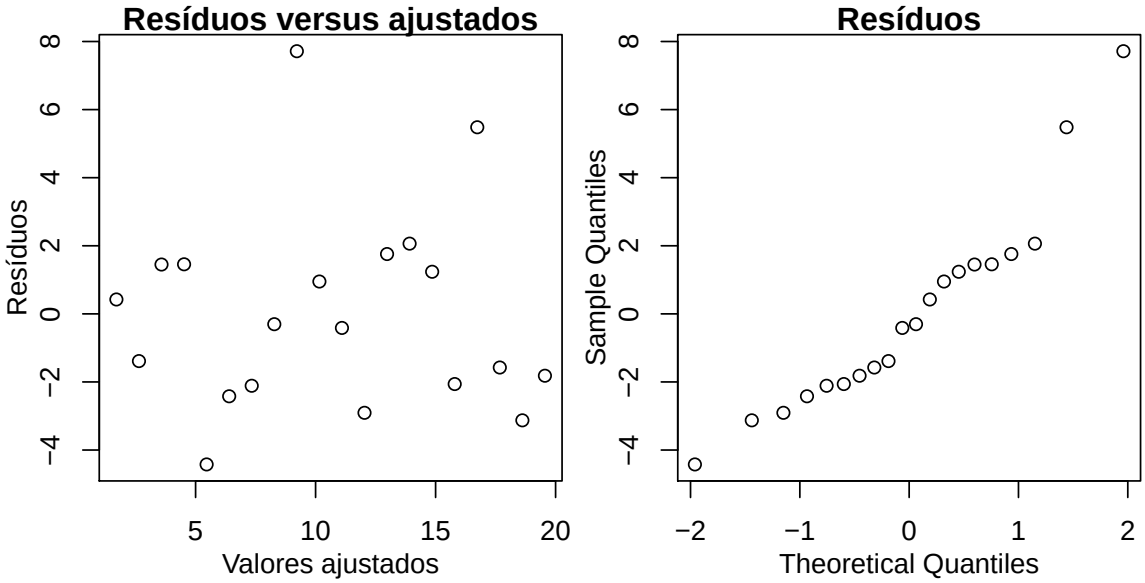
```
> par(mfrow = c(2, 2))
> plot(fm)
```




```
> mm <- read.table("dados/morley.tab")
> mm
```

	Expt	Run	Speed
001	1	1	850
002	1	2	740
003	1	3	900
004	1	4	1070
005	1	5	930
006	1	6	850
007	1	7	950
008	1	8	980
009	1	9	980
010	1	10	880
011	1	11	1000
012	1	12	980
013	1	13	930
014	1	14	650
015	1	15	760

```
> par(mfrow = c(1, 2))
> plot(fitted(fm), resid(fm), xlab = "Valores ajustados", ylab = "Resíduos",
+     main = "Resíduos versus ajustados")
> qqnorm(resid(fm), main = "Resíduos")
```




```
016     1   16   810
017     1   17  1000
018     1   18  1000
019     1   19   960
020     1   20   960
021     2    1   960
022     2    2   940
023     2    3   960
024     2    4   940
025     2    5   880
026     2    6   800
027     2    7   850
028     2    8   880
029     2    9   900
030     2   10   840
031     2   11   830
032     2   12   790
033     2   13   810
034     2   14   880
035     2   15   880
036     2   16   830
037     2   17   800
038     2   18   790
039     2   19   760
040     2   20   800
041     3    1   880
042     3    2   880
043     3    3   880
```

```
044     3    4   860
045     3    5   720
046     3    6   720
047     3    7   620
048     3    8   860
049     3    9   970
050     3   10   950
051     3   11   880
052     3   12   910
053     3   13   850
054     3   14   870
055     3   15   840
056     3   16   840
057     3   17   850
058     3   18   840
059     3   19   840
060     3   20   840
061     4    1   890
062     4    2   810
063     4    3   810
064     4    4   820
065     4    5   800
066     4    6   770
067     4    7   760
068     4    8   740
069     4    9   750
070     4   10   760
071     4   11   910
072     4   12   920
073     4   13   890
074     4   14   860
075     4   15   880
076     4   16   720
077     4   17   840
078     4   18   850
079     4   19   850
080     4   20   780
081     5    1   890
082     5    2   840
083     5    3   780
084     5    4   810
085     5    5   760
086     5    6   810
087     5    7   790
088     5    8   810
089     5    9   820
090     5   10   850
091     5   11   870
092     5   12   870
093     5   13   810
```

