MAP 2112 – Introdução à Lógica de Programação e Modelagem Computacional

1º Semestre - 2018

Prof. Dr. Luis Carlos de Castro Santos

lsantos@ime.usp.br/lccs13@yahoo.com

ROTEIRO

Material dos Profs. D.T. Kaplan, R.J; Pruim e N.J.Horton

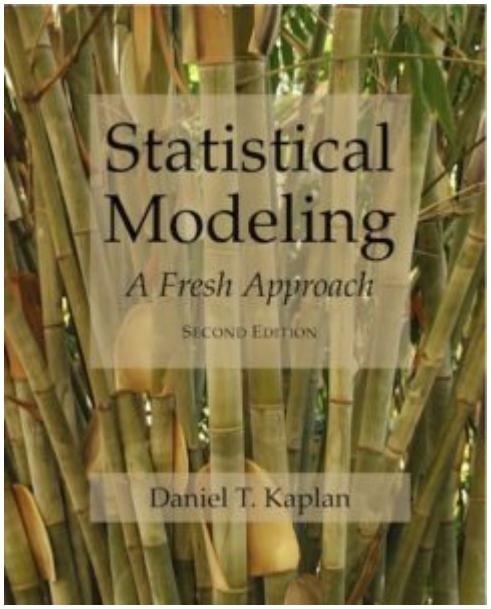
Projeto Mosaic

http://project-mosaic-books.com/?page_id=13

Material disponível html

http://mosaic-web.org/go/SM2-technique/

A partir do capítulo 3



Algum material sobre estatística será retirado do curso Genome560 do Prof. Akey da Universidade de Washington (UW)

http://www.gs.washington.edu/academics/courses/akey/56008/lecture.htm

Chapter 6 Language of models

At the core of the language of modeling is the notation that uses the tilde character (~) to identify the response variable and the explanatory variables. This notation is incorporated into many of operators that you will use. As usual, many of the operators, as well as the datasets required in this section come from the `mosaic` package, so it should be loaded if it isn't loaded already.

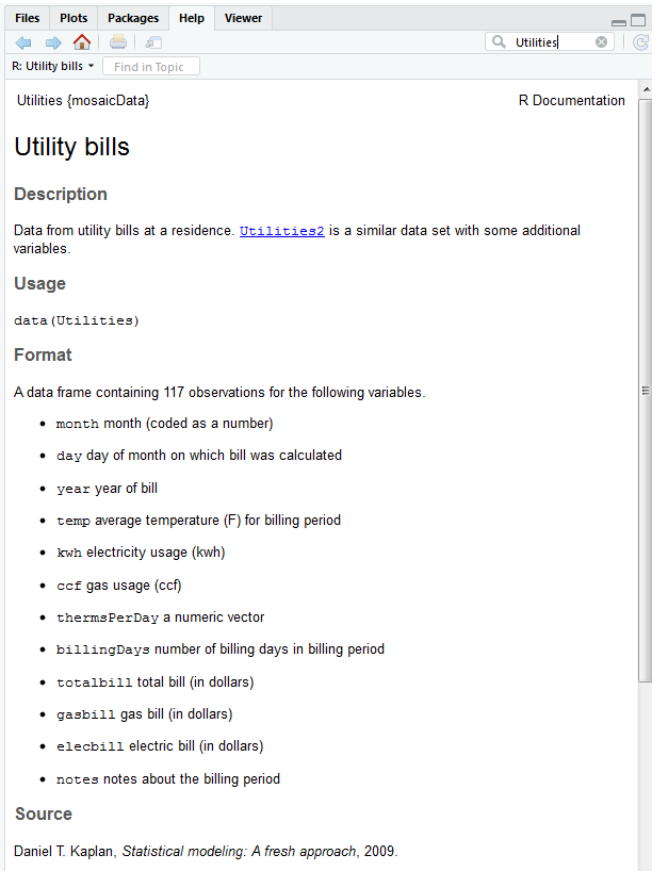
```
require(mosaic)
```

To illustrate the computer commands for modeling and graphically displaying relationships between variables, use the utilities data set:

```
Utils <- Utilities  # from mosaicData
```

Some of the following examples make particular use of these variables #. `ccf` — the natural gas usage in cubic feet during the billing period. #. `month` — the month coded as 1 to 12 for January to December. #. `temp` — the average temperature during the billing period.

```
> head(Utilities)
  month day year temp kwh ccf thermsPerDay billingDays totalbill
1    12  29 1999   26 892 194          5.5          36    173.65
2     1  28 2000   18 533 164          5.6          30    139.18
3     2  26 2000   24 521 228          8.0          29    177.48
4     3  25 2000   41 554  16          0.6          28     61.27
5     4  28 2000   45 638  74          2.2          34    100.33
6     5  30 2000   60 700 129          4.1          32    153.32
  gasbill elecbill             notes
1  112.72    68.25
2   95.88    43.30
3  134.65    42.83
4   15.32    45.95 bad meter reading
5   47.33    53.00
6   89.87    63.45
>
```



Files | Plots | Packages | Help | Viewer

R: Utility bills — Find in Topic — Utilities

Utilities {mosaicData} — R Documentation

Utility bills

Description

Data from utility bills at a residence. `Utilities2` is a similar data set with some additional variables.

Usage

```
data(Utilities)
```

Format

A data frame containing 117 observations for the following variables.

- `month` month (coded as a number)
- `day` day of month on which bill was calculated
- `year` year of bill
- `temp` average temperature (F) for billing period
- `kwh` electricity usage (kwh)
- `ccf` gas usage (ccf)
- `thermsPerDay` a numeric vector
- `billingDays` number of billing days in billing period
- `totalbill` total bill (in dollars)
- `gasbill` gas bill (in dollars)
- `elecbill` electric bill (in dollars)
- `notes` notes about the billing period

Source

Daniel T. Kaplan, *Statistical modeling: A fresh approach*, 2009.

6.1 Bi-variate Plots

The basic idea of a bi-variate (two variable) plot is to examine one variable as it relates to another. The conventional format is to plot the response variable on the vertical axis and an explanatory variable on the horizontal axis.

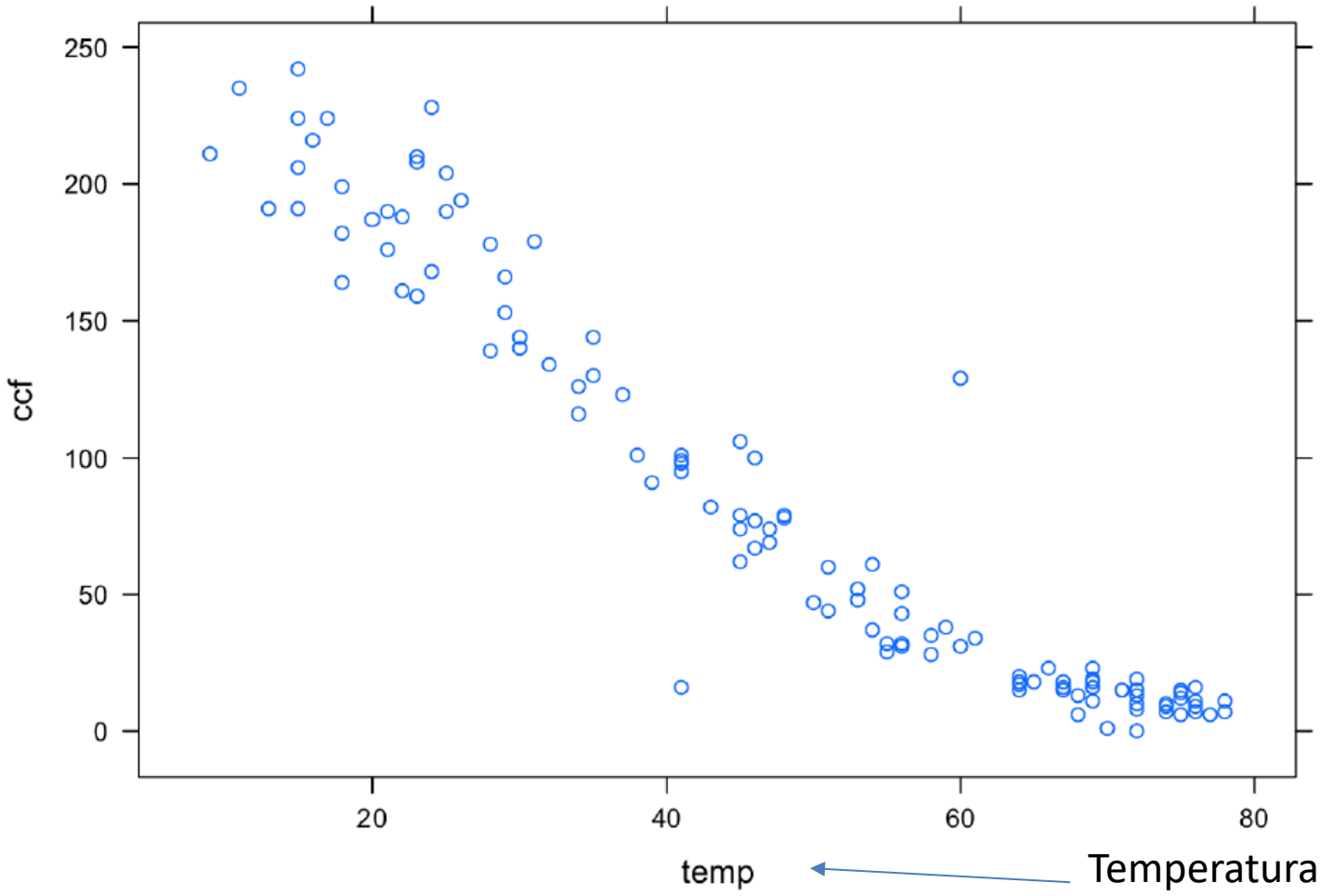
6.1.1 Quantitative Explanatory Variable

When the explanatory variable is quantitative, a scatter-plot is an appropriate graphical format. In the scatter plot, each case is a single point.

Explanatory variable (Independent Variable)	Response Variable (Dependent Variable)
"Explains observed outcomes"	"Measures the outcome of a study"
Example:	A subject takes a new weight-loss pill (Explanatory) and the weight lost (Response) is measured after one month.

The basic computer operator for making scatter plots is `xyplot()` :

```
xyplot( ccf ~ temp, data = Utils)
```



Consumo de gás

Temperatura média do período de leitura

Alguma inferência é possível ?

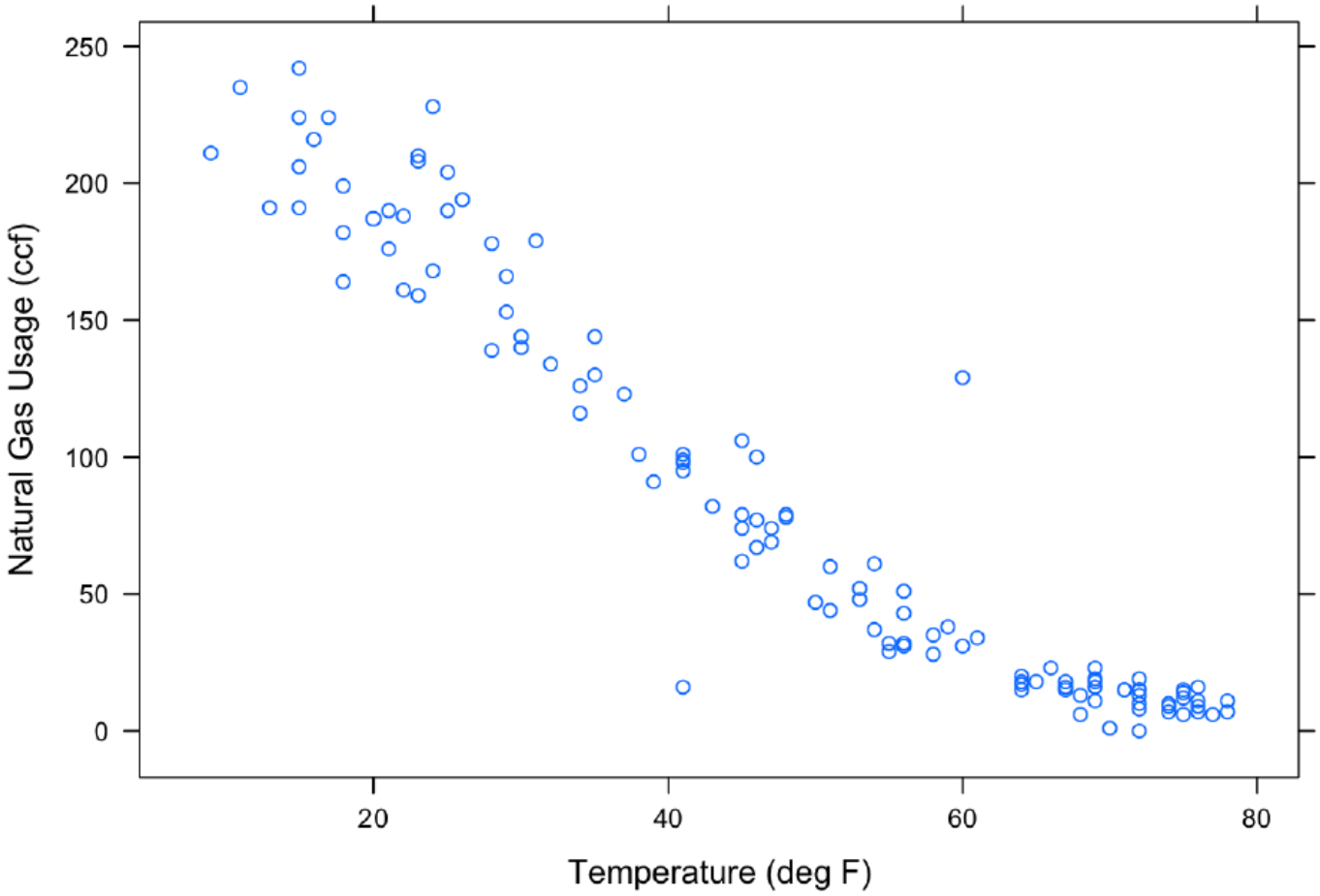
```
xyplot( ccf ~ temp, data = Utils)
```

The first argument is a model formula written using the tilde modeling notation. This formula, `ccf ~ temp` is pronounced “ccf versus temperature.” It is traditional to plot the so-called dependent variable, here `ccf` , on the y–axis.

