
Linguagem R
Prof. Dr. Razer Anthom Nizer Rojas Montaño
UFPR
SEPT
SETOR DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
UNIVERSIDADE FEDERAL DO PARANÁ

1


7 Estatística Básica
Funções Estatísticas Básicas
Gráficos
Regressão Linear
Prof. Dr. Razer A N R Montaño
SEPT / UFPR
2

2

Funções Estatísticas Básicas

- Cria vetor com valores aleatórios em distribuição normal: `rnorm(x)`
- Média: `mean(x)`
- Mediana: Valor do meio, que separa as metades (ou a média dos dois centrais): `median(x)`
- Média Ponderada: `weighted.mean(x, y)`
  - y é um vetor com os pesos
- Desvio Padrão: grau de dispersão dos valores: `sd(x)`
- Valor mínimo: `min(x)`
- Valor máximo: `max(x)`
- Quartil: Valor que deixa 25% da população menor, mediana e 75% maior: `quantile(x)`
- Estatísticas Básicas: `summary(x)`

3

Funções Estatísticas Básicas

```
> lista <- rnorm(10)
> lista
 [1] -0.7864034 -0.4225279  2.2372223 -0.5731214  1.2412395  0.4449866
 [7] -0.4742269  1.5658368 -0.7191717  0.4389552
> mean(lista)
[1] 0.2952789
> median(lista)
[1] 0.008213653
> weighted.mean(lista, c(1:10))
[1] 0.3415991
> sd(lista)
[1] 1.074141
> min(lista)
[1] -0.7864034
> max(lista)
[1] 2.237222
> quantile(lista)
          0%          25%          50%          75%         100%
-0.786403400 -0.548397732  0.008213653  1.042176265  2.237222275
> summary(lista)
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-0.786403 -0.548398  0.008214  0.295279  1.042176  2.237222
```


4

Funções Estatísticas Básicas

- **Amplitude**:
  - Medida de dispersão
  - Só usa 2 valores para o cálculo
  - Ruim com *Outliers*
  - Diferença entre o menor e o maior valor de uma amostra :
  - Usa-se `range()` e `diff()`

```
> x <- c(22, 7, 19, 8, 9, 19, 10)
> range(x)
[1]  7 22
> diff(range(x))
[1] 15
```


5

Funções Estatísticas Básicas

- **Desvio Padrão**:
  - Medida de dispersão ao redor da média dos valores
  - Um valor pequeno, significa amplitude pequena ao redor da média, pouco dispersos
  - Um valor grande, significa amplitude grande ao redor da média, muito dispersos
  - Usa-se `sd()`

```
> x <- c(22, 7, 19, 8, 9, 19, 10)
> sd(x)
[1] 6.294366
```


6

Funções Estatísticas Básicas.

- **Coeficiente de Variação**:
  - Desvio Padrão em porcentagem da média
  - Muito afetado pela escala dos valores
  - Comparar variáveis semelhantes
  - Usa-se

```
> x <- c(22, 7, 19, 8, 9, 19, 10)
 > sd(x)/mean(x)*100
[1] 46.87294
```


7

Tabela de Frequências

- Informa a frequência de um dado categórico no conjunto de dados
- Exemplo:

Tipo de Cliente	Frequência
Inicial	543
Normal	1.078
Silver	930
Gold	3.529
Platinum	322
TOTAL	**6.402**

8

Tabela de Frequências

- Exemplo com a Frequência Relativa e Porcentagem:
- $Frequência\ Relativa = \frac{Frequência\ da\ Categoria}{Soma\ total\ das\ Frequências}$
- $Frequência\ \% = \frac{Frequência\ da\ Categoria}{Soma\ total\ das\ Frequências} * 100$

Tipo de Cliente	Frequência	Fr. Relativa	Fr. %
Inicial	543	0.08481724	8.481724
Normal	1.078	0.16838488	16.838488
Silver	930	0.14526710	14.526710
Gold	3.529	0.55123399	55.123399
Platinum	322	0.05029678	5.029678
TOTAL	**6.402**	**1.0**	**100%**

9

Tabela de Frequências

- Dada uma massa de dados, usa-se `prop.table()` para calcular as frequências relativas e porcentagens
- Exemplo

```
> x <- c(543, 1078, 930, 3529, 322)
> prop.table(x)
[1] 0.08481724 0.16838488 0.14526710 0.55123399 0.05029678
> prop.table(x)*100
[1]  8.481724 16.838488 14.526710 55.123399  5.029678
```

Tipo de Cliente	Frequência	Fr. Relativa	Fr. %
Inicial	543	0.08481724	8.481724
Normal	1.078	0.16838488	16.838488
Silver	930	0.14526710	14.526710
Gold	3.529	0.55123399	55.123399
Platinum	322	0.05029678	5.029678
TOTAL	**6.402**	**1.0**	**100%**

10

Tabela de Frequências

- Criar uma tabela de Frequências: `table()`
- Seja a seguinte massa de dados

```
> grupoA <- rep("Grupo A", sample(1:100, 1))
> grupoB <- rep("Grupo B", sample(1:100, 1))
> grupoC <- rep("Grupo C", sample(1:100, 1))
> grupos <- sample(c(grupoA, grupoB, grupoC))
> grupoC
.... A cada execução mudam as quantidades ....
> table(grupos)
grupos
Grupo A Grupo B Grupo C
      8      24      54
```


11

Tabela de Frequências

- Seja a base de dados IRIS, que contém dados de várias espécies de uma flor

```
> head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
```


12

Tabela de Frequências

- Para descobrir a frequência das espécies nesta base

```
> table(iris$Species)

    setosa versicolor  virginica
        50         50         50
```

- Para descobrir a frequência de dados conforme uma condição

```
> table(iris$Sepal.Length>5.0)

   FALSE   TRUE
      32    118
```


13

Tabela de Frequências

- Classes de Frequências
  - Para dados quantitativos (numéricos)
  - Com uma base de dados, consegue-se separá-la em categorias/classes
  - Deseja-se a tabela de frequências POR CLASSE
- Seja a base de observações abaixo

```
> dados <- c(38, 15, 43, 85, 36, 15, 96, 35, 20, 29, 76,
39, 18, 14, 37, 39, 68, 63, 96, 86, 45, 89, 94, 60, 73, 60, 59,
73, 52, 32)
> summary(dados)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  14.00   35.25   48.50   52.83   73.00   96.00
```


14

Tabela de Frequências.

- Precisa-se definir classes e calcular a frequência nestas classes
  - Ex: 0-24, 25-49, 50-74, 75-100

```
> interv <- seq(0, 100, 25)
> interv
[1]   0  25  50  75 100
```

- Gera-se o nome das classes

```
> classes <- c("0-24", "25-49", "50-74", "75-100")
```

- Então geram-se as frequências em cada classe
  - Usa-se `cut()`: divide o vetor nas diversas faixas de valores
  - Usa-se `table()`: gera a tabela pela contagem em cada faixa gerada

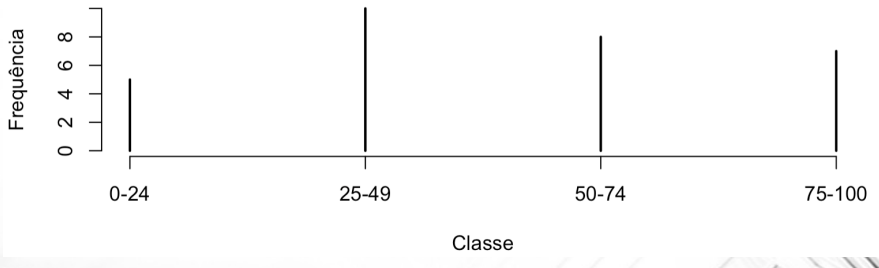
```
> table(cut(dados, breaks=interv, right=FALSE,
include.lowest=T, labels=classes))
0-24  25-49  50-74 75-100
   5     10      8      7
```


15

Gráficos: Tabela de Frequências.

