**Seminar 3**

**Hva vi skal gå igjennom på dette seminaret**

1. Mappestruktur og working directory
2. Laste inn data
3. Omkoding av variabler
4. Subsetting av datasett
5. Plotting: ggplot2
6. Ulike typer plott
7. Lagring av plott

**1 Mappestruktur og working directory**

Når vi skal begynne å jobbe i et script, setter vi først working directory til den mappen vi ønsker å hente og lagre filer til, for eksempel filer med datasett.

Her kan man enten bruke "setwd"-funksjonen eller så kan man trykke "Session" i verktøy-linjen - "Set Working Directory" - "Choose Directory" for å velge en mappe.

setwd("C:/Users//Desktop/STV1020")

Pass på at når du laster ned filer, for eksempel filer med datasett, at du lagrer filen i samme mappe som den du har satt som working directory.

Man kan også lage prosjekter i R, der man oppretter en mappe hvor både script og figurer lagres til. Mer om dette på seminaret.

**2 Laste inn data**

Dette seminaret skal vi bruke et datasett som ligger i en pakke som heter gapminder. For å gjøre det må vi først installere pakken og åpne den.

install.packages("gapminder")

library(gapminder)

Nå er datasettet tilgjengelig i R og du vil se at det kommer opp når du skriver gapminder inn i scriptet. I dag skal vi kode om variablene i datasettet og vi oppretter derfor et objekt av datasettet ved å bruke denne koden, der det før pilen er det vi kaller datasettet for:

gapminder <- gapminder

Nå er det klart til bruk. Men for å øve på å lage og laste inn datasett skal vi lagre dette datasettet i et R-format ved å bruke funksjonen save(). Husk at koden vil være ulik for alle og tilpasset din mappestruktur. Først må ha satt en working directory, ved å bruke koden setwd(). Deretter bruker du save() til å lagre datasettet. Du vil nå se datasettet i den mappen du har bedt R om å lagre det i. Nå kan du laste opp datasettet ved å bruke funksjonen load()

save(gapminder, file = "gapminder.Rdata")

load("gapminder.Rdata")

Da kommer datasettet inn som «gapminder»

**3 Omkoding av variabler**

Når vi omkoder variabler i et datasett, bør vi opprette en ny variabel. Dersom vi ikke gjør dette, erstatter vi informasjonen i den opprinnelige variabelen. Vi trenger informasjonen i den opprinnelige for å teste at omkodingen har fungert som vi ønsker. Det kan også skje at vi gjør en feil som vi ikke kan rette opp uten den opprinnelige variabelen. Først bruker følgende disse funksjonene for å få rask oversikt over data, og du kan også trykke på datasettet i environment:

names(gapminder)

head(gapminder)

summary(gapminder)

**Matematisk omkoding**

Først skal vi lære å kode om variabler ved å bruke matematisk omkoding. For å vise dette bruker vi variabelen year. Den laveste verdien på variabelen er året 1952. Det er året observasjonen ble gjort. Vi vil gjøre dette året til år null der de andre verdiene på variabelen blir hvor mange år etter år null observasjonen ble gjort.

summary(gapminder$year)

gapminder$year\_1952 <- gapminder$year - 1952

summary(gapminder$year\_1952)

Så undersøker vi om omkodingen er blitt gjort riktig ved å bruke funksjonen table(). Det kan gjøres på to måter:

table(gapminder$year\_1952, gapminder$year)

table((gapminder$year\_1952 + 1952) == gapminder$year)

Vi kan også endre retningen på verdiene ved matematisk omkoding. Da ganger vi variablene med (-1) og plusser på antall verdier variablene har (legger til maksimumsverdien som er 55). Nå blir 2007 år null.

gapminder$year\_2007 <- gapminder$year\_1952\*(-1) + 55

table(gapminder$year\_1952, gapminder$year\_2007)

**Omkoding med ifelse()**

En annen nyttig funksjon til omkoding er ifelse(). Argumentene i funksjonen fungerer slik: ifelse(data$gammel\_variabel == gammel verdi på variablene, ny verdi på gammel verdi, hva skal vi gjøre med de andre verdiene). I koden under sier vi at alle enheter med verdien 1952 på variabelen year skal ha verdien 0 og alle andre enheter skal beholde sine gamle verdier.

