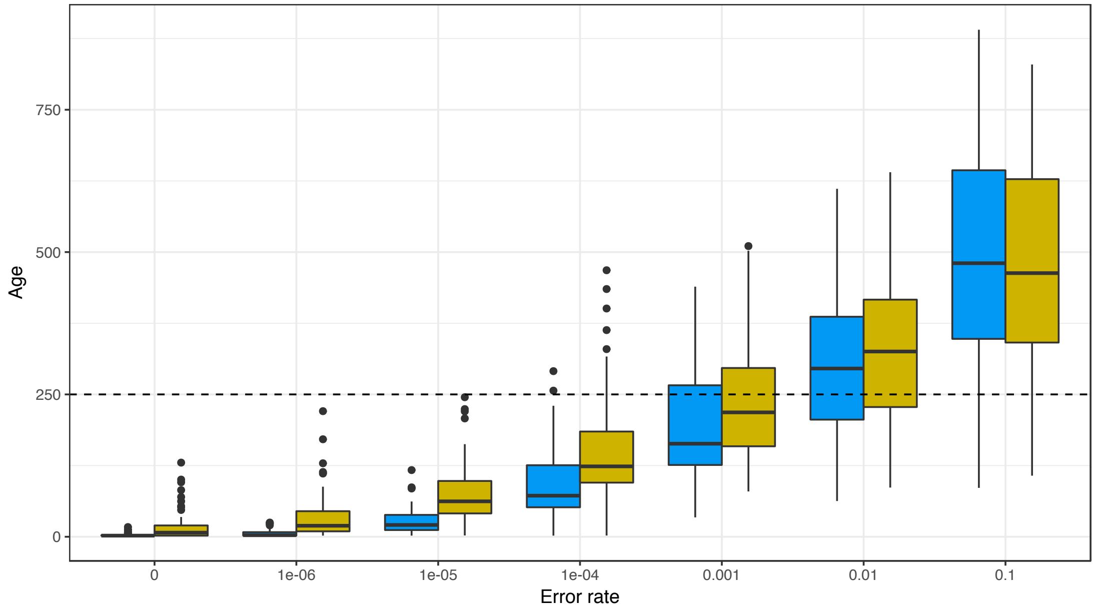
Data Science in R

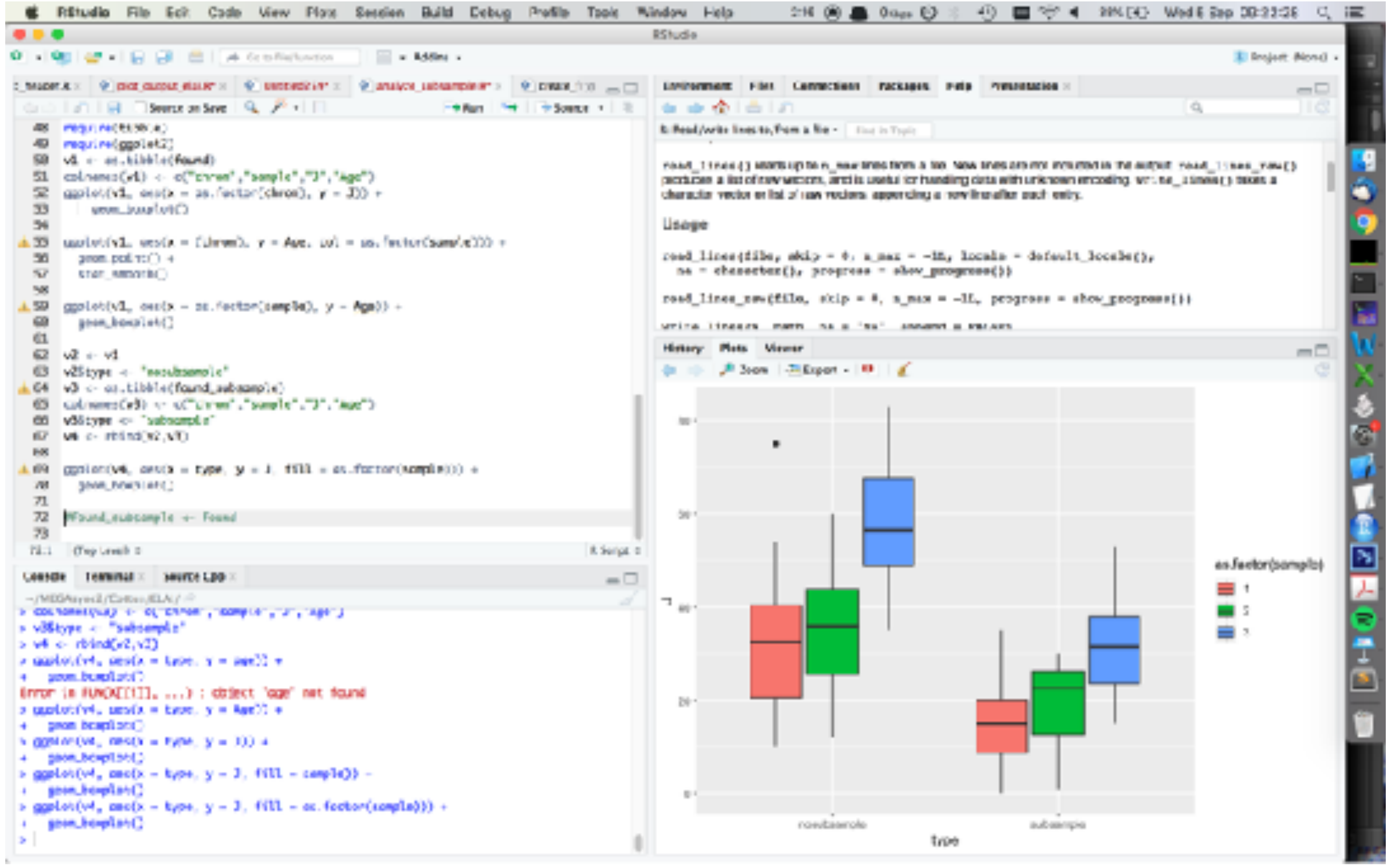
```
table1
#> # A tibble: 6 x 4
#>   country      year  cases population
#>   <chr>       <int>  <int>      <int>
#> 1 Afghanistan  1999    745   19987071
#> 2 Afghanistan  2000   2666   20595360
#> 3 Brazil       1999  37737  172006362
#> 4 Brazil       2000  80488  174504898
#> 5 China        1999 212258 1272915272
#> 6 China        2000 213766 1280428583
```

Dr. Thijs Janzen



What is R?

- Environment for statistical computing and graphics
- Free and open source
- Interpreted programming language



```
require(tidyverse)
require(ggplot2)
v1 <- as.tibble(found)
colnames(v1) <- c("chron","sample","J","Age")
ggplot(v1, aes(x = as.factor(chron), y = J)) +
  geom_boxplot()

ggplot(v1, aes(x = chron, y = Age, col = as.factor(sample))) +
  geom_point() +
  stat_smooth()

ggplot(v1, aes(x = as.factor(sample), y = Age)) +
  geom_boxplot()

v2 <- v1
v2$type <- "nosubsample"
v3 <- as.tibble(found_subsample)
colnames(v3) <- c("chron","sample","J","Age")
v3$type <- "subsample"
v4 <- rbind(v2,v3)

ggplot(v4, aes(x = type, y = J, fill = as.factor(sample))) +
  geom_boxplot()

#found_subsample <- found
```

```
> colnames(v3) <- c("chron","sample","J","Age")
> v3$type <- "subsample"
> v4 <- rbind(v2,v3)
> ggplot(v4, aes(x = type, y = age)) +
+   geom_boxplot()
Error in FUN(X[[i]], ...) : object 'age' not found
> ggplot(v4, aes(x = type, y = Age)) +
+   geom_boxplot()
> ggplot(v4, aes(x = type, y = J)) +
+   geom_boxplot()
> ggplot(v4, aes(x = type, y = J, fill = sample)) +
+   geom_boxplot()
> ggplot(v4, aes(x = type, y = J, fill = as.factor(sample))) +
+   geom_boxplot()
> 
```

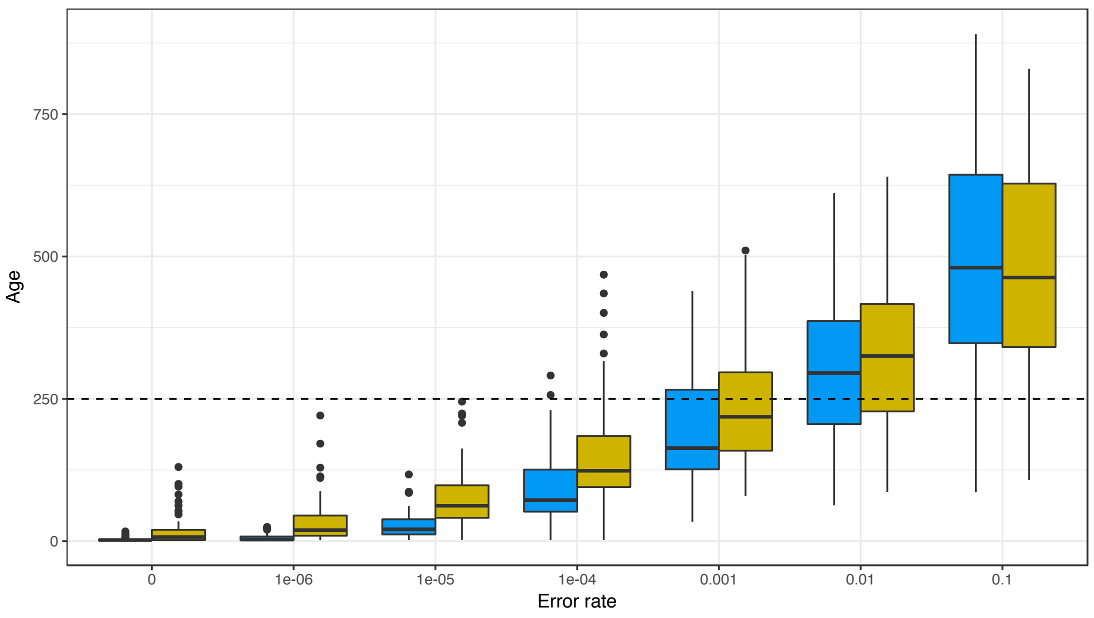
read_lines() reads up to n_max lines from a file. New lines are not included in the output. read_lines_raw() produces a list of raw vectors, and is useful for handling data with unknown encoding. write_lines() takes a character vector or list of raw vectors, appending a new line after each entry.

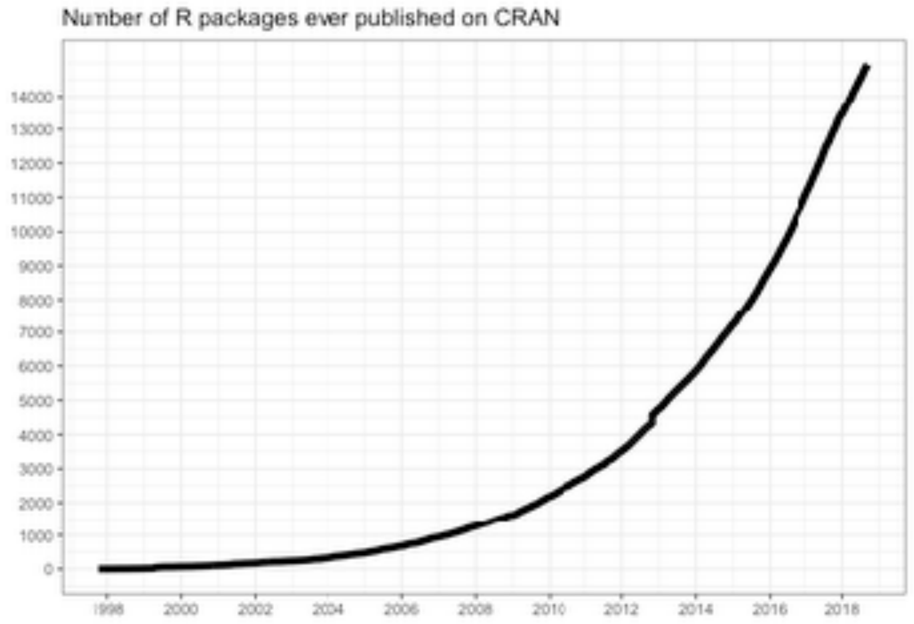
Usage

```
read_lines(file, skip = 0, n_max = -1L, locale = default_locale(),
  na = character(), progress = show_progress())

read_lines_raw(file, skip = 0, n_max = -1L, progress = show_progress())

write_lines(x, path, na = "NA", append = FALSE)
```

Why use R?

- Statistics
- Data visualisation
- Processing and tidying data
- Reproducible research
- Many available custom packages

