INFOX
educación sin fronteras

Clase 1: Introducción al programa

Mg. Gloria Rivas

Agenda

1. **Visualización de datos**

2. **Reportes en látex usando Rmarkdown**

INFOX
educación sin fronteras

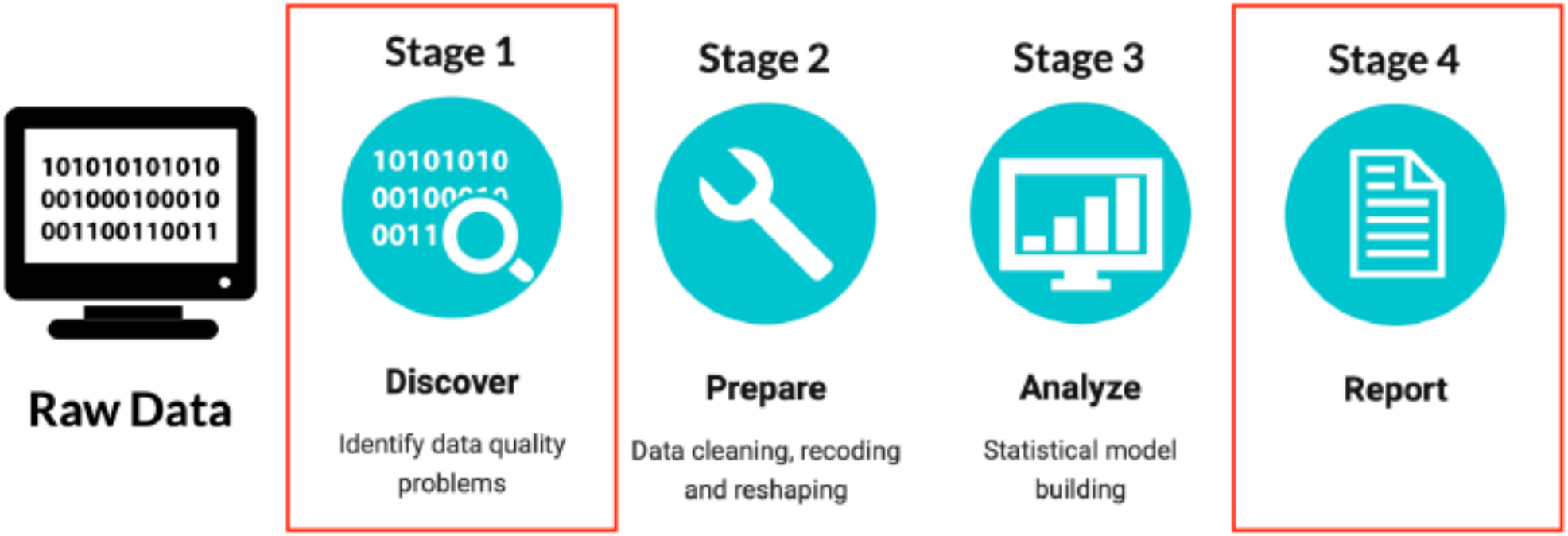
Agenda

1. Visualización de datos

INFOX
educación sin fronteras

El proceso de Data Science

The data science process



Data visualisation is an important tool for generating and communicating insights.

El proceso de Data Science

Exploratory graphs:

- Describing data, detecting patterns and trends.
- Produced instantly using default settings.

Illustrative graphs:

- Illustrating a conclusion, making a convincing argument.
- Require changing graph titles, axis, labels, colors, symbols, adding legends, ... or adding information.

R includes at least three graphical systems: (1) the **standard graphics** package, (2) the **lattice** package for Trellis graphs and (3) the **ggplot2** package based on the idea of the grammar-of-graphics.

In R graphs are build-up in two stages by successively calling graph functions:

(1) Creating an (**exploratory**) default graph.
(2) **Customizing** and **annotating** the default graph.

Exploratory graphs

Exploratory graphs

Example: Bike sharing Chicago.

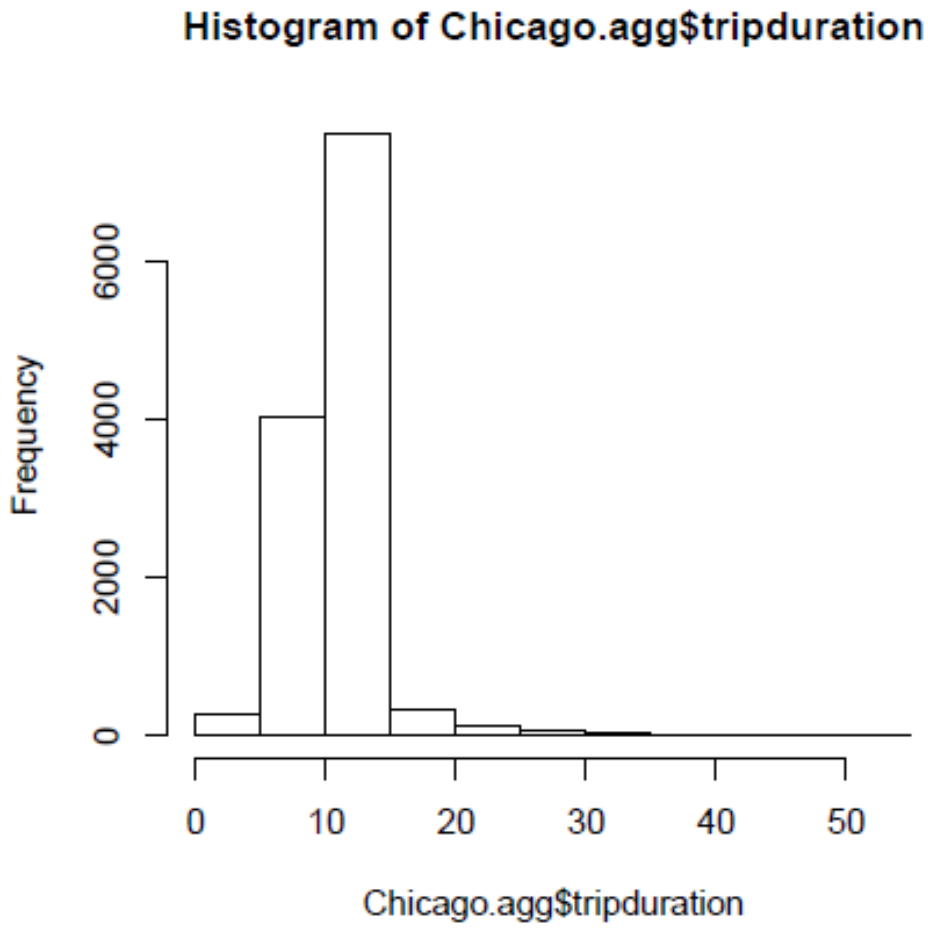
www.divvybikes.com covers information about 9.5 Million bike trips in Chicago. The data set is available in an aggregated version `Chicago.agg.csv` on Canvas.

```
> str(Chicago.agg)
'data.frame':   12433 obs. of  8 variables:
 $ X           : int  1 2 3 4 5 6 7 8 9 10 ...
 $ week        : int  1 1 1 1 1 1 1 1 1 1 ...
 $ usertype    : Factor w/ 3 levels "Customer","Dependent",..: 3 3 3 3 3 ..
 $ gender      : Factor w/ 2 levels "Female","Male": 1 1 1 1 1 1 1 1 1 1 ..
 $ events      : Factor w/ 6 levels "clear","cloudy",..: 1 1 1 1 1 1 1 1 ..
 $ temperature : num  -4 0 1 1.9 3 3.9 5 6.1 7 8.1 ...
 $ tripduration: num  4.22 8.58 5.47 10.4 8.01 ...
 $ no.customers: int  1 14 2 21 9 46 4 2 3 3 ...
```

Exploratory graphs

(Default) histogramm:

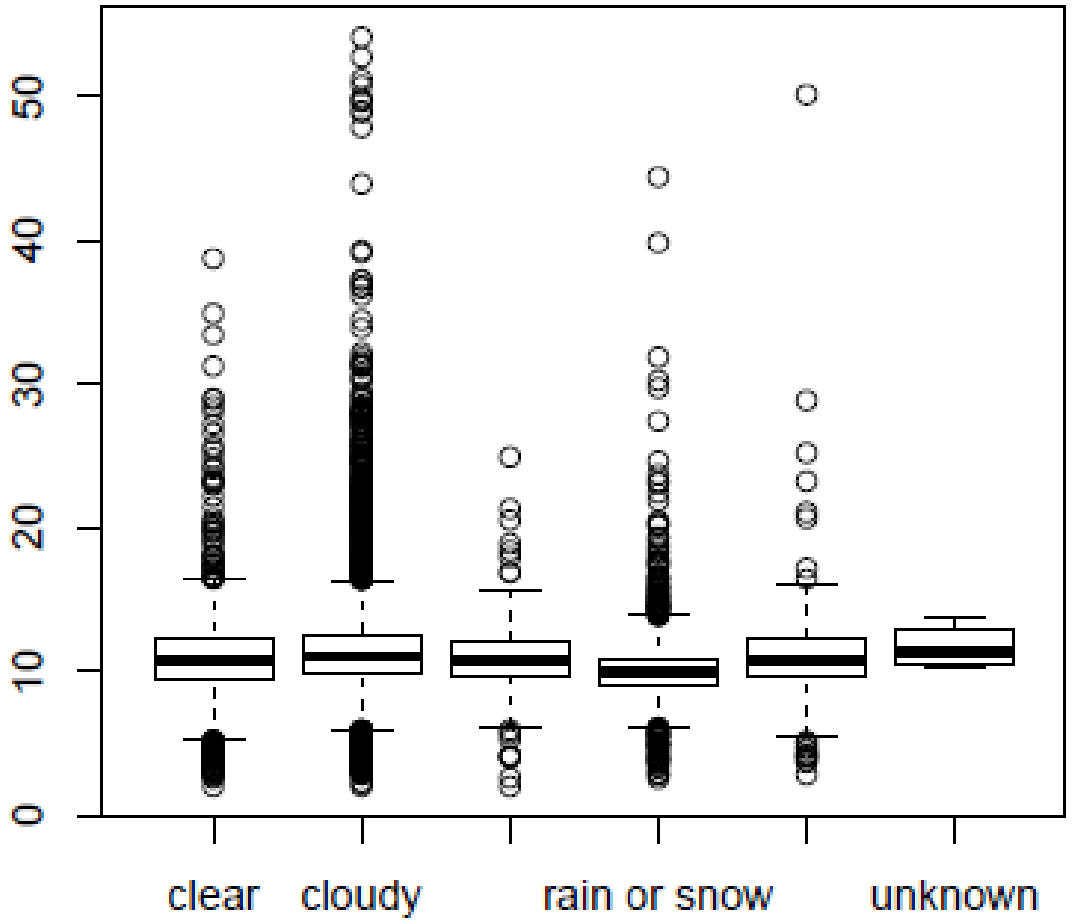
```
> hist(Chicago.agg$tripduration)
```



Exploratory graphs

(Default) boxplot:

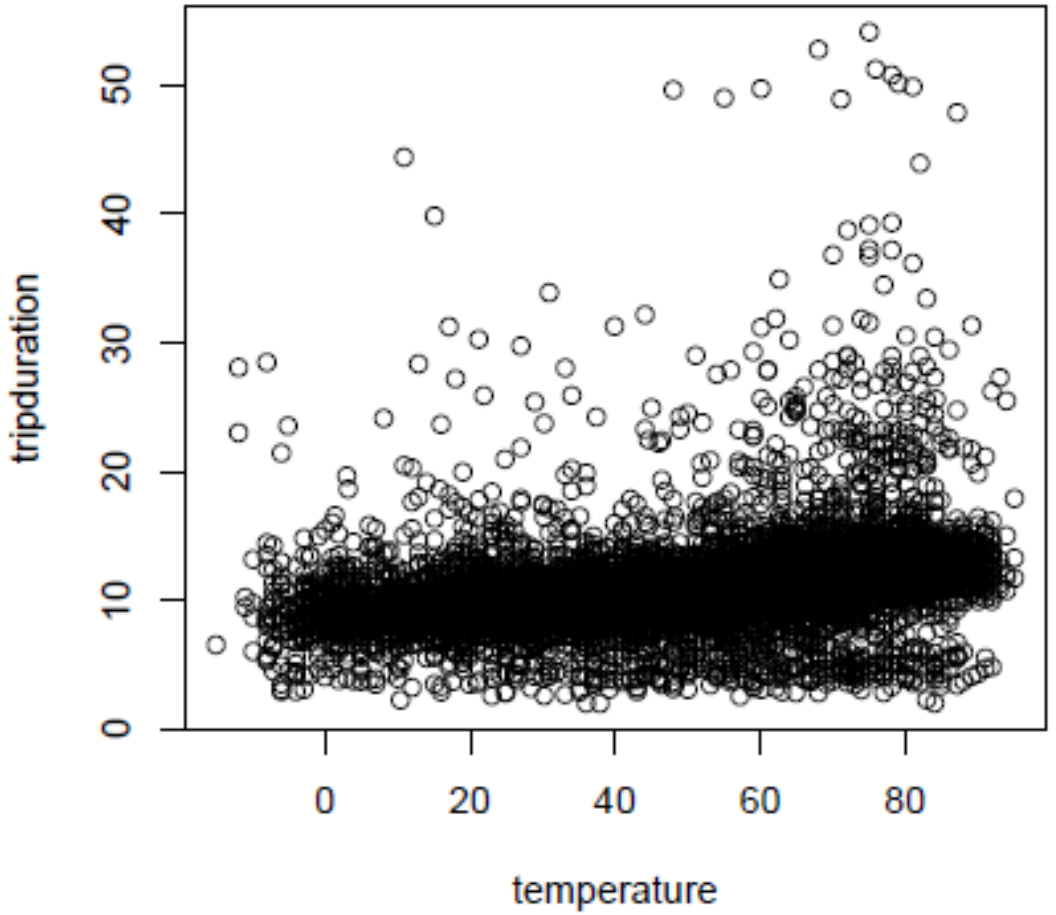
```
> boxplot(tripduration ~ events, data = Chicago.agg)
```



Exploratory graphs

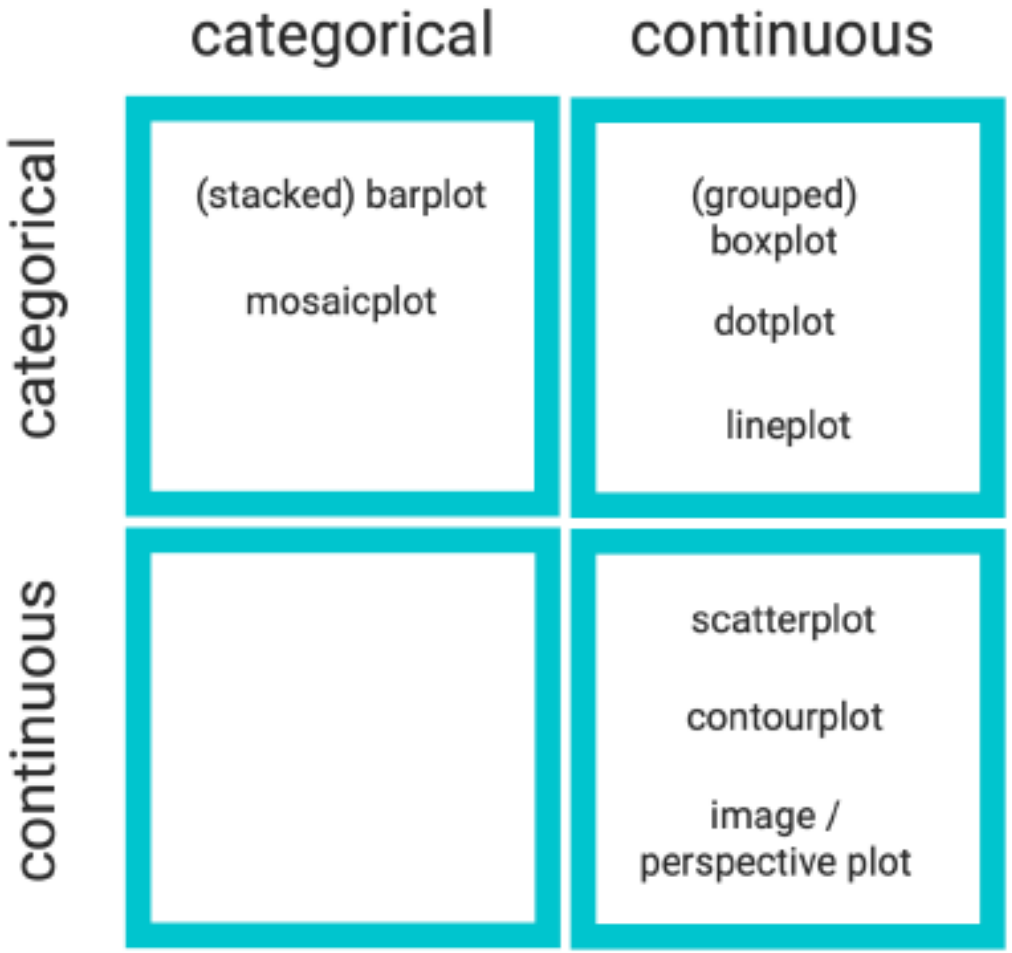
(Default) scatterplot:

```
> plot(tripduration ~ temperature, data = Chicago.agg)
```



Exploratory graphs

How to select among different graph types?



