Data Week Online 2020

Making data work for everyone

bristol.ac.uk/golding

Data Week Online 2020

The Jean Golding Institute

- A central hub for data science and data-intensive research
- One of 5 University of Bristol research institutes
- Connect multidisciplinary experts across the University and beyond
- Events, training, funding, Ask JGI, The Alan Turing Institute

Our priorities

1. Societal challenges
2. Data visualisation
3. Reproducibility & data governance
4. Fundamental research

Making data work for everyone

bristol.ac.uk/golding

Data Week Online 2020

Date	Event	Speaker
Monday 15 June	Data science and COVID 19 & Data Week Introduction	Kate Robson Brown, JGI Director
Monday 15 June	Intermediate Python	Advanced Computing Research Centre
Tuesday 16 June	Talk: Working at and with The Turing Institute: experiences as a Fellow	Jon Crowcroft, Turing Fellow & University of Cambridge
Tuesday 16 June	Talk: increasing engagement with data	Michael Green, Luna 9
Tuesday 16 June	Introduction to data analysis in Python	Advanced Computing Research Centre
Wednesday 17 June	Do you want to be a data Rockstar?	Luke Stoughton, The Information Lab
Wednesday 17 June	Applied data analysis in Python	Advanced Computing Research Centre
Thursday 18 June	Talk: New data on COVID-19 is undermined by old statistical problems	Gibran Hemani, University of Bristol
Thursday 18 June	Managing sensitive research data: from planning to sharing	Library Research Services
Thursday 18 June	Introduction to deep learning	Advanced Computing Research Centre
Friday 19 June	Deep Learning for Health and Life Sciences	Valerio Maggio, University of Bristol
Friday 19 June	Tour of the Tidyverse	Max Kronborg, Mango Solutions
Friday 19 June	Best practices in software engineering	Advanced Computing Research Centre

Tour of the Tidyverse

Max Kronborg

In This Workshop, you'll learn....

- What the **tidyverse** is & why bother using it
- What tools are available in the **tidyverse**
- A brief overview of core functionality

What data are we using?

- Funding of Olympic sports
- UK Sport World Class Performance Programme
- Data from Sydney (2000) to Rio de Janeiro (2016)

For the exercises:

- uni_result.csv – received by email

What is the tidyverse?

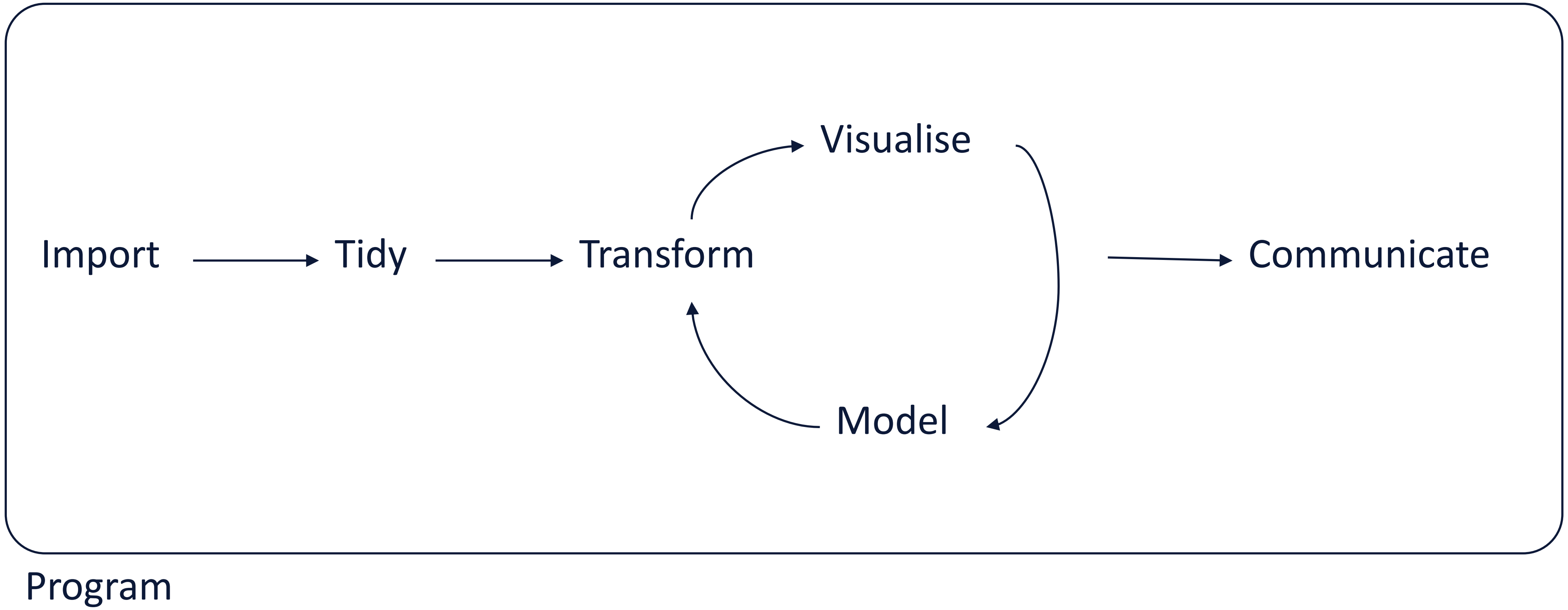
Unified packages!

- Packages that cover the data analysis workflow.
- All built on the same principles.

Shiny
R Studio
ggplot2
dplyr
rmarkdown
knitr
lubridate
stringr
devtools
tidyr
%>%
Ceci n'est pas un pipe.
packrat
ggvis
readr

Tidy philosophy

- Reuse existing data structures
- Combine functions with the **pipe**
- Embrace functional programming
- Design for humans


Import
Tidy
Transform
Visualise
Model
Communicate
Program

What’s in the tidyverse?

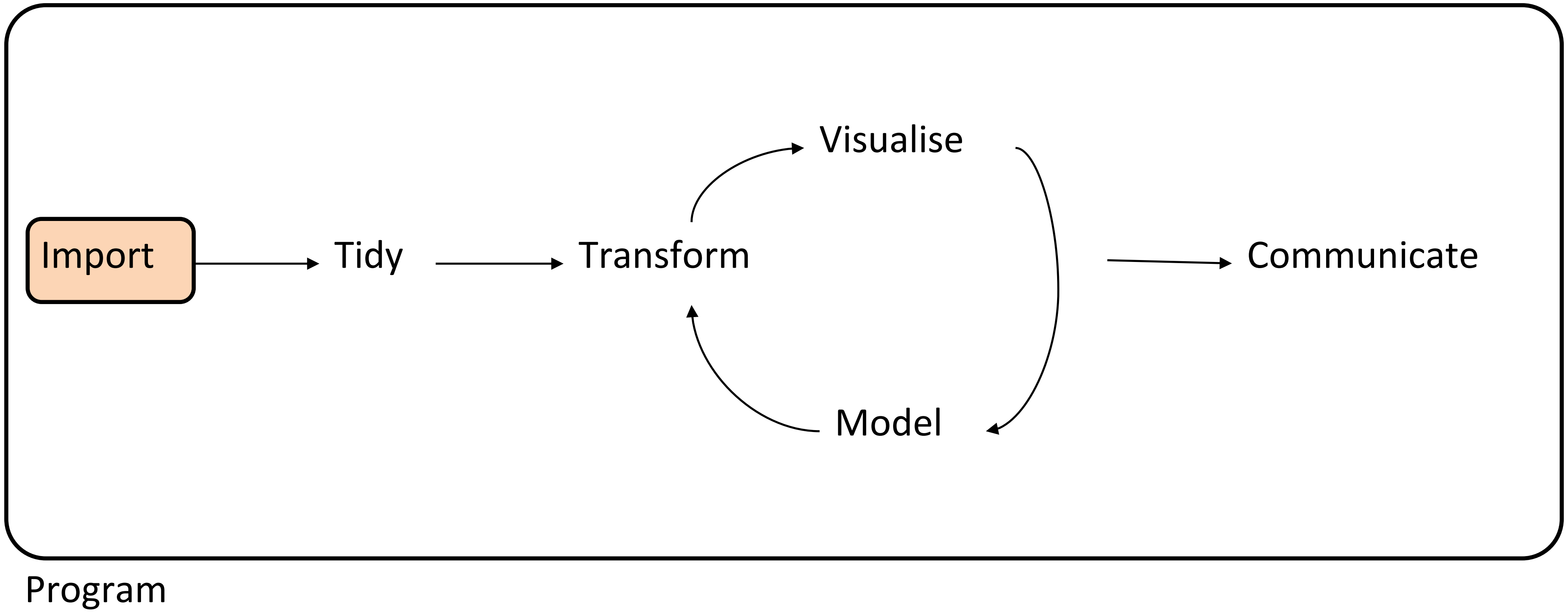
Core packages

`library(tidyverse)` loads in the 6 core tidyverse packages:

ggplot2
readr
purrr
dplyr
tidyr
TIBBLE

There's loads more of the 'tidyverse-adjacent' packages that are installed at the same time as the tidyverse.

A Dive Into The Packages


Import
Tidy
Transform
Visualise
Model
Communicate
Program

Import

- `readr`
- `haven`
- `httr`
- `jsonlite`
- `readxl`
- `rvest`
- `xml2`

Reading Web Data: rvest

```
Read_html(www.myWebPage.co.uk)
```

- Includes lots of functions for extracting specific elements from a web page.