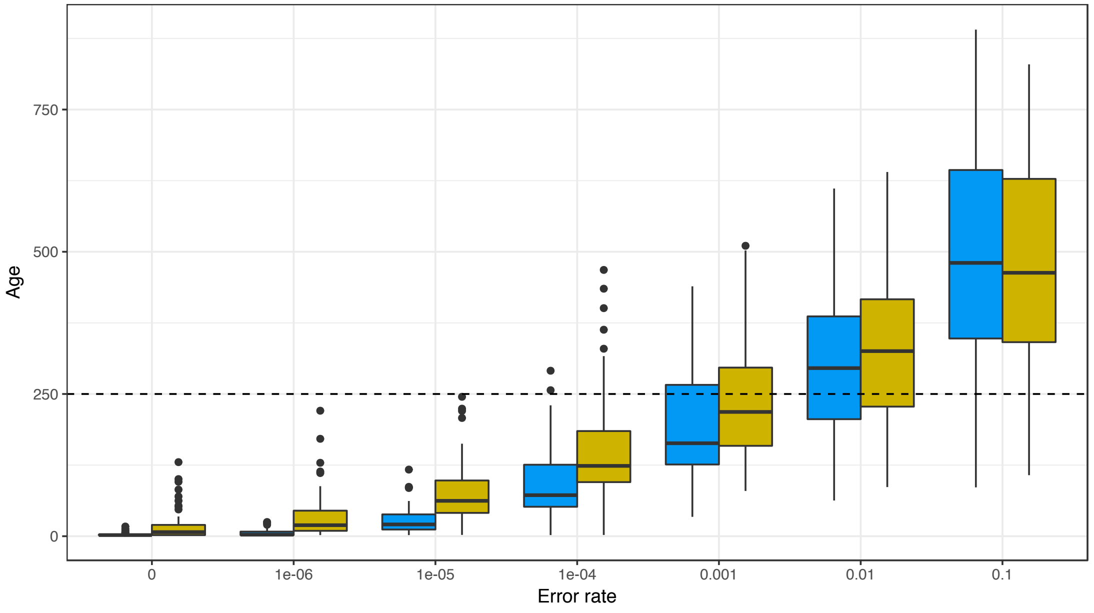


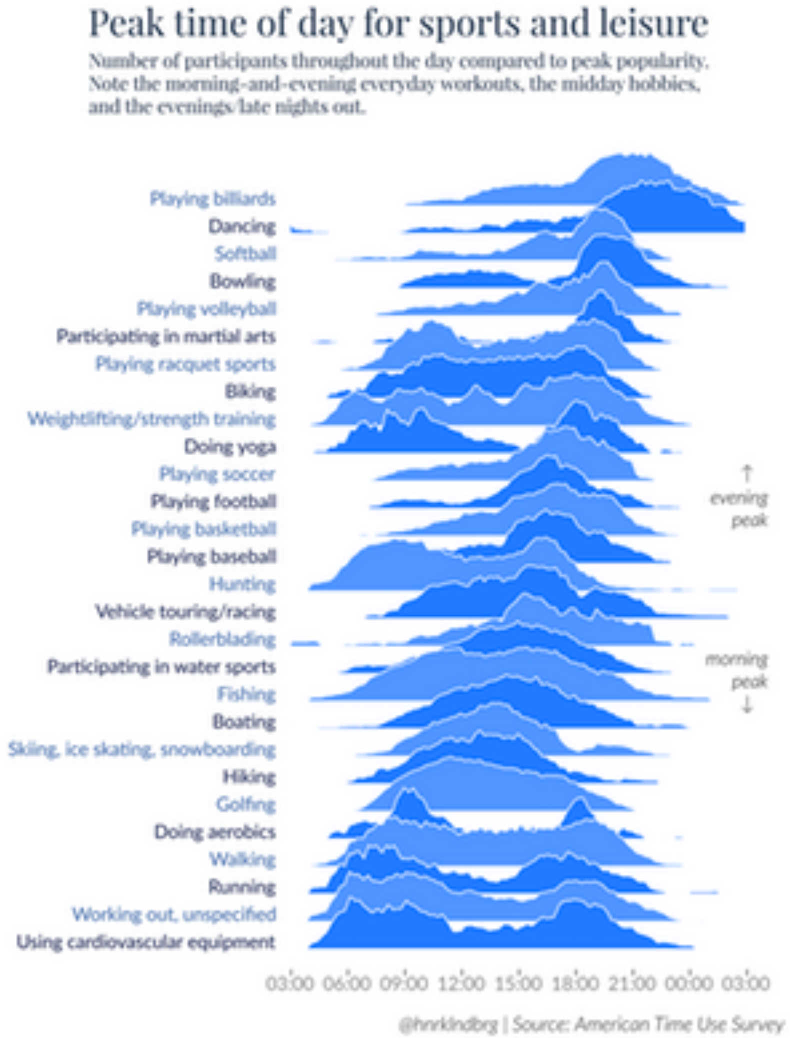


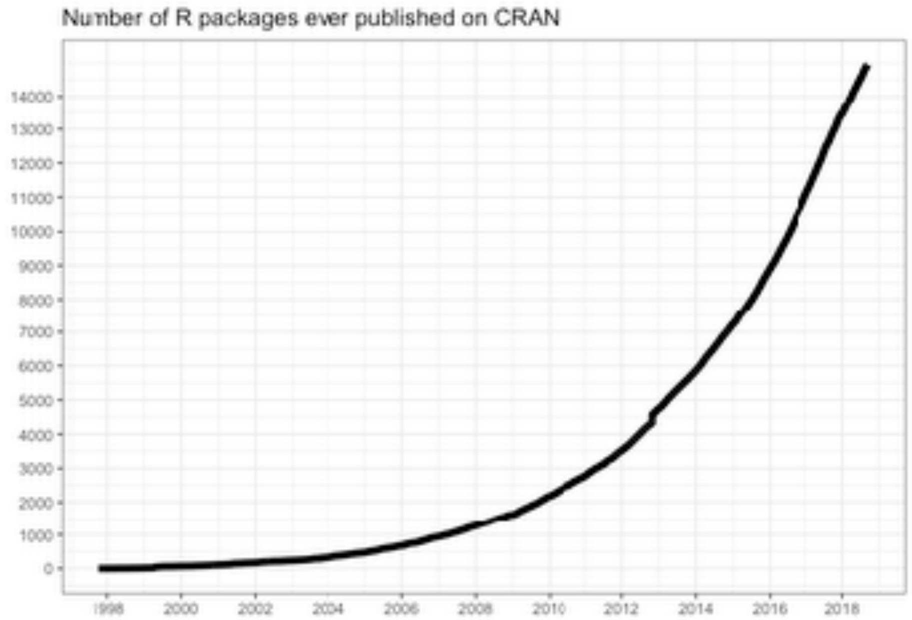
Peak time of day for sports and leisure
Number of participants throughout the day compared to peak popularity. Note the morning-and-evening everyday workouts, the midday hobbies, and the evenings/late nights out.
Playing billiards
Dancing
Softball
Bowling
Playing volleyball
Participating in martial arts
Playing racquet sports
Biking
Weightlifting/strength training
Doing yoga
Playing soccer
Playing football
Playing basketball
Playing baseball
Hunting
Vehicle touring/racing
Rollerblading
Participating in water sports
Fishing
Boating
Skiing, ice skating, snowboarding
Hiking
Golfing
Doing aerobics
Walking
Running
Working out, unspecified
Using cardiovascular equipment
↑ evening peak
morning peak ↓
03:00
06:00
09:00
12:00
15:00
18:00
21:00
00:00
03:00
@hnrklndbrg | Source: American Time Use Survey

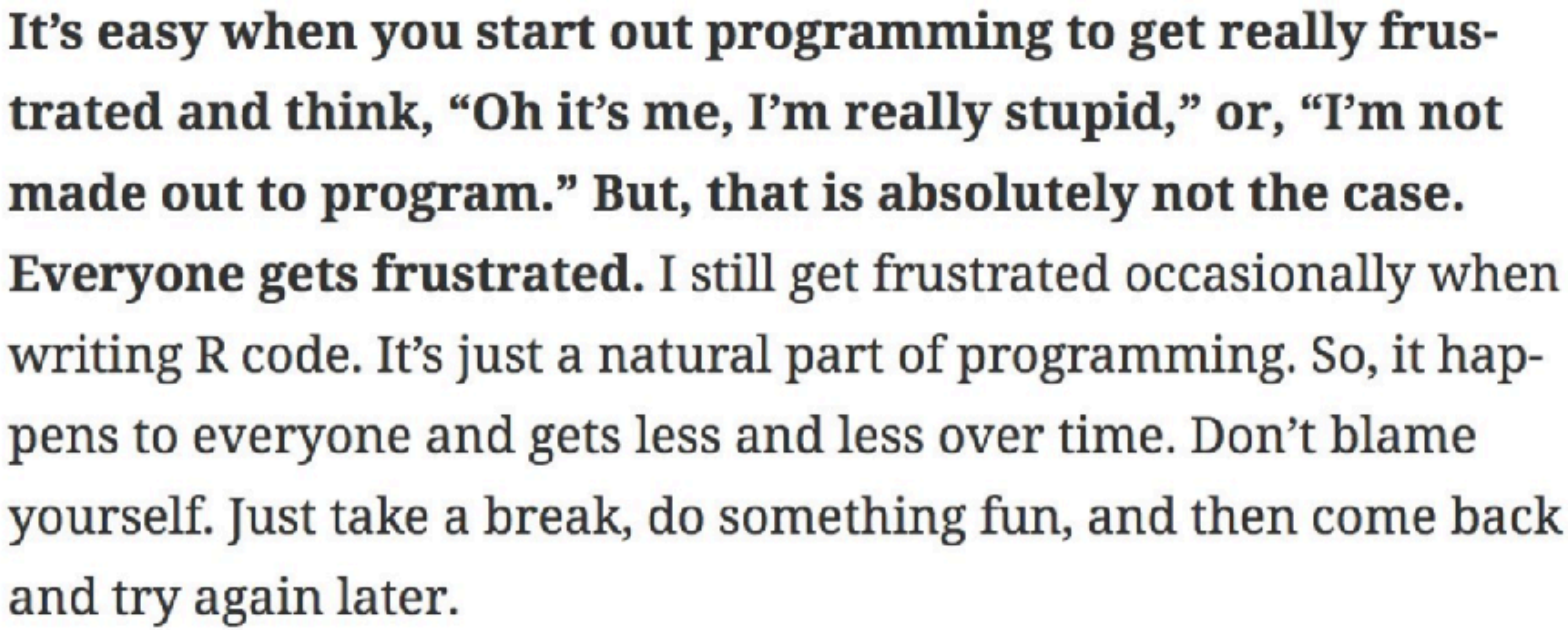
Why use R?

- Statistics
- **Data visualisation**
- **Processing and tidying data**
- Reproducible research
- Many available custom packages







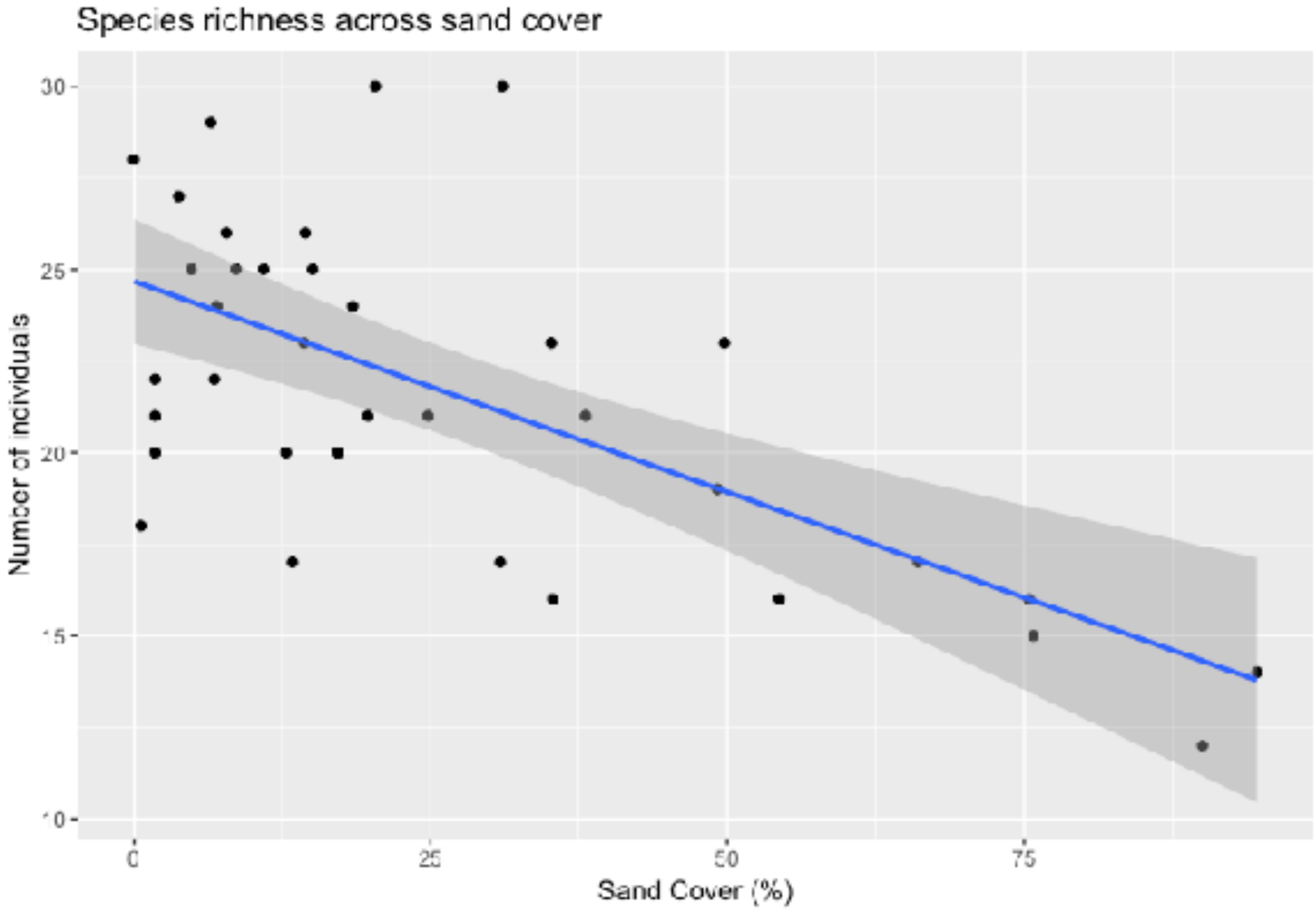


It's easy when you start out programming to get really frustrated and think, "Oh it's me, I'm really stupid," or, "I'm not made out to program." But, that is absolutely not the case. Everyone gets frustrated. I still get frustrated occasionally when writing R code. It's just a natural part of programming. So, it happens to everyone and gets less and less over time. Don't blame yourself. Just take a break, do something fun, and then come back and try again later.

Hadley Wickham,
Chief Scientist at Rstudio
Developer of the tidyverse

Goal of today

- Load data into R
- Re-structure data to improve handling: ‘tidying data’
- Plot results
- Understand basics of tidy workflow



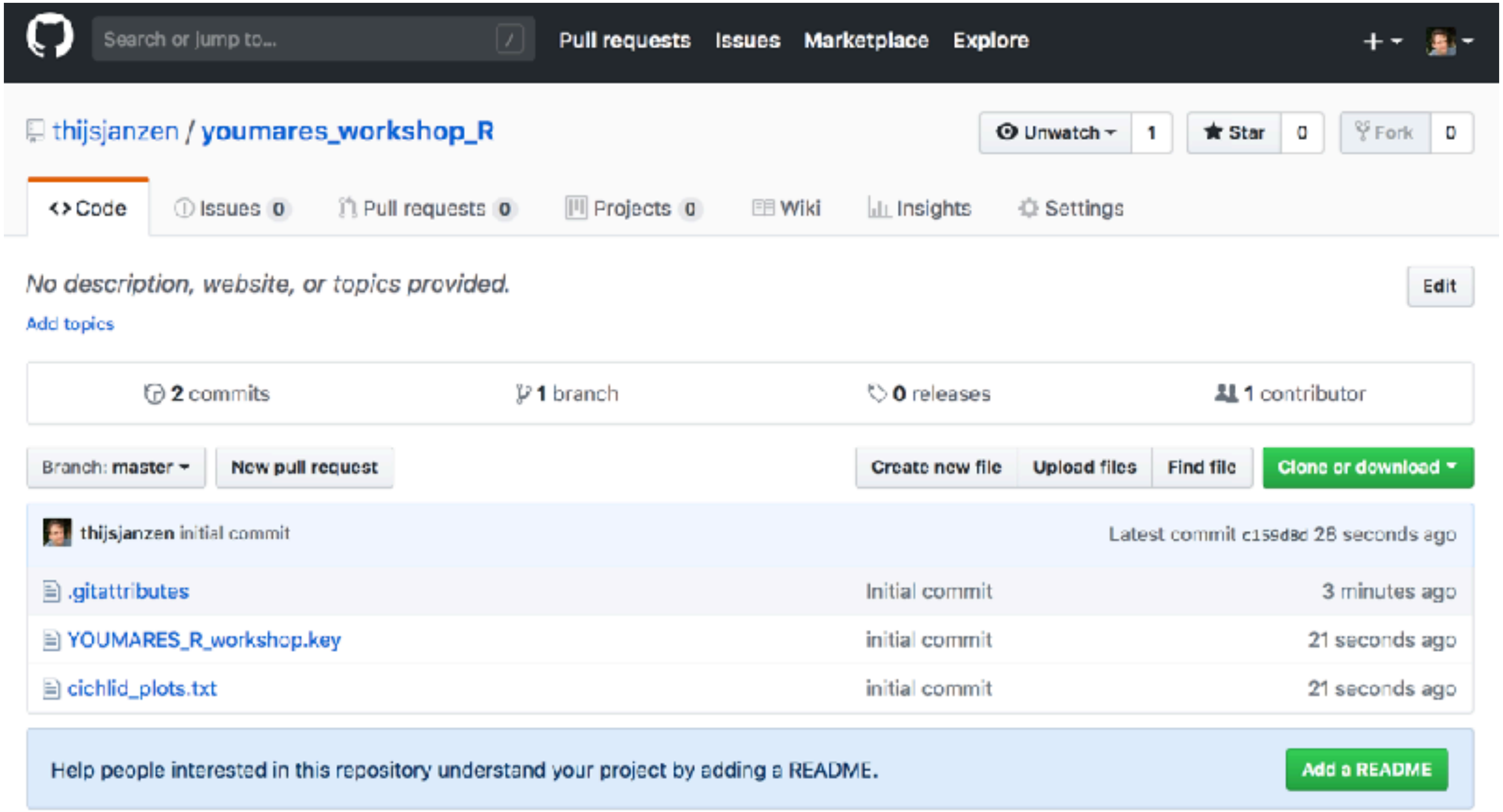
Requirements

- Rstudio (no strict requirement, but makes life easy)
- tidyverse packages:
  - tibble
  - readr
  - ggplot2
  - dplyr

```r
install.packages("tidyverse")
library(tidyverse)
```

Where to find files for this workshop

https://github.com/thijsjanzen/youmares_workshop_R



thijsjanzen / youmares_workshop_R

No description, website, or topics provided.

