Oppgavesett for innførende kurs i microdata.no

Mål:

Bli kjent med de grunnleggende deler av microdata.no, med fokus på datasettoppbygging og deskriptiv statistikk. Til slutt prøver vi oss på en enkel regresjonsanalyse.

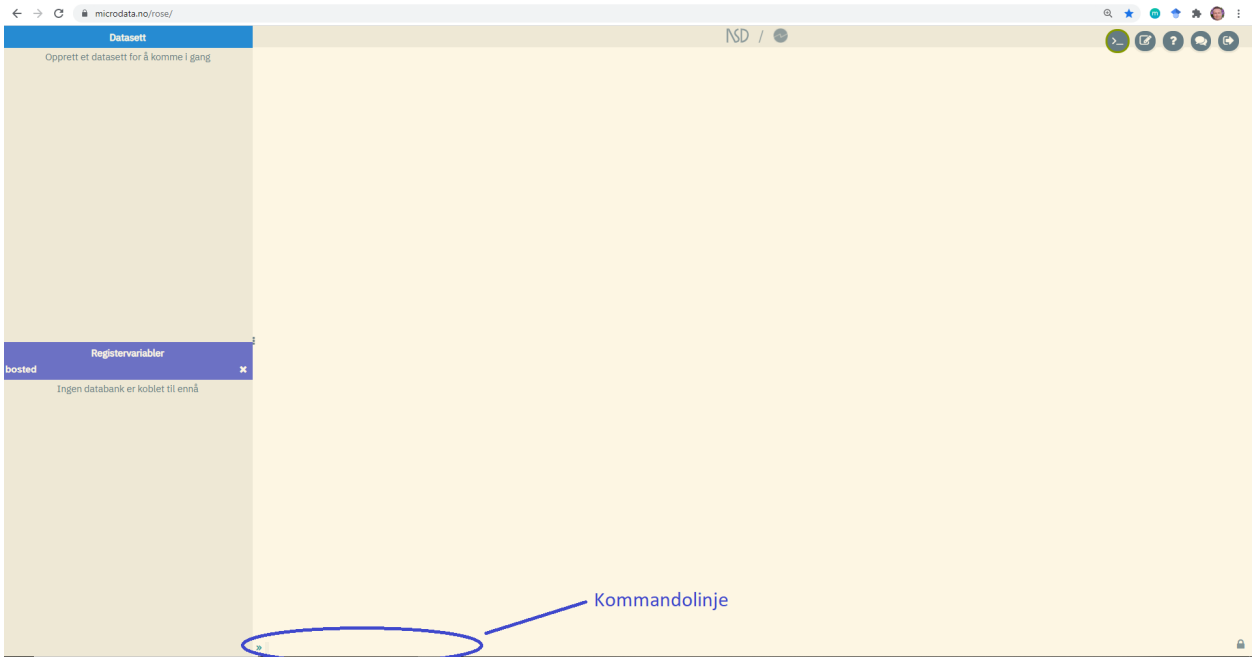
Ekstraoppgave (11): Uthenting av forløpsopplysninger (datasett med flere observasjoner per enhet)

Innlogging/oppstart av microdata.no:

1. Skriv inn webadressen “microdata.no” i url-feltet i din nettleser. Bruk Chrome eller Firefox, evt. andre nettlesere bortsett fra Internet Explorer.
2. Hold musepeker over “Logg inn”-knappen og velg “Analyseverktøyet”
3. Velg innloggingsalternativ og logg inn på samme måte som ved en nettbank - vanligste er BankID på mobil (ha mobil tilgjengelig)

Vi starter med å arbeide i analyseområdet i microdata.no. Dette er det området som møter deg når du har logget inn. Merk at før du har begynt å jobbe, er innholdet i vinduene tomme.

Helt nederst finner du kommandolinjen. Det er her en legger inn kommandoene en ønsker å eksekvere. Oppgave 7 i oppgavesettet går ut på å lagre alt arbeidet ditt som et skript. Dette gjør det mulig å lagre, systematisere, redigere og kjøre sett med kommandoer automatisk.



<u>Nyttige tips:</u>

Kommandoen `help` viser en oversikt over tilgjengelige kommandoer. `help` i kombinasjon med kommandonavn viser dessuten nyttig informasjon om den aktuelle kommando.

Tastekombinasjonen <ctrl> + <z> kan brukes til å angre og gå ett steg tilbake. Så om du gjør en feil, f.eks. koder en variabel på feil måte, kan du gjøre om på dette ved å gå ett trinn tilbake og kode på nytt. En kan i prinsippet klikke <ctrl> + <z> mange ganger og angre alt en har gjort.

Tasten “pil opp” på tastaturet gjør det mulig å hente frem tidligere brukte kommandoer. Disse kan så justeres på og kjøres som nye kommandoer.

For å se hvilket format verdiene til en variabel har, klikk på variabelnavnet i variabellisten for datasettet ditt i øverst til venstre i arbeidsområdet. Der finner du også kode-oversikt og enkel statistikk for den aktuelle variabelen. Det samme kan du gjøre i variabellisten nede til venstre. Denne inneholder alle tilgjengelige variabler i databasen (krever at du først bruker kommandoen `require` for å koble deg til databasen).