- Seja o exemplo anterior, usa-se `plot()` para plotar o gráfico da tabela de frequências

```
> dados <- c(38, 15, 43, 85, 36, 15, 96, 35, 20, 29, 76, 39, 18,
14, 37, 39, 68, 63, 96, 86, 45, 89, 94, 60, 73, 60, 59, 73, 52, 32)
> interv <- seq(0,100,25)
> classes <- c("0-24", "25-49", "50-74", "75-100")
> t <- table(cut(dados, breaks=interv, right=FALSE,
labels=classes))
> plot(t, xlab="Classe", ylab="Frequência")
```



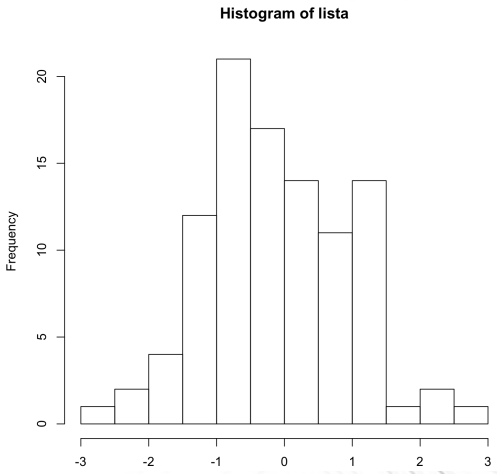
16

Gráficos: Histograma

- Histograma
  - Gráfico de frequência
  - Função `hist()`

- Possui várias configurações, como rótulos, títulos, etc

```
> lista <- rnorm(100)
> hist(lista)
```

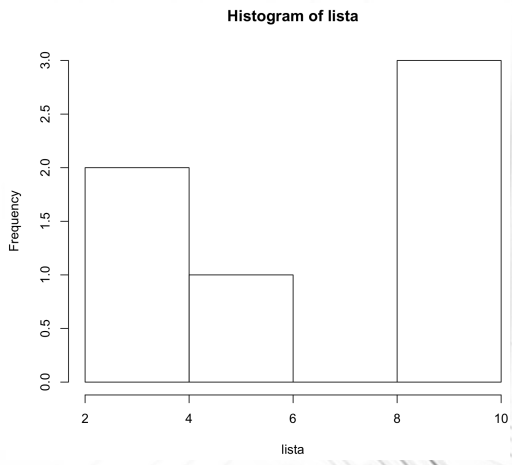


17

Gráficos: Histograma

- Exemplo

```
> lista <- c(10, 10, 10, 5, 2, 2)
> hist(lista)
```

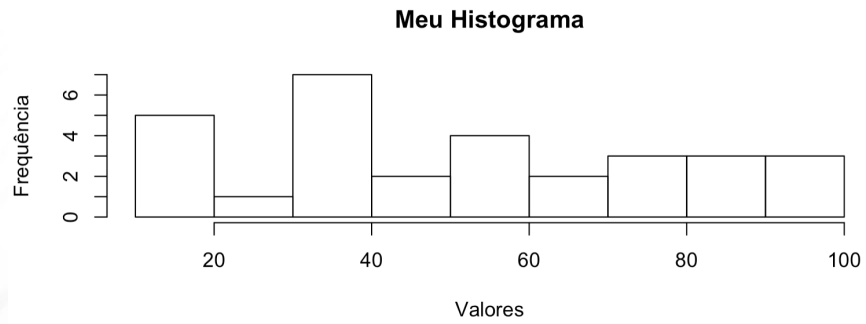


18

Gráficos: Histograma.

- Exemplo

```
> dados <- c(38, 15, 43, 85, 36, 15, 96, 35, 20, 29, 76, 39,
18, 14, 37, 39, 68, 63, 96, 86, 45, 89, 94, 60, 73, 60, 59, 73,
52, 32)
> hist(dados, xlab="Valores", ylab="Frequência", main="Meu
Histograma")
```



19

Gráficos: Histograma

- Seja a base de dados Cars93, do pacote MASS

```
> library("MASS")
> str(Cars93)
```

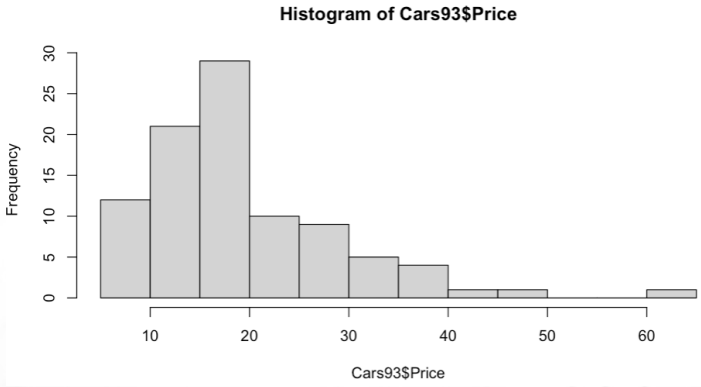
- É formada por:
  - 93 observações
  - 27 atributos
  - Informações de carros disponíveis em 1993
- Dados como:
  - Tipo, Fabricante, Preço, RPM, Peso, etc

20

Gráficos: Histograma

- Seja um histograma de preços

```
> library("MASS")
> str(Cars93)
> hist(Cars93$Price)
```

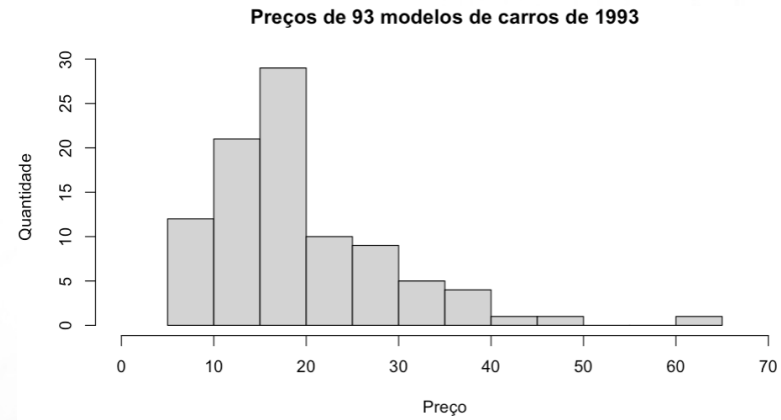


21

Gráficos: Histograma.

