R_intro_8

Karel Fišer, 2017

Preparing data for analysis

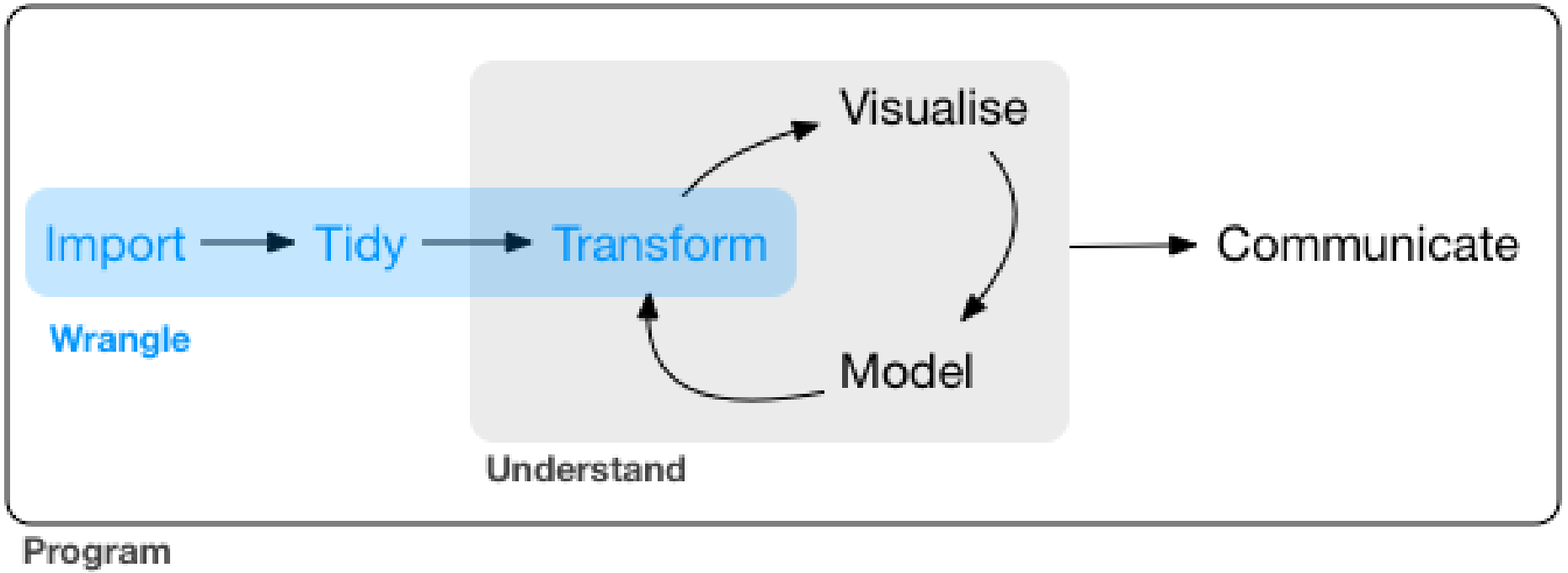
It is often said that 80% of data analysis is spent on the cleaning and preparing data.

Hadley Wickham

Hadley Wickham., Tidy data. The Journal of Statistical Software, vol. 59, 2014. DOI: 10.18637/jss.v059.i10

Dasu, T. and Johnson, T. (2003) Data Quality, in Exploratory Data Mining and Data Cleaning, John Wiley & Sons, Inc., Hoboken, NJ, USA. doi: 10.1002/0471448354.ch4

Preparing data for analysis



http://r4ds.had.co.nz/

Data – what is what

Everything in the table is either variable name, observation identifier or value (… or annotation)

Description of the data in separate file

Consistent analysis habits. e.g. data and scripts live in one folder

(consider making analysis protocol)

Rectangular data

Pretty much all of the data we deal with is rectangular: has columns and rows.

Statistically, we want variables and observations.