Variabler kan slettes eller døpes om etter behov. En bruker da hhv. kommandoene `drop <variabel>` og `rename <navn_gml> <navn_ny>`

I dette oppgavesettet skal vi se litt på hvordan lønnsinntekt har utviklet seg de siste årene, fordelt på ulike demografiske grupper av befolkningen. Vi skal også se litt nærmere på utvalgte yrker.

Det fokuseres på å bygge opp datasett for analyser og å lage informativ deskriptiv statistikk. Til slutt demonstrerer vi hvordan vi avdekker statistiske sammenhenger mellom variabler, og kjører en enkel regresjonsanalyse for å finne årsakssammenhenger knyttet til lønnsinntekt (hva som er med på å bestemme dette).

Fasit finner du på de siste sidene av oppgavesettet.

1) **Koble deg til databasen med variabler**

   Start alltid med å koble deg til databasen med variabler som du i de neste trinnene vil importere til ditt lokale datasett.

   Per i dag er det kun èn tilgjengelig database: SSBs database. Det vil etterhvert bli mulig å koble seg opp mot databaser som inneholder data også fra andre dataeiere.

   Bruk følgende kommando i kommandolinjen for å koble deg til SSBs database:

   ```
   require no.ssb.fdb:12 as ds
   ```

   (Siste del av syntaxen, "`as ds`" brukes til å lage et kortnavn/alias på databasen. Her står du fritt til å velge andre navn om du ønsker.)

2) **Opprett et tomt datasett**

   Neste trinn er å opprette et lokalt datasett som du skal fylle med variabler fra SSBs database. Bruk følgende kommando, og velg et passende navn på datasettet (erstatt <datasettnavn> med et eget navn):

   ```
   create-dataset <datasettnavn>
   ```

Datasettet ditt vil først være tomt. Når du legger til variabler med kommandoen `import`, vil du se oppe i venstre vindu at datasettet fylles opp med stadig flere variabler, for hver import-kommando du eksekverer.

3) Definer populasjon

Etter å ha opprettet et datasett, er det på tide å definere populasjonen vi skal studere. Dette kan også gjøres senere, men det anbefales å gjøre dette så tidlig som mulig siden arbeidet med datatilrettelegging og analyser da blir mindre ressurs-/tidkrevende (jo mindre datasett/populasjon, jo raskere blir responsen).

Populasjoner defineres gjennom å:

- Hente variabler som brukes som populasjonskriterier
- Droppe enheter (individer) som faller utenfor kriteriene

Kommandoen `import` brukes til å hente variabler inn i ditt datasett fra databasen. Variabelopplysningene kobles automatisk opp mot respektive enheter (individer) i din populasjon vha. et innebygget identifikasjonsnummer, og dataene organiseres på tverrsnittsformat (én record per enhet):

`import <databasealias>/<variabelnavn> <YYYY-MM-DD> [as <alias>]`

<databasealias> erstattes med kortnavnet du har laget på SSBs database ("ds").
<variabelnavn> erstattes med navnet på variabelen du skal importere (husk store bokstaver), og <*YYYY-MM-DD*> med måletidspunktet, f.eks. 2019-12-31. Det er hensiktsmessig å bruke egne navn på variabler som importeres. Dette gjøres ved å legge inn "as" pluss et valgfritt navn til slutt i uttrykket.

Ved import av variabler med faste opplysninger (kjønn, fødeland, fødselsdato etc), skal ikke måletidspunkt angis.

Eksempel 1: `import ds/NUDB_BU 2019-07-31 as utd`
Eksempel 2: `import ds/BEFOLKNING_FOEDSELS_AAR_MND as faarmnd`

a) Det er lurt å starte med en universell variabel, siden systemet benytter en left-join-logikk: Første variabel definerer din populasjon (som du senere kan trimme ned til ønsket størrelse).

   Start med å importere variabelen BEFOLKNING_FOEDSELS_AAR_MND (fødselsdato på det numeriske formatet *YYYYMM*) til datasettet ditt. Bruk

kommandoen `import`. Dette er en fast opplysning, så du skal ikke angi måletidspunkt.

Etter at import-kommandoen er kjørt, kan du se i vinduet øverst til venstre at datasettet består av fødselsdato-variabelen pluss en individ-nøkkelvariabel. Ved å klikke på variabelnavnet dukker det opp en boks med metadata for denne variabelen.

Variabler med symbolet “123” foran har numerisk format, mens “abc” betyr at variabelen har tekstformat (alfanumerisk).

I samme boks vil du også se antallet records i datasettet ditt (bestemmes av populasjonen til den første importerte variabelen).

Du har nå laget et datasett som består av hele den norske populasjonen. Gjør deg kjent med populasjonen din ved å studere alderssammensetningen. Til dette trenger du først å lage en ny aldersvariabel basert på variabelen du har importert. Bruk kommandoen `generate` og mål alder per 2019. Variabel-aliaset til fødselsdato-variabelen som du importerte, brukes som input.