In order to keep the model notation concise, the model formula has left out the name of the data frame to which the variables belong. Instead, the frame is specified in the `data =` argument. Since `data` has been set to be `Utils` , the formula `ccf ~ temp` is effectively translated to `Utils$ccf ~ Utils$temp` .

You can specify the axis labels by hand, if you like. For example,

```
xyplot( ccf ~ temp, data=Utils,
        xlab = "Temperature (deg F)",
        ylab = "Natural Gas Usage (ccf)")
```

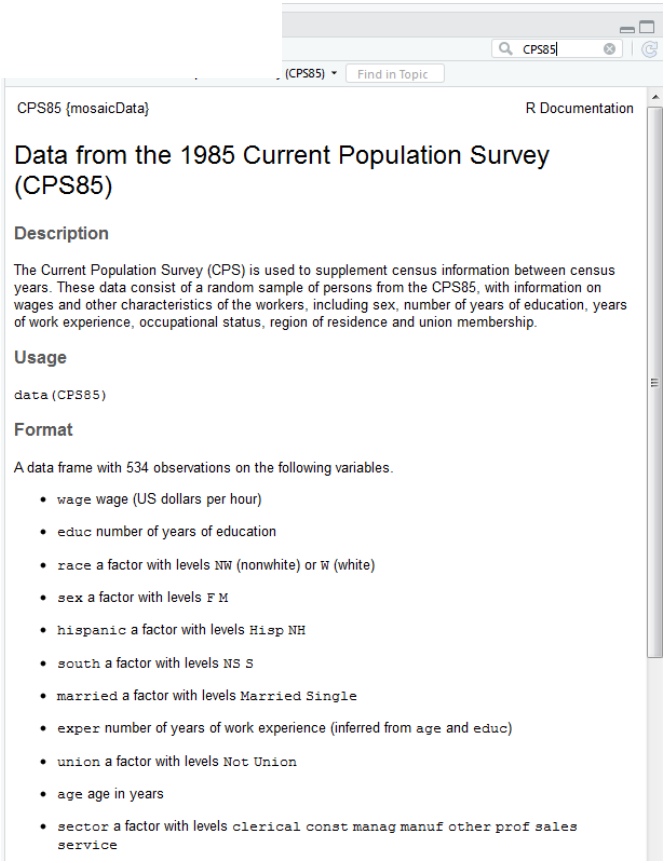


Another illustrative example uses Current Population Survey wage data:`CPS`

```
CPS <- CPS85 # from mosaicData
```

and focuses on the variables `wage` , `sex` , and `sector` .

```
> CPS <- CPS85
> head(CPS)
  wage educ race sex hispanic south married exper union age   sector
1  9.0   10    W   M       NH    NS Married    27   Not  43    const
2  5.5   12    W   M       NH    NS Married    20   Not  38    sales
3  3.8   12    W   F       NH    NS  Single     4   Not  22    sales
4 10.5   12    W   F       NH    NS Married    29   Not  47 clerical
5 15.0   12    W   M       NH    NS Married    40 Union  58    const
6  9.0   16    W   F       NH    NS Married    27   Not  49 clerical
> tail(CPS)
     wage educ race sex hispanic south married exper union age   sector
529 23.25   17   NW   F       NH    NS Married    25 Union  48     prof
530  8.63   12   NW   F       NH    NS Married    18   Not  36 clerical
531 18.50   14    W   F       NH    NS  Single    13   Not  33    manuf
532 22.20   18    W   M       NH    NS Married     8   Not  32     prof
533 16.26   12   NW   M       NH    NS  Single    14 Union  32  service
534 19.47   12    W   M       NH    NS  Single     9   Not  27    other
>
```



CPS85 {mosaicData} R Documentation

Data from the 1985 Current Population Survey (CPS85)

Description

The Current Population Survey (CPS) is used to supplement census information between census years. These data consist of a random sample of persons from the CPS85, with information on wages and other characteristics of the workers, including sex, number of years of education, years of work experience, occupational status, region of residence and union membership.

Usage

```
data(CPS85)
```

Format

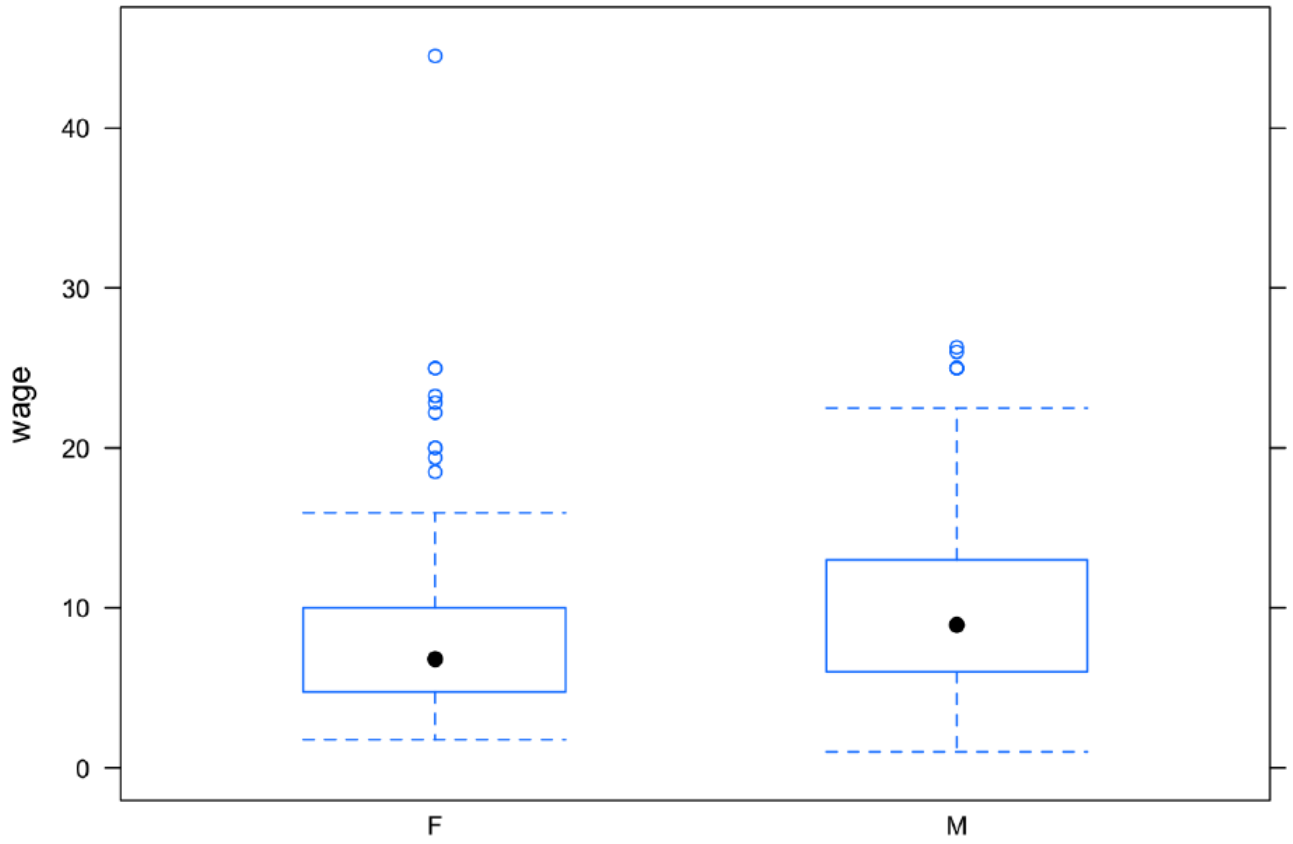
A data frame with 534 observations on the following variables.

- `wage` wage (US dollars per hour)
- `educ` number of years of education
- `race` a factor with levels `NW` (nonwhite) or `W` (white)
- `sex` a factor with levels `F M`
- `hispanic` a factor with levels `Hisp NH`
- `south` a factor with levels `NS S`
- `married` a factor with levels `Married Single`
- `exper` number of years of work experience (inferred from `age` and `educ`)
- `union` a factor with levels `Not Union`
- `age` age in years
- `sector` a factor with levels `clerical const manag manuf other prof sales service`

6.1.2 Categorical Explanatory Variable

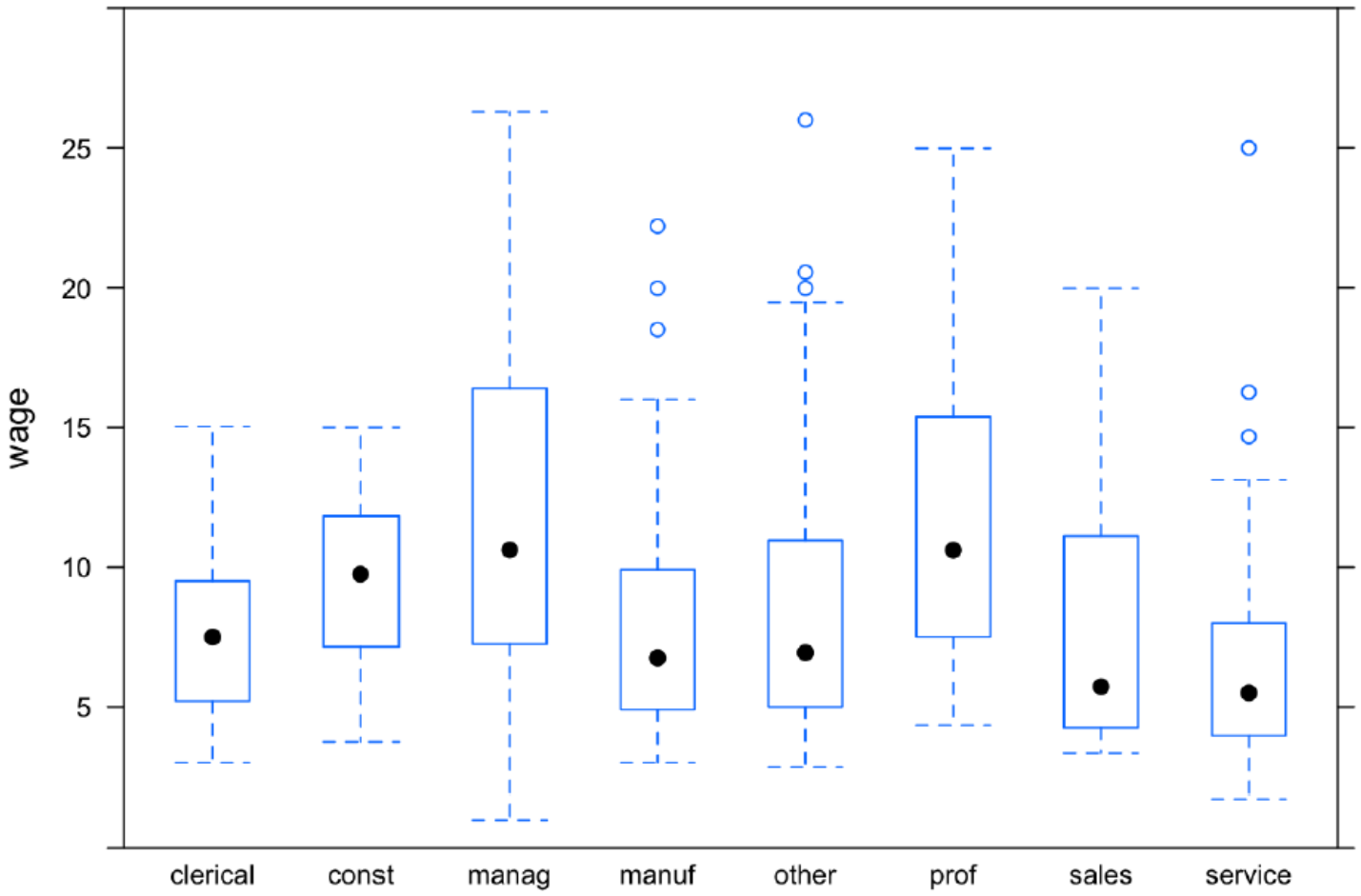
When the explanatory variable is categorical, an appropriate format of display is the box-and-whiskers plot, made with the `bwplot` operator. Here, for example, is the `wage` versus `sex` from the Current Population Survey:

```
bwplot( wage ~ sex, data=CPS)
```