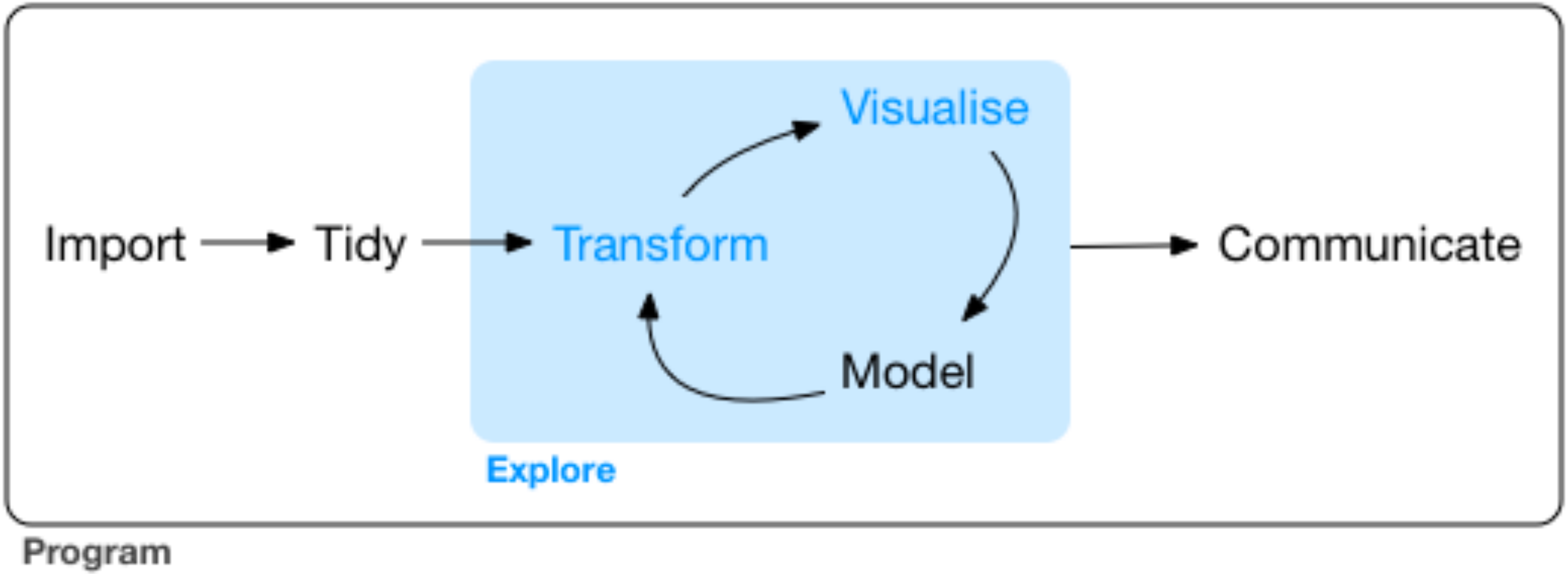
2 commits · 1 branch · 0 releases · 1 contributor

thijsjanzen initial commit — Latest commit c159d8d 28 seconds ago

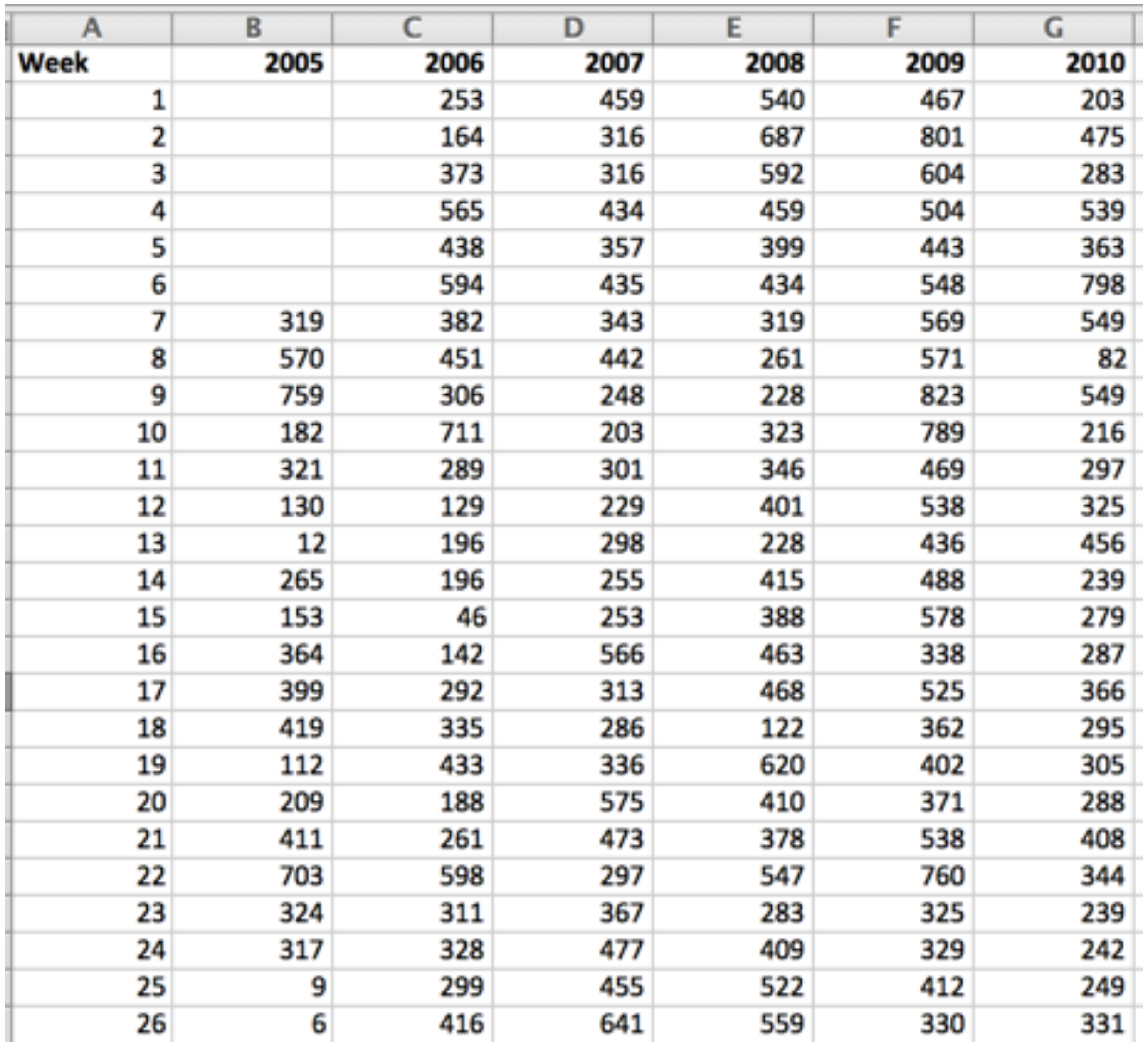
.gitattributes	Initial commit	3 minutes ago
YOUMARES_R_workshop.key	initial commit	21 seconds ago
cichlid_plots.txt	initial commit	21 seconds ago

Help people interested in this repository understand your project by adding a README.

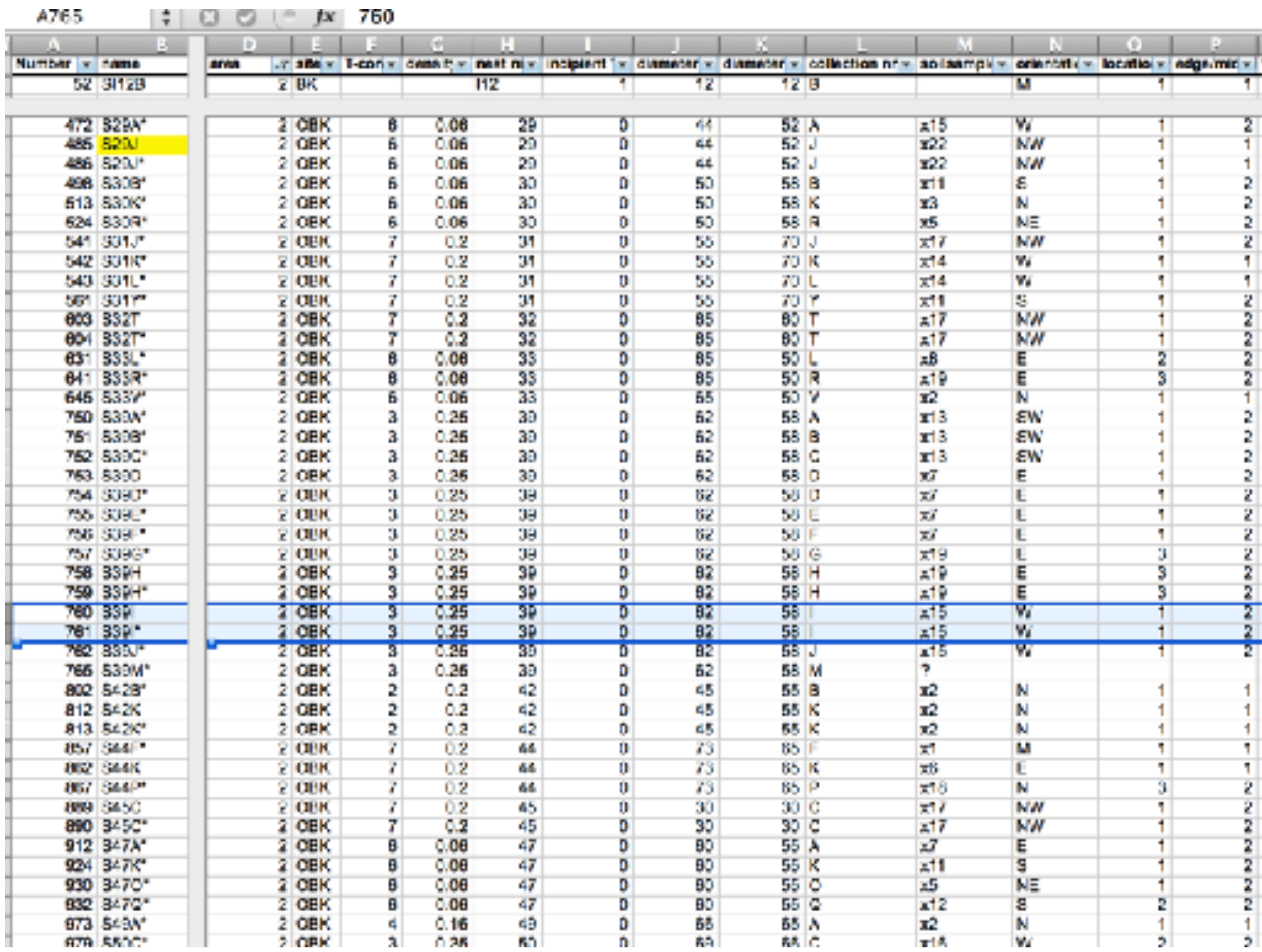
tidyverse


Import
Tidy
Transform
Visualise
Model
Communicate
Explore
Program

Data structure



A	B	C	D	E	F	G
Week	**2005**	**2006**	**2007**	**2008**	**2009**	**2010**
1		253	459	540	467	203
2		164	316	687	801	475
3		373	316	592	604	283
4		565	434	459	504	539
5		438	357	399	443	363
6		594	435	434	548	798
7	319	382	343	319	569	549
8	570	451	442	261	571	82
9	759	306	248	228	823	549
10	182	711	203	323	789	216
11	321	289	301	346	469	297
12	130	129	229	401	538	325
13	12	196	298	228	436	456
14	265	196	255	415	488	239
15	153	46	253	388	578	279
16	364	142	566	463	338	287
17	399	292	313	468	525	366
18	419	335	286	122	362	295
19	112	433	336	620	402	305
20	209	188	575	410	371	288
21	411	261	473	378	538	408
22	703	598	297	547	760	344
23	324	311	367	283	325	239
24	317	328	477	409	329	242
25	9	299	455	522	412	249
26	6	416	641	559	330	331