Read Olympic Data

```
funding <- read_html("https://www.uksport.gov.uk/our
               -work/investing-in-sport/
               historical-funding-figures")


funding <- html_table(funding, header = TRUE)
```

Importing Tabular Data

```
read_csv(“path/to/file.csv”)
```

Other variants:

- `read_tsv`
- `read_delim`

Importing Data from MS Excel:

Use the **readxl** package

```
read_excel(“path/to/file.xlsx”,
           sheet = “sheet 1”)
```

Other cases

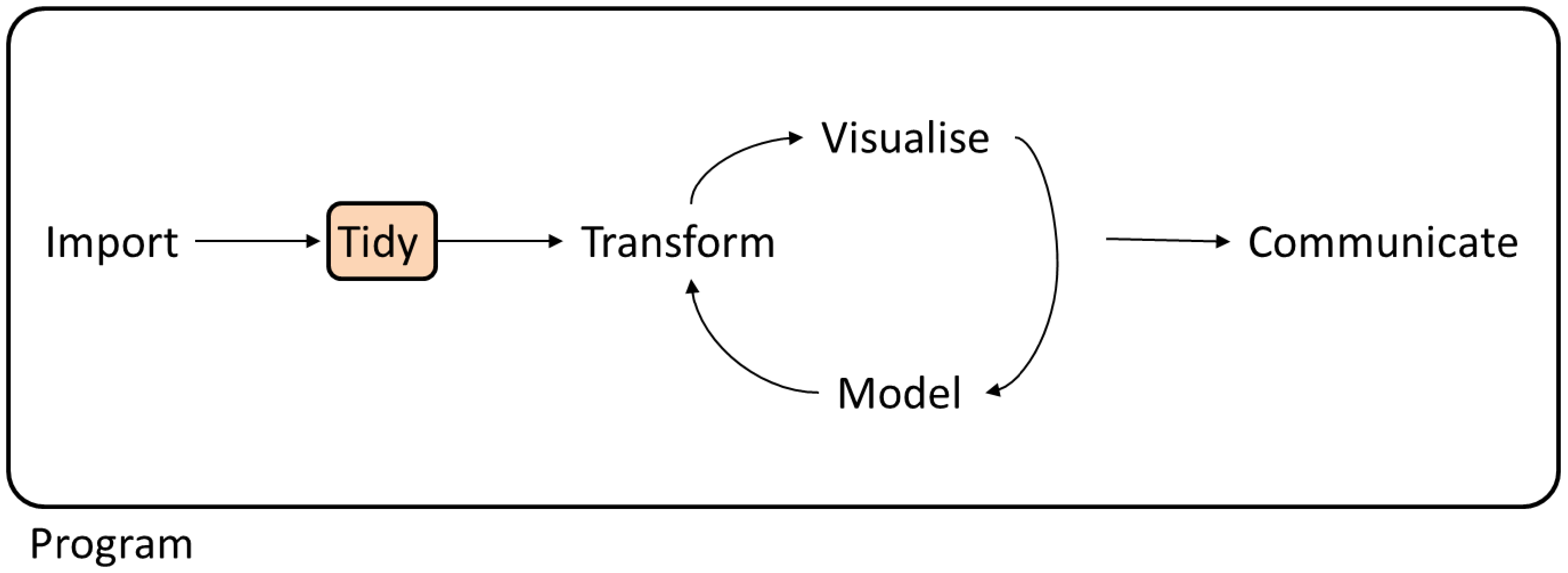
Use haven for **spss/sas** data

Use xml2 for **xml** files

jsonlite for json files

Exercise!

1. Read in the uni_results.csv using the appropriate package and function.


Import
Tidy
Transform
Visualise
Model
Communicate
Program

Tidy

- `tibble`
- `tidyr`

tibble?

- A tibble is an updated data frame
  - Can be created using the **tibble** package
- Extra features:
  - Only prints top 10 rows
  - Doesn't create row names – removes them if they're already there
  - Character columns not converted to factor
- Often created without you realising!

tibble functions

- `glimpse` – useful overview of data
- `tibble/tribble` – creation of tibbles
- `add_row/add_column` – helpful functions for adding elements to an existing tibble
- There are also a number of functions for converting rownames if needed

Creating a tibble Row-wise

```
years <- tribble(
        ~Location, ~Year, ~Month, ~Day,
        "Sydney", 2000, 9, 15,
        "Athens", 2004, 8, 13,
        "Beijing", 2008, 8, 8,
        "London", 2012, 7, 27,
        "Rio de Janeiro", 2016, 8, 5
       )
```

Tidy data

- The **tidyverse** is designed to work with tidy data
- A single structure that is common to all of the packages
    - Makes it easy to move from manipulation to visualisation to modelling without changing the data

Tidy Data Principles

- Each variable has its own column
- Each observation has its own row
- Each value has its own cell

Using tidyr

- tidyr is used to get our data to follow the tidy data principles.
- `pivot_longer` – used to gather multiple columns into 1
- `pivot_wider` – used to spread a single column across multiple
- `separate` – separates a column into 2 columns

Tidying the Olympic Data

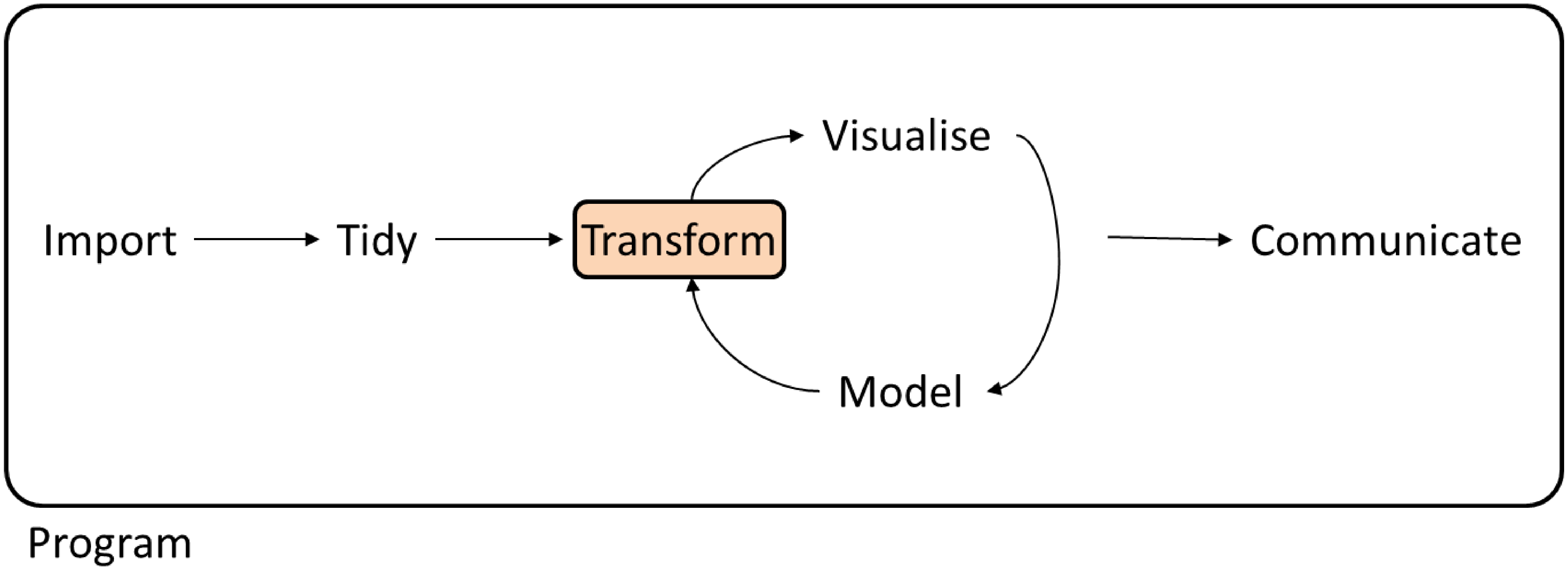
```
summer <- pivot_longer(
                 data = summer,
                 cols = -Sports,
                 names_to =
                    "Location",
                 values_to =
                    "Funding")
```

Working with Missing Values

```
summer <- replace_na(summer,
       replace = list(Funding=0))
```

Exercise!

1. Using the uni_results data that you read in during the last exercise, transform the data so that there is only one column of test scores. (Use the `pivot_longer` function)


Import
Tidy
Transform
Visualise
Model
Communicate
Program

Transform

- `dplyr`
- `forcats`
- `stringr`
- `hms`
- `lubridate`

Data Manipulation with *dplyr*

- 5 main dplyr verbs:
    - `filter`
    - `select`
    - `arrange`
    - `mutate`
    - `summarise`
- `group_by` allows actions to be performed by group

Filtering a dataset

```
noFunding <- filter(summer, Funding == 0)
```

First argument is the dataset

Followed by conditions to filter by

Joining related Data sets

dplyr provides functions for joining two data sets:

- `full_join`
- `left/right_join`
- `inner_join`
- `anti_join`
- `semi_join`

Joining our **`summer`** and **`years`** data

```
summer <- full_join(summer, years)
```

Exercise!