Forslag:

```
generate alder = 2019 - int(faarmnd/100)
```

(Det siste leddet i formelen trekker ut de fire første sifrene fra en 6-sifret fødselsdato, altså årstallet. Vi ønsker kun heltallet fra divisjonen. Derfor bruker vi funksjonen `int()`. Differansen mellom det oppgitte årstallet 2019 og fødselsåret blir da alder målt per 2019.)

b) Studèr aldersfordelingen. Bruk kommandoen `histogram`:

```
histogram alder, discrete
```

Opsjonen `discrete` sørger for at alle alderskohorter får hver sin søyle i søylediagrammet. Dette anbefales når en skal bruke histogram til å se på diskrete verdier (og ikke intervaller av kontinuerlige verdier).

Legg merke til at aldersspennet strekker seg langt over 100. Grunnen til dette er at populasjonen din ikke er skikkelig filtrert. Datasettet ditt inneholder nemlig alle personer i databasen som noensinne har levd, inkludert døde personer.

c) Du trenger en variabel som angir om personer er bosatt, død eller emigrert for å fjerne personer du ikke ønsker å inkludere i din analyse. Til dette bruker du variabelen `ds/BEFOLKNING_STATUSKODE`. Denne variabelen er alfanumerisk og har verdiene '1', '3' og '5' som betyr hhv. bosatt, utvandret og død. Merk også at variabelen er en såkalt tverrsnittsvariabel som bare har verdier for en fast dato per år. Det varierer når de ulike tverrsnittsvariablene måles, men i dette tilfellet er det 1/1 som er det faste måletidspunktet (i kommandolinjen foreslås det gyldige datoer i autocomplete-menyen som dukker opp ved inntasting av kommandoen, og du kan dessuten sjekke gyldige måletidspunkter ved å klikke på variabelen i database-listen nederst til venstre, evt. i variabeloversikten du finner via innloggingssiden).

   Importer først variabelen og bruk måletidspunktet 2019-01-01. Bruk deretter kommandoen `keep if` for å ta vare på kun dem som var bosatt i Norge på denne datoen (du fjerner altså personer som døde før 2019-01-01 eller som var utvandret på denne datoen).

   Forslag:

   ```
   import ds/BEFOLKNING_STATUSKODE 2019-01-01 as regstat
   keep if regstat == '1'
   ```

   Bruk histogram-kommandoen på nytt for å se hvordan aldersfordelingen nå ser ut (husk opsjonen `discrete`).

   Du kan også bruke kommandoen `summarize` for å hente ut oppsummerende statistikk for alder (snitt, standardavvik, median etc): `summarize alder`

d) I dette oppgavesettet vil vi studere lønnsinntekt, og da kan det være greit å avgrense populasjonen ytterligere til personer i alderen 31-49 (da er de fleste ferdige med studier og er kommet i jobb).

   Bruk kommandoen `keep if` eller `drop if` i kombinasjon med standard IF-betingelser til å trimme ned datasettet til å inneholde aldersgruppen 31-49.

   Forslag:
   ```
   keep if alder > 30 & alder < 50
   ```

   Se på antallet enheter i datasettet ditt. Det har nå blitt redusert fra 10.626.204 til 1.368.922 individer.

Ved logiske if-uttrykk brukes følgende tegn: > (større enn), < (mindre enn), == (er lik), != (ulik), >= (større enn eller lik), <= (mindre enn eller lik), | (eller), & (og).

Merk at en bruker kun enkel = med mindre det er i sammenheng med en if-betingelse. Eksempel: `generate mann = 1 if kjønn == '1'`

Funksjonen `int()` gjør om tall til heltall (integer).

Funksjonen `substr()` trekker ut delstrenger fra variabler med verdier på tekstformat (alfanumerisk), der en angir hhv. variabelnavn, posisjon for første karakter, og antallet posisjoner som skal avleses. Eksempel på uttrekking av første karakter fra en tekstverdi: `pos1 = substr(varx,1,1)`

Missing angis ved sysmiss(), der en spesifiserer variabelen inne i parentesen, f.eks. "`drop if sysmiss(inntekt)`" angir at enheter med manglende verdi på variabelen `inntekt` skal slettes.

Numeriske variabler: Verdier angis uten fnutter, f.eks. 1, 3, 10000

Alfanumeriske variabler: Verdier har tekstformat og angis med enkeltfnutter, f.eks. '1', '3', '10000' (dobbeltfnutter kan også brukes).

4) Importere analysevariabler (fylle datasettet med variabler)

Du har nå en ferdig definert populasjon. Neste trinn blir å legge til variabler som du trenger for din analyse. Da bruker du kommandoen `import`.

De fleste data i databasen er organisert som forløpsdata, dvs. at variabelverdier oppdateres fortløpende. Valgfrie måletidspunkter angis da når en bruker import-kommandoen.

Faste opplysninger som kjønn, fødeland, fødselsdato etc: Måletidspunkter skal ikke oppgis

Enkelte variabler i databasen, såkalte tverrsnittsvariabler, er hentet fra statistiske kilder og måles kun ved faste datoer per år. Da må en oppgi en spesifikk måledato som blir foreslått av systemet når en prøver å taste inn dato i kommandolinjen. Du finner gyldige måledatoer i systemets variabeloversikter.