Notice that the outliers are setting the overall vertical scale for the graph and obscuring the detail at typical wage levels. You can use the `ylim` argument to set the scale of the y-axis however you want. For example:

```
bwplot(wage~sector, data=CPS, ylim=c(0,30) )
```



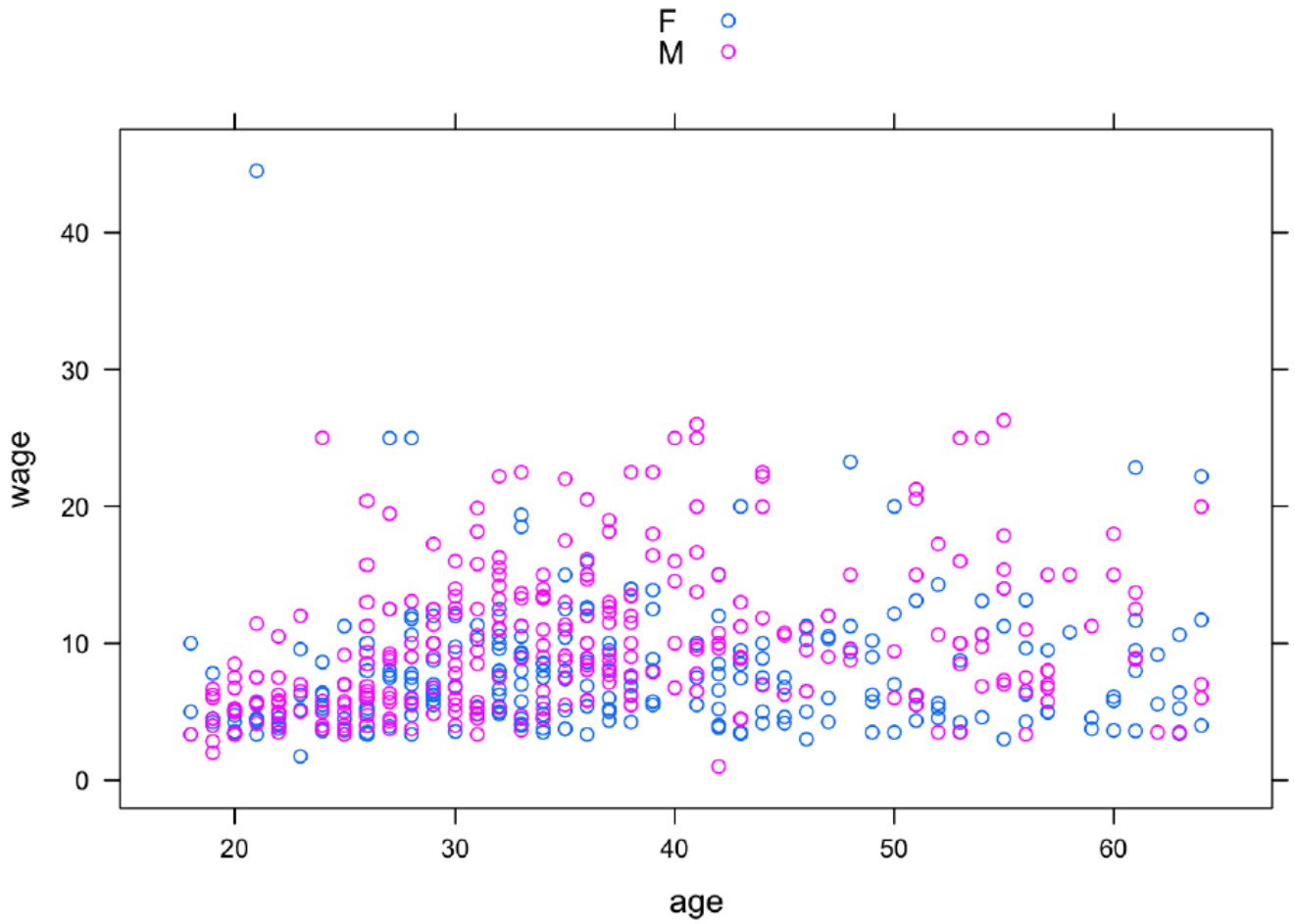
6.1.3 Multiple Explanatory Variables

The two-dimensional nature of paper or the computer screen lends itself well to displaying two variables: a response versus a single explanatory variable. Sometimes it is important to be able to add an additional explanatory variable. The graphics system gives a variety of options in this regard:

1. Coding the additional explanatory variable] using color or symbol shapes.
1. Splitting the plot into the groups defined by the additional explanatory variable.

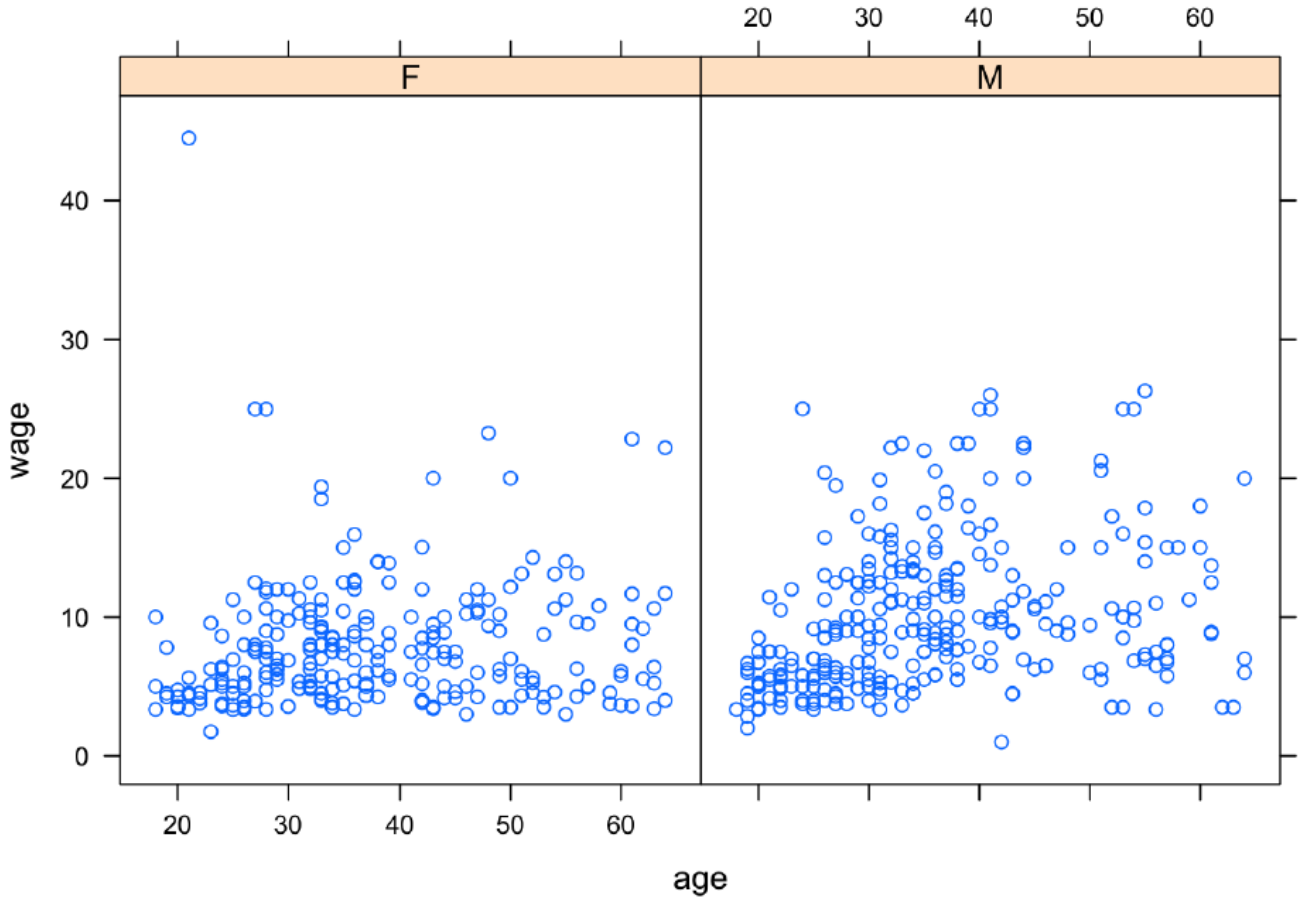
1. Coding the additional explanatory variable] using color or symbol shapes. This is done by using the `groups` argument set to the name of the additional explanatory variable. For example:

```
xyplot( wage ~ age, groups = sex, data = CPS,
    auto.key = TRUE)
```



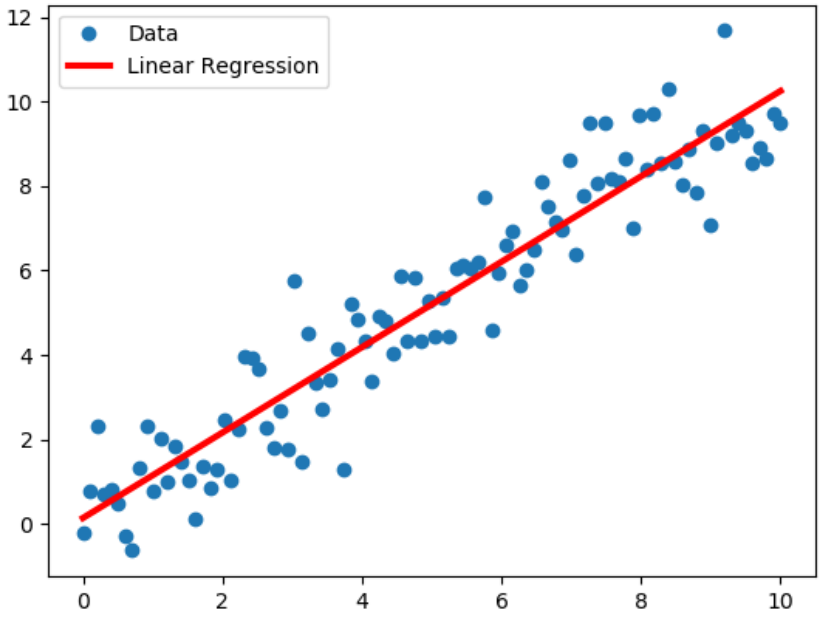
1. Splitting the plot into the groups defined by the additional explanatory variable. This is done by including the additional variable in the model formula using a | separator. For example:

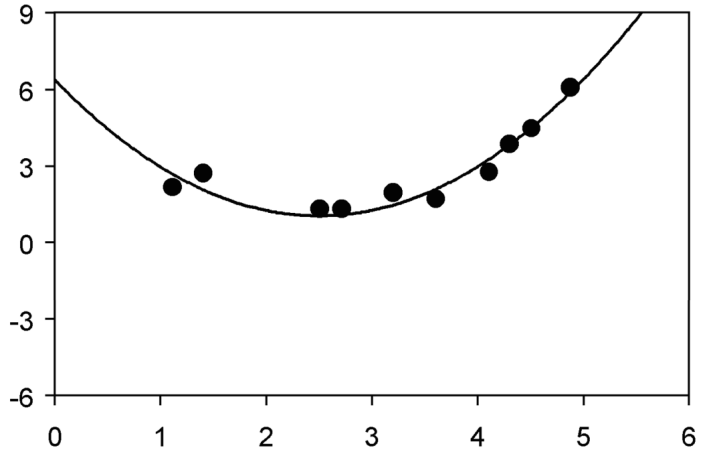
```
xyplot( wage ~ age | sex, data = CPS)
```

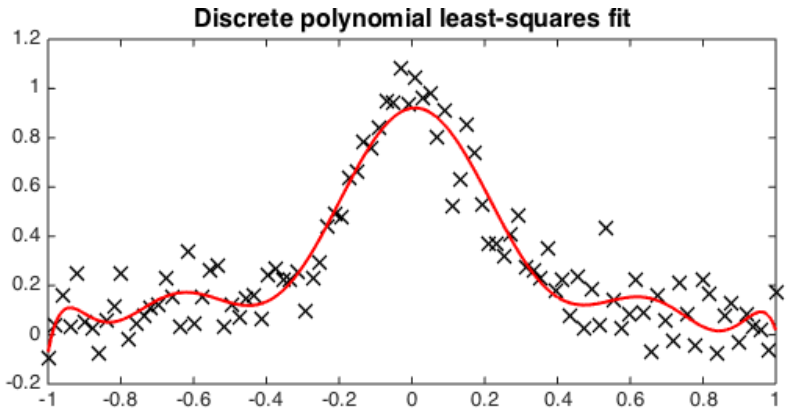


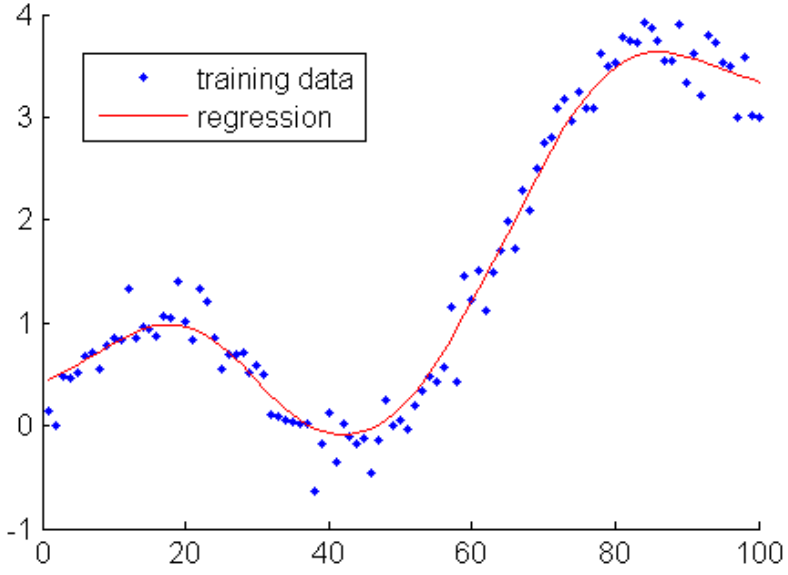
A partir de um conjunto de dados como obter um modelo que permita prever comportamento, ou auxiliar na tomada de decisão ?


