

R Kursus: Tidyverse

08/02/2023

Section 1

Introduktion

Plan for dagen

10.15	Oplæg om grundlæggende R og Tidyverse-pakken Follow-along med eksempel
12.00	Pause med sandwiches
12.30	Øvelse/Konkurrence
14.00	Deadline for konkurrence bidrag
14.10	Annoncering af vindergruppe
14.15	Tak for i dag!

R for dummies: sådan starter du

- Åbn et nyt R Script eller Markdown-fil
- Indlæs de pakker, du skal bruge, med `library()`
- Indlæs data i filen - IKKE i console
 - Definér datasættet, så du kan bruge det senere
 - `data <- read_csv("stinavn/til/datasæt")`
 - Gælder også kodning: skriv altid al koden i din fil
- Brug for hjælp til en pakke eller funktion?
 - Skriv `?funktion` i console, så kommer der info op direkte i R
- Generel hjælp, eller hvis du får en fejl: **BRUG GOOGLE**

Section 2

Tidyverse

Tidyverse

Tidyverse er en samling af flere underliggende pakker

Tidyverse

Tidyverse er en samling af flere underliggende pakker

- De pakker, vi måske har brugt mest:

Tidyverse

Tidyverse er en samling af flere underliggende pakker

- De pakker, vi måske har brugt mest:
 - `dplyr`: bedre til datamanipulation end Base R

Tidyverse

Tidyverse er en samling af flere underliggende pakker

- De pakker, vi måske har brugt mest:
 - `dplyr`: bedre til datamanipulation end Base R
 - `ggplot2`: bruges til mere avanceret visualisering sammenlignet med `plot`-funktionen i Base R

Tidyverse

Tidyverse er en samling af flere underliggende pakker

- De pakker, vi måske har brugt mest:
 - `dplyr`: bedre til datamanipulation end Base R
 - `ggplot2`: bruges til mere avanceret visualisering sammenlignet med `plot`-funktionen i Base R
 - Bonus: `ggplot2` giver meget pænere grafer

Tidyverse

Tidyverse er en samling af flere underliggende pakker

- De pakker, vi måske har brugt mest:
 - `dplyr`: bedre til datamanipulation end Base R
 - `ggplot2`: bruges til mere avanceret visualisering sammenlignet med `plot`-funktionen i Base R
 - Bonus: `ggplot2` giver meget pænere grafer
- Tidyverse er designet til at arbejde med pipe funktioner (`%>%`), som gør koden mere overskuelig.

Pipes

```
library(tidyverse)
```

Hvilken kode er nemmest at læse?

```
x <- mutate(filter(select(cars, dist),  
                    dist > 4 & dist < 20),  
             sum_dist = cumsum(dist))
```

```
y <- cars %>% select(dist) %>%  
  filter(dist > 4 & dist < 20) %>%  
  mutate(sum_dist = cumsum(dist))
```

Pipes

- Det første objekt inden `%>%` er det første argument i den efterfølgende funktion
- Vi bruger `%>%`, når vi skal lave flere ændringer i vores datasæt i en bestemt rækkefølge

```
my_day <- day %>%  
  got_up() %>%  
  had_breakfast() %>%  
  programmed_some_r() %>%  
  had_lunch() %>%  
  programmed_some_r() %>%  
  had_dinner() %>%  
  went_to_bed()
```

Datamanipulation med dplyr

De funktioner, I primært kommer til at bruge, hvis I skal lave datamanipulation:

- `select()` udvælger variable ud fra deres navn
- `filter()`: udvælger cases baseret på deres værdier
- `mutate()`: tilføjer nye variable, som er funktioner af eksisterende variable
- `summarise()`: reducerer flere værdier til én opsummering
- `arrange()`: sorterer rækkerne fra mindste til største værdi

Ekstra:

- `pivot_wider()`: Transformerer dit data til **Wider** format
- `pivot_longer()`: Transformerer dit data til **Longer** format

Datamanipulation med dplyr

Se altid på dit datasæt, før du begynder at manipulere i variablene.

```
head(starwars)
```

Hvis der er mange rækker bruger vi `glimpse()` i stedet:

```
glimpse(starwars)
```

Datamanipulation med dplyr

Select()

`select()` lader dig udvælge en del din dataframe efter kolonne (variabel), hvilket betyder at output kun indeholder bestemte kolonner i den valgte rækkefølge.