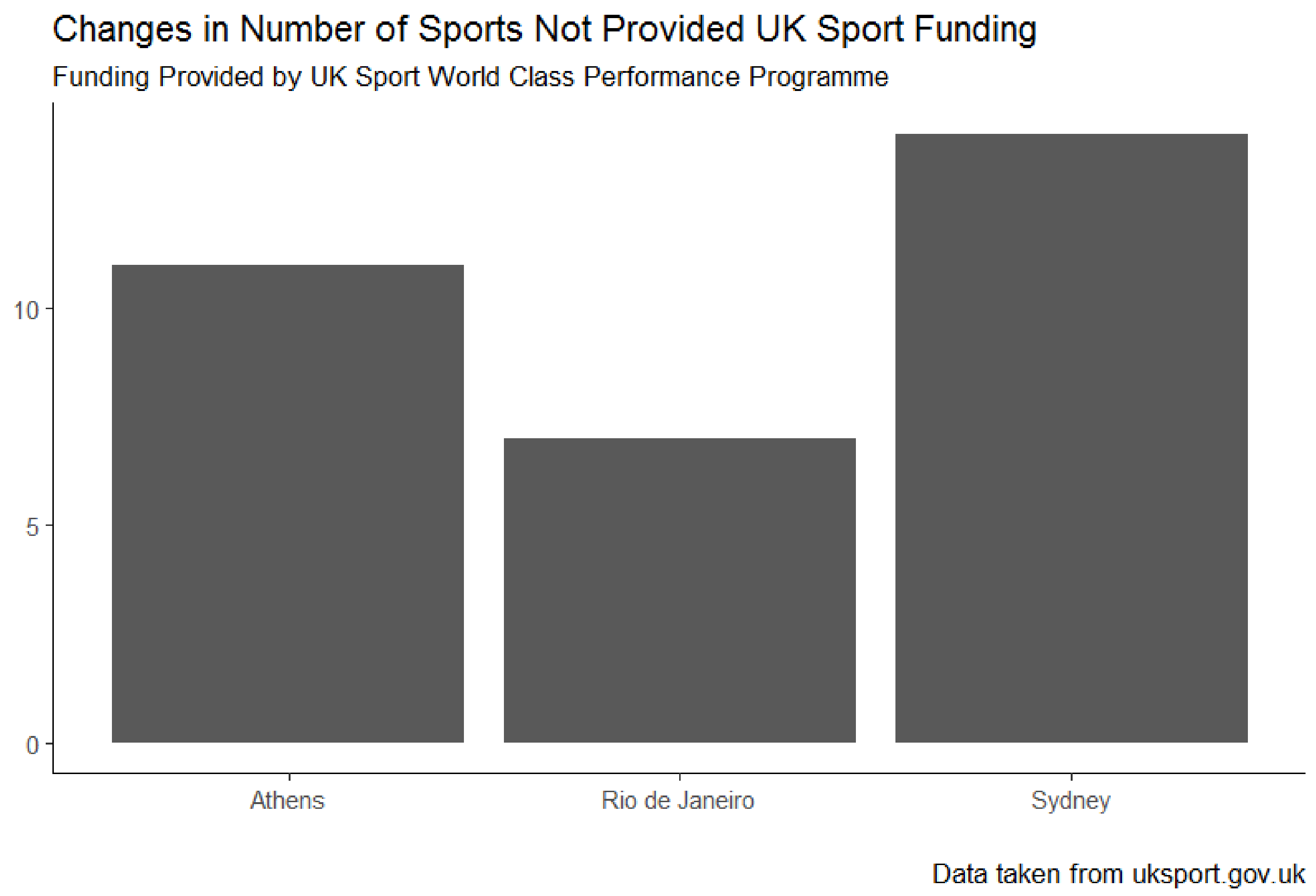
Using the tidied uni_results data:

1. Obtain all the records corresponding to females
2. Sort the initial dataset by descending `test_score`
3. Find the mean `test_score` for test a and b for each `course`

Manipulating Factors: *forcats*

- Factors are a representation of categorical data.
- Forcats allows us to easily manipulate factors, you can:
  - Change names
  - Group levels
  - Reorder levels

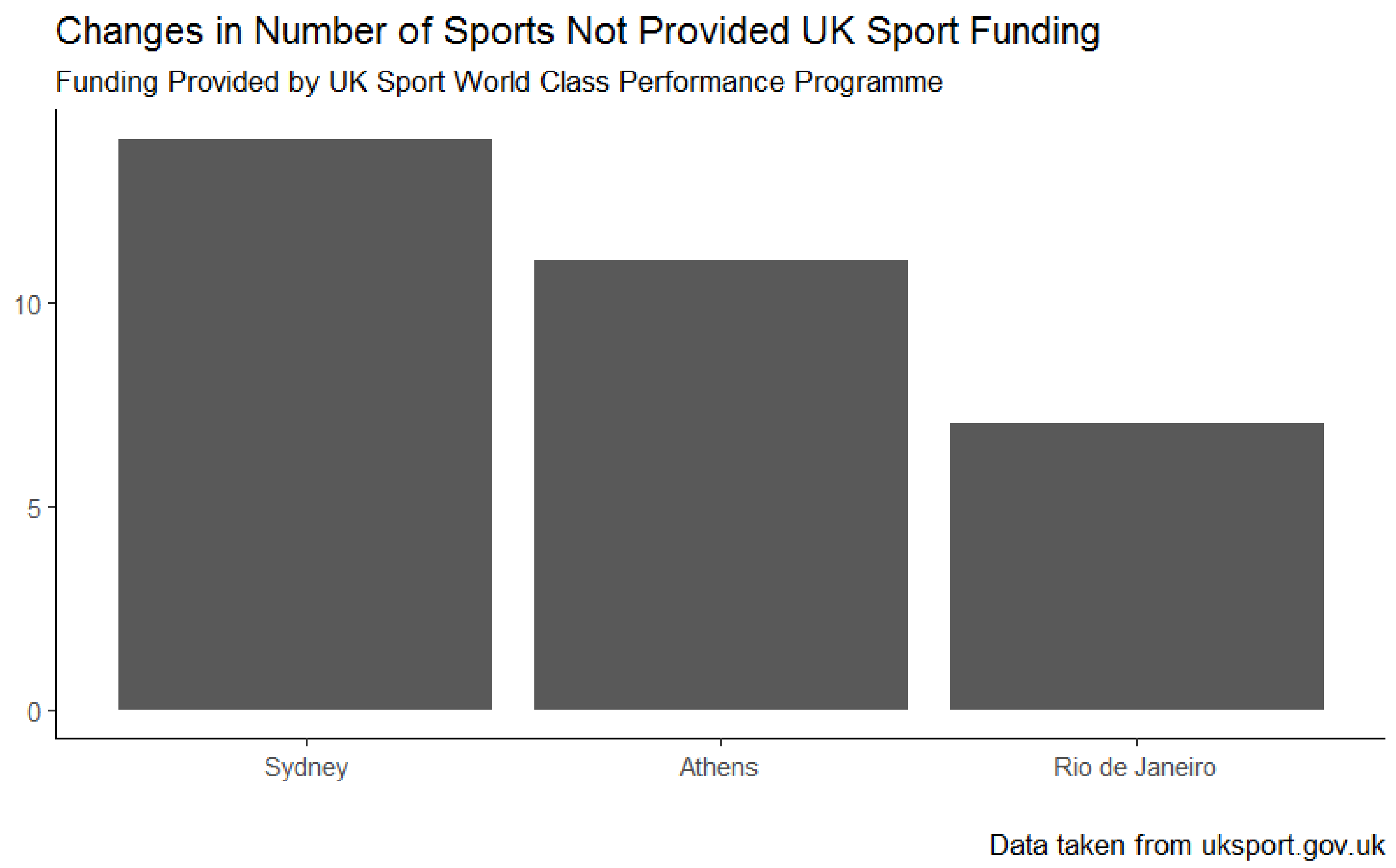
Without Manipulation



Ordering the **`Location`** variable by **`Year`**

```
numberNoFund <- noFunding %>%
  count(Location) %>%
  left_join(years) %>%
  mutate(Location = fct_reorder(Location,
Year))
```

With Manipulation



Manipulating Characters: *stringr*

- Stringr provides consistent functions for manipulating character strings
- Manipulations include:
  - Concatenation
  - Pattern search and replace
  - Subset with strings