DESVIO

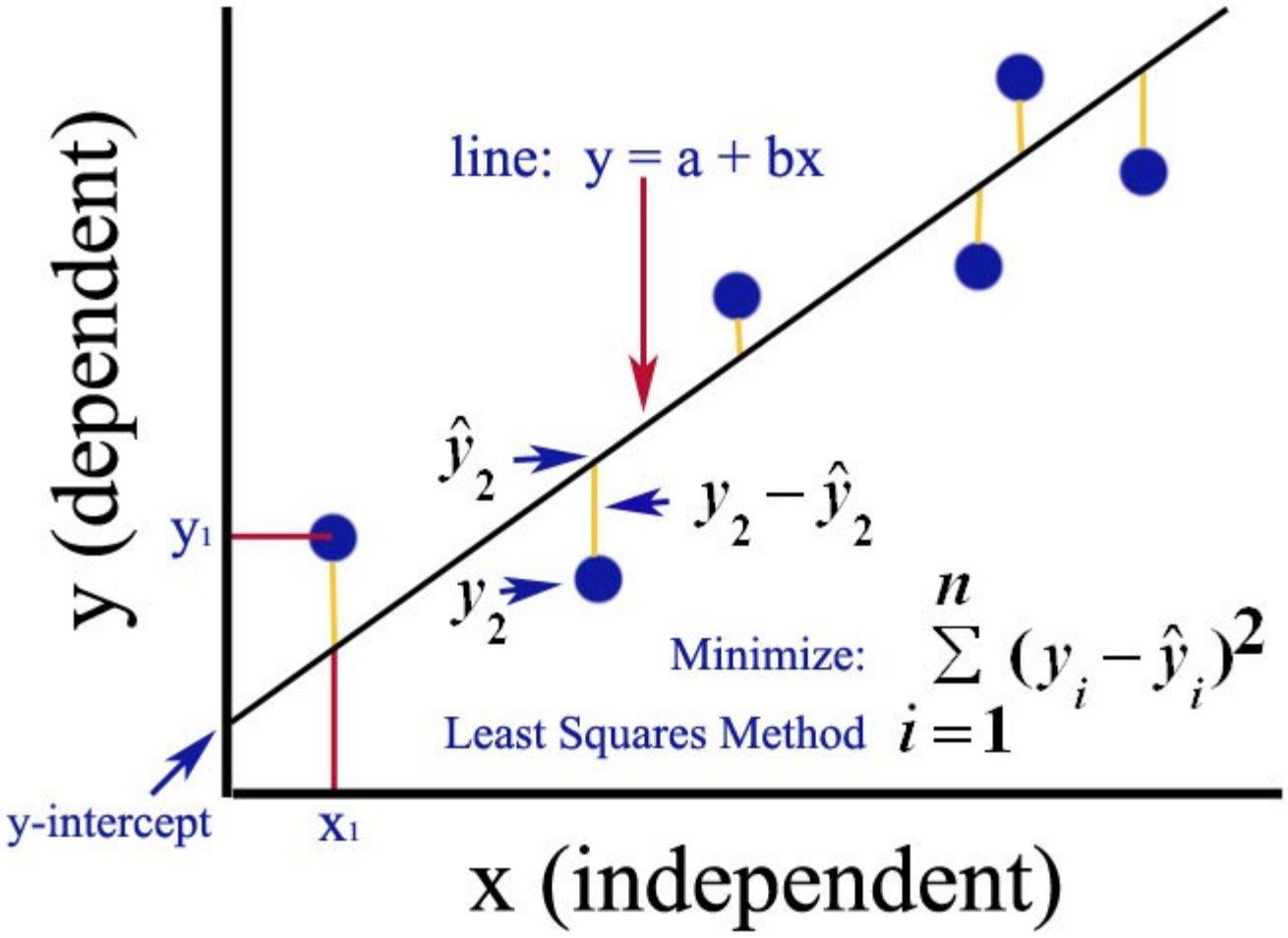

Data
Linear Regression




Discrete polynomial least-squares fit


training data
regression

Ordinary Least Squares (OLS)



https://xaviergeerinck.com/ordinary-least-squares

Simplifying this gives us that our error can be represented as: $d_i = y - (a + bx)$ or the sum for all the errors can be represented as $\sum_{i=1}^{n} d_i^2 = \sum_{i=1}^{n} (y_i - (a + bx))^2$

Calculating our Derivative $\frac{d}{da} = 0$

$\frac{d}{da} \sum (y - (a + bx))^2$

$= \sum \frac{d}{da} (y^2 - 2y(a + bx) + (a + bx)^2)$

$= \sum \frac{d}{da} (y^2 - 2ya - 2ybx + a^2 + 2abx + b^2x^2)$

$= \sum (-2y + 2a + 2bx)$

$= 2 \sum_{i=0}^{n} (a + bx - y)$

$\rightarrow na + \sum_{i=0}^{n} (bx - y) = 0 \leftrightarrow \sum_{i=0}^{n} y = na + b \sum_{i=0}^{n} x$

Calculating our Derivative $\frac{d}{db} = 0$

$\frac{d}{db} \sum (y - (a + bx))^2$

$= \sum \frac{d}{db} (y^2 - 2y(a + bx) + (a + bx)^2)$

$= \sum \frac{d}{db} (y^2 - 2ya - 2ybx + a^2 + 2abx + b^2x^2)$

$\rightarrow \sum (-2yx + 2ax + 2bx^2) = 0$

$\leftrightarrow - \sum yx + a \sum x + b \sum x^2 = 0$

$\leftrightarrow a \sum_{i=0}^{n} x + b \sum_{i=0}^{n} x^2 = \sum_{i=0}^{n} yx$

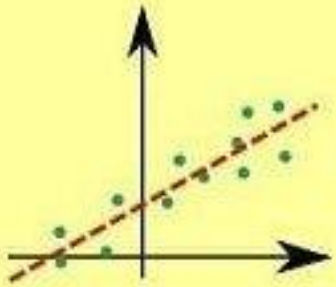
$$\sum_{i=0}^{n} y = na + b \sum_{i=0}^{n} x$$

$$a \sum_{i=0}^{n} x + b \sum_{i=0}^{n} x^2 = \sum_{i=0}^{n} yx$$

Resolver sistema linear

LINEAR REGRESSION

1. Econometric modelling
2. Marketing Mix Model
3. Customer Lifetime Value



Continuous ⇒ Continuous

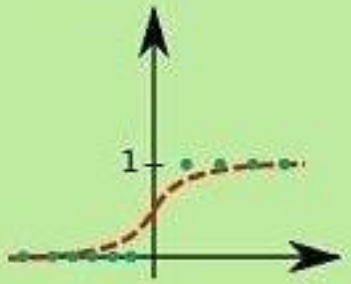
$$y = \alpha_0 + \sum_{i=1}^{N} \alpha_i x_i$$

lm(y ~ x1 + x2, data)

1 unit increase in x increases y by α

LOGISTIC REGRESSION

1. Customer Choice Model
2. Click-through Rate
3. Conversion Rate
4. Credit Scoring



Continuous ⇒ True/False

$$y = \frac{1}{1 + e^{-z}}$$

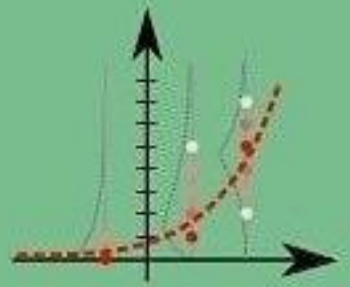
$$z = \alpha_0 + \sum_{i=1}^{N} \alpha_i x_i$$

glm(y ~ x1 + x2, data, family=binomial( ))

1 unit increase in x increases log odds by α

POISSON REGRESSION

1. Number of orders in lifetime
2. Number of visits per user



Continuous ⇒ 0,1,2,...

$$y \sim Poisson(\lambda)$$

$$ln\lambda = \alpha_0 + \sum_{i=1}^{N} \alpha_i x_i$$

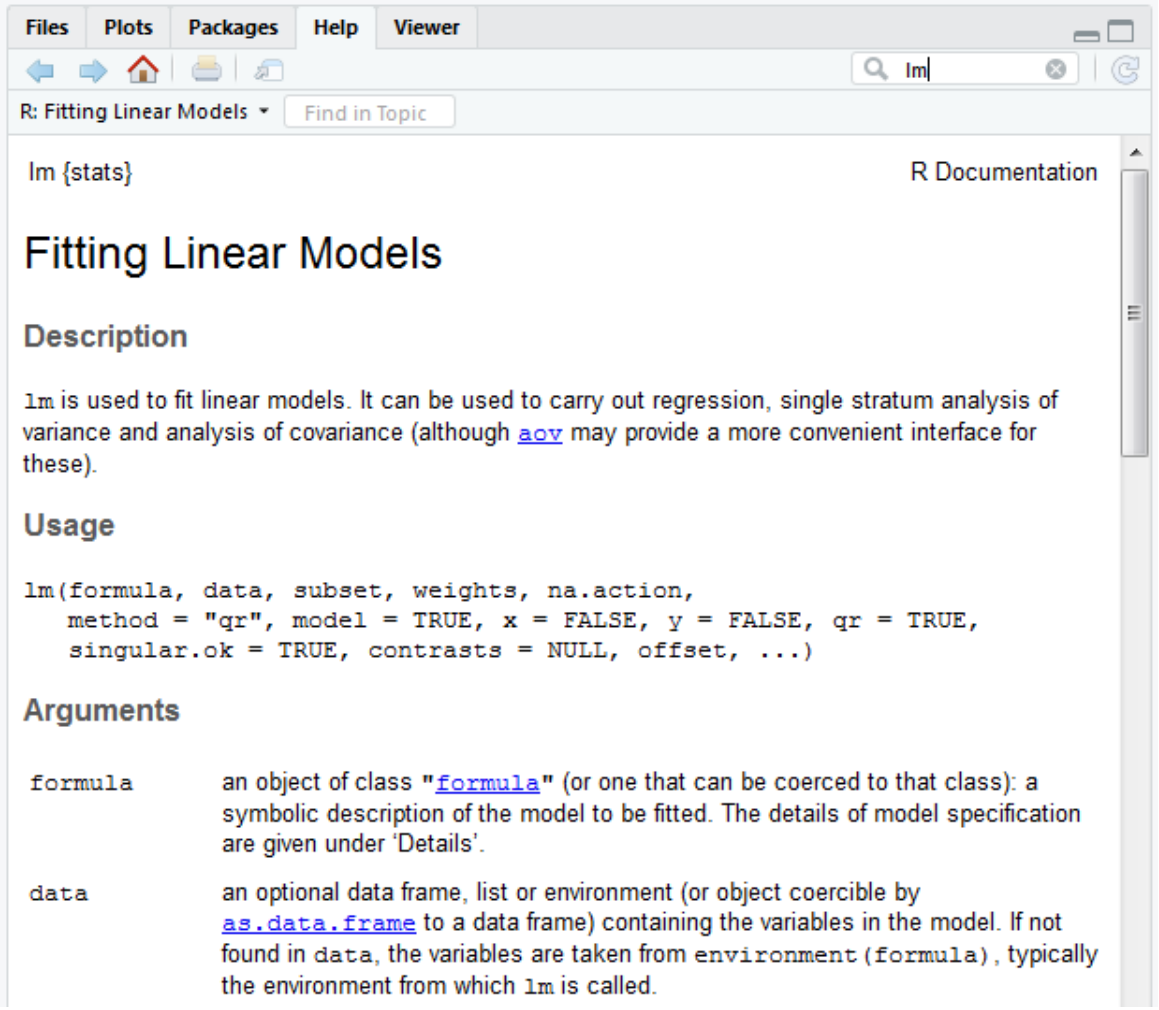
glm(y ~ x1 + x2, data, family=poisson( ))

