

3. Laste inn og lagre data

September 10, 2024

0.1 Laste inn og lagre data

Det er mulig å laste inn data fra mange forskjellige kilder og formater i R. I dette kurset skal vi se på noen av de vanligste formatene som brukes i SSB: Excel-filer (`.csv` og `.xlsx`), SAS-filer (`.sas7bdat`) og Parquet-filer (`.parquet`).

OBS: Når vi bruker GitHub til å lagre og dele systemene våre er det viktig å passe på at sensitive data ikke legges på GitHub. Generelt skal ikke data legges på GitHub, men for å kunne lese data i dette kurset er det lagt ut noen filer som ikke inneholder sensitive data.

Når man leser inn data i R lagres disse som objekter som gis et valgfritt navn. Disse objektene brukes videre i databehandling uten at den opprinnelige filen som ble lest inn blir endret. For å skrive de ferdig behandlede dataene tilbake til en fil gjøres dette eksplisitt med egne funksjoner.

Working directory Et working directory i R refererer til den gjeldende mappen eller katalogen der R vil se etter filer, og der R vil lagre filer som standard når du kjører kommandoer som leser fra eller skriver til filer. Det er med andre ord den “arbeidsmappen” som R bruker som utgangspunkt for filoperasjoner. Hvis working directory er satt til “/ssb/bruker/initialer”, og du vil lese en CSV-fil som heter data.csv, kan du bare skrive “data.csv” når du skal oppgi filstien.

- `getwd()`: se hvilken mappe som er satt som gjeldende working directory
- `setwd()`: hvis du ønsker å endre til en annen mappe, kan du bruke `setwd()` og oppgi stien til den ønskede mappen

```
[ ]: getwd()
      setwd("./presentasjoner") # kjør kun i RStudio, i Jupyter er dette allerede
      ↪working directory
```

Her settes working directory til mappen `presentasjoner` som ligger i Github-repoet til kursmateriellet. Dette er for å gjøre de relative filstiene som brukes nedenfor like når man jobber i Jupyterlab eller RStudio. En relativ filsti refererer til en fil eller mappe i forhold til det nåværende working directory i R. Den beskriver hvordan du kommer til en fil fra det nåværende stedet du jobber (working directory), i stedet for å gi den fulle stien fra roten av filsystemet.

- `.` (punktum): Representerer den nåværende mappen (working directory).
- `..` (to punktum): Representerer mappen over den nåværende mappen, altså en mappe “høyere opp” i hierarkiet.

Filstier Fullstendig filsti er den komplette banen fra roten av filsystemet til en fil. Når du bruker en fullstendig sti, spiller det ingen rolle hvilket working directory du har satt – du spesifiserer den

eksakte plasseringen av filen på datamaskinen din. Bruk fullstendige stier hvis du arbeider med filer på forskjellige steder i filsystemet, eller hvis du ønsker å være helt sikker på hvor filene er lagret, uavhengig av hva working directory er satt til.

I SSB har vi kun tilgang på data som ligger på Linux fra Jupyterlab og RStudio i produksjonssonen. Et eksempel på en fullstendig filsti på Linux er:
`/ssb/bruker/felles/fredagskoding/T07459.sas7bdat`

0.1.1 CSV

CSV-filer (Comma Separated Values) er en type tekstfil som brukes til å lagre tabulære data, som oftest fra regneark eller databaser. Hver linje i filen representerer en rad i tabellen, og verdiene i hver rad er adskilt med et komma (eller noen ganger et annet skilletegn som semikolon eller tabulator). Den første linjen i en CSV-fil inneholder ofte kolonnenavnene. De viktigste egenskapene man må vite om filen man leser inn er hvilken separator (`sep =`) som brukes (f.eks. `;`, `,`), hvilket desimaltegn (`dec =`) som brukes (f.eks. `,`, `.`) og hvilket tegnsatt (`encoding =`) som brukes (f.eks. UTF-8, latin1).

- `read.csv()`: funksjon som leser inn .csv-filer som en data frame
- `write.csv2()`: brukes til å lagre en data frame til en semikolonseparert .csv-fil

```
[ ]: sykemelding <- read.csv("../data/sykemelding.csv", sep = ";", dec = ",",  
  ↪ encoding = "UTF-8")
```

```
[ ]: kommunedata <- read.csv("../data/kommunedata.csv", sep = ",")
```

```
[ ]: # write.csv2(kommunedata, "../data/kommunedata_ny.csv", row.names = FALSE)
```

0.1.2 XLSX

- `read_excel()`: funksjon som leser inn .xlsx-filer som en data frame
- `write.xlsx()`: funksjon for å lagre en data frame til som en .xlsx-fil

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

```
[ ]: # openxlsx::write.xlsx(fylkesinndeling, file = "../data/fylkesinndeling_ny.  
  ↪xlsx",  
#           rowNames = FALSE,  
#           showNA = FALSE,  
#           overwrite=T)
```

0.1.3 SAS

Den mest brukte pakken for å lese SAS-filer i R er `haven`. Denne pakken er en del av tidyverse-familien og gir en enkel måte å importere data fra SAS, Stata, og SPSS. Det anbefales ikke å lagre objekter i R som SAS-filer, men dersom dette må gjøres anbefales det å lagre en .csv-fil som kan leses inn i SAS.

- `read_sas()`: funksjon som leser inn SAS-filer som en data frame

```
[ ]: trygd <- haven::read_sas("../data/trygd.sas7bdat")
```

0.1.4 Parquet

Parquet-filer er en kolonneorientert lagringsfilformat som er optimalisert for rask lesing og lagring av store datamengder.

- `read_parquet()`: funksjon som leser inn .parquet-filer som en data frame
- `write_parquet()`: funksjon for å lagre en data frame som en .parquet-fil

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

```
[ ]: # arrow::write_parquet(befolkning_per_fylke, "../data_ny/  
↪befolkning_per_fylke_ny.parquet")
```

0.1.5 Data fra API-er

I dette kurset vil det ikke gjennomgås hvordan å laste ned data fra API-er, men her er det lagt til eksempler på hvordan man laster ned data fra statistikkbanken og KLASS.

Statistikkbanken For å lese inn data direkte fra API-et til SSBs Statistikkbank kan man bruke pakken `PxWebApiData`.

I eksempelet nedenfor leses tabellen [07459: Alders- og kjønnsfordeling i kommuner, fylker og hele landets befolkning](#) inn.

```
[ ]: befolkning <- PxWebApiData::ApiData(07459,  
                                         ContentsCode = T,  
                                         Region = T,  
                                         Kjonn = T,  
                                         Alder = T,  
                                         Tid = "2024")[[2]]
```

KLASS Data fra KLASS API kan leses direkte inn i R med pakken `klassR`. Klass er SSBs system for dokumentasjon av kodeverk (klassifikasjoner og kodelister). Klassifikasjoner er «offisielle» kodeverk, og alle klassifikasjoner i Klass har navn som begynner med «Standard for..», f.eks. Standard for sivilstand. I en klassifikasjon skal kategoriene være gjensidig utelukkende og uttømmende, dvs. at klassifikasjonen inneholder alle kategorier som tilhører området klassifikasjonen dekker.

I eksempelet nedefor lastes kodelisten [Standard for fylkesinndeling](#) inn:

```
[ ]: fylkesinndeling <- klassR::GetKlass(104, date = "2024-01-01")
```