Variables go in columns, observations go in rows (with occasional exceptions)

Hadley Wickham

reminder ...

from previous lectures

- **change rownames**
- **order**
- **change columns order**
- **add column**
- **substitute value**
- **NAs**
- **unique**

get data into R

If spreadsheet data, check them in spreadsheet software for:

- **multiline headers**
- **colour or formatting bears important info**
- **+/- decimal separator**
- **+/- encoding**
- **sheets**
- **outlier cells**

export spreadsheet to text

```
> library(xlsx)
> dx <- read.xlsx("iris.xls", sheetIndex = 1)
```

When iris.xls → iris.txt → R

- **tab (or other delimited)**
- **encoding**
- **place on disk :-)**

```
> d <- read.table("iris.txt")
```

examine & fix data

- fix column names
- missing (and strange) values
- character vectors vs factors
- letter cases
- numbers as strings
- strip white spaces

colnames, NAs

```
> dd <- read.table("iris_ex.txt", header = TRUE)
> str(dd)
> head(dd)
> dd
>
> colnames(dd)[4] <- "Petal.Width"
>
> dd[dd=="na"] <- NA
> # or use na.strings = "na" in read.table()
> # is.na()
```

character vectors and factors

```
> dd <- read.table("iris_ex.txt", header = TRUE, stringsAsFactors = FALSE)
> str(dd)
> unique(dd$measured_by)

>
> dd$measured_by <- factor(dd$measured_by)
> dd$measured_by <- factor(dd$measured_by, levels = c("John", "Caroline"))

>
> as.numeric(dd$measured_by)
> as.numeric(factor(c(4, 4, 5)))
> # 1 1 2
> as.numeric(as.character(factor(c(4, 4, 5))))
> # 4 4 5

>
> levels(dd$measured_by)[levels(dd$measured_by)=="John"] <- "Male"
> # dd$measured_by[dd$measured_by=="John"] <- "Male"  # if it was character
```

letter cases, numerics, white space

```
> tolower(c("Big", "big", "BIG"))
> # "big" "big" "big"
>
> str(dd)
> as.numeric(dd$Sepal.Width)
> dd$Sepal.Width[is.na(as.numeric(dd$Sepal.Width))]
> # [1] "three"
> # Warning message:
> #   NAs introduced by coercion
>
> trimws(c("a", "   b", "c   ")) # or (better?)
  strip.white=TRUE in read.table()
> # "a" "b" "c"
```

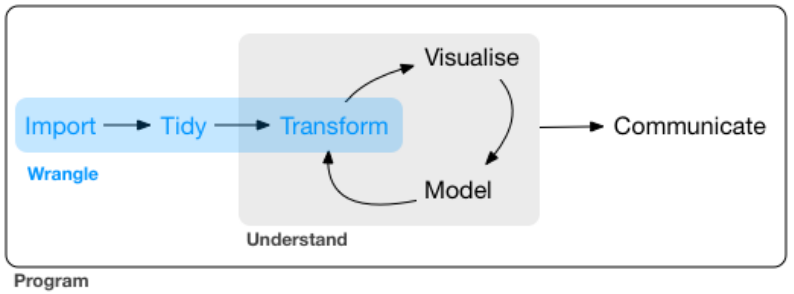
Your turn

iris.xls → iris.txt → R

examine & fix iris_ex.txt

tidy and manipulate data

http://r4ds.had.co.nz/




```
> library(tidyverse)
```

Multiple variables in one column → separate

```
> tidyr::table3
> separate(table3, rate, into = c("cases", "population"))
```

country	year	rate
Afghanistan	1999	**745** / 19987071
Afghanistan	2000	**2666** / 20595360
Brazil	1999	**37737** / 172006362
Brazil	2000	**80488** / 174504898
China	1999	**212258** / 1272915272
China	2000	**213766** / 1280428583

table3

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

```
> # By default, separate() will split values wherever it sees a
  non-alphanumeric character.
> # unite()
```