A765 *fx* 760

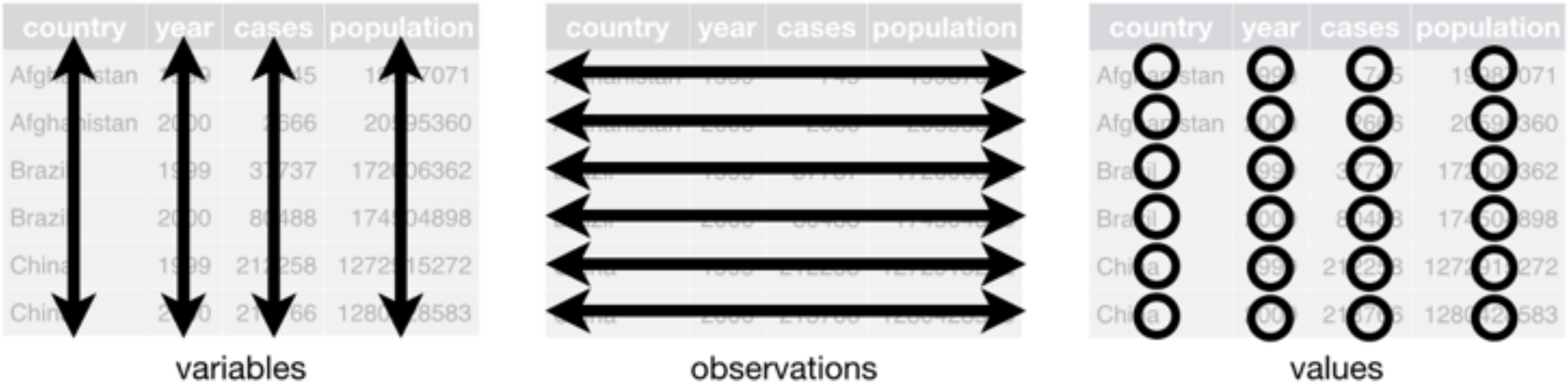
A	B	D	E	F	G	H	I	J	K	L	M	N	O	P
Number	name	area	site	T-con	densit	nest n	incipient	diameter	diameter	collection n	soil sampl	orientati	locatio	edge/mid
52	SI12B	2	BK			I12	1	12	12	B		M	1	1
472	S29A*	2	OBK	6	0.06	29	0	44	52	A	x15	W	1	2
485	S20J	2	OBK	6	0.06	20	0	44	52	J	x22	NW	1	1
486	S20J*	2	OBK	6	0.06	20	0	44	52	J	x22	NW	1	1
498	S30B*	2	OBK	6	0.06	30	0	50	58	B	x11	S	1	2
513	S30K*	2	OBK	6	0.06	30	0	50	58	K	x3	N	1	2
524	S30R*	2	OBK	6	0.06	30	0	50	58	R	x5	NE	1	2
541	S31J*	2	OBK	7	0.2	31	0	55	70	J	x17	NW	1	2
542	S31K*	2	OBK	7	0.2	31	0	55	70	K	x14	W	1	1
543	S31L*	2	OBK	7	0.2	31	0	55	70	L	x14	W	1	1
561	S31Y*	2	OBK	7	0.2	31	0	55	70	Y	x11	S	1	2
603	S32T	2	OBK	7	0.2	32	0	85	80	T	x17	NW	1	2
604	S32T*	2	OBK	7	0.2	32	0	85	80	T	x17	NW	1	2
631	S33L*	2	OBK	8	0.08	33	0	85	50	L	x8	E	2	2
641	S33R*	2	OBK	8	0.08	33	0	85	50	R	x19	E	3	2
645	S33V*	2	OBK	6	0.06	33	0	65	50	V	x2	N	1	1
750	S30A*	2	OBK	3	0.25	30	0	62	58	A	x13	SW	1	2
751	S30B*	2	OBK	3	0.25	30	0	62	58	B	x13	SW	1	2
752	S30C*	2	OBK	3	0.25	30	0	62	58	C	x13	SW	1	2
753	S30D	2	OBK	3	0.25	30	0	62	58	D	x7	E	1	2
754	S39D*	2	OBK	3	0.25	39	0	62	58	D	x7	E	1	2
755	S39E*	2	OBK	3	0.25	39	0	62	58	E	x7	E	1	2
756	S39F*	2	OBK	3	0.25	39	0	62	58	F	x7	E	1	2
757	S39G*	2	OBK	3	0.25	39	0	62	58	G	x19	E	3	2
758	S39H	2	OBK	3	0.25	39	0	82	58	H	x19	E	3	2
759	S39H*	2	OBK	3	0.25	39	0	82	58	H	x19	E	3	2
760	S39I	2	OBK	3	0.25	39	0	82	58	I	x15	W	1	2
761	S39I*	2	OBK	3	0.25	39	0	82	58	I	x15	W	1	2
762	S39J*	2	OBK	3	0.25	39	0	82	58	J	x15	W	1	2
765	S30M*	2	OBK	3	0.25	30	0	62	58	M	?			
802	S42B*	2	OBK	2	0.2	42	0	45	55	B	x2	N	1	1
812	S42K	2	OBK	2	0.2	42	0	45	55	K	x2	N	1	1
813	S42K*	2	OBK	2	0.2	42	0	45	55	K	x2	N	1	1
857	S44F*	2	OBK	7	0.2	44	0	73	65	F	x1	M	1	1
862	S44K	2	OBK	7	0.2	44	0	73	65	K	x6	E	1	1
867	S44P*	2	OBK	7	0.2	44	0	73	65	P	x16	N	3	2
889	S45C	2	OBK	7	0.2	45	0	30	30	C	x17	NW	1	2
890	S45C*	2	OBK	7	0.2	45	0	30	30	C	x17	NW	1	2
912	S47A*	2	OBK	8	0.08	47	0	80	55	A	x7	E	1	2
924	S47K*	2	OBK	8	0.08	47	0	80	55	K	x11	S	1	2
930	S47O*	2	OBK	8	0.08	47	0	80	55	O	x5	NE	1	2
932	S47Q*	2	OBK	8	0.08	47	0	80	55	Q	x12	S	2	2
973	S49A*	2	OBK	4	0.16	49	0	68	65	A	x2	N	1	1
979	S50C*	2	OBK	3	0.25	50	0	69	65	C	x16	W	2	2

Tidy data

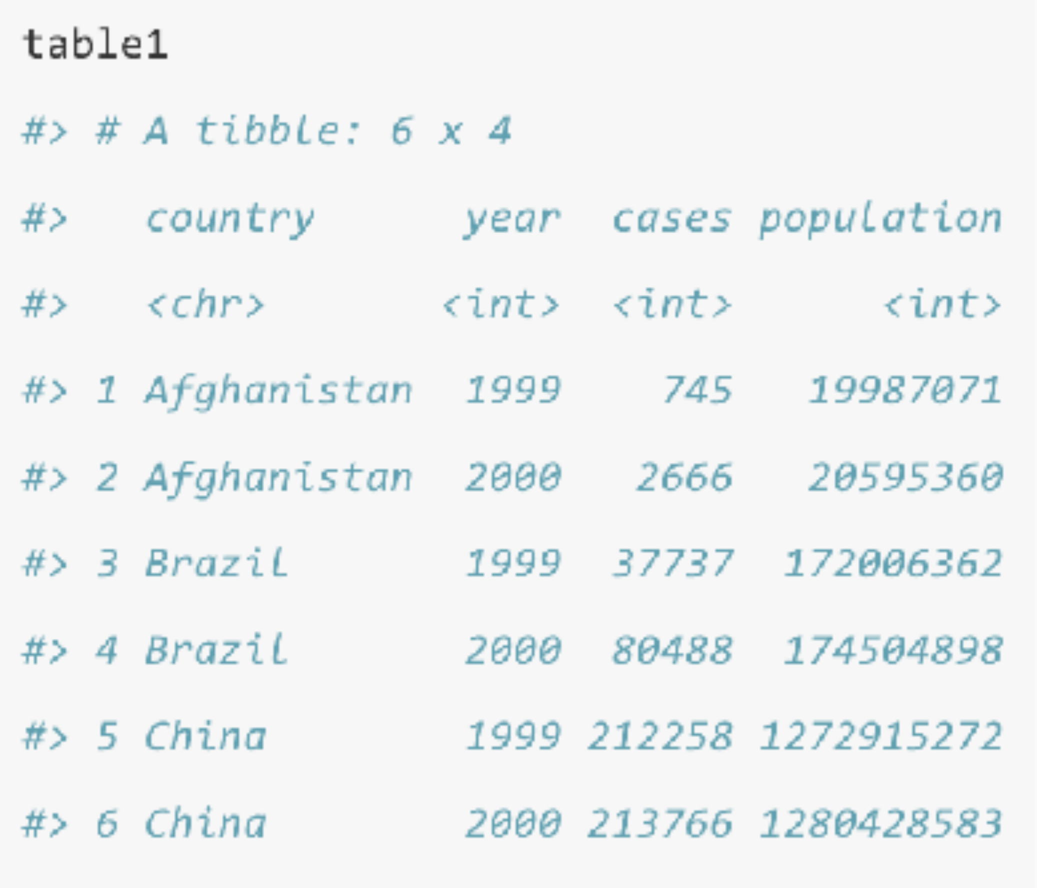
- Each variable has it’s own column

- Each observation has it’s own row

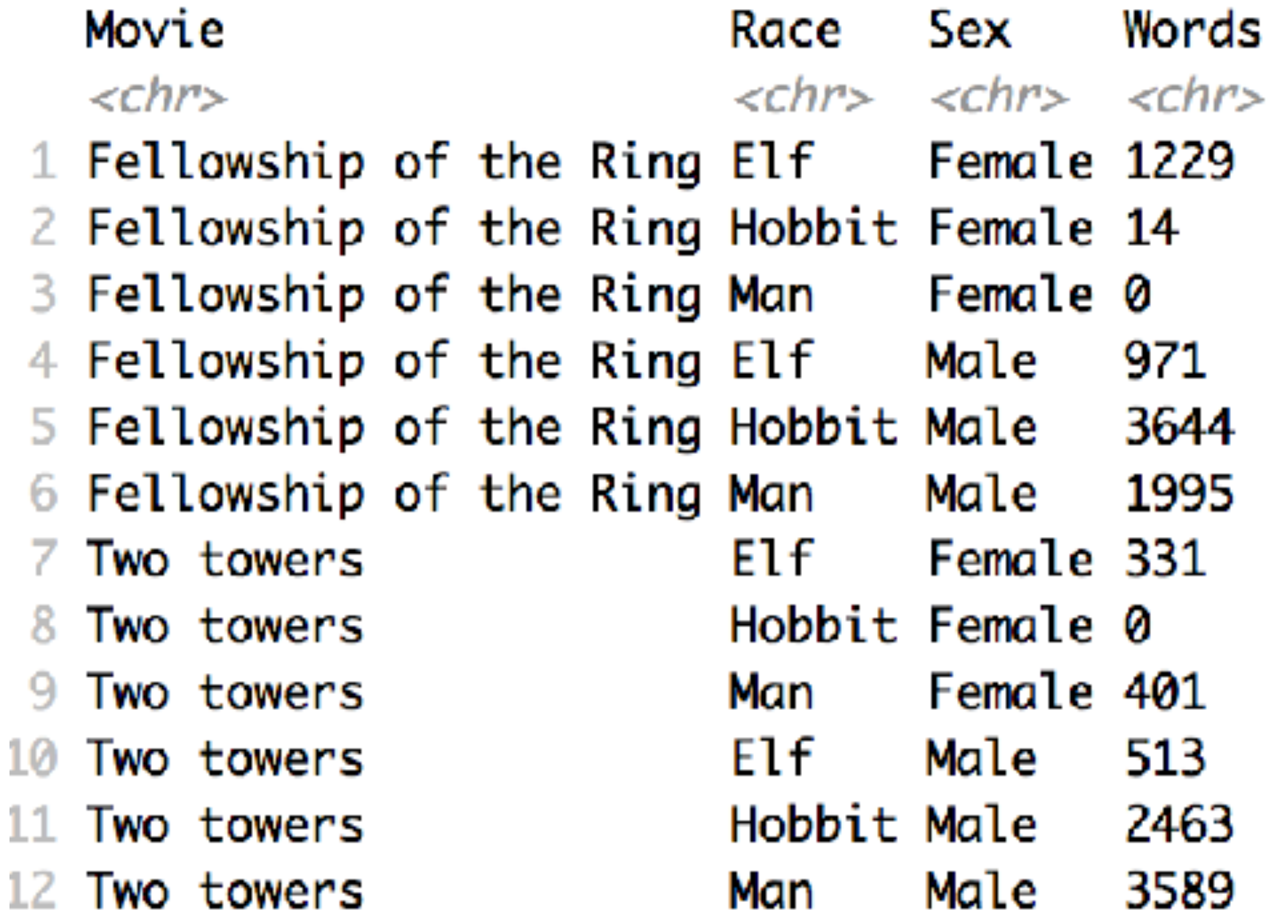
- Each value has it’s own cell



Examples tidy data



```
table1
#> # A tibble: 6 x 4
#>   country      year  cases population
#>   <chr>       <int>  <int>      <int>
#> 1 Afghanistan  1999    745   19987071
#> 2 Afghanistan  2000   2666   20595360
#> 3 Brazil       1999  37737  172006362
#> 4 Brazil       2000  80488  174504898
#> 5 China        1999 212258 1272915272
#> 6 China        2000 213766 1280428583
```



```
   Movie                      Race   Sex    Words
   <chr>                      <chr>  <chr>  <chr>
 1 Fellowship of the Ring Elf    Female 1229
 2 Fellowship of the Ring Hobbit Female 14
 3 Fellowship of the Ring Man    Female 0
 4 Fellowship of the Ring Elf    Male   971
 5 Fellowship of the Ring Hobbit Male   3644
 6 Fellowship of the Ring Man    Male   1995
 7 Two towers             Elf    Female 331
 8 Two towers             Hobbit Female 0
 9 Two towers             Man    Female 401
10 Two towers             Elf    Male   513
11 Two towers             Hobbit Male   2463
12 Two towers             Man    Male   3589
```

But… my data is not tidy?

- When recording data, your data is often not tidy
- There are two functions (amongst others) to help you make your data tidy:
    - gather
    - spread

Importing data into R

`read_tsv, read_csv, read_delim`

```
read_tsv(file, col_names = TRUE)

read_csv(file, col_names = TRUE)

read_delim(file, delim, col_names = TRUE)
```

Reading data into R

```
fish_counts <- read_tsv(file = “cichlid_plots.txt”)

fish_counts
```

```
lotr_words <- read_tsv(file = “lotr_words.txt”)

lotr_words
```

Tidying data: gather

- The function `gather` combines multiple columns into one column, and adds an extra indicator column.

- gather(data, key, value, columns)

  - `data` = the data to be converted

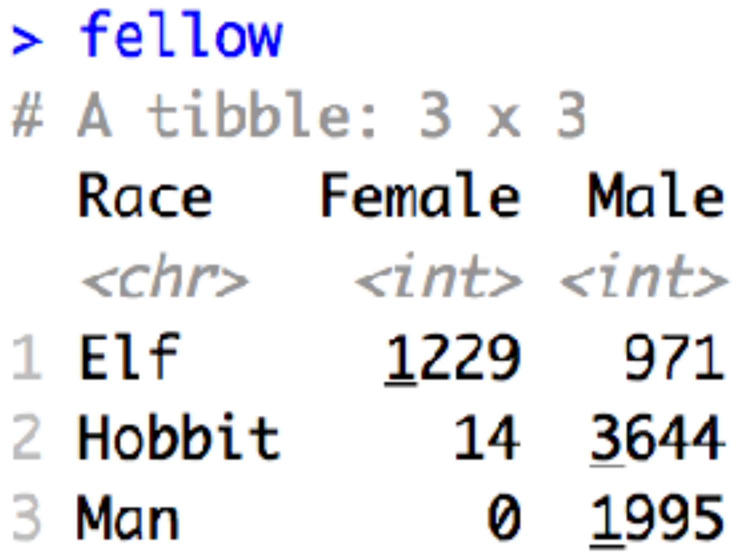
  - `key` = variable name that is going to contain the column name

  - `value` = variable name that is going to contain the gathered data

  - `columns` = selection of which columns need to be gathered

Example gather

```
fellow <- read_tsv("fellowship.txt")
```

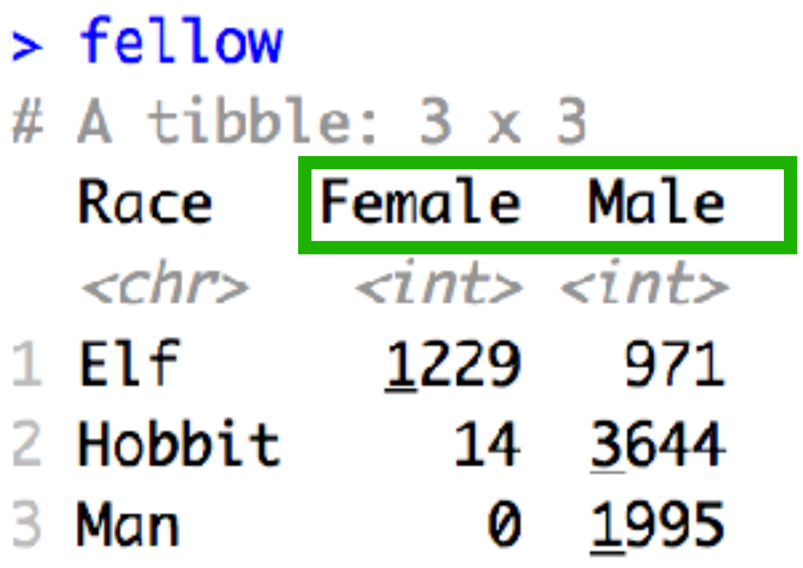


```
> fellow
# A tibble: 3 x 3
  Race   Female  Male
  <chr>   <int> <int>
1 Elf      1229   971
2 Hobbit     14  3644
3 Man         0  1995
```

```
fellow_gathered <- gather(data    = fellow,
                          key     =
                          value   =
                          columns =
```

