

What is R?



- **R** is a 'statistical programming language'
- created mid-90s as a free version of **S**
- widespread adoption since v2 (2004)

- **RStudio** is an 'integrated development environment' (IDE)
- created 2011 'to improve **R** experience'
- widespread adoption since 2012



R vs RStudio

This is R

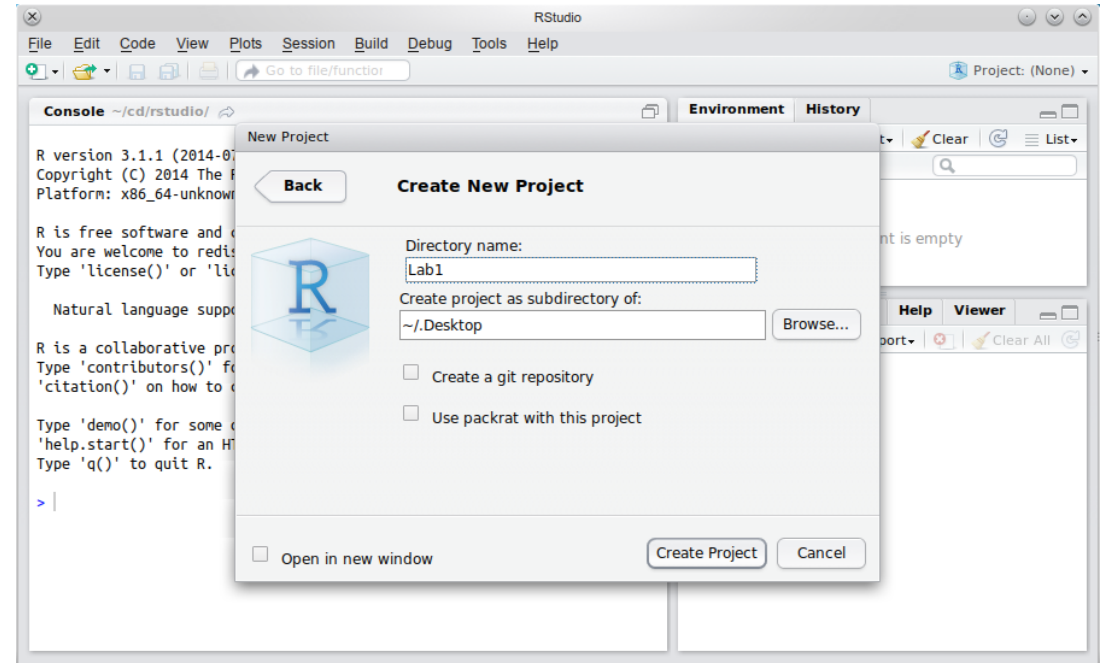
```
model <- lm(RT ~ (age+freq+handedness)^2, data=words)
summary(model)
```

R vs RStudio

This is R

```
model <- lm(RT ~ (age+freq+handedness)^2, data=words)
summary(model)
```

This is RStudio



RMarkdown



- **RMarkdown** is a 'text markup language'
- created 2012 as a markup language for **R**
- widespread adoption since 2015

RMarkdown

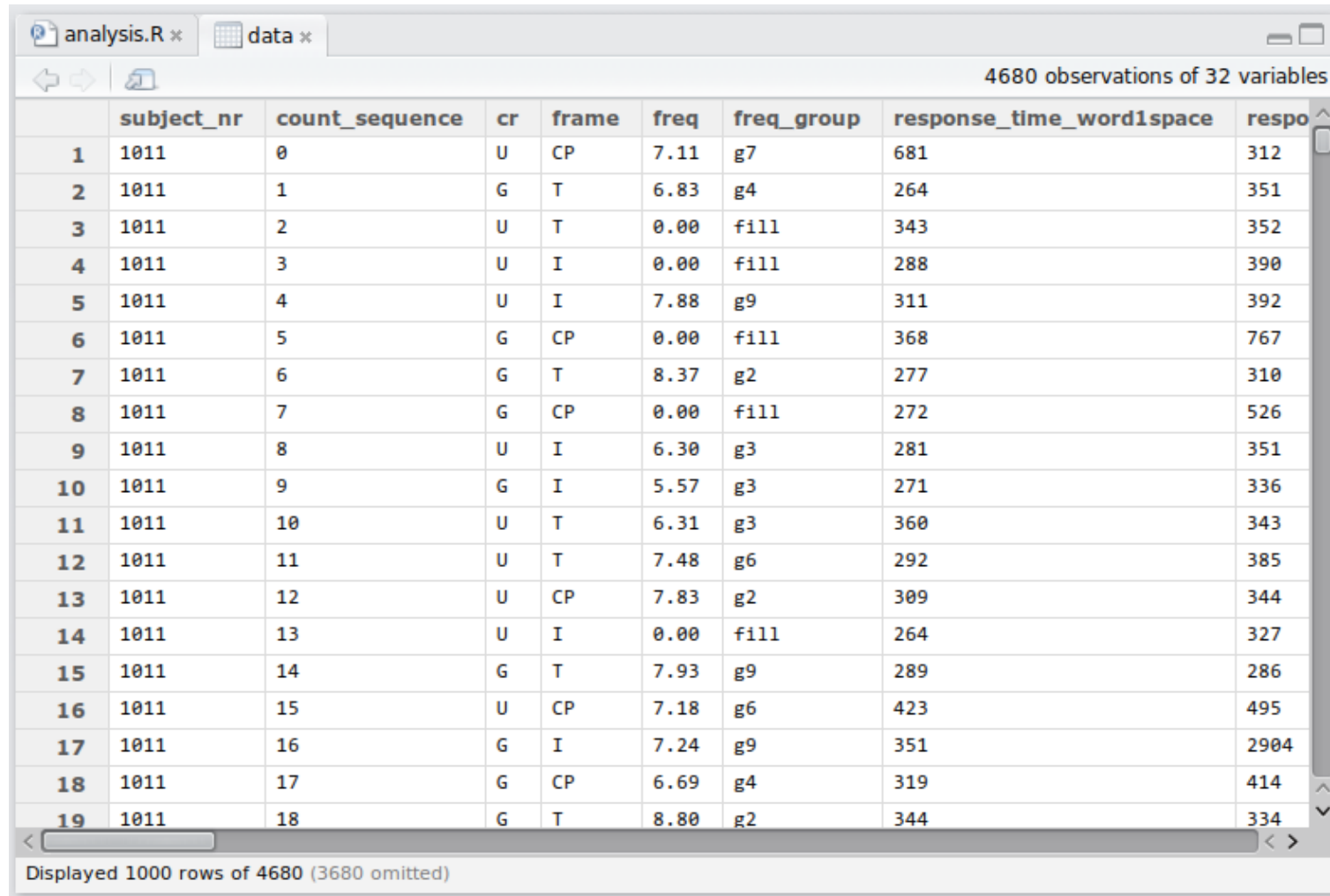
```
### About RMarkdown
_This_ is some RMarkdown, which uses 'simple' codes to mark up text.
- it can include R code like `r sqrt(2)`
- it's simple to format things like bulleted lists
  + or even sublists
```

About RMarkdown

This is some **RMarkdown**, which uses 'simple' codes to mark up text.