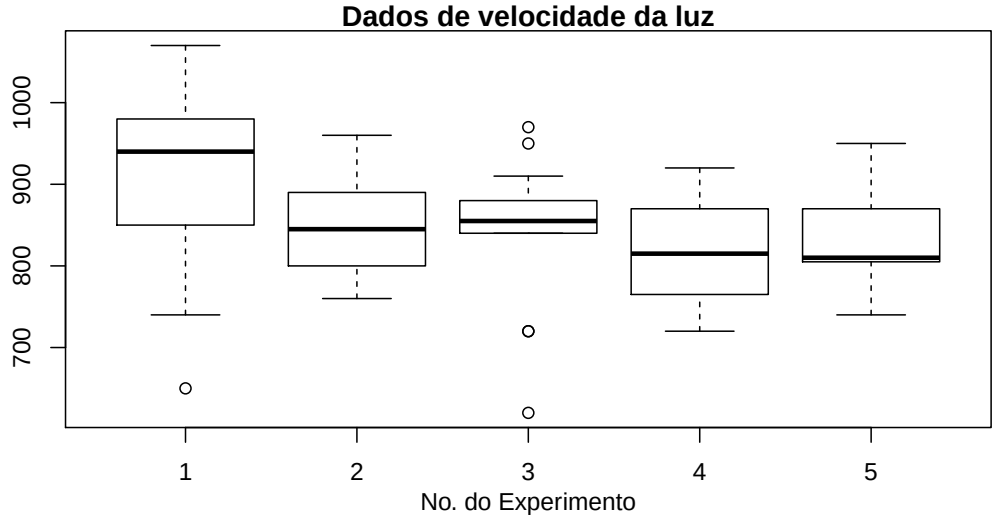
```
094     5  14    740
095     5  15    810
096     5  16    940
097     5  17    950
098     5  18    800
099     5  19    810
100     5  20    870
```

Na análise que faremos as variáveis `Expt` e `Run` não serão tratadas como numéricas, mas sim como categóricas. Para isto temos que redefini-las como fatores.

```
> mm$Expt <- factor(mm$Expt)
> mm$Run <- factor(mm$Run)
```

Agora vamos anexar o objeto que contém os dados e comparar dados dos 5 experimentos.

```
> attach(mm)
> plot(Expt, Speed, main = "Dados de velocidade da luz", xlab = "No. do Experimento")
```



Vamos fazer uma análise de variância segundo o modelo para experimentos em blocos ao acaso com "runs" and "experiments" como fatores e inspecionar os resultados.

```
> fm <- aov(Speed ~ Run + Expt, data = mm)
> fm

Call:
   aov(formula = Speed ~ Run + Expt, data = mm)

Terms:
                    Run   Expt Residuals
Sum of Squares   113344  94514    410166
Deg. of Freedom      19      4        76

Residual standard error: 73.46374
Estimated effects may be unbalanced

> summary(fm)
```

```
            Df Sum Sq Mean Sq F value   Pr(>F)
Run         19 113344  5965.5  1.1053 0.363209
Expt         4  94514 23628.5  4.3781 0.003071 **
Residuals   76 410166  5396.9
---
Signif. codes:  0
```

```
> anova(fm)
```

```
Analysis of Variance Table

Response: Speed
          Df Sum Sq Mean Sq F value   Pr(>F)
Run       19 113344  5965.5  1.1053 0.363209
Expt       4  94514 23628.5  4.3781 0.003071 **
Residuals 76 410166  5396.9
---
Signif. codes:  0
```

```
> names(fm)
```

```
 [1] "coefficients"  "residuals"     "effects"       "rank"
 [5] "fitted.values" "assign"        "qr"            "df.residual"
 [9] "contrasts"     "xlevels"       "call"          "terms"
[13] "model"
```

```
> fm$coef
```

```
  (Intercept)          Run2          Run3          Run4          Run5
 9.506000e+02 -5.200000e+01 -2.800000e+01  6.000000e+00 -7.600000e+01
         Run6          Run7          Run8          Run9         Run10
-1.040000e+02 -1.000000e+02 -4.000000e+01 -1.000000e+01 -3.800000e+01
        Run11         Run12         Run13         Run14         Run15
 4.000000e+00 -1.754382e-13 -3.600000e+01 -9.400000e+01 -6.000000e+01
        Run16         Run17         Run18         Run19         Run20
-6.600000e+01 -6.000000e+00 -3.800000e+01 -5.000000e+01 -4.400000e+01
        Expt2         Expt3         Expt4         Expt5
-5.300000e+01 -6.400000e+01 -8.850000e+01 -7.750000e+01
```

```
> model.tables(fm)
```

```
Tables of effects

 Run
Run
    1     2     3     4     5     6     7     8     9    10    11    12    13
 41.6 -10.4  13.6  47.6 -34.4 -62.4 -58.4   1.6  31.6   3.6  45.6  41.6   5.6
   14    15    16    17    18    19    20
-52.4 -18.4 -24.4  35.6   3.6  -8.4  -2.4

 Expt
Expt
    1     2     3     4     5
 56.6   3.6  -7.4 -31.9 -20.9
```

```
> model.tables(fm, type = "means")

Tables of means
Grand mean

852.4

 Run
Run
  1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
894 842 866 900 818 790 794 854 884 856 898 894 858 800 834 828 888 856 844 850

 Expt
Expt
    1     2     3     4     5
909.0 856.0 845.0 820.5 831.5
```

