Big data pipelines

Hadley Wickham
@hadleywickham
Chief Scientist, RStudio

February 2015

What is big data?

Big	Can’t fit in memory on one computer: **>1 TB**
Medium	Fits in memory on a server: **10 GB-1 TB**
Small	Fits in memory on a laptop: **<10 GB**

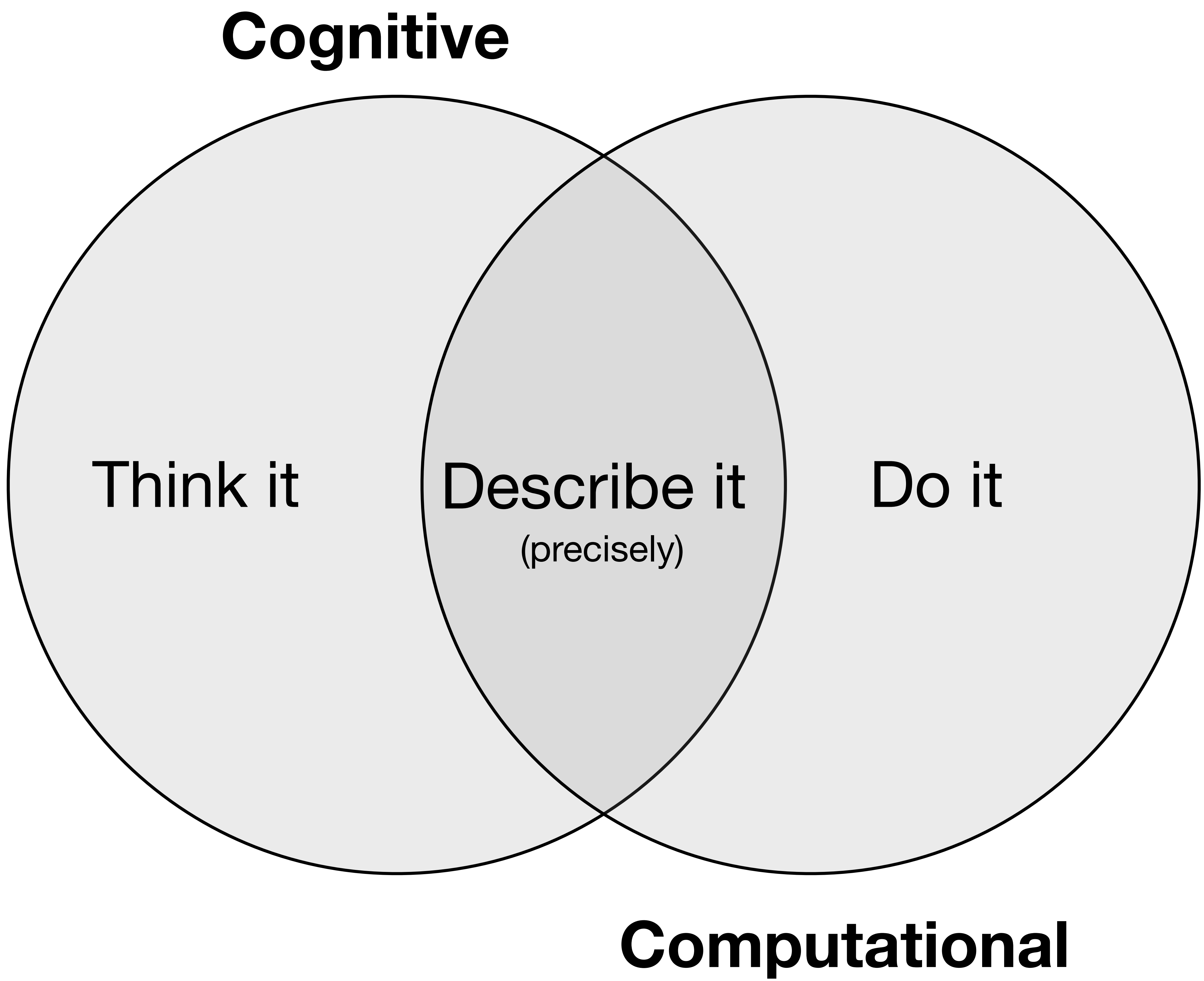

The big data **mirage**
http://www.flickr.com/photos/cleftclips/11315248764/

1. Can be reduced to a small/medium data problem with subsetting/sampling/summarising (90%)
2. Can be reduced to a very large number of small data problems (9%)
3. Is irreducibly big (1%)

Small data is still big!