Example gather

```
fellow <- read_tsv(“fellowship.txt”)
```



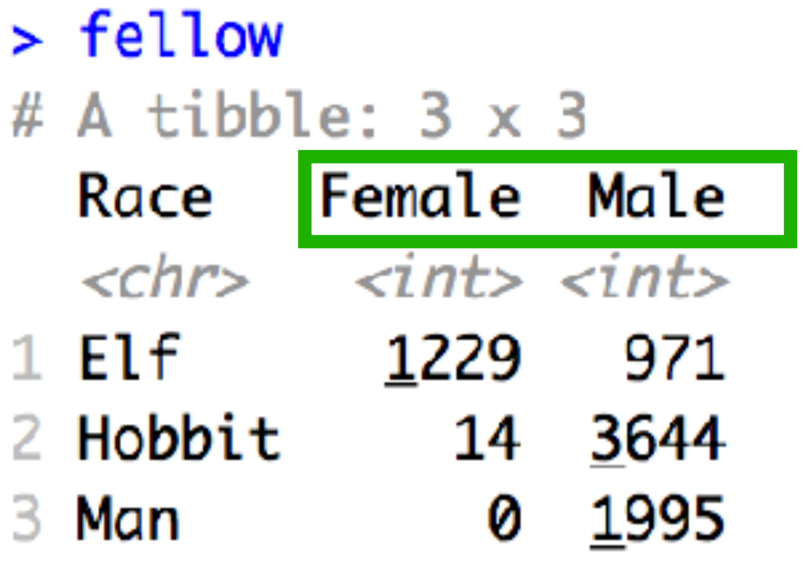
```
> fellow
# A tibble: 3 x 3
  Race    Female  Male
  <chr>    <int> <int>
1 Elf       1229   971
2 Hobbit      14  3644
3 Man          0  1995
```

key

```
fellow_gathered <- gather(data    = fellow,
                          key     =
                          value   =
                          columns =
```

Example gather

```
fellow <- read_tsv(“fellowship.txt”)
```



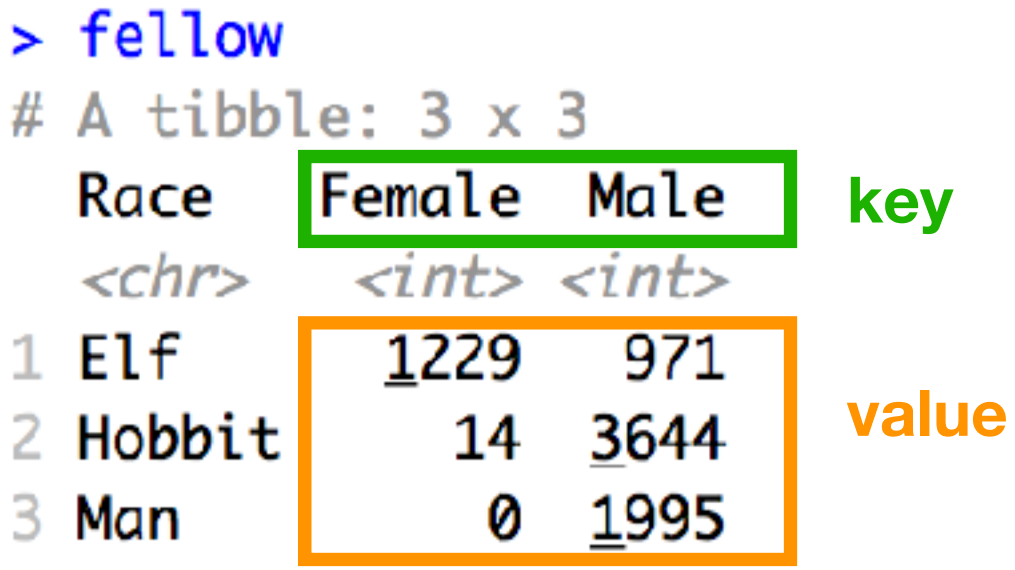
```
> fellow
# A tibble: 3 x 3
  Race    Female  Male
  <chr>    <int> <int>
1 Elf       1229   971
2 Hobbit      14  3644
3 Man          0  1995
```

key

```
fellow_gathered <- gather(data    = fellow,
                          key     = "Sex",
                          value   =
                          columns =
```

Example gather

```
fellow <- read_tsv(“fellowship.txt”)
```



```
> fellow
# A tibble: 3 x 3
  Race   Female  Male
  <chr>   <int> <int>
1 Elf      1229   971
2 Hobbit     14  3644
3 Man         0  1995
```

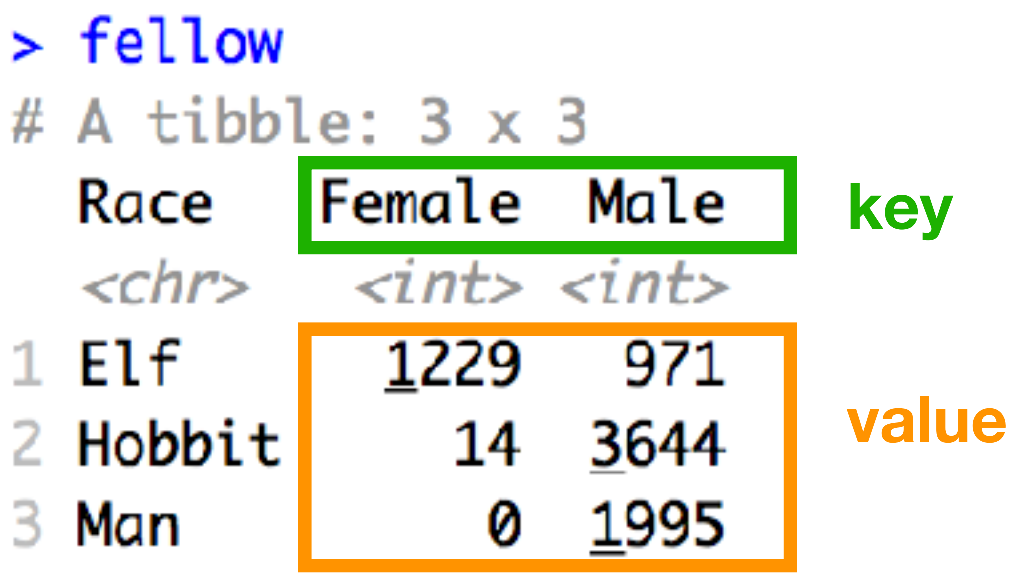
key

value

```
fellow_gathered <- gather(data    = fellow,
                          key     = "Sex",
                          value   =
                          columns =
```

Example gather

```
fellow <- read_tsv(“fellowship.txt”)
```

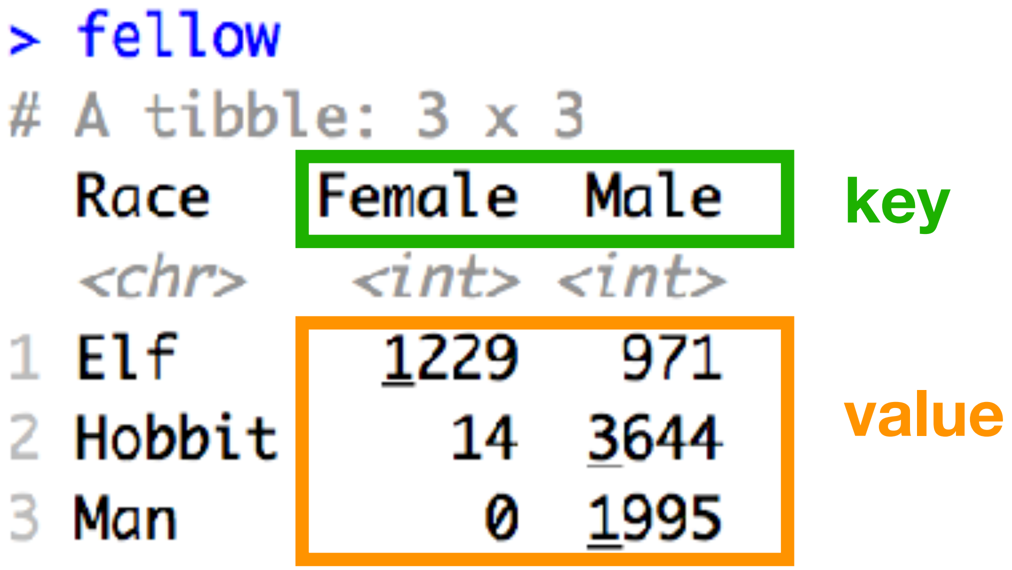


```
> fellow
# A tibble: 3 x 3
  Race   Female  Male
  <chr>   <int> <int>
1 Elf      1229   971
2 Hobbit     14  3644
3 Man         0  1995
```

```
fellow_gathered <- gather(data    = fellow,
                          key     = "Sex",
                          value   = "Words",
                          columns =
```

Example gather

```
fellow <- read_tsv(“fellowship.txt”)
```

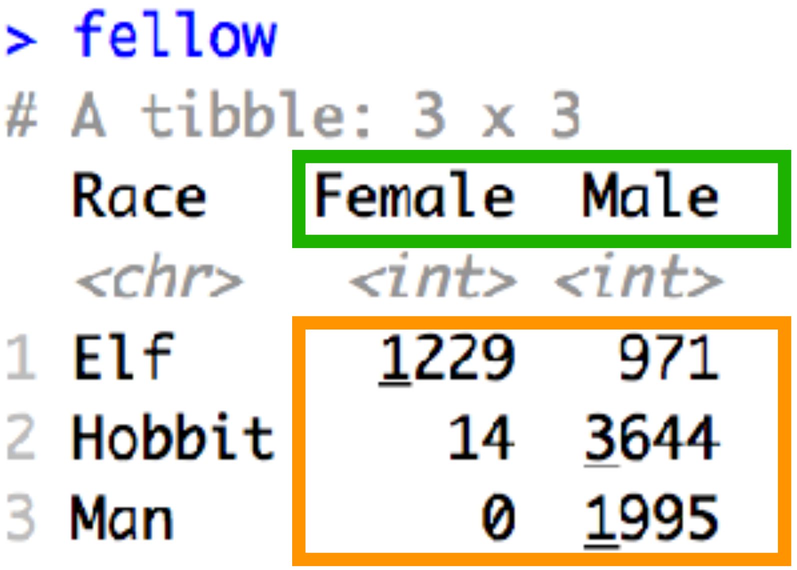


```
> fellow
# A tibble: 3 x 3
  Race   Female  Male
  <chr>   <int> <int>
1 Elf      1229   971
2 Hobbit     14  3644
3 Man         0  1995
```

```
fellow_gathered <- gather(data    = fellow,
                          key     = "Sex",
                          value   = "Words",
                          columns = c("Male","Female"))
```

Example gather

```
fellow <- read_tsv(“fellowship.txt”)
```

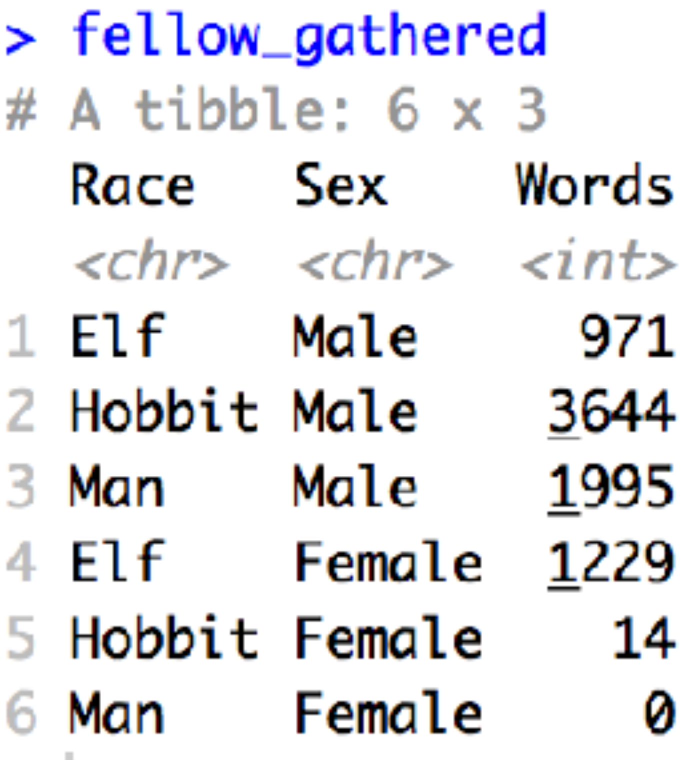


```
> fellow
# A tibble: 3 x 3
  Race   Female  Male
  <chr>   <int> <int>
1 Elf      1229   971
2 Hobbit     14  3644
3 Man         0  1995
```

key

value

```
fellow_gathered <- gather(data    = fellow,
                          key     = "Sex",
                          value   = "Words",
                          columns = c("Male","Female"))
```



```
> fellow_gathered
# A tibble: 6 x 3
  Race   Sex    Words
  <chr>  <chr>  <int>
1 Elf    Male     971
2 Hobbit Male    3644
3 Man    Male    1995
4 Elf    Female  1229
5 Hobbit Female    14
6 Man    Female     0
```

How to indicate the columns?

- Use the names:

```
gather(fellow, key = “Sex”, value = “Words”, “Female”, “Male”)
```

- Use the index:

```
gather(fellow, key = “Sex”, value = “Words”, 2:3)
```

- Use all columns (except the first):

```
gather(fellow, key = “Sex”, value = “Words”, -1)
```

Plotting data

- OK, we have tidy data now
- How to visualise results?

ggplot

- ggplot: the Grammar of Graphics
- Plots are constructed out of building blocks:
    - data
    - aesthetic mapping
    - geometric object
    - statistical transformations
    - scales
    - coordinate systems
    - labels

https://tutorials.iq.harvard.edu/R/Rgraphics/Rgraphics.html

ggplot

```
ggplot(data, aes(x = .. , y = ..) ) +

    geom_     +

    stat_     +

    xlab()    +
```

aesthetics: indicate what is on the x axis, on the y-axis,
and if you need grouping of your data

geom_point / geom_line / geom_bar etc. : indicates the type of plot
(scatter, line, barplot, box plot etc)

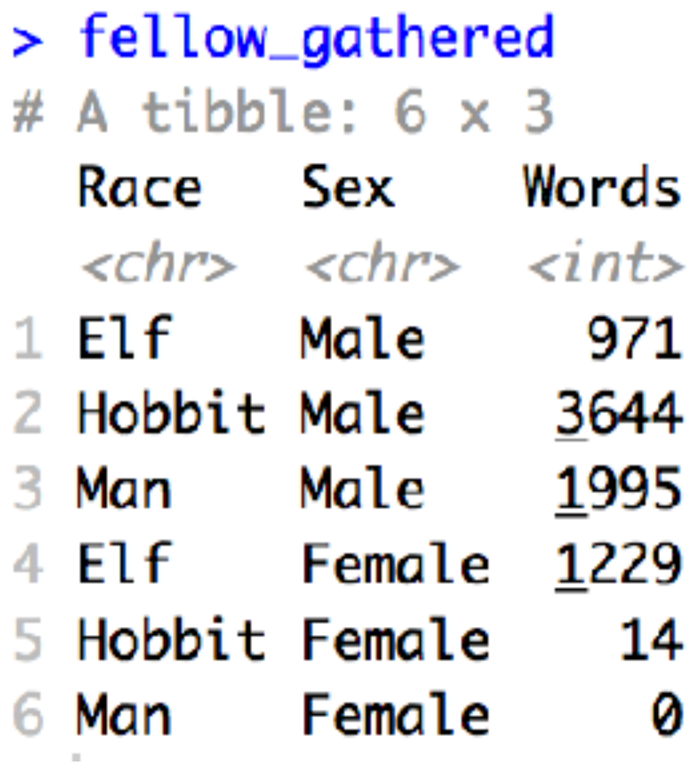
stat_smooth() : indicates additional statistics

plotting lotr

Let’s create a bar plot, split per race and sex

```
ggplot(data = fellow_gathered, aes(x = Race, y = Words, fill = Sex)) +
       geom_bar(stat = "identity", position = "dodge")
```

plotting lotr



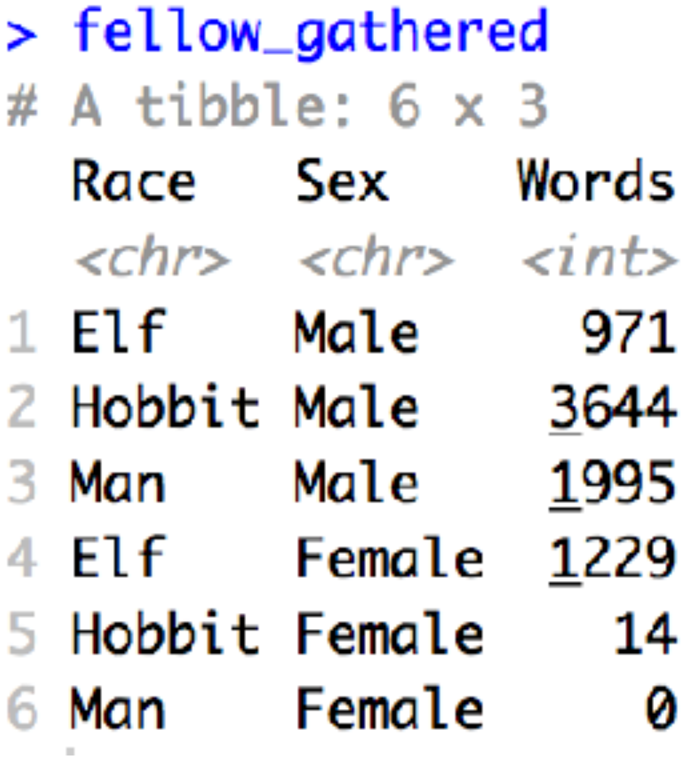
```
> fellow_gathered
# A tibble: 6 x 3
  Race   Sex    Words
  <chr>  <chr>  <int>
1 Elf    Male     971
2 Hobbit Male    3644
3 Man    Male    1995
4 Elf    Female  1229
5 Hobbit Female    14
6 Man    Female     0
```

