Introduction to R

Master course

October 5th, 2016

Sarah Bonnin
CRG Bioinformatics Core Facility

Centre for Genomic Regulation

Barcelona Biomedical Research Park

Today - outline

- R Basics
- R Studio
- Input and Output
- Library / packages
- Graphing in R: ggplot2
- R Markdown
- Writing functions in R

R Basics

What is R?

- Used for **data manipulation**, **calculation** and **graphical display.**

- **Open source !**

https://www.r-project.org/

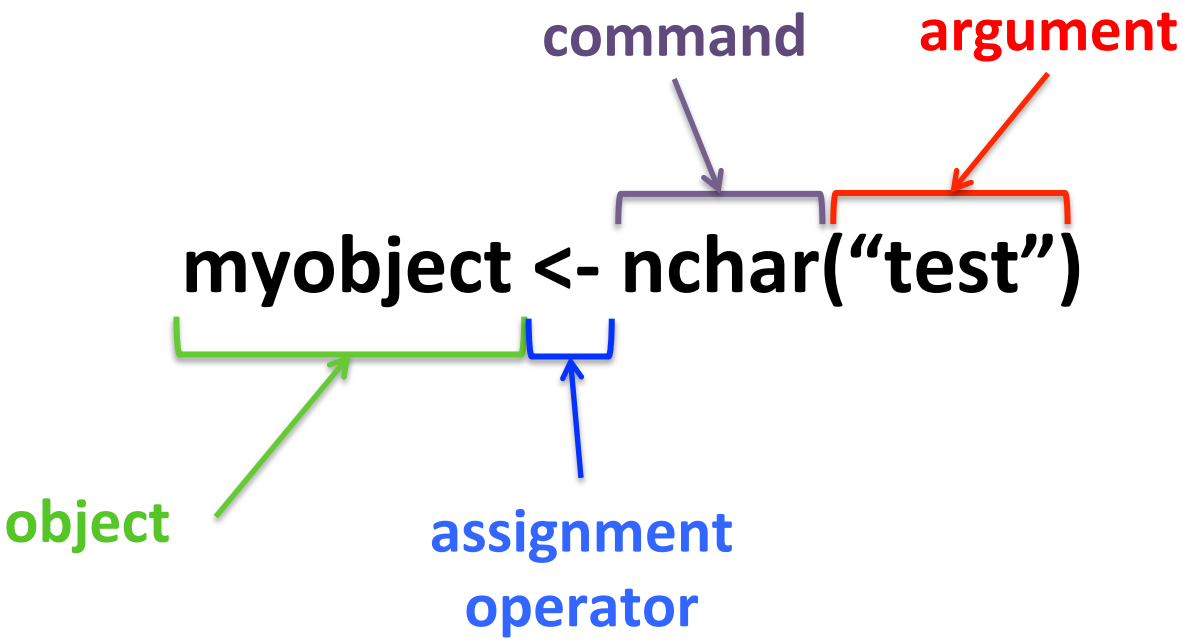
- **Interactive, flexible**

- **Very active community of developers and users!**

R base syntax

a <- function(arguments)

example:



Trying a few commands in the terminal

```
 getwd()
[1] "/Users/sbonnin”

setwd("Desktop/")

my_directory <- "R_introduction”

dir.create(my_directory)

setwd("R_introduction/")

getwd()
[1] "/Users/sbonnin/Desktop/R_introduction"
```

R syntax

- Case sensitive
- Comment lines start with **#**
- Commands separated by **;** or a new line

```
> my_directory
[1] "R_introduction"
>
>
> my_Directory
Error: object 'my_Directory' not found
>
>
> # my_Directory
>
>
> getwd(); 2+4
[1] "/Users/sbonnin/R_introduction"
[1] 6
```

Getting help on a function

- help(**getwd**)
- ?**getwd**

R Studio

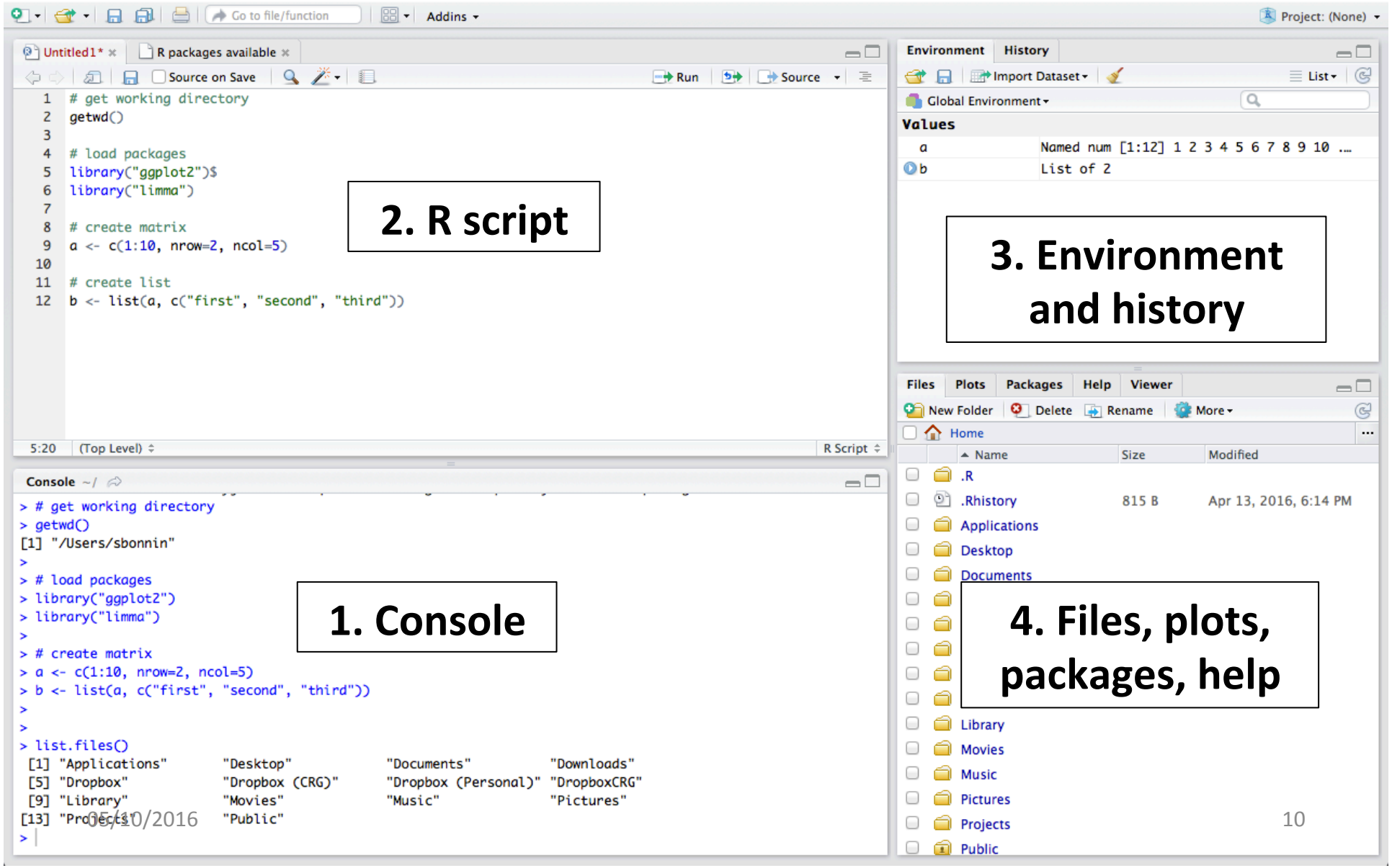
R studio?

