Introduction to R
From ZeRo to HeRo!

Isabella Gollini

@IsabellaGollini

UCD DUBLIN

FORWARDS

Ladies

Diversity in Data Workshop
Talent Garden, Dublin City University, Ireland
22 September 2018

What can R do?

- Data Handling
- Analysis
- Reporting
- Programming

What will we do today?

1. Introduction to R and RStudio
2. Data import and handling
3. Data manipulation and summaries with *dplyr*
4. Graphics with *ggplot2*
5. RStudio Projects
6. Dynamic documents with R Markdown

The R Ecosystem

Base

base

create R objects
summaries
math functions

recommended

statistics
graphics
example data

Contributed Packages

CRAN

cran.r-project.org

main repos
~13000 pkgs

Bioconductor

OPEN SOURCE SOFTWARE FOR BIOINFORMATICS

bioconductor.org

bioinformatics
>1500 pkgs

GitHub

github.com

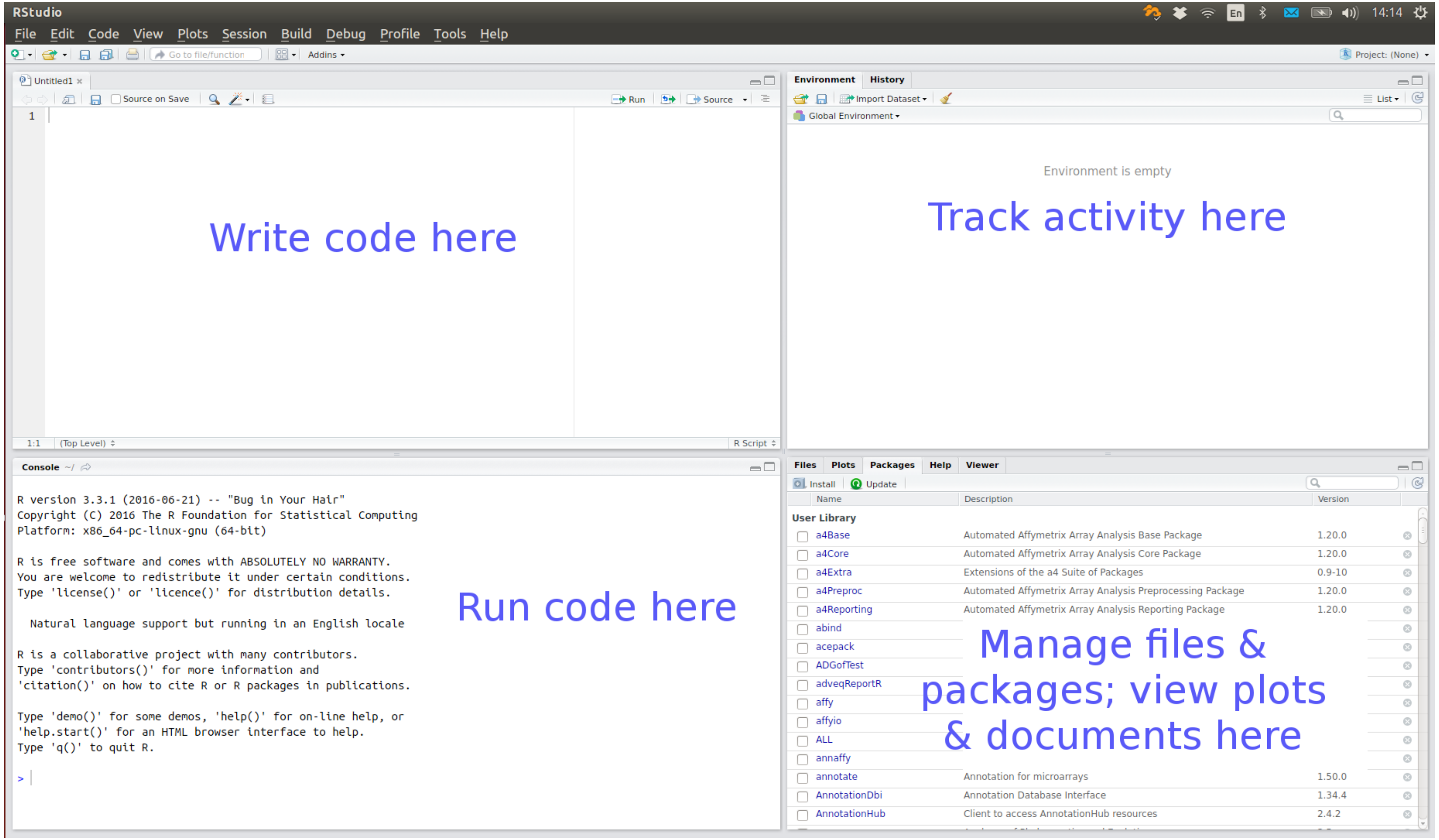
devel pkgs
GitHub-only pkgs

R Demo

We can type commands directly into the R console

```
3 + 4
?"+" # look up help for "+". You need quotes!
x <- 3 + 4 # store 3 + 4 into the object x
x # print the object R
y <- log(x) # store the natural log of x in the object y
y = log(x) # same as before using the = instead of <-
3 == 4 # == is the comparison operator for equal
3 != 4 # != is the comparison operator for not equal
?log # look up help for the function log
ls() # list of objects in the current workspace
rm(x) # remove the object x
```