- it can include R code like 1.4142
- it's simple to format things like bulleted lists
 - or even sublists

Managing Datasets



analysis.R * data * 4680 observations of 32 variables

	subject_nr	count_sequence	cr	frame	freq	freq_group	response_time_word1space	respo
1	1011	0	U	CP	7.11	g7	681	312
2	1011	1	G	T	6.83	g4	264	351
3	1011	2	U	T	0.00	fill	343	352
4	1011	3	U	I	0.00	fill	288	390
5	1011	4	U	I	7.88	g9	311	392
6	1011	5	G	CP	0.00	fill	368	767
7	1011	6	G	T	8.37	g2	277	310
8	1011	7	G	CP	0.00	fill	272	526
9	1011	8	U	I	6.30	g3	281	351
10	1011	9	G	I	5.57	g3	271	336
11	1011	10	U	T	6.31	g3	360	343
12	1011	11	U	T	7.48	g6	292	385
13	1011	12	U	CP	7.83	g2	309	344
14	1011	13	U	I	0.00	fill	264	327
15	1011	14	G	T	7.93	g9	289	286
16	1011	15	U	CP	7.18	g6	423	495
17	1011	16	G	I	7.24	g9	351	2904
18	1011	17	G	CP	6.69	g4	319	414
19	1011	18	G	T	8.80	g2	344	334

Displayed 1000 rows of 4680 (3680 omitted)

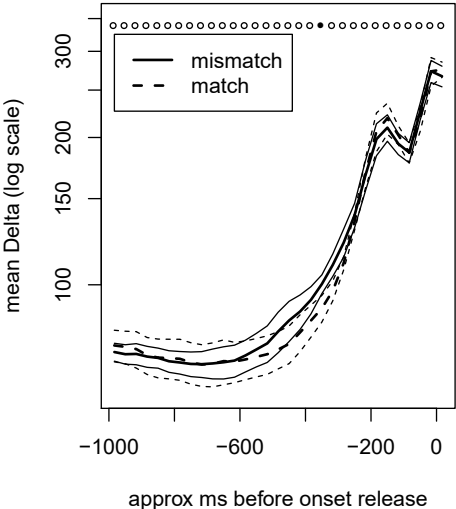
Doing Statistics

```
Generalized linear mixed model fit by maximum likelihood (Laplace
Approximation) [glmerMod]
Family: binomial ( logit )
Formula: DV ~ sc(Fv0) * sc(EvC) + (1 | Code) + (0 + (sc(Fv0) * sc(EvC)) |
Code) + (1 | Item)
Data: feminine
Control: glmerControl(optimizer = "bobyqa")

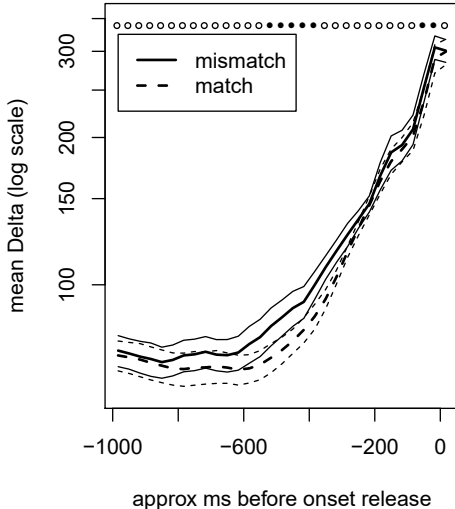
      AIC      BIC   logLik deviance df.resid
 879.3   943.6  -427.7   855.3    1558
...
Fixed effects:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)   -1.0566     1.1485  -0.92  0.35758
sc(Fv0)         1.2453     0.3505   3.55  0.00038 ***
sc(EvC)        -0.0915     0.3080  -0.30  0.76638
sc(Fv0):sc(EvC) 0.0221     0.6321   0.04  0.97207
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
...
```

Publication-Quality Graphics

Frame-by-Frame /k/ Production



Frame-by-Frame /t/ Production



Data Visualisation



<https://www.facebook.com/notes/facebook-engineering/visualizing-friendships/469716398919/>

RMarkdown: Books

For example: <https://bookdown.org/csgillespie/efficientR/>

The screenshot shows the 'Efficient R programming' book page. On the left is a table of contents with '1 Introduction' selected. The main content area displays the '1 Introduction' chapter, which includes a paragraph about the book's scope and a 'Prerequisites' section.

Efficient R programming

Welcome to Efficient R Programming

Preface

1 Introduction

- Prerequisites
- 1.1 Who this book is for and how ...
- 1.2 What is efficiency?
- 1.3 What is efficient R programmi...
- 1.4 Why efficiency?
- 1.5 Cross-transferable skills for ef...
- 1.6 Benchmarking and profiling
- 1.7 Book resources

2 Efficient set-up

3 Efficient programming

4 Efficient workflow

5 Efficient input/output

6 Efficient data carpentry

7 Efficient optimization

8 Efficient hardware

9 Efficient collaboration

10 Efficient learning

1 Introduction

This chapter introduces the book. It describes the wide range of people it was written for, in terms of R and programming experience, and how you can get the most out of it. Anyone setting out to improve efficiency should have an understanding of precisely what they mean by the term, and this is discussed, with reference to *algorithmic* and *programmer* efficiency in Section 1.2, and with reference to R in particular in 1.3. It may seem obvious, but it's also worth thinking about *why* anyone would bother with efficient code now that powerful computers are cheap and accessible. This is covered in Section 1.4.