Column headers are values → gather

```
> tidyr::table4a
> gather(table4a, `1999`, `2000`, key = "year",
  value = "cases")
```

table4a

country	1999	2000
A	0.7K	2K
B	37K	80K
C	212K	213K

country	year	cases
A	1999	0.7K
B	1999	37K
C	1999	212K
A	2000	2K
B	2000	80K
C	2000	213K

key value

== wide to long data conversion

for long to wide use **`spread()`**

Your turn

separate table3 with explicit separator

iris_ex.txt -> wide to long

only some cases needed → filter

```
> filter(starwars, species == "Droid")
> # A tibble: 5 x 13
>    name height  mass hair_color  skin_color eye_color birth_year
>   <chr>  <int> <dbl>      <chr>       <chr>     <chr>      <dbl>
> 1 C-3PO    167    75       <NA>        gold    yellow        112
> 2 R2-D2     96    32       <NA> white, blue       red         33
> 3 R5-D4     97    32       <NA>  white, red       red         NA
> 4 IG-88    200   140       none       metal       red         15
> 5   BB8     NA    NA       none        none     black         NA
> # ... with 6 more variables: gender <chr>, homeworld <chr>,
> #   species <chr>, films <list>, vehicles <list>, starships
  <list>
```

only some variables needed → select

```
> select(starwars, name, birth_year)


> # A tibble: 87 x 2
>                  name birth_year
>                 <chr>      <dbl>
>  1     Luke Skywalker       19.0
>  2              C-3PO      112.0
>  3              R2-D2       33.0
>  4        Darth Vader       41.9
>  5        Leia Organa       19.0
```

new variable from existing ones → mutate

```
> mutate(starwars, bmi = mass/((height/100)^2))
> transmute(starwars, bmi = mass/((height/100)^2))
> # A tibble: 87 x 1
>         bmi
>       <dbl>
>  1 26.02758
>  2 26.89232
>  3 34.72222
>  4 33.33007
>  5 21.77778
```

summary of groups of cases → summarise

```
> summarise(starwars,  mass = mean(mass, na.rm = TRUE))
> # mean(starwars$mass, na.rm = TRUE)
>
> sw_by_species <- group_by(starwars, species)
> summarise(sw_by_species,
>           mass = mean(mass, na.rm = TRUE))
> # A tibble: 38 x 2
>      species   mass
>        <chr>  <dbl>
>  1    Aleena  15.00
>  2  Besalisk 102.00
>  3    Cerean  82.00
```

%>% is not a pipe

pass output as input for next function

`x %>% f(y)` turns into `f(x, y)`

```
> starwars %>%
>   group_by(species) %>%
>   summarise(
>     n = n(),
>     mass = mean(mass, na.rm = TRUE)
>   ) %>%
>   filter(n > 2)
```