RStudio IDE



Write code here

Environment is empty

Track activity here

```
R version 3.3.1 (2016-06-21) -- "Bug in Your Hair"
Copyright (C) 2016 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

Run code here

User Library

	Name	Description	Version
□	a4Base	Automated Affymetrix Array Analysis Base Package	1.20.0
□	a4Core	Automated Affymetrix Array Analysis Core Package	1.20.0
□	a4Extra	Extensions of the a4 Suite of Packages	0.9-10
□	a4Preproc	Automated Affymetrix Array Analysis Preprocessing Package	1.20.0
□	a4Reporting	Automated Affymetrix Array Analysis Reporting Package	1.20.0
□	abind		
□	acepack		
□	ADGofTest		
□	adveqReportR		
□	affy		
□	affyio		
□	ALL		
□	annaffy		
□	annotate	Annotation for microarrays	1.50.0
□	AnnotationDbi	Annotation Database Interface	1.34.4
□	AnnotationHub	Client to access AnnotationHub resources	2.4.2

Manage files & packages; view plots & documents here

R Studio Features

Features provided by RStudio include:

- syntax highlighting, code completion, smart indentation
- interactively send code chunks from editor to R
- organise multiple scripts, help files, plots
- search code and help files

RStudio Shortcuts from the R Console

RStudio provides a few shortcuts to help write code in the R console

- `↑`/`↓` go back/forward through history one command at a time
- `Ctrl`/`⌘` + `↑` review recent history and select command
- `Tab` view possible completions for part-written expression

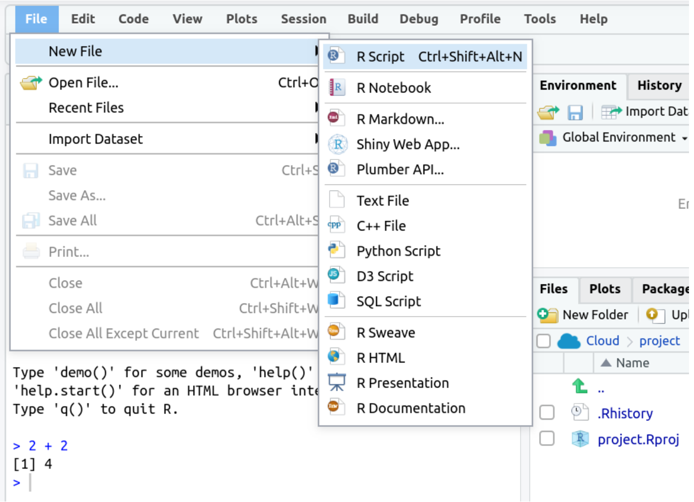
Code completion (using `Tab`) is also provided in the source editor

For more RStudio shortcuts go to `Help` > `Keyboard Shortcuts Help`

R Scripts

Rather than typing commands individually in the console window, it is often more useful to keep a record of everything you have run.

You can store commands in the R Script in the source window and run these en bloc or line by line.



```
Type 'demo()' for some demos, 'help()'
'help.start()' for an HTML browser inte
Type 'q()' to quit R.

> 2 + 2
[1] 4
>
```

Text files saved with a `.R` suffix are recognised as R code.

R Studio Demo

```r
data() # find out what standard data sets there are
plot(iris) # plot Fisher's iris data
head(iris, 4) # print the first 4 rows of the iris data
View(iris) # view the iris dataset on the viewer
summary(iris,  # summaries of the iris dataset up to the second digit
        digits = 2) # you can split the command in two lines
sum(3, 4) # do the sum of 3 and 4
log(sum(1:10)) # Sum the numbers from 1 to 10 and then takes the natural log
```

Your Turn!

1. In RStudio, open a new R Script.

2. Try typing some of the code from the R/R Studio demo and using some of the shortcuts to complete your commands and run them.

Data Structures

Data structures are the building blocks of code. In R there are four main types of structure:

- vectors and factors
- matrices and arrays
- lists
- data frames

The first and the last are sufficient to get started.

Vectors

A single number is a special case of a numeric vector. Vectors of length greater than one can be created using the concatenate function, `c`.

```r
x <- c(1, 3, 6)
```

The elements of the vector must be of the same type: common types are numeric, character and logical

```r
y <- c("red", "yellow", "green")
z <- c(TRUE, FALSE)
```

Missing values (of any type) are represented by the symbol `NA`.

Data Frames

Data sets are stored in R as *data frames*. These are structured as a list of objects, typically vectors, of the same length

```
str(iris)
```

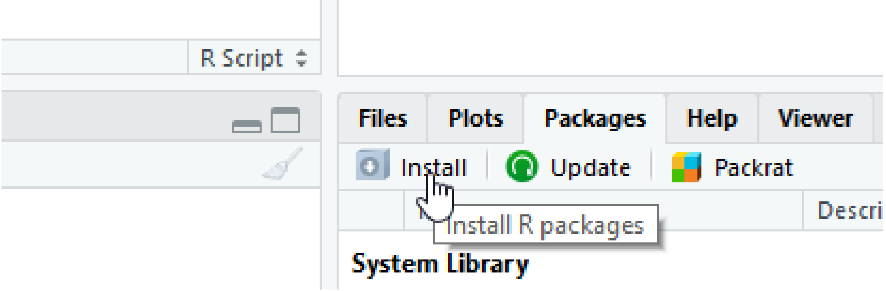
```
# 'data.frame':    150 obs. of  5 variables:
#  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
#  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
#  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
#  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
#  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

Here `Species` is a factor, a special data structure for categorial variables.

Install Packages

Most day-to-day work will require at least one contributed package.

CRAN packages can be installed from the Packages tab



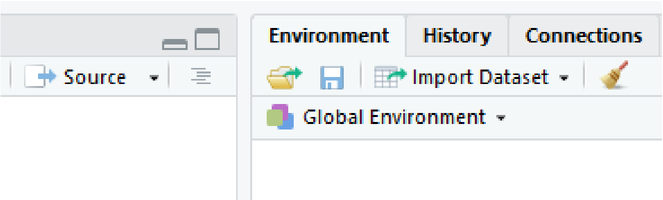
To use an installed package in your code, you must first load it from your package library

```r
library(ggplot2)
```

Sometimes an RStudio feature will require a contributed package. A pop-up will ask permission to install the package the first time, after that RStudio will load it automatically.