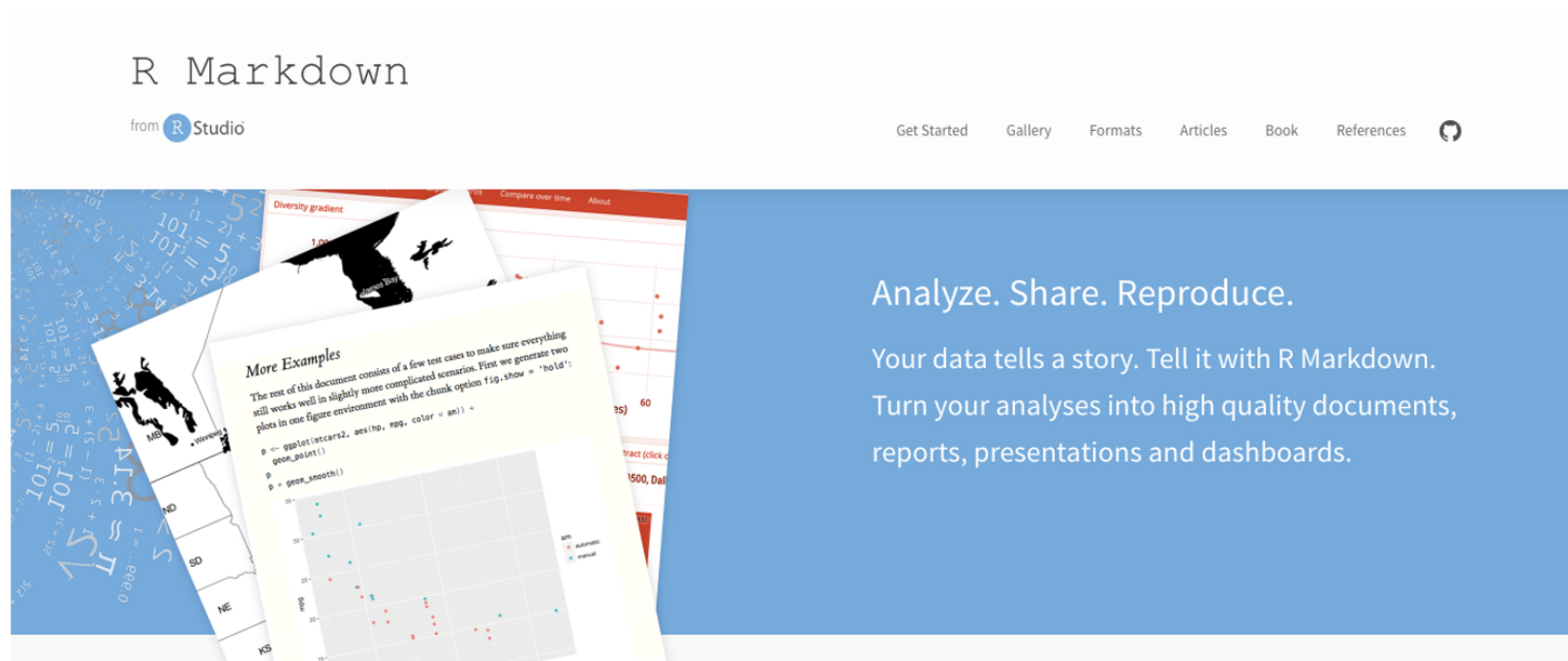
This book happily is not completely R-specific. Non R programming skills that are needed for efficient R programming, which you will develop during the course of following this book, are covered in Section 1.5. Unusually for a book about programming, this section introduces touch typing and consistency: cross-transferable skills that should improve your efficiency beyond programming. However, this is first and foremost a book about programming and it wouldn't be so without code examples in every chapter. Despite being more conceptual and discursive, this opening chapter is no exception: its penultimate section (1.6) describes these two essential tools in the efficient R programmer's toolbox, and how to use them with a couple of illustrative examples. The final thing to say at the outset is how to use this book in conjunction with the book's associated package and its source code. This is covered in Section 1.7.

Prerequisites


As emphasised in the next section, it's useful to run code and experiment as you read. This *Prerequisites* section ensures you have the necessary packages for each chapter. The prerequisites for this chapter are:


R Markdown: Websites

For example: <https://rmarkdown.rstudio.com/>



R Markdown

from  R Studio

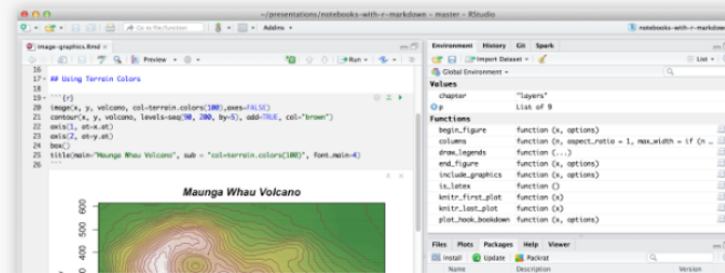
Get Started Gallery Formats Articles Book References 

Analyze. Share. Reproduce.

Your data tells a story. Tell it with R Markdown.

Turn your analyses into high quality documents, reports, presentations and dashboards.

R Markdown documents are fully reproducible. Use a productive **notebook interface** to weave together narrative text and code to produce elegantly formatted output. Use **multiple**



```
17 # Using Terrain Colors
18
19 # [ ]
20 image(x, y, volcano, col=terrain.colors(100),axes=FALSE)
21 canvas(x, y, volcano, levels=cag(50, 200, by=5), add="R", col="brown")
22 axis(2, at=y.ar)
23 axis(1, at=x.ar)
24 hwd()
25 title(main="Maunga Whau Volcano", sub = "col=terrain.colors(100)", font.main=4)
26
```



- USMR course materials (the readings, these lecture slides, etc) are all created in **RStudio**, using **RMarkdown** and **R**