Your turn

filter

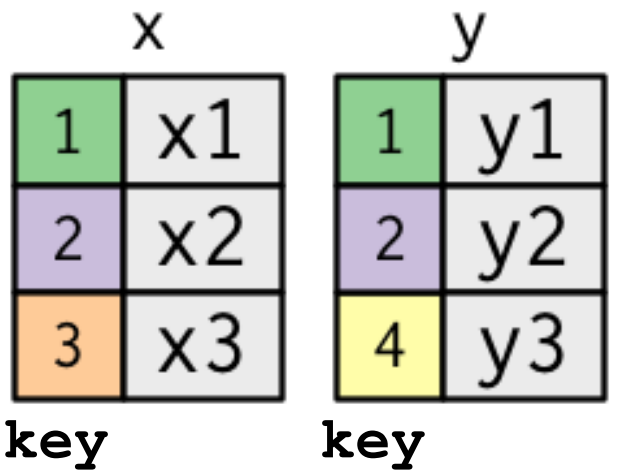
arrange

summarise

%>%

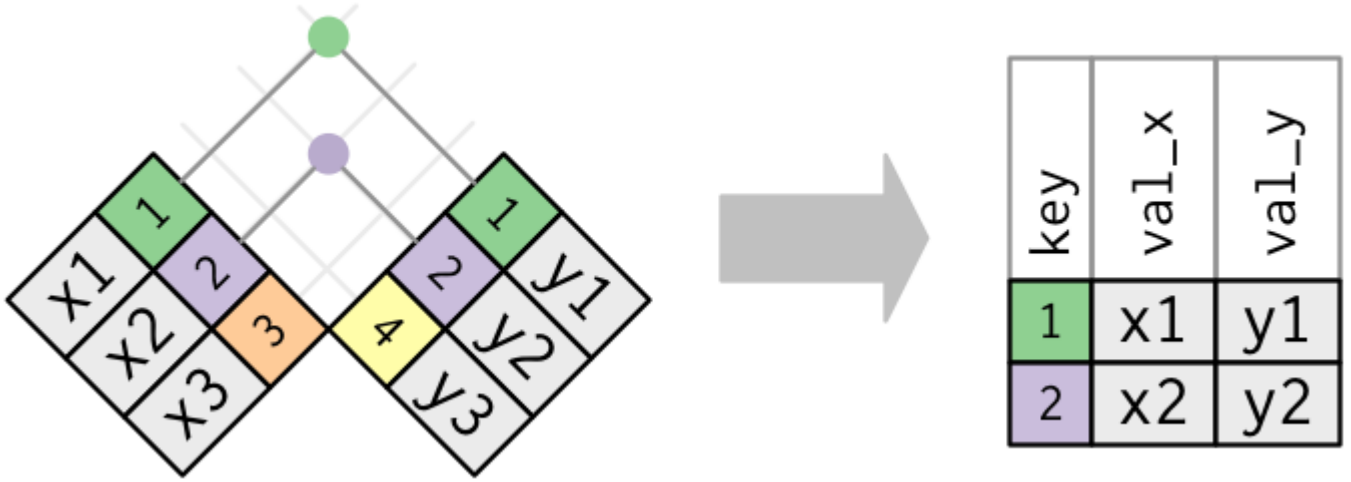
data in two tables → join

```
> x <- tibble(key = c(1,2,3),
>             val_x = c("x1", "x2", "x3"))
> y <- tibble(key = c(1,2,4),
>             val_y = c("y1", "y2", "y3"))
```



