4. Databehandling

September 10, 2024

0.1 Databehandling

Det er mange forskjellige pakker og tilnærminger til databehandling i R. I dette kurset vil vi ha et spesielt fokus på `tidyverse`. Tidyverse er bygget på prinsippet om "tidy data", hvor hver variabel danner en kolonne, hver observasjon/enhet danner en rad, og hver celle i datasettet inneholder en enkelt verdi.

> `%>%` er en "pipe"-operator i Tidyverse som brukes til å kjede sammen funksjoner på en lesbar måte. Den lar deg sende resultatet av en funksjon som input til neste funksjon uten å måtte lagre midlertidige variabler eller lage innfløkte nestede funksjonskall. Dette gjør koden mer lesbar og lettere å forstå.

[ ]:
```
library(tidyverse)
```

0.1.1 Endre variabelnavn (`rename`)

- `rename()`: funksjon for å endre navn på valgte variabler. Først oppgis det nye navnet og deretter det opprinnelige navnet (separert med `=`)
- `rename_all()`: funksjon for å gjøre endringer på alle kolonnenavn. Ettersom R er case-sensitivt (skille mellom små og store bokstaver) kan det være lurt å endre alle kolonnenavn enten til små bokstaver (med funksjonen `tolower`) eller store bokstaver (med `toupper`).

[ ]:
```
befolkning_per_fylke <- arrow::read_parquet("../data/befolkning_per_fylke.
↪parquet")
```

[ ]:
```
colnames(befolkning_per_fylke)
```

[ ]:
```
befolkning_per_fylke %>%
  rename(fylkesnummer = Region,
         personer = value)
```

[ ]:
```
befolkning_per_fylke <- befolkning_per_fylke %>%
  rename_all(tolower) %>%
  rename(fylkesnummer = region,
         personer = value)
```

0.1.2 Selektering av kolonner (`select`)

- `select()`: funksjon for å selektere ønskede variabler. Om man heller ønsker å kun oppgi hvilke variabler som skal utelates settes tegnet `-` foran kolonnenavnet
- `all_of()`: funksjon for å tillate oppramsing av kolonnenavn i en vektor i `select()`
- `any_of()`: tilsvarende som `all_of()`, men her får man ikke feilmelding dersom ikke alle variablene finnes i datasettet

[ ]:
```
befolkning_per_fylke %>%
  select(fylkesnummer, personer)
```

[ ]:
```
befolkning_per_fylke %>%
  select(-tid)
```

[ ]:
```
kolonner <- c("fylkesnummer", "personer")
kolonner
```

[ ]:
```
befolkning_per_fylke <- befolkning_per_fylke %>%
  select(all_of(kolonner))

befolkning_per_fylke
```

[ ]:
```
kolonner <- c("fylkesnummer", "personer", "personer2")

befolkning_per_fylke %>%
  select(any_of(kolonner))
```

0.1.3 Filtrering av rader (`filter`)

- `filter()`: beholder eller fjerner rader etter betingelser på én eller flere kolonner (`TRUE`/`FALSE`). Det er mulig å kombinere flere betingelser med `&` (og) eller `|` (eller). Se liste over vanlige betingelser tidligere i kursmateriellet (under "Boolske verdier")

[ ]:
```
oslo <- befolkning_per_fylke %>%
  filter(fylkesnummer == "03")
```

[ ]:
```
befolkning_per_fylke %>%
  filter(fylkesnummer %in% c("03", "34"))
```

[ ]:
```
befolkning_per_fylke %>%
  filter(personer > 500000)
```

[ ]:
```
befolkning_per_fylke %>%
  filter(fylkesnummer != "03" & personer > 500000)
```

For å filtrere rader med missing-verdier (`NA`) kan man bruke funksjonen `is.na()`. F.eks. `filter(is.na(variabel))` for å beholde kun rader med missing eller `filter(!is.na(variabel))` for å fjerne alle rader med missing.

```
[ ]: befolkning_per_fylke %>%
       filter(is.na(personer) | is.na(fylkesnummer))
```

```
[ ]: befolkning_per_fylke <- befolkning_per_fylke %>%
       filter(!is.na(personer))
```

