Tidyverse og gg2plot

Simon

15/11/2022

1. Kort intro:
2. Tidyverse
3. GG2plot

Section 1

Kort intro:

Kort intro:

Tidyverse

Tidyverse består af flere underliggende pakker, en af disse er dplyr som jeg bruger til ca. 90% af det data manipulation jeg laver. Hvorfor?

- Syntax er meget det samme som SQL
- De fleste funktioner i dplyr er hurtigere end base R funktioenr
- Designet til at arbejde med pipe funktioner som gør koden mere overskueligt.

gg2plot

- Bruges til mere advanceret visualisering sammenlignet med Base-R's plot funktion.

Section 2

Tidyverse

tibbles

Tidyverse benytter tibbles som er lavet på følgende 3 måder:

1. benyt selve tibble() funktionen
2. Benyt as_tibble() på en eksisterende tabel
3. Hvis du bruger en hver dplyr funktion på en data frame bliver denne til en tibble.

Pipes

Det objekt der står inden "pipe" bliver sendt ind som det første argument i det der står efter.

Se nedenstående eksempel.

```
library(tidyverse)
x= c(1,2,3,4,5,6,7,8,9,10)
mean(x)
x %>% mean()
```

Pipes

Så vi bruger pipes når vi skal lave flere ændringer til vores dataset i en bestemt rækkefølge.

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

Pipes

Eksempel på data manipulation med Tidyverse: *Yderligere beskrivelse af funktioner kommer*

Vi bruger dataset fra cars pakken

```
library(car)
```

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

```
### Hvis ik jeg bruger pipes:
mutate(filter(select(cars,dist),dist > 4 & dist< 20)
       ,sum_dist= cumsum(dist))
```

Pipes

Vigtigt

Når jeg bruger pipe omskrives det første objekt ikke, hvis jeg ønsker dette brug "margrittr" :

```
library(magrittr)
```

Så vent lidt med at kør nedenstående

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

head(cars)
```

pipes

Alternativt giv et nyt navn

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

vigtige dplyr functions

1. filter() picks cases based on their values.
2. select() picks variables based on their names.
3. arrange() changes the ordering of the rows.
4. mutate() adds new variables that are functions of existing variables
5. summarise() reduces multiple values down to a single summary.

Eksempler med dplyr functions

Lad os tage et lidt større dataset den her gang

```
head(starwars)
```

Hvis der er mange rækker brug glimpse() i stedet:

```
glimpse(starwars)
```

Eksempler med dplyr functions

Filter()

The verb filter() lets you subset a dataframe by rows (observations), meaning the output will filter for only rows which fulfill a certain condition.

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

Vigtigt Brug "==", "<=", ">=", ">" eller "<" for at definer betingelse.

Eksempler med dplyr functions

Select()

The verb select() lets you subset a dataframe by column (variable), meaning the output will only contain certain columns in the stated order

```
### Her vælger jeg name og mass
starwars %>%
  select(name, mass)
### Her fjerner jeg name og mass
starwars %>%
  select(-c(name, mass))
```

Eksempler med dplyr functions

arrange()

The verb arrange() defines the way the rows of your dataframe are ordered

```
#Fra størst til mindst:

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

#Fra mindst til størt:

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

Eksempler med dplyr functions

Mutate

The verb mutate() lets you manipulate existing variables or create new ones.
Vi kan lave en ny variable:

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

```
## # A tibble: 3 x 3
##    mass height   bmi
##   <dbl>  <int> <dbl>
## 1    77    172 13244
## 2    75    167 12525
## 3    32     96  3072
```

Eksempler med dplyr functions

Mutate

The verb mutate() lets you manipulate existing variables or create new ones.
Eller ændre en nuværrende variabel

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

```
## # A tibble: 3 x 2
##   name           mass
##   <chr>          <chr>
## 1 Luke Skywalker Fat
## 2 C-3PO          Fat
## 3 R2-D2          Thin
```

Eksempler med dplyr functions

Summarize

The verb summarize() reduces your dataset to one observation, which is summarized according to a defined function.

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

```
## # A tibble: 1 x 1
##   `mean(mass)`
##          <dbl>
## 1         84.5
```

Section 3

GG2plot

Intro til gg2plot

Vi kigger i dag på de to vigtigste elementer under gg2plot, som de selv beskriver på følgende måde:

- **Aestetics**: Devine the "surface" of your plot, in terms of what has to be mapped (size, color) on the x and y (and potentially adittional) axes. Aesteticts are defined within the aes() function.
- **Geometries**: Visual elements you can see in the plot itself, such as bars, lines, and points. They are defined within various geom_XYZ() functions.

Intro til gg2plot

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

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

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 1 variable

Lad os bruge et nyt dataset: Det kan hentes direkte fra github med nedenstående kode:

```
bike <- readRDS(url("https://github.com/SDS-AAU/SDS-master/raw/master/00_data/bikes_montreal.rds?dl=1"))
```

Brug glimpse for at se dataet:

```
glimpse(bike)
```

```
bike %>% head()
```

visualisering af 1 variable

Histogram

- Vi kan lave histogram med geom_histogram()

```
bike %>% ggplot(aes(x = duration_sec)) +
  geom_histogram()
```

density plot

- vi kan lave density plot med geom_density()

```
bike %>% ggplot(aes(x = duration_sec)) +
  geom_density()
```

visualisering af 2 variable:

```
bike %>%
  count(start_day, weekday) %>%
  ggplot(aes(x=start_day, y=n, color = weekday)) +
  geom_point()
```

- Lad os tjekke hvorfor vi skriver y=n!

visualisering af 2 variable:

Vi kan også tilføje titel samt yderlige beskriverlser med labs() funktionen:

```
bike %>%
  count(start_day, weekday) %>%
  ggplot(aes(x=start_day, y=n, color = weekday)) +
  geom_point() +
labs(title = "Cykelturer fordelt på dage", caption = "Kilde: C
  xlab("Start dag") +
  ylab("Antal cykelturer")
```

- For mere info om ggplot2 check (https://app.datacamp.com/learn/courses/introduction-to-data-visualization-with-ggplot2)