- Melhorando o histograma

```
> hist(Cars93$Price, xlab="Preço x $1000", ylab="Quantidade",
xlim=c(0,70), main="Preços de 93 modelos de carros em 1993")
```

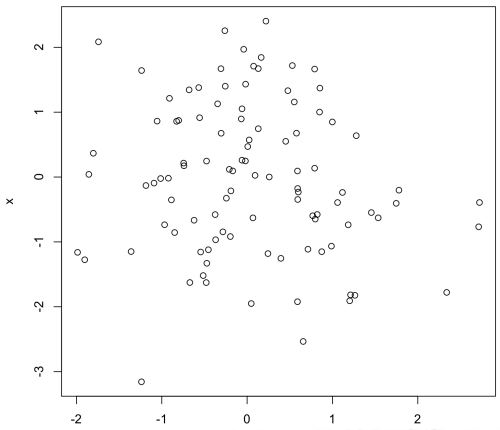


22

Gráficos: Gráfico de Dispersão

- Função `plot(x, y)`
- Cria um gráfico de pontos
  - Relação entre os dados
- Possui várias configurações

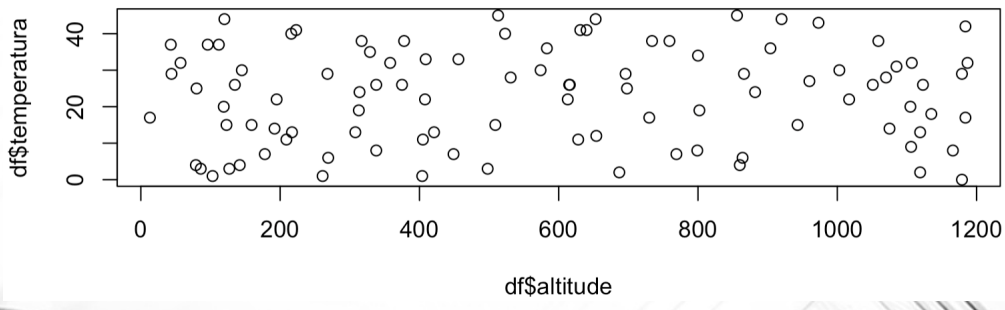
```
> x <- rnorm(100)
> y <- rnorm(100)
> plot(x, y)
```



Gráficos: Gráfico de Dispersão

- Seja a seguinte massa de dados: temperaturas x alturas

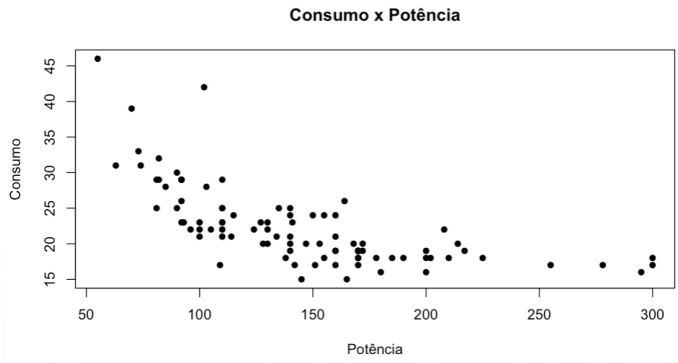
```
> n_dados <- 100
> v_temp <- sample(0:45, n_dados, replace=T)
> v_altitude <- sample(0:1200, n_dados, replace=T)
> df <- data.frame(temperatura=v_temp,
altitude=v_altitude)
> plot(df$altitude, df$temperatura)
```



Gráficos: Gráfico de Dispersão

- Seja a base Cars93
- Queremos um gráfico que mostre a relação entre a potência do carro e o consumo na cidade (MPG – miles per gallon)

```
> plot(Cars93$Horsepower, Cars93$MPG.city,
xlab="Potência", ylab="Consumo", main="Consumo x Potência",
pch=16)
```

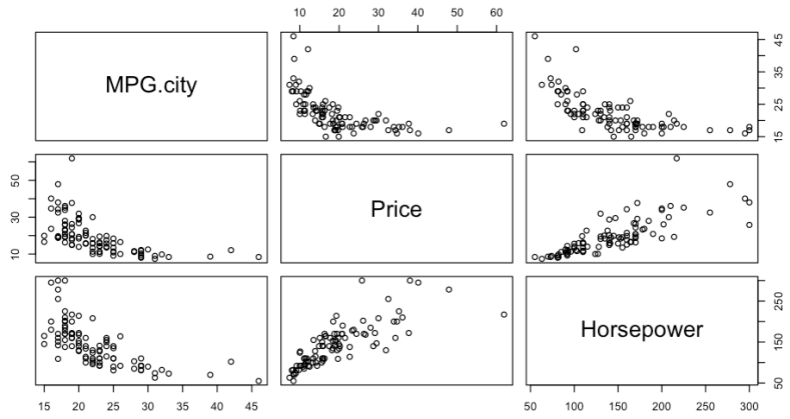


25

Gráficos: Gráfico de Dispersão.

- Para gerar relacionamento entre mais de duas variáveis

```
> subconjunto <- subset(Cars93, select=c(MPG.city, Price,
Horsepower))
> pairs(subconjunto)
```

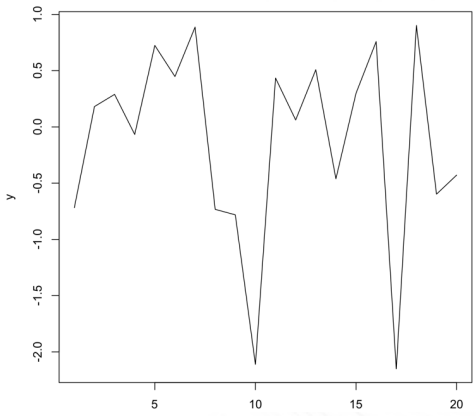


26

Gráficos: Gráfico de Linhas

- Função `plot(x, y, type="l")`
- Cria um gráfico de linhas
- Possui várias configurações

```
> x <- 1:20
> y <- rnorm(20)
> plot(x, y, type="l")
```

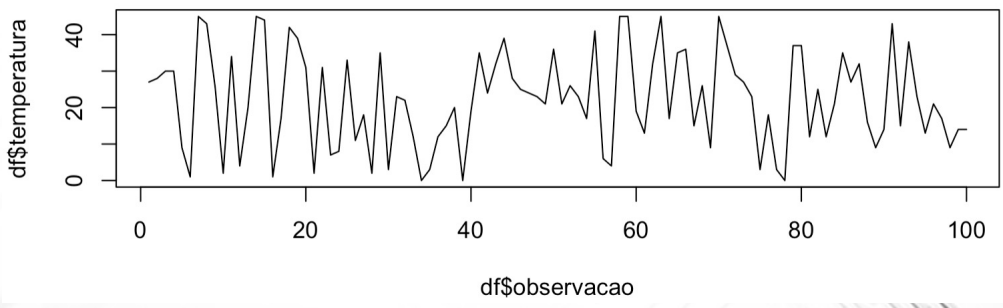


27

Gráficos: Gráfico de Linhas

- Seja a seguinte base de dados, observações de temperaturas em vários dias

```
> n_dados <- 100
> v_temp <- sample(0:45, n_dados, replace=T)
> v_obs <- 1:n_dados
> df <- data.frame(observacao=v_obs, temperatura=v_temp)
> plot(df$observacao, df$temperatura, type="l")
```

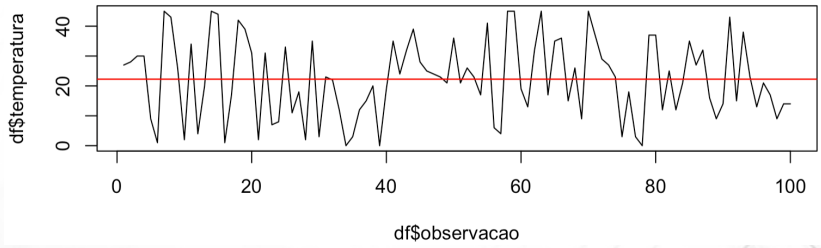


28

Gráficos: Gráfico de Linhas.