En fjerde variant er årlige akkumulerte verdier som viser den samme årsverdien uansett hvilken dato en velger innen et år. Dette gjelder hovedsakelig økonomiske opplysninger som inntekt, formue, gjeld etc. Det er da det samme hvilken dato en velger innenfor et aktuelt år.

Importer følgende variabler ved hjelp av kommandoen `import` (husk også kortnavnet til databasen):

a) BEFOLKNING_KJOENN (kjønn, fast opplysning)
b) BEFOLKNING_FODELAND (fødeland, fast opplysning)
c) NUDB_BU per 2019-07-31 (6-sifret NUS-utdanningskode for høyeste fullførte utdanning, forløpsopplysning)
d) BEFOLKNING_KOMMNR_FAKTISK per 2019-01-01 (4-sifret kommunekode for bosted, tverrsnittsopplysning)
e) INNTEKT_WLONN per 2015-12-31 (årlig lønnsinntekt, akkumulert verdi => oppgitt verdi blir den samme uansett hvilken dato en velger innen det aktuelle året)
f) INNTEKT_WLONN per 2016-12-31
g) INNTEKT_WLONN per 2017-12-31
h) INNTEKT_WLONN per 2018-12-31
i) INNTEKT_WLONN per 2019-12-31

NB! Personer med 0 i lønnsinntekt => missingverdi i databasen. Men dette er greit i denne analysen, siden vi kun er interesserte i å lage statistikk for positive inntekter. Om det er ønskelig å inkludere personer med null i inntekt, gjøres dette på følgende måte:

```
replace lønn = 0 if sysmiss(lønn)
```

5) Kjøre enkel deskriptiv statistikk

Nå når vi har populasjonen og variablene klare, kan vi sette i gang og lage deskriptiv statistikk. En har en rekke hjelpemidler til rådighet.

For å hente ut statistiske tall for numeriske variabler bruker en vanligvis kommandoen summarize. For å hente ut frekvenser/opptellinger fordelt på ulike kategorier brukes kommandoen `tabulate`. En kan også kombinere `tabulate` og `summarize` for å vise statistiske tall fordelt på kategorier (f.eks. `tabulate kjønn, summarize(lønn)`).

Det er også mulig å lage grafiske fremvisninger som kakediagram (`piechart`), søylediagram (`barchart`), histogram (`histogram`), anonymisert plotdiagram (`hexbin`) og flytdiagram (`sankey`).

Deskriptiv statistikk kan enkelt kopieres og eksporteres inn i Excel- eller Google-regneark for videre bearbeiding og laging av egne grafiske fremstillinger. På

denne måten får en dessuten vist alle desimaler, og ikke bare standardfremvisningen i systemet. Grafisk output fra microdata.no kan kun eksporteres som bildefil og kan bare redigeres i bilderedigeringsprogram som “Paint”.

Vi starter med å lage endimensjonal statistikk over lønnsutviklingen for vår populasjon. Bruk kommandoen `summarize` for å vise hvordan lønnsinntekten har endret seg i perioden 2015-2019. Du kan liste opp alle variablene du vil måle i en enkelt kommando.

Forslag:
```
summarize lønn15 lønn16 lønn17 lønn18 lønn19
```

Tallene kan fremstilles grafisk som søylediagram gjennom kommandoen `barchart`. Syntaxen har samme logikk ved at en lister opp variablene på samme måte som for `summarize`. I tillegg må en spesifisere hva slags statistikk en vil vise for søylediagrammene.

Forslag:
```
barchart (mean) lønn15 lønn16 lønn17 lønn18 lønn19
```

I tillegg til gjennomsnitt (mean) kan du også vise antall individer med gyldige verdier (i dette tilfellet antall individer med positive verdier) gjennom å oppgi `count` i stedet for `mean`.

Forslag:
```
barchart (count) lønn15 lønn16 lønn17 lønn18 lønn19
```

Ved å angi `median` får du frem medianverdier i stedet. Prøv også dette!

Studér også lønnsfordelingen i utvalget ditt for 2019. Bruk kommandoen `histogram` og benytt variabelen `lønn19`. Tips: Du kan bruke opsjonen `freq` for å vise antall som måleenhet på y-aksen i stedet for standardvisningen. Opsjonen `normal` kan dessuten brukes til å legge over en standard normalfordeling til referansebruk. Alle opsjoner kan brukes i kombinasjon.

Forslag:
```
histogram lønn19, freq normal
```

6) Kjøre todimensjonal deskriptiv statistikk

Det er mulig å lage statistikk og grafisk fremstilling med flere dimensjoner. Da trenger vi først å tilrettelegge variabler for dette formålet. Vi ønsker å se på hvordan lønn fordeler

seg mellom kjønn, fødeland og utdanningsnivå. Kjønn er det ikke nødvendig å kode om, siden den kun har to kategorier. Men fødeland og utdanningsnivå derimot, trenger vi å lage grove kategorier for. Hvis ikke vil statistikken bli uoversiktlig.

a)
La oss starte med å lage en dummyvariabel for fødeland: Norsk (=1) og ikke-norsk (=0). Kall variabelen "norsk". Pass på å alltid kode dummyvariabler som numeriske verdier (uten fnutter), ellers vil de ikke kunne benyttes i regresjonsanalyser.