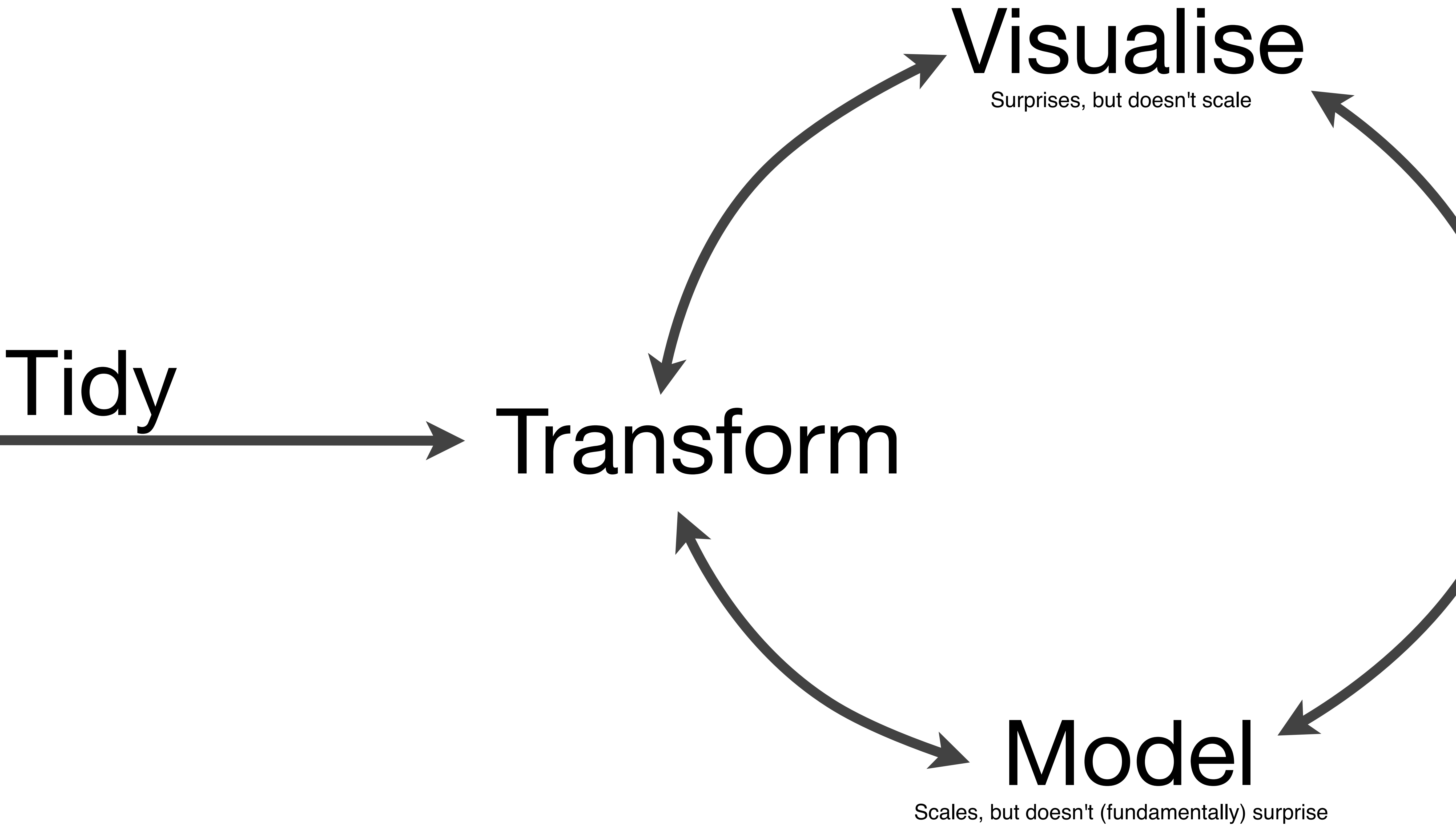
Doubles	2×10^9	200 vars, 10 million obs
Integers	4×10^9	40 vars, 100 million obs
Characters	16×10^9	5,000 copies of war and peace