Illustrative graphs

Illustrative graphs

The final graph should help others to quickly built up a good mental model of the data and the business problem at hand. Therefore, it is necessary to change different aspects of the appearance of a plot (**customizations**) as well as to add extra information (**annotations**).

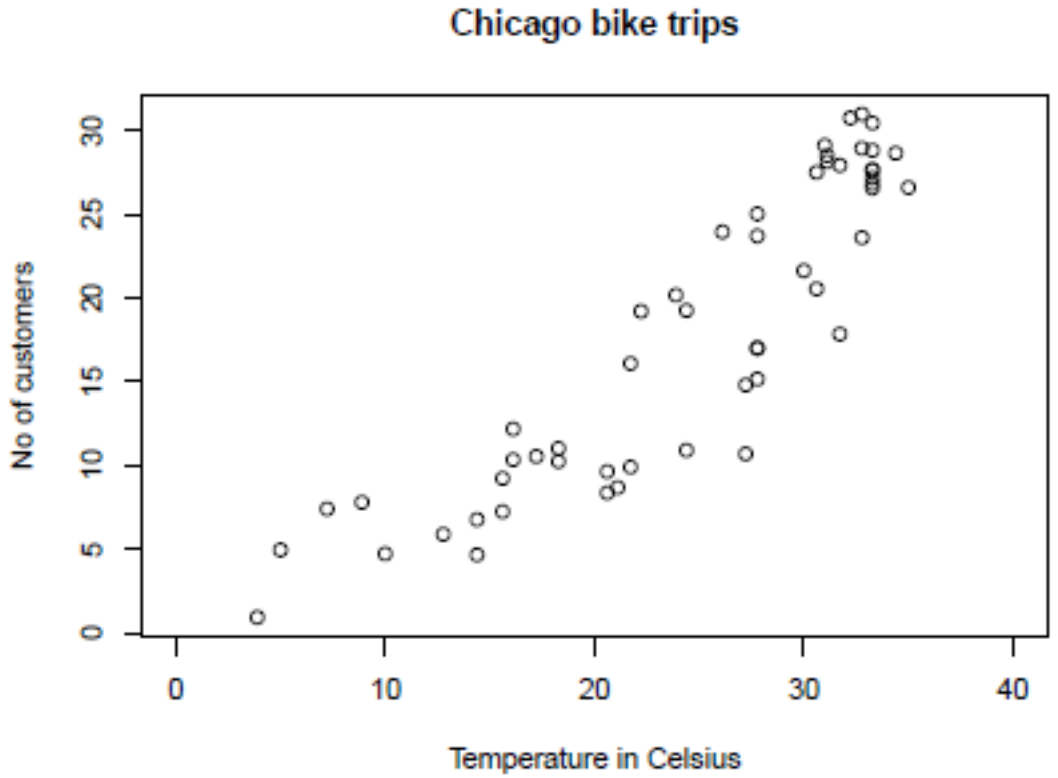
Customizations

- Change the **title**, **axis labels**, and **coordinate system** with main, xlim and xlab / ylab:

```
> # (as usual) perform some data preparation tasks fist:
> # 1. transform fahrenheit to celsius
> Chicago.agg$temperature <- (Chicago.agg$temperature - 32) * (5/9)
> # 2. reshape the data to weekly observations
> temp <- aggregate(temperature ~ week, FUN = max, data = Chicago.agg)
> customers <- aggregate(no.customers ~ week, FUN = sum, data = Chicago.agg)
> data <- data.frame("week" = temp$week, "temperature" = temp$temperature,
+                    "customers" = customers$no.customers/10000)
```

Illustrative graphs

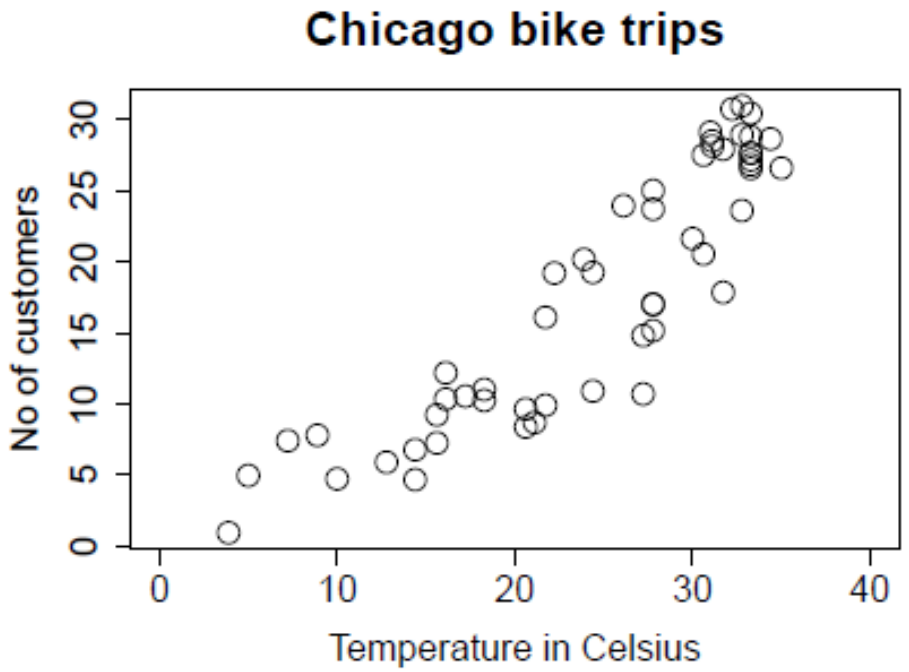
```
> plot(customers ~ temperature, main = "Chicago bike trips",
+      xlim = c(0, 40), xlab = "Temperature in Celsius", ylab = "No of customers",
+      data = data)
```



Illustrative graphs

- Change the **size** of data symbols, title and axis labels with cex:

```
> plot(customers ~ temperature, main = "Chicago bike trips",
+      xlim = c(0, 40), xlab = "Temperature in Celsius", ylab = "No of customers",
+      cex = 2, cex.main = 2, cex.lab = 1.5, cex.axis = 1.5,
+      data = data)
```

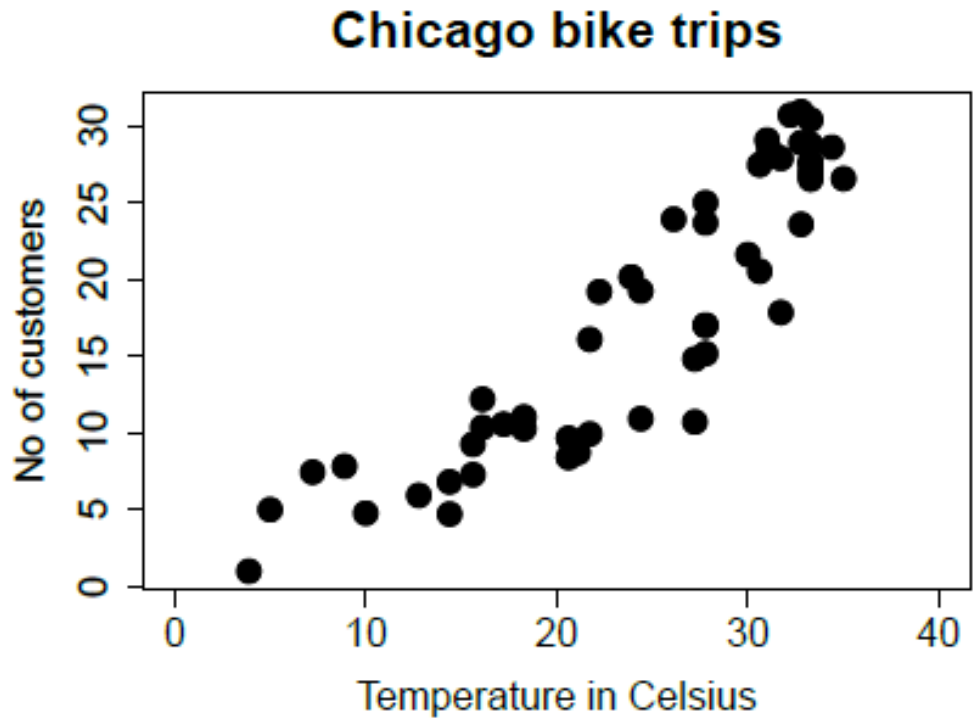


The size of text is specified in "points". The cex argument controls the font size by specifying a multiplicate modifier (**character expansion** factor = `fontsize * cex`).

Illustrative graphs

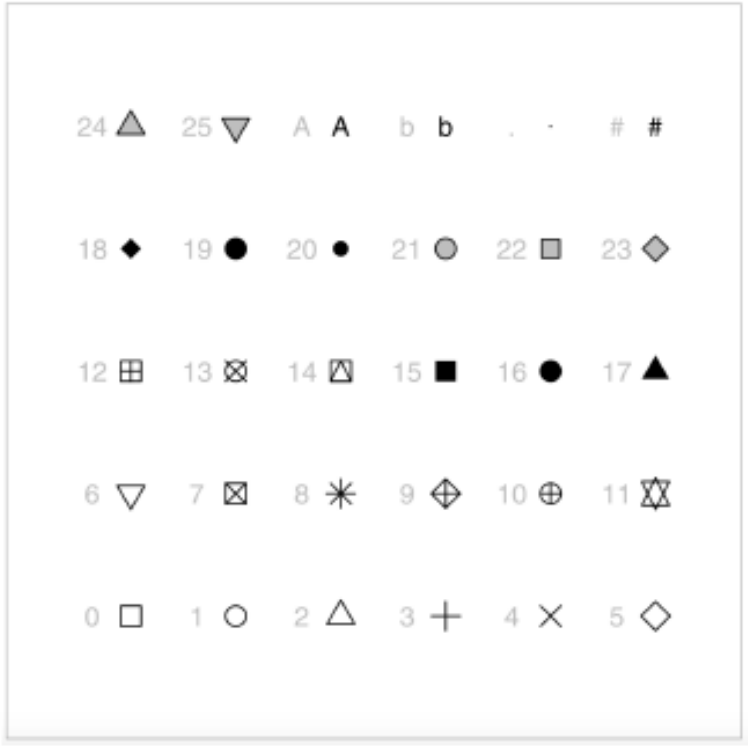
- Change the **plotting symbol** with pch:

```
> plot(customers ~ temperature, main = "Chicago bike trips",
+      xlim = c(0, 40), xlab = "Temperature in Celsius", ylab = "No of customers",
+      cex = 2, cex.main = 2, cex.lab = 1.5, cex.axis = 1.5,
+      pch = 19,
+      data = data)
```



Illustrative graphs

The plotting symbols are controlled by the `pch` (**plotting character**) argument. R provides a fixed set of 26 symbols:



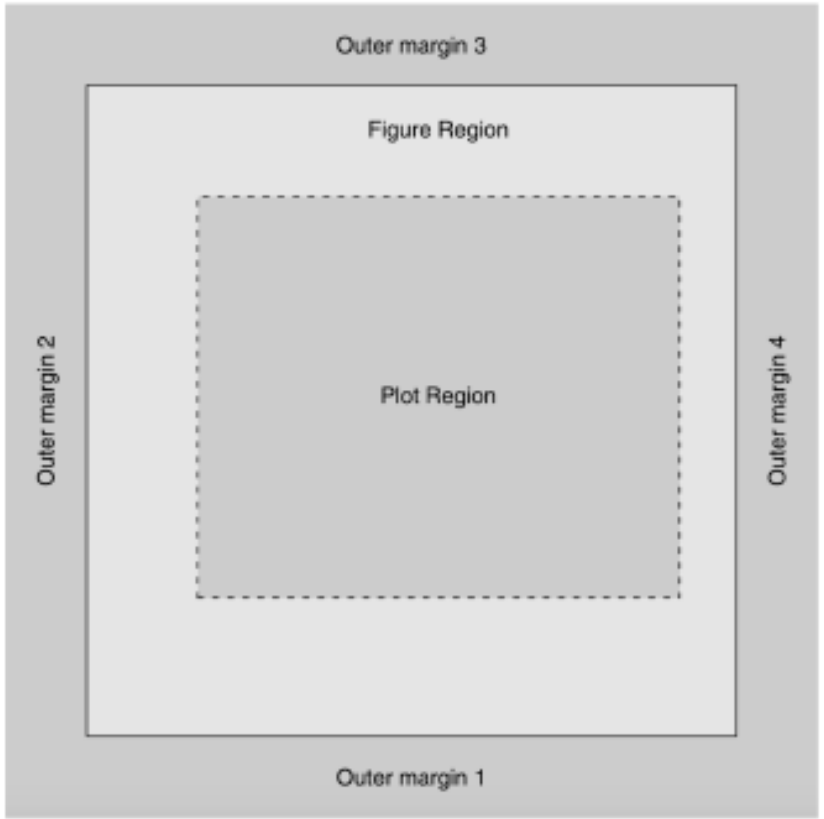

```
List of R's plotting symbols
```

Some of the predefined plotting symbols (pch between 21 and 25) allow a fill color separate from the border color. In these cases, the fill color can be controlled with the `bg` setting.

Illustrative graphs

- Change the **plot regions** and the **figure margins**:

In base R every page is split up into three main regions: (1) the **outer margins**, (2) the current **figure region**, and (3) the current **plot region**:



The plot regions for a single default R graph.