Let’s create a bar plot, split per race and sex

```
ggplot(data = fellow_gathered, aes(x = Race, y = Words, fill = Sex)) +
      geom_bar(stat = "identity", position = "dodge")
```

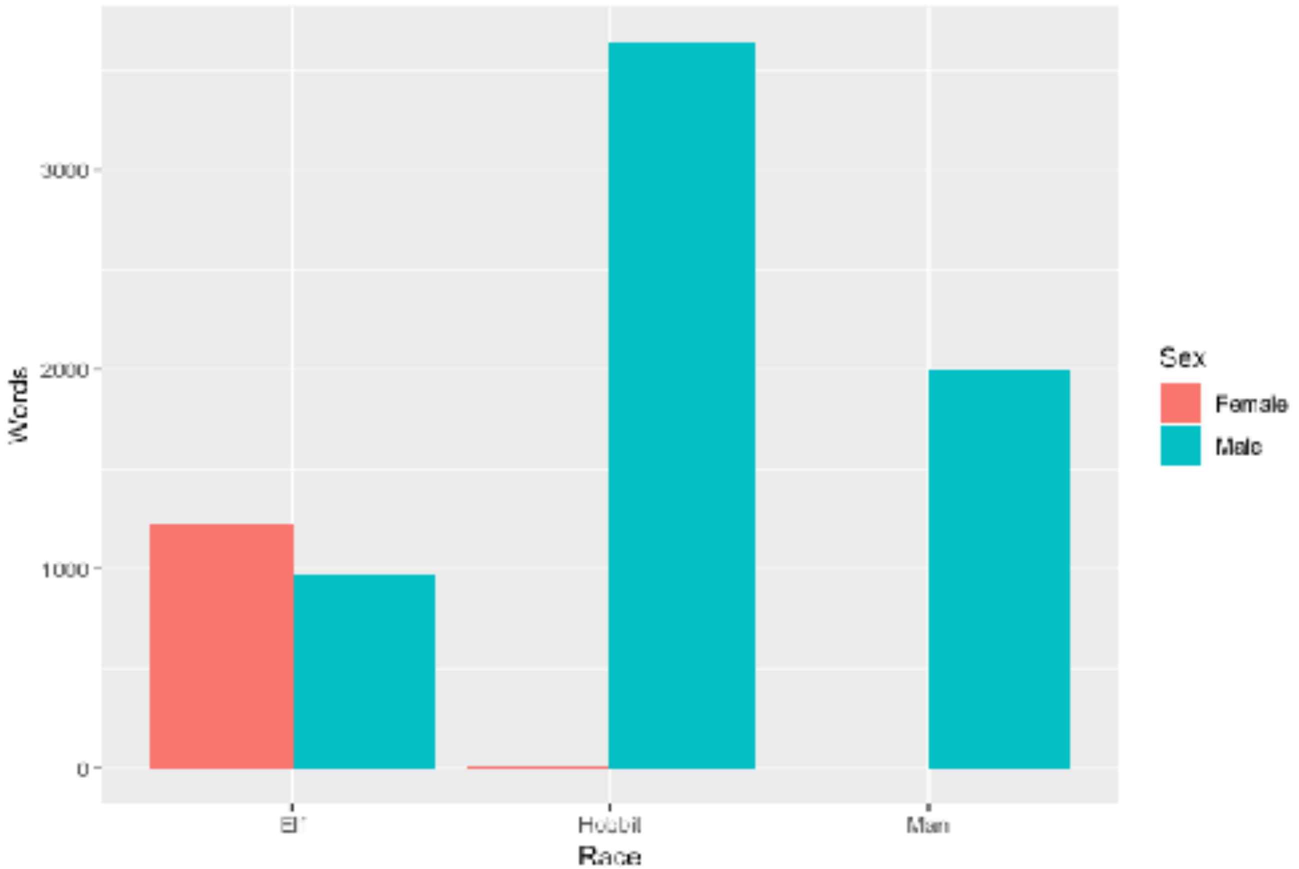
plotting lotr

Let’s create a bar plot, split per race and sex



```
> fellow_gathered
# A tibble: 6 x 3
  Race   Sex    Words
  <chr>  <chr>  <int>
1 Elf    Male     971
2 Hobbit Male    3644
3 Man    Male    1995
4 Elf    Female  1229
5 Hobbit Female    14
6 Man    Female     0
```

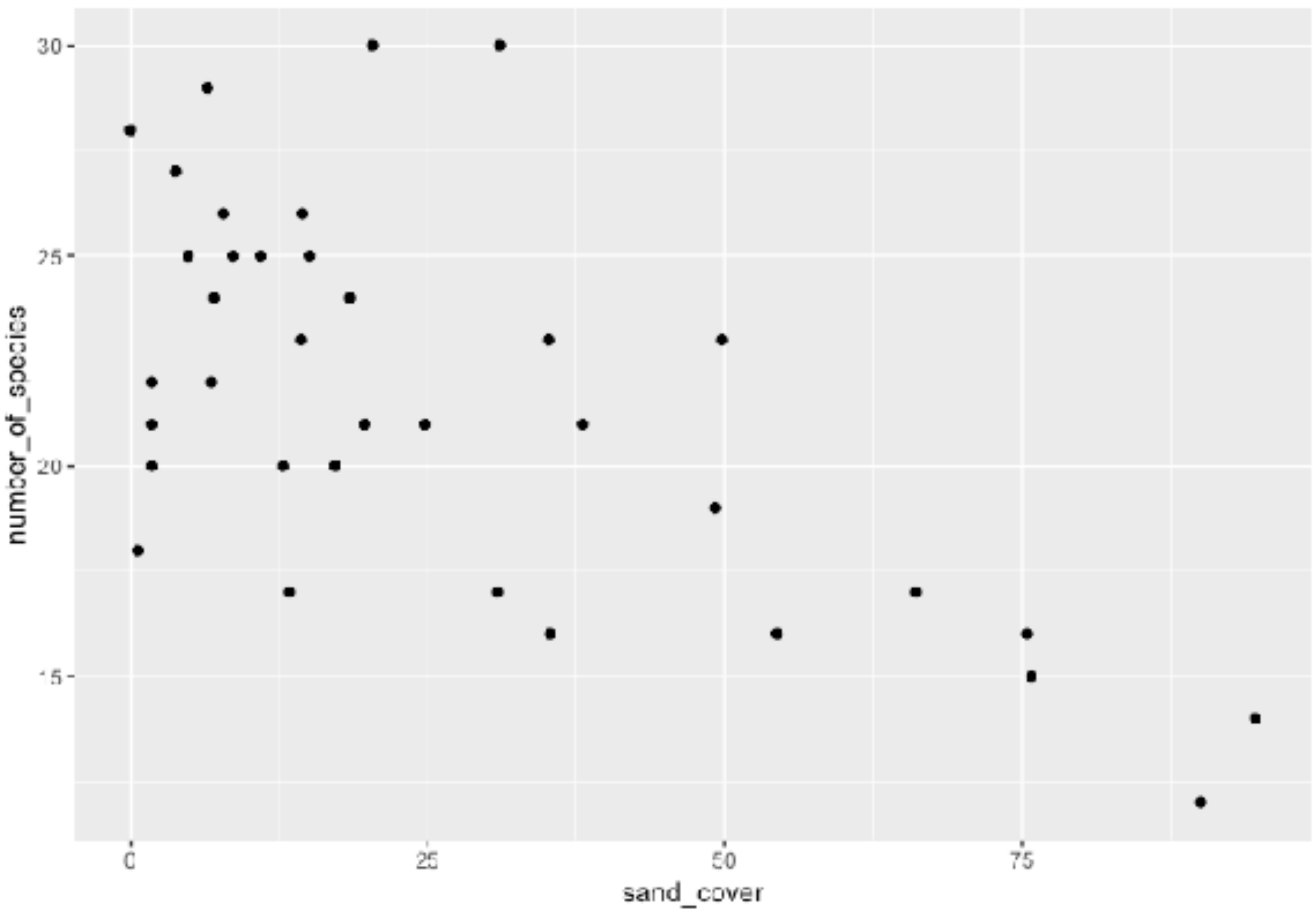
```
ggplot(data = fellow_gathered, aes(x = Race, y = Words, fill = Sex)) +
       geom_bar(stat = "identity", position = "dodge")
```



ggplot

-
  ```
  ggplot(fish_counts, aes(x = sand_cover, y = number_of_species)) +
          geom_point()
  ```

  - aesthetic mapping
  - geometric object



Examples gather

The fish counts dataset contains one column for every species, but rather, we would like to have one column indicating species, and one column indicating the observed number.

How can we do that with gather?

```
fish_counts2 <- gather(data    = fish_counts,
                       key     =
                       value   =
                       columns =
```

Examples gather

The fish counts dataset contains one column for every species, but rather, we would like to have one column indicating species, and one column indicating the observed number.

How can we do that with gather?

```
fish_counts2 <- gather(data    = fish_counts,
                       key     = “Species”,
                       value   =
                       columns =
```

Examples gather

The fish counts dataset contains one column for every species, but rather, we would like to have one column indicating species, and one column indicating the observed number.

How can we do that with gather?

```
fish_counts2 <- gather(data    = fish_counts,
                       key     = “Species”,
                       value   = “Count”,
                       columns =
```

Examples gather

The fish counts dataset contains one column for every species, but rather, we would like to have one column indicating species, and one column indicating the observed number.

How can we do that with gather?

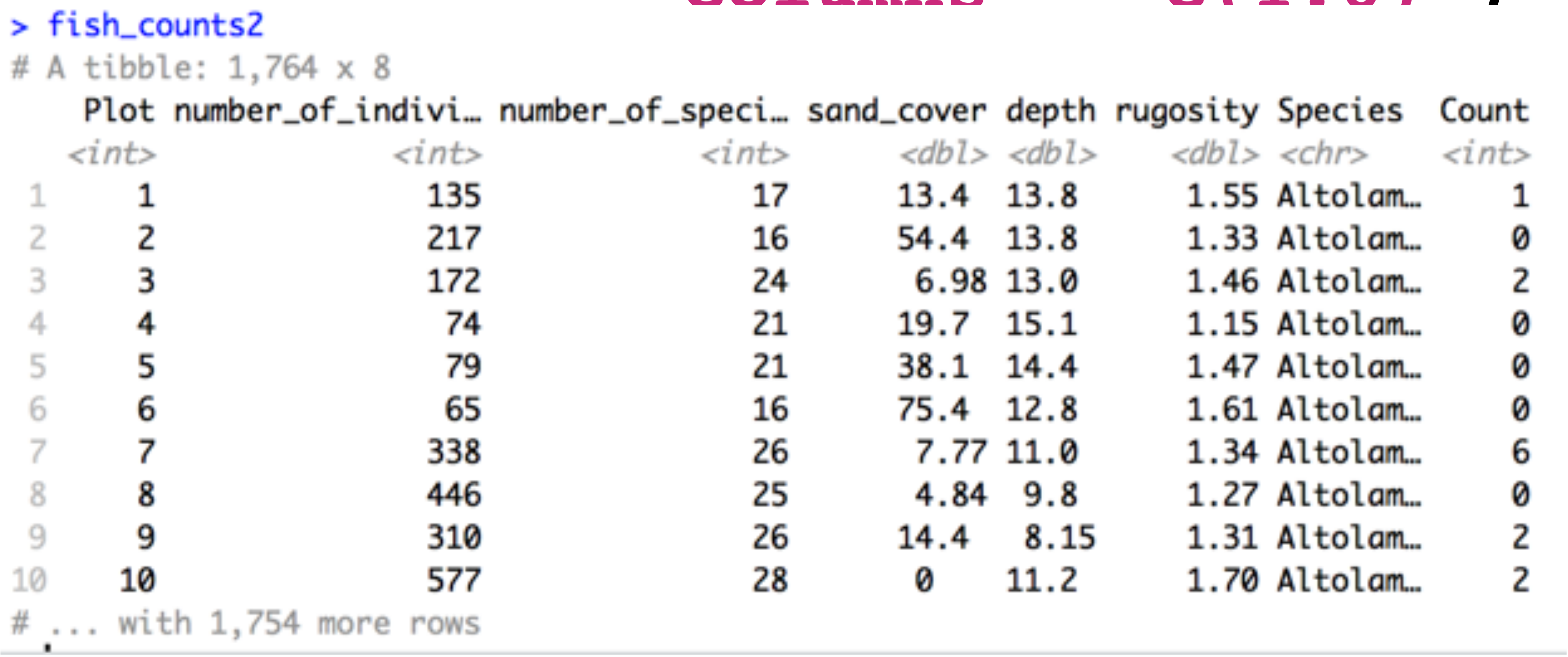
```
fish_counts2 <- gather(data    = fish_counts,
                       key     = “Species”,
                       value   = “Count”,
                       columns = -c(1:6) )
```

Examples gather

The fish counts dataset contains one column for every species, but rather, we would like to have one column indicating species, and one column indicating the observed number.