1 unit increase in x multiplies y by e^{α}


DESVIO

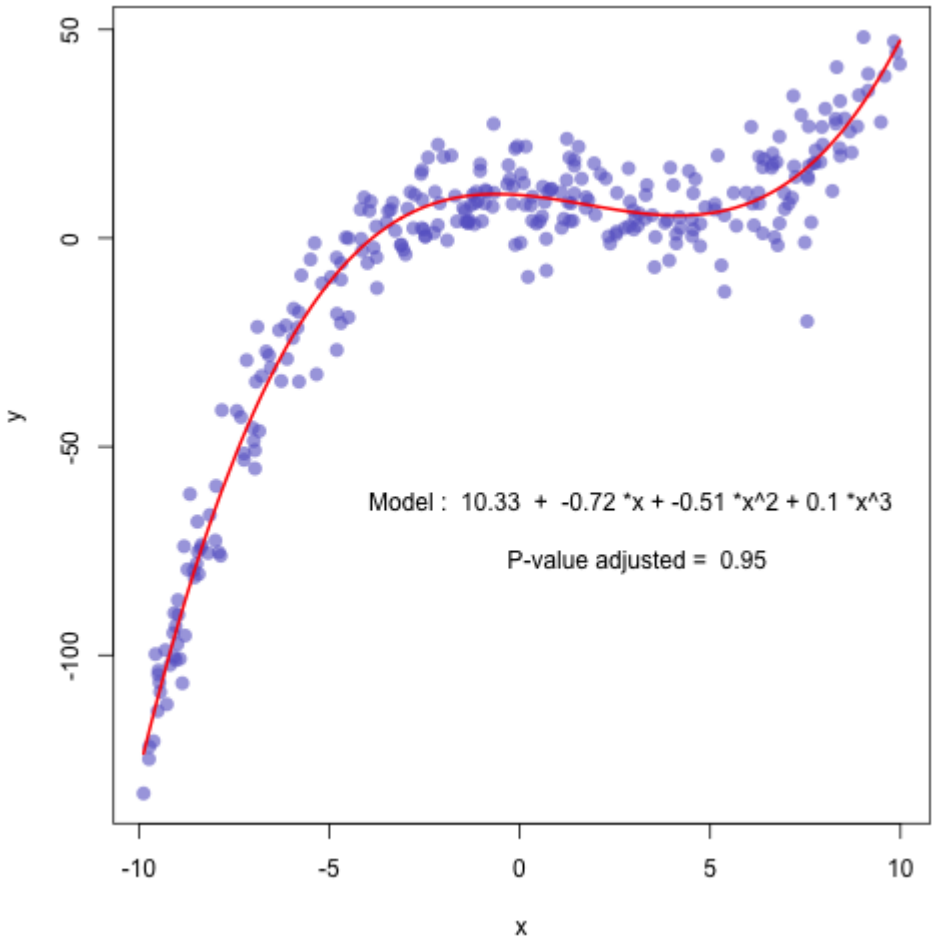
6.2 Fitting Models and Finding Model Values

The `lm` operator (short for "Linear Model") will translate a model design into fitted model values.



Usando o conhecimento de modelagem construa um modelo para o consumo de gás em função da temperatura média

Buscando inspiração na galeria de gráficos do R



http://www.r-graph-gallery.com/44-polynomial-curve-fitting/

http://www.r-graph-gallery.com/44-polynomial-curve-fitting/

```r
# We create 2 vectors x and y
x <- runif(300,  min=-10, max=10)
y <- 0.1*x^3 - 0.5 * x^2 - x + 10 + rnorm(length(x),0,8)

# plot of x and y :
plot(x,y,col=rgb(0.4,0.4,0.8,0.6),pch=16 , cex=1.3)

# Can we find a polynome that fit this function ?
model=lm(y ~ x + I(x^2) + I(x^3))

#I can get the features of this model :
summary(model)
model$coefficients
summary(model)$adj.r.squared

#For each value of x, I can get the value of y estimated by the model, and add it to the current plot !
myPredict <- predict( model )
ix <- sort(x,index.return=T)$ix
lines(x[ix], myPredict[ix], col=2, lwd=2 )
# I add the features of the model to the plot
coeff=round(model$coefficients , 2)
text(3,-70 , paste("Model : ",coeff[1] , " + " , coeff[2] , "*x"  , "+" , coeff[3] , "*x^2" , "+" , coeff[4] ,
```

Dissecando o script obtido

https://en.wikipedia.org/wiki/The_Anatomy_Lesson_of_Dr._Nicolaes_Tulp
Rembrandt, 1632

poly.R* ×

Source on Save | Run

```
1  # We create 2 vectors x and y
2
3  x <- runif(300,  min=-10, max=10)
4
5  y <- 0.1*x^3 - 0.5 * x^2 - x + 10 + rnorm(length(x),0,8)
6
```

Os dados do gráfico são produzidos artificialmente.

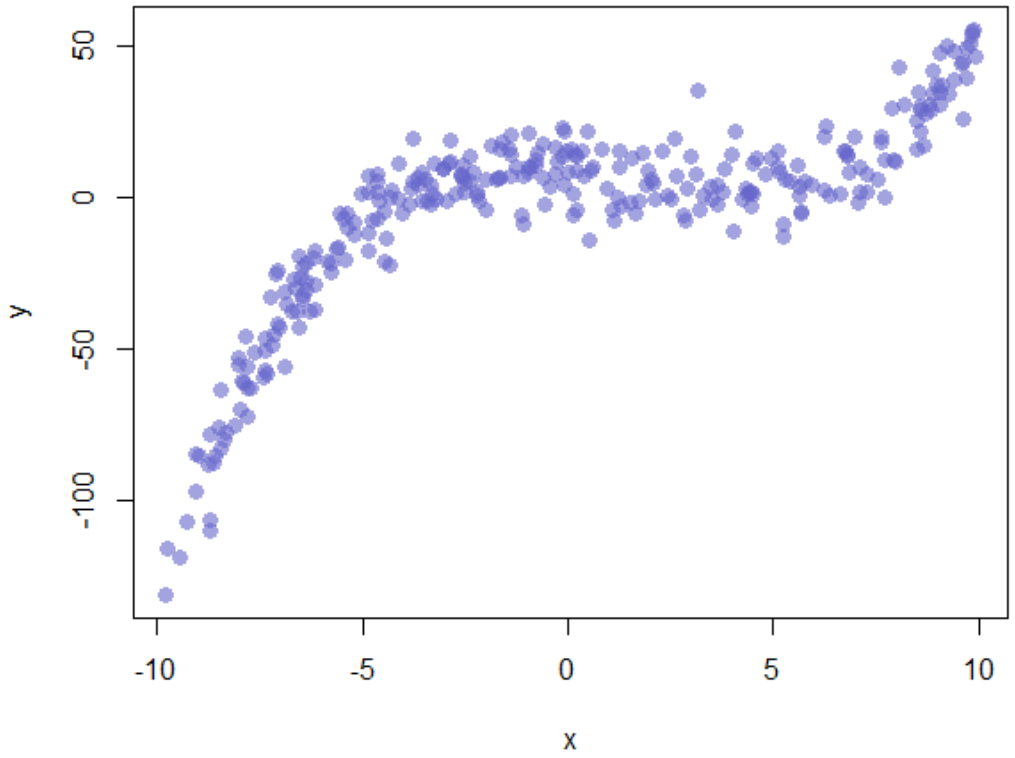
x é um vetor de 300 pontos aleatórios uniformente distribuídos no intervalo [-10 , 10]

y é construído através de uma perturbação aleatória normalmente distribuída de média 0 e desvio 8 em torno de um polinômio de grau 3

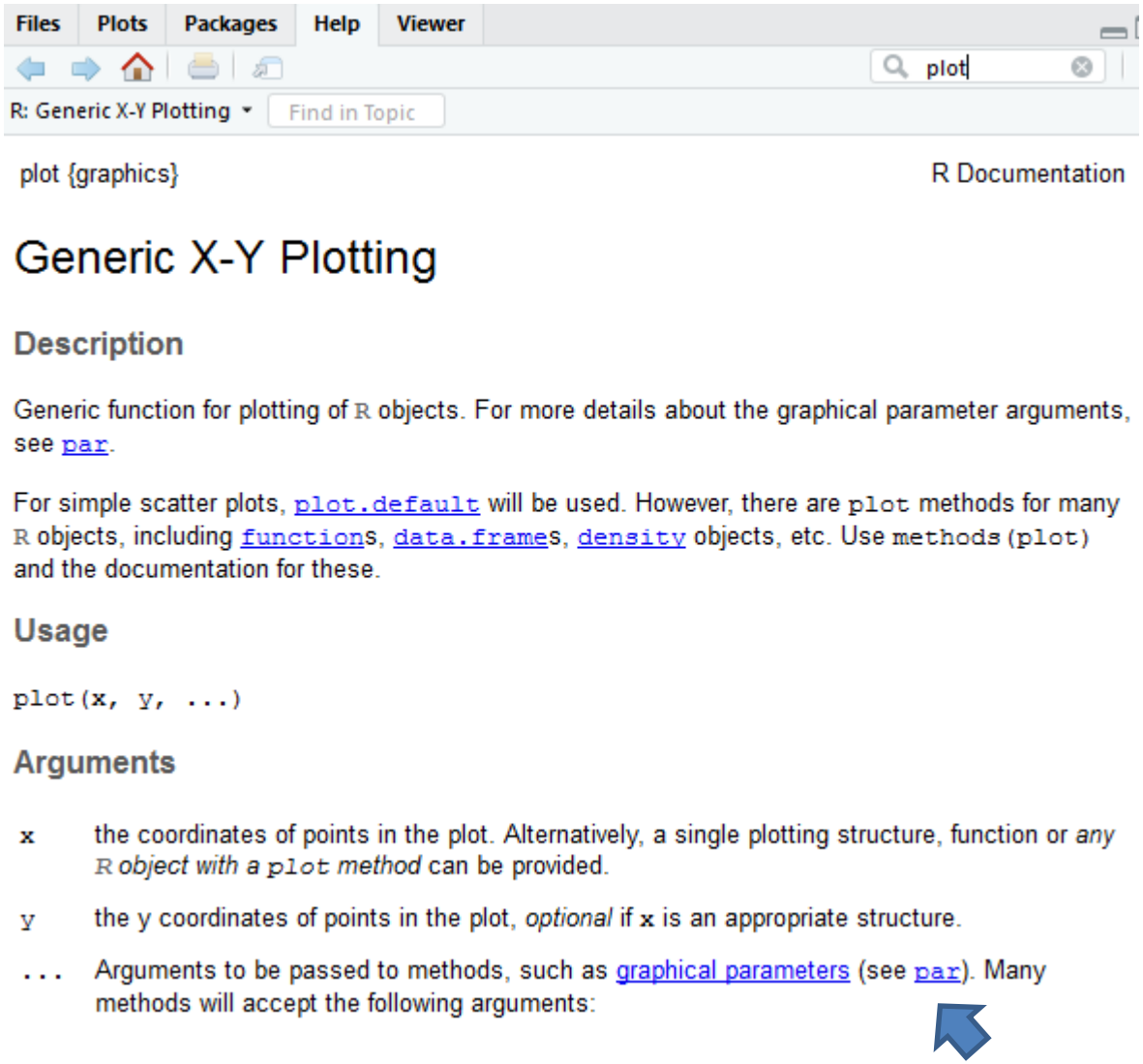
A produção de dados artificiais é um recurso útil para debugar scripts

```
6  # plot of x and y :
7
8  plot(x,y,col=rgb(0.4,0.4,0.8,0.6),pch=16 , cex=1.3)
9
```

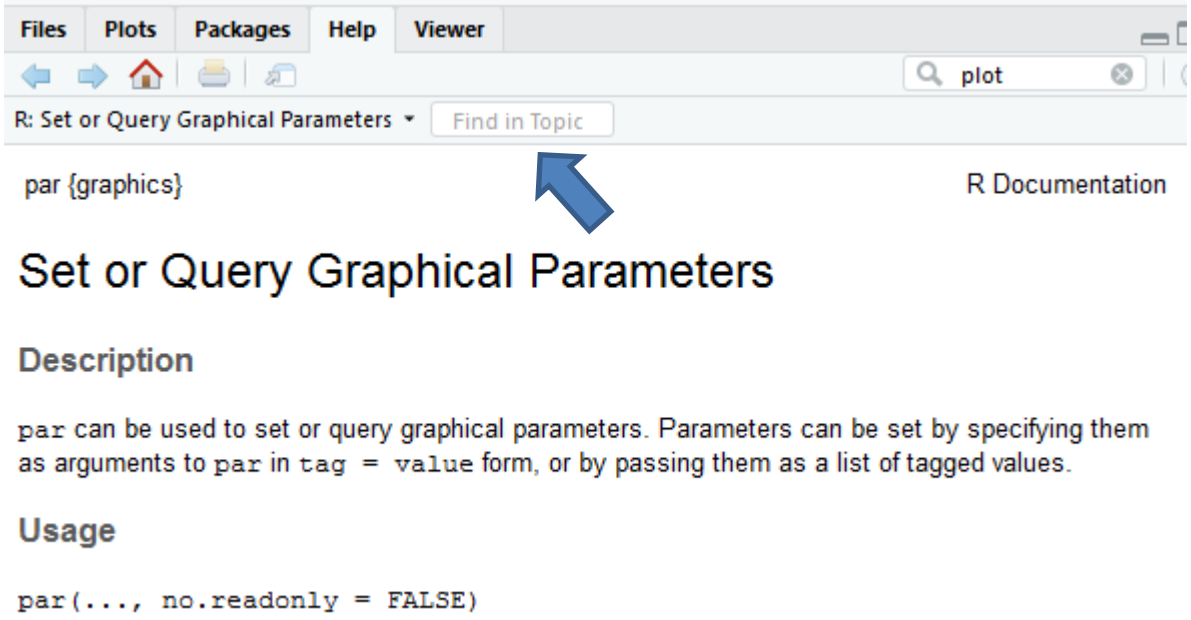
Executando esse comando isoladamente obtemos



Explorando as opções do comando...