A screenshot of the RStudio interface. The main editor window shows RMarkdown code for a document titled 'lecture_1.Rmd'. The code includes a bulleted list, a section header '### About RMarkdown', and a paragraph of text. The console window at the bottom shows the execution of the `colors=brewer.pal(12, 'Dark2')` command, resulting in a warning message. The right-hand pane shows the 'Environment' tab with a table of data objects: 'heights' (102 obs. of 4 variables) and 'pp' (List of 2). The bottom right corner features a word cloud with various words in different colors and orientations.

```
lecture_1.Rmd* x tweaks.css x
- it's simple to format things like bulleted lists
  + or even sublists
...
]

.pt4[
### About RMarkdown
_This_ is some RMarkdown, which uses 'simple' codes to mark up
text.

- it can include R code like `r sqrt(2)`
- it's simple to format things like bulleted lists
  + or even sublists
]

|---
192:1 | About RMarkdown | R Markdown |

Console Terminal x R Markdown x Jobs x
~/teaching/usmr/usmr_lectures/
+ colors=brewer.pal(12, 'Dark2')
Warning message:
In brewer.pal(12, "Dark2") :
```

Environment	History	Connections	Git	Tutorial
Global Environment				
Data				
heights	102 obs. of 4 variables			
pp	List of 2			

Files Plots Help Viewer

Zoom Export

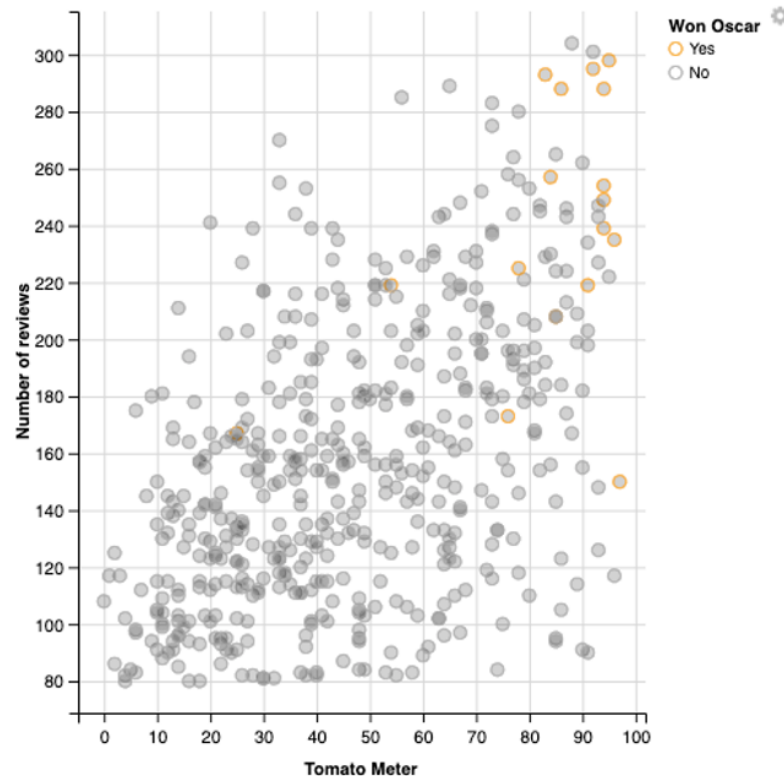
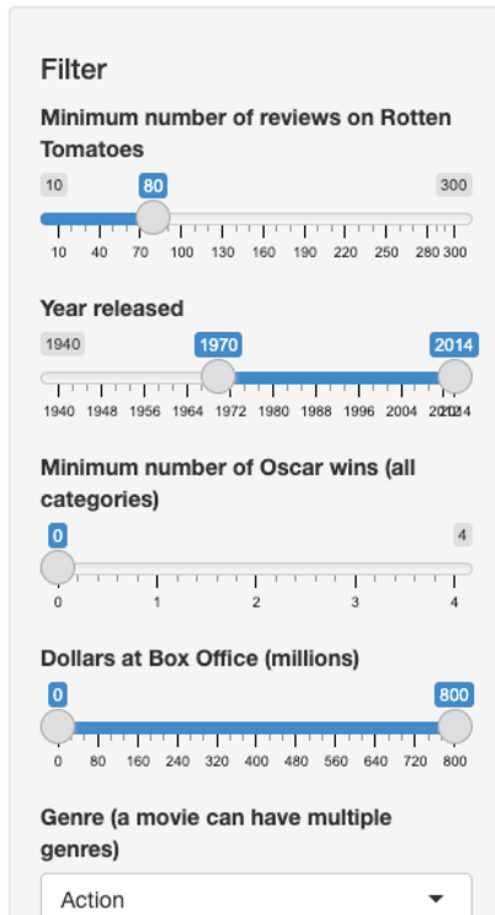
Word cloud: daught, two, way, may, see, one, eth, ask, ever, miss, tell, shall, might, soon, dear, found, cer, n, manner, se, bennet, jane, time, still, m, young, aed

Online Interactive Visualisation

For example: <https://shiny.rstudio.com/gallery/movie-explorer.html>

Shiny from R Studio

Movie explorer




Number of movies selected:
505

Online Interactive Visualisation

For example: <https://gallery.shinyapps.io/086-bus-dashboard/>

Twin Cities Buses



Route

2

Show

- Northbound
- Southbound
- Eastbound
- Westbound

Note: a route number can have several different trips, each with a different path. Only the most commonly-used path will be displayed on the map.

Zoom to fit buses

Refresh interval

1 minute

Data refreshed 25 seconds ago.

Refresh now

Source data updates every 30 seconds.

Color	Direction	Number of vehicles
■	Northbound	0
■	Southbound	6

The R Community

- if it serves no purpose, chances are that someone's already done it too

```
library(cowsay)
say("hello USMR")
```

```
##
## -----
## hello USMR
## -----
##      \
##       \
##        \
##         | \_ _ _ / |
##        ==) ^Y^ (==
##           ^
##          )=*(
##         /       \
##        /| | | | \
##       \| | | | / \
##      jgs // _ // _ _ _ /
##           \_)
##
```

Why Use R?