Tips:

i) Norsk ⇔ fødeland = '000'
ii) Innvandrerbakgrunn ⇔ øvrige koder inkludert missing (personer med midlertidig opphold)

Ved generering av kategoriske variabler, f.eks. dummyvariabler, er den enkleste metoden å først tilegne alle individer én av verdiene (bruk `generate`), for deretter å bruke `replace` til å endre verdi basert på gitte kriterier slik at klassifiseringen blir riktig. En kan bruke så mange `replace`-kommandoer en ønsker for å komme i mål med hele omkodingen av en gitt variabel, men bare én `generate`-kommando.

Et alternativ til `generate`/`replace` er å bruke den mer avanserte kommandoen `recode`. Merk at sistnevnte krever at variabler er på numerisk format.

Eksempel A:
```
generate mann = 1
replace mann = 0 if kjønn == '2'
```

Eksempel B:
```
generate norsk = 0
replace norsk = 1 if land == ‘000’
```

Eksempel C:
```
destring kjønn
recode kjønn (2 = 0)
rename kjønn mann
```

Bruk kommandoen `tabulate` til å sjekke kodingen av dummyvariabelen "norsk". Bruk også opsjonen `cellpct` for prosentvis visning.

Prøv også kommandoen `piechart` for å lage et kakediagram over fordelingen norsk vs ikke-norsk.

b)

En enkelt måte å grovinndele utdanningsnivå på, er å trekke ut første siffer i den standardiserte NUS-koden (hierarkisk kode som går fra 0 (lavest) til 8 (høyest).

Lag grovinndeling av utdanningsnivå (1. siffer-nivå):

Trekk ut første siffer i den 6-sifrede utdanningskoden ved å bruke funksjonen `substr()`.

Tips: `substr(utd,1,1)` trekker ut første siffer fra den alfanumeriske variabelen `utd`. Inputparameter nr. 2 angir startposisjon, mens parameter nr. 3 angir hvor mange posisjoner som skal avleses.

Bruk kommandoen `tabulate` for å vise kodingen av utdanningsnivå. Bruk gjerne opsjonen `cellpct`.

Gjør variabelen med den grove utdanningsinndelingen om til numerisk format slik at den kan brukes til rangeringer og i matematiske uttrykk. Bruk kommandoen `destring` til dette formålet, etterfulgt av navnet på variabelen du ønsker å gjøre numerisk.

En kan lage alternative inndelinger av utdanningsnivå, i stedet for å bruke standard 1. siffer NUS-inndeling (= groveste NUS-nivå). Om en f.eks. ønsker 3 kategorier, kan en gjøre det slik:

```
import NUDB_BU 2011-06-01 as utd
generate utdnivå = substr(utd,1,1)
destring utdnivå, force
replace utdnivå = 1 if utdnivå <= 3
replace utdnivå = 2 if utdnivå == 4 | utdnivå == 5
replace utdnivå = 3 if utdnivå > 5
```

Merk at `NUDB_BU` er alfanumerisk. For å gjøre det lettere å omkode, brukes destring-kommandoen for å konverte verdiene til numerisk format (force-opsjonen brukes for å sikre at eventuelle verdier som inneholder andre tegn enn tall får verdien sysmiss). Da kan en bruke <, >, <=, >= etc, noe som ikke er mulig for alfanumeriske verdier. Husk at ledende 0-er blir fjernet ved numerisk konvertering. F.eks. '0301' blir til 301 ved konvertering til numerisk format.

Alternativt kan en bruke kommandoen recode i stedet for generate/replace:

```
import NUDB_BU 2011-06-01 as utd
generate utdnivå = substr(utd,1,1)
destring utdnivå, force
recode utdnivå (0/3 = 1)(4/5 = 2)(6/8 = 3)
```

Den siste linjen kan alternativt uttrykkes slik:

```
recode utdnivå (0 1 2 3 = 1)(4 5 = 2)(6 7 8 = 3)
```

Nå når vi har variablene klare, kan vi sette i gang med å lage flerdimensjonal statistikk.

For visning av frekvenser/antall fordelt på ulike kategorier, bruker en `tabulate` der en bare legger til flere enn én variabel i uttrykket. Jo flere variabler, jo flere dimensjoner. To variabler gir en toveis-tabell etc. Dette kan igjen kombineres med bruk av `summarize` som opsjon for visning av statistiske målinger i stedet for antall, f.eks. `tabulate kjønn sivilstand, summarize(lønn).`

c)
Start med å lage oppsummerende statistikk over lønnsinntekt for årene 2015-2019, men bare for dem med utdanningsnivå < 2. Bruk kommandoen `summarize`.

Gjør det samme, men denne gang for dem med utdanningsnivå > 6.