Data Input via Import Dataset

Using the Import Dataset dialog in RStudio



we can import files stored locally or online in the following formats:

- `.txt`/`.csv` via `read_delim`/`read_csv` from **readr**.
- `.xlsx` via `read_excel` from **readxl**.
- `.sav/.por`, `.sas7bdat` and `.dta` via `read_spss`, `read_sas` and `read_stata` respectively from **haven**.

Most of these functions also allow files to be compressed, e.g. as `.zip`.

Your Turn

1. Use the Import Dataset button, import a data set. RStudio will ask to install packages as required.

2. Try changing some of the import options to see how that changes the preview of the data and the import code.

3. Install the **skimr** package and use the `skim` function to summarise the data set.

Tibbles

The functions used by *Import Dataset* return data frames of class `"tbl_df"`, aka **tibbles**. The main differences are

	data.frame	**tibble**
Printing (default)	Whole table	10 rows; columns to fit Prints column type
Subsetting	`dat[, 1]`, `dat$X1`, `dat[[1]]` all return vector	`dat[,1]` returns tibble `dat$X1`, `dat[[1]]` return vector
Strings	Converted to factor (default)	Left as character
Variable names	Made *syntactically valid* e.g. `Full name` -> `Full.name`	Left as is use e.g. ``dat$`Full name` ``

Data Input via Code

The **rio** package provides a common interface to the functions used by *Import Dataset* as well as many others.

The data format is automatically recognised from the file extension. To read the data in as a tibble, we use the `setclass` argument.

```r
library(rio)
compsci <- import("compsci.csv", setclass = "tibble")
cyclist <- import("cyclist.xlsx", setclass = "tibble")
```

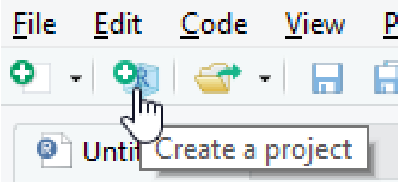
See `?rio` for the underlying functions used for each format and the corresponding optional arguments, e.g. the skip argument to `read_excel` to skip a certain number of rows.

R Studio Projects

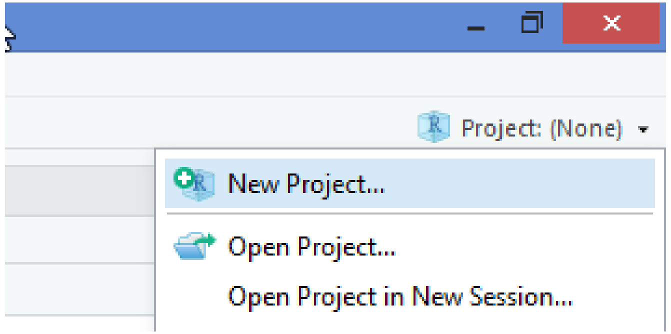
An Rstudio project is a context for work on a specific project

- automatically sets working directory to project root
- has separate workspace and command history
- works well with version control via git or svn

Create a project from a new/existing directory via the File Menu or the New Project button



Switch project, or open a different project in a new RStudio instance via the Project menu.



R Markdown Documents

R markdown documents (`.Rmd`) intersperse code chunks (R, Python, Julia, C++, SQL) with markdown text

YAML header

```
---
title: "Report"
output: html_document
---
```

Markdown text

```
## First section

This report summarises the `cars` dataset.
```

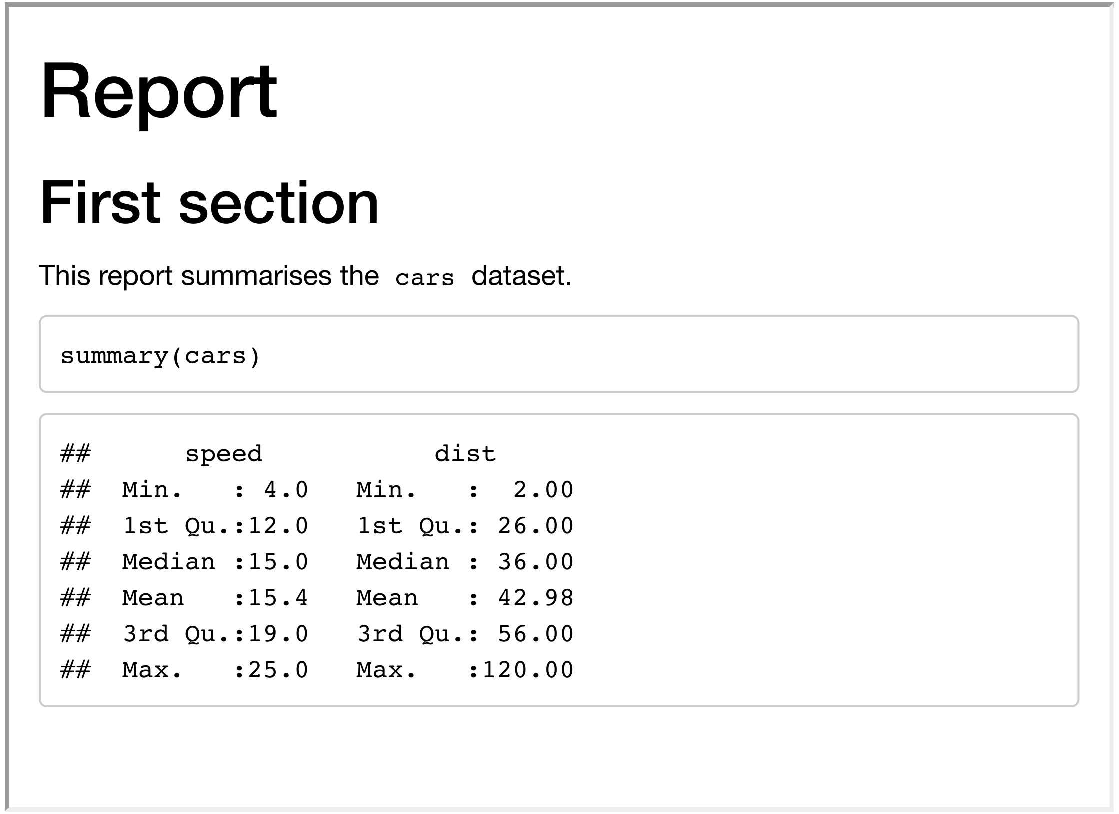
R code chunk

~~~
```{r summary-cars}
summary(cars)
```
~~~