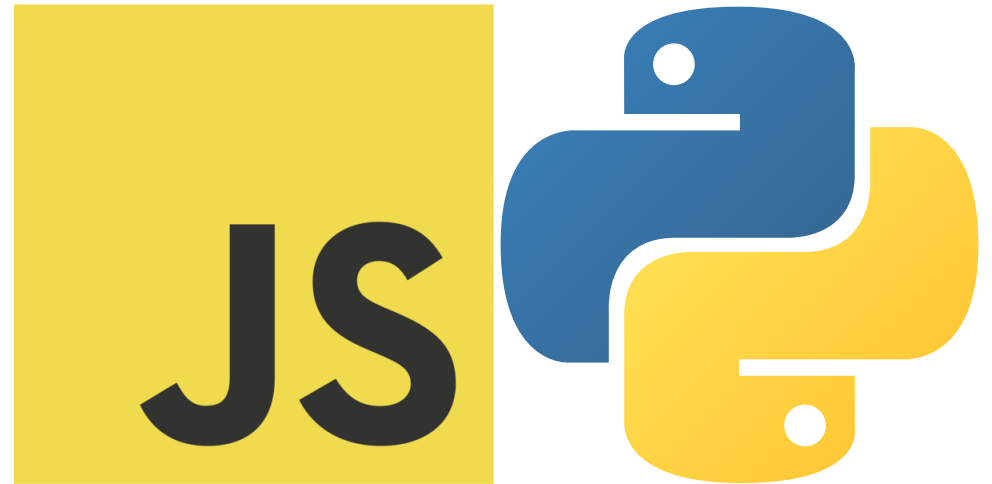
- because it's a *language*, I can easily show you what I did and you can copy it
- because it's a *language*, statisticians can use it to implement leading-edge stats
- because it's *free*, anyone can use it---and anyone can access your research
- because it's *open source*, anyone can fix or improve R

Devilish stuff

doing stats

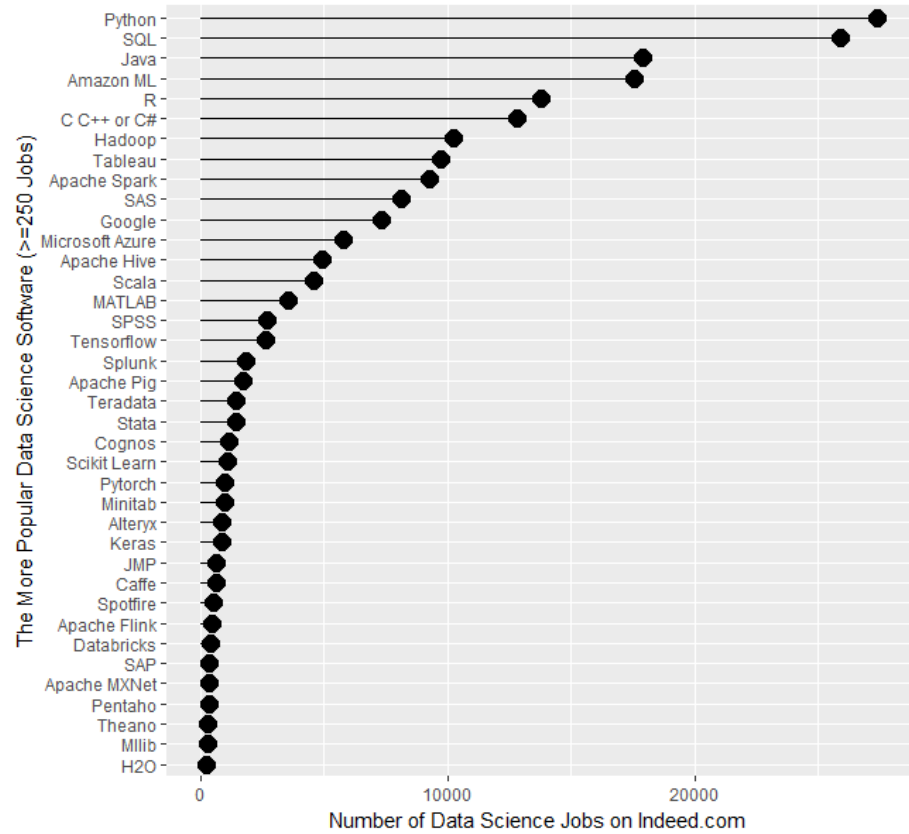
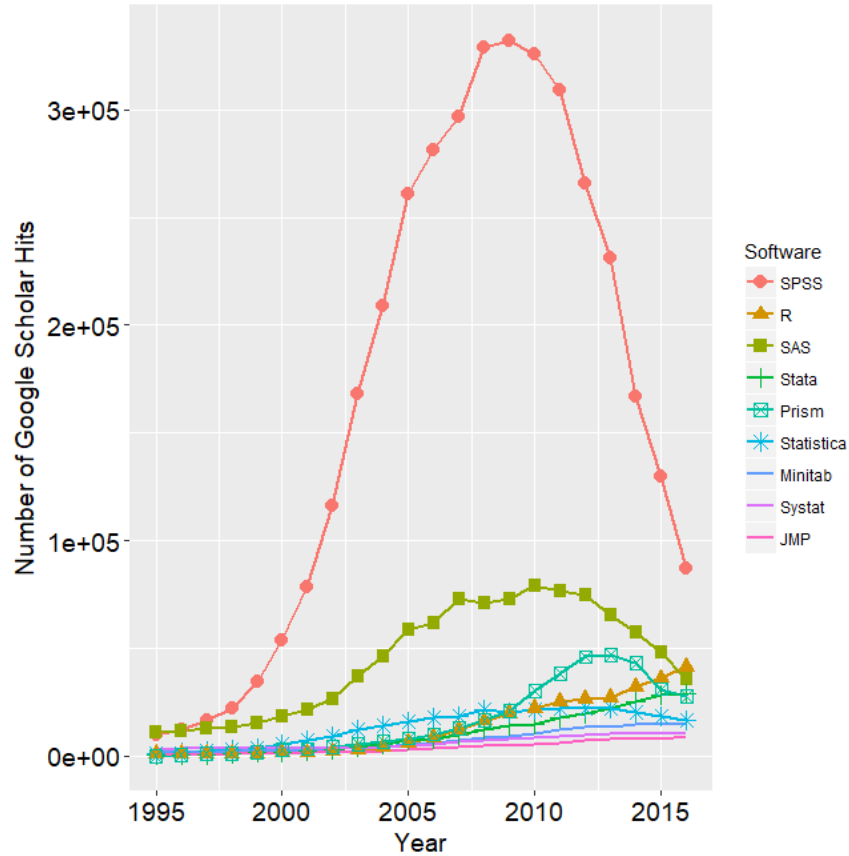


coding



NB all indices in R start at 1

Why use R??