- No exemplo anterior, pode-se adicionar uma linha média com `abline()`
  - `abline(a, b, v, h)`: desenha uma linha no gráfico
    - `a`: intercepto
    - `b`: inclinação
    - `v`: linha vertical
    - `h`: linha horizontal

```
> plot(df$observacao, df$temperatura, type="l")
> abline(h=mean(df$temperatura), col="red")
```

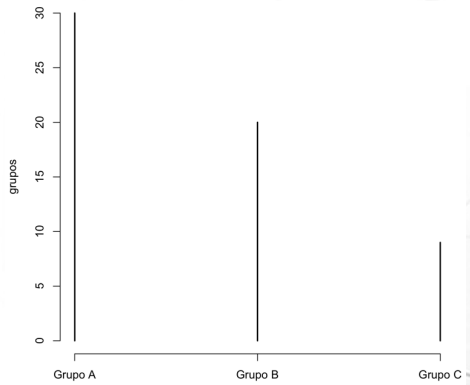


29

Gráficos: Gráfico de Barras

- Função `plot(x)`
- Cria um gráfico de barras
- Possui várias configurações

```
> grupoA <- rep("Grupo A", 30)
> grupoB <- rep("Grupo B", 20)
> grupoC <- rep("Grupo C", 9)
> grupos <- c(grupoA, grupoB, grupoC)
> grupos <- table(grupos)
> grupos
grupos
Grupo A Grupo B Grupo C
     30      20       9
> plot(grupos)
```

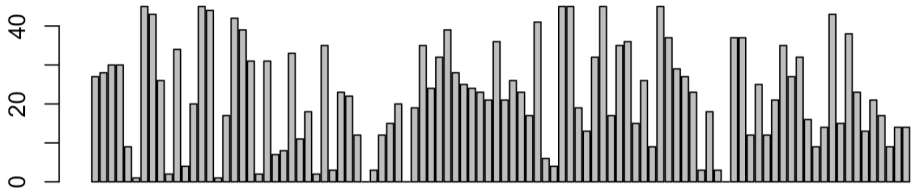


30

Gráficos: Gráfico de Barras

- Usa-se também: `barplot()`
  - Recebe um vetor ou uma matriz
- Usando as temperaturas do exemplo anterior

```
> n_dados <- 100
> v_temp <- sample(0:45, n_dados, replace=T)
> v_obs <- 1:n_dados
> df <- data.frame(observacao=v_obs, temperatura=v_temp)
> barplot(df$temperatura)
```



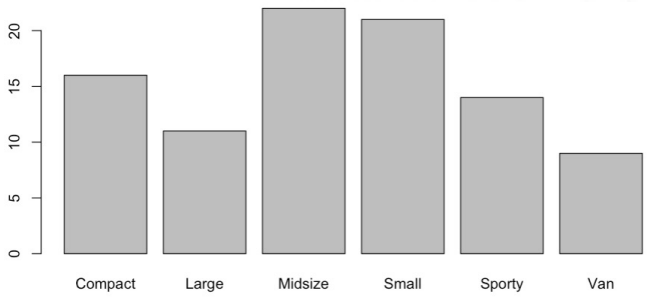
Gráficos: Gráfico de Barras

- Seja a base Cars93
- Queremos um gráfico de barras com as quantidades dos carros por tipo
- Para conseguir a tabela de frequências por tipo

```
> table(Cars93$Type)
```

- Para plotar o gráfico

```
> barplot(table(Cars93$Type))
```

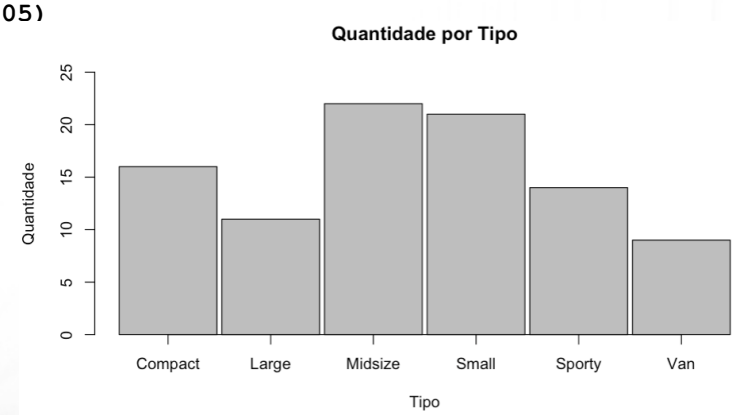


Gráficos: Gráfico de Barras.

- Melhorando o gráfico

- Para plotar o gráfico

```
> barplot(table(Cars93$Type), main="Quantidade por Tipo",
ylim=c(0,25), xlab="Tipo", ylab="Quantidade", axis.lty="solid",
space=.05)
```



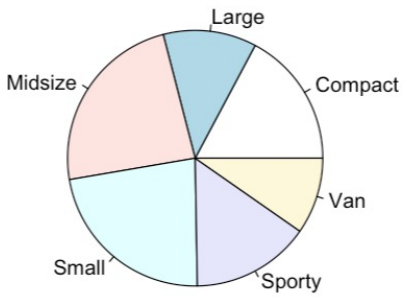
33

Gráficos: Gráfico de Pizza

- Para um gráfico de pizza

- Para plotar o gráfico

```
> pie(table(Cars93$Type))
```

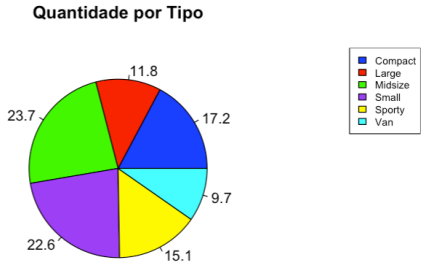


34

Gráficos: Gráfico de Pizza.