Options can be controlled on a document or chunk level whether to show code and/or output.

Rendering

The `.Rmd` file can be rendered to produce a document (HTML, PDF, docx) integrating the code output.



Report

First section

This report summarises the `cars` dataset.

```
summary(cars)
```

```
##      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  
```

Demo - Getting Started with R Markdown

1. To create a new R Markdown document; go to `File > New File > R Markdown`
   - The first time you use R Markdown on your machine you may be asked to install some R packages; if so, press `Yes`.
2. Select the type of output document you want to create. You are able to produce `HTML` output on any computer.
3. Open the "Markdown Quick Reference": `Help > Markdown Quick Reference`

R Markdown is a very powerful tool; an extended guide with tutorials is available on the RMarkdown website: https://rmarkdown.rstudio.com/lesson-1.html

Your Turn

1. Copy the `infant` directory from GitHub to your laptop. Open `infant.Rproj`, which is an RStudio project file.
2. From the Files tab, open the `infant.Rmd` R markdown file.
3. In the chunk labelled `import-data`, write some code that will import the `infant.xlsx` file and create a tibble named `infant`.
4. Load any required packages in the `setup` chunk.
5. Run the code in the `import-data` chunk (the `setup` chunk is run automatically).
6. Use `View()` in the console to inspect the result.

Data Pre-processing

The imported data are a subset of records from a US Child Health and Development Study, corresponding to live births of a single male foetus.

The data requires a lot of pre-processing

- converting numeric variables to factors
- converting coded values to missing values
- filtering rows and selecting columns

The *dplyr* package can be used to help with many of these steps

dplyr

The *dplyr* package provides the following key functions to operate on data frames

- `filter()`
- `arrange()`
- `select()` (and `rename()`)
- `distinct()`
- `mutate()` (and `transmute()`)
- `summarise()`

`filter()`

`filter()` selects rows of data by criteria

```r
library(dplyr)
infant <- filter(infant, smoke != 9 & age > 18)
```

Building block	R code
Binary comparisons	>, <, ==, <=, >=, `!=`
Logical operators	or | and `&`, not `!`
Value matching	e.g. `x %in% 6:9`
Missing indicator	e.g. `is.na(x)`

`select()`

`select()` selects variables from the data frame.

Select named columns:

```
select(infant, gestation, sex)
```

Select a sequence of columns, along with an individual column

```
select(infant, pluralty:gestation, parity)
```

Select all columns *except* specified columns

```
select(infant, -(id:outcome), -sex)
```

`mutate()`

`mutate()` computes new columns based on existing columns. Re-using an existing name replaces the old variable.

```
mutate(infant,
       wt = ifelse(wt == 999, NA, wt),
       wt = wt * 0.4536)
```

To only keep the computed columns, use `transmute()` in place of `mutate()`.

Your Turn

1. In the `select-variables` chunk, use `select()` to remove the redundant variables `pluralty`, `outcome` and `sex`, as well a characteristics of the father `drace`, ...`dwt`. Over-write the `infant` data frame.

2. The variable `gestation` gives the length of the pregnancy in days. In the `filter-variables` chunk, filter the data to exclude extremely premature babies (gestation less than 28 weeks) and extremely late babies (gestation more than 52 weeks).

3. The value `999` has been used to code an unknown value in the `ht` variable. In the `mutate-variables` chunk, use `mutate` to update `ht`, replacing `999` with `NA`.

Chaining

We can use `%>%` to pipe the data from one step to the next

```
infant <- infant %>%
    filter(smoke != 9 & age > 18) %>%
    select(-(id:outcome), -sex) %>%
    mutate(wt = ifelse(wt == 999, NA, wt),
           wt = wt * 0.4536)
```

Any function with data as the first argument can be added to the data pipeline.

`summarise()`

`summarise()` function is for computing single number summaries of variables, so typically comes at the end of a pipeline or in a separate step

```
stats <- summarise(infant,
                   `Mean mother's weight (kg)` = mean(wt, na.rm = TRUE),
                   `Sample size` = sum(!is.na(wt)),
                   `Total sample size` = n())
stats
```

```
# # A tibble: 1 x 3
#   `Mean mother's weight (kg)` `Sample size` `Total sample size`
#                         <dbl>         <int>               <int>
# 1                        58.4          1167                1203
```

In an R markdown document we can convert the result to a markdown table using `kable` from the **knitr** package

```
kable(stats, align = "ccc")
```