- **Free and open source** Integrated Development Environment for R

- Available for Windows, Mac OS and Linux

R Studio

Screen: 4 windows



Data Structures

Vectors

Matrices

Data frames

List

Data Types

- Every object has a **data type** that tells what sort of value it is:
  - Numeric (Numbers)
  - Character (Text)
  - Logical (True / False)

Vectors

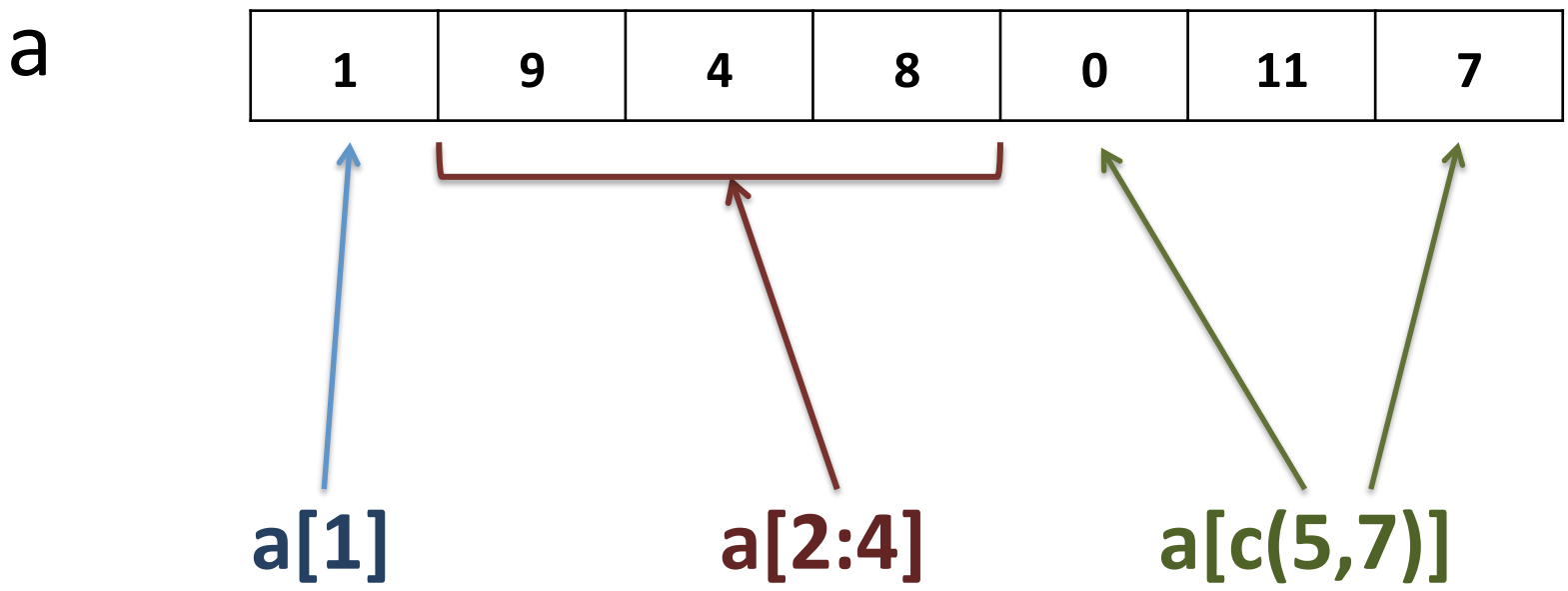
- Sequence of data elements of the same type

- Assignment of values to vector using the **c** command (combining elements)

```
a <- c(1, 9, 4, 8, 0, 11, 7)
```

Vectors

- Fetch elements of a vector:



Exercise 1:
Vector manipulation

Exercise 1

Vector manipulation

- Create vector **x** of 1000 random numbers.
  - Use **rnorm** function (random generation of numbers from a normal distribution)
- Compute the mean, median, minimum and maximum of this vector.
  - Functions: **mean**, **median**, **min**, **max**.
- Create **y**, vector containing the first 100 elements of **x**.
  - x[?]
- Compute summary statistics of **y** using the **summary** function.

Matrices

- A matrix is a vector of **2 dimensions**

- All columns in a matrix must have :
  - the same **type** (numeric, character, logical)
  - the same **length**

```
b <- matrix(c(1, 0, 34, 44, 12, 4),
            nrow=3,
            ncol=2)
```

Matrices

- Fetch rows, columns or single elements of a matrix:

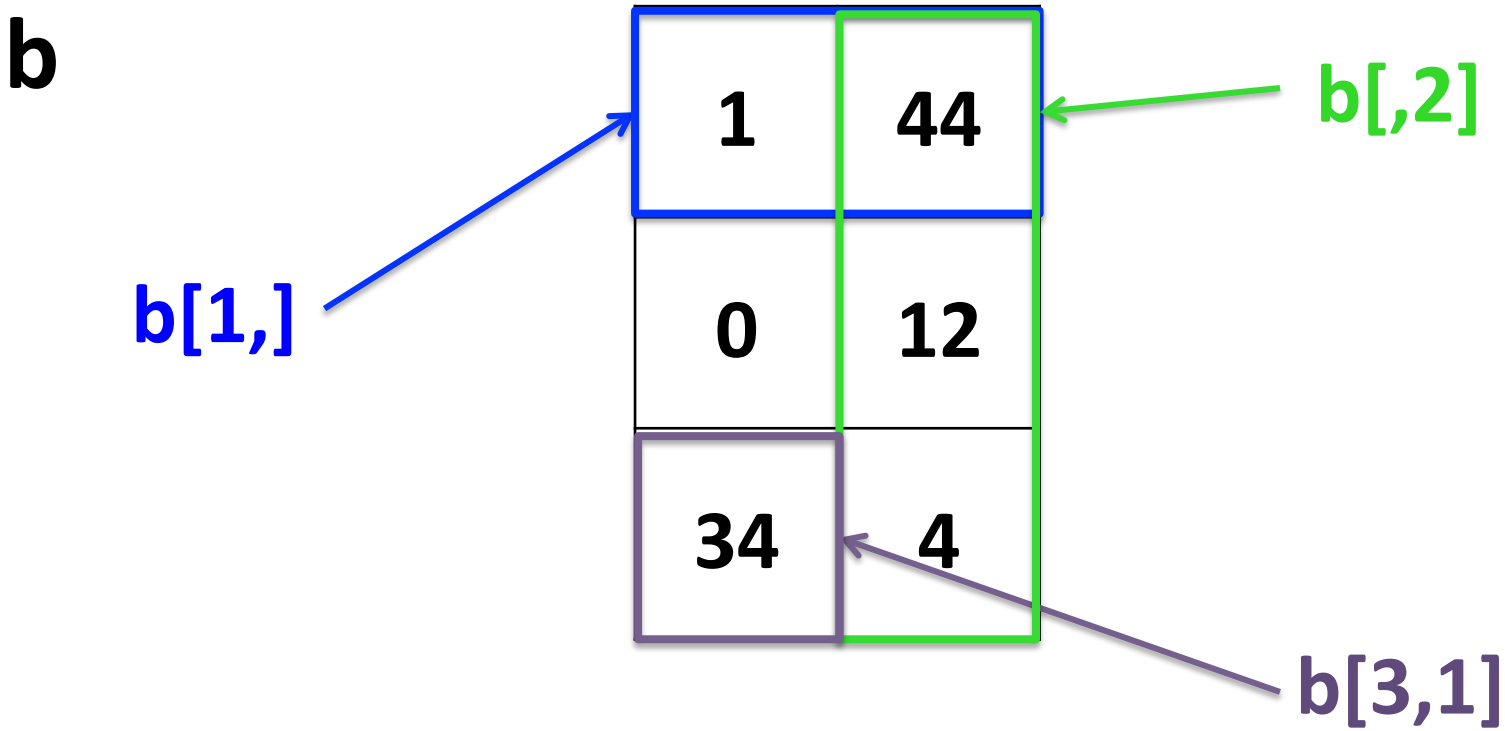
b[,]

row indices

column indices

Matrices

- Fetch rows, columns or single elements of a matrix:



Data frames

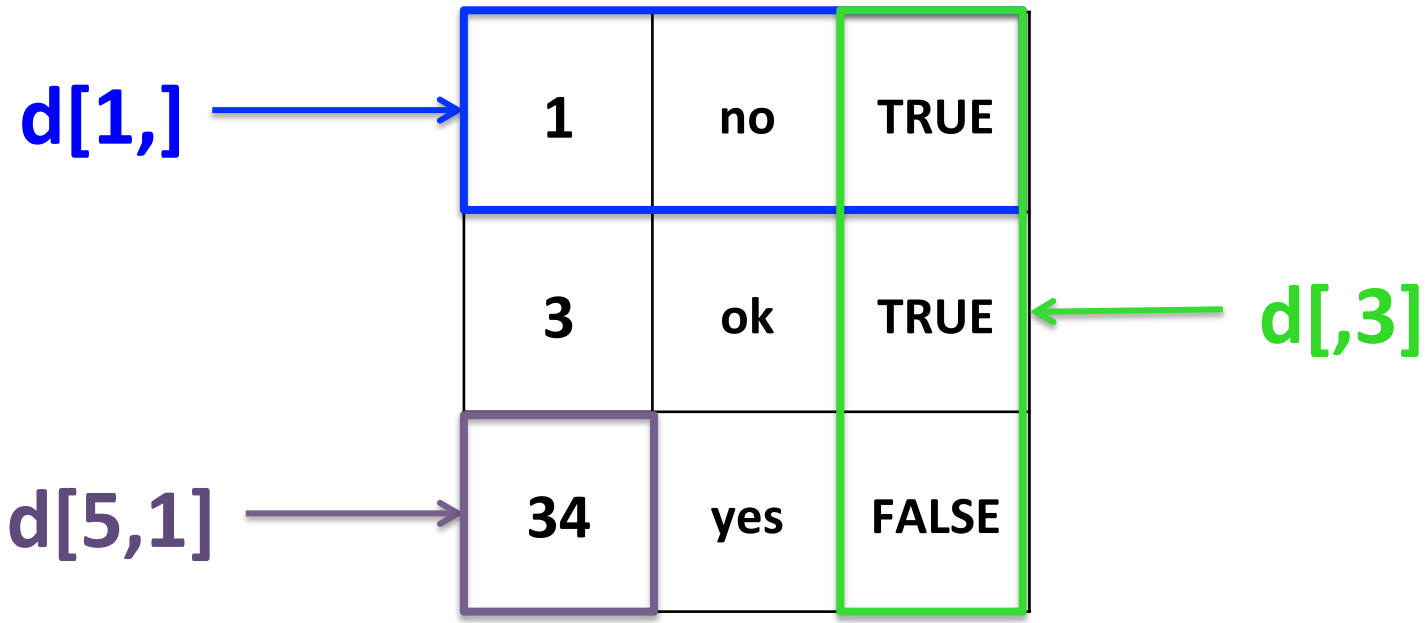
- Structures of 2 dimensions.
- More general than matrices.
- Different columns can have **different types** but must have the **same length.**

```
d <- data.frame(c(1, 3, 34),
                c("no", "ok", "yes"),
                c(TRUE, TRUE, FALSE))
```

Data frames

- Fetch rows, columns or single elements of a data frame:

d



Dimension names

- Column and / or row names can be added to matrices and data frames:

```
colnames(d) <- c("numb", "charac", "logic")
rownames(d) <- c("row1", "row2", "row3")
```

Lists

- Linear structures.

- A component of a list can be **any data structure:**

  → matrix, vector, data frame, list…

Lists

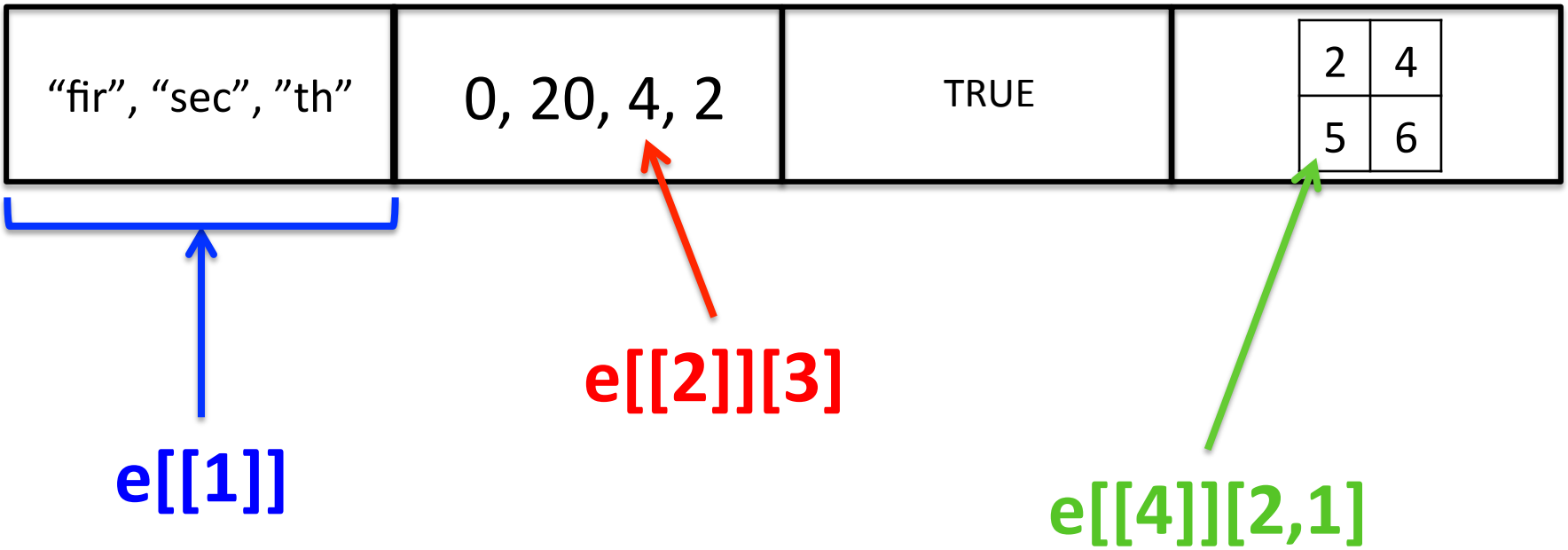
- Create a list:

```
e <- list()    # Empty list that can be filled later on

e <- list(c("fir", "sec", "th"),    # 1st element of the list
          c(0, 20, 4, 2),    # 2d element of the list
          TRUE,    # 3d element of the list
          matrix(c(2, 5, 4, 6),    # 4th element of the list
          nrow=2, ncol=2))
```

Lists

- Accessing elements of a list **e**:



Input and Output
File manipulation

Writing and reading a text file

- Let's start from a simple matrix

```
b <- matrix(c(1:6), nrow=3)
```

- Write this matrix to a file

```
write.table(b, "matrixb.txt")
```

… and read it back!

```
b <- read.table("matrixb.txt")
```

Producing graphs

- Graphs can be saved into many formats, including: **pdf, jpeg, bmp, tiff**.
- Open the file:

```
pdf("my_graph.pdf")
```

- Produce a graph:

```
plot(1:10)
```

- And close it:

```
dev.off()
```

Exercise 2:
data structures + I/O

Exercise 2

Data structures + I/O

- Create a data frame from this shopping list (save it in object **shop**):

Yogurt	6
Sugar	1
Apple	4

- Add column names to the data frame:
  respectively **item** and **quantity**.

- Write this data frame to file **shopping_list.txt:**
  - without quotes
  - without row names
  - with column names
  - fields separated with tab

Libraries / packages

Library/packages

- **Packages** are collections of R functions, data, and compiled code in a well-defined format.

- The directory where packages are stored is called the **library**.