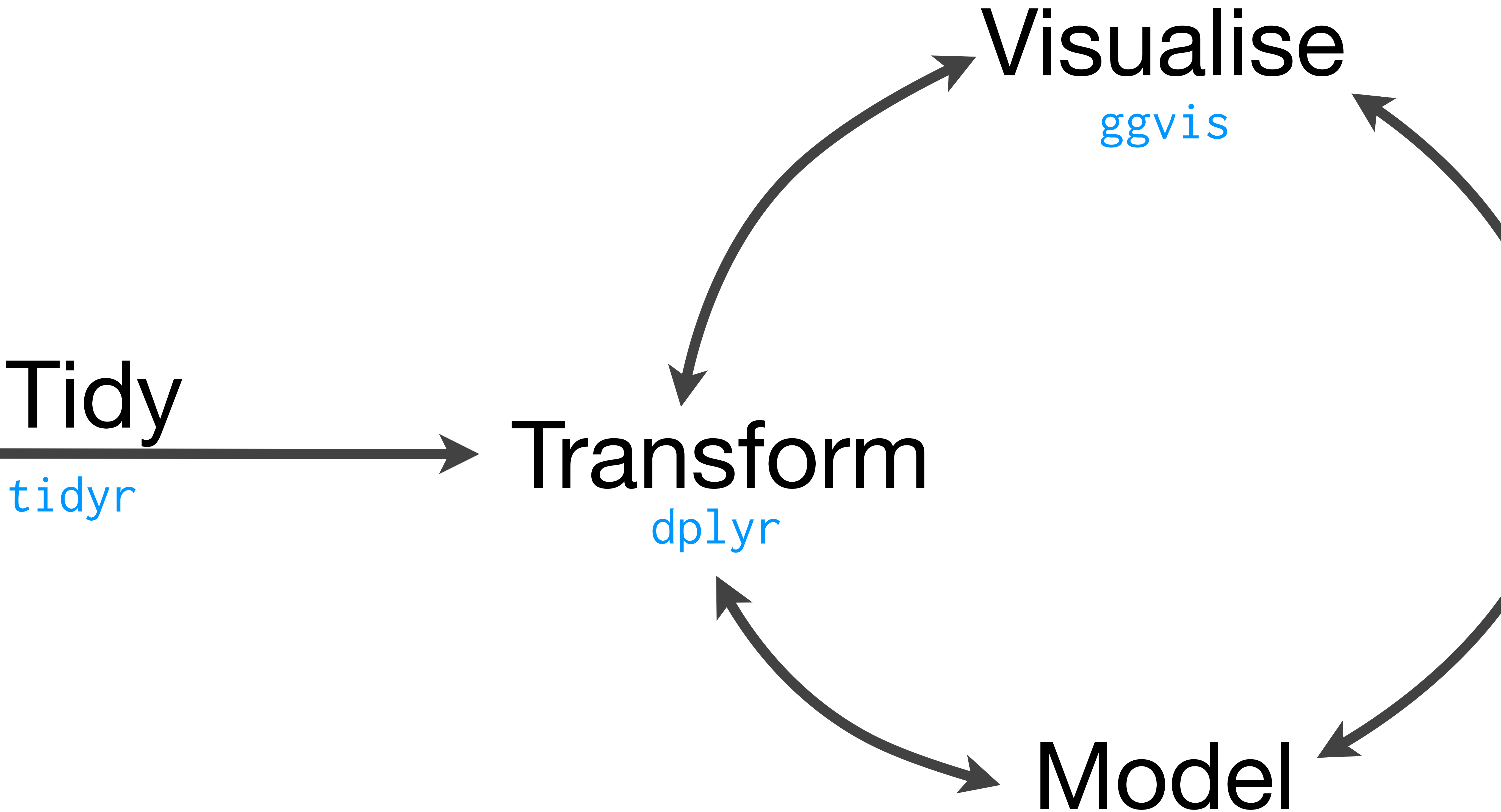
Pipelines


Cognitive
Think it
Describe it
(precisely)
Do it
Computational

Cognition time ≫ Computation time

http://www.flickr.com/photos/mutsmuts/4695658106


Visualise
Surprises, but doesn't scale
Tidy
Transform
Model
Scales, but doesn't (fundamentally) surprise