Grouped Operations

Grouping can be set on a data frame using `group_by`. This is most useful `summarise()`

```
infant <- infant %>% filter(race != 10) %>%
    mutate(`Ethnic group` = recode(race, `6` = "Latino", `7` = "Black", `8` = "Asian",
                                   `9` = "Mixed", .default = "White"))
res <- infant %>% group_by(`Ethnic group`) %>%
    summarise(`Mean weight (kg)` = mean(wt, na.rm = TRUE))
kable(res, align = "lc")
```

Ethnic group	Mean weight (kg)
Asian	49.84
Black	62.63
Latino	54.12
Mixed	58.46
White	57.86

Your Turn

`cut()` (in the **base** package) can be used to create a factor from a continuous variable, e.g.

```
cut(c(0.12, 1.37, 0.4, 2.3), breaks = c(0, 1, 2, 3))
# [1] (0,1] (1,2] (0,1] (2,3]
# Levels: (0,1] (1,2] (2,3]
```

where (0, 1] is the set of numbers > 0 and ≤ 1, etc.

1. The infant birth weight `bwt` is given in ounces. In the `table-bwt` chunk use `mutate()` to create a new factor called `Birth weight (g)`, by converting `bwt` to grams through multiplication by 28.35 and then converting the result to a factor with the following categories: (1500, 2000], (2000, 2500], (2500, 3000], (3000, 3500], and (3500, 5000].
2. An infant is categorised as low weight if its birth weight is ≤ 2500 grams, regardless of gestation. Use `group_by()` to group the data by the new factor weight factor, then pipe to `summarise()` to count the number of infants in each weight category.

Plots

In RStudio, graphs are displayed in the Plots window. The plot is sized to fit the window and will be rescaled if the size of the window is changed.

Back and forward arrows allow you to navigate through graphs that have been plotted.

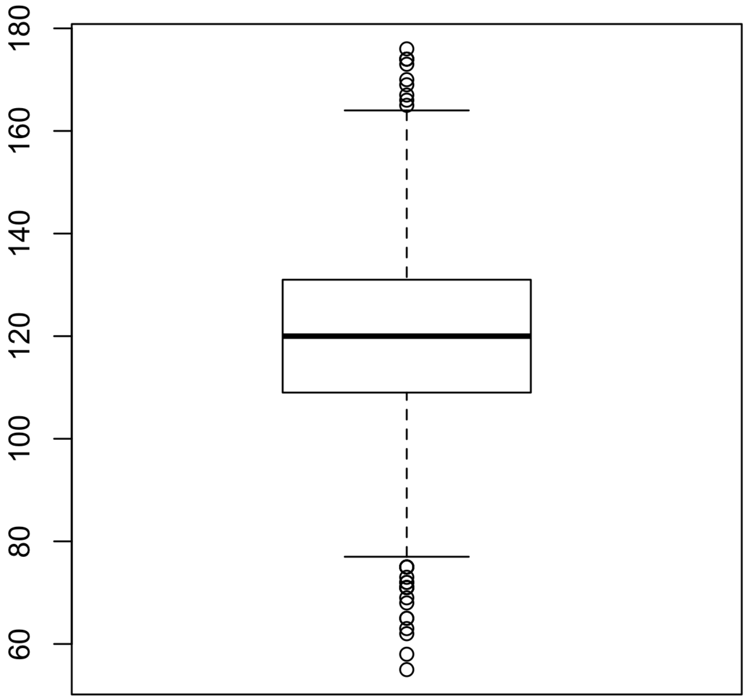
Graphs can be saved in various formats using the Export drop down menu, which also has an option to copy to the clipboard.

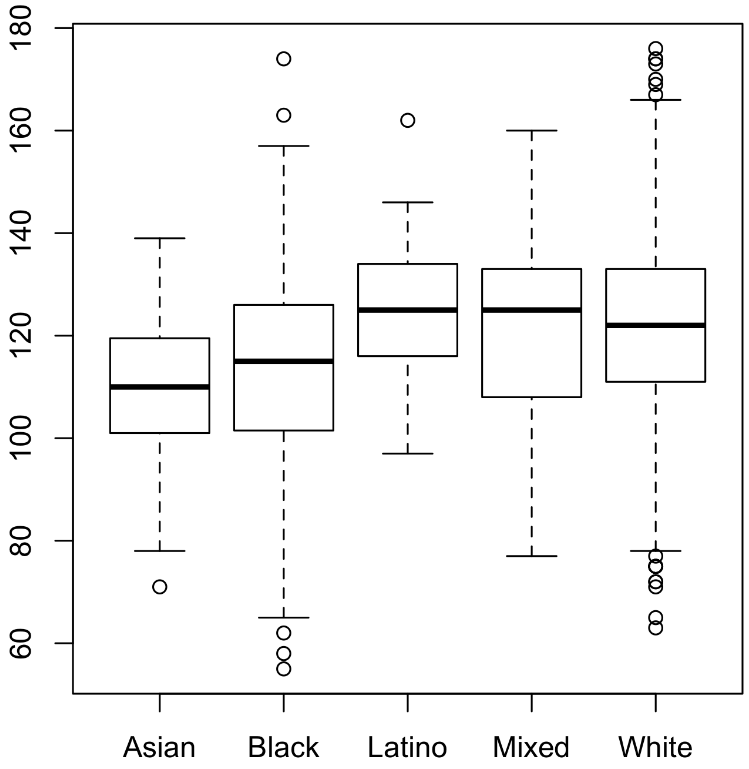
First we consider "no-frills" plots, for quick exploratory plots.

```
infant <- infant %>%
    mutate(gestation = ifelse(gestation == 999, NA, gestation))
```

