**Seminar 3**

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

1. Repetisjon: mappestruktur
2. Repetisjon: om laste inn data
3. Omkoding av variabler
4. Subsetting av datasett
5. Plotting: Intro til ggplot2
6. Ulike typer plott
7. Lagring av plott

**1 Repetisjon: mappestruktur**

(Gjenta det som har blitt sagt på seminar 1 og 2, se på hva det er og legg inn her)

**2 Laste inn data**

I dag 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.

gapminder <- gapminder

Nå er det egentlig 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 du sette 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 bet R om å lagre det. Nå kan du laste opp datasettet ved å bruke funksjons load()

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

load("gapminder.Rdata")

**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 (dette hender). Først bruker jeg disse funksjonene for å få litt oversikt over data. Du kan også trykke på datasettet i environment

show(variable.names(gapminder))

head(gapminder)

summary(gapminder)

**Matematisk omkoding**

Først skal vi lære å kode om variablene ved å bruke matematiske omkoding. For å vise dette bruker jeg variablene year. Den laveste verdien på variablene er året 1952. Det er året observasjonen ble gjort. Jeg vil gjøre dette året til år null de andre verdiene på variablen skal bli hvor mange år etter år null ble observasjonen gjort.

summary(gapminder$year)

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

summary(gapminder$year\_1952)

Så undersøker jeg om omkodingen er blitt gjort riktig ved å bruke funksjonen table(). Jeg gjør dette på to måter.

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

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

Jeg kan også endre retningen på verdienes ved matematisk omkoding. Da ganger jeg variablene med (-1) og plusser på antall verdier variablene har. Nå blir 2007 år null.

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

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

**Om koding med ifelse()**

En svært nyttig funksjon til omkoding, er ifelse(). Argumentene i funksjonen fungere slik: ifelse(data$gammel\_variabel == gammel verdi på variablene, ny verdi på, hva skal vi gjøre med de andre verdiene). I koden under sier jeg at alle enheter med verdien 1952 på variablene 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 jeg om omkodingen har skjedd riktig

table(gapminder$year\_0)

table(gapminder$year\_0, gapminder$year)

Jeg vil også kode om verdiene på variable lifeExp og gi alle enhetene med en verdi lavere gjennomsnittet på variablene verdien 1. gjennomsnittet finner du ved å brukedette funkjsonen mean(). I koden under vil jeg ikke beholde de gamle verdiene, men gjør i stedet alle enheter med en verdi under gjennomsnittet missing (NA)

gapminder$lifeExp\_2 <- ifelse( gapminder$lifeExp < 60.71, 1, gapminder$lifeExp)

gapminder$lifeExp\_2 <- ifelse(gapminder$lifeExp < 60.71, 1, NA)

Med ifelse() kan du også kode om flere verdier samtidig slik jeg gjør her:

gapminder$lifeExp\_2 <- ifelse(gapminder$lifeExp < 60.71, 1,

ifelse(gapminder$lifeExp > 60.71, 2, gapminder$lifeExp))

table(gapminder$lifeExp\_2)

Jeg kan også bruke ifelse() til å kode om en variabel med et nominalt målnivå til en dummy variabel

gapminder$continent\_dum <- ifelse(gapminder$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 dylpyr. Med dplyr bruker vi %>% for å binde sammen ulike operasjoner vi vil utføre på et datasett. Dette kalles for piping. Det er en effektiv og ryddig måte å kode på. Derfor kommer jeg ofte til å bruke pipes i seminarene. Først må du installer å åpne pakken

install.packages("dplyr")

library(dplyr)

Den første funksjonen select() brukes til å velge ut hvilke variabler du vil ha med i det nye datasettet

gapminder %>%

select(pop)

gapminder %>%

select(pop, country, year)

Du kan også bruke funksjonen filter() til å velge ut spesielle enheter som tilfredsstiller de kravene du setter. Under har jeg brukt funksjonen til å bare ta med enhetene med verdien 1954 på variabelen year og som har en verdi høyere end gjennomsnittet på variablene pop

gapminder %>%

filter(year == 1952)

gapminder %>%

filter(year == 1952,

pop > mean(pop))

Nå kan du subsette! Bare bruk funksjonen du har lært og opprett et nytt datasett som er slik du ønsker. Her er to eksempler:

gapminder\_2 <- gapminder %>%

select(pop, country, year)

gapminder\_2 <- gapminder %>%

select(pop, country, year) %>%

filter(year == 1952,

pop > mean(pop))

**5 Intro til ggplot2**

Det finnes flere ulike måter å plotte på i R. Pakken som vi skal bruke heter ggplot2, og er en del av tidyverse-universet. Vi starter med å laste inn pakken:

#Installerer og laster inn ggplot2:

install.packages("ggplot2")

library(ggplot2)

Alternativt kan du laste inn tidyverse-pakken. Der får du både ggplot2 og dplyr, sammen med en hel del andre pakker fra samme "univers".

#Alternativ:

#install.packages("tidyverse")

#library(tidyverse)

Det første vi skal gjøre nå er å bli litt kjent med hvordan vi bygger plott med ggplot2. Kort fortalt må vi sette sammen flere argumenter, eller deler med informasjon, for at ggplot skal vite hva slags plott vi ønsker å lage. Når jeg går gjennom dette nå så kommer jeg gradvis til å legge på flere lag i plottet. Når dere koder selv så trenger dere ikke kjøre hver kodebit separat, alt kan kjøres samlet på én gang, slik det ser ut i den siste koden vi lager. Nå deler jeg det opp kun for å forklare.

*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 dataene i et “tidy” format, dvs. at hver rad (vannrett) er en observasjon, og hver kolonne er ulike variabler (loddrett). Dette kan dere lese mer om i kapittel 3 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!

Det neste vi gjør 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.

Inne i 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))

Når vi kjører koden, så ser vi at kontinentene er på plass. Hva mangler? Vi må 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()

Vi kan 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", title = “Et plott 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 jeg linker 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 jo 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(data=gapminder, aes(x=continent, y=lifeExp, fill=continent)) +

geom\_boxplot()

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

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

geom\_density()

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

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

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

Hva skjer hvis vi endrer 1) alpha til 1 og 2) size til 0.5?

ggplot(data=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(data=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(data=gapminder,

mapping= aes(x=lifeExp,

y=gdpPercap)) + geom\_point()

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

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

geom\_point() + geom\_smooth()

Skille kontinentene fra hverandre ved bruk av ulike farger:

ggplot(data=gapminder, mapping = 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.

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

geom\_point() + geom\_smooth(colour="black") + facet\_wrap(vars(continent)) +

labs(x = "Forventet levealder",

y = "GDP per capita",

title = "Et plot med Gapminderdata")

**7 Lagring av plott**

Til sist skal vi gå litt raskt gjennom hvordan man kan lagre plottene sine. Underveis i arbeidet, så kan du lagre plottene dine som objekter i environment:

gm\_h <- 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:

gm\_h + labs(title = "Et plot med Gapminderdata")

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.

ggsave(filename = "gdplevealder.png")

eller i pdf-format:

ggsave(filename = "gdplevealder.pdf")

Du trenger egentlig ikke skrive "filename", så lenge du bruker ggsave.

ggsave("gdplevealder.png")

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)