gapminder$year\_0 <- ifelse(gapminder$year == 1952, 0, gapminder$year)

Så undersøker vi om omkodingen er riktig

table(gapminder$year\_0)

Vi vil også kode om verdiene på variabelen lifeExp og gi alle enhetene med en verdi lavere gjennomsnittet verdien 1. Gjennomsnittet finner du ved å bruke funksjonen mean(). I kode nummer 2 vil vi ikke beholde de gamle verdiene, men gjør i stedet alle enheter med en verdi over gjennomsnittet om til 0.

gapminder$lifeExp\_2 <- ifelse(gapminder$lifeExp < 59.47, 1, gapminder$lifeExp)

gapminder$lifeExp\_2 <- ifelse(gapminder$lifeExp < 59.47, 1, 0)

Med ifelse() kan man også kode om flere verdier samtidig.

Vi kan også bruke ifelse() til å kode om en variabel med et nominalt målnivå til en dummyvariabel, der Asia får kategorien 1 og de andre kontinentene får kategorien 0.

gapminder$continent\_dummy <- ifelse(gapminder$continent == "Asia", 1, 0)

table(gapminder$continent\_dum, gapminder$continent)

**4 Subsetting av datasett**

Å subsette betyr å lage et nytt datasett basert på et gammelt. Til dette bruker vi funksjoner i pakken dplyr, som finnes i tidyverse-universet. Derfor henter vi tidyverse fra biblioteket. Med dplyr bruker vi %>% for å binde sammen ulike operasjoner vi vil utføre på et datasett. Det er en effektiv og ryddig måte å kode på. Først må vi hente pakken fra biblioteket:

library(tidyverse)

Funksjonen select() brukes til å velge ut hvilke variabler vi vil ha med i det nye datasettet

gapminder\_subset <- gapminder %>%

select(pop, country, year)

Funksjonen filter() velger ut spesielle enheter som tilfredsstiller de kravene du setter. Under spesifiserer vi at vi kun vil ha med observasjoner med verdien 1952 på variabelen year, og som også har en verdi høyere enn gjennomsnittet på variabelen pop

gapminder\_subset <- gapminder\_subset %>%

filter(year == 1952,

pop > mean(pop))

Nå vi subsette, med alt i en operasjon. Bruk funksjonene og opprett et nytt datasett som er slik du ønsker. For eksempel:

gapminder2 <- gapminder %>%

select(pop, country, year) %>%

filter(year == 1952,

pop > mean(pop))

**5 Plotting: ggplot2**

Det finnes flere ulike måter å plotte på i R. Pakken vi skal bruke heter ggplot2, som også er en del av tidyverse-universet. Den trenger vi ikke hente fra biblioteket fordi vi allerede har gjort det tidligere i scriptet.

For å bygge plott i ggplot må vi sette sammen flere argumenter, eller deler med informasjon, for at ggplot skal vite hva slags plott vi ønsker å lage. Når vi går gjennom dette, kommer vi gradvis til å legge på flere lag i plottet. Når dere koder selv så trenger dere ikke kjøre hver kodebit separat, men heller alt kjøres samlet på én gang, slik det ser ut i den siste koden vi lager. Nå deler jeg det opp kun for å forklare grundig hvordan plots bygges opp. Du kan hoppe over denne «bygge delen» dersom du føler du har kontroll på dette, og heller se på plot-delen der alle argumenter er samlet opp.

*PS: Dette trenger dere ikke tenke så mye på akkurat nå, men til en annen gang så er det lurt å huske på at ggplot2 vil ha long-format data, dvs. at hver rad (vannrett) er en observasjon, og hver kolonne (loddrett) er ulike variabler. Dette kan dere lese mer om i kapittel 3 (s. 56) i Healy-boka på pensum.*

Steg 1 er å fortelle ggplot hvor den skal hente data fra. Det gjør vi ved å spesifisere datasett, i vårt tilfelle gapminder:

ggplot(data = gapminder)

Hva skjer når vi kjører koden nå? Vi får et helt tomt plott. Mer info må til!