<https://r4stats.com/articles/popularity>

Data in R

- you can type **data** directly in to R

```
# a number  
1.2
```

```
## [1] 1.2
```

```
# characters (a string)  
"fáilte"
```

```
## [1] "fáilte"
```

- and you can do **operations** on data

```
1.2 + 7 * 2
```

```
## [1] 15.2
```

Variables



- you can assign data to **variables**

```
bodyTemp <- 37.8
```

- and use those variables

```
bodyTemp * (9/5) + 32 # to Fahrenheit
```

```
## [1] 100
```

- **NB** spelling/capitalization matter

```
BodyTemp - 37
```

```
## Error in eval(expr, envir, enclos): object 'BodyTemp' not found
```

Statistics is about **groups** of things

```
allTemps <- c(37.8, 0, 37.4)
# vector maths
allTemps * (9/5) + 32
```

```
## [1] 100.04 32.00 99.32
```

- note the **vectorization** of the calculation
- R is designed from the bottom up to deal with groups



Not everything is a number

```
allHair <- c("brown","white","black")  
allHair
```

```
## [1] "brown" "white" "black"
```

- these are called **character strings**
 - can be anything
- **categories** (nominal data) are from a limited set
 - called **factors** in R

```
as.factor(allHair)
```

```
## [1] brown white black  
## Levels: black brown white
```



Basic types of data (stats)

- **Nominal**
(‘names of things’: e.g., hair colour)
- **Ordinal**
(order, no number: e.g., small-medium-large)
- **Interval**
(number without a true zero: e.g., body temp in °C)
- **Ratio**
(number with a true zero: e.g., height)



NOIR in R

Type	R Variable Type
Nominal	character/factor
Ordinal	number
Interval	number
Ratio	number

- nominal

```
allHair <- as.factor(c("brown", "white", "black"))
allHair
```

```
## [1] brown white black
## Levels: black brown white
```

- interval

```
allTemps <- c(37.8, 0, 37.4)
allTemps
```

```
## [1] 37.8 0.0 37.4
```

NOIR in R

Type	R Variable Type
Nominal	character/factor
Ordinal	number
Interval	number
Ratio	number

- nominal

```
allHair <- as.factor(c("brown", "white", "black"))
allHair
```

```
## [1] brown white black
## Levels: black brown white
```

- interval

```
allTemps <- c(37.8, 0, 37.4)
allTemps
```

```
## [1] 37.8 0.0 37.4
```



- ordinal data can also be represented as **ordered factors** (`as.ordered()`)

Break it down

```
allHair <- c("brown", "white", "black")
```

`allHair`

- **variable** (can be anything that isn't *reserved*)

`<-`

- **assignment** ("goes in to")

`c()`

- **function** (`c()` *combines* its **arguments**)

`"brown"`

- **character** (arbitrary sequence of symbols)

Dataframes

- data can be grouped into a **dataframe**
- each *line* represents one set of observations
- each *column* represents one type of information
 - (a bit like a spreadsheet)

```
people <- data.frame(name=c('Johanna', 'Casper', 'Steve'),  
                    temp=allTemps,  
                    hair=as.factor(allHair),  
                    height=c(132, 205, 181))
```

```
people
```

```
##      name temp  hair height  
## 1 Johanna 37.8 brown   132  
## 2 Casper  0.0 white   205  
## 3 Steve  37.4 black   181
```



Can you run an **function** on a **dataframe**?

- youbetcha!

```
summary(people)
```

```
##      name          temp      hair      height
## Length:3      Min.   : 0.0  black:1  Min.    :132
## Class :character 1st Qu.:18.7  brown:1 1st Qu.:156
## Mode  :character Median :37.4  white:1 Median :181
##                               Mean  :25.1      Mean  :173
##                               3rd Qu.:37.6      3rd Qu.:193
##                               Max.   :37.8      Max.   :205
```

- or on a vector

```
mean(people$temp) # just the temp column from people
```

```
## [1] 25.07
```

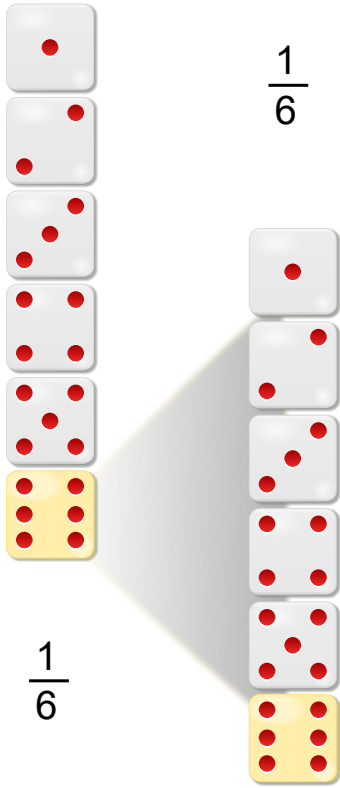
We know a little about R

- we've seen some R code
- we know about basic data types
- we know what variables are
- we've seen vectors, and dataframes
- we've seen a couple of examples of functions

Part 3



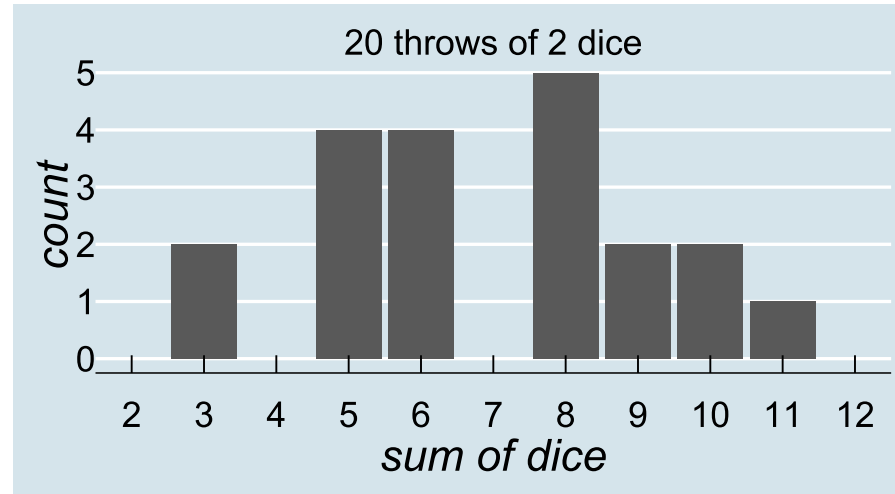
How likely are you to throw 12 with two dice?