No help do comando plot não encontrei os parâmetros apresentados, resolvi procurar no help de “par”



Digitando as opções na busca do tópico ...

Files | Plots | Packages | Help | Viewer

plot

R: Set or Query Graphical Parameters ▾ pch < >

page. This value may be FALSE if there are multiple figures on the page.

pch

Either an integer specifying a symbol or a single character to be used as the default in plotting points. See points for possible values and their interpretation. Note that only integers and single-character strings can be set as a graphics parameter (and not NA nor NULL).

Some functions such as points accept a vector of values which are recycled.

'pch' values

Values of pch are stored internally as integers. The interpretation is

- NA_integer_: no symbol.
- 0:18: S-compatible vector symbols.
- 19:25: further R vector symbols.
- 26:31: unused (and ignored).
- 32:127: ASCII characters.
- 128:255 native characters *only in a single-byte locale and for the symbol font*. (128:159 are only used on Windows.)
- -32 ... Unicode code point (where supported).

16 aparentemente representa o círculo cheio, testei outros valores obtendo outros símbolos (0 quadrado vazio, 1 círculo vazio, 2 triângulo vazio ...)

cex

A numerical value giving the amount by which plotting text and symbols should be magnified relative to the default. This starts as 1 when a device is opened, and is reset when the layout is changed, e.g. by setting mfrow.

Magnitude do símbolo em relação ao default

`col`

A specification for the default plotting color. See section 'Color Specification'.

Some functions such as `lines` and `text` accept a vector of values which are recycled and may be interpreted slightly differently.

colors {grDevices}

Color Names

Description

Returns the built-in color names which `R` knows about.

Color Specification

Colors can be specified in several different ways. The simplest way is with a character string giving the color name (e.g., `"red"`). A list of the possible colors can be obtained with the function `colors`. Alternatively, colors can be specified directly in terms of their RGB components with a string of the form `"#RRGGBB"` where each of the pairs `RR`, `GG`, `BB` consist of two hexadecimal digits giving a value in the range `00` to `FF`. Colors can also be specified by giving an index into a small table of colors, the `palette`: indices wrap round so with the default palette of size 8, `10` is the same as `2`. This provides compatibility with S. Index `0` corresponds to the background color. Note that the palette (apart from `0` which is per-device) is a per-session setting.

R: RGB Color Specification ▾ | Color Spe | < | >

Usage

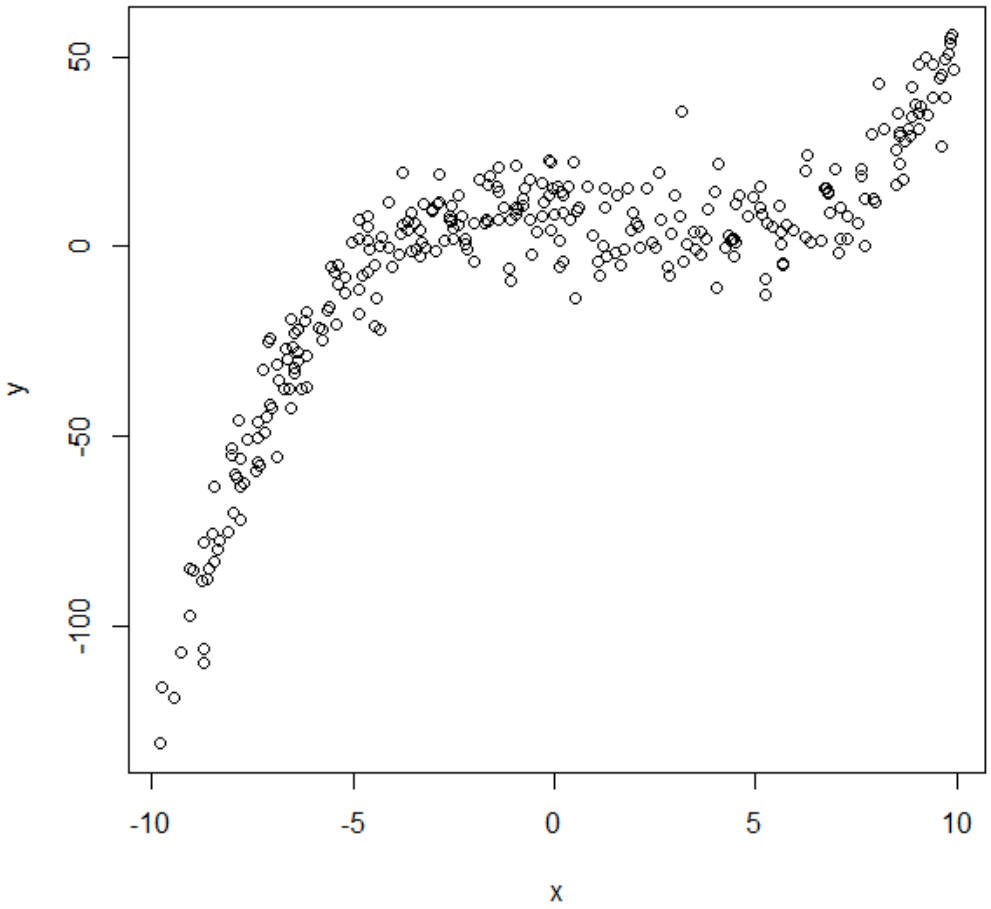
```
rgb(red, green, blue, alpha, names = NULL, maxColorValue = 1)
```

Arguments

`red, blue, green, alpha`	numeric vectors with values in *[0, M]* where *M* is `maxColorValue`. When this is `255`, the `red`, `blue`, `green`, and `alpha` values are coerced to integers in `0:255` and the result is computed most efficiently.
`names`	character vector. The names for the resulting vector.
`maxColorValue`	number giving the maximum of the color values range, see above.

A cor dos símbolos escolhidos é definida pelos valores de rgb

O que seria o “default” ?



```
10  # Can we find a polynome that fit this function ?
11
12  model=lm(y ~ x + I(x^2) + I(x^3))
```

A proposta de modelo é um polinômio de grau 3

```
14  #I can get the features of this model :
15  summary(model)
```

Analisando o sumário

Coeficientes do modelo
f(x)=a0 + a1*x + a2*x^2 + a3*x^3

```
> summary(model)

Call:
lm(formula = y ~ x + I(x^2) + I(x^3))

Residuals:
    Min      1Q  Median      3Q     Max
-23.686  -4.884  -0.340   5.443  32.287

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  9.577938   0.724493  13.220  < 2e-16 ***
x           -1.536756   0.217235  -7.074 1.09e-11 ***
I(x^2)      -0.492813   0.016889 -29.180  < 2e-16 ***
I(x^3)       0.107811   0.003437  31.366  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 8.318 on 296 degrees of freedom
Multiple R-squared:  0.9364,    Adjusted R-squared:  0.9358
F-statistic:  1454 on 3 and 296 DF,  p-value: < 2.2e-16
```

Qualidade de ajuste

Mathematical Representation of R-squared

R-squared is given by

$$R^2 = 1 - \frac{SSE}{SST}$$

Where SSE is the sum of squared errors of our regression model

$$SSE = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

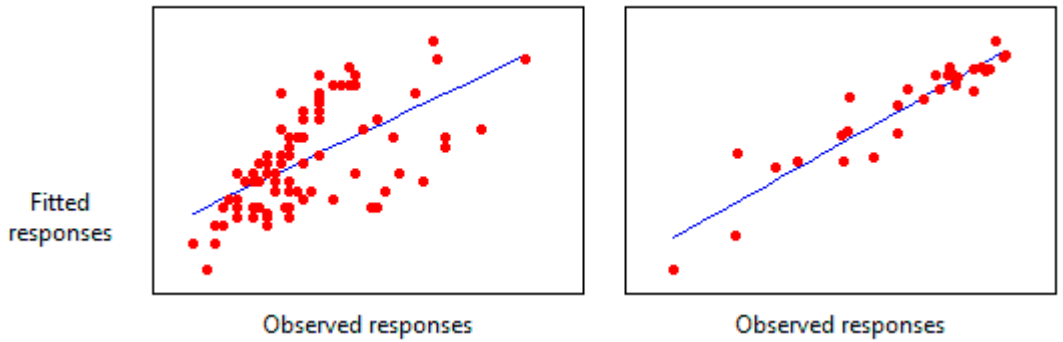
And SST is the sum of squared errors of our baseline model.

$$SST = \sum_{i=1}^{n} (y_i - \bar{y}_i)^2.$$

Dentre as diversas interpretações de R^2, uma delas é:

Qual a porcentagem da variância dos dados é capturada pelo modelo proposto

Plots of Observed Responses Versus Fitted Responses for Two Regression Models



R^2 = 0,380

R^2 = 0,874


DESVIO

```
16  model$coefficients
17  summary(model)$adj.r.squared
```

Executando esses comandos isoladamente obtemos

```
> model$coefficients
(Intercept)           x      I(x^2)      I(x^3)
  9.5779375  -1.5367565  -0.4928132   0.1078108
> summary(model)$adj.r.squared
[1] 0.9357924
```

vetor dos coeficientes

valor de R^2

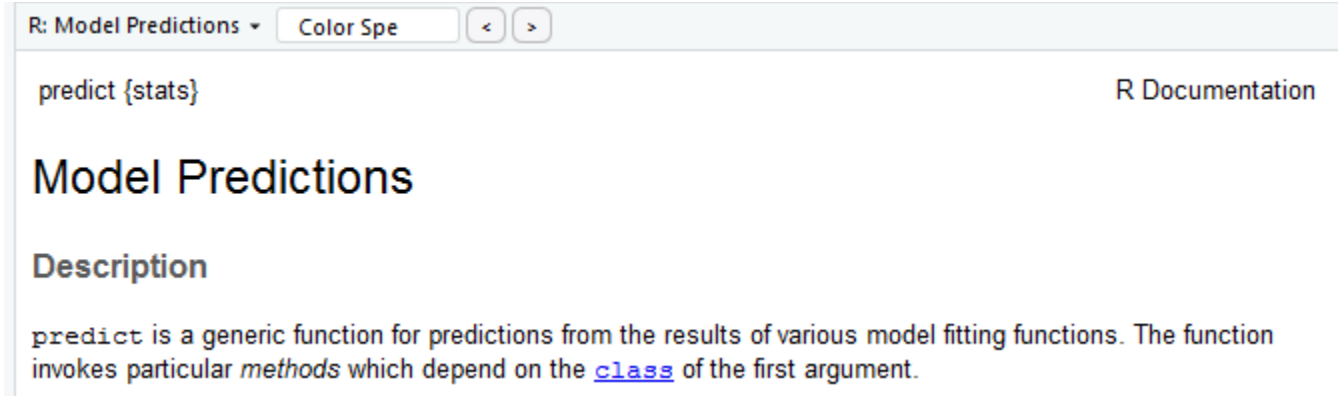
Lembrando que os dados foram gerados artificialmente

```
4  y <- 0.1*x^3 - 0.5 * x^2 - x + 10 + rnorm(length(x),0,8)
```

Percebam que os valores dos coeficientes do modelo se aproximam dos coeficientes usados na geração dos dados, conforme esperado

O que falta é incluir o modelo no gráfico

```
19  #For each value of x, I can get the value of y estimated by the model, and add it to the current plot !
20  myPredict <- predict( model )
```



A função “predict” permite avaliar a função do modelo associando a um objeto

O vetor x foi gerado numa distribuição aleatória uniforme no intervalo [-10, 10]

```
> x
  [1] -2.865461758 -1.370526189 -7.035768786 -9.738448490  4.311321322 -7.936315285 -1.074313028  2.802020903
  [9]  9.836772401 -0.088128443 -0.313009513 -6.531153303  5.096418890 -0.922090216  0.223395675 -5.849097734
 [17] -5.426837145  1.914239926  1.497443966 -8.458712394 -9.289188408  2.855909844  8.572303993  1.961848447
 [25]  1.218014960  0.520554478  9.701904478  0.152836447  3.655761573  2.030824353 -5.222626445 -4.836681467
 [33]  4.586192467 -0.948583372 -6.497464632  4.933965392 -7.900247192  7.290898981  2.292899434  1.143190777
 [41] -3.424453619 -0.937371091  0.008819452 -6.382672777  0.592612056 -8.494485086 -4.444881347 -5.746009615
 [49] -4.304190380  7.901882059 -1.075293534  5.599697796  7.612380697 -1.737515810 -8.723830390 -3.290250166
 [57]  4.474518932 -3.247693330  2.608282450  6.812291080  7.122633294 -2.172814379 -2.390122288  7.908908520
 [65]  2.886315258  4.821572974  2.106068931  8.061632230 -4.125396898 -6.174797802  7.729018866  0.066789715
 [73]  7.541150860 -6.216127551  5.162061048  4.489977853  8.874496366  0.952931740  4.234877354 -2.221898003
```

Esse vetor não é ordenado !