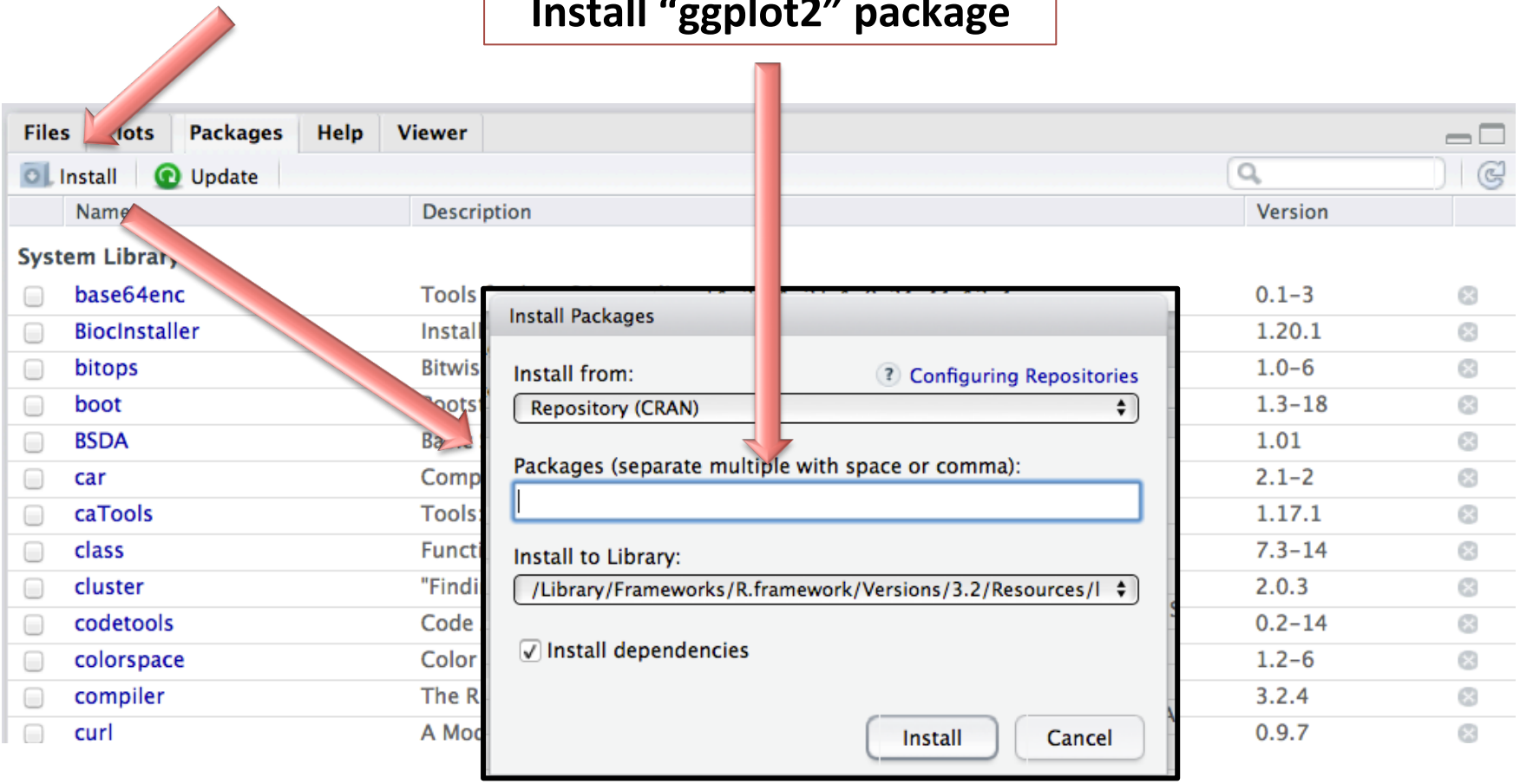
Definitions from http://www.statmethods.net/interface/packages.html

Standard packages

- About 25 standard packages are supplied with R by default (example: base, stats, graphics).
- On October 3d 2016, **9278 packages** were available!
- Anyone can contribute to the R package repository

Install a package with R studio

Install "ggplot2" package



Listing functions from packages

- **ls("package:ggplot2")**

```
 [1] "%+%"                    "%+replace%"
 [3] "Coord"                  "CoordCartesian"
 [5] "CoordFixed"             "CoordFlip"
 [7] "CoordMap"               "CoordPolar"
 [9] "CoordQuickmap"          "CoordTrans"
[11] "Geom"                   "GeomAbline"
[13] "GeomAnnotationMap"      "GeomArea"
[15] "GeomBar"                "GeomBlank"
```

…

Graphing in R:
short introduction to ggplot2

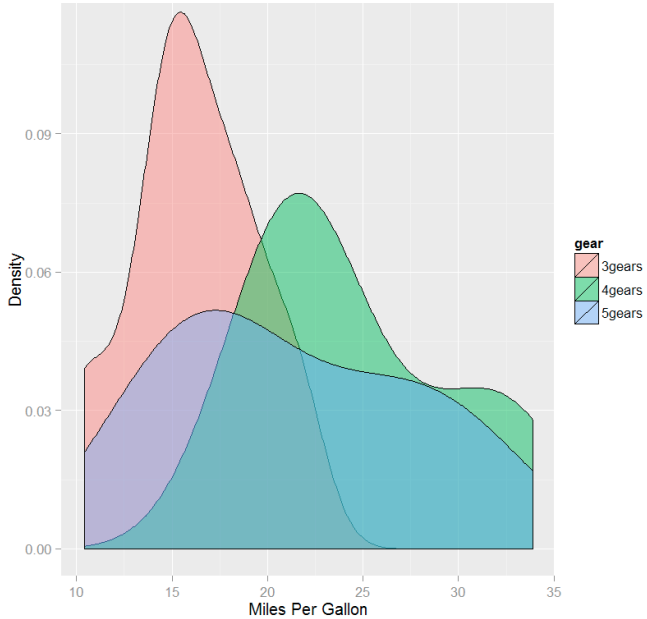
ggplot2

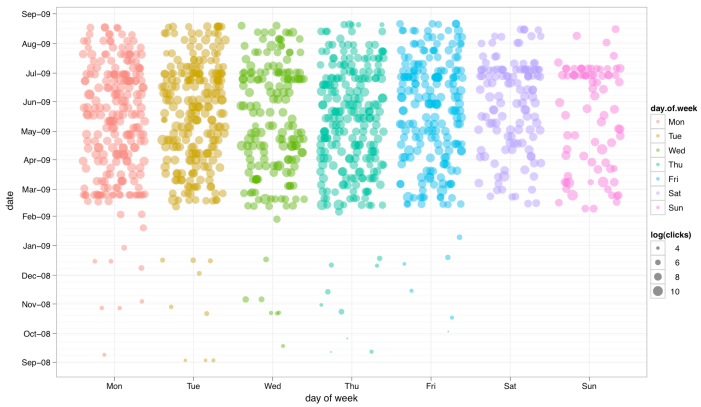
- Graphing package inspired by the Grammar of Graphics seminal work of Leland Wilkinson.

- A tool that enables to concisely describe the components of a graphic.

ggplot2

- Flexible
- Customizable
- Pretty
- Well documented





ggplot2

- Base layer to plot two variables

```
ggplot(dataframe, aes(x, y))
```

- Add layer specifying what kind of plot you want.

Example of a scatter plot:

```
ggplot(dataframe, aes(x, y)) + geom_point()
```

Exercise 3:
ggplot2

Exercise 3

ggplot2

- Create a data frame containing columns **x** and **y**, each a random generation of 100 normal distribution values.
  - Remember **rnorm** function.

- Produce a scatter plot using variable **x** and **y** from the data frame.

- Save plot in a **jpeg** file.

Dataset

- **diamonds** data set in ggplot2 package

- **dim**(diamonds); **colnames**(diamonds); **head**(diamonds)

- subset diamonds:

```
diams2 <- diamonds[sample(rownames(diamonds), 1000), ]
```

Scatter plots

A scatter plot is a type of plot or mathematical diagram using Cartesian coordinates to display values for typically two variables for a set of data.

[Wikipedia]

Scatter plots

- p <- **ggplot**(diams2, aes(carat, price, **color**=cut)) + **geom_point**()