- pretty easy to work out
- one-in-six chance of throwing a six
- one-in-six chance of throwing a second six
 - NB., these observations are *independent*
 - (wouldn't matter if you threw one dice twice or two dice together)
- $\frac{1}{36}$ chance of throwing two sixes

Are my dice fair?

- one way to find out: throw two dice many times and count the outcomes



What would fair dice look like?



- we need a lot of throws
- first rule of coding: be lazy
- let the computer do the work

Using RStudio

The screenshot displays the RStudio interface with the following components:

- Source Editor:** Contains R code for a faceted plot. The code uses `ggplot2` to create a plot of 'Proportion of looks' vs 'Time from target onset (ms)'. It features two panels: 'like' absent and 'like' present. The plot includes a shaded gray area representing a confidence interval, a dashed vertical line at 0 ms labeled '[target]', and a legend for 'Looks to' with 'cohort competitor' (blue) and 'unrelated' (orange).
- Environment:** Shows loaded data objects: `bdata` (39680 obs. of 19 variables) and `gdata` (124 obs. of 6 variables). Functions include `do_graphs` (631.8 kB) and `make_gdata`.
- Plots:** Displays two faceted plots. The left plot is titled "'like' absent" and the right is "'like' present". Both plots show the proportion of looks over time, with a shaded gray area and a dashed vertical line at 0 ms. The legend indicates 'Looks to' with 'cohort competitor' (blue) and 'unrelated' (orange).
- Console:** Shows the execution of the R code, including the `labs` function and the `panel1cohcompref` and `panel2cohcompref` functions.

Using RStudio

The image shows a screenshot of the RStudio interface. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. Below the menu is a toolbar with icons for file operations and execution. The main editor window displays R code for creating a plot. The Environment pane on the right shows loaded data objects: bdata (39680 obs. of 19 variables), gdata (124 obs. of 6 variables), and do_graphs (Large function (631.8 kB)). The Files pane shows the current project directory. The bottom pane is split into Console and Terminal, showing the execution of the R code. The plot area displays two side-by-side line graphs: 'like' absent and 'like' present. Both graphs show the 'Proportion of looks' on the y-axis (0.0 to 0.5) against 'Time from target onset (ms)' on the x-axis (-800 to 400). A vertical dashed line at 0 ms is labeled '[target]'. The legend indicates 'Looks to' with 'cohort competitor' (blue) and 'unrelated' (orange). The 'like' absent plot shows a higher proportion of looks to the cohort competitor before the target onset. The 'like' present plot shows a higher proportion of looks to the cohort competitor after the target onset.

control projects

EDITOR
write code and reports
(stuff you're going to keep)

various info about the
state of R

look at stuff like graphics,
help, files on the system

CONSOLE
muck about; test stuff;
see results

```
labs(title = "\\like\\ absent")

panel2cohcompref <- gdata %>% filter(likeornot == 'like') %>%
ggplot(aes(x=time_tgtonset, y=propfix, color=lookto, fill=lookto,
ymax=max(propfix),
geom_rect(mapping=aes(ymin=0, ymax=1, xmin=-433,
xmax=200),fill="lightgray",linetype=0,alpha=0.5) +
geom_point(aes(ymin=0, ymax=1, xintercept=propfix, fill=lookto),alpha=0.3,color = NA) +
geom_line(position=nd, size=2) +
geom_vline(xintercept=c(0,-633), linetype="dashed", size=1) +
geom_vline(xintercept = c(0,(-633+286)), linetype="dashed", size=1)
+
scale_color_calc() +
scale_fill_calc() +
coord_cartesian(ylim = c(0,0.5), xlim=c(-800,501)) +
scale_x_continuous(breaks=seq(-9000,9000,200)) +
scale_y_continuous(breaks=seq(0,1,0.1)) +
labs(x = "Time from target onset (ms)", y = "Proportion of looks") +
annotate("text", size = 7, x = -770, y = 0.02, label = "...the") +
do_graphs(gdata)

labs(x = "Time from target onset (ms)", y = "Proportion of looks")
+
annotate("text", size = 7, x = -770, y = 0.02, label = "...the") +
annotate("text", size = 7, x = -540, y = 0.02, label = "like...")
+
annotate("text", size = 7, x = 100, y = 0.021, label = "[target]")
+
theme(legend.background = element_rect(fill="lightblue", size=15),
legend.position=(0.52, 0.15)) +
labs(title = "\\like\\ absent", color="Looks to", fill="Looks to")
+
panel1cohcompref + panel2cohcompref
```

create some dice

Now we can throw dice a *lot* of times

```
dice <- function(num=1) {  
  sum(sample(1:6, num, replace=TRUE))  
}  
  
dice()
```

```
## [1] 1
```

Now we can throw dice a *lot* of times

```
dice <- function(num=1) {  
  sum(sample(1:6, num, replace=TRUE))  
}
```

```
dice()
```

```
## [1] 1
```

```
dice(2)
```

```
## [1] 7
```

Throw two dice many times

```
replicate(250,dice(2))
```

```
## [1] 11 7 7 4 10 9 7 6 12 3 4 9 6 11 10 6 7 8 9 9 7 10 8 6 12
## [26] 10 7 6 7 7 8 11 3 5 6 11 3 4 7 9 7 11 3 6 8 8 10 8 8 6
## [51] 3 4 7 7 10 9 9 7 5 11 7 8 5 3 9 5 4 3 8 4 11 10 5 10 6
## [76] 7 6 11 5 7 7 10 5 5 2 6 7 5 4 7 5 9 8 6 6 9 7 9 8 6
## [101] 6 8 6 4 5 5 6 4 10 9 3 4 9 5 6 5 9 3 8 5 4 7 6 5 7
## [126] 8 11 9 10 7 8 6 6 12 8 3 7 7 9 6 4 4 6 5 7 7 2 5 9 12
## [151] 6 11 6 7 4 5 5 8 6 8 11 7 5 7 6 4 7 11 3 5 2 7 8 8 7
## [176] 10 6 7 7 3 3 5 4 3 4 11 3 9 7 8 7 10 9 9 6 10 7 7 6 10
## [201] 9 5 9 9 7 5 7 9 3 7 5 5 3 6 7 5 4 3 8 6 11 11 6 9 7
## [226] 10 9 6 4 8 10 7 8 4 5 8 8 7 9 9 11 11 6 5 9 10 11 9 7 9
```