- Melhorando o gráfico

```
> t <- table(Cars93$Type)
> porcentagens<- round(100*t/sum(t), 1)
> cores=c("blue", "red", "green", "purple", "yellow", "cyan")
> pie(table(Cars93$Type), labels=porcentagens,
main="Quantidade por Tipo", col=cores)
> legend("topright", legend=names(t), cex=0.7, fill=cores)
```

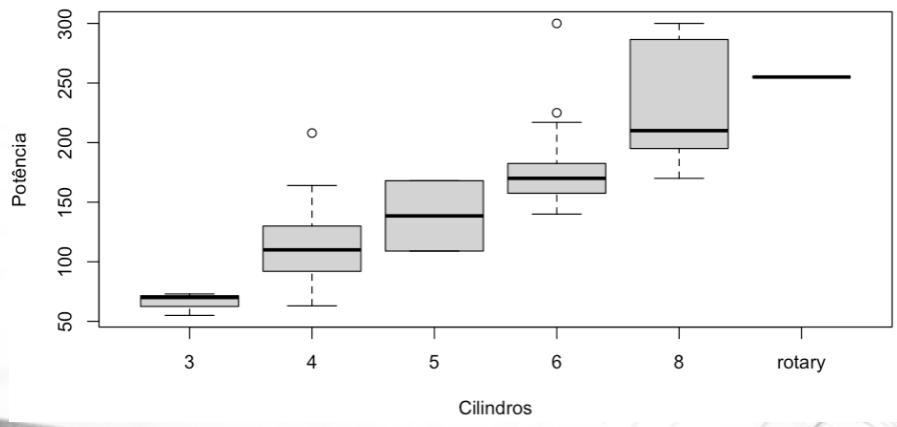


35

Gráficos: Boxplot

- Para a base Cars93
  - Relacionamento entre Potência e Cilindros

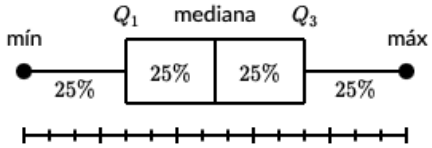
```
> boxplot(Cars93$Horsepower ~ Cars93$Cylinders,
xlab="Cilindros", ylab="Potência")
```



36

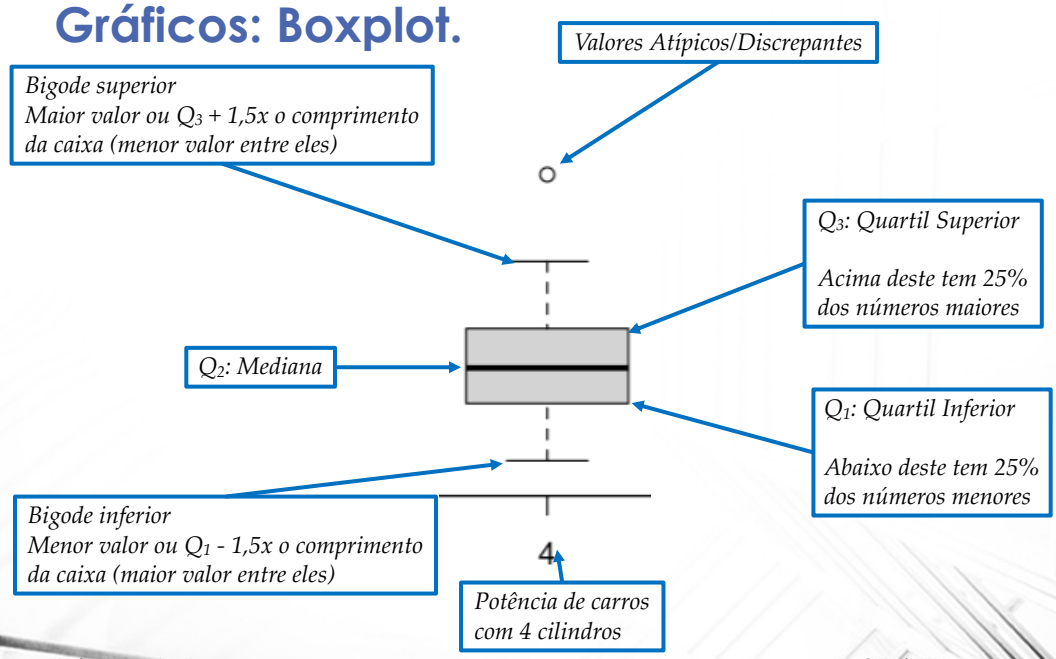
Gráficos: Boxplot

- Baseado em quartis
  - Q_1 : 25% dos dados são inferiores a Q_1 e 75% superiores a Q_1
  - Q_2 - Mediana: Média dos valores do meio
  - Q_3 : 25% dos dados são superiores a Q_3 e 75% inferiores a Q_3
  - Os bigodes são valores mínimos e máximos
    - Superior: Maior valor OU Q_3 + 1,5 x o tamanho do quadro (o que for menor)
    - Inferior: Menor valor OU Q_1 + 1,5 x o tamanho do quadro (o que for maior)



37

Gráficos: Boxplot.



38

ggplot2

- Criado por Hadley Wicknam
- gg = gramática de gráficos
- Ideia: combinar elementos básicos de gráficos
- Referência: https://r-graphics.org
- Instalação

```
> install.packages("ggplot2")
```

- Carga

```
> library(ggplot2)
```


39

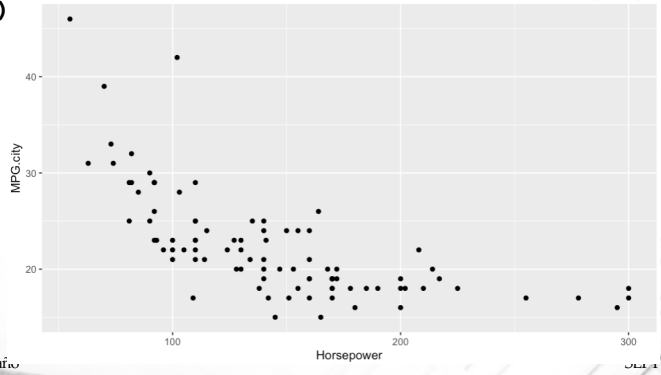
ggplot2

- Funcionamento básico

```
> ggplot(dataframe, aes(x=coluna_x, y=coluna_y) +
tipo_do_gráfico + labels
```

- Exemplo

```
> ggplot(Cars93, aes(x=Horsepower, y=MPG.city)) +
geom_point()
```



40

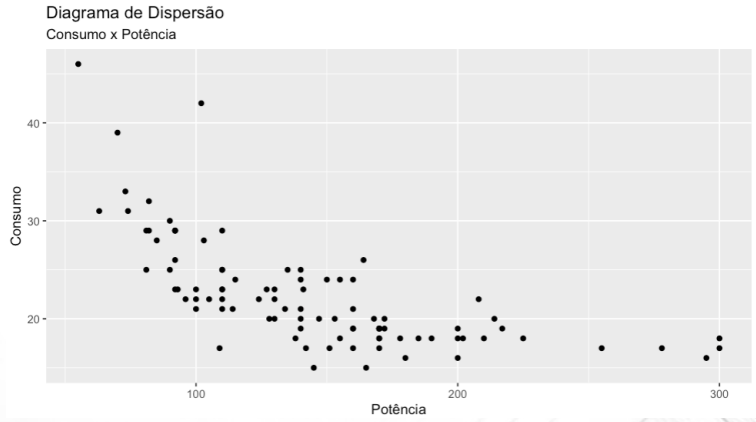
ggplot2

- Tipos de gráficos:
  - **`geom_point()`** : gráfico de pontos, gráfico de dispersão
  - **`geom_line()`** : gráfico de linhas
  - **`geom_col()`** : gráfico de barras (colunas)
  - **`geom_bar()`** : gráfico de colunas com a quantidade de dados (como um histograma)
  - **`geom_histogram()`** : histograma (alias para geom_bar() + stat_bin())
  - **`geom_boxplot()`** : gráfico de caixa e bigode
  - **`stat_function()`** : gráfico da curva de uma função
- Labels:
  - **`labs(x=texto, y=texto, title=texto, subtitle=texto)`** : manipula os rótulos dos eixos

ggplot2: Diagrama de Dispersão.