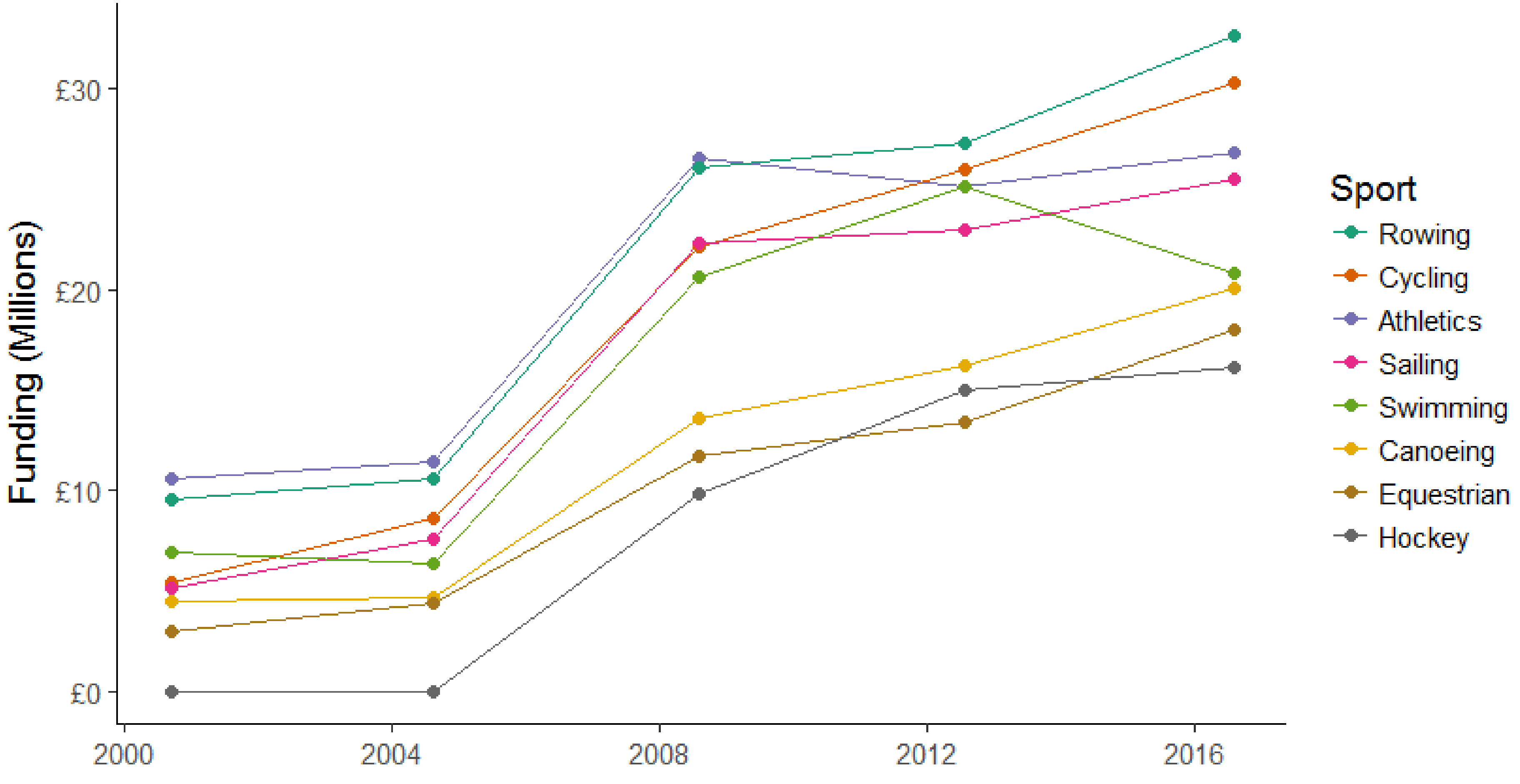
Manipulating Dates: *lubridate*

- Easy to use functions to convert characters to date format:
    - ymd
    - mdy
    - dmy
- Extract elements from dates
- Arithmetic of time and dates

Creating Olympic Start Dates

```
summer <- mutate(summer,
     Date = str_c(Year, Month, Day, sep = "-"),
      Date = ymd(Date)
     )
```

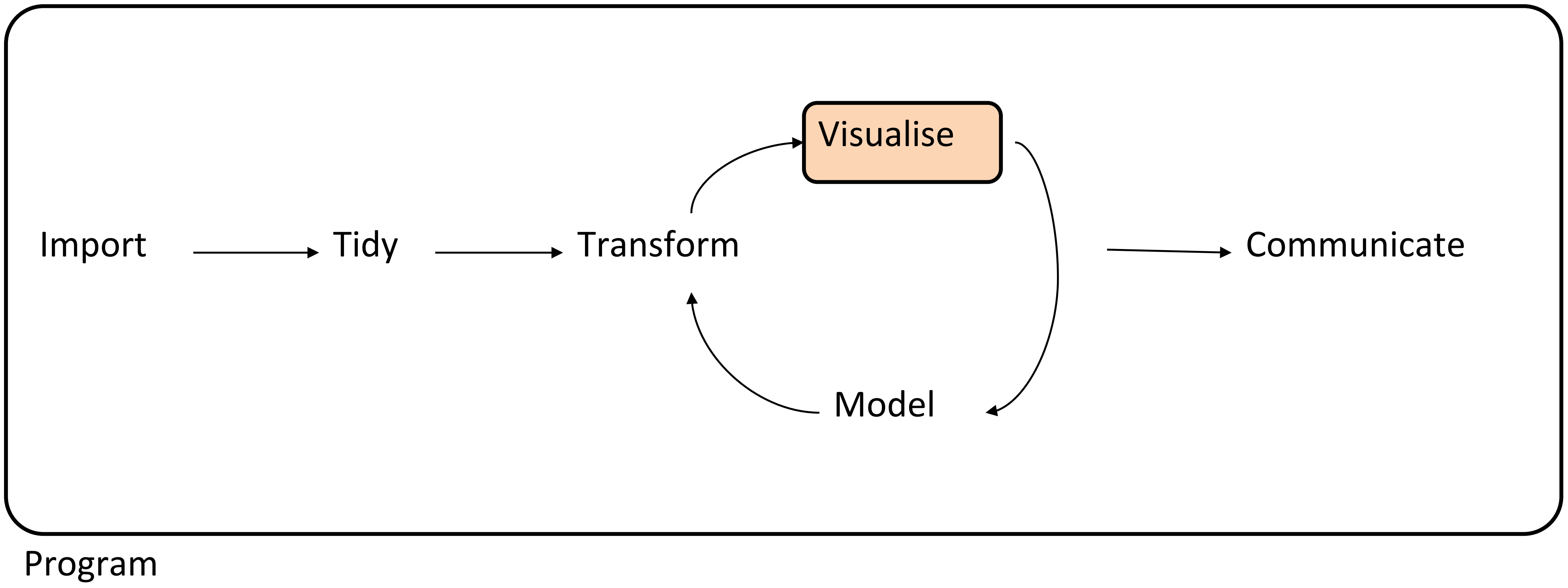
Overall Increasing Funding of Top Sports



Data taken from uksport.gov.uk

Exercise!

1. Convert the `test` variable to a factor, rename the values to “A” and “B”.
2. Make sure that the `graduation_date` variable is indeed a date object. (You can check this using `class(uni_results$graduation_date)`)


Visualise
Import
Tidy
Transform
Communicate
Model
Program

Visualise

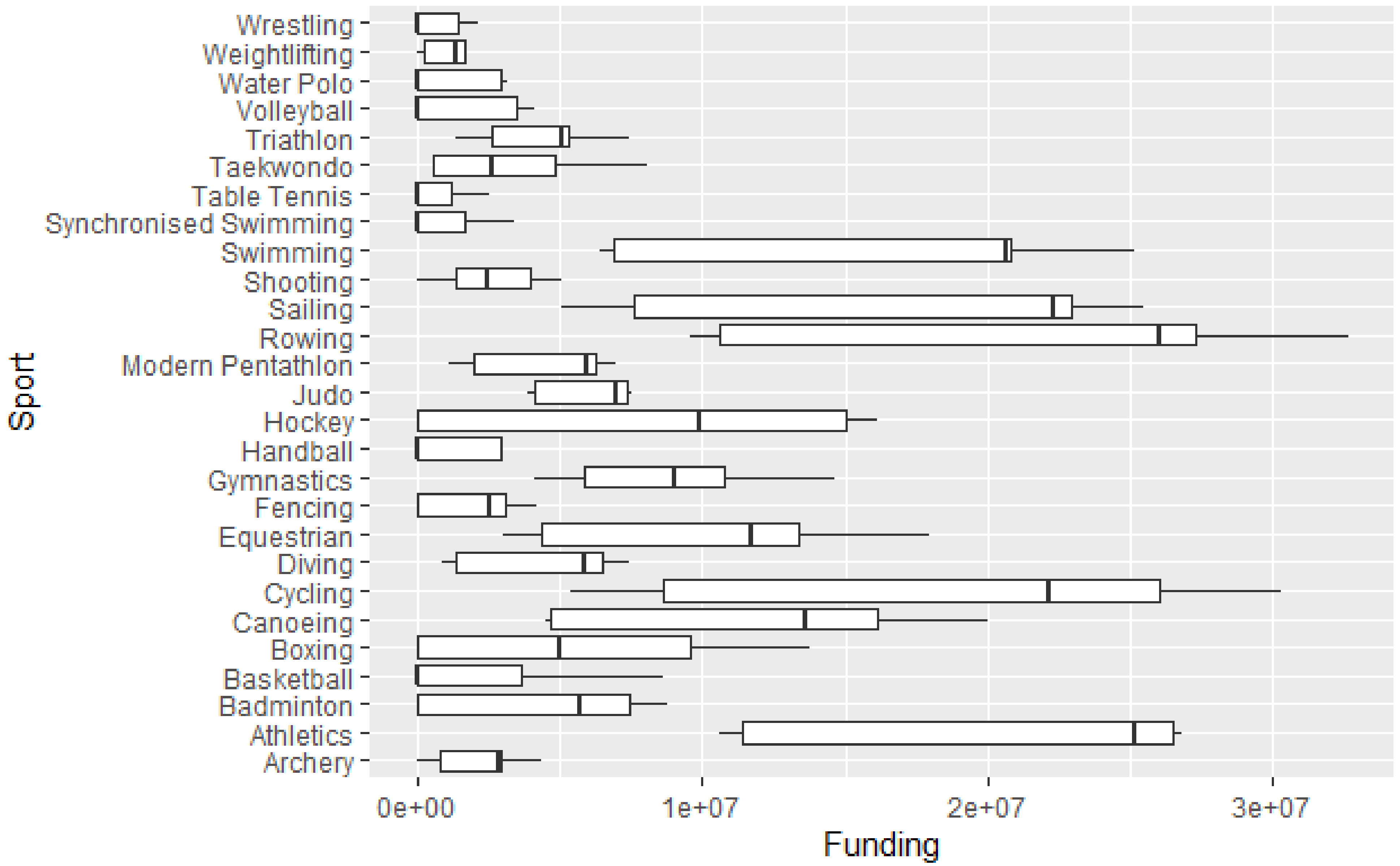
- `ggplot2`

Creating graphics: ggplot2

- Create quick plots with qplot
- Specify the type of plot using the geoms
- Add titles and labels with labs