Throw two dice many times

```
replicate(250,dice(2))
```

```
## [1] 11 7 7 4 10 9 7 6 12 3 4 9 6 11 10 6 7 8 9 9 7 10 8 6 12
## [26] 10 7 6 7 7 8 11 3 5 6 11 3 4 7 9 7 11 3 6 8 8 10 8 8 6
## [51] 3 4 7 7 10 9 9 7 5 11 7 8 5 3 9 5 4 3 8 4 11 10 5 10 6
## [76] 7 6 11 5 7 7 10 5 5 2 6 7 5 4 7 5 9 8 6 6 9 7 9 8 6
## [101] 6 8 6 4 5 5 6 4 10 9 3 4 9 5 6 5 9 3 8 5 4 7 6 5 7
## [126] 8 11 9 10 7 8 6 6 12 8 3 7 7 9 6 4 4 6 5 7 7 2 5 9 12
## [151] 6 11 6 7 4 5 5 8 6 8 11 7 5 7 6 4 7 11 3 5 2 7 8 8 7
## [176] 10 6 7 7 3 3 5 4 3 4 11 3 9 7 8 7 10 9 9 6 10 7 7 6 10
## [201] 9 5 9 9 7 5 7 9 3 7 5 5 3 6 7 5 4 3 8 6 11 11 6 9 7
## [226] 10 9 6 4 8 10 7 8 4 5 8 8 7 9 9 11 11 6 5 9 10 11 9 7 9
```

- ...and record the result

```
d <- replicate(250,dice(2))
```

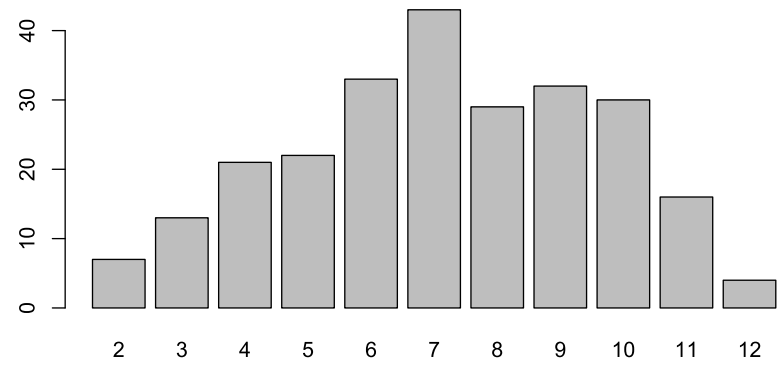

Make a graph

```
table(d)
```

```
## d
##  2  3  4  5  6  7  8  9 10 11 12
##  7 13 21 22 33 43 29 32 30 16  4
```

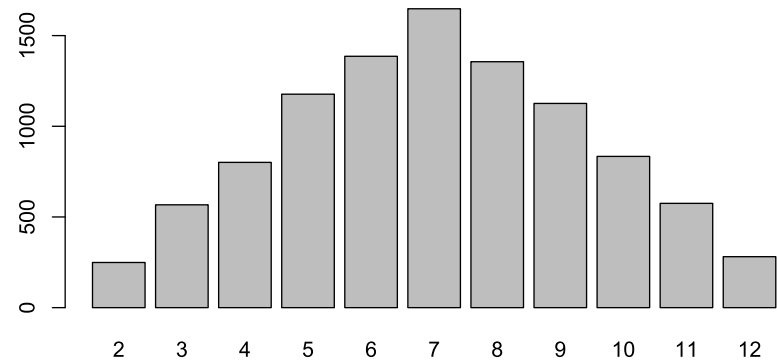
Make a graph

```
barplot(table(d))
```

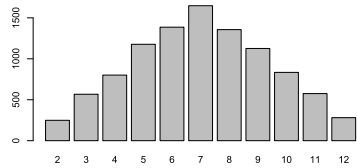


Many more throws

```
d <- replicate(10000,dice(2))  
barplot(table(d))
```



10,000 dice throws



- we can work out the proportion of throws that summed to 12

```
sum(d == 12) / 10000
```

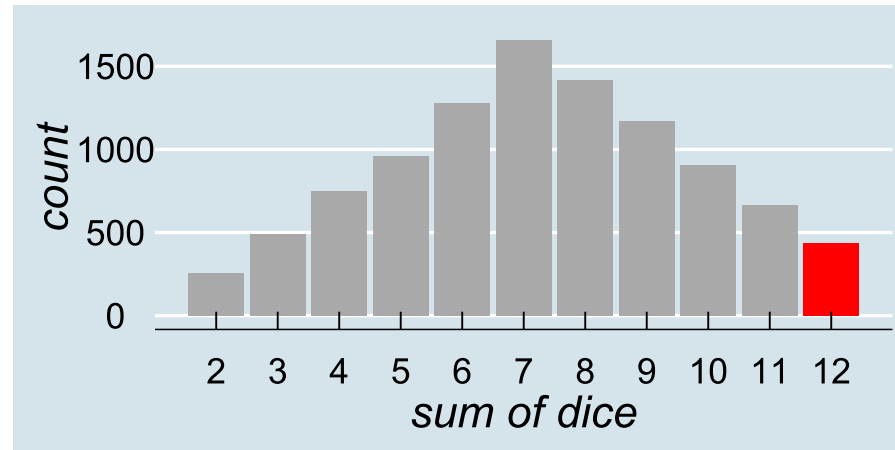
```
## [1] 0.0281
```

- and we know what that proportion should be if the dice are fair

```
1/36
```

```
## [1] 0.02778
```

Some more (fake) dice throws



are the patterns from the dice *different enough* from what we would expect from fair dice for us to conclude that they're unfair?

Statistical questions

- so the million-dollar question is a *negative* question

are we dissatisfied with the suggestion that the pattern of results we have observed should be attributed to chance?

- if we are, then maybe we can persuade you of a different explanation
- but note that the different explanation is not *proven*, it's *suggested*

Acknowledgements

- icons by Diego Lavecchia from the [Noun Project](#)