Steg 2 er å fortelle ggplot hva vi vil måle på x- og y-aksen. Nå skal vi starte med et stolpediagram, der vi vil se hvor mange observasjoner det er i datasettet fordelt på de fem kontinentene. Da trenger vi bare å gi ggplot beskjed om at vi vil ha “continent” på x-aksen. Y-aksen vår blir automatisk til antall. Senere i dette seminaret skal vi se på et eksempel hvor vi gir informasjon om begge aksene.

Inni parentesen hvor vi har fortalt hvilket datasett vi vil bruke, så legger vi til en “mapping”. aes står for aestethics, og det vi gjør ved bruk av aes er å fortelle ggplot hvordan vi vil at variablene skal vises visuelt. Vi forteller ggplot at vi vil at kontinentene skal vises på x-aksen.

ggplot(data = gapminder, aes(x = continent))

Steg 3 er å skrive hva slags plott vi vil lage. Vi vil lage et stolpediagram, og velger geom\_bar. Hvis du prøver å skrive inn “geom\_” vil du se at du får en hel rekke forslag til ulike plott. Flere av disse skal vi komme tilbake til.

ggplot(data = gapminder, aes(x = continent)) +

geom\_bar()

Da har vi fått opp et plott som viser observasjonene i datasettet fordelt på kontinenter. Vi kan dobbeltsjekke at dette ser riktig ut ved å lage et summary av variabelen.

summary(gapminder$continent)

Ser bra ut. Vi kan også legge inn andre typer av aestetics i mappingen. Her er det flere muligheter, som dere vil bli kjent med etter hvert. Én av dem er å bruke ulike farger for å gi informasjon. Hvis vi vil at hvert kontinent skal få sin egen farge, så kan vi gjøre det slik:

ggplot(gapminder, aes(x = continent, fill = continent)) +

geom\_bar()

I steg 4 kan vi også endre teksten på x- og y-aksen, og gi plottet en tittel, ved å legge til labs:

ggplot(gapminder, aes(x = continent, fill = continent)) +

geom\_bar() +

labs(x = "Kontinenter",

y = "Antall land/år observasjoner",

title = “Søylediagram over kontinenter i datasettet”)

Da har vi gått gjennom det helt grunnleggende i ggplot. Dette er en pakke med veldig mange ulike muligheter, og man blir bedre kjent med de for hver gang man bruker den. Anbefaler å prøve seg fram, og sjekke ut noen av ressursene som er linket til i bunnen av dokumentet.

**6 Ulike typer plott**

Nå skal vi lage ulike typer av plott ved brukt av ggplot. Hva slags plott du vil lage avhenger av hva slags variabler det er du vil undersøke og målenivået på variablene. For eksempel er det ikke så nyttig å lage et stolpediagram hvor du har en kontinuerlig variabel på x-aksen (for eksempel forventet levealder, som er oppgitt i desimaltall). Det én stolpe for hver unike verdi = veldig, veldig mange stolper.

Da kan det være mer nyttig med et **histogram**, som lager stolper for intervaller:

ggplot(gapminder, aes(lifeExp)) +

geom\_histogram()

Hvis vi ikke sier noe annet, så kommer ggplot til å velge hvor stort hvert intervall skal være. Vi kan sette hver stolpe til å være ett år bred ved å skrive inn binwidth = 1:

ggplot(gapminder, aes(lifeExp)) +

geom\_histogram(binwidth = 1)

Hvis vi vil se hvordan dette fordeler seg på hvert kontinent, så kan vi legge inn fill:

ggplot(gapminder, aes(lifeExp, fill = continent)) +

geom\_histogram(binwidth = 1)

Dette gir oss en viss oversikt over hvordan forventet levealder varierer mellom kontinentene, men hvis vi vil se næremere på dette så kan et **boxplot** være et bedre alternativ:

ggplot(gapminder, aes(x = continent, y = lifeExp, fill = continent)) +

geom\_boxplot()