Boxplots

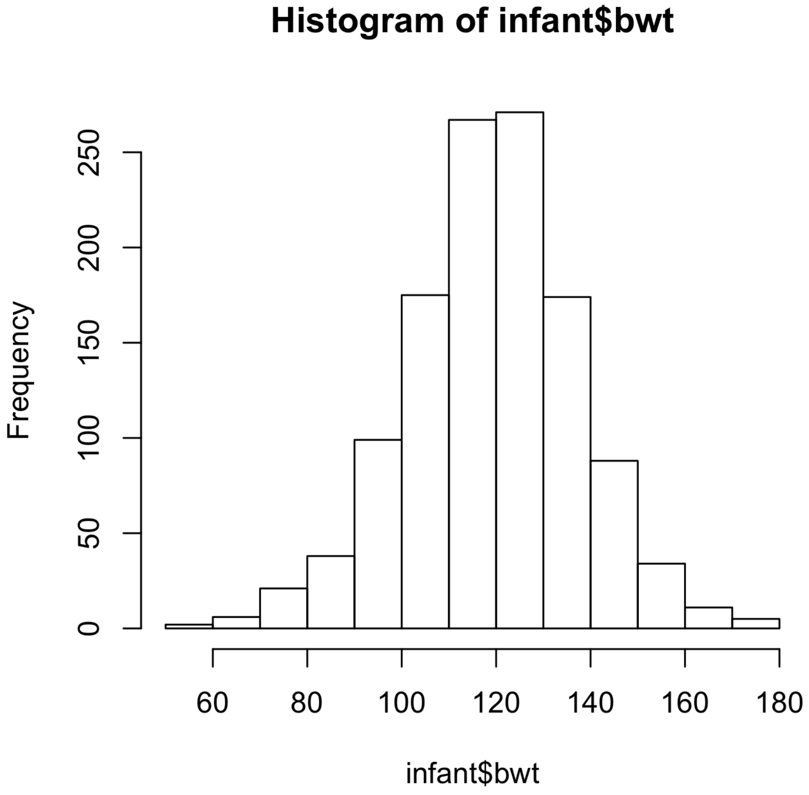
```
boxplot(infant$bwt)
with(infant, boxplot(bwt ~ `Ethnic group`))
```