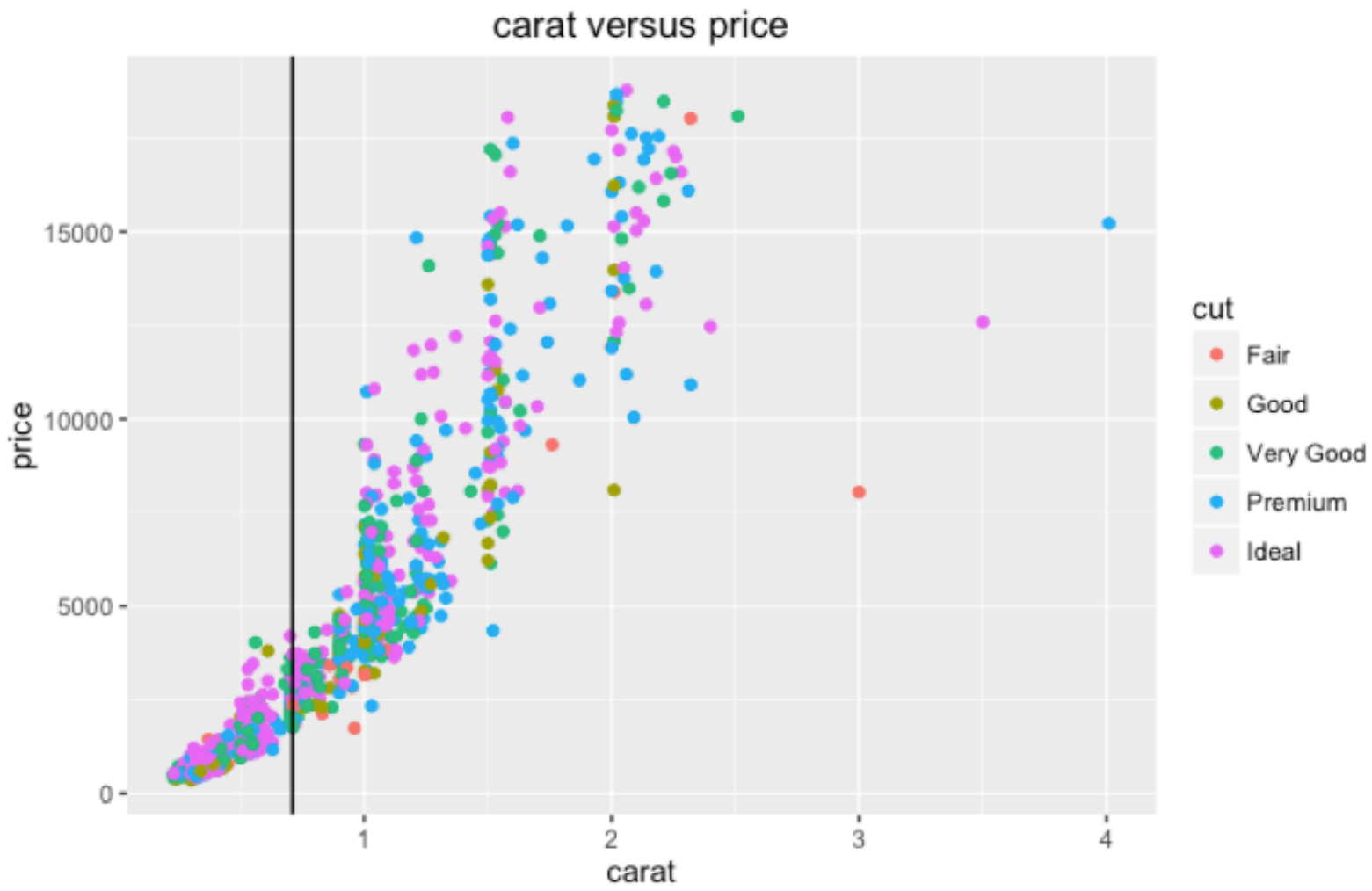
- add title with **ggtitle**

  p2 <- p + **ggtitle**("carat versus price")

- draw a vertical line at the median carat value with **geom_vline**

  p3 <- p2 + **geom_vline**(xintercept=median(diams2$carat))

Scatter plots



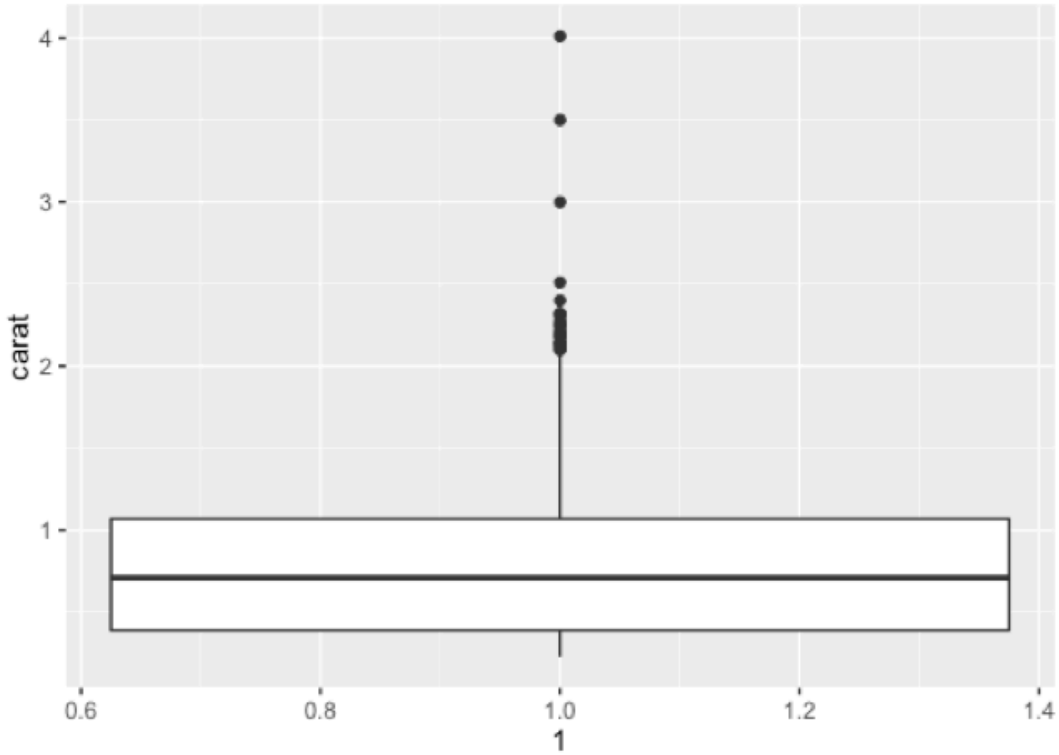
Boxplots

In descriptive statistics, a boxplot is a convenient way of graphically depicting groups of numerical data through their quartiles.

[Wikipedia]

Boxplots

- **ggplot**(diams2, aes(x=1, y=carat)) + **geom_boxplot**()

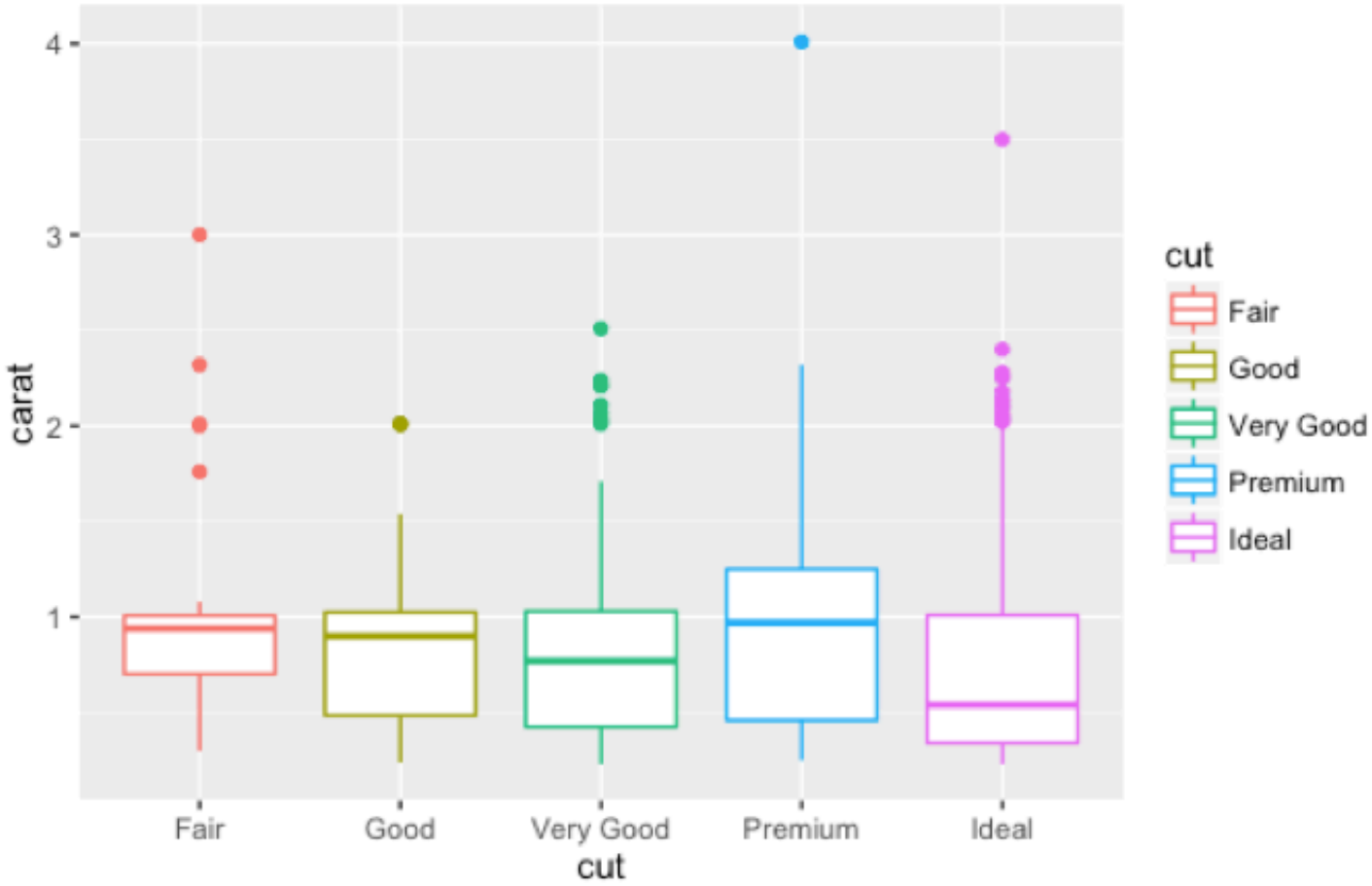


Boxplots

- p <- **ggplot**(diams2, aes(x=cut, y=carat)) + **geom_boxplot**()