Illustrative graphs

The `plot()`-function draws plotting symbols and lines within the plot region and axes and labels in the figure margins or outer margins. The size of these regions can be controlled via the `par()`-function and the graphics state settings `oma` (or `omi` in inches) for the size of the **outer margins** and `mar` (or `mai` in inches) for the size of the **figure margins**.

Typing `par()` will result in a complete listing of the current graphics state settings:

```
> par()
...
$mar
[1] 5.1 4.1 4.1 2.1
...
$oma
[1] 0 0 0 0
...
```

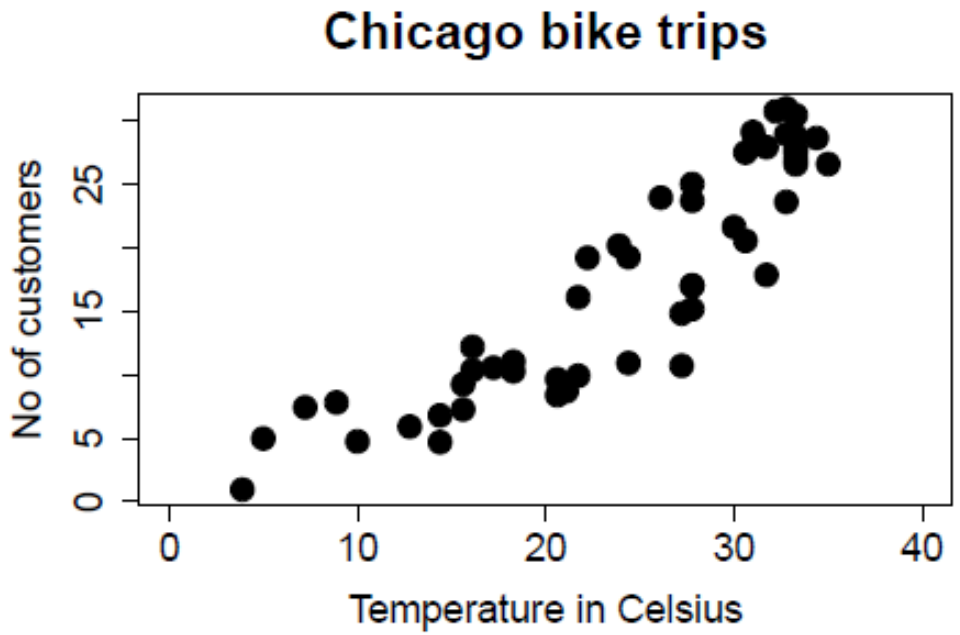
Changes of the graphic state settings in `par()` only come into effect when the new plot is started. Then, they have a **persistent** effect and apply to **all** successive plots in the active R session until a different setting is specified.

Illustrative graphs

We save the default graphic state settings in an object named `old.par` to be able switch back to it later:

```
> old.par <- par()
> par(mar = c(5.1, 0, 4.1, 0) + 0.1) # set left & right figure margins to 0

> plot(customers ~ temperature, main = "Chicago bike trips",
+      xlim = c(0, 40), xlab = "Temperature in Celsius", ylab = "No of customers",
+      cex = 2, cex.main = 2, cex.lab = 1.5, cex.axis = 1.5,
+      pch = 19,
+      data = data)
```

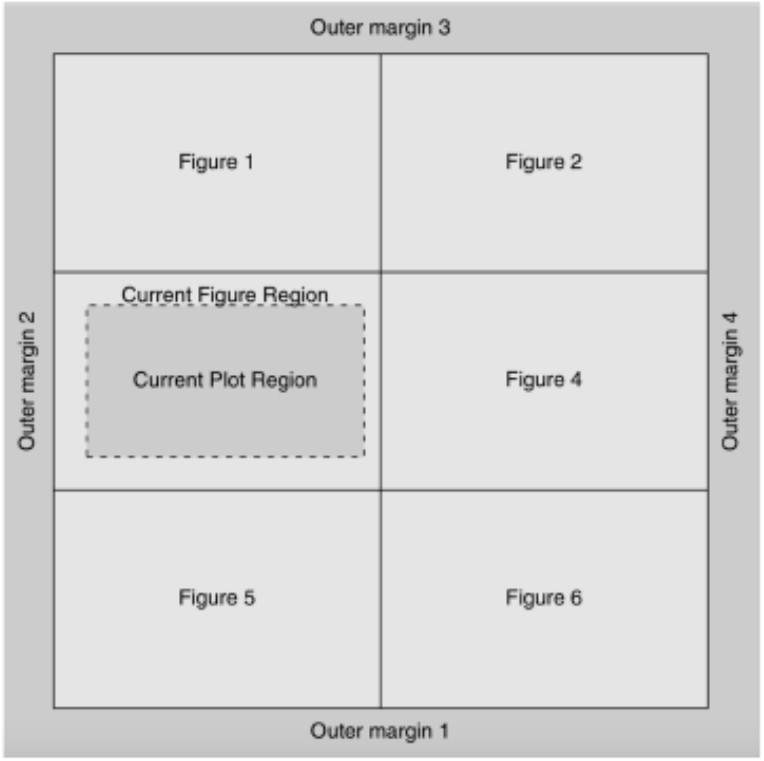


Illustrative graphs

```
> par(old.par)  # switch back to default settings
```

- Change the **number of figures** in the plot region:

The number of figures in the plot region can be controlled via the `mfrow` and `mfcol` graphics state settings. Both of these consist of two values indicating a number of rows, nr, and a number of columns, nc; these settings result in $nr \times nc$ figure regions of equal size:



The plot regions for multiple default R graphs.

Illustrative graphs

The top-left figure region is used first. If the setting is made via mfrow then the figure regions along the top row are used next from left to right, until that row is full. After that, figure regions are used in the next row down, from left to right, and so on. When all rows are full, a new page is started.

```
> par(mfrow = c(1,2))   # fill-up figure regions row-wise
> hist(Chicago.agg$tripduration)
> plot(tripduration ~ temperature, data = Chicago.agg)
> barplot(tripduration ~ event, data = Chicago.agg)
> par(old.par)          # switch back to default settings
```

If the setting is made via mfcol, figure regions are used by column instead of by row.

Illustrative graphs

Annotations

Sometimes it is not enough to just modify the default graphic output. In many situations, further graphical output needs to be added to achieve the desired result.

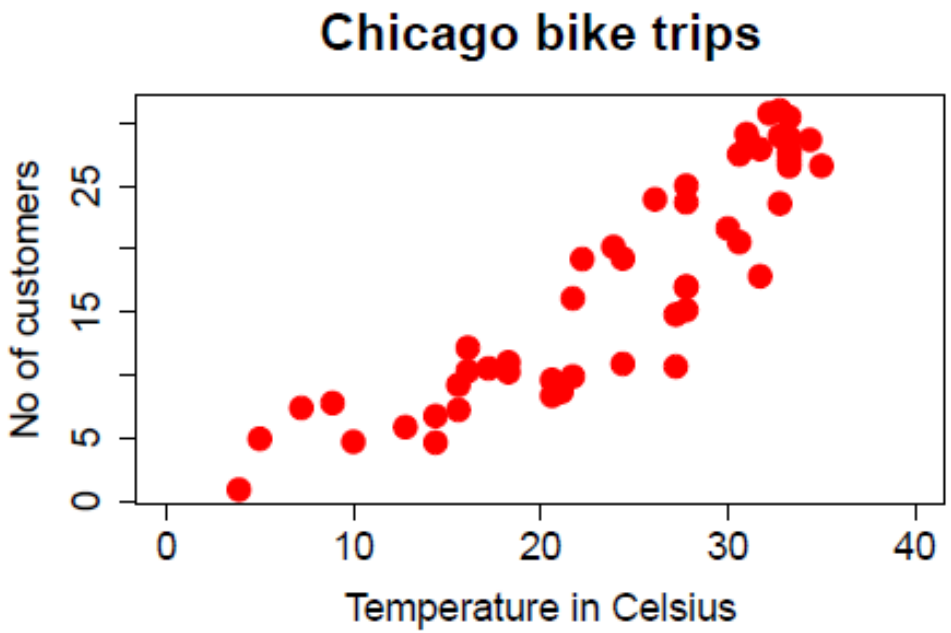
- Adding **colors**:

The `col` argument is the most commonly used for adding (or changing) a default graphs color. The primary use is to specify the color of plotting symbols, lines, text, and so on that are drawn in the plot region.

Illustrative graphs

Named colors: the easiest way to specify a color in R is simply to use the color's name. For example, "red" can be used to specify that plotting symbols in a scatterplot should be red:

```
> plot(customers ~ temperature, main = "Chicago bike trips",
+      xlim = c(0, 40), xlab = "Temperature in Celsius", ylab = "No of customers",
+      cex = 2, cex.main = 2, cex.lab = 1.5, cex.axis = 1.5,
+      pch = 19,
+      col = "red",
+      data = data)
```



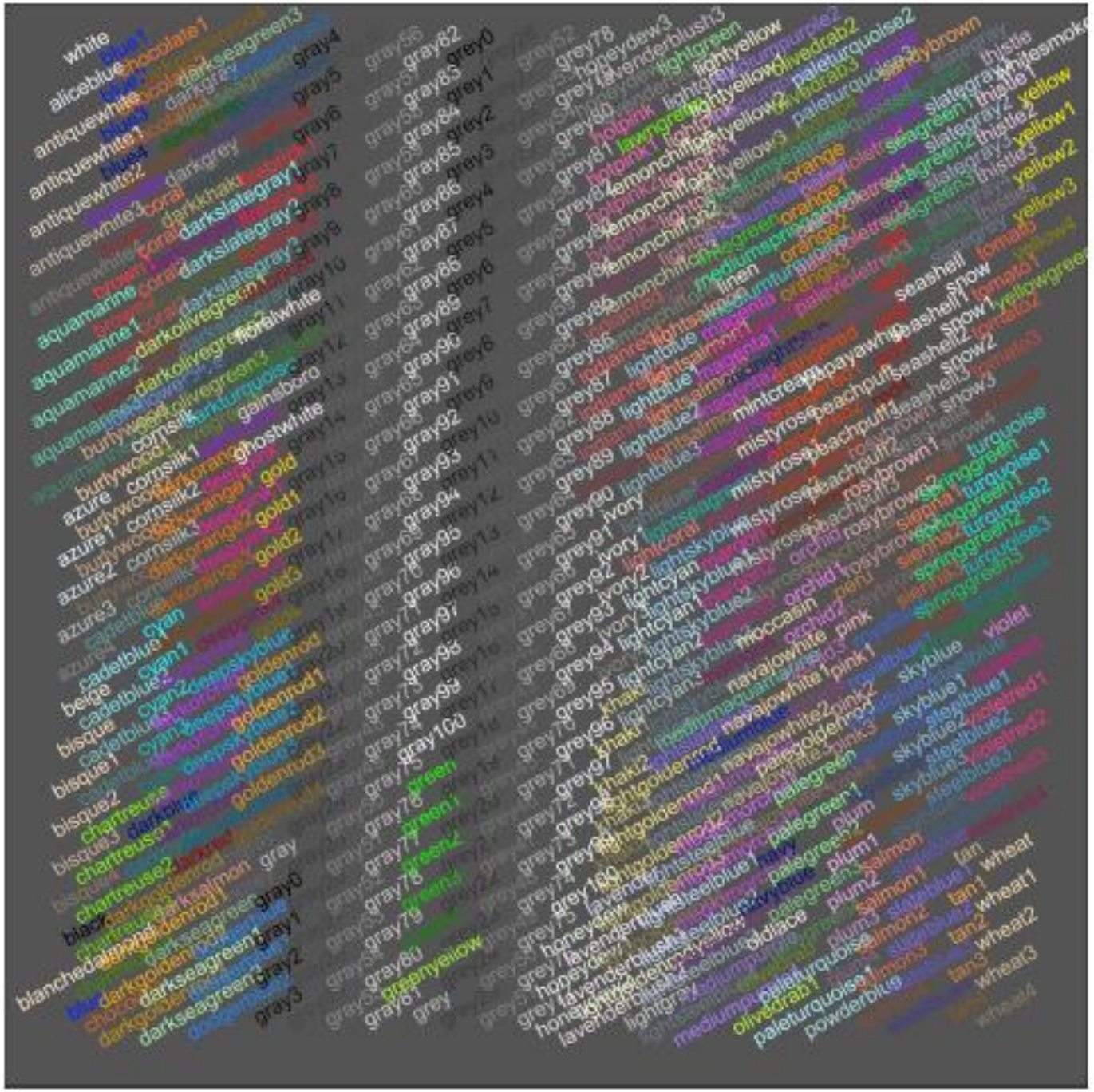
Illustrative graphs

R understands 657 different color names. The full list of known color names can be obtained with the empty function call `colors()` or `colours()`.

```
> colors()
```

```
[1] "white"          "aliceblue"      "antiquewhite"   "antiquewhite1"
 [5] "antiquewhite2" "antiquewhite3" "antiquewhite4" "aquamarine"
 [9] "aquamarine1"   "aquamarine2"   "aquamarine3"   "aquamarine4"
[13] "azure"         "azure1"         "azure2"         "azure3"
[17] "azure4"        "beige"          "bisque"         "bisque1"      ...
```

Full list of named colors in R



Illustrative graphs

RGB colors: it is also possible to specify colors using one of the standard color-space descriptions. The `rgb()`-function allows a color to be specified as a **Red-Green-Blue** (RGB) triplet of intensities. In RGB the color `"red"` is specified as `rgb(1, 0, 0)` (i.e., as much red as possible, no blue, and no green). To see the RGB values for a particular color name use the the `col2rgb()`-function:

```
> col2rgb("red")
      [,1]
red    255
green    0
blue     0
```

HSV colors: there is also an `hsv()`-function for specifying a color as a **Hue-Saturation-Value** (HSV) triplet. **Hue** corresponds to a position on the rainbow, from red (0), through orange, yellow, green, blue, indigo, to violet (1); **saturation** determines whether the color is dull or bright; and **value** determines whether the color is light or dark.

Illustrative graphs

The HSV specification for the (very bright) color red is `hsv(0, 1, 1)`. The function `rgb2hsv()` converts a color specification from RGB to HSV:

```
> rgb2hsv(c(255,0,0))
```

```
  [,1]
h    0
s    1
v    1
```

An alternative way to provide an RGB color specification is to provide 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 zero (00) to 255 (FF). In this specification, the color red is given as `"#FF0000"`.

! In R all color models translate to hex!

Color sets: More than one color is often required within a single plot and in such cases it can be difficult to select colors that are aesthetically pleasing or are related in some way (e.g., a set of colors in which the brightness of the colors decreases in regular steps).

The functions in the following table select a set of colors by taking regular steps along a path through the HSV color space:

Illustrative graphs

Setting	Description
rainbow()	Colors vary from red through orange, yellow, green, blue, and indigo, to violet.
heat.colors()	Colors vary from white, through orange, to red.
terrain.colors()	Colors vary from white, through brown, to green.
topo.colors()	Colors vary from white, through brown then green, to blue.
cm.colors()	Colors vary from light blue, through white, to light magenta.
gray.colors()	A set of shades of grey.