0.1.4 Lage nye eller endre eksisterende variabler (`mutate`)

- `mutate()`: opprett ny variabel (ved å oppgi et kolonnenavn som ikke finnes fra før) eller endre en eksisterende variabel (ved å oppgi et kolonnenavn som finnes fra før). Det er mulig å kombinere den med andre funksjoner for å transformere eller analysere dataene dine samtidig som du oppretter nye variabler.
- `case_when()`: brukes inne i funksjonen `mutate()` til å lage nye variabler basert på flere betingelser. Den fungerer som en slags `if-else`-konstruksjon, der du kan spesifisere flere betingelser (vilkår), og hva som skal skje hvis hver av dem er sanne. Tegnet `~` (tilde) brukes for å skille mellom en betingelse og verdien som skal returneres hvis betingelsen er sann.

```
[ ]: befolkning_per_fylke <- befolkning_per_fylke %>%
       mutate(aargang = 2024)
```

```
[ ]: befolkning_per_fylke %>%
       mutate(aargang = 2024,
              tall = 1,
              aargang_t1 = aargang-tall)
```

```
[ ]: case_when(2024 == 2024 ~ "Året er 2024",
                TRUE ~ "Året er ikke 2024")
```

```
[ ]: befolkning_per_fylke %>%
       mutate(size = case_when(personer < 500000 ~ "Lavere enn 500 000 personer",
                               personer >= 500000 ~ "Høyere enn eller lik 500 000␣
     ↪personer"))
```

```
[ ]: befolkning_per_fylke <- befolkning_per_fylke %>%
       mutate(landsdel = case_when(fylkesnummer %in% c("03", "31", "32", "33") ~␣
     ↪"Oslo og Viken",
                                   fylkesnummer %in% c("34") ~ "Innlandet",
                                   fylkesnummer %in% c("39", "40", "42") ~ "Agder og␣
     ↪Sør-Østlandet",
                                   fylkesnummer %in% c("11", "15", "46") ~␣
     ↪"Vestlandet",
                                   fylkesnummer %in% c("50") ~ "Trøndelag",
                                   fylkesnummer %in% c("18", "55", "56") ~␣
     ↪"Nord-Norge",
                                   TRUE ~ "Uoppgitt"
       ))
```

```
befolkning_per_fylke
```

0.1.5 Aggregering og gruppering av data (`group_by` og `summarise`)

- `group_by()`: funksjon for å gruppere et datasett etter én eller flere variabler. Denne brukes i kombinasjon med f.eks. `summarise()` for å gjøre beregninger per undergrupper i et datasett.
- `summarise()`: funksjon for å gjøre én eller flere beregninger på et datasett. Uten `group_by()` blir resultatet for hele datasettet, mens med `group_by()` blir resultatene gruppert etter valgte variabler. Funksjoner som kan brukes i `summarise()` er bl.a. `sum()`, `mean()`, `median()`, `min()` og `max()`
- `slice_min()`: : hent ut minimumsverdi på en variabel (per gruppe)
- `slice_max()`: hent ut maksimumsverdi på en variabel (per gruppe)
- `arrange()` brukes til å sortere data etter en eller flere kolonner, og kan kombineres med `group_by()` hvis du ønsker å sortere innen hver gruppe.
- `ungroup()`: Etter at du har brukt `group_by()`, kan det noen ganger være nødvendig å "frigjøre" dataene fra grupperingen ved å bruke `ungroup()` for å fortsette analysen uten gruppeinndeling.

[ ]:
```
befolkning_per_landsdel <- befolkning_per_fylke %>%
  group_by(landsdel) %>%
  summarise(personer_sum = sum(personer))
```

[ ]:
```
befolkning_per_fylke %>%
  group_by(landsdel) %>%
  summarise(personer_sum = sum(personer),
            personer_mean = mean(personer))
```

0.1.6 Dublettsjekk

- `tally()`: tell opp antall observasjoner i hver gruppe
- `count()`: er en kombinasjon av `group_by()` og `tally()`.

[ ]:
```
befolkning_per_fylke_2 <- befolkning_per_fylke %>%
  add_row(befolkning_per_fylke[1,]) # Legger til dublett
```

[ ]:
```
befolkning_per_fylke_2 %>%
  group_by(fylkesnummer, aargang) %>%
  tally() %>%
  filter(n > 1)
```

> Dersom du har dubletter i dataene dine anbefales det å prøve å finne ut hvordan disse har oppstått og enten fjerne de tidligere i prosessen eller bruke logiske filtre for å fjerne disse. Det er mulig å bruke funksjonen `distinct()` med argumentet `.keep_all = TRUE` for å kun beholde unike verdier etter valgte grupperingsvariabler, men her kan det være tilfeldig hvilke verdier som fjernes per gruppe. Bruk derfor denne kun om du vil fjerne helt identiske verdier (der det ikke spiller noen rolle hvilken av dublettene som fjernes)

