

2: Tidy data

Videregående kvantitative metoder i studiet af politisk adfærd

Frederik Hjorth

adjunkt, Ph.D.

fh@ifs.ku.dk

fghjorth.github.io

@fghjorth

Institut for Statskundskab
Københavns Universitet

13. september 2018

1 Opsamling fra sidst

2 Tidy data

3 Piping

4 Øvelse

5 Kig fremad

Leeper, *Really Introductory Introduction*:

- Getting started
 - brug af R som regnemaskine: fx. $(2+4)/7$
 - parsing errors ctr. syntax errors
 - nye vektorer: fx. `dice <- c(2,2,3,4)`
 - ekstrahering fra vektorer: fx. `dice[1:3]`
 - ny data frame: fx. `df <- data.frame(dice,number=1:4)`
 - data framens struktur: `str(df)`
 - centrale tendenser: `summary(df)`

- Real data
 - installer pakker: `install.packages()`
 - indlæs pakker: `library()`
 - importér data: `import()` fra rio-pakken

- Randomness
 - sample fra en vektor: `sample()`
- Plots
 - pakke: `ggplot2`
 - fx. `ggplot(iris, aes(x=Sepal.Length)) + geom_histogram`

- Basic programming tools
 - funktioner: `fx.ftoc <- function(f){ c<-((f-35)*5)/9 ; print(c) }`
 - for loops: `fx.for (i in 1:10) print(i*i)`

DataCamp: Introduction to the Tidyverse

- 1: Data wrangling
 - gapminder-datasættet
 - 'filter'
 - 'arrange'
 - 'mutate'
- 3: Grouping and summarizing
 - 'summarize'
 - 'group_by'

Tænkere bag 'tidy data': Hadley Wickham

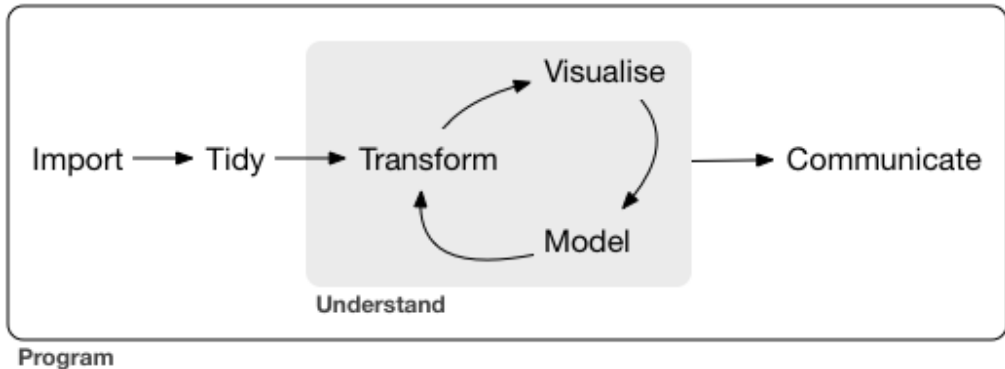


i alt forfatter til ≈ 63 R-pakker (a.k.a. 'hadleyverse' eller 'tidyverse')

Den nye generation af R-udviklere:



Et typisk data science workflow:



Centrale Hadley-pakker til databehandling:

- ggplot2 (visualisering)
- dplyr (databehandling)
- tidyr (omformning)
- readr (import af csv o. lign.)
- purrr (funktioner og loops)
- tibble (smartere version af data frames)
- stringr (behandling af character-objekter)
- forcats (faktorer)

→ alle disse kan loades samtidig med
`library(tidyverse)`



Andre nyttige hadleyverse-pakker:

- haven (import af data fra Stata, SPSS etc.)
- readxl (Excel-filer)
- rvest (web scraping)
- httr (API'er)
- lubridate (dato-objekter)

Den hyppigste, mest tidskrævende form for tidying: 'nørkling' med data

```
head(salary.selected)
```