Agora, somente para ilustração de comparação de modelos, ajustamos um sub-modelo sem o fator "runs" e comparamos os dois modelos via análise de variância.

```
> fm0 <- update(fm, . ~ . - Run)
> anova(fm0, fm)

Analysis of Variance Table

Model 1: Speed ~ Expt
Model 2: Speed ~ Run + Expt
  Res.Df    RSS Df Sum of Sq      F Pr(>F)
1     95 523510
2     76 410166 19    113344 1.1053 0.3632
```

É importante saber interpretar os coeficientes segunda a parametrização utilizada. Por *default* a parametrização é feita tomando o primeiro grupo como referência.

```
> fm0$coef

(Intercept)       Expt2       Expt3       Expt4       Expt5
      909.0       -53.0       -64.0       -88.5       -77.5

> mds <- tapply(Speed, Expt, mean)
> mds

    1     2     3     4     5
909.0 856.0 845.0 820.5 831.5

> mds[-1] - mds[1]

    2     3     4     5
-53.0 -64.0 -88.5 -77.5
```

E este comportamento é controlado por `options()`. Por exemplo, contrastes de Helmert são definidos como se segue.

```
> options()$contrast

        unordered           ordered
"contr.treatment"      "contr.poly"

> options(contrasts = c("contr.helmert", "contr.poly"))
> fm0 <- update(fm, . ~ . - Run)
> fm0$coef

(Intercept)       Expt1       Expt2       Expt3       Expt4
    852.400     -26.500     -12.500     -12.375      -5.225

> mean(Speed)

[1] 852.4

> (mds[2] - mds[1])/2

    2
-26.5

> (2 * mds[3] - mds[1] - mds[2])/6

    3
-12.5

> (3 * mds[4] - mds[1] - mds[2] - mds[3])/12

      4
-12.375

> (4 * mds[5] - mds[1] - mds[2] - mds[3] - mds[4])/20

     5
-5.225
```

Enquanto que contrastes de cada tratamento contra a média geral são obtidos da forma:

```
> options(contrasts = c("contr.sum", "contr.poly"))
> fm0 <- update(fm, . ~ . - Run)
> fm0$coef

(Intercept)       Expt1       Expt2       Expt3       Expt4
      852.4        56.6         3.6        -7.4       -31.9

> mds - mean(Speed)

    1     2     3     4     5
 56.6   3.6  -7.4 -31.9 -20.9
```