Lag deretter en tabell som viser gjennomsnittlig lønnsinntekt for 2019 fordelt på kjønn.

Forslag:
```
tabulate kjønn, summarize(lønn19)
```

Gjør det samme, men fordel på hhv. fødeland (norsk) og utdanningsnivå (1-sifret) i stedet for kjønn.

Lag et søylediagram som viser gjennomsnittlig lønnsinntekt over årene 2015-2019, fordelt på kjønn. Bruk kommandoen `barchart` med opsjonen `over()`.

Forslag:
```
barchart (mean) lønn15 lønn16 lønn17 lønn18 lønn19, over(kjønn)
```

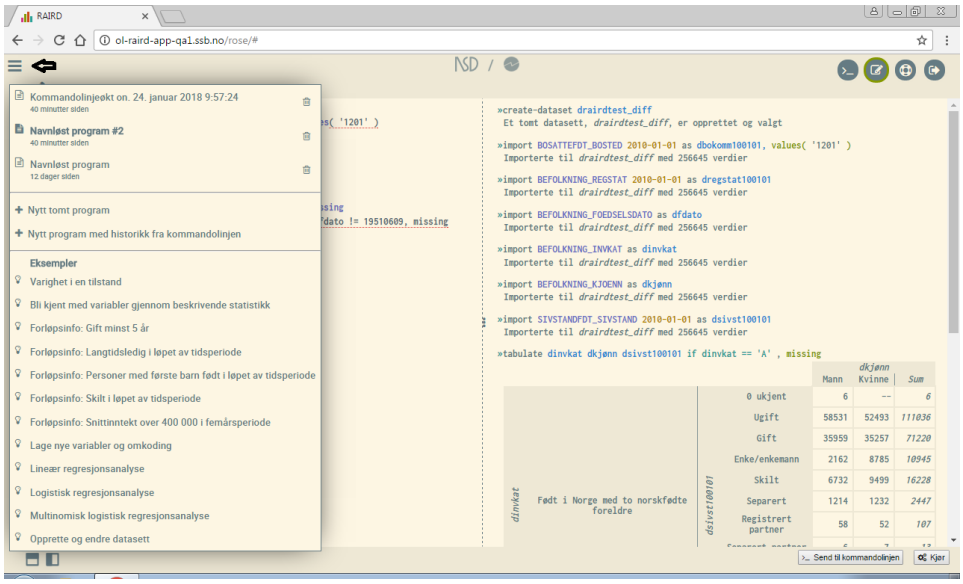
Gjør det samme, men fordel på hhv. fødeland (norsk) og utdanningsnivå.

7) Ta vare på arbeidet du har gjort ved å lagre det som et skript

Et skript kan brukes til å foreta diverse justeringer i analysen. Alle kommandolinjene kan deretter kjøres på nytt i en bolk.

a) Klikk på skript-knappen (knapp nr. 2 oppe i høyre hjørne) for å gå inn i skript-området
b) Klikk på menyknappen helt opp til venstre

c) Bla deg ned til og klikke på “Nytt program med historikk fra kommandolinjen”



d) Alt arbeidet ditt er nå lagt inn i et skript.
e) Legg inn et navn på skriptet ditt i feltet like over skriptområdet. Skriptet ditt vil nå være lagret med dette navnet og kan finnes igjen ved å klikke på menyknappen oppe til venstre.
f) Alternativt til a)-e) kan du bare skrive `save '.......'` i kommandolinjen, der du fyller inn et passende navn på skriptet inni enkelt-fnuttene. Deretter går du inn i skriptvinduet og jobber videre med skriptet ditt.
g) Du kan nå kjøre arbeidet ditt samlet ved å klikke på “Kjør” eller “Send til kommandolinjen”, eventuelt gjøre justeringer først og deretter kjøre på nytt. Så lenge skriptet ditt har et definert navn, vil det bli autolagret når du jobber i det, på samme måte som i Google Docs.

   Merk: Systemet husker tidligere kjørte kommandosekvenser. Så om du kjører samme skript to ganger på rad, så vil andre kjøring bare hente frem forrige resultat. Om du justerer på noe eller legger nye linjer til på slutten av skriptet, hentes de kjørte sekvenser frem fra minnet, og bare det nye kjøres. Endres derimot linjer helt i starten av et skript, anses alle sekvensene som nye, og alt kjøres på nytt igjen.

   NB! Når et skript kjøres ved å bruke knappen “Send til kommandolinjen”, sendes alt til arbeidsområdet, og det som måtte ligge der fra før vil bli overskrevet av den

nye kjøringen. Pass derfor på at arbeidet ditt i arbeidsområdet alltid blir tatt vare på via skript.

NB! En kan sikkerhetskopiere ved å kopiere skript over i egne dokumenter. Om en har tenkt å lime innholdet inn i et nytt skript etterpå, må en passe på å benytte et rent tekstformat via programmer som “Notisblokk” o.l. (tekst som kopieres inn i skripteditoren direkte fra programmer som Word o.l. kan skape krøll siden teksten da inneholder diverse formateringer som kan skape problemer ved kjøring av skript).