##	name	department_name	total.earnings
## 1	Abadi,Kidani A	Assessing Department	\$46,591.98
## 2	Abasciano,Joseph	Boston Police Department	\$97,579.88
## 3	Abban,Christopher John	Boston Fire Department	\$124,623.25
## 4	Abbasi,Sophia	Green Academy	\$18,249.83
## 5	Abbate-Vaughn,Jorgelina	BPS Ellis Elementary	\$85,660.28
## 6	Abberton,James P	Public Works Department	\$50,337.63

Messy data:

	treatmenta	treatmentb
John Smith	–	2
Jane Doe	16	11
Mary Johnson	3	1

Tidy data:

person	treatment	result
John Smith	a	–
Jane Doe	a	16
Mary Johnson	a	3
John Smith	b	2
Jane Doe	b	11
Mary Johnson	b	1

Principper i tidy data:

- ① Hver variabel udgør en kolonne
- ② Hver observation udgør en række
- ③ Hver enhed udgør en tabel

NB: good for data entry \neq tidy \rightarrow det typiske datasæt er messy!

Endnu et eksempel på messy ctr. tidy:

TABLE 1 “Messy” data on video game sales and review scores. Sales data from VGChartz; review data from Metacritic. Data set sourced from Rush Kirubi, via Kaggle (bit.ly/2KNZgM9).

Name	Publisher	Global sales (million units)	Critic	User
Wii Sports	Nintendo	82.53	Score = 76, Count = 51	Score = 8, Count = 322
Mario Kart Wii	Nintendo	35.52	Score = 82, Count = 73	Score = 8.3, Count = 709
Wii Sports Resort	Nintendo	32.77	Score = 80, Count = 73	Score = 8, Count = 192
New Super Mario Bros.	Nintendo	29.80	Score = 89, Count = 65	Score = 8.5, Count = 431
Wii Play	Nintendo	28.92	Score = 58, Count = 41	Score = 6.6, Count = 129

TABLE 2 “Tidy” data on video game sales and review scores.

Name	Publisher	Global sales (million units)	Rater	Score
Wii Sports	Nintendo	82.53	Critic	76
Wii Sports	Nintendo	82.53	User	80
Mario Kart Wii	Nintendo	35.52	Critic	82
Mario Kart Wii	Nintendo	35.52	User	83
Wii Sports Resort	Nintendo	32.77	Critic	80
Wii Sports Resort	Nintendo	32.77	User	80
New Super Mario Bros.	Nintendo	29.80	Critic	89
New Super Mario Bros.	Nintendo	29.80	User	85
Wii Play	Nintendo	28.92	Critic	58
Wii Play	Nintendo	28.92	User	66

Kilde: <https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/j.1740-9713.2018.01169.x>

De fire verber i databehandling:

- ① *filter*: udvælg observationer
- ② *transform*: tilføj eller omkode variable
- ③ *aggregate*: opsummere værdier til én eller færre
- ④ *sort*: ændre observationers rækkefølge

→ vi kan foretage (næsten) al databehandling med `dplyr`

Implementering af de fire verber i dplyr:

	verbum	funktion i dplyr
1	filter	<code>filter()</code> til rækker, <code>select()</code> til kolonner
2	transform	<code>mutate()</code>
3	aggregate	<code>group_by()</code> og <code>summarise()</code>
4	sort	<code>arrange()</code>

Uvurderligt opslagsværk til databehandling: data wrangling cheat sheet

Data Wrangling with dplyr and tidy

Cheat Sheet



Syntax - Helpful conventions for wrangling

dplyr::tbl_df(iris)
Converts data to tbl class. tbl's are easier to examine than data frames. R displays only the data that fits onscreen:

```
Source: local data frame [30 x 5]
  Sepal.Length Sepal.Width Petal.Length
1 5.4 4.7 1.5
2 5.0 4.9 1.4
3 5.4 4.7 1.5
4 4.9 4.2 1.4
5 5.4 4.7 1.5
```

dplyr::glimpse(iris)
Information dense summary of tbl data.

utils::View(iris)
View data set in spreadsheets like display (note capital V).

```
iris
  Sepal.Length Sepal.Width Petal.Length
1 5.4 4.7 1.5
2 5.0 4.9 1.4
3 5.4 4.7 1.5
4 4.9 4.2 1.4
5 5.4 4.7 1.5
```

dplyr::%>%
Pastes object on left hand side as first argument (or argument of function on right hand side).

```
x %>% f(y) is the same as f(x, y)
```

Piping with %>% makes code more readable, e.g.

```
iris %>%
  group_by(Species) %>%
  summarise(mg = mean(Sepal.Width)) %>%
  arrange(mg)
```

Source: local data frame [30 x 5]

Tidy Data - A foundation for wrangling in R

In a tidy data set:

- Each variable is saved in its own column
- Each observation is saved in its own row

Tidy data complements R's **vectorized operations**. It will automatically preserve observations as you manipulate variables. No other format works as intuitively with R.