- add colors to the boxes lines with **color** in aes
  - p <- **ggplot**(diams2, aes(x=cut, y=carat, **color**=cut)) + **geom_boxplot**()

Boxplots


4
3
2
1
carat
Fair
Good
Very Good
Premium
Ideal
cut
cut
Fair
Good
Very Good
Premium
Ideal

Barplots

A bar chart or bar graph is a chart that presents grouped data with rectangular bars with lengths proportional to the values that they represent. [Wikipedia]

Barplots

- p <- **ggplot**(diams2, aes(x=cut)) + **geom_bar**()

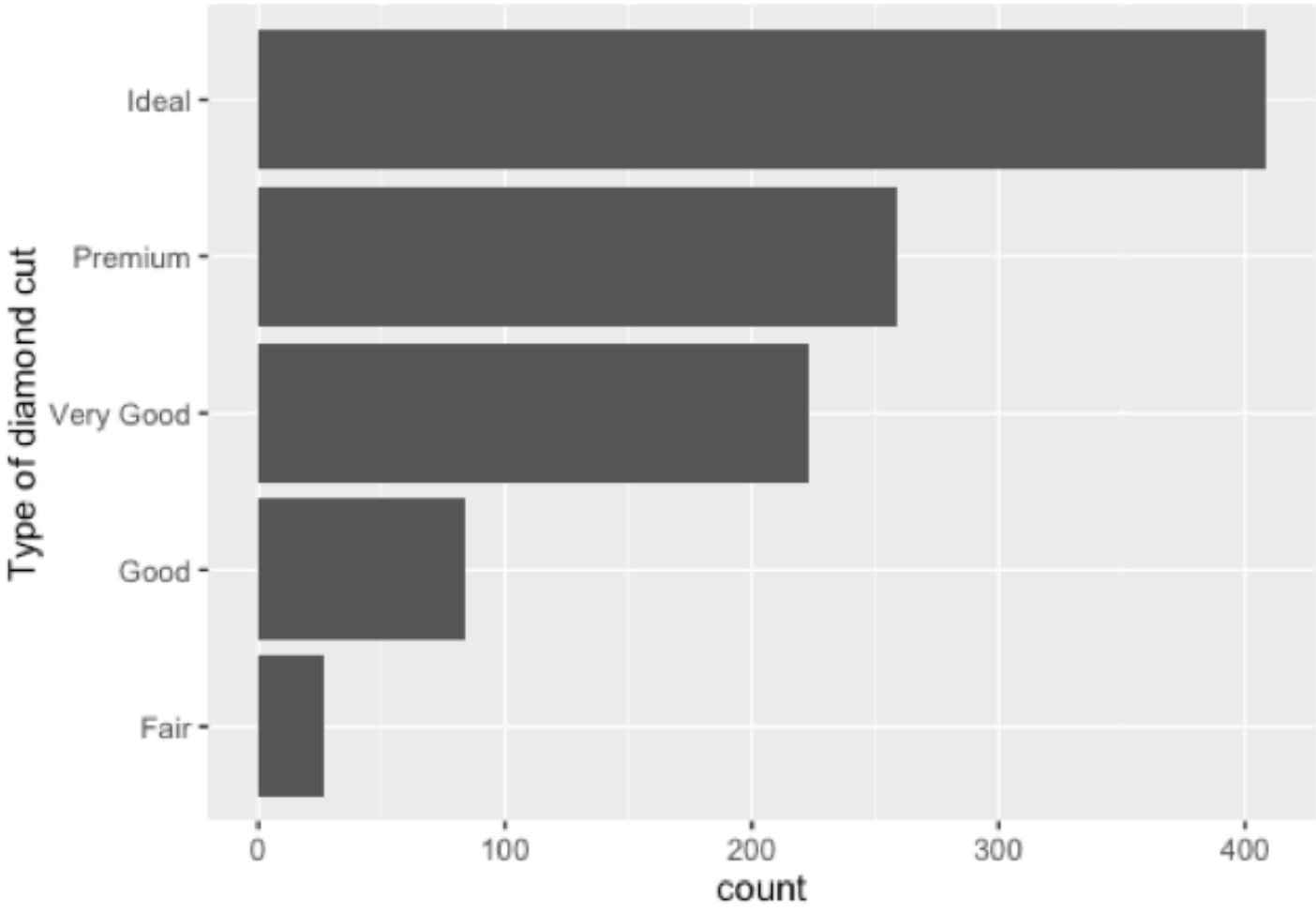
- Change x axis label with **scale_x_discrete**:

  p2 <- p + **scale_x_discrete**(name="Type of diamond cut")

- Swapping x and y axis with **coord_flip**():

  p3 <- p2 + **coord_flip**()

Barplots


Ideal
Premium
Very Good
Good
Fair
Type of diamond cut
0
100
200
300
400
count

Histograms

A histogram is a plot that lets you discover, and show, the underlying frequency distribution of a set of continuous data.

[Wikipedia]

Histograms

- **ggplot**(diams2, aes(x=price)) + **geom_histogram**()

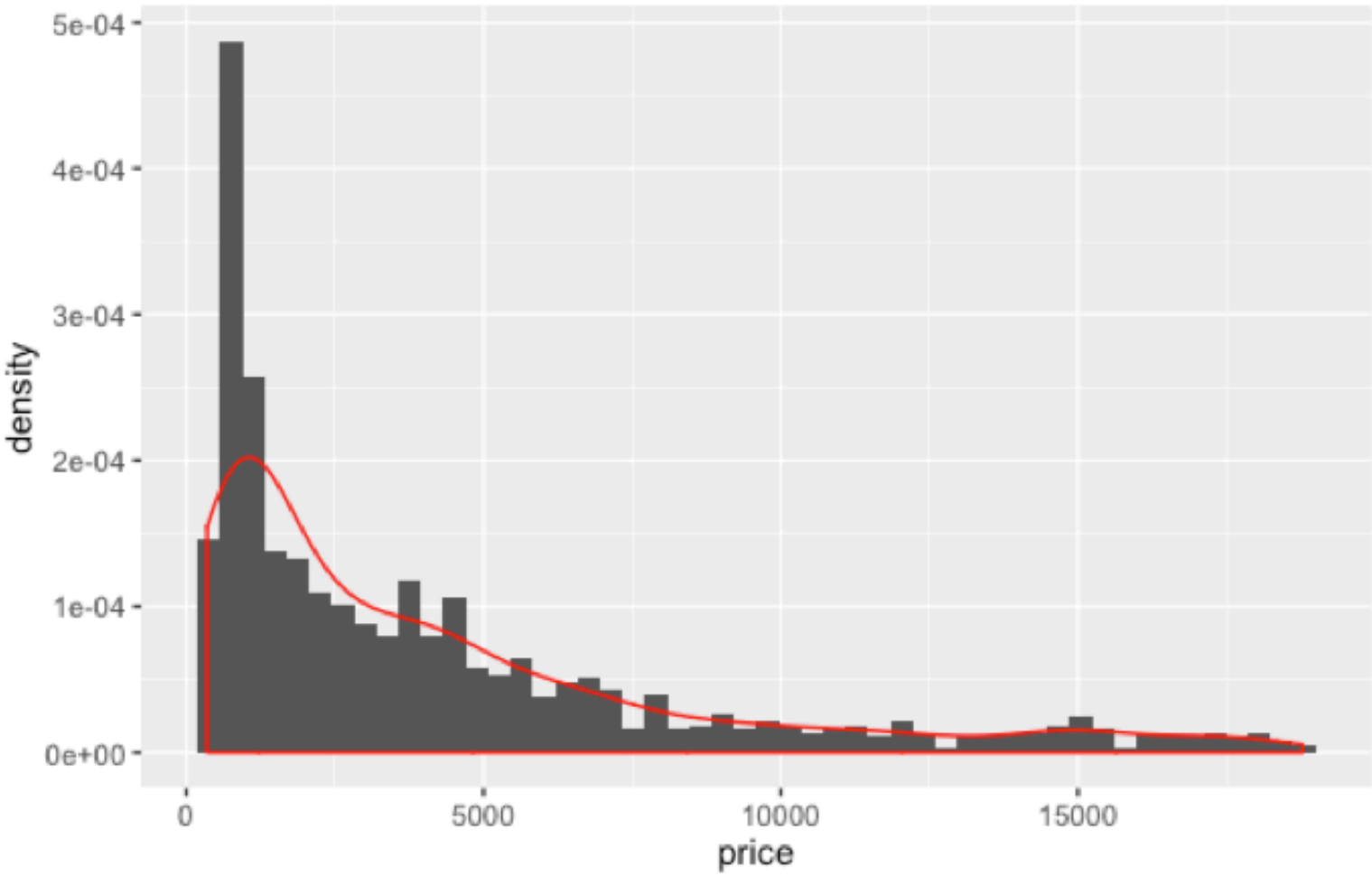
- Increase resolution with **bins** option:

  ggplot(diams2, aes(x=price)) + **geom_histogram**(**bins**=50)

- Add a density curve:

```
ggplot(diams2, aes(x=price))
    + geom_histogram(aes(y=..density..), bins=50)
    + geom_density(color="red")
```

Histograms


5e-04
4e-04
3e-04
2e-04
1e-04
0e+00
density
0
5000
10000
15000
price

RMD Markdown

RMD: R MarkDown

- Format for writing **reproducible, dynamic reports** with R.

- Work directly inserted into formatted documents (HTML, PDF and Word)

- Easy to use!

RMD: R MarkDown

cmd.R* | R packages available | Test_rmd.Rmd

Knit Word

```
---
title: "Test_rmd"
author: "Sarah Bonnin"
date: "14 Apr 2016"
output: word_document
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```

R Markdown

This is an R Markdown document. Markdown is a si
details on using R Markdown see <http://markdow

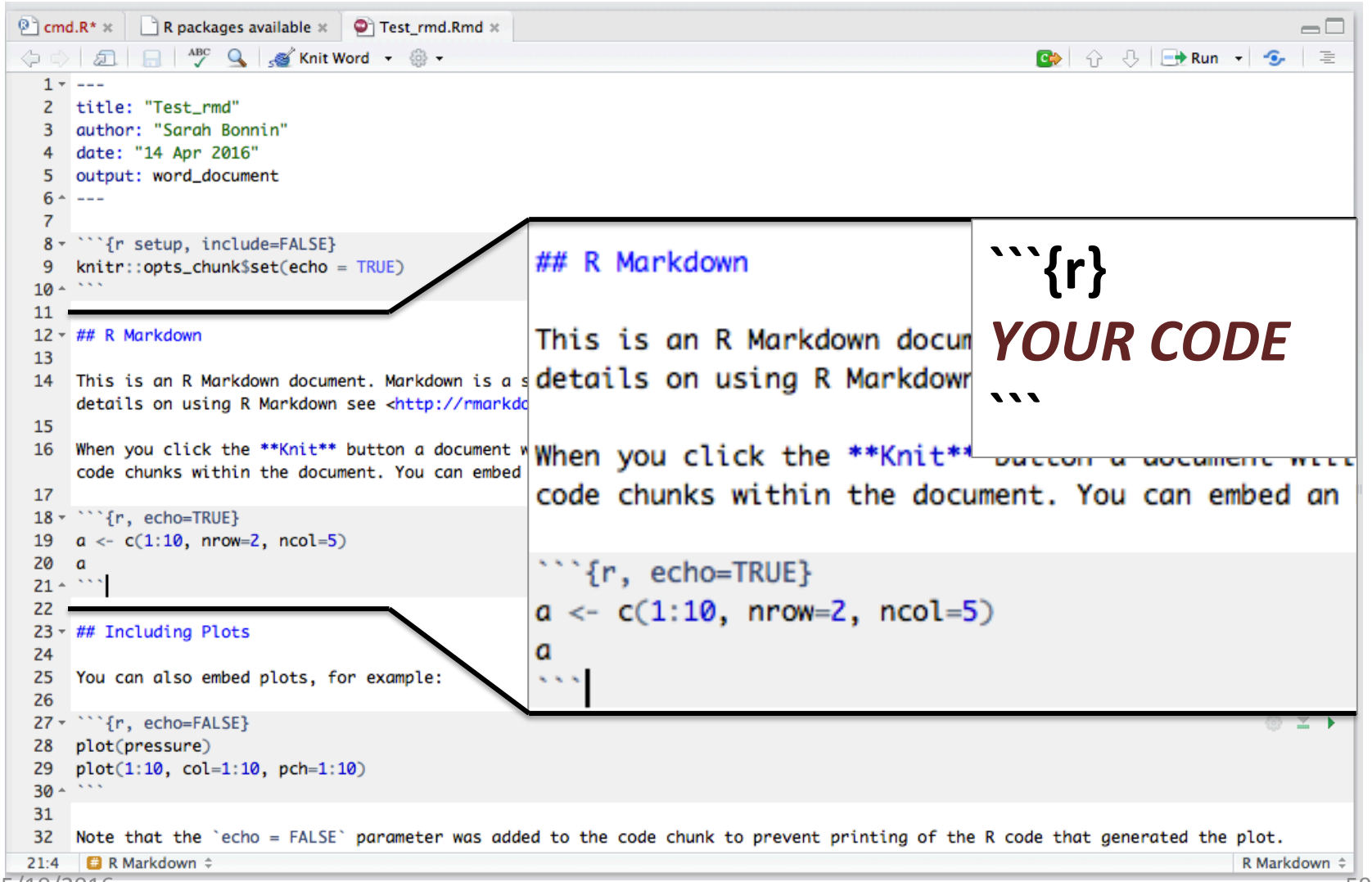
When you click the **Knit** button a document wi
code chunks within the document. You can embed a

```{r, echo=TRUE}
a <- c(1:10, nrow=2, ncol=5)
a
```

Including Plots

You can also embed plots, for example:

```{r, echo=FALSE}
plot(pressure)
plot(1:10, col=1:10, pch=1:10)
```

Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.
```

21:4 R Markdown | R Markdown

RMD: R MarkDown



RMD: R MarkDown

cmd.R* | R packages available | Test_rmd.Rmd

Knit Word | Run

```
---
title: "Test_rmd"
author: "Sarah Bonnin"
date: "14 Apr 2016"
output: word_document
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```

R Markdown

This is an R Markdown document. Markdown is a simple forma
details on using R Markdown see <http://rmarkdown.rstudio.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```{r, echo=TRUE}
a <- c(1:10, nrow=2, ncol=5)
a
```

Including Plots

You can also embed plots, for example:

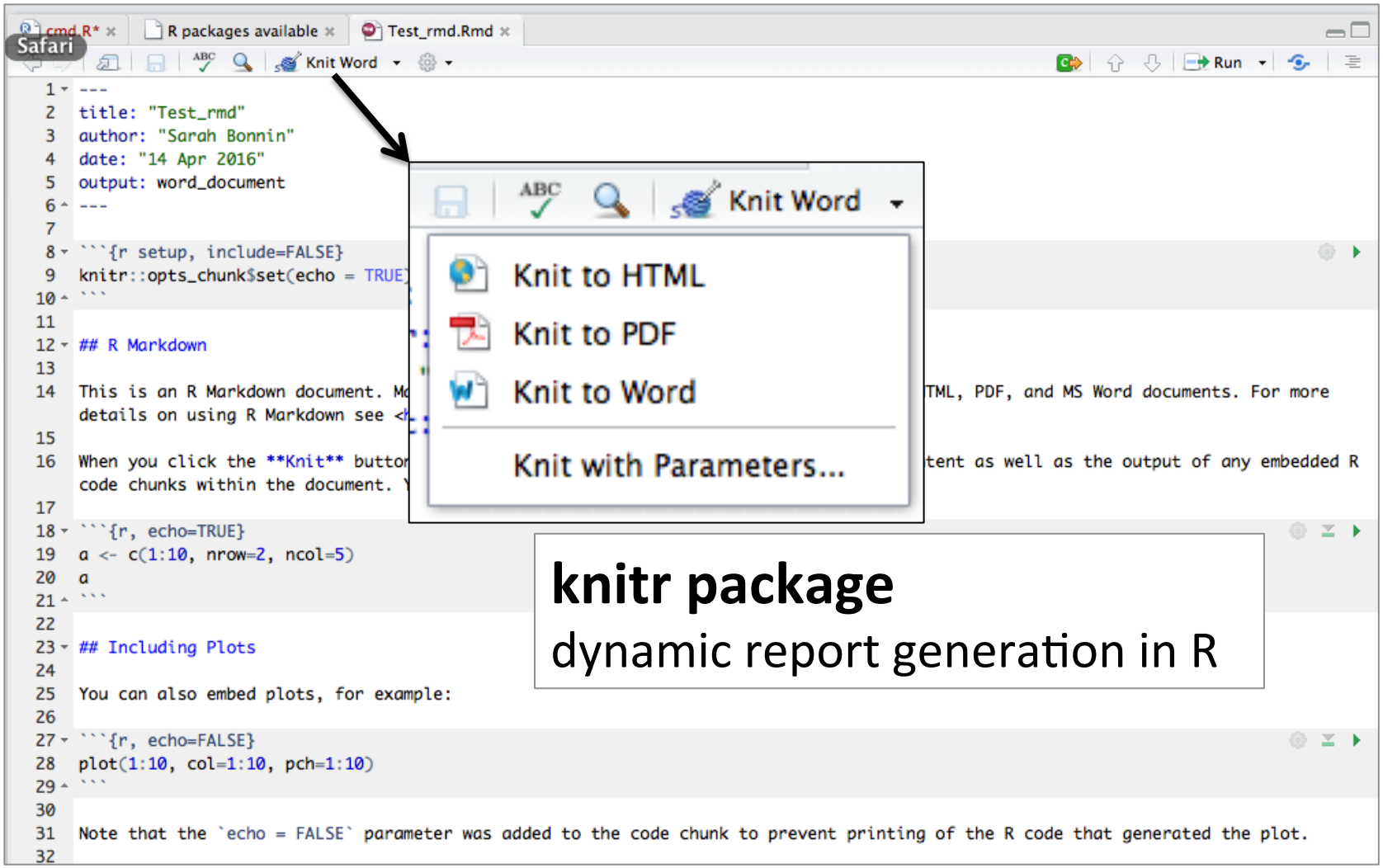
```{r, echo=FALSE}
plot(pressure)
plot(1:10, col=1:10, pch=1:10)
```

Note that the `echo = FALSE` parameter was added
```

21:4 | R Markdown | R Markdown

echo=TRUE eval=TRUE
code + output in report
echo=FALSE eval=TRUE
only output in report

```
## Including Plots

You can also embed plots, for example:

```{r, echo=FALSE}
plot(1:10, col=1:10, pch=1:10)
```
```

RMD: R MarkDown



knitr package

dynamic report generation in R

RMD: R MarkDown

Knit to HTML
→ Creates a .html file (knitr package)

exercise3.html | Open in Browser | Find

CRG Bioinformatics Unit, sarah.bonnin@crg.eu

October 5th, 2016

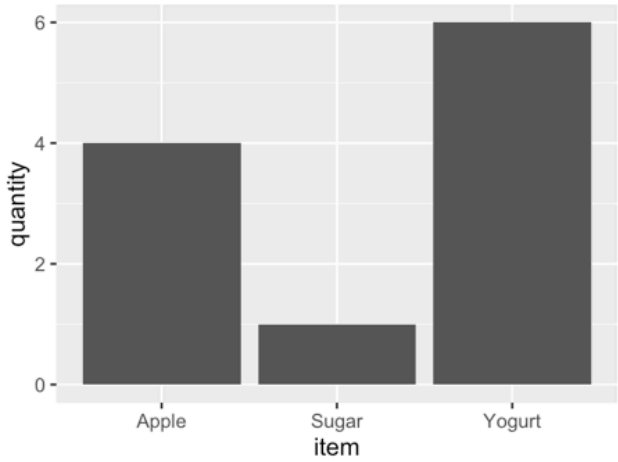
Let's see how things works

Read in the shopping list from exercise 2:

Load ggplot2

Produce a simple barplot

```
ggplot(a, aes(x=item, y=quantity)) + geom_bar(stat="identity")
```



Exercise 4:
first RMD script

RMD file

- Open file Exercise4.Rmd in R studio: it will serve as a template.
  →File in https://biocore.crg.eu/wiki/CRG_PhD_%26_Masters_Course_2016_Introduction_to_Statistics_in_R#MODULE_I._Introduction_to_R._Basic_functions_and_plotting._Oct_5.2C_2016

- In the RMD file:
  – Try to implement some of the plots produced before.
  – Play with parameters (echo, eval, fig.height, fig.height)
  – Produce a pdf file.

Writing functions in R

User-written functions

- Functions are pieces of code written **to carry out specified tasks** and be able to repeat them easily.

- R allows you to create your own functions.

Functions' structure

Function's name

Creating the function

myfunction's list of arguments

```
myfunction <- function(arg1, arg2, ... ){
    commands
    return(something)
}
```

what myfunction does

Value / object myfunction returns

Objects in functions

```
myfunction <- function(arg1){
    a <- arg1
    return(a+1)
}
```

```
> myfunction(10)
[1] 11
> a
Error: object 'a' not found
```

```
> a <- 12
> myfunction(10)
[1] 11
> a
[1] 12
```

Exercise 5: Functions

Exercise 5

Functions

- Create a new file: myfunctions.R
- Write a function that outputs the **minimum** and a **maximum** of a numerical vector.
- Save myfunctions.R
- Source **myfunctions.R** with the **source** function.
- Test your function on a simple vector, for example:

ve <- c(0, 3, -2, 1.5, 8, 4.3, -2.1)

More useful basic commands

Size/dimensions of objects

number of elements in vector, factor or list	**length(x)**
dimensions of matrix or data frame	**dim(x)**
number of rows of matrix or data frame	**nrow(x)**
number of columns of matrix or data frame	**ncol(x)**

Elementary arithmetic operators

addition	+
subtraction	-
division	/
multiplication	*
exponentiation	^ or **

Logical operators

inferior	**<**
inferior or equal	**<=**
superior	**>**
superior or equal	**>=**
equality	**==**
inequality	**!=**
intersection (“and”)	**&**
union (“or”)	**|**

Obtaining summary statistics

average/mean	**mean(x)**
median	**median(x)**
minimum	**min(x)**
maximum	**max(x)**
variance	**var(x)**
correlation	**cor(x)**

Some common arithmetic functions

natural logarithm	**log(x)**
exponential function e^x	**exp(x)**
sine	**sin(x)**
cosine	**cos(x)**
tangent	**tan(x)**
absolute value	**abs(x)**
square root	**sqrt(x)**

Objects stored in the global environment

- Listing:

ls() or **objects()**

- Removing one object from environment:

rm(x)

- Removing several objects:

rm(x, y)

- Removing all object from environment:

rm(list=ls())

Saving objects or session

- Save objects x and y into “myobjects.RData” file
  - **save(x, y, file=“myobjects.RData”)**
- Load objects x and y into current directory:
  - **load(“myobjects.RData”)**
- Save the current workspace (all objects):
  - **save.image(file=“.RData”)**

Command history

- Last 25 commands:
  - **history()**
- All previous commands
  - **history(max.show=Inf)**
- Save command history:
  - **savehistory()**
- Load command history:
  - **loadhistory()**

Information about the current session

sessionInfo()

```
R version 3.2.4 (2016-03-10)
Platform: x86_64-apple-darwin13.4.0 (64-bit)
Running under: OS X 10.9.5 (Mavericks)

locale:
[1] C/UTF-8/C/C/C/C

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] ggplot2_2.1.0

loaded via a namespace (and not attached):
[1] colorspace_1.2-6 scales_0.4.0     plyr_1.8.3       gtable_0.2.0
[5] Rcpp_0.12.4      grid_3.2.4       munsell_0.4.3
```

R version

Platform and OS version

Packages attached