Er du i mål med oppgave 7? Da kan du prøve de litt mer avanserte oppgavene 8-11.

Du kan velge selv om du vil jobbe videre i skripteditoren eller i kommandolinjen i de neste oppgaver. Det anbefales å jobbe i skripteditoren og kjøre kommandoene derfra pga. muligheten for å lagre og systematisere arbeidet ditt. Editoren kan blant annet brukes til å kopiere og lime slik at en slipper å benytte så mye manuell programmering. Kommandolinjen derimot passer best for eksplorativt arbeid og for å gjøre seg kjent med variabler og syntax.

8) Lage finraffinert statistikk: Se nærmere på utvalgte yrkesgrupper

Nå som vi behersker deskriptiv statistikk, kan vi gå et steg videre og se på hvordan lønnsinntekt varierer over yrkesgrupper.

a)
Til dette trenger vi først å importere en variabel som angir yrkeskoder (4-sifret alfanumerisk kode): REGSYS_ARB_YRKE_STYRK08.

Yrkesvariabelen er en såkalt tverrsnittsvariabel som måles per 16/11 hvert år. Importer denne inn i datasettet ditt og bruk måledatoen 2019-11-16. Passende alias kan være `yrke19`.

Yrkesvariabelen har mange koder, så det blir for detaljert for vårt formål. Vi velger derfor å trekke ut interessante yrker vi vil se på:

- Ledere (koder som starter med ‘1’)
- Allmennleger (‘2211’)
- Legespesialister (‘2212’)

- Sykepleiere (‘2223’)
- Lærere (koder som starter med ‘23’)
- IT-utviklere (koder som starter med ‘25’)
- Landbruksyrker (koder som starter med ‘61’)

I tillegg må vi angi en restgruppe som inneholder øvrige yrker, samt en gruppe for personer som ikke er i jobb.

Bruk kommandoene `generate` og `replace` til å lage en ny variabel som består av en grovinndeling med de utvalgte yrkene listet opp ovenfor. Kall variabelen `yrkesgr`.

Tips:
- Begynn med kommandoen `generate` og sett alle verdier for den nye variabelen `yrkesgr` til 9 (restgruppe)
- I neste linje bruker du `replace` til å erstatte med verdien 1 i de tilfeller hvor `yrke19` starter med ‘1’ (ledere)
- Gjenta prosessen for hver yrkeskode du vil trekke ut og erstatt med stadig høyere koder helt til du kommer til 7
- Til slutt erstatter du verdiene for `yrkesgr` med 999 dersom `yrke19` har missingverdi (`replace yrkesgr = 999 if sysmiss(yrke19)`)

(Om dette blir for vanskelig, er det lov å titte litt i fasiten bakerst i oppgavesettet)

b)
Verdi-labler kan brukes til å vise hva de ulike verdiene betyr, slik at statistikken blir mer lesbar og forståelig. Til dette brukes kommandoene `define-labels` og `assign-labels`.

Lag forståelige labels til de ulike yrkeskodene dine ved hjelp av `define-labels`, og knytt dem til den aktuelle variabelen med `assign-labels`.

Merk at labels som inneholder spesialtegn eller mellomrom må angis med enkeltfnutter. Om du er usikker, kan du bare bruke fnutter konsekvent. Da blir det alltid riktig. I eksempelet under brukes lablene “Mann” og “Kvinne”. Disse inneholder ingen spesialtegn og kan fint angis uten fnutter. Verdiene må alltid angis på det formatet de har (med fnutter om det er alfanumeriske verdier, uten fnutter ellers).

Eksempel:
```
define-labels kjønnlabel 1 Mann 2 Kvinne
assign-labels kjønn kjønnlabel
```

c)
Lag en oversikt over antall personer med de ulike yrkene vha. kommandoen `tabulate`. Benytt den nye variabelen `yrkesgr`. Bruk også opsjonen `cellpct` for å vise prosentvis fordeling.

Lag en tabell som viser gjennomsnittlig lønnsinntekt i 2019 fordelt på de utvalgte yrkene vha. kommando-kombinasjonen `tabulate`/`summarize`.

Lag et søylediagram for en grafisk fremvisning av de samme tallene vha. kommandoen `barchart`. Bruk opsjonen `over()` for å splitte opp i søyler basert på yrke.

Lag til slutt et søyledigram som viser lønnsutviklingen over 2015-2019 fordelt på de utvalgte yrkene vha. `barchart`. Bruk også her opsjonen `over()`.