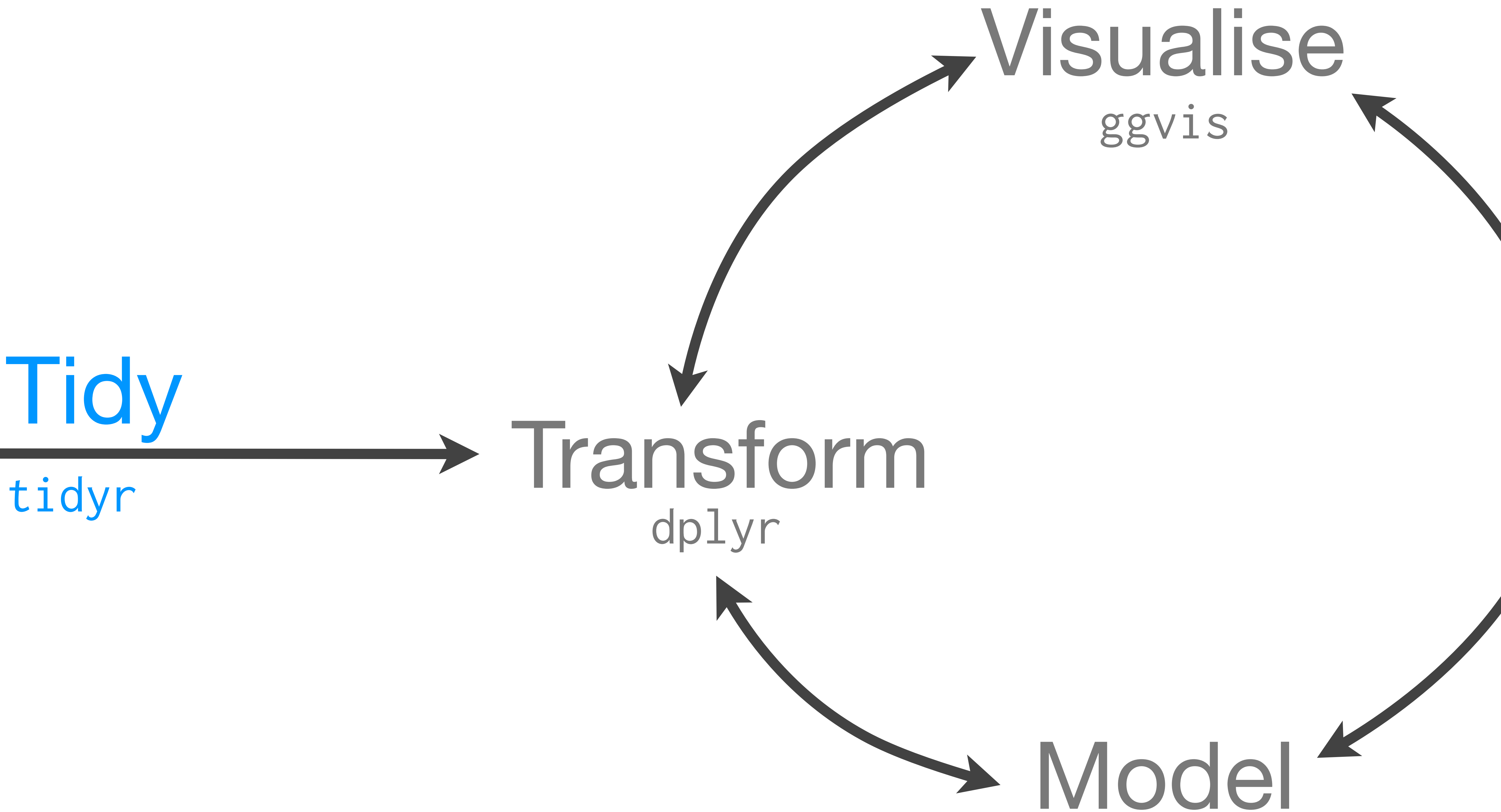

Visualise
ggvis
Tidy
tidyr
Transform
dplyr
Model

```
x %>% f(y)
# f(x, y)

x %>% f(z, .)
# f(z, x)

x %>% f(y) %>% g(z)
# g(f(x, y), z)

# Turns function composition (hard to read)
# into sequence (easy to read)
```

```r
foo_foo <- little_bunny()

bop_on(
  scoop_up(
    hop_through(foo_foo, forest),
    field_mouse
  ),
  head
)

# vs
foo_foo %>%
  hop_through(forest) %>%
  scoop_up(field_mouse) %>%
  bop_on(head)
```

```
# Any function can use it. Only needs a simple
# property: the type of the first argument
# needs to be the same as the type of the result.

# tidyr: pipelines for messy -> tidy data
# dplyr: pipelines for manipulation of tidy data
# ggvis: pipelines for visualisations

# rvest: pipelines for html/xml DOMs
# purrr: pipelines for lists
```

tidyr


Visualise
ggvis
Tidy
tidyr
Transform
dplyr
Model

Tidy data = data that makes data analysis easy

Storage	Meaning
Table / File	Data set
Rows	Observations
Columns	Variables

```
library(tidyr)
library(dplyr, warn = FALSE)
tb <- tbl_df(read.csv("tb.csv", stringsAsFactors = FALSE))
tb
#> Source: local data frame [5,769 x 22]
#>
#>   iso2 year m04 m514 m014 m1524 m2534 m3544 m4554 m5564 m65 mu f04 f514
#> 1    AD 1989  NA   NA   NA    NA    NA    NA    NA    NA  NA NA  NA   NA
#> 2    AD 1990  NA   NA   NA    NA    NA    NA    NA    NA  NA NA  NA   NA
#> 3    AD 1991  NA   NA   NA    NA
#> 4    AD 1992  NA   NA   NA    NA
#> 5    AD 1993  NA   NA   NA    NA
#> 6    AD 1994  NA   NA   NA    NA
#> 7    AD 1996  NA   NA    0     0
#> 8    AD 1997  NA   NA    0     0     1     2     2     1   6 NA  NA   NA
#> 9    AD 1998  NA   NA    0     0     0     1     0     0   0 NA  NA   NA
#> 10   AD 1999  NA   NA    0     0     0     1     1     0   0 NA  NA   NA
#>..  ...  ... ...  ...  ...   ...   ...   ...   ...   ... ... .. ...  ...
#> Variables not shown: f014 (int), f1524 (int), f2534 (int), f3544 (int),
#>  f4554 (int), f5564 (int), f65 (int), fu (i
```