- Usa-se **`geom_point()`**

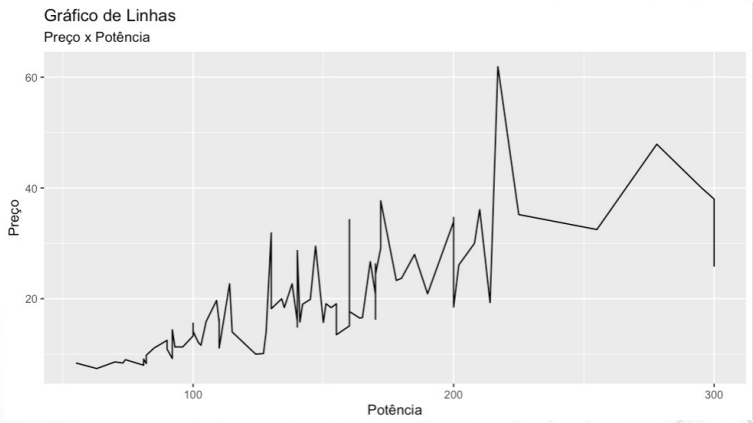
```
> ggplot(Cars93, aes(x=Horsepower, y=MPG.city)) +
  geom_point() +
  labs(x="Potência", y="Consumo", title="Diagrama de Dispersão",
subtitle="Consumo x Potência")
```



ggplot2: Gráfico de Linhas.

- Usa-se **geom_line()**

```
> ggplot(Cars93, aes(x=Horsepower, y=Price)) + geom_line() +
labs(x="Potência", y="Preço", title="Gráfico de Linhas", subtitle="Preço
x Potência")
```

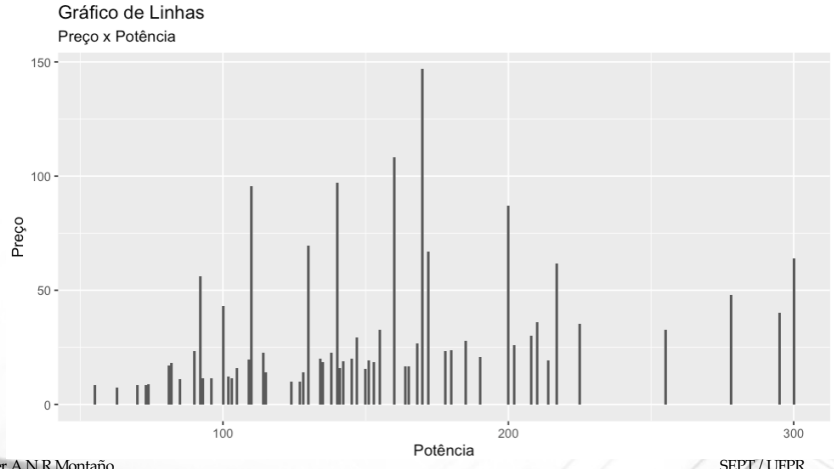


43

ggplot2: Gráfico de Barras.

- Usa-se **geom_col()**

```
> ggplot(Cars93, aes(x=Horsepower, y=Price)) + geom_col() +
labs(x="Potência", y="Preço", title="Gráfico de Linhas", subtitle="Preço
x Potência")
```

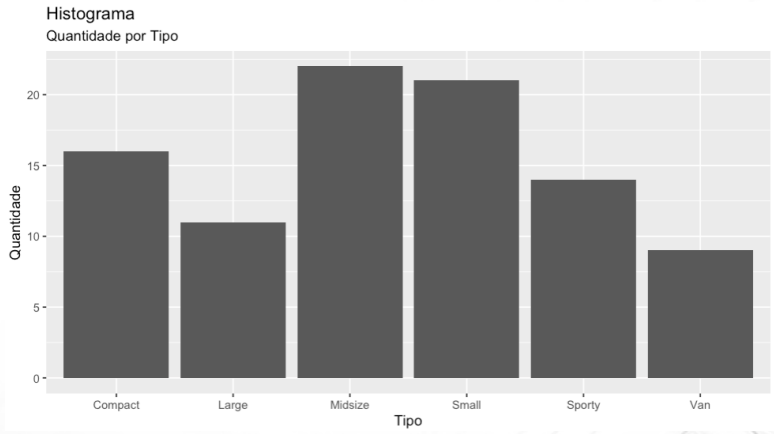


44

ggplot2: Histograma.

- Usa-se **geom_histogram(stat="count")**

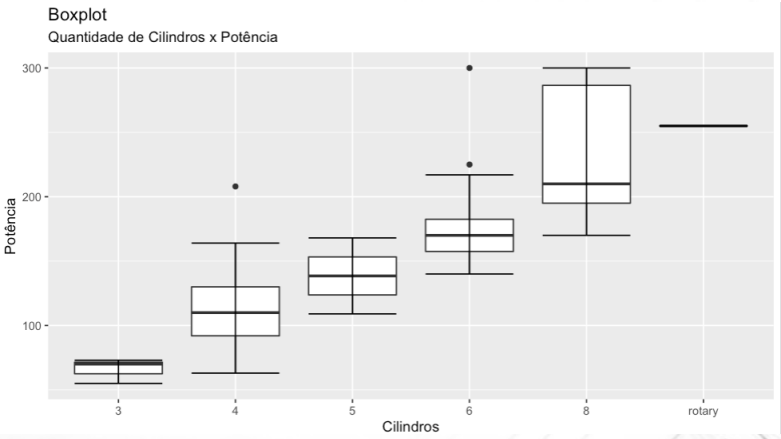
```
> ggplot(Cars93, aes(x=Type)) + geom_histogram(stat="count") +
labs(x="Tipo", y="Quantidade", title="Histograma", subtitle="Quantidade
por Tipo")
```



ggplot2: Boxplot.

- Usa-se **geom_boxplot(stat="count")**

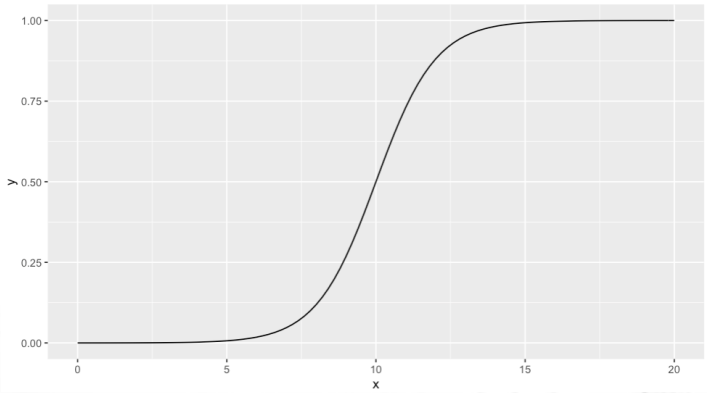
```
> ggplot(Cars93, aes(x=Cylinders, y=Horsepower)) + geom_boxplot() +
labs(x="Cilindros", y="Potência", title="Boxplot", subtitle="Quantidade
de Cilindros x Potência") + stat_boxplot(geom='errorbar')
```



ggplot2: Função..