Illustrative graphs

```
> n <- 5
> rainbow(n)
```

```
[1] "#FF0000FF" "#CCFF00FF" "#00FF66FF" "#0066FFFF" "#CC00FFFF"
```

```
> terrain.colors(n)
```

```
> heat.colors(n)
```

```
> gray.colors(n)
```

The RColorBrewer package provides color palettes from **Cynthia Brewer's ColorBrewer tool** (see also http://colorbrewer2.org). The RColorBrewer color sets have been carefully selected by a color expert and include distinct palettes for representing nominal and ordinal categories.

```
> library("RColorBrewer")
> ?RColorBrewer   # check the helppage to obtain the available color sets
```

Illustrative graphs

- **Diverging palettes** put emphasis on mid-range critical values and extremes at both ends of the data range.

```
> brewer.pal(5, "Spectral")
[1] "#D7191C" "#FDAE61" "#FFFFBF" "#ABDDA4" "#2B83BA"
```

- **Sequential palettes** for ordered data that progress from low to high

```
> brewer.pal(5, "Blues")
[1] "#EFF3FF" "#BDD7E7" "#6BAED6" "#3182BD" "#08519C"
```

Illustrative graphs

- **Qualitative palettes** for nominal or categorical data (they do not imply differences between groups).

```
> brewer.pal(5, "Accent")
```

```
[1] "#7FC97F" "#BEAED4" "#FDC086" "#FFFF99" "#386CB0"
```

More color sets:

```
> library(colorspace)
```

colorspace diverging palettes

```
> diverge_hcl(5)
```

```
[1] "#023FA5" "#A1A6C8" "#E2E2E2" "#CA9CA4" "#8E063B"
```

colorspace sequential palettes:

```
> sequential_hcl(5)
```

```
[1] "#023FA5" "#6A76B2" "#A1A6C8" "#CBCDD9" "#E2E2E2"
```

Illustrative graphs

colorspace qualitative palettes:

```
> rainbow_hcl(5)
[1] "#E495A5" "#BDAB66" "#65BC8C" "#55B8D0" "#C29DDE"
```

- Wes Anderson movie palettes

```
> library(wesanderson)
> wes_palettes
$BottleRocket1
[1] "#A42820" "#5F5647" "#9B110E" "#3F5151" "#4E2A1E" "#550307" "#0C1707"

$BottleRocket2
[1] "#FAD510" "#CB2314" "#273046" "#354823" "#1E1E1E"
```

Illustrative graphs

```
$Rushmore1
[1] "#E1BD6D" "#EABE94" "#0B775E" "#35274A" "#F2300F"

$Rushmore
[1] "#E1BD6D" "#EABE94" "#0B775E" "#35274A" "#F2300F"

$Royal1
[1] "#899DA4" "#C93312" "#FAEFD1" "#DC863B"

$Royal2
[1] "#9A8822" "#F5CDB4" "#F8AFA8" "#FDDDA0" "#74A089"
```

- Do-it-yourself sequential palettes

```
> colorRampPalette(c("#FFD700", "gray30"))(10) # use ESE yellow
 [1] "#FFD700" "#EBC708" "#D7B811" "#C3A919" "#AF9922" "#9C8A2A" "#887A33"
 [8] "#746B3B" "#605C44" "#4D4D4D"
```

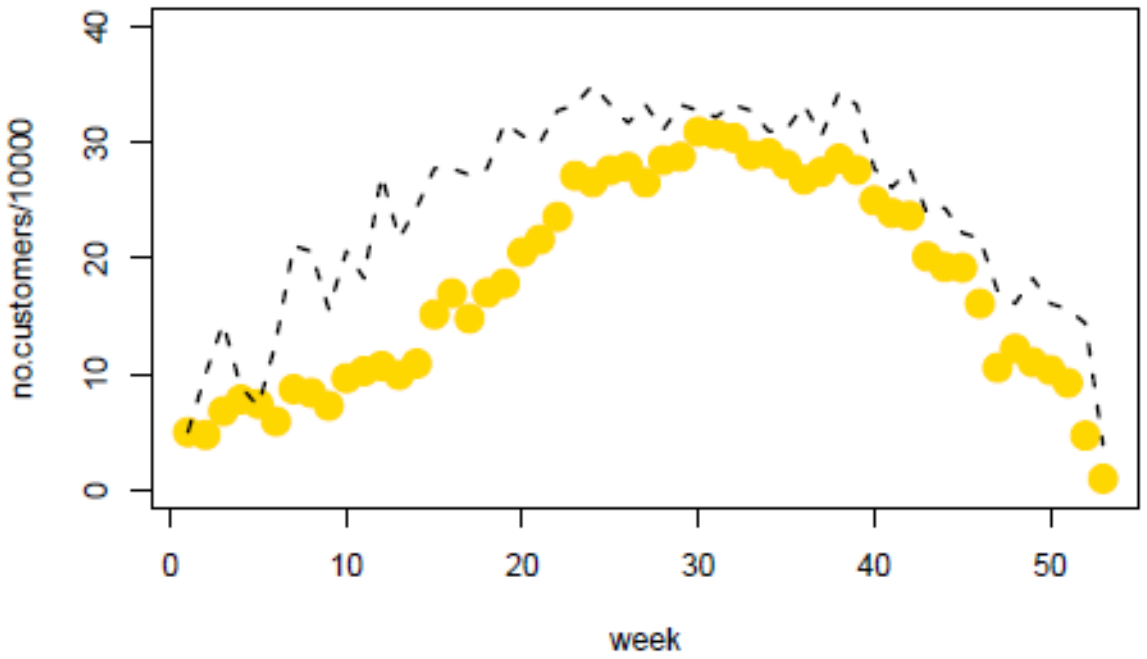
Illustrative graphs

- Adding **data**:

The functions `points()` and `lines()` add graphical output to the plot region. The `lines()`-function draws lines between (x, y) locations (`NA` values in the (x, y) locations will create breaks in the line), and the `points()`-function draws plotting symbols at (x, y) locations.

To differentiate the data points commonly the **line types** (`lty`) and **line width** (`lwd`) or the **plotting characters** (`pch`) are changed.

```
> # plot the number of customers over the 31 weeks
> plot(no.customers/10000 ~ week, ylim = c(0, 40),
+      col = "#FFD700", pch = 19, cex = 2,
+      data = customers)
> # add temperature line (curve)
> lines(temperature ~ week, lty="dashed", lwd = 2,
+       data = temp)
```



Illustrative graphs

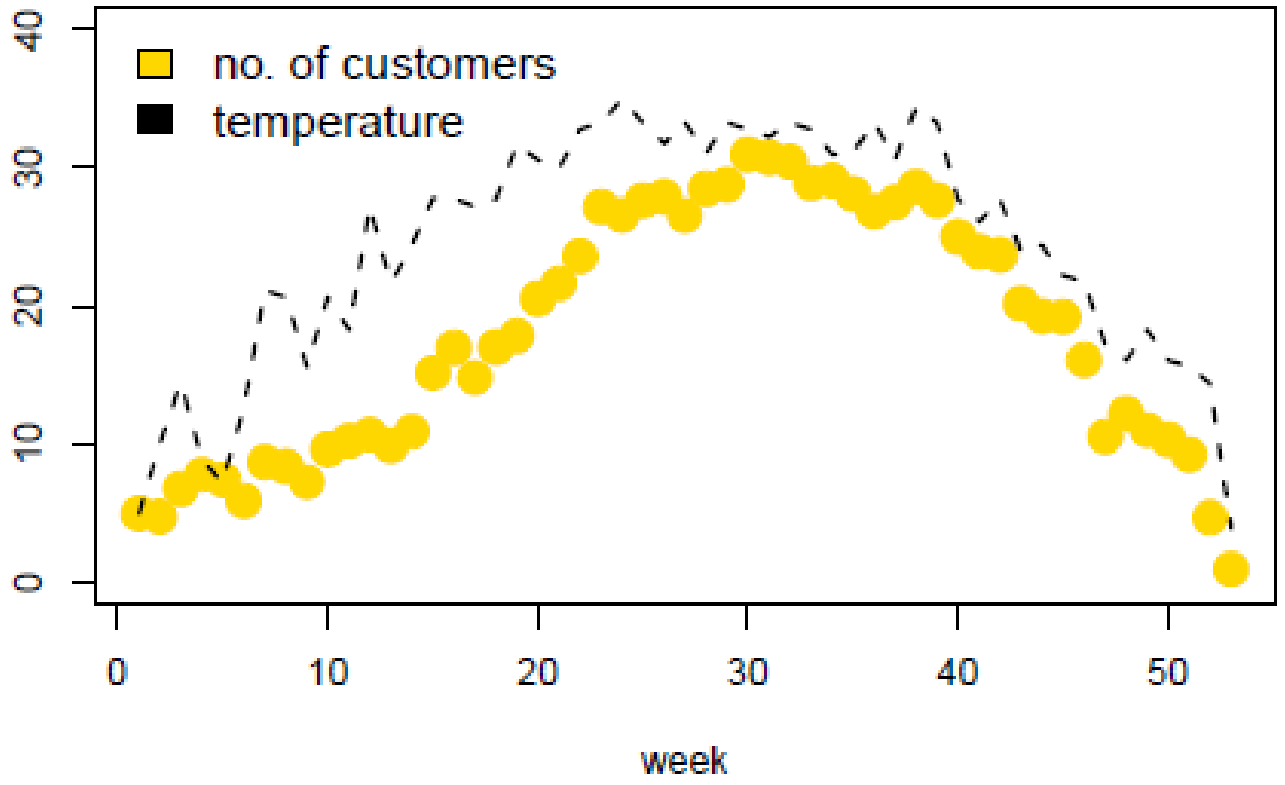
- Adding **legends**:

A legend or key can be added to a plot with the `legend()`-function. The legend is usually drawn within the plot region. The function has many arguments, which allow for flexibility in the specification of the contents and layout of the legend.

```
> plot(no.customers/10000 ~ week, ylim = c(0, 40), ylab = "",
+       col = "#FFD700", pch = 19, cex = 2,
+       data = customers)
> lines(temperature ~ week, lty="dashed", lwd = 2,
+        data = temp)
```

Illustrative graphs

```
> legend("topleft", c("no. of customers", "temperature"),
+        cex=1.25, fill=c("#FFD700", "black"), bty = "n")
```



! It is entirely the responsibility of the user to ensure that the legend corresponds to the plot. There is no automatic checking that data symbols in the legend match those in the plot, or that the labels in the legend have any correspondence with the data.

Ggplot

ggplot and the grammer of graphics

R for Data Science, Chapter 1 & 22, or https://r4ds.had.co.nz/data-visualisation.html & https://r4ds.had.co.nz/graphics-for-communication.html Chapter 1.3 & 5.28

If you want to do anything beyond very simple graphs, it is recommended to switch to the add-on package `ggplot2`.

`ggplot2` is based on the idea that a graph can be constructed by semantic components. This allows to build graphical features up in a series of layers:

1. **aesthetic** mapping of the data (`aes()`), defines how variables are connected to visual properties or outputs (e.g. color, size, shape)
2. **geometric** objects representing the data:

Geometric	Mapping
geom_histogram()	Histogramm
geom_density()	Densityplot
geom_bar()	Barplot
geom_point()	Points
geom_lines()	Lines
geom_boxplot()	Boxplot

Ggplot

3. **coordinate systems**.
4. **faceting** the data; splitting by some predefined criteria to display sup-graphs.
5. **themes** to control non-data elements.

Themes	Description
theme_bw()	white background with grid lines
theme_grey()	grey background and white grid lines (default)
theme_classic()	white background and no grid lines
theme_minimal()	minimal theme with no background annotations
theme_linedraw()	black lines of various widths on white backgrounds
theme_light()	light grey lines and axes

Ggplot

6. **scales** map values in the data space to values in the aesthetic space (color, size, labels, . . .) and are reported on the plot using axes and legends.

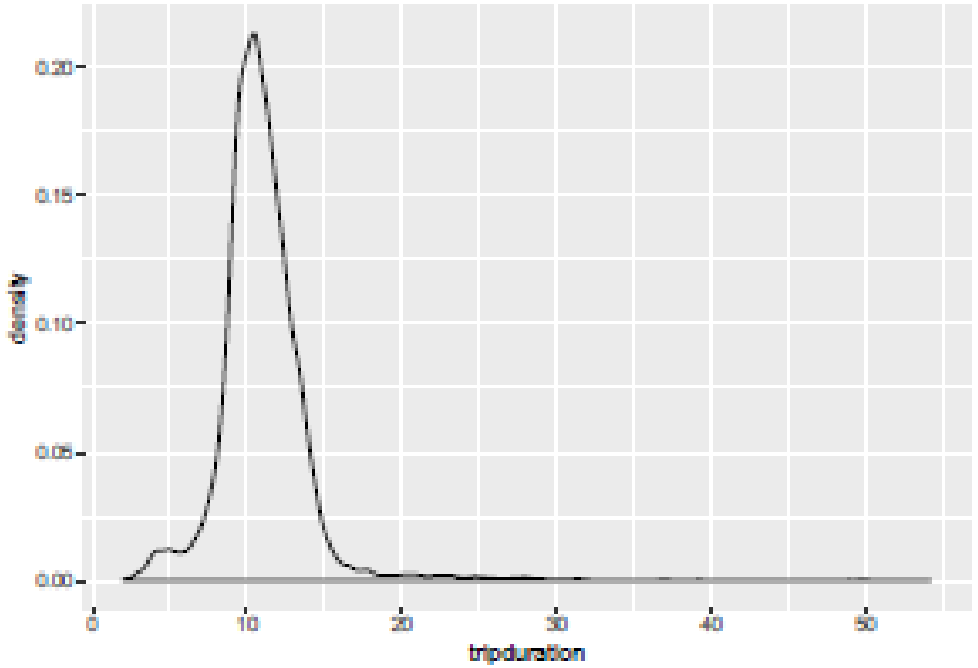
Scale	Description
scale_shape_discrete()	shape scale with discrete values
scale_x_log10(), scale_y_log10()	Log-transform x or y axis
scale_x_sqrt(), scale_y_sqrt()	Square root transformation of x or y axis
scale_trans(x, y)	Possible values: "log2", "log10", "sqrt",. . .
scale_x_continuous(),	minimal theme with no background
scale_y_continuous()	annotations
scale_x_reverse(), scale_y_reverse()	reverse x or y coordinates
scale_colour_discrete()	color scale with discrete value
scale_colour_grey()	grey colors used in the plot
scale_color_brewer(palette)	library(RColorBrewer) display.brewer.all()
scale_color_manual(values)	specify colors to be used manually

Ggplot

```
> library(ggplot2)
```

(Default) densityplot:

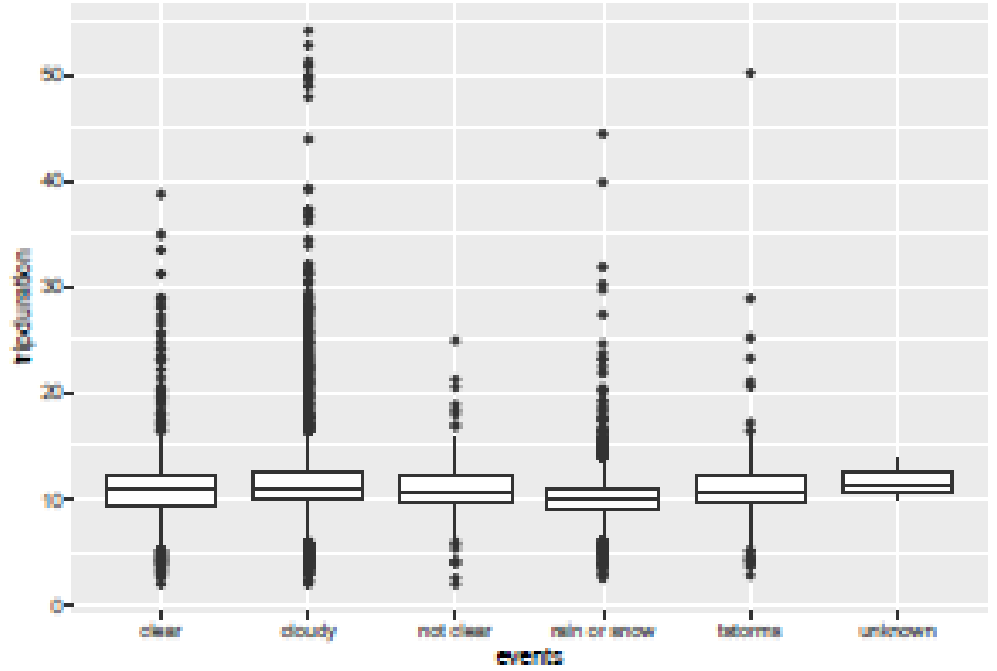
```
> ggplot(Chicago.agg, aes(x = tripduration)) +
+     geom_density()
```



Ggplot

(Default) boxplot:

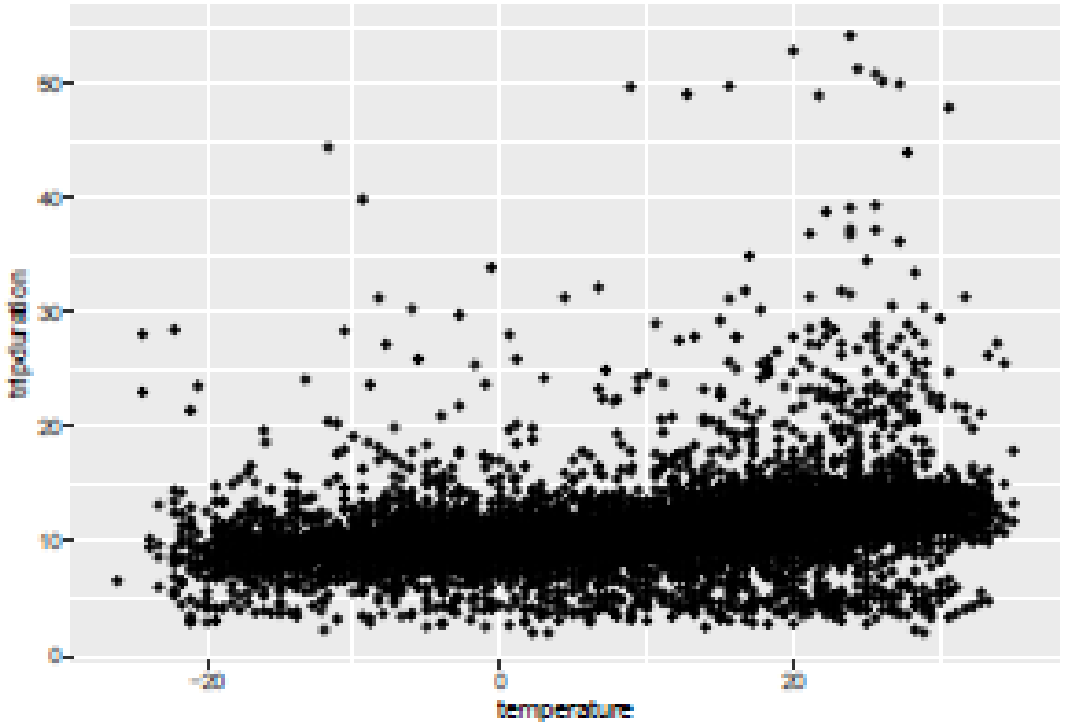
```
> ggplot(Chicago.agg, aes(x = events, y = tripduration)) +
+     geom_boxplot()
```



Ggplot

(Default) scatterplot

```
> ggplot(Chicago.agg, aes(x = temperature, y = tripduration)) +
+     geom_point()
```

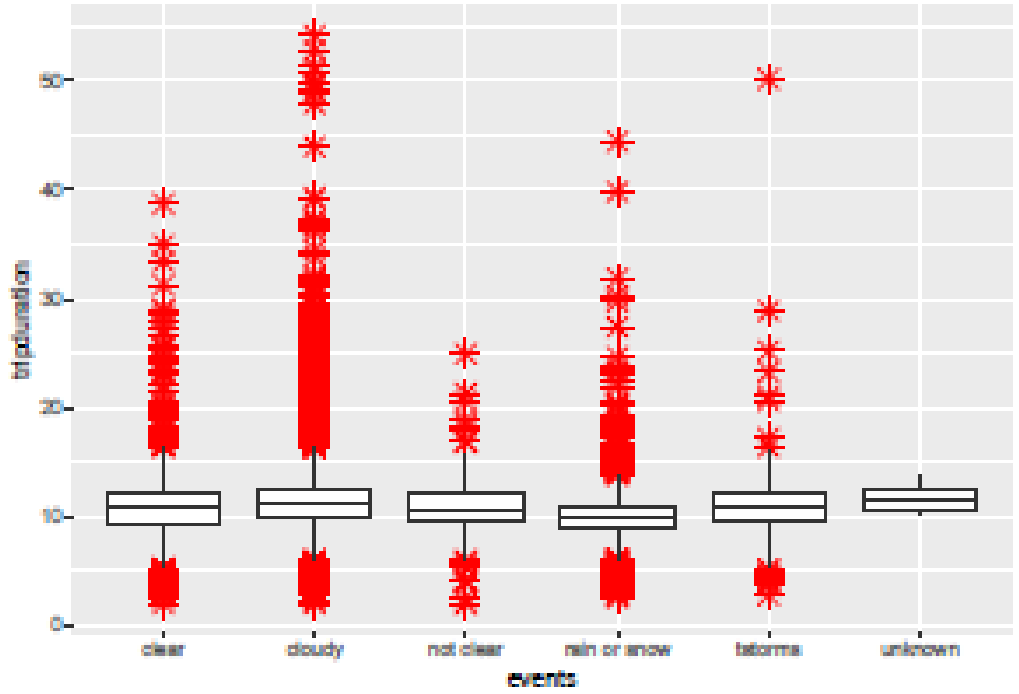


Ggplot

Customizations

Change outlier (color, shape and size)

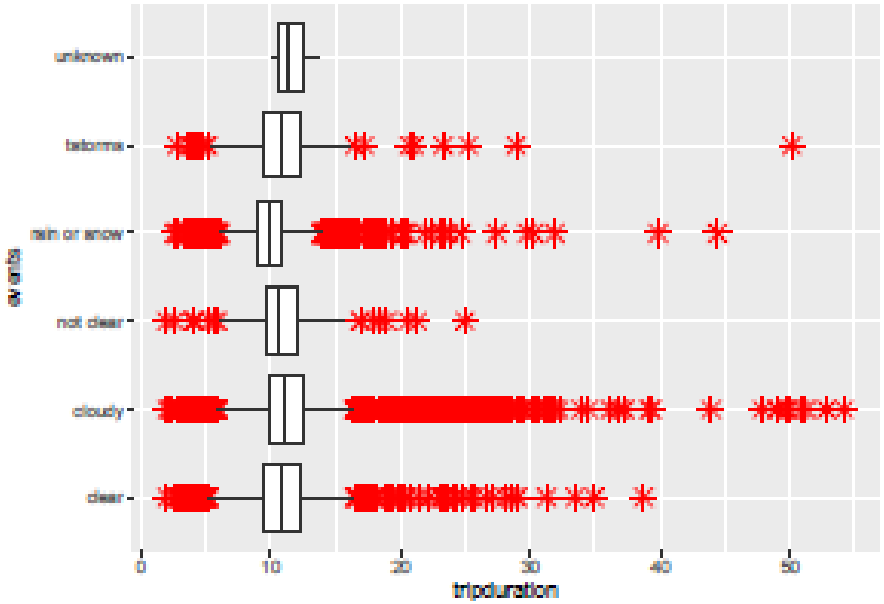
```
> p <- ggplot(Chicago.agg, aes(x = events, y = tripduration)) +
+   geom_boxplot(outlier.colour = "red", outlier.shape = 8, outlier.size = 4)
> p
```



Ggplot

Add coordiante system flip (rotate the boxes)

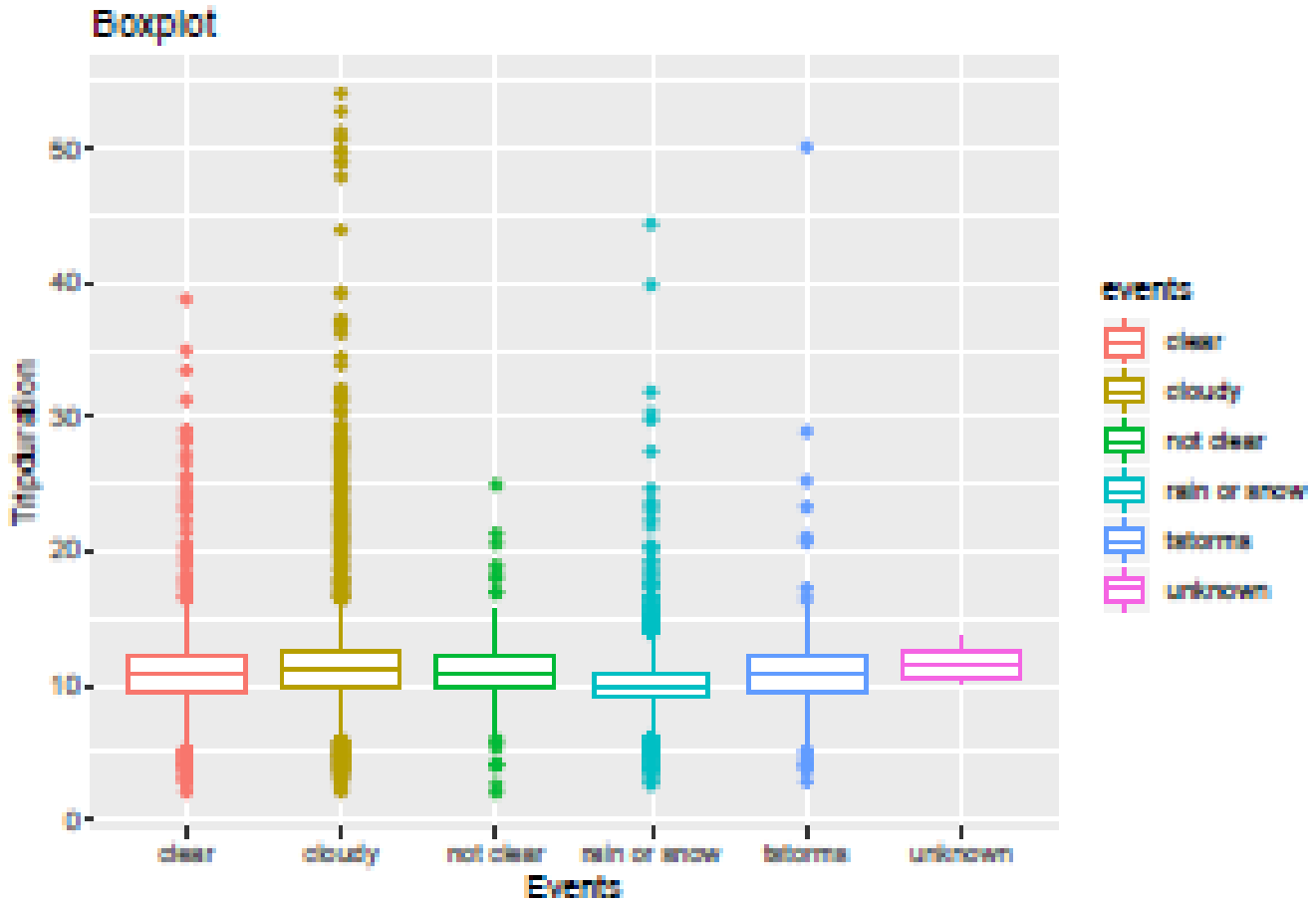
```
> p + coord_flip()
```



Change box plot line colors by groups, add scale labels and plot title

```
> p <- ggplot(Chicago.agg, aes(x = events, y = tripduration, color = events)) +
+   geom_boxplot() +
+   labs(title = "Boxplot", x = "Events", y = "Tripduration")
> p
```

Ggplot

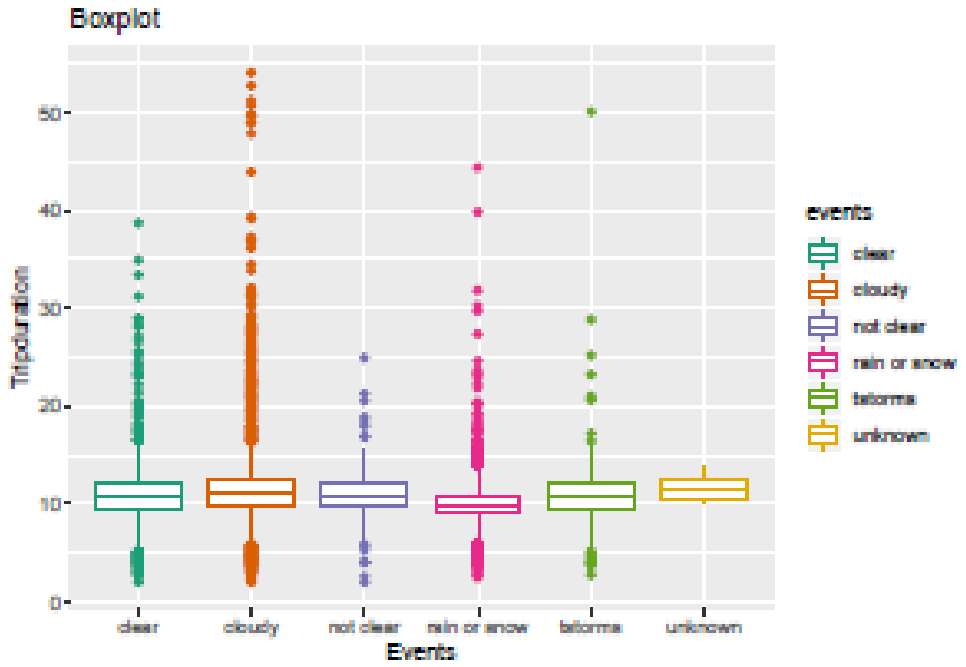


INFOX
educación sin fronteras

Ggplot

Add box line colors by groups using brewer color palettes

```
> p + scale_color_brewer(palette = "Dark2")
```



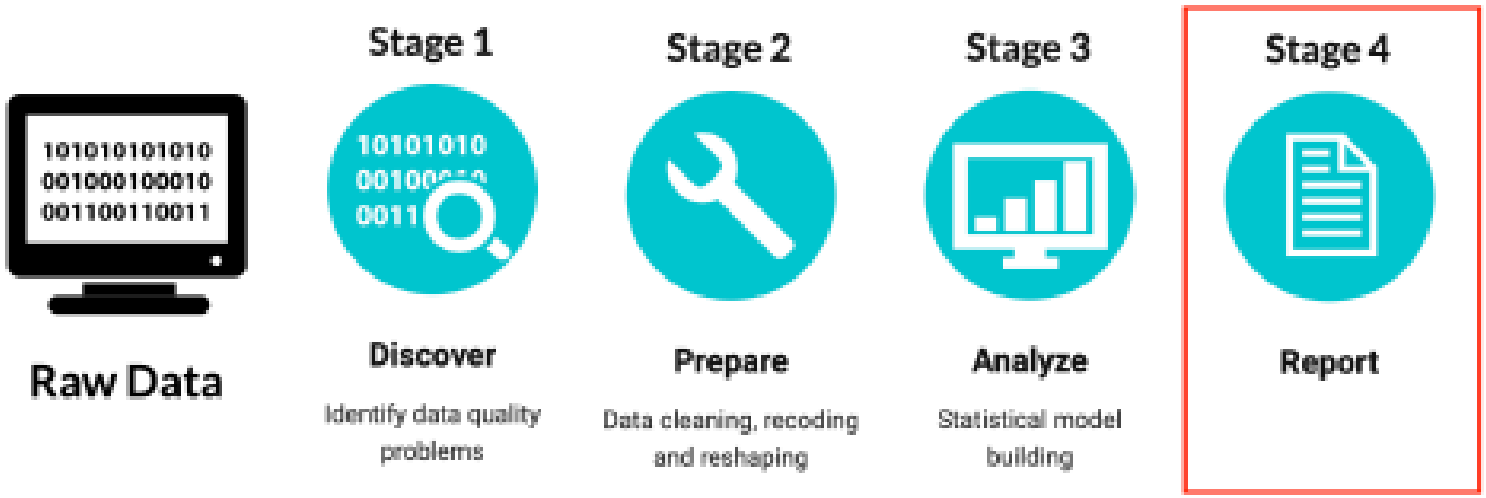
Agenda

2. Reportes en látex usando Rmarkdown

INFOX
educación sin fronteras

Reporting

The data science process



It doesn't matter how great your analysis is unless you can explain it to others: you need to **communicate** your results!