How can we do that with gather?

```
fish_counts2 <- gather(data    = fish_counts,
                       key     = “Species”,
                       value   = “Count”,
                       columns = -c(1:6) )
```

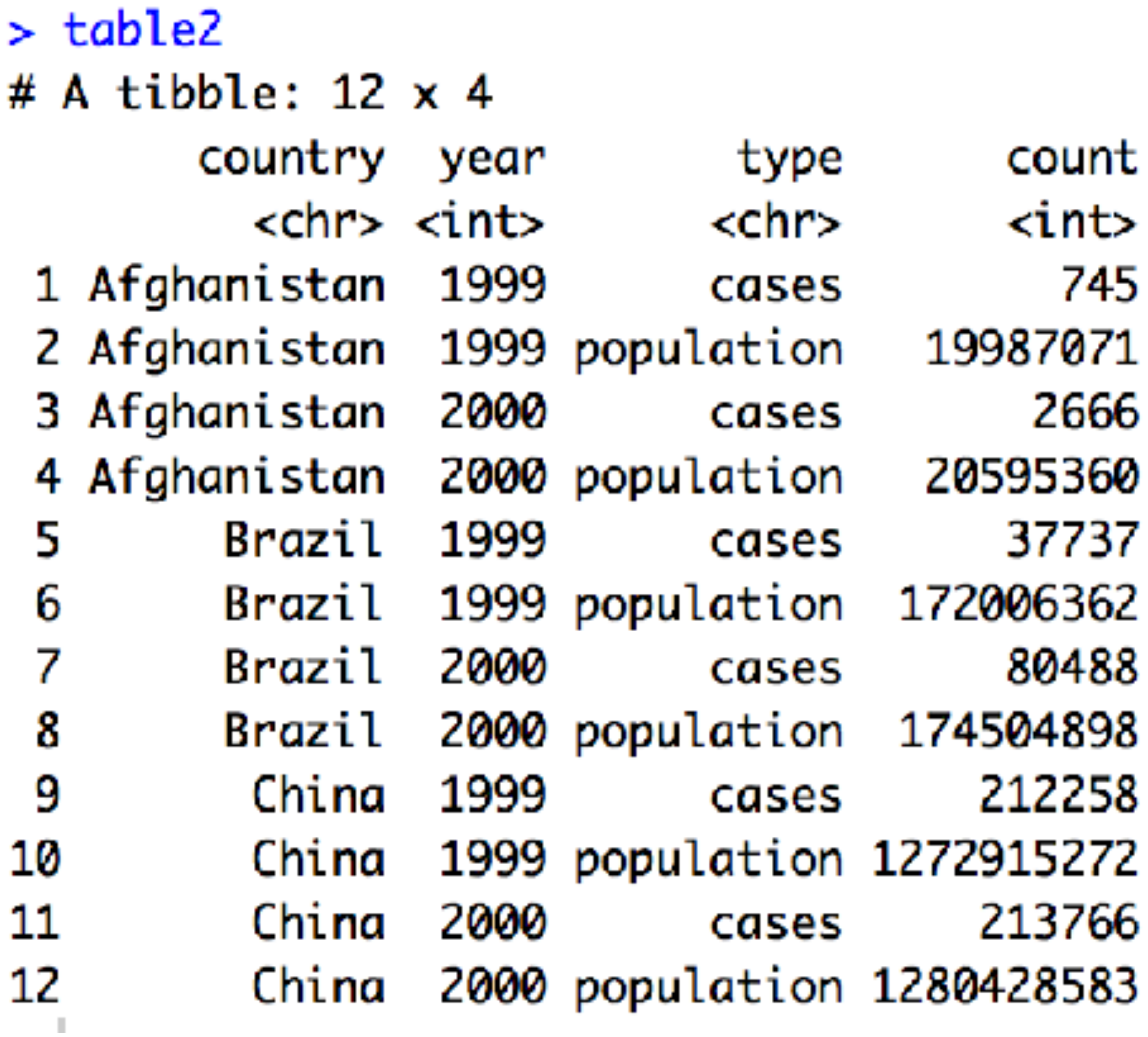


```
> fish_counts2
# A tibble: 1,764 x 8
    Plot number_of_indivi… number_of_speci… sand_cover depth rugosity Species   Count
   <int>             <int>            <int>      <dbl> <dbl>    <dbl> <chr>     <int>
 1     1               135               17      13.4  13.8      1.55 Altolam…      1
 2     2               217               16      54.4  13.8      1.33 Altolam…      0
 3     3               172               24       6.98 13.0      1.46 Altolam…      2
 4     4                74               21      19.7  15.1      1.15 Altolam…      0
 5     5                79               21      38.1  14.4      1.47 Altolam…      0
 6     6                65               16      75.4  12.8      1.61 Altolam…      0
 7     7               338               26       7.77 11.0      1.34 Altolam…      6
 8     8               446               25       4.84  9.8      1.27 Altolam…      0
 9     9               310               26      14.4   8.15     1.31 Altolam…      2
10    10               577               28       0    11.2      1.70 Altolam…      2
# ... with 1,754 more rows
```

Spreading

- Collecting observations that are scattered among multiple rows

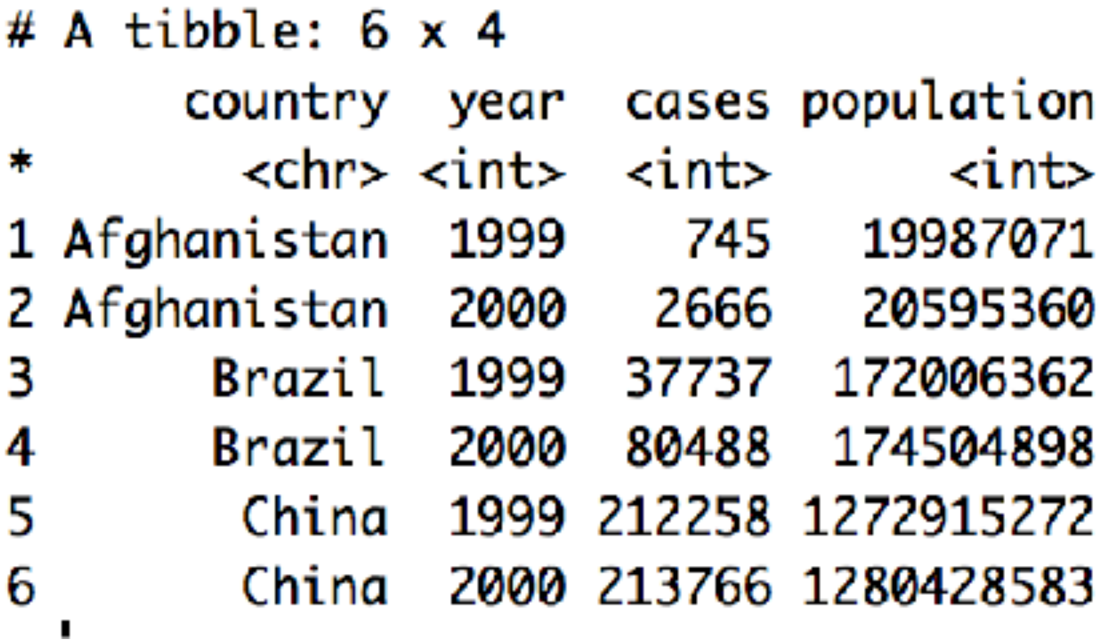
- Spreading is the exact opposite of gather



```
> table2
# A tibble: 12 x 4
       country  year       type      count
         <chr> <int>      <chr>      <int>
 1 Afghanistan  1999      cases        745
 2 Afghanistan  1999 population   19987071
 3 Afghanistan  2000      cases       2666
 4 Afghanistan  2000 population   20595360
 5      Brazil  1999      cases      37737
 6      Brazil  1999 population  172006362
 7      Brazil  2000      cases      80488
 8      Brazil  2000 population  174504898
 9       China  1999      cases     212258
10       China  1999 population 1272915272
11       China  2000      cases     213766
12       China  2000 population 1280428583
```

- `spread(data, key, value)`
  - `data` = the data to be converted
  - `key` = variable name that needs to be spread
  - `value` = column of corresponding data values

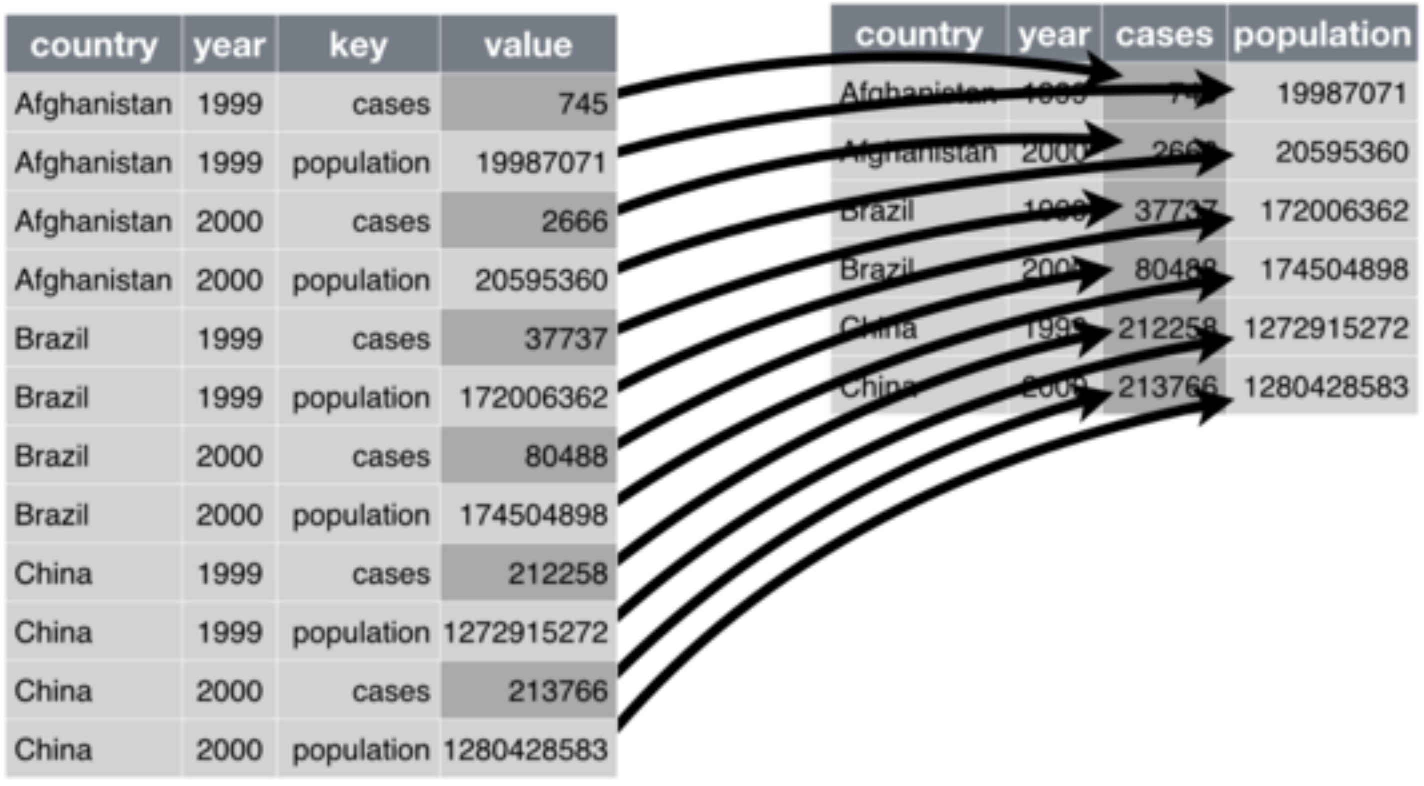
```
table2_b <- spread(table2, key = "type", value = "count")
```



```
# A tibble: 6 x 4
      country  year  cases population
*       <chr> <int>  <int>      <int>
1 Afghanistan  1999    745   19987071
2 Afghanistan  2000   2666   20595360
3      Brazil  1999  37737  172006362
4      Brazil  2000  80488  174504898
5       China  1999 212258 1272915272
6       China  2000 213766 1280428583
```

Gather

Spread

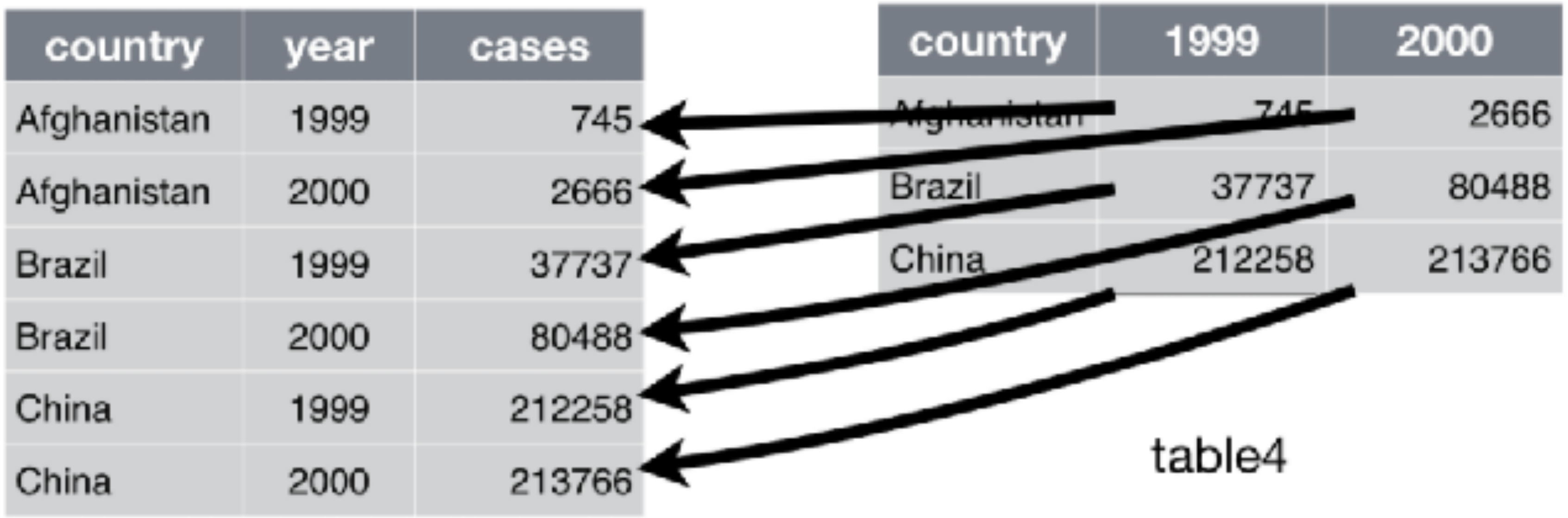


country	year	key	value
Afghanistan	1999	cases	745
Afghanistan	1999	population	19987071
Afghanistan	2000	cases	2666
Afghanistan	2000	population	20595360
Brazil	1999	cases	37737
Brazil	1999	population	172006362
Brazil	2000	cases	80488
Brazil	2000	population	174504898
China	1999	cases	212258
China	1999	population	1272915272
China	2000	cases	213766
China	2000	population	1280428583

table2

country	year	cases	population
Afghanistan	1999	745	19987071
Afghanistan	2000	2666	20595360
Brazil	1999	37737	172006362
Brazil	2000	80488	174504898
China	1999	212258	1272915272
China	2000	213766	1280428583

