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	Øvelser
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, sum er funktioner af eksisterende variable
- `summarise()`: reducerer flere værdier til én opsummering
- `arrange()`: sorterer rækkerne fra mindste til største værdi

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 subset en dataframe by column (variable), 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() lets you subset a dataframe by rows (observations), 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 variable:

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

Visualisering af data med ggplot2

- Aesthetics: visual properties of geoms, such as x and y position, line color, point shapes, etc.
- Geoms: geometric objects that are drawn to represent the data, such as bars, lines, and points.
- Scales: bestemmer mapping af værdierne i data space til værdierne i det aesthetic space.
- Labels og Legends: bestemmer bl.a. titel på plot, aksemærkater m.v.

Visualisering af data med ggplot2

Step 1: Aesthestics

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

```
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 map dataværdier til de visuelle værdier af en aesthetic

- `scale_*_continuous`: Set scales for continuous data
  - n.breaks: number of breaks visualized.
  - breaks: points for certain breaks.
  - limits: modify start and end value.

* 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))
```

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 = "bottom")
```

Visualisering af data med ggplot2

Sæt det hele sammen