Her vælger jeg name og mass

```
starwars %>%  
  select(name, mass)
```

Her fjerner jeg name og mass

```
starwars %>%  
  select(-c(name, mass))
```


Datamanipulation med dplyr

Filter()

`filter()` lader dig udvælge en del af din dataframe efter rækker (observationer), hvilket betyder at output filtreres til kun at indeholde rækker, som opfylder en bestemt betingelse.

```
starwars %>%  
  filter(skin_color == "gold")
```

Vigtigt: Brug `==`, `\<=`, `\>=`, `\>` eller `\<` for at definere betingelsen.

Datamanipulation med dplyr

Mutate

`mutate()` lader dig manipulere med eksisterende variable eller lave nye.

Vi kan lave en ny variabel:

```
starwars %>%  
  select(mass, height) %>%  
  mutate(bmi = mass*height) %>%  
  head(3)
```

Datamanipulation med dplyr

Mutate

Eller ændre en nuværende variabel:

```
starwars %>%  
  select(name, mass) %>%  
  mutate(mass = ifelse(mass < 50 , "Thin", "Fat")) %>%  
  head(3)
```

Datamanipulation med dplyr

Summarize

`summarize()` reducerer dit datasæt til én observation, som er summeret ud fra en defineret funktion.

```
starwars %>%  
  drop_na() %>%  
  summarise(mean(mass))
```

Datamanipulation med dplyr

arrange()

arrange() definerer måden, hvorpå rækkerne i din dataframe er sorteret.

#Fra størst til mindst:

```
starwars %>%  
  select(mass) %>%  
  arrange(desc(mass))
```

#Fra mindst til størst:

```
starwars %>%  
  select(mass) %>%  
  arrange(mass)
```

Datamanipulation med dplyr

Long dataformat

Et data format oftest brugt af fx. OECD, inkluderer flere variable i en kolonne, og kaldes **longer**.

```
fish_encounters %>% head(5)
```

Datamanipulation med dplyr

Wider dataformat

Det data format I nok har set mest indtil videre er **wider** formatet, en variabel per kolonne.

```
relig_income %>%  
  head(5)
```

Datamanipulation med dplyr

pivot_wider

Hvordan går vi fra longer format til wider?

Eksempel på pivot_wider():

```
fish_encounters %>%  
  pivot_wider(names_from = station,  
              values_from = seen) %>%  
  head(5)
```


Datamanipulation med dplyr

pivot_longer

Hvordan går vi fra **wider** format til **longer**?

Eksempel på pivot_longer()

```
relig_income %>%  
  pivot_longer(!religion, names_to = "income",  
               values_to = "count") %>%  
  head(5)
```

Visualisering af data med ggplot2

- Aesthetics: de visuelle egenskaber for geom, som positioner for x og y, linjefarve, punktformer, m.v.
- Geoms: geometriske objekter, som repræsenterer data, som barer, linjer og punkter
- Scales: bestemmer afblding fra data til aesthetics
- Labels og Legends: bestemmer bl.a. titel på plot, aksemærkater m.v.

Visualisering af data med ggplot2

Step 1: Aesthetics

Hver gang vi laver et plot starter vi med at specificere aesthetics:

```
data %>% ggplot(aes(x = data$x, y = data$y,  
                    fill= data$z))
```

Step 2: Geoms

Herefter tilføjer vi hvad selve plottet skal indeholde med geometries:

```
data %>% ggplot(aes(x = data$x, y = data$y,  
                    fill= data$z)) +  
  geom_point()
```

Visualisering af data med ggplot2

1 variabel

```
starwars %>%  
  ggplot(aes(height)) +  
  geom_histogram()
```

2 variable

```
txhousing %>%  
  filter(city %in% c("Dallas", "Houston")) %>%  
  ggplot(aes(x = date, y = median, color = city)) +  
  geom_point()
```

Visualisering af data med ggplot2

Scales

Scales afbilder dataværdier til de visuelle værdier af en aesthetic: størrelse, farve, form, akser m.v.