- Communicating to **decision makers**, who want to focus on the conclusions and not the code behind the analysis.
- Collaborating with **other (data) scientists**, who are interested in both the conclusions, and the code.

Reporting

RMarkdown

R for Data Science, Chapter 1 & 22, or https://r4ds.had.co.nz/r-markdown.html Chapter 5.27, 5.29 & 5.30

RMarkdwon is a file format for designing documents that allow to combine code, results, and written text and to store your results in a variety of formats.

RMarkdown documents rely on three different frameworks:

1. **YAML** for render parameters
2. **knitr** for embedded R code
3. **markdown** for formatted text

Reporting

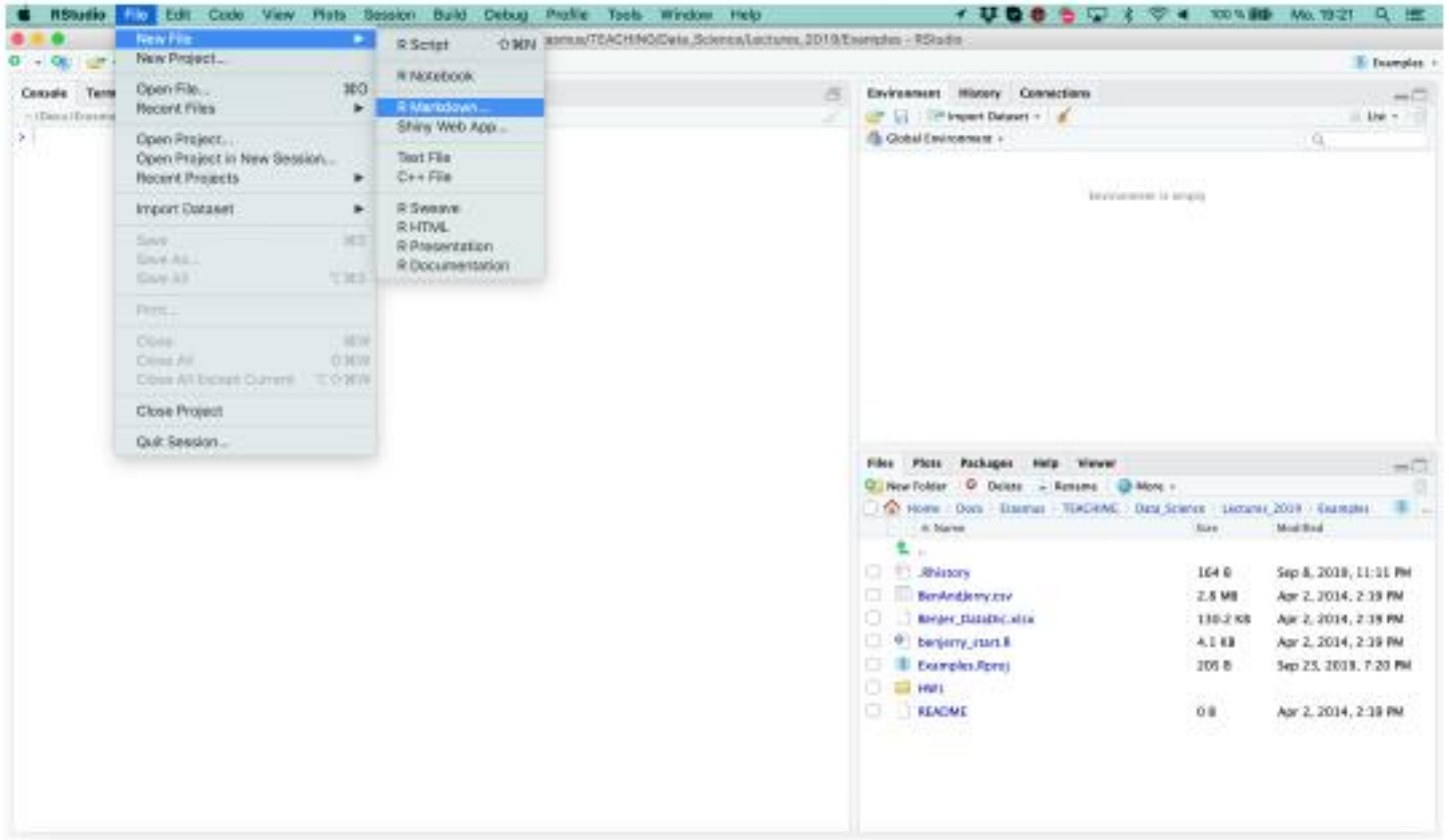
⇒ Reports are fully reproducible (although code is not necessarily displayed).

⇒ Reports can be updated automatically with `knitr` (time saving, e.g. when placing figures and tables).

! The necessary add-on packages (`rmarkdown` and `knitr`) are automatically installed in your R package library when installing RStudio. But RStudio does not build PDF and Word documents from scratch. You will need to have Microsoft Word (or a similar program) installed to produce Word Files. For rendering pdf files you will also need a **TeX** distribution (**miktex** for windows https://miktex.org/download, **mactex** for mac https://tug.org/mactex/mactex-download.html).

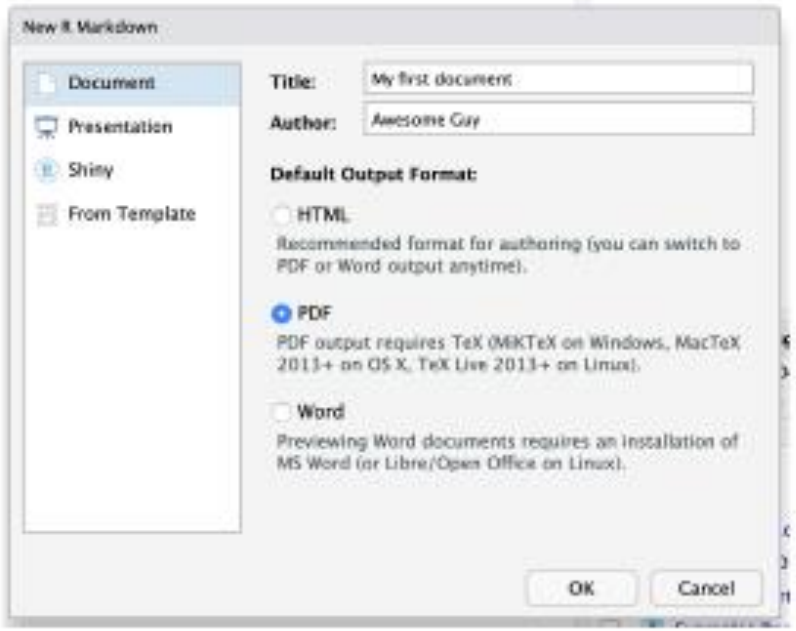
Reporting

Getting started



Create a new RMarkdown file - Step 1.

Reporting




```
Create a new RMarkdown file - Step 2.
```

Reporting

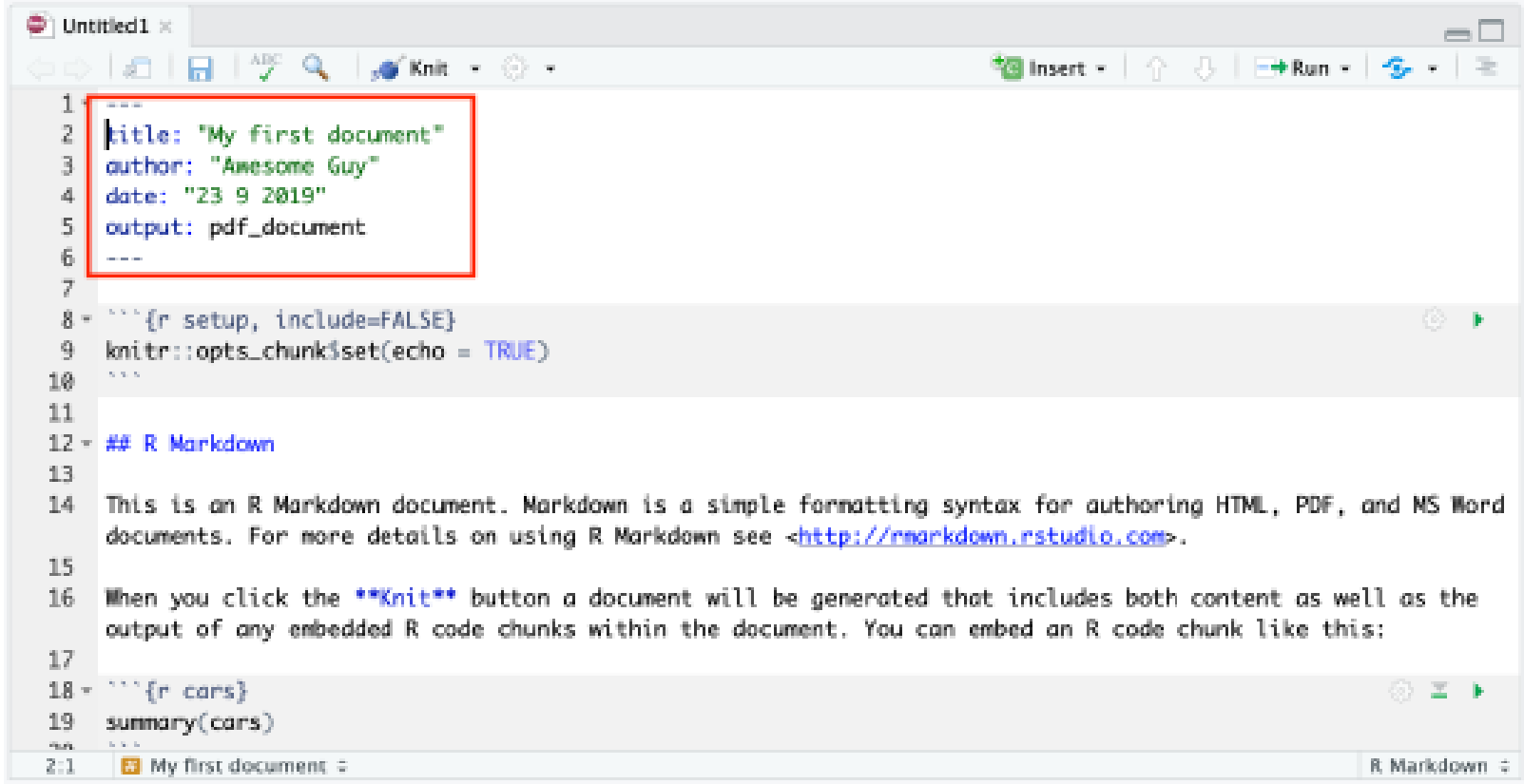
Create a new RMarkdown file - Step 3.

⇒ A template RMarkdown script is provided.

Reporting

1. YAML for render parameters

The YAML header (at the top of the page in between two lines of three dashes) includes the set up information used by `knitr` during rendering to produce the file:



YAML for render parameters.

Reporting

Selection of available document output formats:

Format	Document	Presentation
HTML	`html_document`	`ioslides_presentation`, `xaringan`, `reveal.js`
PDF	`pdf_document`	`beamer_presentation`
Word	`word_document`	`powerpoint_presentation`

- A **table of contents** can be added with the `toc` option. The depth of headers that it applies to is specified with the `toc_depth` option:

```
---
title: "My first document"
output:
  pdf_document:
    toc: true
    toc_depth: 2
---
```

If the table of contents depth is not explicitly specified, it defaults to 3 (meaning that all level 1,

Reporting

2, and 3 headers will be included in the table of contents).

- **Section numberings** can be added to the headers eith the `number_sections` option:

```
---
title: "My first document"
output:
  pdf_document:
    toc: true
    number_sections: true
---
```

- **Width** and **height** of graphical output can be controlled (for example) with the `fig_width` and `fig_height` options:

```
---
title: "My first document"
output:
  pdf_document:
    fig_width: 7
    fig_height: 6
---
```

Reporting

- Enhance the default display of **data frames** (output) with the `df_print` option:

```
---
title: "My first document"
output:
  pdf_document:
    df_print: kable
---
```

- Change the **syntax highlighting** style:

```
---
title: "My first document"
output:
  pdf_document:
    highlight: tango
---
```

Available highlighting styles are: default, tango, pygments, kate, monochrone, espresso, zenburn, haddock, null (prevents syntax highlighting).

Reporting

- Further customizations of the template to create the PDF document:

```
---
title: "My first document"
output:
  pdf_document:
    fontsize: 11pt
---
```

Setting	Description
`fontsize`	Font size (e.g., 10pt, 11pt, or 12pt)
`documentclass`	LaTeX document class (e.g., `article`)
`classoption`	Options for documentclass (e.g., `oneside`)
`geometry`	Options for geometry class (e.g., `margin=1in`)
`linkcolor`, `urlcolor`, `citecolor`	Color for internal, external, and citation links

Reporting

2. Markdown formatted text

Markdown is a set of very easy-to-read conventions for formatting plain text:

Markdown formatted text.

Reporting

- bold and italic text
- ordered and unordered lists
- headers (section titles)
- hyperlinks...

Markdown reference guide:

Toolbar > **Help** > **Markdown Quick Reference**

Mathematical expressions

- **Inline** mathematical expressions can be written in a pair of dollar signs using the LaTeX syntax (see e.g. https://www.overleaf.com/learn/latex/Mathematical_expressions):

```
$f(k) = {n \choose k} p^{k} (1-p)^{n-k}$
```

The output looks like: $f(k) = \binom{n}{k} p^k (1-p)^{n-k}$.