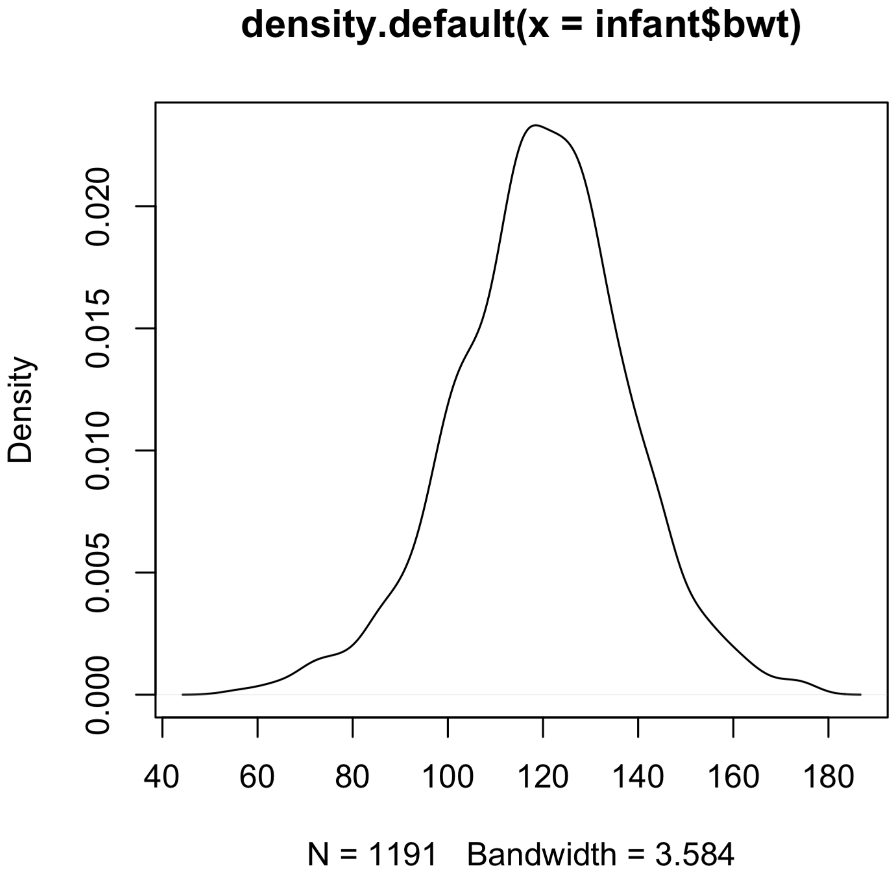




Histogram/Density

```
hist(infant$bwt)
plot(density(infant$bwt))
```



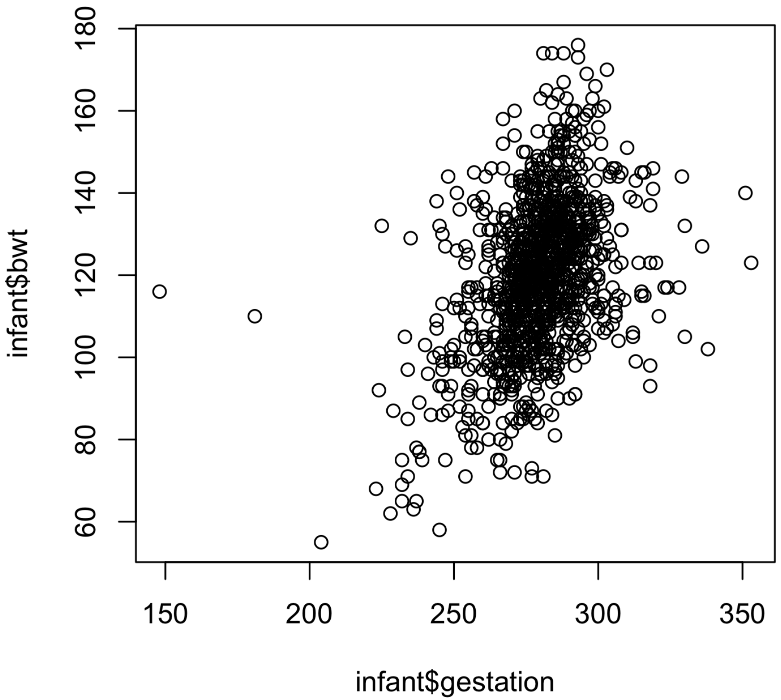


Scatterplots

```
plot(bwt ~ gestation, data = infant)
```

or

```
plot(infant$gestation, infant$bwt)
```



ggplot2

ggplot2 is a package that produces plots based on the *grammar of graphics* (Leland Wilkinson), the idea that you can build every graph from the same components

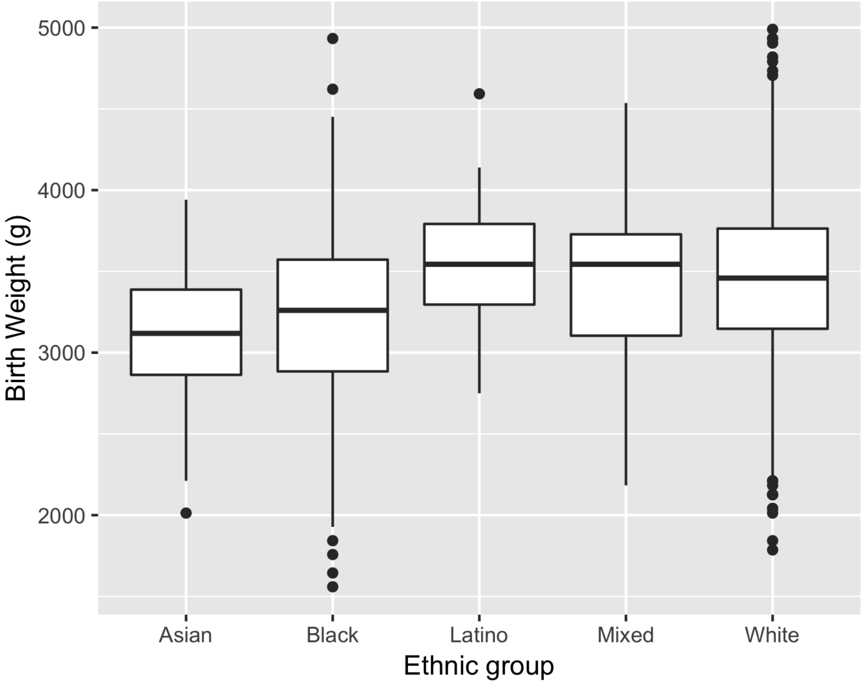
- a data set
- a co-ordinate system
- and *geoms* visual marks that represent data points

To display values, you map variables in the data to visual properties of the geom (*aesthetics*), like size, colour, x-axis and y-axis.

It provides a unified system for graphics, simplifies tasks such as adding legends, and is fully customisable.

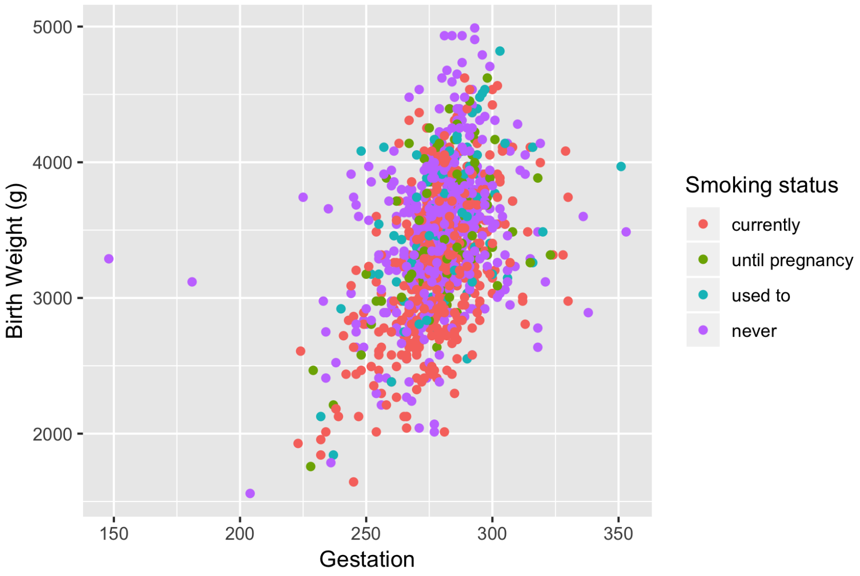
ggplot2 boxplots

```
library(ggplot2)
ggplot(infant, aes(y = bwt * 28.35, x = `Ethnic group`)) +
    geom_boxplot() +
    ylab("Birth Weight (g)")
```