R sætter selv skaleringen ud fra data - men det kan vi ændre.

- `scale*_continuous`: Bestem skalering af kontinuert data
 - `n.breaks`: antal “breaks”, som visualiseres
 - `breaks`: Tving hvilke bestemte lokationer, aksemarkeringer skal være ved
 - `limits`: modificér start- og slutværdi
 - `labels`: angiver hvordan aksemarkeringerne skrives - kan fx. være nyttigt til at angive at data er valuta

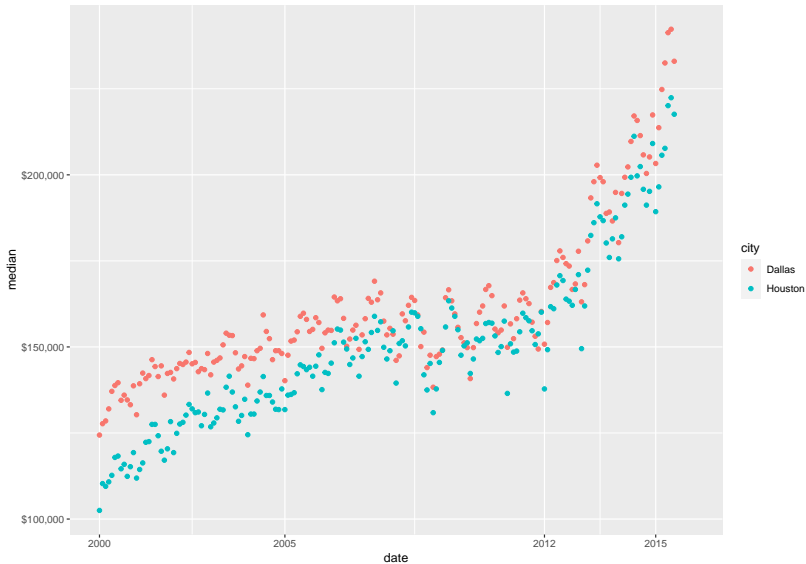
* erstattes med x eller y

Visualisering af data med ggplot2

Scales

```
txhousing %>%  
  filter(city %in% c("Dallas", "Houston")) %>%  
  ggplot(aes(x = date, y = median, color = city)) +  
  geom_point() +  
  scale_x_continuous(breaks = c(2000, 2005, 2012, 2015),  
                    limits = c(2000, 2016)) +  
  scale_y_continuous(labels = scales::dollar)
```

Visualisering af data med ggplot2



Visualisering af data med ggplot2

Scales

```
txhousing %>%  
  filter(city %in% c("Dallas", "Houston")) %>%  
  ggplot(aes(x = date, y = median, color = city)) +  
  geom_point() +  
  scale_x_continuous(n.breaks = 4,  
                     limits = c(2005, 2010))
```


Visualisering af data med ggplot2

Labels og Legends

Brug `labs()` til at navngive elementer i grafen

```
txhousing %>%  
  filter(city %in% c("Dallas", "Houston")) %>%  
  ggplot(aes(x = date, y = median, color = city)) +  
  geom_point() +  
  labs(title = "Title",  
        subtitle = "Subtitle",  
        x = "New x label",  
        y = "New y label",  
        caption = "Source: the source",  
        color = "Color")
```

Visualisering af data med ggplot2

Labels og Legends

```
txhousing %>%  
  filter(city %in% c("Dallas", "Houston")) %>%  
  ggplot(aes(x = date, y = median, color = city)) +  
  geom_point() +  
  theme(legend.position = c(.9, .8))
```

Visualisering af data med ggplot2

Texas Housing Prices
2005–2010



Source: R package ggplot2