O comando sort ordena vetores, e o parâmetro “ix” é um vetor com o índice da ordem

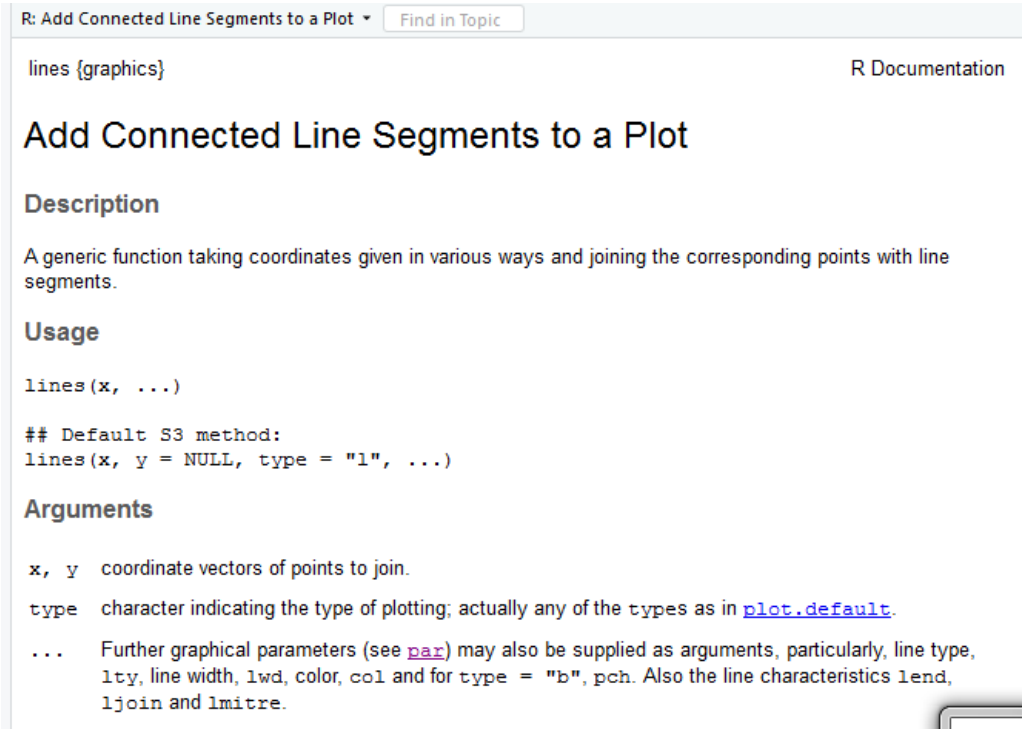
```
> ix <- sort(x,index.return=T)$ix
> ix
  [1] 297   4 169  21 116 173 187 159 240  55 155 197 255  46  20 223 246 205 190 185 100  81   6  37 144 160
 [27] 249  85 277 186  96 180 120 177 136 174 199 132 280 279 288   3 176 283 226 154 110 162 158 236  12  35
 [53] 195 230 267  44  93 166 290  74  70 164 200  16 134  48  90 178 263 179  17 262 273  31 274 145  97 165
 [79] 115  32 109 203  89 217  92 234 252  47 151 244  83  49 220  69 163 129 104 286 210 150  87 175 111 182
[105]  41 250 265  56  58 108 119 284 287 206   1 300 224 237 295 147 172 121 193  63 161 232  80 123  62 221
[131] 198 211  54 133 117 242 264 135 127 269   2 259 225  51   7  34 128  42  14 137 192 258 235 141 248 167
[157]  11 143 239 191 268 204  10 146  43  72 183 282  28 112 114  15  91  94 149  26 170  45 231  78 126  40
```

```
24  lines(x[ix], myPredict[ix], col=2, lwd=2 )
```

O comando “lines” adiciona segmentos de reta entre pontos dados (ou seja, uma poligonal) Como o número de pontos é alto não se percebe no gráfico

ix é o índice que ordena a sequência de pontos

O vetor x[ix] é uma componente do vetor x associada ao índice ix, myPredict[ix] é a componente correspondente de saída do modelo



R: Add Connected Line Segments to a Plot — Find in Topic

lines {graphics} — R Documentation

Add Connected Line Segments to a Plot

Description

A generic function taking coordinates given in various ways and joining the corresponding points with line segments.

Usage

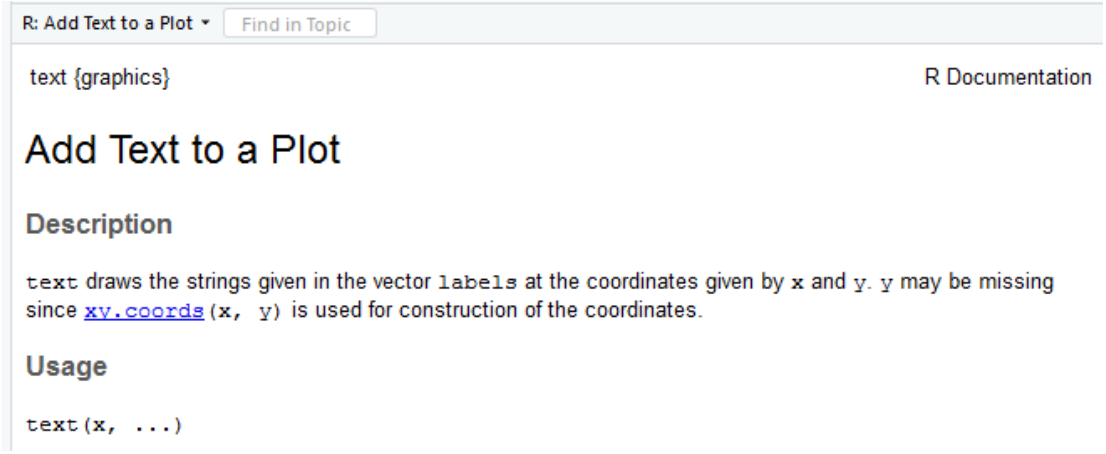
```
lines(x, ...)

## Default S3 method:
lines(x, y = NULL, type = "l", ...)
```

Arguments

x, y	coordinate vectors of points to join.
type	character indicating the type of plotting; actually any of the types as in plot.default.
...	Further graphical parameters (see par) may also be supplied as arguments, particularly, line type, lty, line width, lwd, color, col and for type = "b", pch. Also the line characteristics lend, ljoin and lmitre.

“col” define a cor da linha, e “lwd” é o fator que especifica a espessura da linha

Resta incluir o texto que descreve o modelo no gráfico



R: Add Text to a Plot — Find in Topic

text {graphics} — R Documentation

Add Text to a Plot

Description

`text` draws the strings given in the vector `labels` at the coordinates given by `x` and `y`. `y` may be missing since `xy.coords(x, y)` is used for construction of the coordinates.

Usage

```
text(x, ...)
```

arredonda os coeficientes para 2 casas decimais

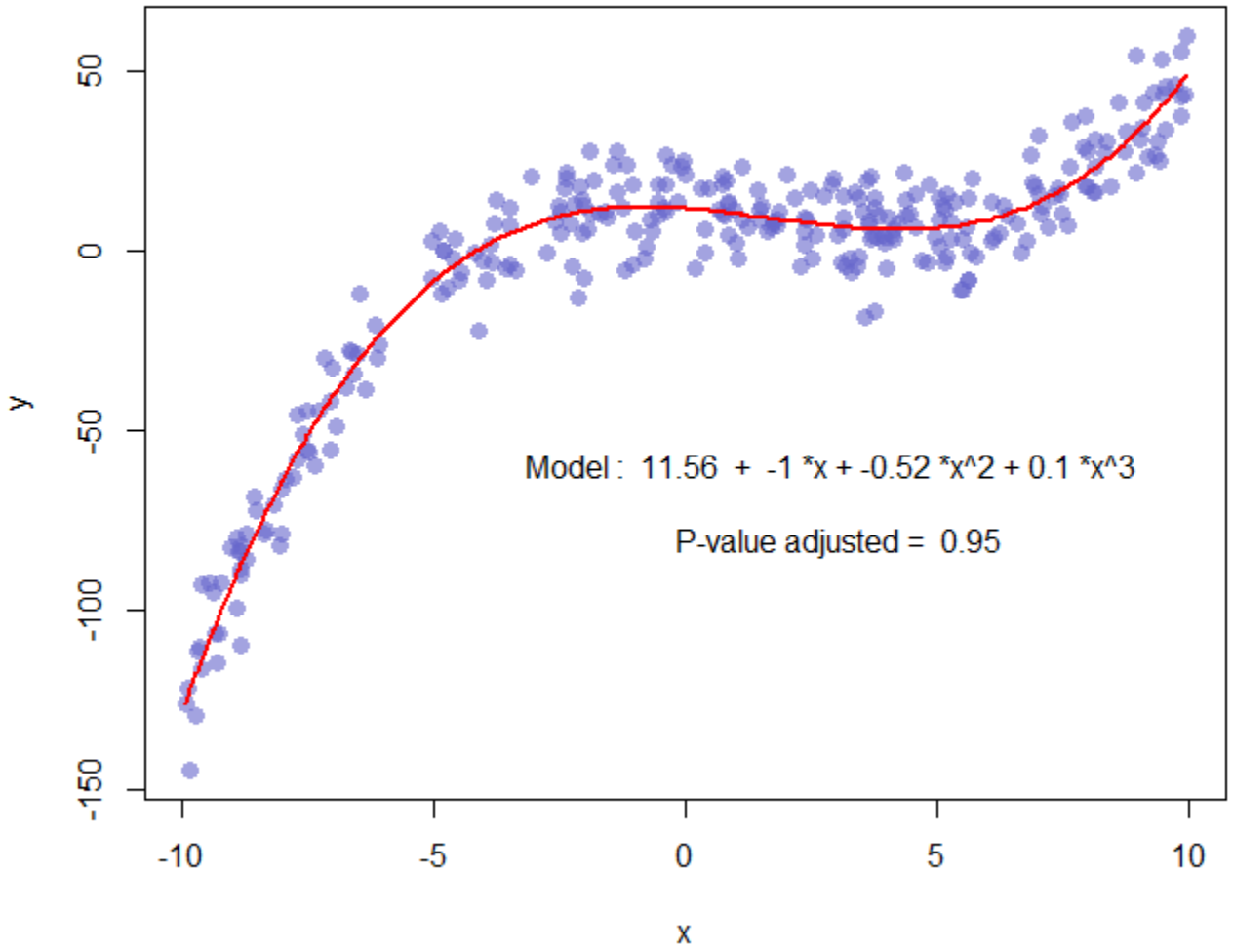
pula linha (2x)

```
26  # I add the features of the model to the plot
27  coeff=round(model$coefficients , 2)
28
29  text(3,-70 , paste("Model : ",coeff[1] , " + " , coeff[2] , "*x"  ,
30                     "+" , coeff[3] , "*x^2" , "+" , coeff[4] , "*x^3" , "\n\n" ,
31                     "P-value adjusted = ",round(summary(model)$adj.r.squared,2)))
```

Define o conteúdo a ser escrito

posiciona o texto tendo como referência a escala do gráfico

Finalmente



Porque os valores mudaram se eu rodei o mesmo script ?
O que eu preciso fazer para que não mudem ?