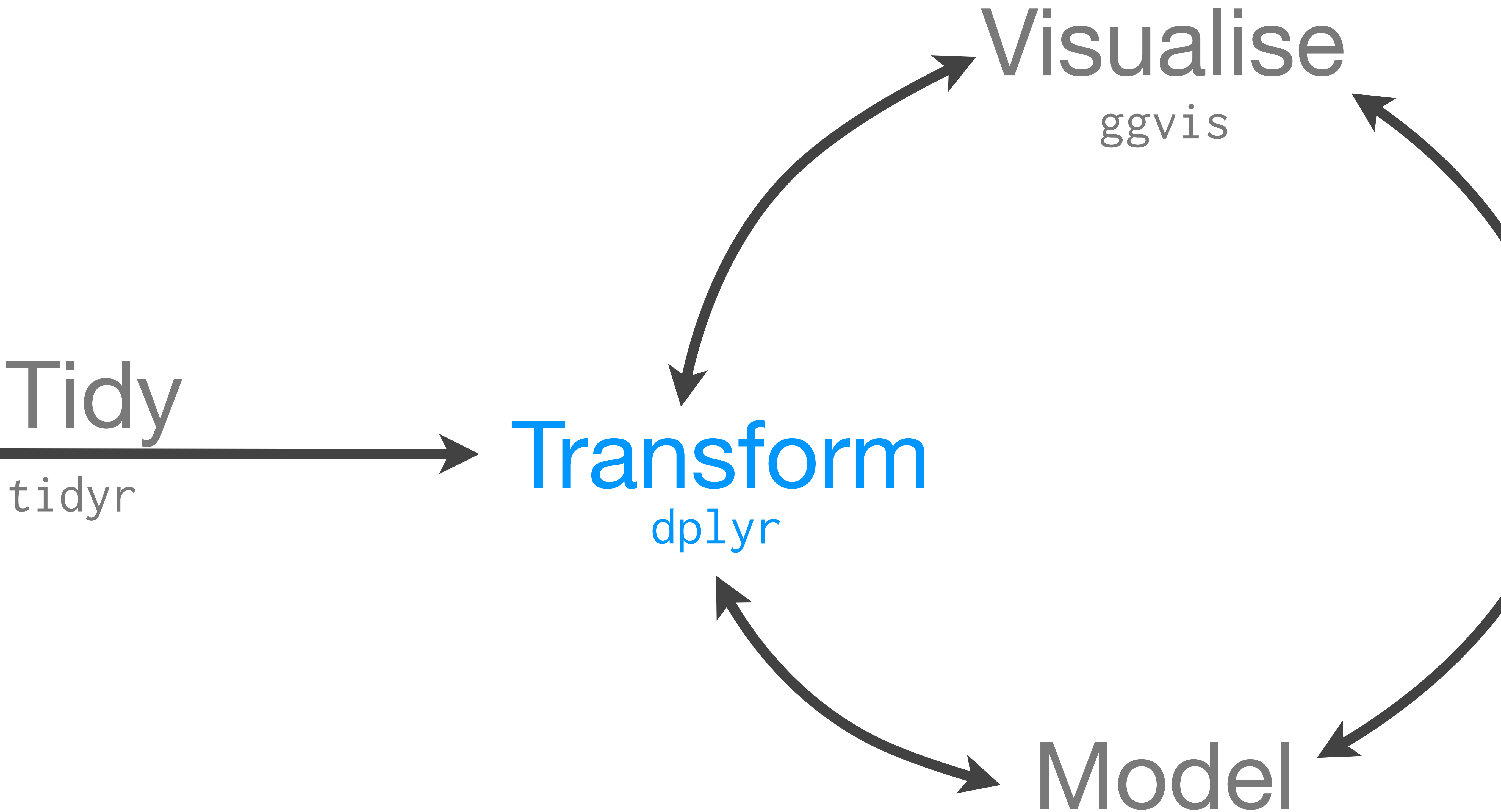
What are the variables in this dataset? (Hint: f = female, u = unknown, 1524 = 15-24)

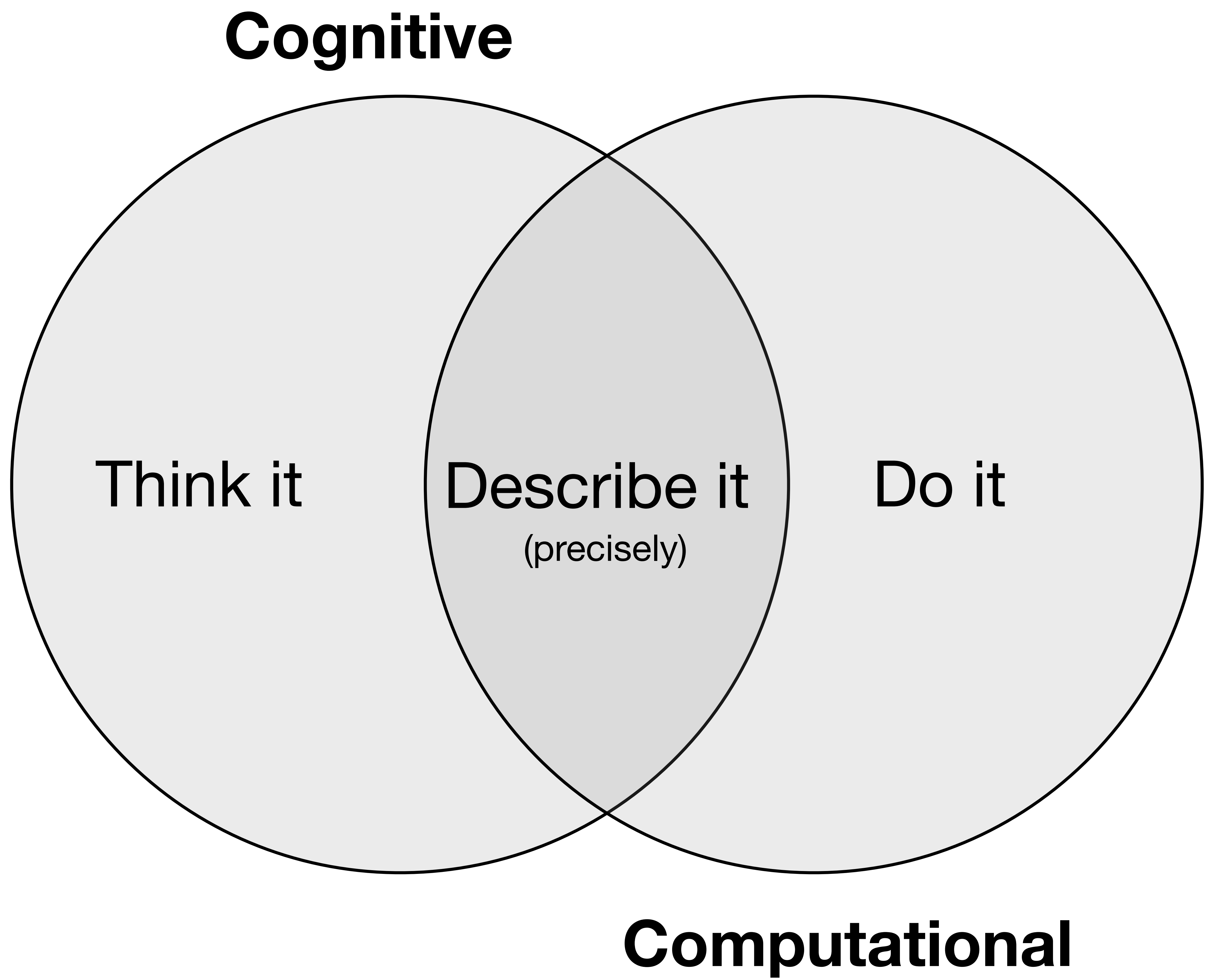
```r
# To convert this messy data into tidy data
# we need two verbs. First we need to gather
# together all the columns that aren't variables

tb2 <- tb %>%
  gather(demo, n, -iso2, -year, na.rm = TRUE)
tb2
```

```
# Then separate the demographic variable into
# sex and age
tb3 <- tb2 %>%
  separate(demo, c("sex", "age"), 1)
tb3


# tidyr provides a few other useful verbs:
# spread (opposite of gather)
# extract (like separate, but uses regexp groups)
# unite (opposite of extract/gather)
# nest & unnest, …
```