- **Display style** mathematical expressions can be written in a pair of double dollar signs:

```
$$f(k) = {n \choose k} p^{k} (1-p)^{n-k}$$
```

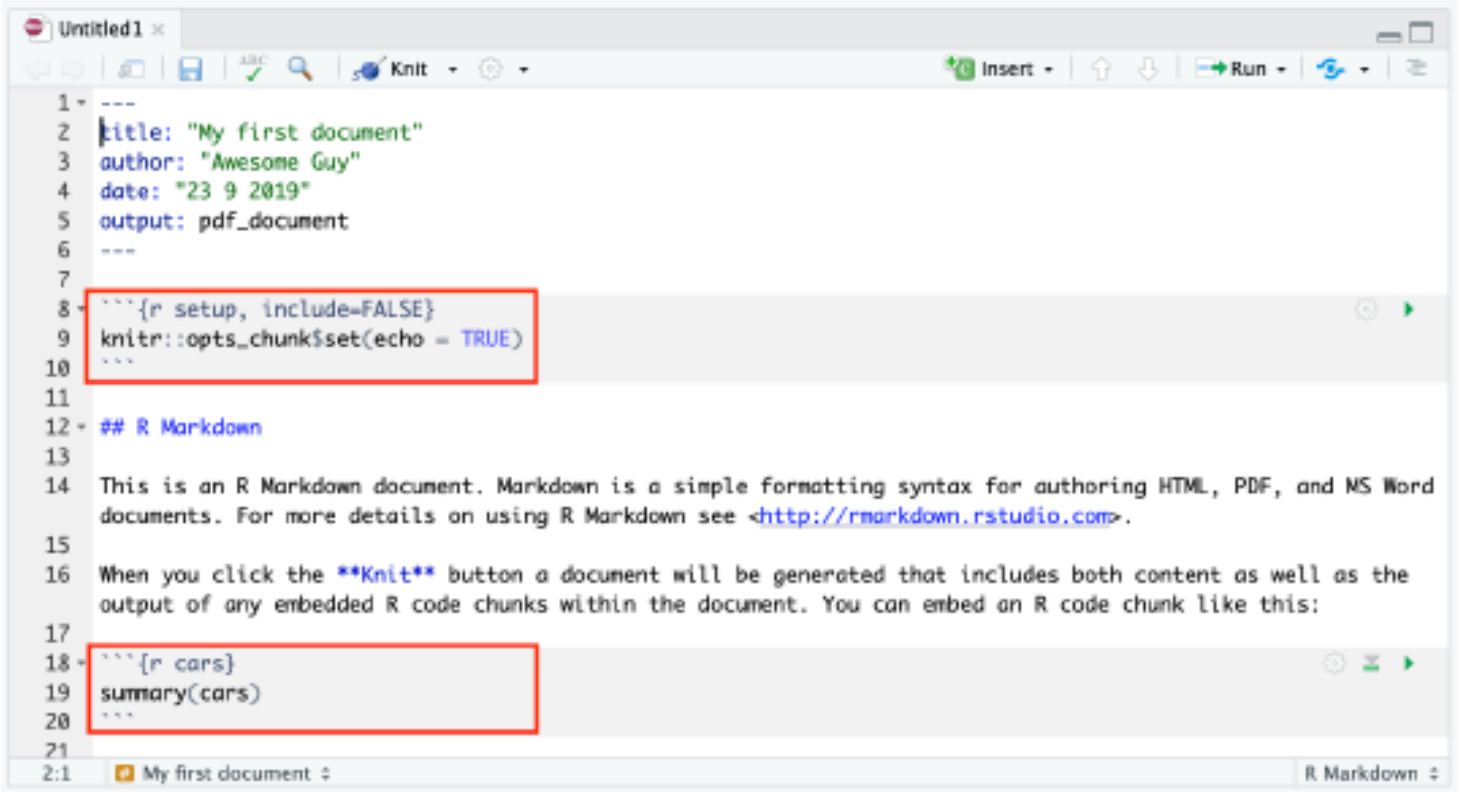
The output looks like:

$$f(k) = \binom{n}{k} p^k (1-p)^{n-k}$$

Reporting

3. knitr for embedded R code

The knitr package extends the basic markdown syntax to include chunks of executable R code. To embed a chunk of R code into your report, surround the code with two lines that each contain three backticks. After the first set of backticks, include {r}, which alerts knitr that you have included a chunk of R code. The result will look like this:

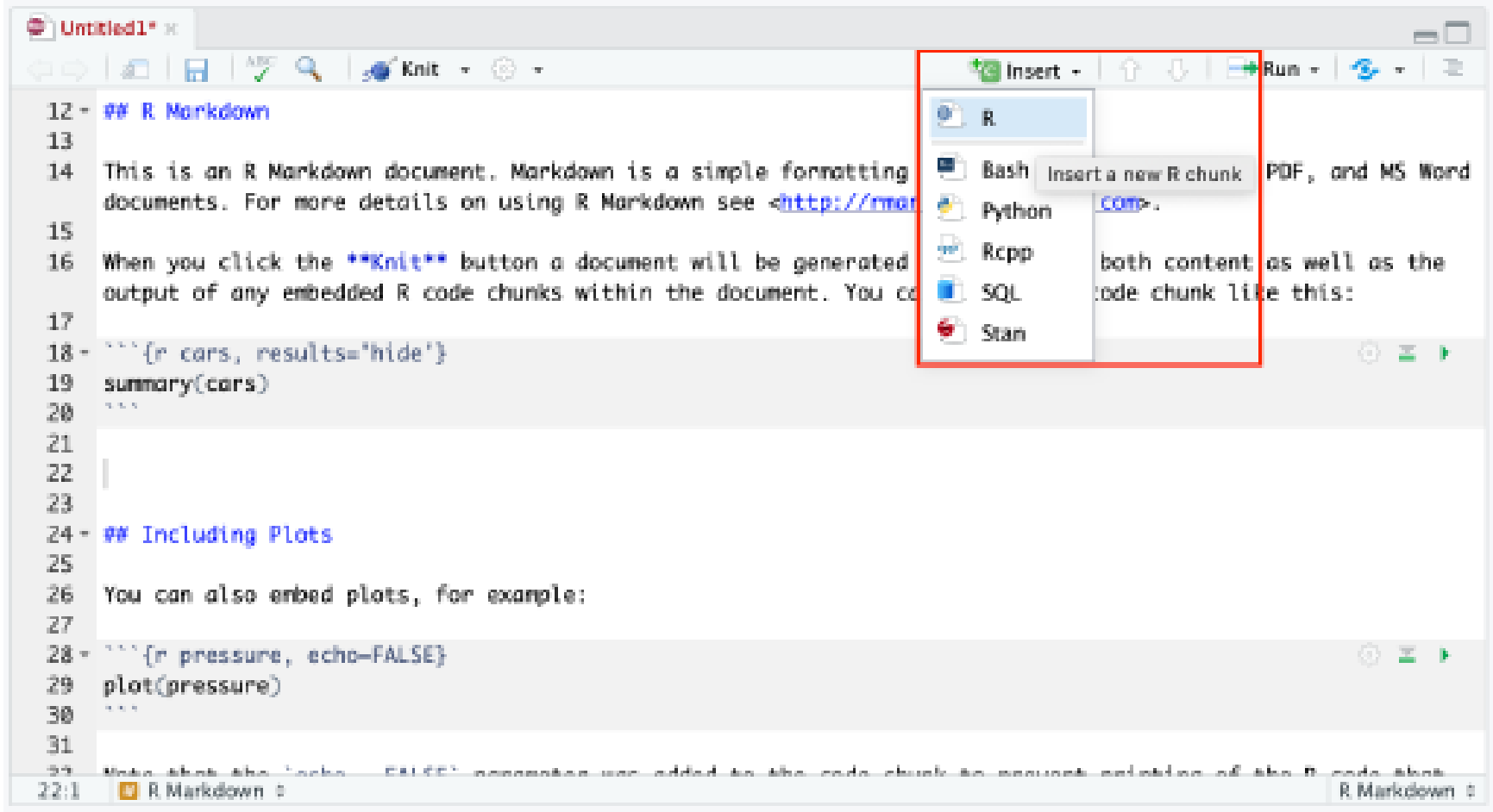


Code chunks for embedded R code.

In a code chunk you can produce text output, tables, or graphics. When you render the report, knitr will run the code and add the results to the output file. In the output file you can have displayed just the code, just the results, or both.

Reporting

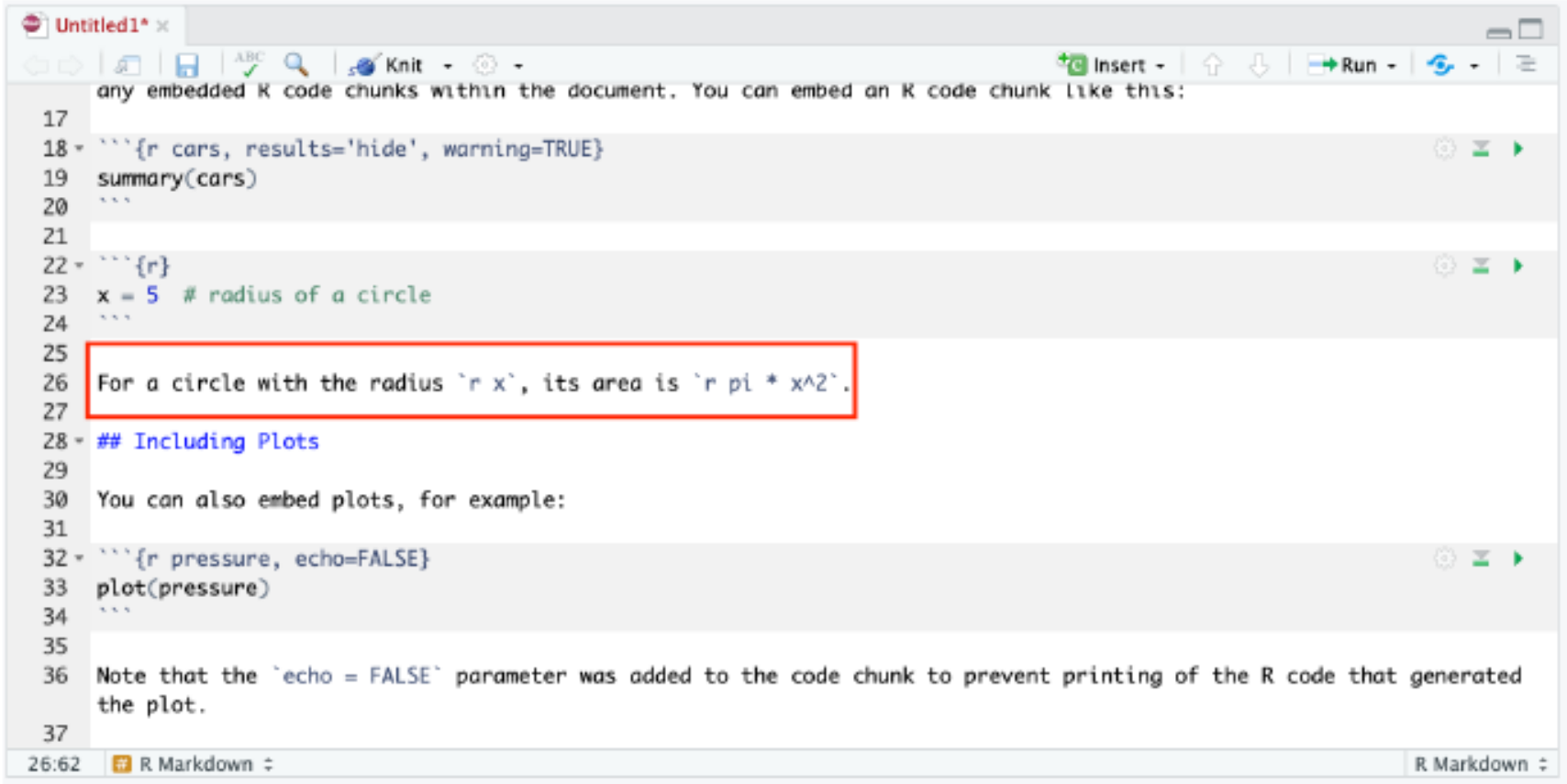
! A new code chunk can be insered using either via the keyboard shortcut `Ctrl + Alt + I` (`Cmd + Option + I` on macOS) or the RStudio toolbar:



Add new R-code chunk.

Reporting

Besides code chunks, you can also insert values of R objects inline in text. For example:



Inline R-code.

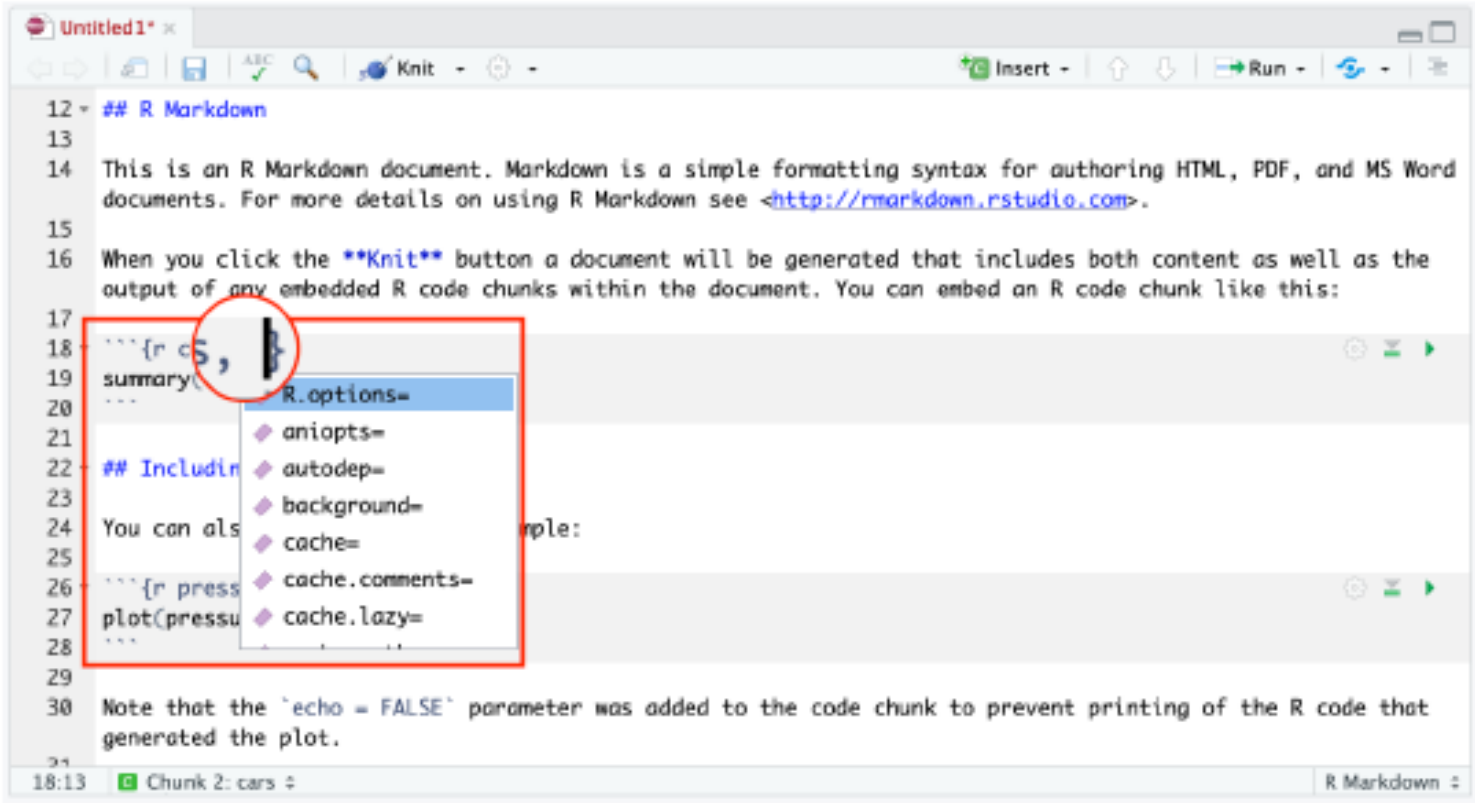
Reporting

Chunk options

You have fine control over the output via **chunk options**, which can be provided inside the curly braces. For example, text output can be hidden via the chunk option `results = 'hide'`, or the height for a specific figure can be set to 4 inches via `fig.height = 4`. Multiple chunk options are separated by commas:

```
{r results = 'hide', fig.height = 4}
```