Usando o conhecimento de modelagem construa um modelo para o consumo de gás em função da temperatura média

Adaptando o script para os dados desejados: ccf x temp

poly_ccf.R

Source on Save | Run | Source

```
1
2  # carregando o pacote mosaic
3  require(mosaic)
4
5  # carregando os dados
6  Utils <- Utilities
7
8  x <- Utils$temp
9  y <- Utils$ccf
10
11 # plot of x and y :
12
13 plot(x,y,col=rgb(0.4,0.4,0.8,0.6),pch=16 , cex=1.3,
14      xlab = "Temperature (deg F)",
15      ylab = "Natural Gas Usage (ccf)")
16
17 # Can we find a polynome that fit this function ?
18
19 model=lm(y ~ x + I(x^2) + I(x^3))
20
21 #I can get the features of this model :
22 summary(model)
23 model$coefficients
24 summary(model)$adj.r.squared
25
26 #For each value of x, I can get the value of y estimated by the model,
27 # and add it to the current plot !
28 myPredict <- predict( model )
29
30 ix <- sort(x,index.return=T)$ix
31
32 lines(x[ix], myPredict[ix], col=2, lwd=2 )
33
34 # I add the features of the model to the plot
35 coeff=round(model$coefficients , 3)
36
37 text(53,232 , paste("Model : ",coeff[1] , " + " , coeff[2] , "*x"   ,
38                     "+" , coeff[3] , "*x^2" , "+" , coeff[4] , "*x^3" , "\n" ,
39                     "R^2 = ",round(summary(model)$adj.r.squared,2)))
```

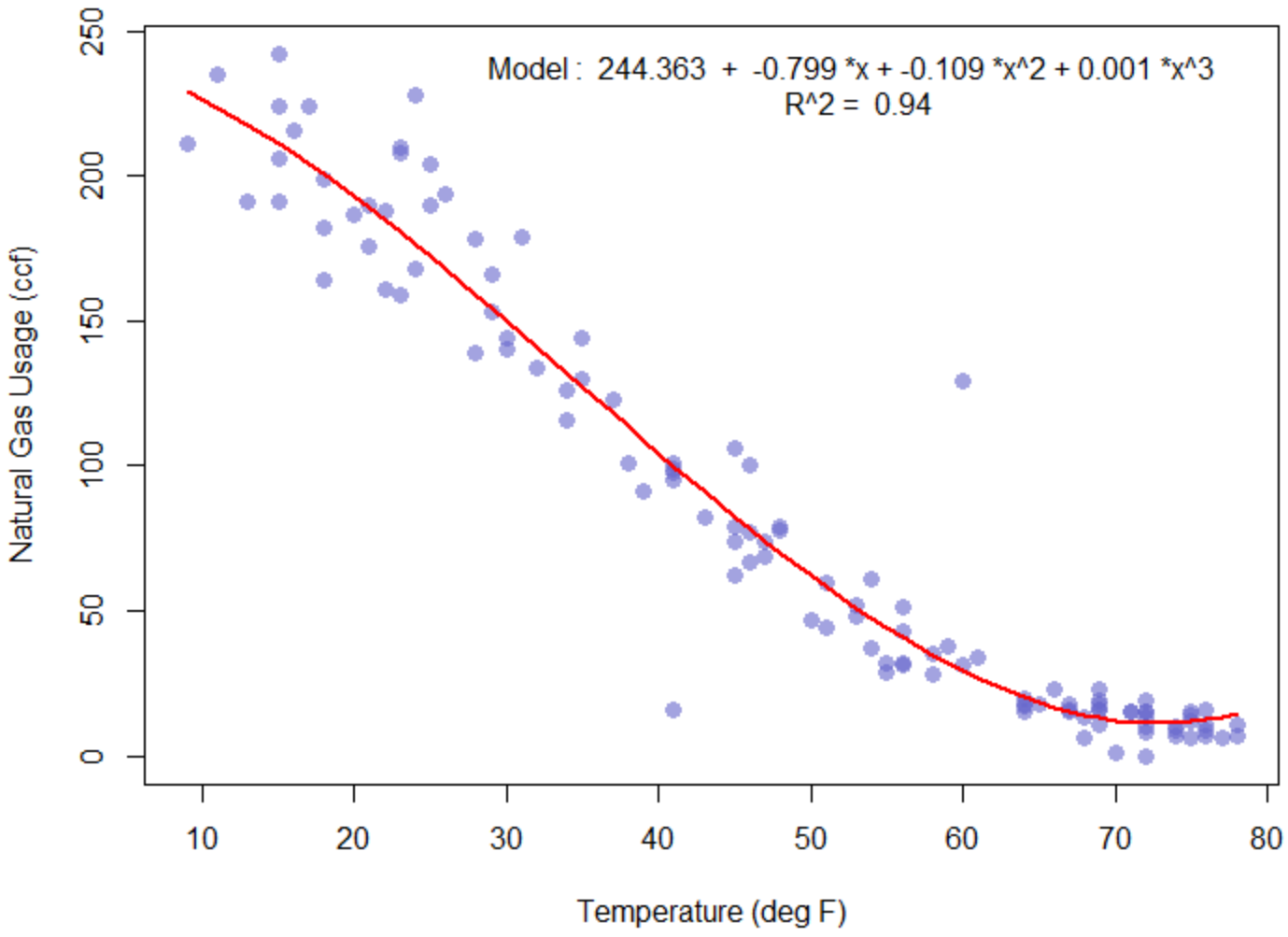
Carregando o pacote mosaic para usar os dados de Utilities

Associando as variáveis de interesse as variáveis do script anterior

incluindo os labels das variáveis

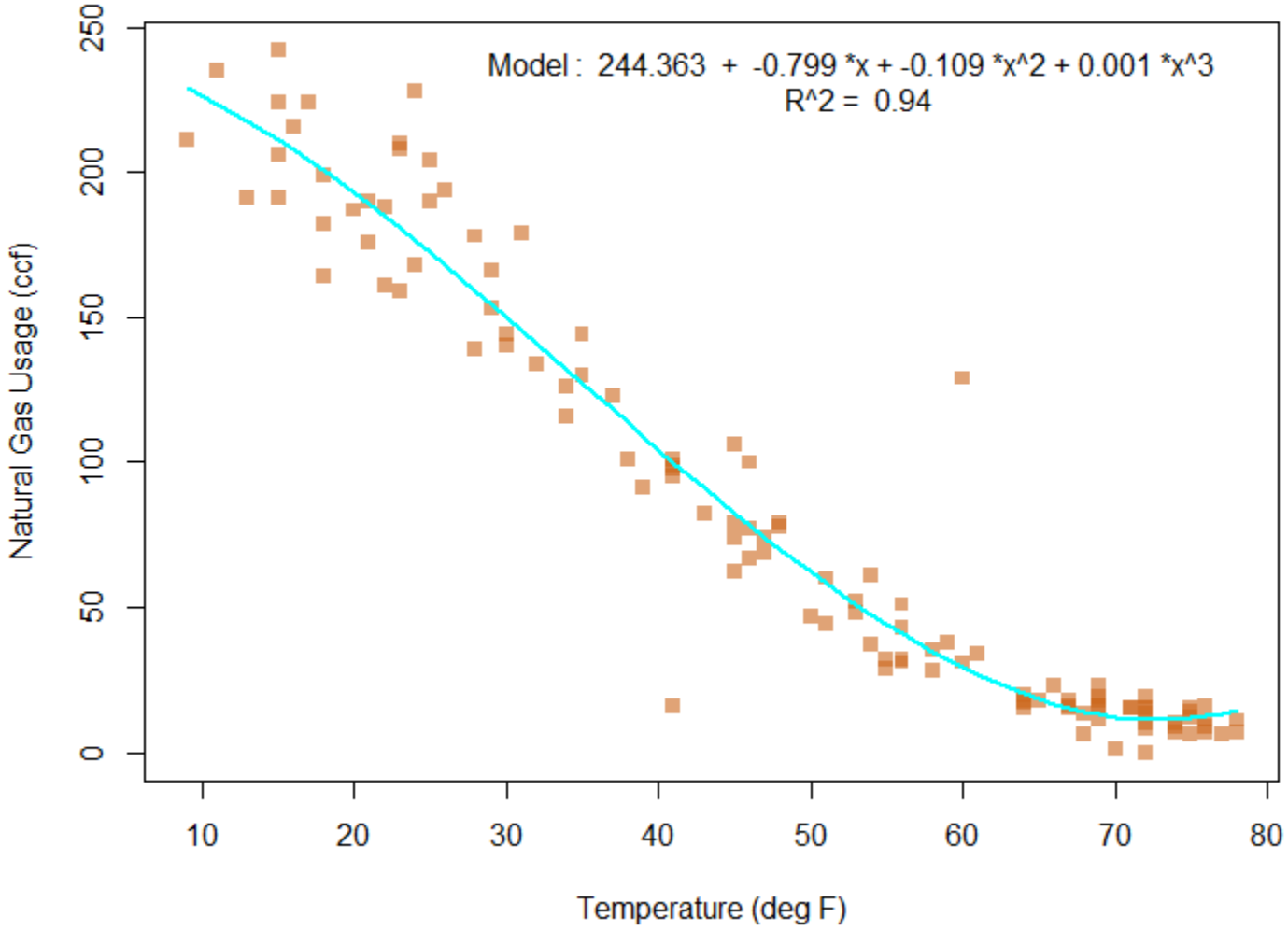
Aumentando os significativos

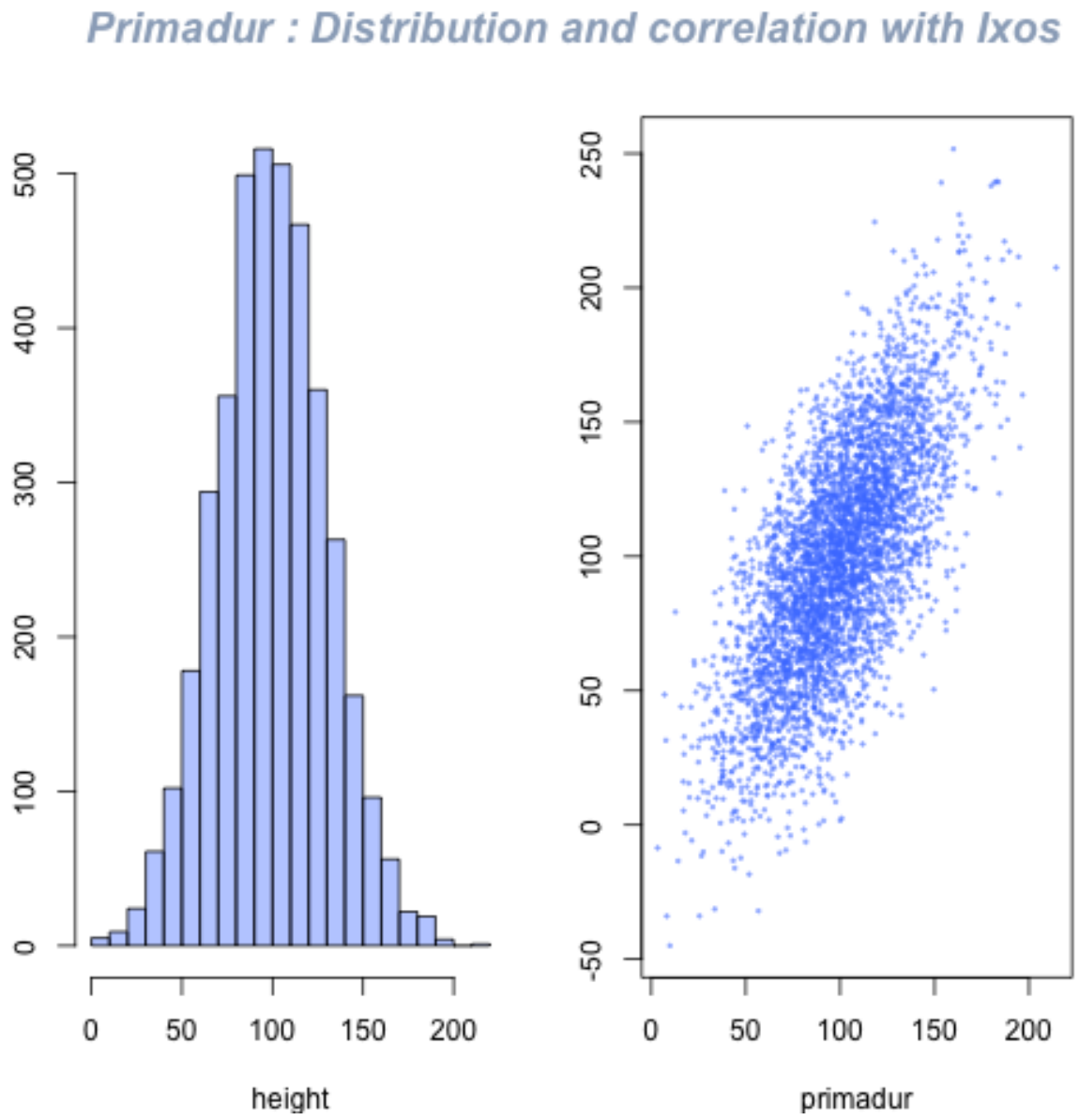
Posicionando o texto por tentativa e erro


Model : 244.363 + -0.799 *x + -0.109 *x^2 + 0.001 *x^3
R^2 = 0.94
Natural Gas Usage (ccf)
0
50
100
150
200
250
10
20
30
40
50
60
70
80
Temperature (deg F)

Mudando as linhas de estilo obtém-se outras possibilidades visuais

```
plot(x,y,col=rgb(0.8,0.4,0.1,0.6),pch=15 , cex=1.1,
lines(x[ix], myPredict[ix], col=5, lwd=2 )
```





http://www.r-graph-gallery.com/1-one-title-for-2-graphs/