Google for **“tidyr”** & **“tidy data”**

dplyr


Visualise
ggvis
Tidy
tidyr
Transform
dplyr
Model

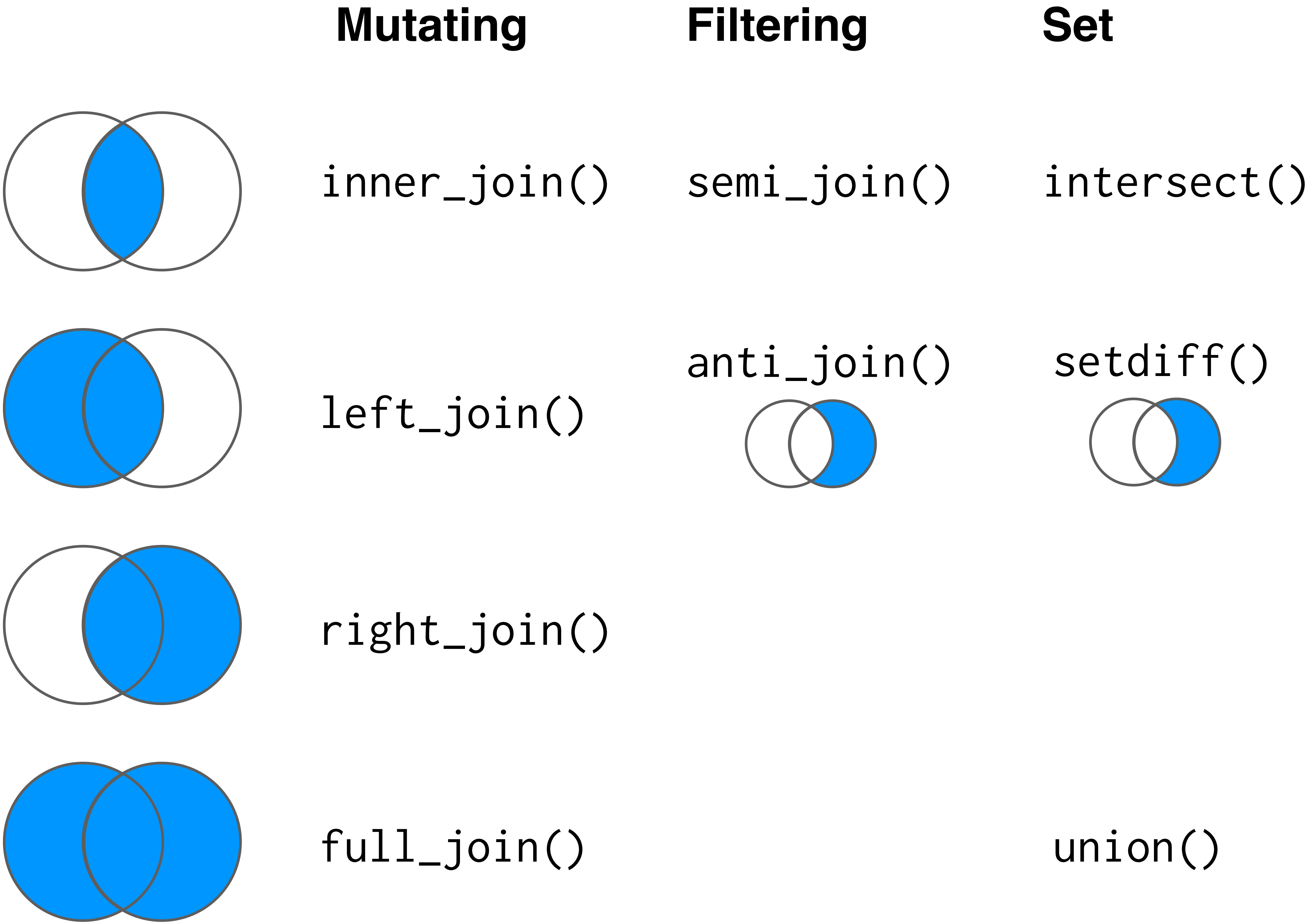

Cognitive
Think it
Describe it
(precisely)
Do it
Computational

One table verbs

+ group by

- **select**: subset variables by name
- **filter**: subset observations by value
- **mutate**: add new variables
- **summarise**: reduce to a single row
- **arrange**: re-order the rows

Demo



Mutating
Filtering
Set
inner_join()
semi_join()