Reshaping Data - Change the layout of a data set

gather() Gather columns into rows.
spread() Spread rows into columns.

separate() Separate one column into several.
unite() Unite several columns into one.

Subset Observations (Rows)

filter() Extract rows that meet logical criteria.
distinct() Remove duplicate rows.
sample_n() Randomly select fraction of rows.
sample_frac() Randomly select fraction of rows.
slice() Select rows by position.
top_n() Select and order top n entries (by group if grouped data).

Subset Variables (Columns)

select() Select columns by name or helper function.
rename() Rename columns of a data frame.

Helper functions for select - select

contains() Select columns whose name contains a character string.
ends_with() Select columns whose name ends with a character string.
everything() Select every column.
matches() Select columns whose name matches a regular expression.
num_range() Select columns named with a numeric range.
one_of() Select columns whose name is in a group of names.
starts_with() Select columns whose name starts with a character string.
between() Select columns between two values.
where() Select columns based on a logical condition.
where_sml() Select columns based on a logical condition (smaller).

Summarise Data

summarise() Summarize data into single row of values.
summarize_each() Apply summary function to each column.
count() Count number of rows with each unique value of variable (with or without weights).

summary() Summarise uses **summary functions**, functions that take a vector of values and return a single value, such as:

- first**: First value of a vector.
- last**: Last value of a vector.
- nth**: nth value of a vector.
- n**: # of values in a vector.
- n_distinct**: # of distinct values in a vector.
- IQR**: IQR of a vector.
- min**: Minimum value in a vector.
- max**: Maximum value in a vector.
- mean**: Mean value of a vector.
- median**: Median value of a vector.
- var**: Variance of a vector.
- sd**: Standard deviation of a vector.

Group Data

group_by() Group data into rows with the same value of Species.
ungroup() Remove grouping information from data frame.
iris %>% group_by(Species) %>% summarise(...) Compute separate summary row for each group.

iris %>% group_by(Species) %>% summarise(...) Compute separate summary row for each group.

Make New Variables

mutate() Compute and append one or more new columns.
mutate_each() Apply window function to each column.
transmute() Compute one or more new columns. Drop original columns.

lead() Copy with values shifted by 1.
lag() Copy with values lagged by 1.
dense_rank() Ranks with no gaps.
min_rank() Ranks. Ties get min rank.
percent_rank() Ranks rescaled to [0,1].
row_number() Ranks. Ties get to first value.
ntile() Bin vector into n buckets.
between() Are values between a and b?
cume_dist() Cumulative distribution.

cumall() Cumulative all.
cumany() Cumulative any.
cummean() Cumulative mean.
cumsum() Cumulative sum.
cummax() Cumulative max.
cumin() Cumulative min.
cumprod() Cumulative prod.
pmax() Element-wise max.
pmin() Element-wise min.

iris %>% group_by(Species) %>% mutate(...) Compute new variables by group.

Combine Data Sets

left_join() Join matching rows from b to a.
right_join() Join matching rows from a to b.
inner_join() Join data. Retain only rows in both sets.
full_join() Join data. Retain all values, all rows.

semi_join() Join data. Retain only rows in both sets.
anti_join() Join data. Retain all values, all rows.

intersect() Rows that appear in both y and z.
union() Rows that appear in either or both y and z.
setdiff() Rows that appear in y but not z.

bind_rows() Append z to y as new rows.
bind_cols() Append z to y as new columns. Functions match rows by position.

ligger også på fagets hjemmeside under extras

%>%

magrittr

Ceci n'est pas un pipe.