0.1.7 Koble og binde sammen data

Når man kobler sammen to datasett må disse kobles etter én eller flere koblingsnøkler (kolonner). Disse spesifiseres i argumentet `by =` og det er mulig å oppgi en vektor med flere kolonnenavn (f.eks. `c("variabel1", "variabel2")`). Koblingsnøklene bør ha samme navn i begge datasettene når man kobler, men det er mulig å koble ulike kolonnenavn ved å oppgi begge, f.eks. `c("variabel1_datasett1" = "variabel1_datasett2")`

- `left_join()`: kobling der alle observasjoner fra "venstre" datasett beholdes
- `right_join()`: kobling der alle observasjoner fra "høyre" datasett beholdes
- `full_join()`: kobling der alle observasjoner fra "venstre" og "høyre" datasett beholdes
- `inner_join()`: kobling der kun observasjoner som finnes i både i "venstre" og "høyre" datasett beholdes
- `bind_rows()`: bind sammen datasett radvis. Bør kun brukes på datasett med med helt lik struktur (f.eks. ulike årganger av samme fil)

[ ]:
```
fylkesinndeling <- readxl::read_excel("../data/fylkesinndeling.xlsx")
```

[ ]:
```
befolkning_per_fylke %>%
  left_join(fylkesinndeling, by = "fylkesnummer")
```

[ ]:
```
befolkning_per_fylke %>%
  right_join(fylkesinndeling, by = "fylkesnummer")
```

[ ]:
```
befolkning_per_fylke %>%
  full_join(fylkesinndeling, by = "fylkesnummer")
```

[ ]:
```
befolkning_per_fylke %>%
  full_join(fylkesinndeling, by = "fylkesnummer") %>%
  filter(!is.na(personer))
```

[ ]:
```
befolkning_per_fylke %>%
  inner_join(fylkesinndeling, by = "fylkesnummer") %>%
  mutate(fylkesnavn = case_when(fylkesnummer == "18" ~ "Nordland",
                                fylkesnummer == "50" ~ "Trøndelag",
                                fylkesnummer == "55" ~ "Troms",
                                fylkesnummer == "56" ~ "Finnmark",
                                TRUE ~ fylkesnavn)) %>%
  select(fylkesnummer, fylkesnavn, everything())
```

[ ]:
```
befolkning_per_fylke <- arrow::read_parquet("../data/befolkning_per_fylke.
↪parquet")
befolkning_per_fylke_t1 <- arrow::read_parquet("../data/befolkning_per_fylke_t1.
↪parquet")
```

[ ]:
```
befolkning_per_fylke_alle <- bind_rows(befolkning_per_fylke,␣
↪befolkning_per_fylke_t1)
```

0.1.8 Pivotere datasett (`pivot_wider` og `pivot_longer`)

- `pivot_wider()`: pivotere fra "langt" (long) til "bredt" (wide) format. Må spesifisere hvilke kolonner som skal forbli i langt format (`id_cols`), hvilke grupperingsvariabler som skal bli til nye kolonner (`names_from`) og hvilken variabel verdiene i de nye kolonnene skal hentes fra (`values_from`)
- `pivot_longer()`: pivotere fra "bredt" (wide) til "langt" (long) format. Må spesifisere hvilke kolonner som skal slås sammen til langt format (`cols`), navnet på den nye grupperingskolonnen (`names_to`) og navnet til den nye verdikolonnen (`values_to`)

[ ]:
```
befolkning_per_kjonn_fylke <- arrow::read_parquet("../data/
↪befolkning_per_kjonn_fylke.parquet")

head(befolkning_per_kjonn_fylke)
```

[ ]:
```
befolkning_per_kjonn_fylke_wide <- befolkning_per_kjonn_fylke %>%
  pivot_wider(id_cols  = "Region",
              names_from = "Kjonn",
              values_from = "personer")

befolkning_per_kjonn_fylke_wide
```

[ ]:
```
befolkning_per_kjonn_fylke_wide <- befolkning_per_kjonn_fylke_wide %>%
  mutate(Totalt = Menn+Kvinner) %>%
  filter(Kvinner > Menn)
```

[ ]:
```
befolkning_per_kjonn_fylke_wide %>%
  pivot_longer(cols = c("Menn", "Kvinner", "Totalt"),
               names_to = "Kjønn",
               values_to = "personer")
```