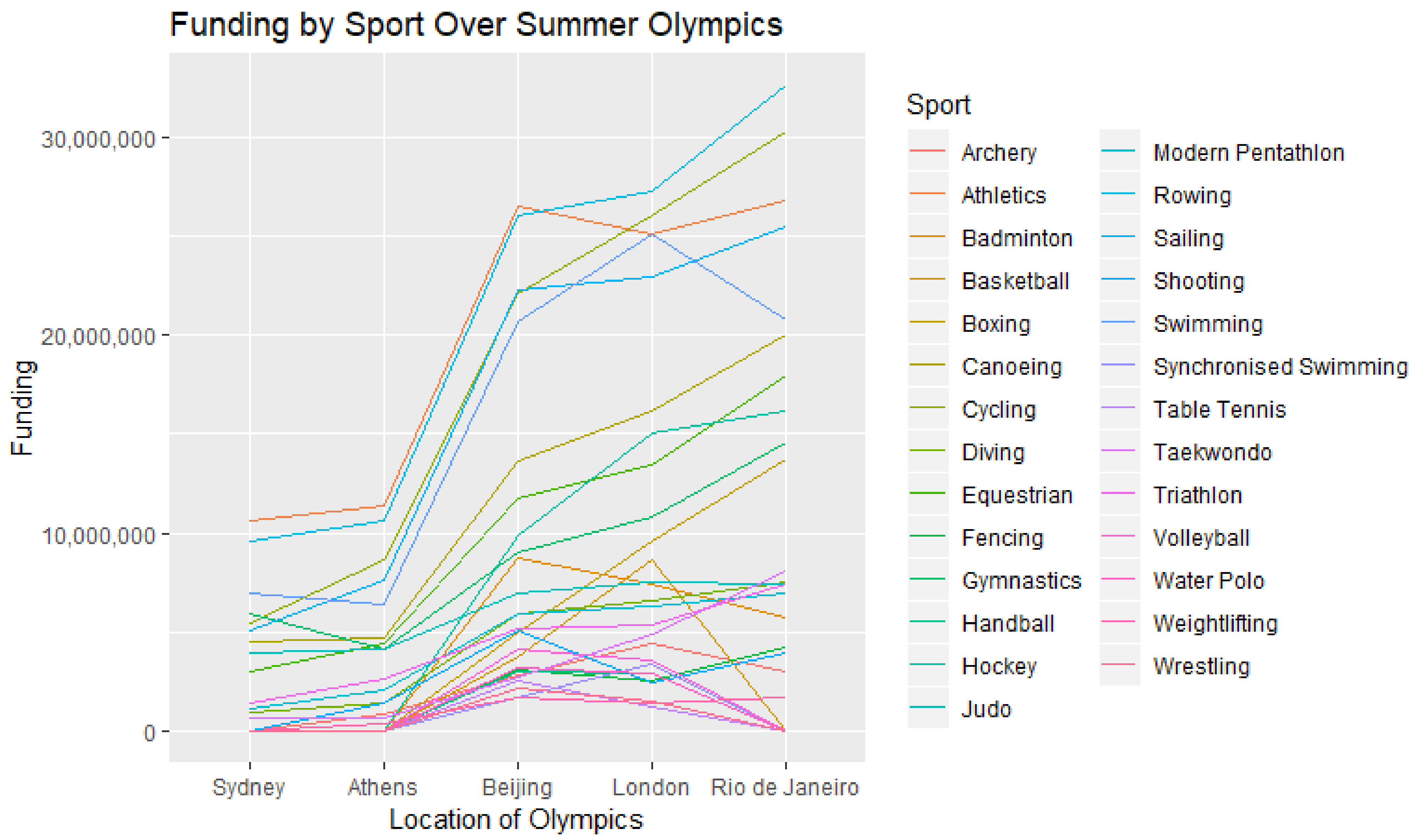
Using **`qplot`**

```
qplot(data = summer, x = Funding, y = Sport,
      geom = 'boxplot')
```


Wrestling
Weightlifting
Water Polo
Volleyball
Triathlon
Taekwondo
Table Tennis
Synchronised Swimming
Swimming
Shooting
Sailing
Rowing
Modern Pentathlon
Judo
Hockey
Handball
Gymnastics
Fencing
Equestrian
Diving
Cycling
Canoeing
Boxing
Basketball
Badminton
Athletics
Archery
Sport
0e+00
1e+07
2e+07
3e+07
Funding

Using **ggplot**

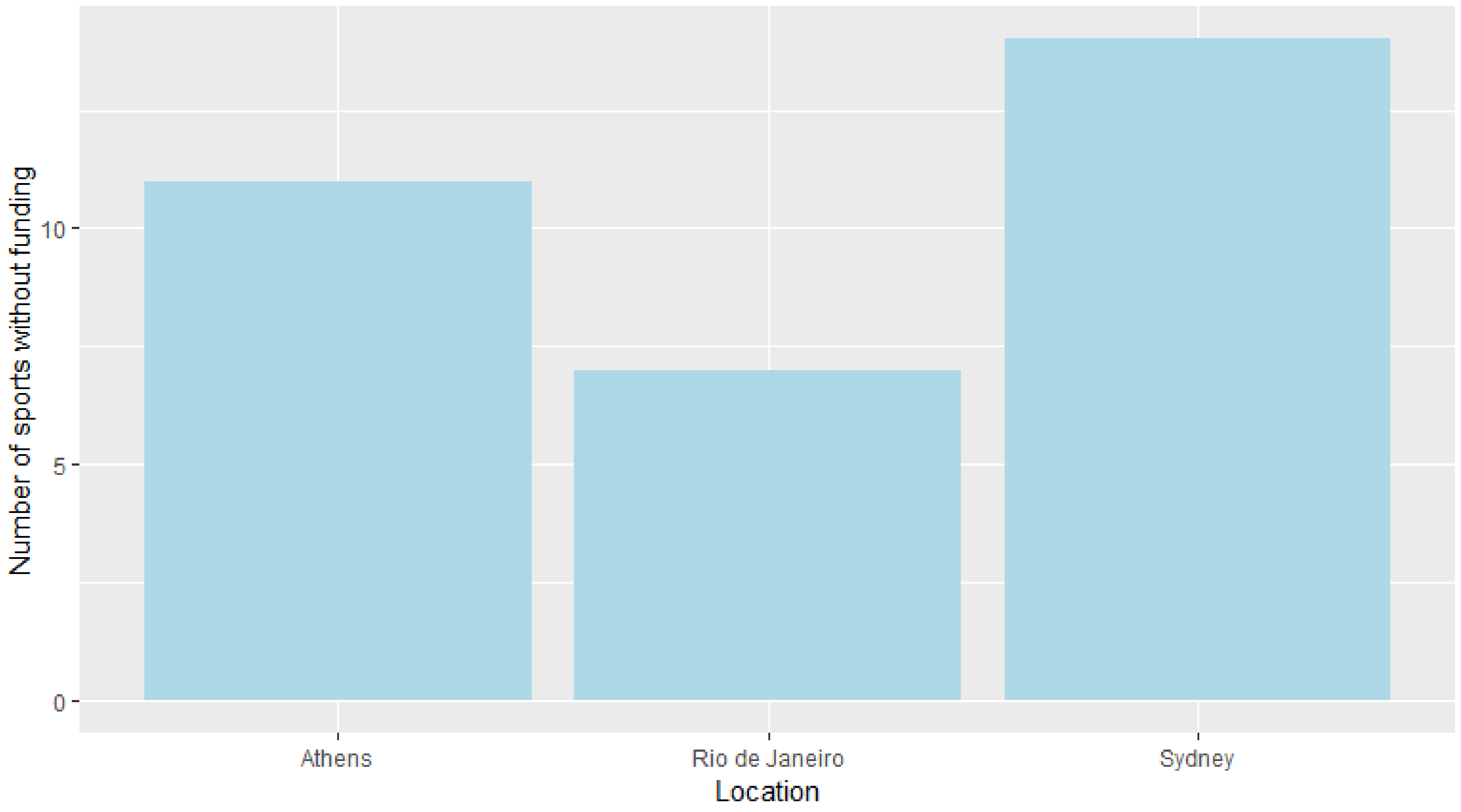
```
ggplot(data = summer,
       mapping = aes(x = fct_reorder(Location,Year),y = Funding,
                     group = Sport, colour = Sport)) +
  geom_line() +
  ggtitle("Funding by Sport Over Summer Olympics") +
  xlab("Location of Olympics") +
  scale_y_continuous(labels = scales::comma)
```