Opsamling fra sidst
○○○○○

Tidy data
○○○○○○○○○○○○○○

Piping
○●○○○○○

Øvelse
○○○

Kig fremad
○○



Data: datasættet i pakken 'gapminder'

```
> gapminder
# A tibble: 1,704 x 6
  country      continent  year lifeExp      pop gdpPercap
  <fct>        <fct>    <int>   <dbl>   <int>    <dbl>
1 Afghanistan Asia      1952    28.8  8425333    779
2 Afghanistan Asia      1957    30.3  9240934    821
3 Afghanistan Asia      1962    32.0 10267083    853
4 Afghanistan Asia      1967    34.0 11537966    836
5 Afghanistan Asia      1972    36.1 13079460    740
6 Afghanistan Asia      1977    38.4 14880372    786
7 Afghanistan Asia      1982    39.9 12881816    978
8 Afghanistan Asia      1987    40.8 13867957    852
9 Afghanistan Asia      1992    41.7 16317921    649
10 Afghanistan Asia      1997    41.8 22227415    635
# ... with 1,694 more rows
> |
```

<https://youtu.be/Z8t4k0Q8e8Y?t=54s>

Vi ønsker at gøre flg.:

- ① gruppér observationerne efter land
- ② udregn gennemsnitslevetid og medianindbyggertal over hele perioden for hvert land
- ③ reducer til lande m. mere end 10M indbyggere
- ④ sortér efter land (omvendt alfabetisk) og gennemsnitslevetid

Klodset løsning I: Gem hvert skridt i et nyt objekt (→ rod)

```
by_country <- group_by(gapminder, country)

country_sum <- summarize(by_country, avglifeexp=mean(lifeExp),
                          medianpop=median(pop))

big_countries <- filter(country_sum, medianpop > 10000000)

result1 <- arrange(big_countries, desc(country), avglifeexp)
```


Klodset løsning II: Skriv funktionerne indlejret (nested) i hinanden (→ totalt ulæseligt)

```
result2 <-  
  arrange(  
    filter(  
      summarize(  
        group_by(gapminder,  
                  country  
                ),  
        avglifeexp=mean(lifeExp),  
        medianpop=median(pop)  
      ),  
      medianpop > 10000000  
    ),  
    desc(country),  
    avglifeexp  
  )
```

Smart løsning: brug pipede funktioner (→ enkelt, letlæseligt)

```
result3 <- gapminder %>%  
  group_by(country) %>%  
  summarize(avglifeexp=mean(lifeExp),  
    medianpop=median(pop)) %>%  
  filter(medianpop > 10000000) %>%  
  arrange(desc(country), avglifeexp)
```

bemærk: den originale data frame 'pipes' ned gennem funktionerne, så angives kun én gang



»This directory contains data on nearly 3 million tweets sent from Twitter handles connected to the Internet Research Agency, a Russian "troll factory" and a defendant in an indictment filed by the Justice Department in February 2018, as part of special counsel Robert Mueller's Russia investigation. (...) The basis for the Twitter handles included in this data are the November 2017 and June 2018 lists of Internet Research Agency-connected handles that Twitter provided to Congress. This data set contains every tweet sent from each of the 2,752 handles on the November 2017 list since May 10, 2015.«

Kilde: <https://github.com/fivethirtyeight/russian-troll-tweets>

- ① hent et af datasættene fra
`https://github.com/fivethirtyeight/russian-troll-tweets`
- ② hvad er de hyppigste og næsthyppigste sprog?
- ③ i hvilken region har trolde flest følgere?
- ④ hvilket sprog retweetes oftest?
- ⑤ hvor ofte nævnes hhv. Trump og Clinton i tweetsne?

Næste gang: Data fra online-kilder

- screen scraping
- API'er
- pensum: MRMN kap. 9+14
- vigtigt i kap. 9: 9.1.10+
- case: Hjorth (ananas i egen juice ~→ eksempel på data fra online-kilder, læs kursorisk)
- ekstra hjemmearbejde: lav en twitter API key

Opsamling fra sidst
○○○○○

Tidy data
○○○○○○○○○○○○○

Piping
○○○○○○○

Øvelse
○○○

Kig fremad
○●

Tak for i dag!