Gather

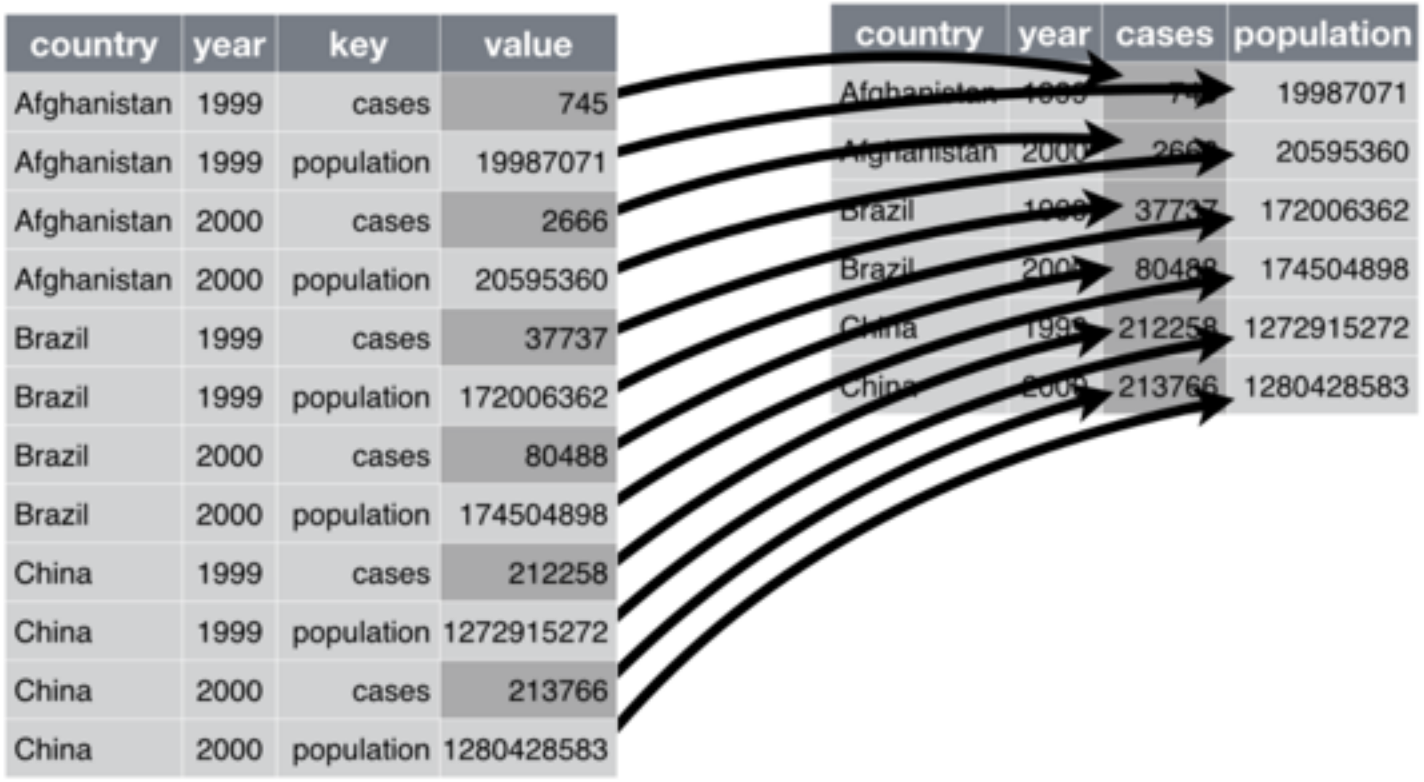


country	year	cases
Afghanistan	1999	745
Afghanistan	2000	2666
Brazil	1999	37737
Brazil	2000	80488
China	1999	212258
China	2000	213766

country	1999	2000
Afghanistan	745	2666
Brazil	37737	80488
China	212258	213766

table4

Spread



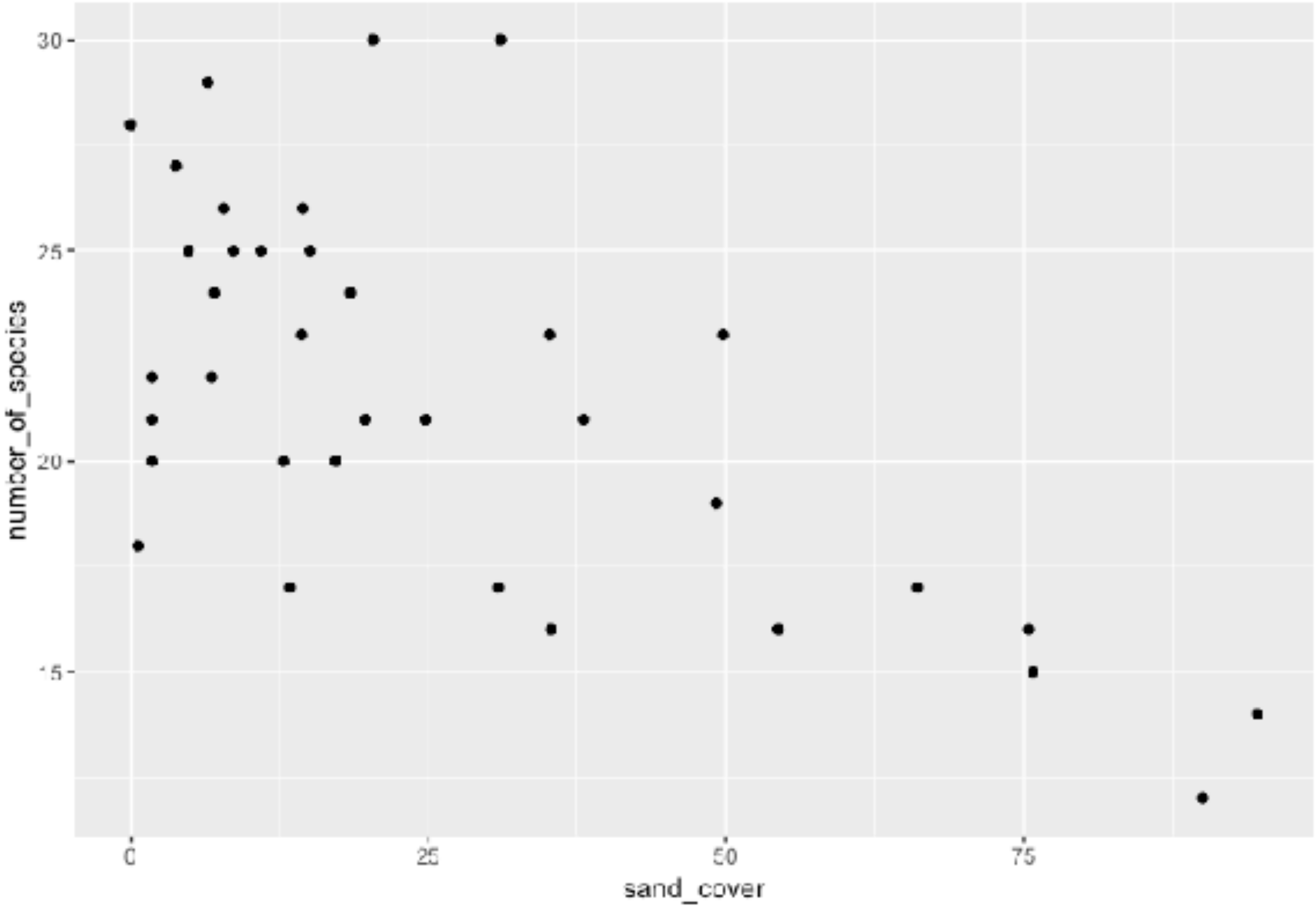
country	year	key	value
Afghanistan	1999	cases	745
Afghanistan	1999	population	19987071
Afghanistan	2000	cases	2666
Afghanistan	2000	population	20595360
Brazil	1999	cases	37737
Brazil	1999	population	172006362
Brazil	2000	cases	80488
Brazil	2000	population	174504898
China	1999	cases	212258
China	1999	population	1272915272
China	2000	cases	213766
China	2000	population	1280428583

table2

country	year	cases	population
Afghanistan	1999	745	19987071
Afghanistan	2000	2666	20595360
Brazil	1999	37737	172006362
Brazil	2000	80488	174504898
China	1999	212258	1272915272
China	2000	213766	1280428583

ggplot

-
  ```
  ggplot(fish_counts, aes(x = sand_cover, y = number_of_species)) +
          geom_point()
  ```
  - aesthetic mapping
  - geometric object



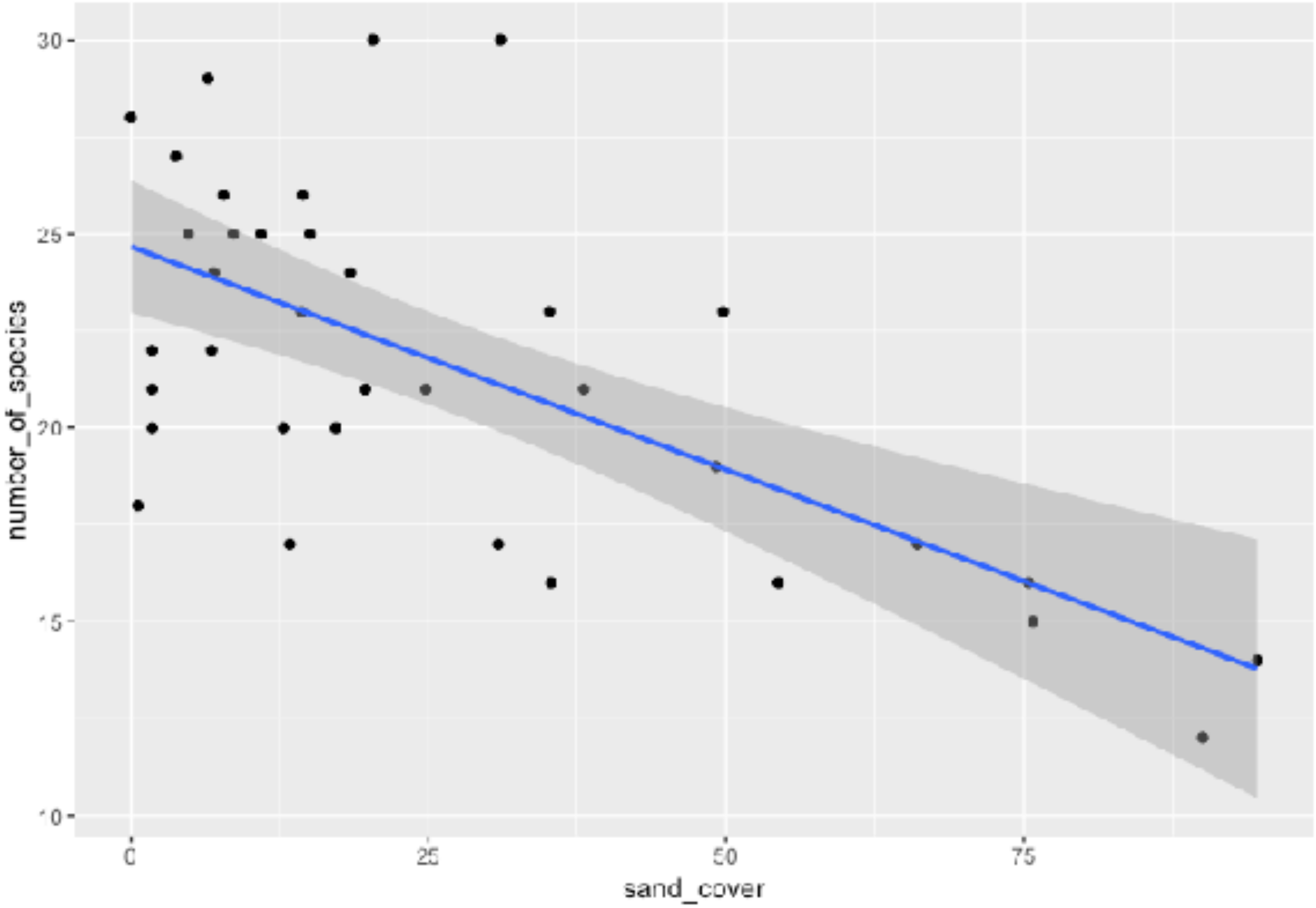
ggplot

-
  ```
  ggplot(fish_counts, aes(x = sand_cover, y = number_of_species)) +
          geom_point() +
          stat_smooth(method = “lm”)
  ```

  - aesthetic mapping

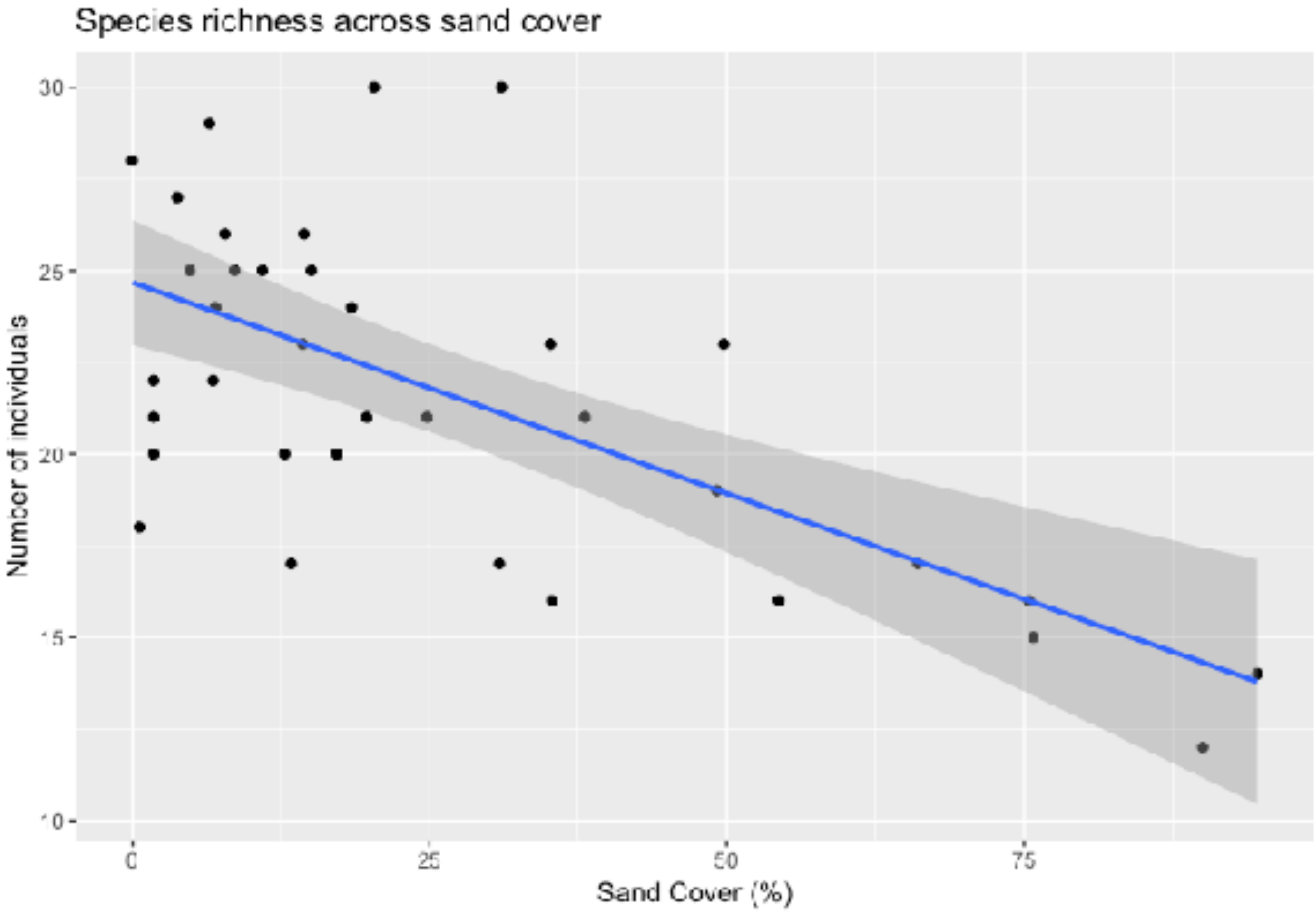
  - geometric object

  - statistical transformation



ggplot

-
  ```
  ggplot(fish_counts, aes(x = sand_cover, y = number_of_species)) +
           geom_point() +
           stat_smooth() +
           xlab(“Sand Cover (%)”) +
           ylab(“Number of individuals”) +
           ggtitle(“Species richness across sand cover”)
  ```



- aesthetic mapping
- geometric object
- statistical transformation
- labels

Summary

- Load data into R using `read_tsv`
- Tidy your data using `gather` & `spread`
- Visualize your results using `ggplot`

Exercises

- If you have your own data:
  - is it tidy?
  - what do you need to do to make it tidy?
-

Thank you!

- Further reading:
  - R for data science
    (free on http://r4ds.had.co.nz/)
  - ggplot:
    https://tutorials.iq.harvard.edu/R/Rgraphics/Rgraphics.html