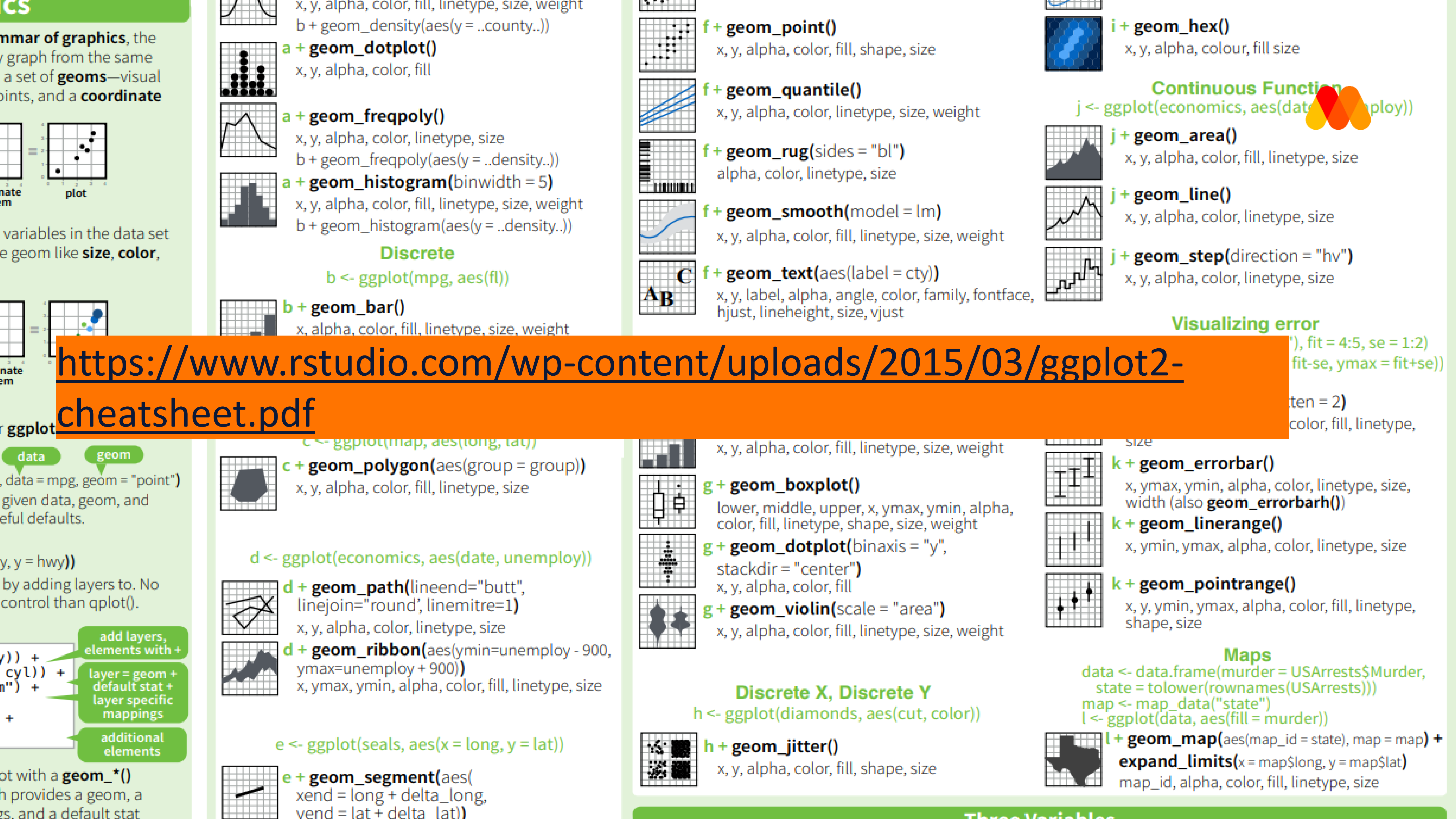

Funding by Sport Over Summer Olympics
Funding
30,000,000
20,000,000
10,000,000
0
Sydney
Athens
Beijing
London
Rio de Janeiro
Location of Olympics
Sport
Archery
Athletics
Badminton
Basketball
Boxing
Canoeing
Cycling
Diving
Equestrian
Fencing
Gymnastics
Handball
Hockey
Judo
Modern Pentathlon
Rowing
Sailing
Shooting
Swimming
Synchronised Swimming
Table Tennis
Taekwondo
Triathlon
Volleyball
Water Polo
Weightlifting
Wrestling

geom_bar

```
ggplot(data = noFunding,
       mapping = aes(x = Location)) +
  geom_bar(fill = "lightblue") +
  ggtitle("Counts of Sports with No Funding by Olympics") +
  ylab("Number of sports without funding")
```

Counts of Sports with No Funding by Olympics





cs

mmar of graphics, the
graph from the same
a set of geoms—visual
oints, and a coordinate

nate
em

plot

variables in the data set
e geom like size, color,

nate
em

r ggplot

data

geom

, data = mpg, geom = "point")
given data, geom, and
eful defaults.

y, y = hwy))

by adding layers to. No
control than qplot().

```
y)) +
 cyl)) +
m") +

 +
```

add layers, elements with +

layer = geom + default stat + layer specific mappings

additional elements

ot with a **geom_*()**
h provides a geom, a
s, and a default stat

x, y, alpha, color, fill, linetype, size, weight
b + geom_density(aes(y = ..county..))

a + **geom_dotplot()**
x, y, alpha, color, fill

a + **geom_freqpoly()**
x, y, alpha, color, linetype, size
b + geom_freqpoly(aes(y = ..density..))

a + **geom_histogram**(binwidth = 5)
x, y, alpha, color, fill, linetype, size, weight
b + geom_histogram(aes(y = ..density..))

Discrete

b <- ggplot(mpg, aes(fl))

b + **geom_bar()**
x, alpha, color, fill, linetype, size, weight

c <- ggplot(map, aes(long, lat))

c + **geom_polygon**(aes(group = group))
x, y, alpha, color, fill, linetype, size

d <- ggplot(economics, aes(date, unemploy))

d + **geom_path**(lineend="butt",
linejoin="round', linemitre=1)
x, y, alpha, color, linetype, size

d + **geom_ribbon**(aes(ymin=unemploy - 900,
ymax=unemploy + 900))
x, ymax, ymin, alpha, color, fill, linetype, size

e <- ggplot(seals, aes(x = long, y = lat))

e + **geom_segment**(aes(
xend = long + delta_long,
yend = lat + delta_lat))

f + **geom_point()**
x, y, alpha, color, fill, shape, size

f + **geom_quantile()**
x, y, alpha, color, linetype, size, weight

f + **geom_rug**(sides = "bl")
alpha, color, linetype, size

f + **geom_smooth**(model = lm)
x, y, alpha, color, fill, linetype, size, weight

f + **geom_text**(aes(label = cty))
x, y, label, alpha, angle, color, family, fontface,
hjust, lineheight, size, vjust

x, y, alpha, color, fill, linetype, size, weight

g + **geom_boxplot()**
lower, middle, upper, x, ymax, ymin, alpha,
color, fill, linetype, shape, size, weight

g + **geom_dotplot**(binaxis = "y",
stackdir = "center")
x, y, alpha, color, fill

g + **geom_violin**(scale = "area")
x, y, alpha, color, fill, linetype, size, weight

Discrete X, Discrete Y

h <- ggplot(diamonds, aes(cut, color))

h + **geom_jitter()**
x, y, alpha, color, fill, shape, size

i + **geom_hex()**
x, y, alpha, colour, fill size

Continuous Functi

j <- ggplot(economics, aes(dat ploy))

j + **geom_area()**
x, y, alpha, color, fill, linetype, size

j + **geom_line()**
x, y, alpha, color, linetype, size

j + **geom_step**(direction = "hv")
x, y, alpha, color, linetype, size

Visualizing error

"), fit = 4:5, se = 1:2)
fit-se, ymax = fit+se))

ten = 2)
color, fill, linetype,
size

k + **geom_errorbar()**
x, ymax, ymin, alpha, color, linetype, size,
width (also **geom_errorbarh()**)

k + **geom_linerange()**
x, ymin, ymax, alpha, color, linetype, size

k + **geom_pointrange()**
x, y, ymin, ymax, alpha, color, fill, linetype,
shape, size

Maps

data <- data.frame(murder = USArrests$Murder,
state = tolower(rownames(USArrests)))
map <- map_data("state")
l <- ggplot(data, aes(fill = murder))