Há algumas opções de contrastes implementadas no R e além disto o usuário pode implementar contrastes de sua preferência. Para entender melhor os resultados acima analise as saídas dos comandos abaixo.

```
> contr.treatment(5)

  2 3 4 5
1 0 0 0 0
2 1 0 0 0
3 0 1 0 0
4 0 0 1 0
5 0 0 0 1

> contr.helmert(5)

  [,1] [,2] [,3] [,4]
1   -1   -1   -1   -1
2    1   -1   -1   -1
3    0    2   -1   -1
4    0    0    3   -1
5    0    0    0    4

> contr.sum(5)

  [,1] [,2] [,3] [,4]
1    1    0    0    0
2    0    1    0    0
3    0    0    1    0
4    0    0    0    1
5   -1   -1   -1   -1

> contr.poly(5)

                .L         .Q            .C         ^4
[1,] -6.324555e-01  0.5345225 -3.162278e-01  0.1195229
[2,] -3.162278e-01 -0.2672612  6.324555e-01 -0.4780914
[3,] -3.287978e-17 -0.5345225  2.164914e-16  0.7171372
[4,]  3.162278e-01 -0.2672612 -6.324555e-01 -0.4780914
[5,]  6.324555e-01  0.5345225  3.162278e-01  0.1195229
```

Se ainda não estivar claro experimente para cada uma destas examinar a matrix do modelo com os comandos abaixo (saídas não mostradas aqui)

```
> options(contrasts = c("contr.treatment", "contr.poly"))
> model.matrix(Speed ~ Expt)
> options(contrasts = c("contr.helmert", "contr.poly"))
> model.matrix(Speed ~ Expt)
> options(contrasts = c("contr.sum", "contr.poly"))
> model.matrix(Speed ~ Expt)
```

Ao final desanexamos o objeto e limpamos novamente a área de trabalho.

```
> detach()
> rm(fm, fm0)
```

5 Gráficos com `contour` e `image`

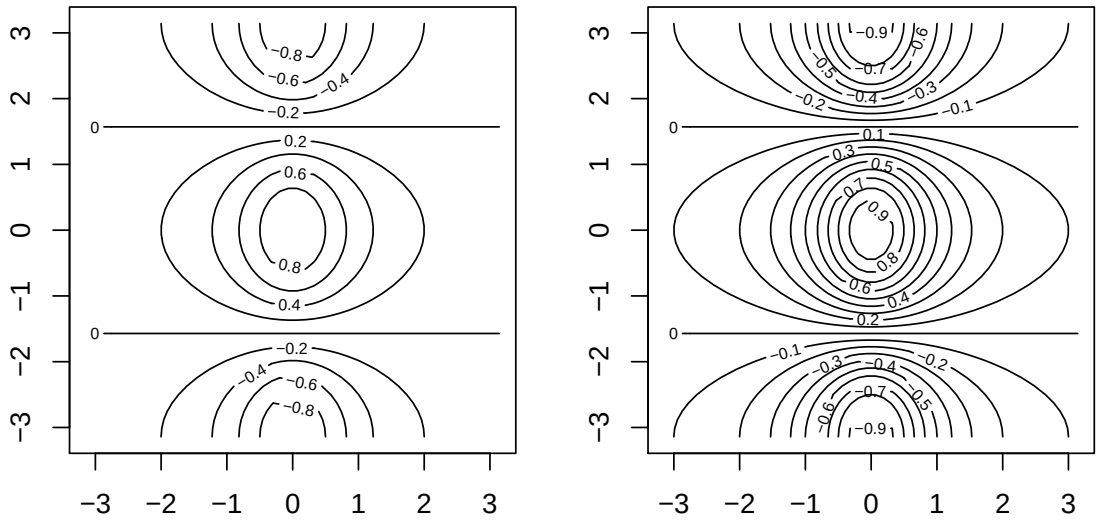
Vamos mostrar aqui gráficos tri-dimensionais usando como exemplo uma função de duas variáveis.

Vamos definir `x` como um vetor de 50 valores igualmente espaçados no intervalo $[-\pi\,\pi]$ e idem para outro vetor `y`. Depois definimos `f` como uma matrix quadrada com linhas e colunas indexadas por x e y respectivamente com os valores da função $f(x, y) = cos(y)/(1 + x^2)$.

```
> x <- seq(-pi, pi, len = 50)
> y <- x
> f <- outer(x, y, function(x, y) cos(y)/(1 + x^2))
```

Para garantir a escala apropriada no gráfico temos que definir uma região quadrada, o que não é *default* do programa. Antes disto gravamos parâmetros gráficos atuais. Podemos um mapa de contorno de `f` e se quisermos adicionamos mais linhas para maiores detalhes. Agora

```
> par(pty = "s", mfrow = c(1, 2))
> contour(x, y, f)
> contour(x, y, f, nlevels = 15)
```



com `fa` que é a "parte assimétrica" (`t()` é transposição) e fazendo um mapa de contorno.

```
> fa <- (f - t(f))/2
> contour(x, y, fa, nlevels = 15)
```