inner join

```
> inner_join(x, y, by = "key")
```

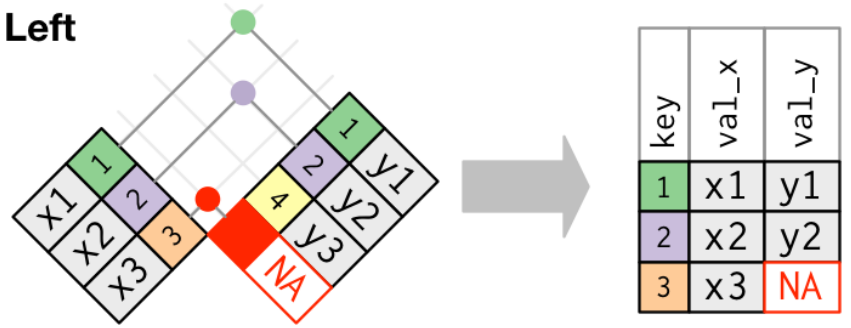


key	val_x	val_y
1	x1	y1
2	x2	y2

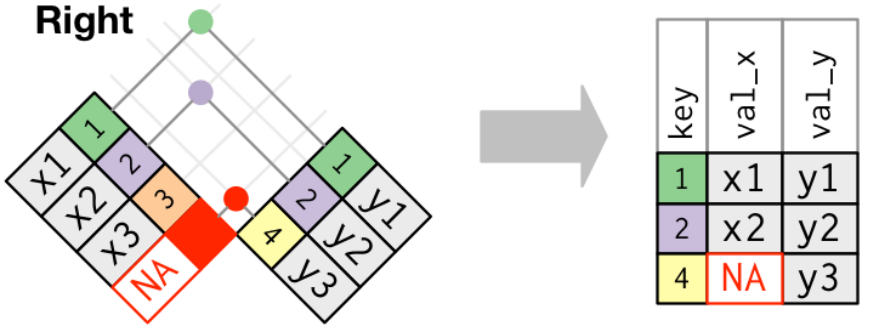
outer joins

- A **left join** keeps all observations in x.

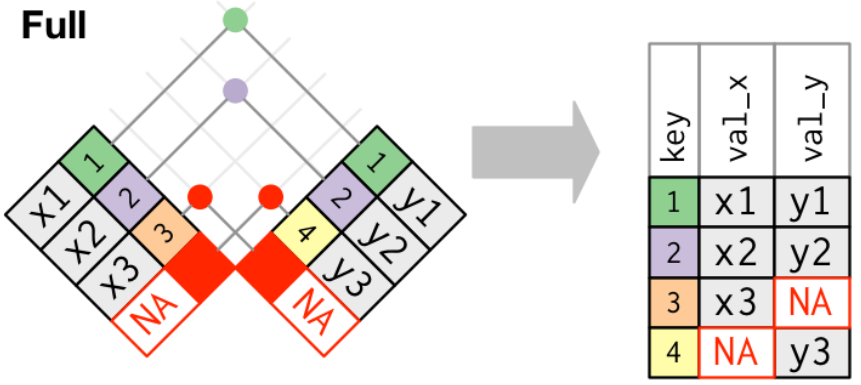
```
left_join(x, y, by="key")
```



- A **right join** keeps all observations in y.



- A **full join** keeps all observations in x and y.



tidy data

Resulting tidy data should be easier to analyse (e.g. model) and visualise (e.g. ggplot2.

However what is tidy depends also on your angle of looking at the data.

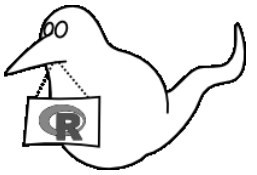
example ...

```
> library(tidyverse)
> if (file.exists("iris_ex.txt")) {
>   dd <- read.table("iris_ex.txt", header = TRUE, stringsAsFactors = FALSE)
> } else {
>   cat("\n Where are y flowers? \n")


> str(dd)
> colnames(dd)[4] <- "Petal.Width"
> dd[dd=="na"] <- NA
> dd$Sepal.Width[is.na(as.numeric(dd$Sepal.Width))] <- 3
> # all could have been done with mutate()


> dd %>%
>   gather(flower_part, measurement, -c(Species, measured_by)) %>%
>   mutate(measurement = as.numeric(measurement)) %>%
>   filter(measurement < 60) %>%
>   separate(flower_part, into = c("flower_part", "dimension")) %>%
>   ggplot(aes(x = Species, y = measurement, color = flower_part)) +
>   geom_jitter() +
>   facet_wrap(~dimension) +
>   geom_smooth(aes(x = as.numeric(as.factor(Species))),
>               method = "lm", se = FALSE, na.rm = TRUE) +
>   labs(title = "iris",
>        subtitle = "The sepal and petal lengths and widths of 3 species of iris flowers.",
>        y = "Measurement [cm]")
```

http://www.bioinformin.net



https://cran.r-project.org/web/packages/tidyr/vignettes/tidy-data.html

http://www.cookbook-r.com/Manipulating_data/

http://r4ds.had.co.nz/

https://github.com/tidyverse/tidyverse

https://www.rstudio.com/resources/cheatsheets/

RStudio

Your turn

think of data you will likely analyse in near future

prepare such data outside of R

load data in R and manipulate them so their shape fits different tasks (correlations, modelling, plotting)