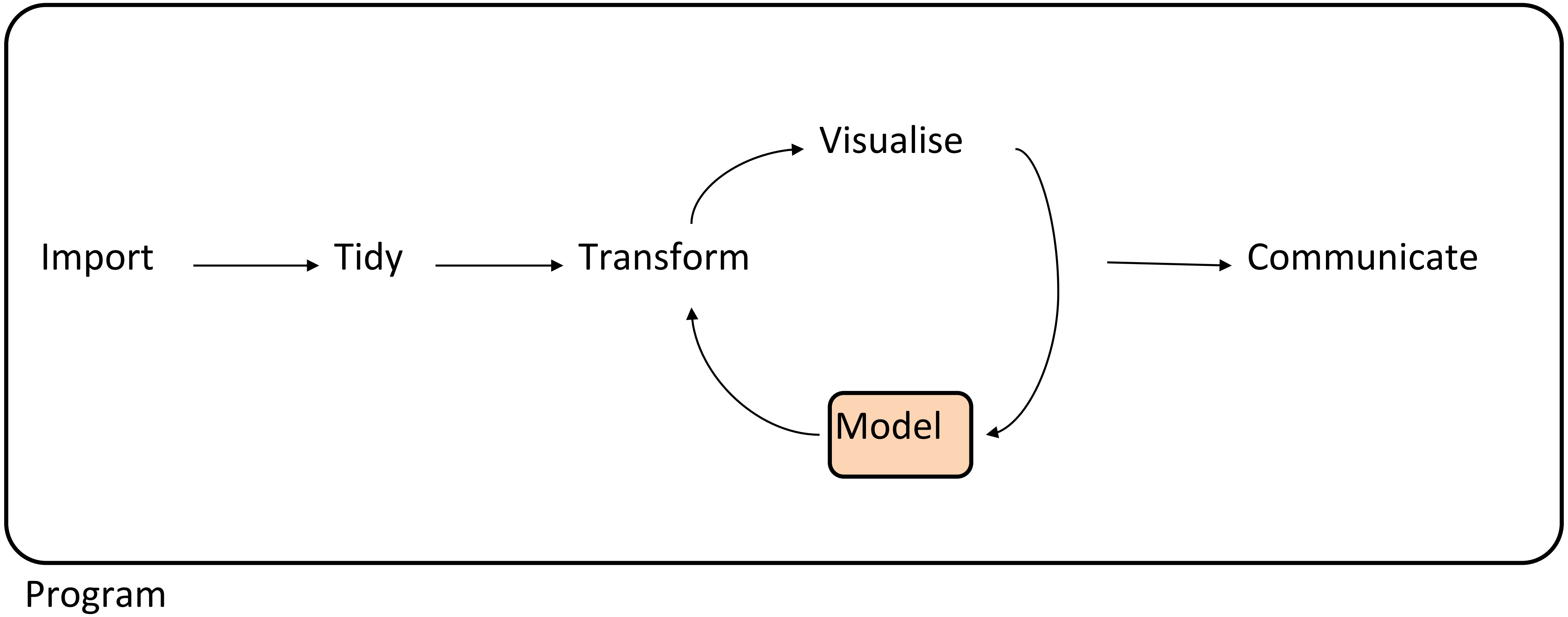
l + **geom_map**(aes(map_id = state), map = map) +
expand_limits(x = map$long, y = map$lat)
map_id, alpha, color, fill, linetype, size

Three Variables

https://www.rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf

Exercise!

1. Using the `ggplot` function, create a box plot of `test_score` by `test.`
2. Ensure the plot is well labelled
3. Colour the plot by `course`


Import
Tidy
Transform
Visualise
Model
Communicate
Program

Model

- `modelr`
- `broom`

Fitting Models: *modelr*

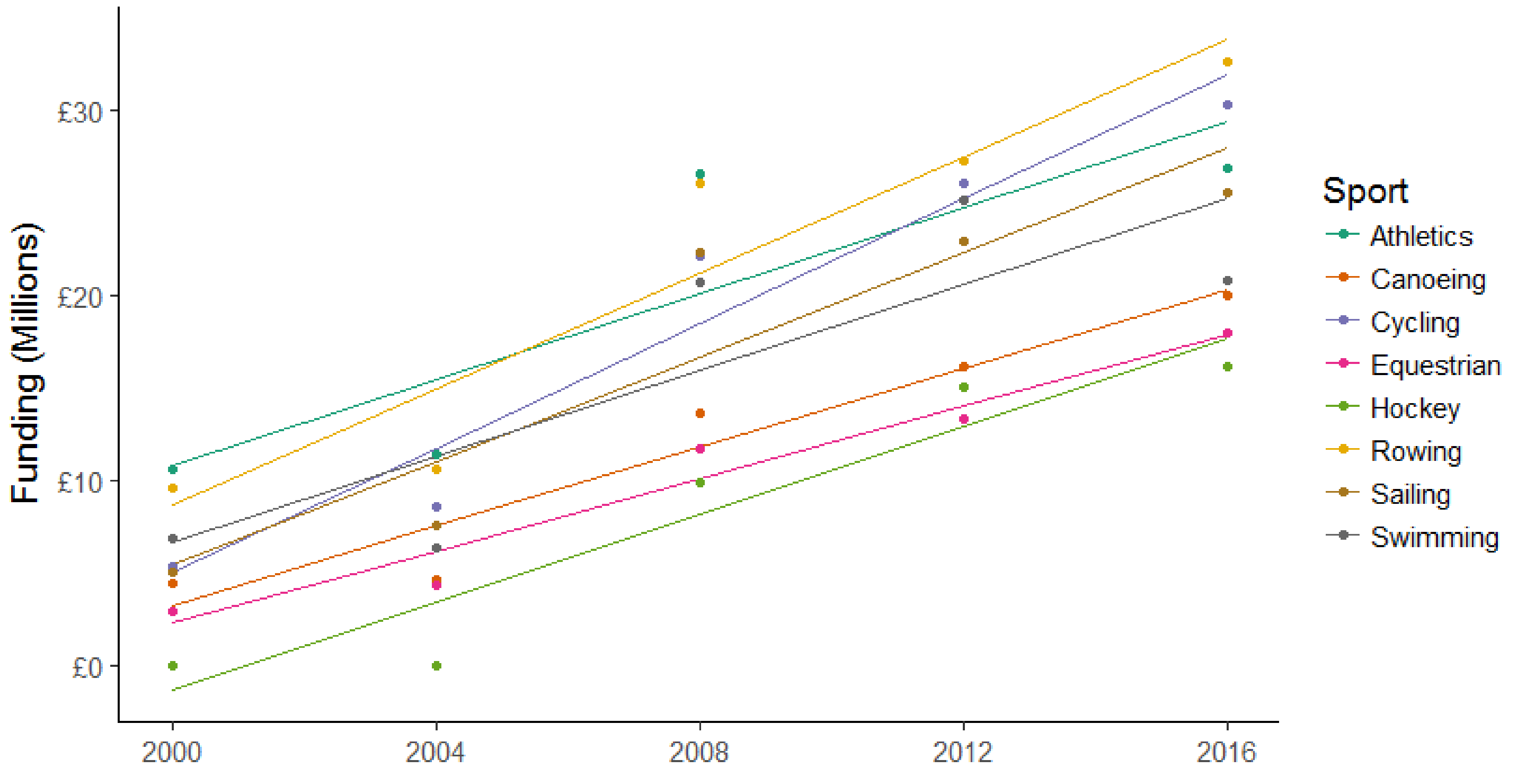
- Functionality for bootstrapping/ cross validation
- Model metrics – rsquare, rmse
- Extracting predictions and residuals

Predicting Olympic Funding

```
fundingModel <- lm(Funding ~ Sport*Year,
                    data = summer)


modelGrid <- data_grid(summer,
                                  Year, Sport)
modelGrid <- modelGrid %>%
                    add_predictions(fundingModel)
```

Fitted Model for Top Funded Sports

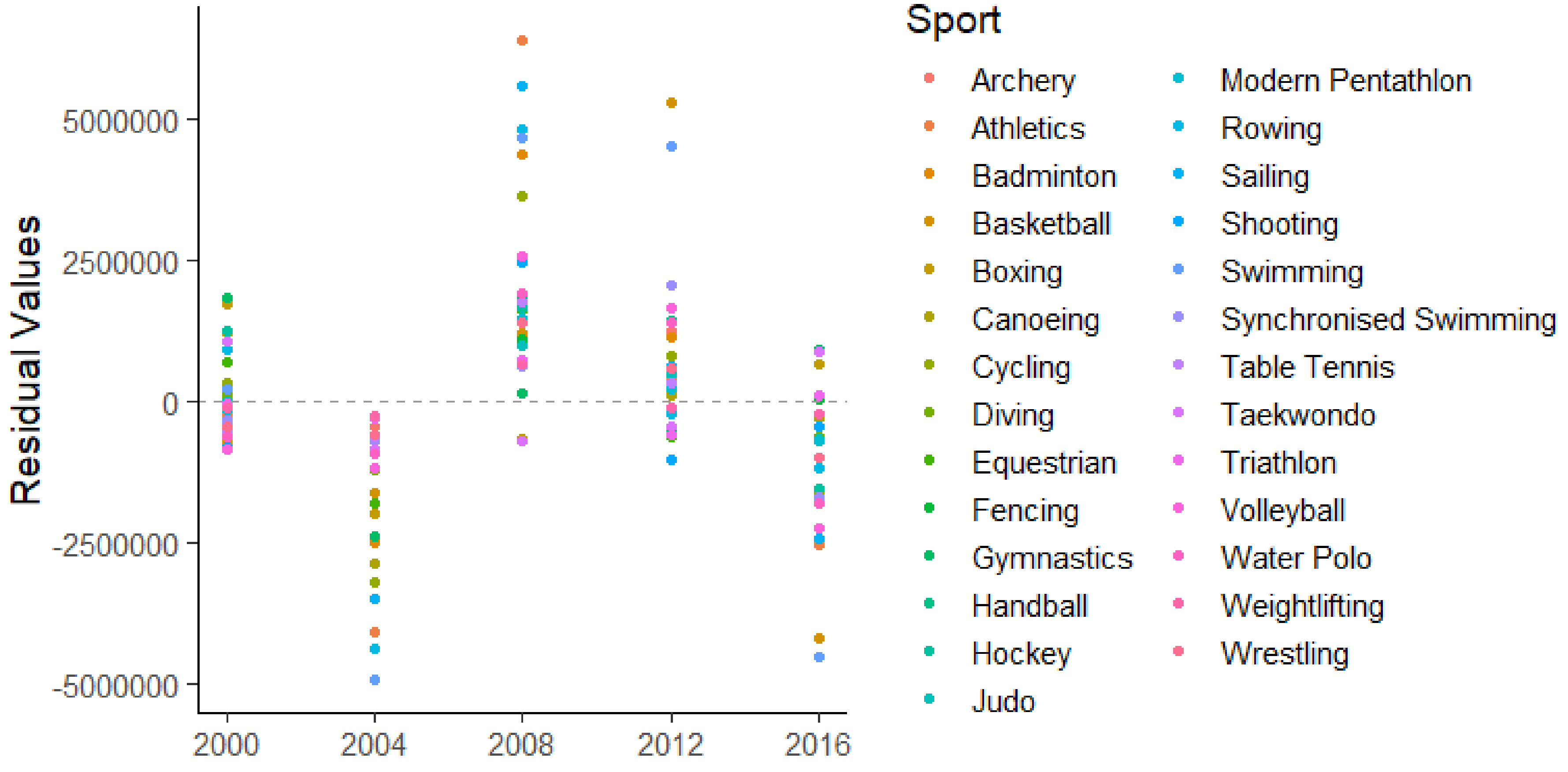


Data taken from uksport.gov.uk

Investigating Model Residuals

```
fundingResid <- summer %>%
                add_residuals(fundingModel)
```