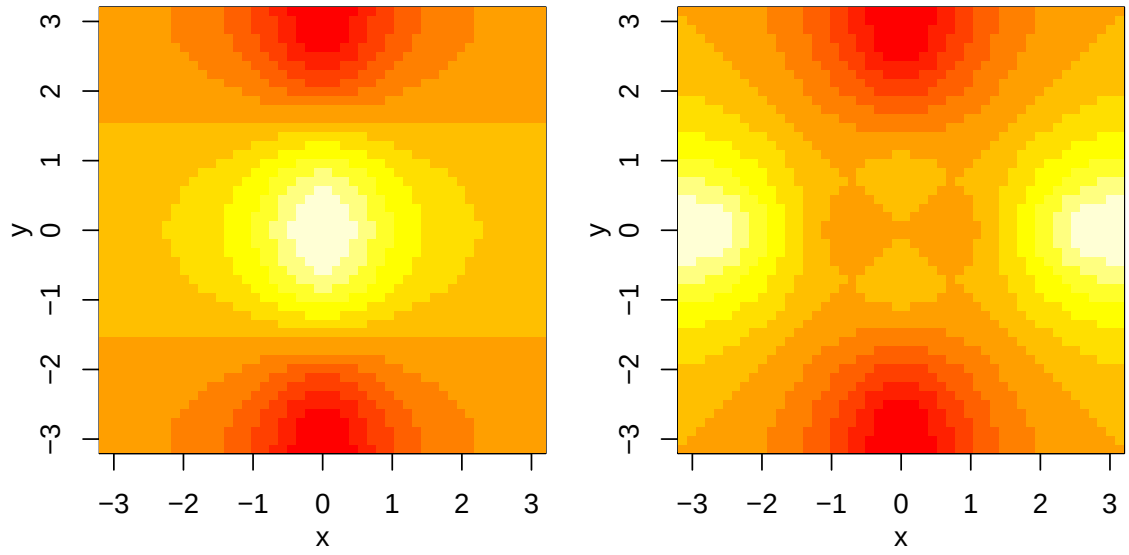
Fazendo um gráfico de imagem E apagando objetos novamente antes de prosseguir.

```
> objects()
```

```
 [1] "civ.gi.tb"    "civil.tb"     "f"            "fa"           "filhos.dp"
 [6] "filhos.fac"   "filhos.me"    "filhos.qt"    "filhos.tb"    "ins.sal.tb"
[11] "instrucao.tb" "instrucao1"   "mds"          "milsa"        "mm"
[16] "salario.qt"   "w"            "x"            "y"
```

```
> rm(x, y, f, fa)
```

```
> par(pty = "s", mfrow = c(1, 2))
> image(x, y, f)
> image(x, y, fa)
```



6 Mais algumas funcionalidades

Para encerrar esta sessão vejamos mais algumas funcionalidades do R. O R pode fazer operação com complexos (1i denota o número complexo i). Plotando complexos significa parte imaginária versus real e isto deve ser um círculo. Suponha que desejamos amostrar pontos dentro do círculo de raio unitário. uma forma simples de fazer isto é tomar números complexos com parte real e imaginária padrão e para mapear qualquer externo ao círculo no seu recíproco definimos `w` como se segue. Com isto todos os pontos estão dentro do círculo unitário, mas a distribuição não é uniforme. Depois um segundo método que usa a distribuição uniforme. Os pontos devem estar melhor distribuídos sobre o círculo

Apagando os objetos ...

```
> rm(th, w, z)
```

```
> th <- seq(-pi, pi, len = 100)
> z <- exp((0+1i) * th)
> par(pty = "s", mfrow = c(1, 2))
> w <- rnorm(100) + rnorm(100) * (0+1i)
> w <- ifelse(Mod(w) > 1, 1/w, w)
> plot(w, xlim = c(-1, 1), ylim = c(-1, 1), pch = "+", xlab = "x",
+     ylab = "y")
> lines(z)
> w <- sqrt(runif(100)) * exp(2 * pi * runif(100) * (0+1i))
> plot(w, xlim = c(-1, 1), ylim = c(-1, 1), pch = "+", xlab = "x",
+     ylab = "y")
> lines(z)
```