intersect()
left_join()
anti_join()
setdiff()
right_join()
full_join()
union()

Move computation; not the data

http://www.flickr.com/photos/criana/7502316518/

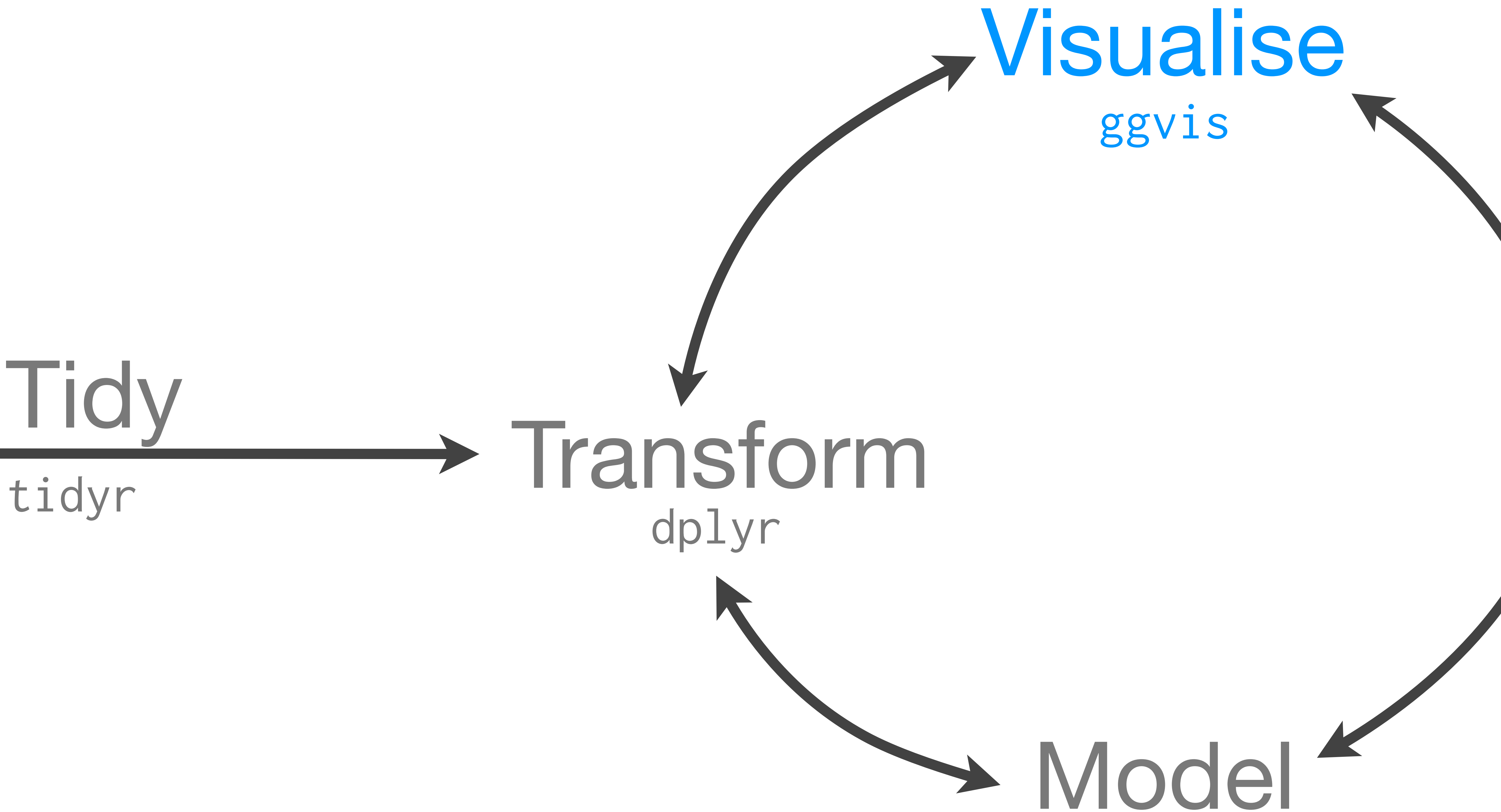
dplyr sources

- Local data frame
- Local data table
- Local data cube (experimental)
- RDMS: Postgres, MySQL, SQLite, Oracle, MS SQL, JDBC, Impala
- MonetDB, BigQuery

Google for
“dplyr”

ggvis

with Winston Chang


Visualise
ggvis
Tidy
tidyr
Transform
dplyr
Model

What is ggvis?