Residual Values Suggests Further Fitting Required



Data taken from uksport.gov.uk

Assessing Model Quality: *broom*

- Provides functions for extracting details on the model fit
  - Model coefficients – `tidy`
  - Model diagnostics – `glance`
- Useful for working with multiple models to compare fit

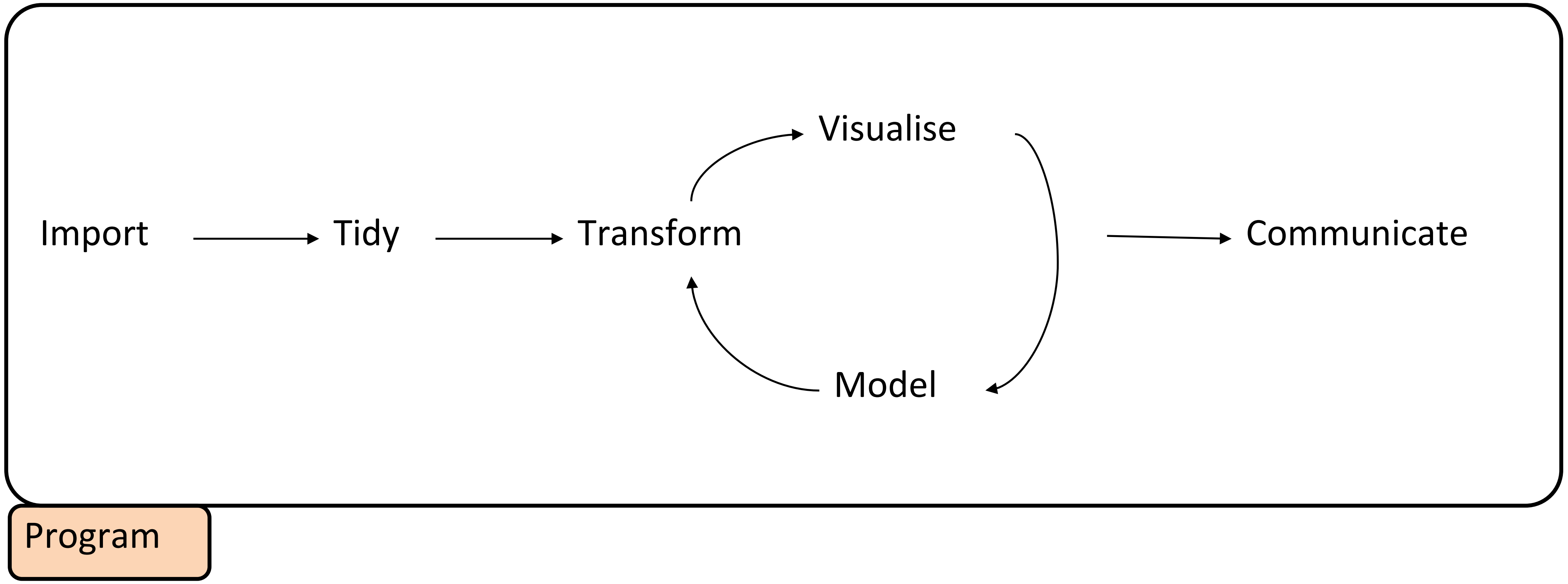
Fit of Olympics Models

```
tidy(fundingModel)

glance(fundingModel)
```

Exercise!

1. Create a linear model from your uni_results dataset that uses `course` and `test` to predict `test_score`.
2. Add the predictions from your model to the uni_results dataset.
3. Apply `tidy` and `glance` to your model


Import
Tidy
Transform
Visualise
Model
Communicate
Program

Program

- `purrr`

Iterating: *purrr*

- Iterate (over a vector of values) or apply to multiple set/subsets of data
- Output can be one of many types based on the function used:
  - `map`
  - `map_df`
  - `map_dbl`
  - …

Model For Each Sport

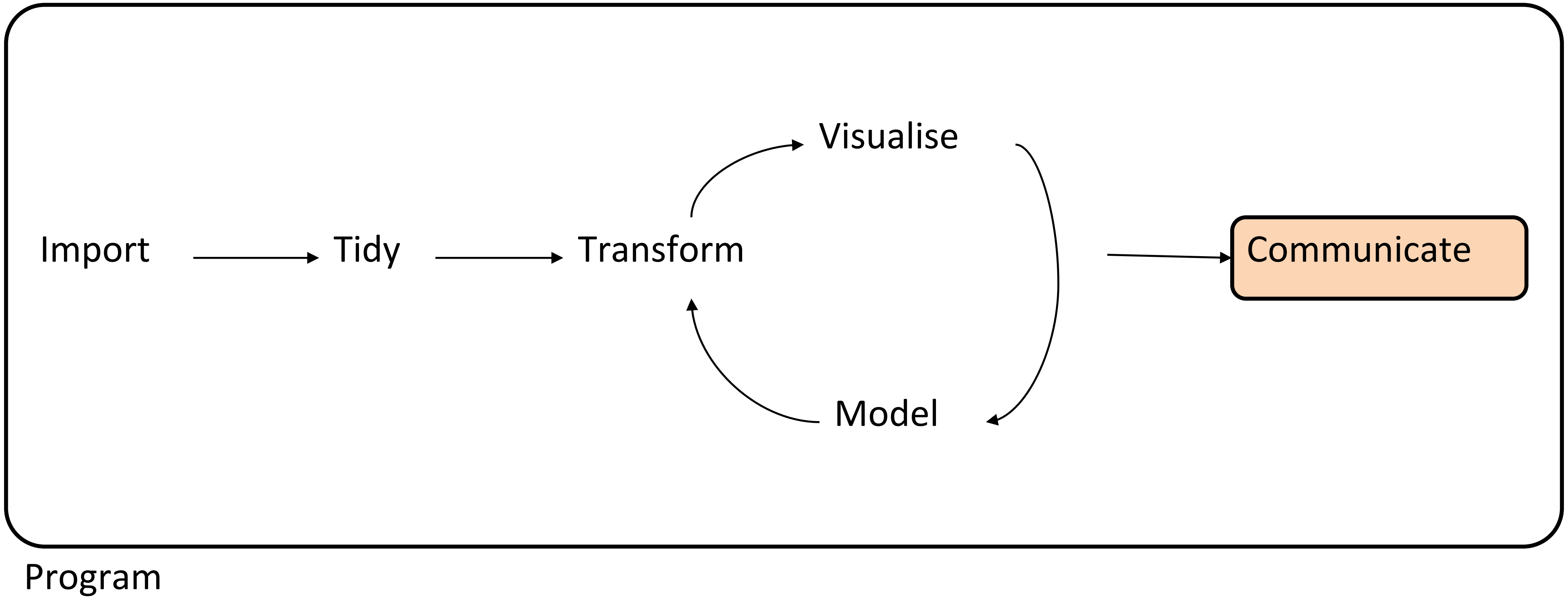
```
sportData <- summer %>%
  group_by(Sport) %>%
  nest()

sportsModels <- map(sportData$data,
                  ~lm(Funding ~ Year, data = .))

sportResid <- map2_df(sportData$data,
                        sportsModels,
                        add_residuals, .id = "Sport")
```

Exercise!

1. Iterate over the **mtcars** dataset – finding the mean value for each column
2. For each course – find the variance (using the `var` function) of the test scores.


Import
Tidy
Transform
Visualise
Model
Communicate
Program

Communicate

- There are no tidyverse packages directly aimed at communicating results (other than creating graphics)
- There are lots of packages that can be used to present results:
  - `shiny`
  - `rmarkdown`
  - `flexdashboard`
  - …

Summary

Living in the Tidyverse

- Single unified approach to manipulating and analysing data
- Provides packages for all stages of the analysis lifecycle

More resources!

- Rstudio Cheat Sheets: https://www.rstudio.com/resources/cheatsheets/

- R for Data Science, *Hadley Wickham & Garrett Grolemund, O'Reilly:* http://r4ds.had.co.nz/

MANGO
SOLUTIONS

We **empower organisations** to make **informed decisions**, using advanced **analytics** and **AI/ML** techniques to meet their objectives and deliver **data-driven value**. We do this through our unique combination of people, values and strategic priorities.

@MangoTheCat

www.mango-solutions.com

info@mango-solutions.com

+44 1249 705450

Chippenham Office

Mango Solutions
2 Methuen Park
Chippenham
SN14 0GB

London Office

Dawson House
5 Jewry Street
London
EC3N 2EX

Data Week Online 2020

Share your participation

#Dataweekonline2020

Keep in touch

@JGIBristol
jgi-admin@bristol.ac.uk
bristol.ac.uk/golding