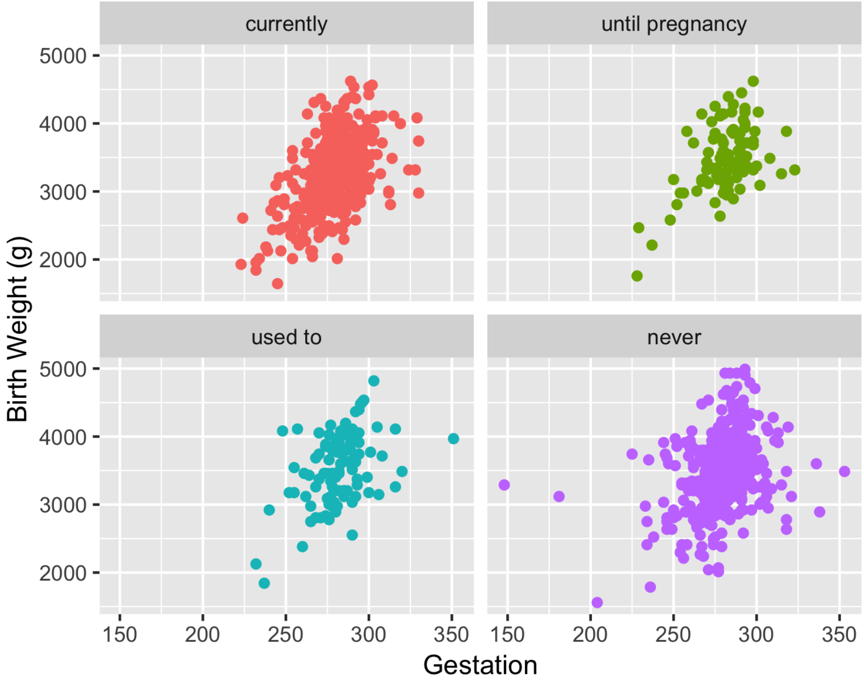
ggplot2 with Colour

```
infant <- mutate(infant, smoke = recode_factor(smoke,
                                       `1` = "currently", `2`= "until pregnancy",
                                       `3` = "used to", `0` = "never"))
p <- ggplot(infant, aes(y = bwt * 28.35, x = gestation, color = smoke)) +
    geom_point() + labs(x = "Gestation", y = "Birth Weight (g)", color = "Smoking status")
p
```



ggplot2 Facetting

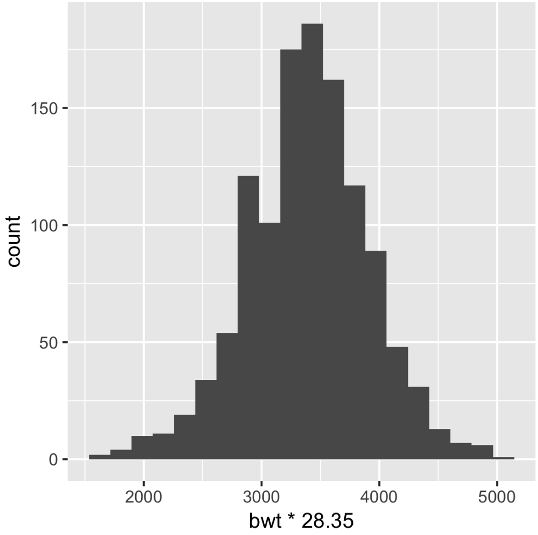
```
p + facet_wrap(~ smoke) + guides(color = FALSE)
```

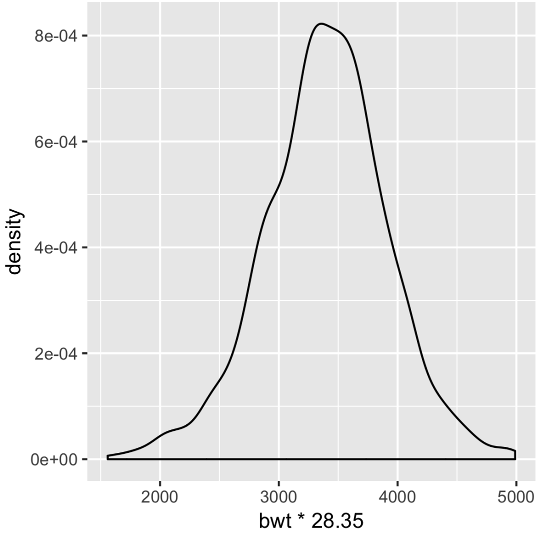


Plots in R Markdown

The chunk options enable you to arrange plots (caption used as alt text here)

~~~
```{r, fig.show = "hold", fig.align = "center", fig.width = 4, fig.height = 4,
    fig.cap = "Left: ggplot histogram. Right: ggplot density curve", out.width = "30%"}
ggplot(infant, aes(x = bwt * 28.35)) + geom_histogram(bins = 20)
ggplot(infant, aes(x = bwt * 28.35)) + geom_density()
```
~~~





Your Turn!

1. Do some nice graph!

2. Include them into your R Markdown file, select appropriate size for the figure and caption.

3. Briefly comment your findings in your R Markdown file.

Learning more/getting support

RStudio cheatsheets (`rmarkdown`, `dplyr`, `ggplot2`) https://www.rstudio.com/resources/cheatsheets/)

R for Data Science (data handling, basic programming and modelling, R markdown) http://r4ds.had.co.nz

RStudio Community (friendly forum focused on "tidyverse" packages, including `dplyr`, `ggplot2`) https://community.rstudio.com/

Going further

Quick-R (basic "how to"s from data input to advanced statistics)
http://www.statmethods.net/

Task views http://cran.r-project.org/web/views/ provide an overview of R's support for different topics, e.g. Bayesian inference, survival analysis, ...

http://www.rseek.org/ – search the web, online manuals, task views, mailing lists and functions.

Package vignettes, many books, e.g. on https://bookdown.org/.

Acknowledgments and License

Huge thanks to @HeathrTurnr and the @R_Forwards teaching team for helping me with the material for this workshop! Errors and omissions are my own!

This work is licensed under the Creative Commons Attribution-Noncommercial 3.0 United States License.
To view a copy of this license, visit http://creativecommons.org/licenses/by-nc/3.0/us/