Further code chunk options:



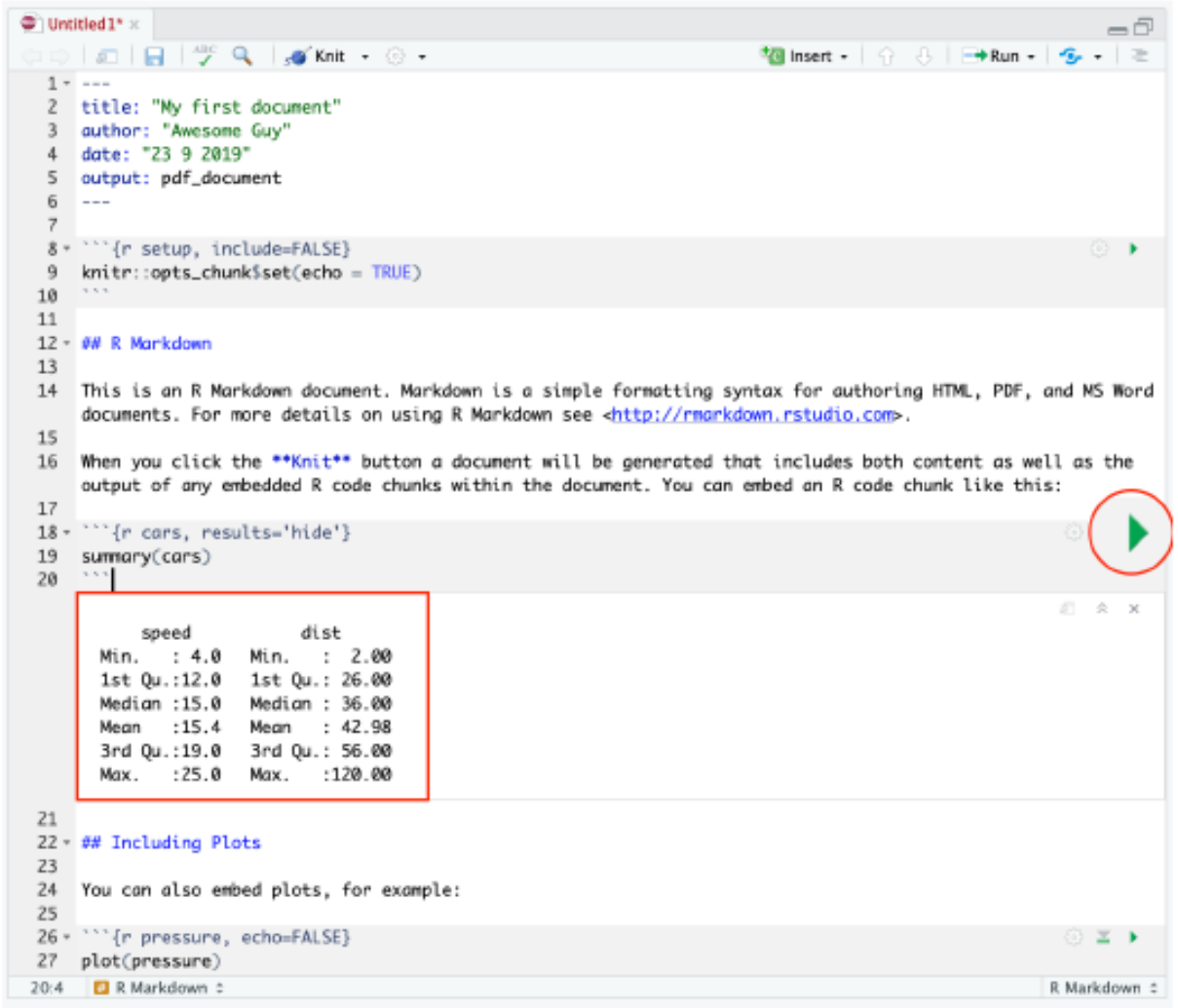
Reporting

```
List of chunk options.
```

Setting	Description
`echo = TRUE`	Omit code from the final report (and include the result only).
`results = 'hide'`	(Opposite) include code and omit results.
`eval = TRUE`	Wether to evaluate a code chunk.
`warning = TRUE`	Display warning messages?
`message = TRUE`	Display code messages?
`fig.height, fig.width`	Specify figure height and width.
`fig.align`	Specify figure to right, left or center align.
`out.width, out.height`	Width and height to which figures are scaled.

Reporting

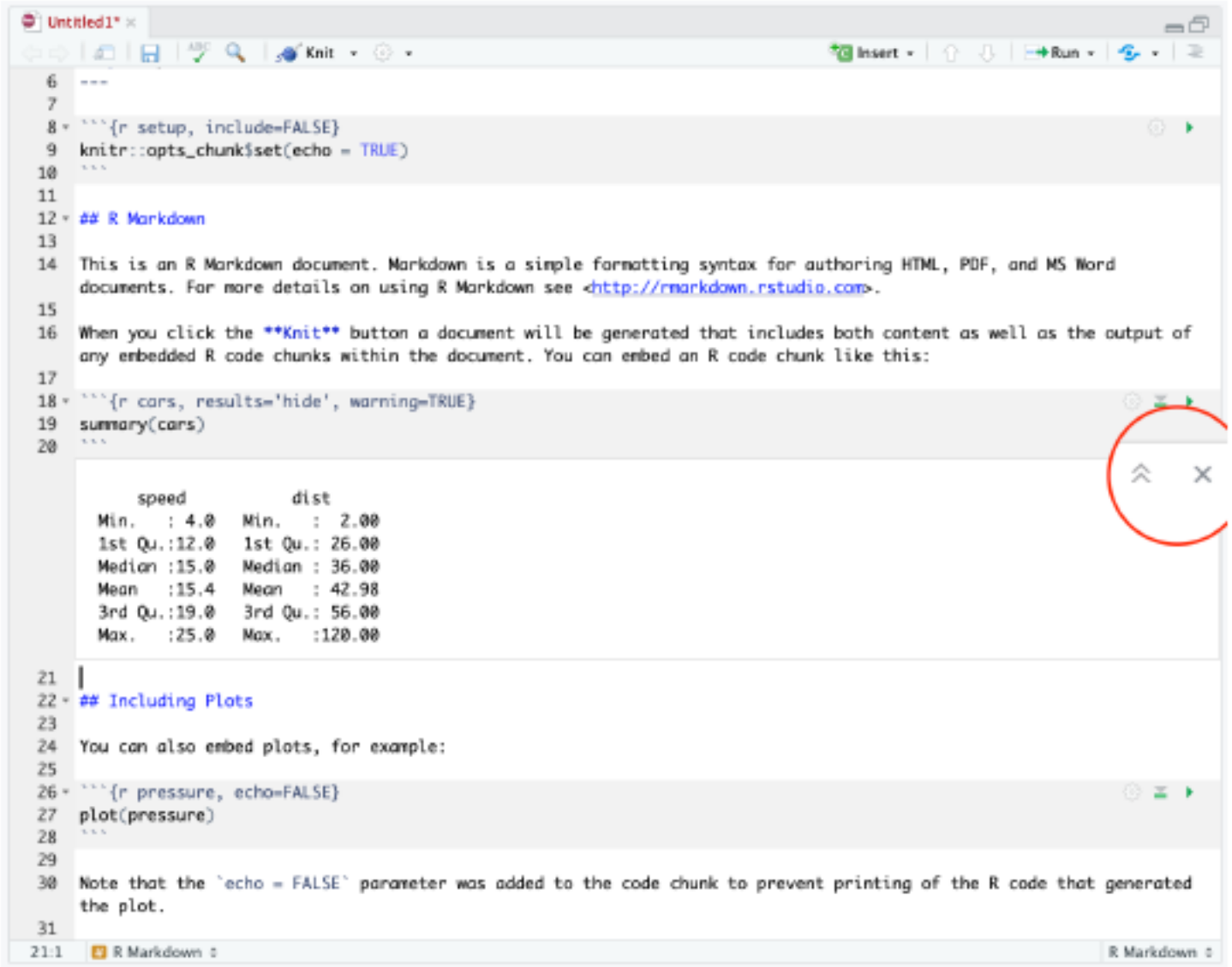
In an RMarkdown file code and output are interleaved. Thus, you can run each code chunk independently by clicking the **Run** icon (or use the shortcut `Ctrl/ Cmd + Shift + Enter`, like in an R-Script file). RStudio executes the code and displays the results inline with the code:



The output produced from an RMarkdown chunk is shown in the chunk output rather than, for example, the RStudio Viewer or the Plots pane. Console output (including warnings and messages) appears both at the console and in the chunk output.

Reporting

You can clear an individual chunk's output by clicking the **X** button in the upper right corner of the output, or collapse it by clicking the chevron:



Clear R-code chunk output.

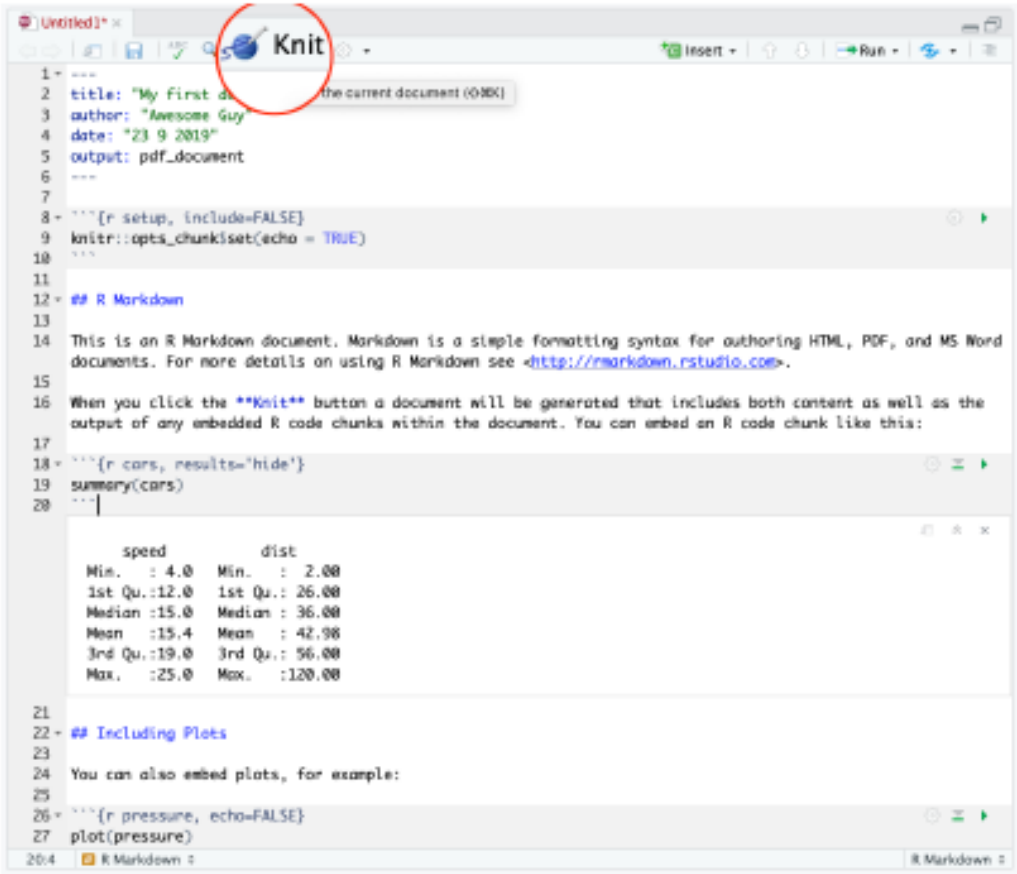
INFOX
educación sin fronteras

Reporting

Compile an RMarkdown file

The `rmarkdown` package will call the `knitr` package. `knitr` will run each chunk of R code in the document and append the result of the code in the document next to the code chunk.

To produce a complete report containing all text, code, and results, click "Knit" (or use the shortcut `Ctrl/Cmd + Shift + K`).



Compile an RMarkdown file.

Reporting

When you knit the document, `rmarkdown` sends the `.Rmd` file to `knitr`. The `knitr` pacakge will run each chunk of R code in the document and append the result of the code in the document next to the code chunk. The file generated by `knitr` is then processed by `pandoc` (http://pandoc.org/) which is responsible for creating the finished file. The advantage of this two step workflow is that you can create also create `.html` and `.docx` output formats.

Reporting

Formatting tables

There are several packages for producing tables, including `xtable`, `Hmisc`, `stargazer`, kableExtraandpander' (Note: not all packages are compatible with every output format).

For example, the `kable()`-function from the package `kableExtra` takes a matrix or data frame as input and returns it into a nicely formatted table for use with RMarkdown:

```
> library(kableExtra)
> # create a matrix like input object
> df <- summary(cars)
> kable(df, format = "latex", booktabs = TRUE, caption = "Demo Table") %>%
+   kable_styling(latex_options = c("striped", "hold_position"),
+                 full_width = FALSE) %>%
+   add_footnote(c("table footnote"))
```

Table 4: Demo Table

speed	dist
Min. : 4.0	Min. : 2.00
1st Qu.:12.0	1st Qu.: 26.00
Median :15.0	Median : 36.00
Mean :15.4	Mean : 42.98
3rd Qu.:19.0	3rd Qu.: 56.00
Max. :25.0	Max. :120.00

[a] table footnote

Reporting

Whereas stargazer creates summary tables taking the raw data as input:

```
> library(stargazer)
> stargazer(cars, type = "latex", title = "Demo Table", digits = 2,
+           summary.stat = c("mean", "sd", "median", "min", "max"))
```

% Table created by stargazer v.5.2.2 by Marek Hlavac, Harvard University. E-mail: hlavac at fas.harvard.edu % Date and time: Mo, Sep 30, 2019 - 22:21:38

Table 5: Demo Table

Statistic	Mean	St. Dev.	Median	Min	Max
speed	15.40	5.29	15	4	25
dist	42.98	25.77	36	2	120

¡GRACIAS!

INFOX
educación sin fronteras