- Usa-se **stat_function()**

```
> func <- function(x) {
  1 / (1 + exp(-x + 10))
}
> ggplot(data.frame(x=c(0, 20)), aes(x=x)) + stat_function(fun=func,
geom="line")
```



47

Gráficos: Salvar em Arquivo

- Para gerar os gráficos (`plot`, `abline`, `lines`)
  - Pode-se salvar em **PDF**

```
pdf(file="teste.pdf")
...
plot(...)
pines(...)
ggplot()
...
dev.off()
```


48

Gráficos: Salvar em Arquivo.

- Para gerar os gráficos (`plot`, `abline`, `lines`)
  - Pode-se salvar em **PNG**

```
png(file="teste.png")
...
plot(...)
pines(...)
ggplot()
...
dev.off()
browseURL("teste.png")
```


Exercícios.

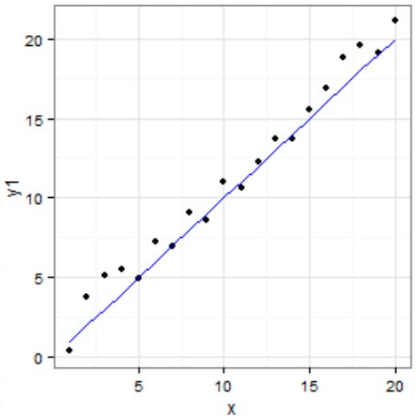
1. Execute os exercícios apresentados nos slides
2. Gere os gráficos em arquivos PDF e PNG

Regressão

- Dado um gráfico de pontos
  - Pode-se passar uma reta para resumir a relação entre os dados
  - Essa reta também pode ser usada para predizer novos valores
- Equação da reta:

```
y1 = a + b*x
```

- Os dados `y1` e `x` são conhecidos
  - Deve-se encontrar `a` e `b`
  - `y1` : variável dependente
  - `x` : variável independente
  - `a` : Representa interceptador com o eixo vertical
  - `b` : Coeficiente angular, inclinação
- Ex.: Métodos dos Mínimos Quadrados



Regressão

- Exemplo de dados
  - Relação entre Km e valor de venda de um carro usado

Quilometragem (1000km) (x)	Preço de Venda (dez. Euros) (y)
40	1000
30	1500
30	1200
25	1800
50	800
60	1000
65	500
10	3000
15	2500
20	2000
55	800
40	1500
35	2000
30	2000

Regressão

- Funções
  - `lm(<fórmula>, <dados>)` : Encontra um modelo linear
  - `summary(<modelo>)` : Dá o resultado sumarizado do modelo gerado
- Exemplos

```
> x <- c(40, 30, 30, 25, 50, 60, 65, 10, 15, 20, 55, 40, 35, 30)
> y <- c(1000, 1500, 1200, 1800, 800, 1000, 500, 3000, 2500, 2000, 800, 1500,
2000, 2000)
> modelo <- lm(y ~ x)
> modelo

Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)            x
    2933.60        -38.56
```


53

Regressão

- Para obter a sumarização do modelo

```
> summary(modelo)

Call:
lm(formula = y ~ x)

Residuals:
    Min      1Q  Median      3Q     Max
-576.94 -196.81   29.71  203.48  451.95

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 2933.597    213.781  13.722 1.07e-08 ***
x            -38.555      5.414  -7.121 1.21e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 325.3 on 12 degrees of freedom
Multiple R-squared:  0.8086, Adjusted R-squared:  0.7927
F-statistic: 50.71 on 1 and 12 DF,  p-value: 1.212e-05
```

$y = 2933.596 - 38.555 * x$

54

Regressão

- Analisando a estrutura do modelo, consegue-se obter outros dados

```
> str(modelo)
List of 12
 $ coefficients : Named num [1:2] 2933.6 -38.6
  ..- attr(*, "names")= chr [1:2] "(Intercept)" "x"
 $ residuals    : Named num [1:14] -391 -277 -577 -170 -206 ...
  ..- attr(*, "names")= chr [1:14] "1" "2" "3" "4" ...
 $ effects      : Named num [1:14] -5772.8 -2316.2 -474.4 -52.6 -161.3 ...
  ..- attr(*, "names")= chr [1:14] "(Intercept)" "x" "" "" ...
 $ rank         : int 2
 $ fitted.values: Named num [1:14] 1391 1777 1777 1970 1006 ...
  ..- attr(*, "names")= chr [1:14] "1" "2" "3" "4" ...
 $ assign       : int [1:2] 0 1
........
> modelo$coefficients
(Intercept)           x
 2933.59723   -38.55517
```


Regressão

- Funções importantes
  - `coef(<modelo>)` : Mostra os coeficientes do modelo
  - `predict(<modelo>, <novos dados>)` : Prediz o valor de novos dados com o modelo gerado
  - `resid(<modelo>)` : Apresenta os resíduos gerados no ajuste do modelo

Regressão

- Exemplo

```
> x <- c(40, 30, 30, 25, 50, 60, 65, 10, 15, 20, 55, 40, 35, 30)
> y <- c(1000, 1500, 1200, 1800, 800, 1000, 500, 3000, 2500,
2000, 800, 1500, 2000, 2000)
> modelo <- lm(y ~ x)
> coef(modelo)
(Intercept)           x
 2933.59723   -38.55517
```


57

Regressão

- Para fazer **predição** de novos valores, cria-se uma coleção de dados e aplica-se **`predict()`**
- Exemplo

Mesmo nome da coluna x

```
> novos <- data.frame( x=c(10, 20, 30) )
> predict(modelo, novos)
       1        2        3
2548.046 2162.494 1776.942
> resid(modelo)
         1          2          3          4          5          6          7
-391.39040 -276.94211 -576.94211 -169.71796 -205.83869  379.71301   72.48887
         8          9         10         11         12         13         14
 451.95448  144.73033 -162.49381  -13.06284  108.60960  415.83375  223.05789
```


58

Regressão

- Com a equação de reta

```
y = 2933.596 – 38.555 * x
```

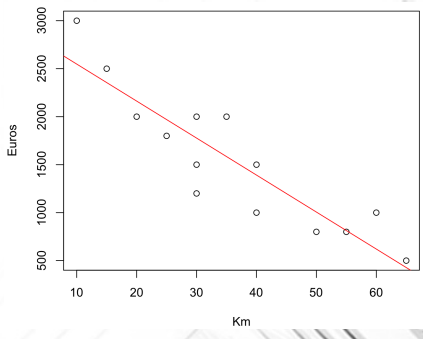
- Pode-se criar uma função no R

```
> venda <- function(v) {
+ return(2933.597 - 38.555 * v)
+ }
> venda(40)
[1] 1391.397
> venda(30)
[1] 1776.947
> venda(25)
[1] 1969.722
```


Regressão

- Depois do cálculo, pode-se plotar o gráfico de dispersão e a reta encontrada
  - `abline()` plota uma reta.
  - `abline(modelo)` plota a reta de regressão.