En annen metode kan være å bruke et **density plot/tetthetsplott**:

ggplot(gapminder, aes(x = lifeExp)) +

geom\_density()

Inne i parentesen til plottypen kan vi legge til spesifikasjoner om tykkelse på linja, farge osv.

ggplot(gapminder, aes(x = lifeExp)) +

geom\_density(size = 1.5, fill = "pink", alpha = 0.3)

Hva skjer hvis vi endrer alpha til 1 og size til 0.5?

ggplot(gapminder, aes(x = lifeExp)) +

geom\_density(size = 0.5, fill = "pink", alpha = 1)

Hvis du ønsker å separere plottene, slik at de ulike kontinentene ikke ligger over hverandre, men får hvert sitt plott, så kan du legge til en facet\_wrap:

ggplot(gapminder, aes(x = lifeExp)) +

geom\_density(size = 0.5, fill = "pink", alpha = 1) +

facet\_wrap(vars(continent))

Med et **scatterplot** kan vi vise sammenhengen mellom to kontinuerlige variabler. Her må vi gi ggplot informasjon om hvilken variabel som skal mappes på x-aksen og på y-aksen. For å lage scatterplottet bruker vi geom\_point.

ggplot(gapminder, aes(x = lifeExp, y = gdpPercap)) +

geom\_point()

Vi kan legge til en linje som viser gjennomsnittet i observasjonene ved å bruke geom\_smooth:

ggplot(gapminder, aes(x = lifeExp, y = gdpPercap)) +

geom\_point() +

geom\_smooth()

Skille kontinentene fra hverandre ved bruk av ulike farger:

ggplot(gapminder, aes(x = lifeExp, y = gdpPercap, col = continent)) +

geom\_point() +

geom\_smooth()

Vi kan legge inn en facet-wrap for at det skal se litt ryddigere ut, og samtidig gi geom\_smooth en farge som vises bedre, nemlig svart.

ggplot(gapminder, aes(x = lifeExp, y = gdpPercap, col = continent)) +

geom\_point() +

geom\_smooth(colour = "black") +

facet\_wrap(vars(continent))

Her kan vi repetere hvordan vi legger til nye titler til x- og y-aksen, samt tittel for hele plottet:

ggplot(gapminder, aes(x = lifeExp, y = gdpPercap, col = continent)) +

geom\_point() +

geom\_smooth(colour = "black") +

facet\_wrap(vars(continent)) +

labs(x = "Forventet levealder",

y = "BNP per innbygger",

title = "Et scatterplot med Gapminder-data")

**7 Lagring av plott**

Til sist skal vi gå gjennom hvordan man kan lagre plottene. Underveis i arbeidet kan du lagre plottene dine som objekter i environment ved å gi de navn slik:

p <- ggplot(gapminder, aes(lifeExp, fill = continent)) +

geom\_histogram(binwidth = 1)

Dette gjør at du også kan bygge videre på plottene dine uten å få fryktelig lange koder. Hvis vi vil legge til en tittel til plottet, så kan vi nå bare gjøre slik:

p + labs(title = "Et scatterplot med Gapminder-data")

Du kan lagre plottene dine i working directory ved å bruke ggsave. For å lagre det siste plottet du har kjørt i R, gjør følgende: Bruk ggsave, gi plottet et navn, og spesifiser filtypen du vil lagre i.

I .png format:

ggsave("gdplevealder.png")

I .pdf-format:

ggsave("gdplevealder.pdf")

For å lagre et plott som er et objekt gjør du følgende: ggsave(“navnpåplott.filtype”, = objektnavnet)

ggsave("histogram\_gapminder.png", plot = gm\_h)

**Ressurser for ggplot2:**

[**https://ggplot2.tidyverse.org/**](https://ggplot2.tidyverse.org/)

[**https://socviz.co/makeplot.html#makeplot**](https://socviz.co/makeplot.html#makeplot)

[**http://euclid.psych.yorku.ca/www/psy6135/tutorials/gapminder.html**](http://euclid.psych.yorku.ca/www/psy6135/tutorials/gapminder.html)