- A grammar of graphics (like `ggplot2`)
- Reactive (interactive & dynamic) (like `shiny`)
- A pipeline (a la `dplyr`)
- Of the web (drawn with `vega`)

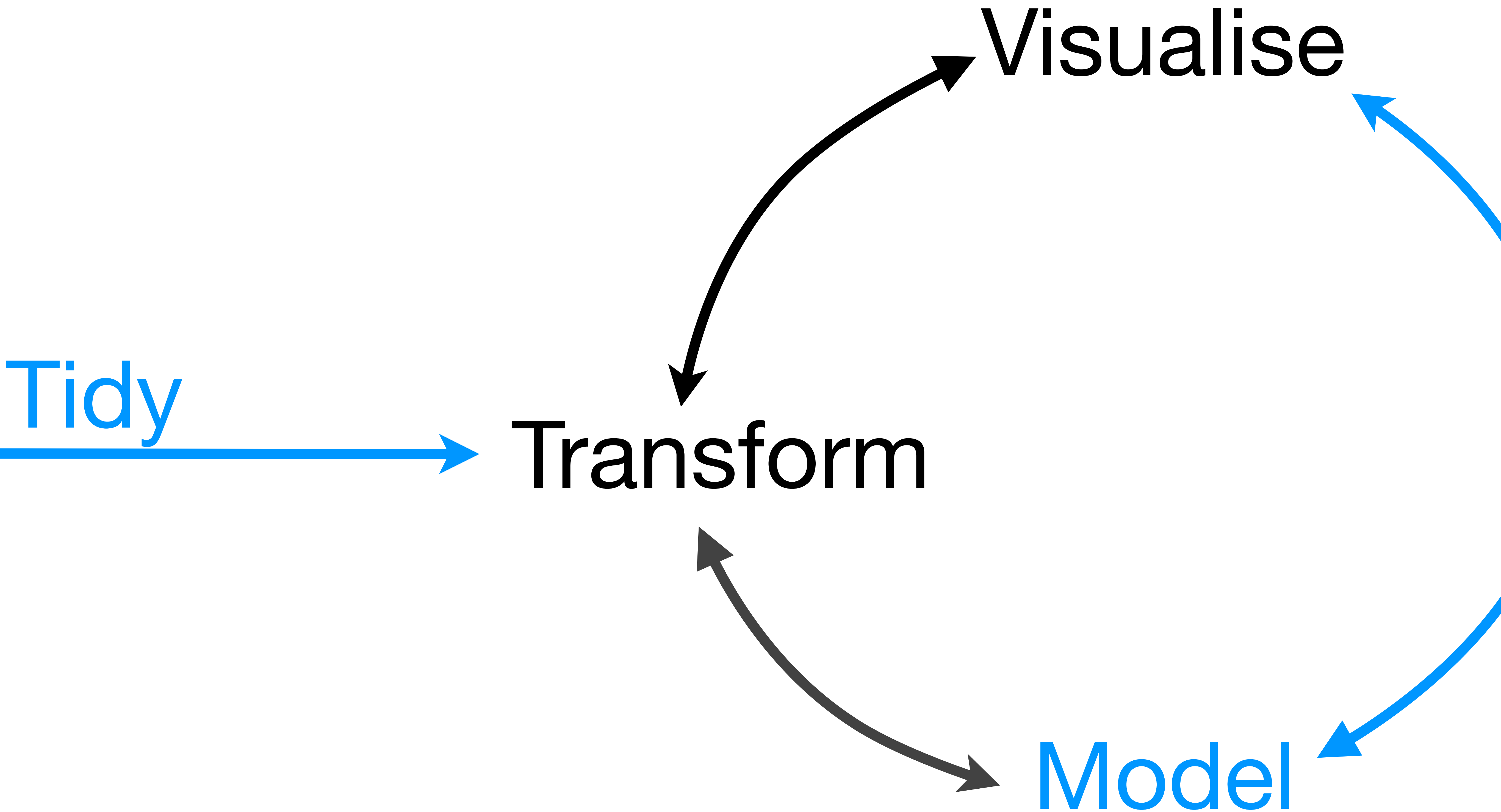
Demo

4-ggvis.R
4-ggvis.Rmd

Google for

“ggvis”

Challenges/ Future work



Visualise
Tidy
Transform
Model

broom

- https://github.com/dgrtwo/broom
- By David Robinson
- Provides verbs to convert model objects into tidy data frames

Modelling

R provides a huge variety of modelling tools, and the formula interface is common.

But… otherwise there's not a lot of consistency, and I think a lot of room for making easier to use (Zelig and caret notwithstanding)

End game

Provide a **fluent** interface where you spent your mental energy on the specific data problem, not general data analysis process.

The best tools become invisible with time!

(Currently focusing on data ingest: on disk, from databases, web apis & scraping)