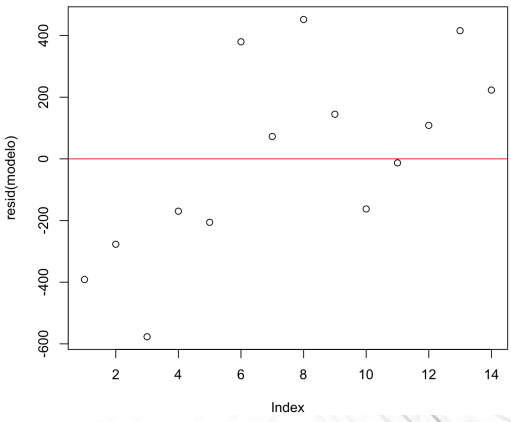
```
> x <- c(40, 30, 30, 25, 50, 60, 65, 10, 15, 20, 55, 40, 35, 30)
> y <- c(1000, 1500, 1200, 1800, 800, 1000, 500, 3000, 2500, 2000, 800, 1500,
2000, 2000)
> modelo <- lm(y ~ x)
> plot(x=x, y=y, xlab="Km", ylab="Euros")
> abline(modelo, col="red")
```



Regressão

- Como a regressão APROXIMA uma reta dos dados, um erro é gerado: RESÍDUOS
  - Obtém-se os resíduos com a função `resid()`
- Consegue-se plotar estes resíduos

```
> plot(resid(modelo))
> abline(0,0,col="red")
```

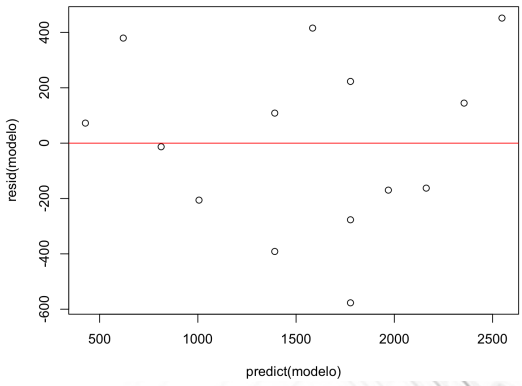


61

Regressão.

- Resíduos x Valor Predito

```
> plot(resid(modelo) ~ predict(modelo))
> abline(0,0,col="red")
```



62

Exercícios.

- Efetuar a análise de regressão para os seguintes dados. Mostre as estatísticas, equação de reta e plote o gráficos de dispersão com a reta, e os gráficos de resíduos

Variável Independente (x)	Variável Dependente (y)
6.5	1.4
5.8	1.5
7.8	1.7
8.1	1.9
10.4	2.1
12.3	2.2
13.1	2.4
17.4	3.2
20.1	3.7
24.5	4.2
25.5	4.8
27.1	5.2

63

Carga de Arquivos

- Para ler um arquivo CSV usa-se `read.csv`

```
dados <- read.csv("/Users/razer/teste.csv")
```

- Gera um Data Frame dos dados
- Parâmetros
  - `header = TRUE | FALSE`
  - `sep = ","`
  - `dec = "."`
  - `fileEncoding = "UTF-8"`
- Dado o arquivo `fruitohms.csv` (Moodle)
  - Contém a relação entre % de suco e resistência elétrica (ohms) em Kiwis
  - Faça download e coloque em uma pasta de seu conhecimento

64

Carga de Arquivos

- Exemplo

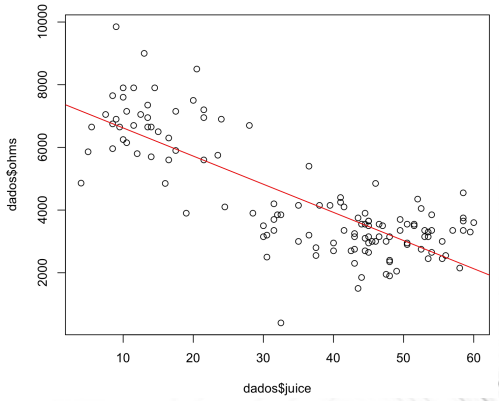
```
> dados <- read.csv("/Users/razer/fruitohms.csv")
> modelo <- lm(ohms ~ juice, data=dados)
> summary(modelo)

Call:
lm(formula = ohms ~ juice, data = dados)

Residuals:
    Min      1Q  Median      3Q     Max
-4198.4  -793.9     3.8   614.3  3139.5

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 7519.404    233.779   32.16   <2e-16 ***
juice        -89.877      6.022  -14.93   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1122 on 126 degrees of freedom
Multiple R-squared:  0.6387, Adjusted R-squared:  0.6358
F-statistic: 222.7 on 1 and 126 DF,  p-value: < 2.2e-16
```


65

Carga de Arquivos.

- Exemplo

```
> plot(dados$ohms ~ dados$juice)
> abline(modelo)
```



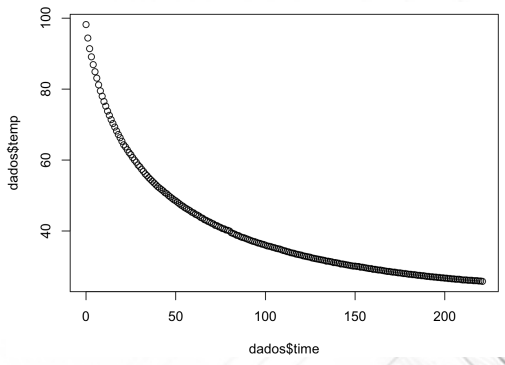
66

Exercícios.

- Efetuar a análise de regressão para os dados do arquivo `GAGurine.csv` que está no Moodle. Mostre as estatísticas, equação de reta e plote o gráficos de dispersão com a reta, e os gráficos de resíduos
  - Este arquivo contém a medição de níveis de GAG (glicosaminoglicanos) na urina de crianças de certa idade
  - http://www.razer.net.br/datasets/GAGurine.csv

Regressão Polinomial

- A variável independente é descrita por um polinômio (`x^2`, `x^3`, etc)
- Base CoolingWater.csv
  - Relação entre o tempo e resfriamento da água
  - http://www.razer.net.br/datasets/CoolingWater.csv



Regressão Polinomial

- O que se quer é um modelo :

$$y = a + bx + cx^2 + \ ... + zx^n$$

- Pode-se usar uma fórmula com **`I()`** ou a função **`poly()`**:

```
> dados <- read.csv("/Users/razer/CoolingWater.csv")

> modelo <- lm(temp ~ poly(time, 2, raw=TRUE), data=dados)
> modelo <- lm(temp ~ time + I(time^2)), data=dados)
```

- A opção **`raw`** serve para contornar a questão sobre **`x`** ser altamente dependente de **`x^2`**, gerando polinômios ortogonais
  - TRUE : gera os polinômios simples
  - FALSE (default) : gera os polinômios ortogonais

69

Regressão Polinomial.

- Fazer a regressão da seguinte forma:

```
> dados <- read.csv("/Users/razer/CoolingWater.csv")
> modelo <- lm(temp ~ poly(time, degree=2), data=dados)
> plot(dados$temp ~ dados$time)
```

- Depois, gere um vetor de predição

```
> predicao <- predict(modelo, dados)
```

- Depois plote a curva estimada

```
> lines(y=predicao, x=dados$time, type="l", col="red")
```


70

Exercícios.

1. Para os dados CoolingWater, gere as estimativas com um polinômio de grau 3 e de grau 4. Compare as curvas de predição.
2. Use a base cars do R
   - Gere modelos polinomiais de graus 2, 3, 4 e 5
   - Faça as predições com cada modelo
   - Plote os pontos dos dados
   - Plote as linhas de cada uma das predições efetuadas, em cores diferentes
3. Gerar os PDFs dos gráficos com as predições de grau 2, 3 e 4, para o problema CoolingWater, usando scripts R.