9) Bruke opplysninger om foreldre i analyse

Microdata.no har blant annet opplysninger om fars og mors identitetsnummer tilgjengelig i databasen. Disse kan brukes til å hekte på alt av opplysninger om mor og far. I trinnene under demonstrerer vi hvordan dette gjøres.

a) Importer fars og mors id-numre til datasettet ditt. Variablene heter hhv. BEFOLKNING_FAR_FNR og BEFOLKNING_MOR_FNR. Dette er faste opplysninger og du skal derfor ikke angi måletidspunkt. Bruk aliasene `far_fnr` og `mor_fnr`

b) Opprett et nytt datasett med kommandoen `create-dataset`, som du bruker til å legge til foreldreopplysninger på vanlig måte. Kall datasettet `foreldre`

c) Du befinner deg automatisk i det nye datasettet `foreldre`. Importer variablene INNTEKT_WLONN, REGSYS_ARB_YRKE_STYRK08 og NUDB_BU til dette datasettet på samme måte som du gjorde i ditt opprinnelige datasett. Bruk også samme måletidspunkter, for lønnsvariabelen importerer du kun per 2019-12-31. Tilrettelegg dessuten utdanningsvariabelen på samme måte (1.siffer) og gi passende navn som angir far, f.eks. `lønn19_far`, `yrke19_far`

d) Kopier/klone far-variablene ved hjelp av kommandoen `clone-variables` og kall dem tilsvarende, men angi i stedet mor. Eksempel: `clone-variables lønn19_far -> lønn19_mor`

e) Du har nå et dobbelt sett med lønns-, yrkes- og utdanningsvariabler for hhv. far og mor som du i neste omgang skal koble på ditt opprinnelige datasett. Bruk kommandoen `merge` til dette formålet, og bruk hhv. fars og mors id-numre som

koblingsnøkler. Eksempel: `merge lønn19_far yrke19_far utdnivå_far into totalpop on far_fnr`

f) Forflytt deg over til ditt opprinnelige datasett og lag statistikk:
   i) Statistikk over egen lønn, fars lønn og mors lønn i 2019 vha. `summarize`
   ii) Sjekk om det er korrelasjon (statistisk sammenheng) mellom egen lønn og fars lønn. Gjør det samme også med mors lønn, og sjekk dessuten fars og mors lønn opp mot hverandre. Bruk kommandoen `correlate` og list opp de to variablene som skal sammenliknes, f.eks. `correlate lønn19 lønn19_far`
   iii) Gjør tilsvarende som ii) med eget, mors og fars utdanningsnivå.
   iv) Gjør som i ii) og sjekk egen lønn vs. fars lønn for menn (`if kjønn == '1'`). Sjekk også egen lønn vs. mors lønn for kvinner (`if kjønn == '2'`). Gjør tilsvarende for utdanningsnivå.

g) Bruk yrkesvariabelen for far og trekk ut utvalgte yrker slik som i 8a). Bruk et variabelnavn som angir far, f.eks. `yrkesgr_far`. Om du jobber i skript-editoren, kan du kopiere kommando-linjene du har brukt tidligere, og justere litt. Da går det kjappere. Dette krever at du først overfører arbeidet ditt til skriptområdet. Se 7) for hvordan dette gjøres. Bruk lablene du allerede har laget, og knytt dem til fars yrkeskoder (`assign-labels`). Du trenger ikke lage de samme lablene to ganger.

h) Kjør følgende kommando for å vise sammenheng mellom fars og eget yrke (kjører bare for de utvalgte yrkene og holder "annet" og "uten jobb" utenfor):

```
tabulate yrkesgr_far yrkesgr if yrkesgr_far < 9, rowpct
```

Gjør det samme, men denne gangen kun for menn (kjønn == '1'):

```
tabulate yrkesgr_far yrkesgr if yrkesgr_far < 9 & kjønn == '1', rowpct
```

(*Blir trinnene ovenfor for vanskelige, ta en titt i fasiten bakerst. Det er lov å ta en titt!*)

10) Regresjonsanalyse

a) Gjennomfør en lineær regresjonsanalyse der en analyserer effekter av ulike bakgrunnskarakteristika på lønnsinntekt i 2019 ved å bruke kommandoen `regress`. Benytt variabler du har laget tidligere i oppgavesettet:

   - Lønnsinntekt per 2019 (avhengig variabel)

- Fars lønnsinntekt per 2019
- Alder
- Norsk
- Mann (lag en dummyvariabel for mann ut i fra variabelen `kjønn`)
- Oslo (lag en dummyvariabel som settes lik 1 dersom `bosted` == '0301')
- Høy utdanning (lag en dummyvariabel for høyt utdanningsnivå ut i fra variabel for høyeste fullførte utdanning, nivå >= 7 => verdien 1, resten 0)

NB! Husk at den avhengige variabelen skal listes først i regress-uttrykket.

Forslag:

```
generate mann = 0
replace mann = 1 if kjønn == '1'

generate oslo = 1 if bosted == '0301'
replace oslo = 0 if bosted != '0301'
tabulate oslo, cellpct

generate høy_utd = 1 if utdnivå >= 7
replace høy_utd = 0 if utdnivå >= 0 & utdnivå < 7

regress lønn19 alder norsk oslo mann høy_utd lønn19_far
```

b) Bruk kommandoen `regress-predict` til å hente ut residualverdiene fra regresjonsmodellen ovenfor. Tips: Du spesifiserer variablene på nøyaktig samme måte som i `regress`-uttrykket i a). Spesifiser i tillegg opsjonen `residuals()` for å angi at du vil generere en ny variabel som inneholder residualene (inni parentesen angir du et navn på variabelen med residualene, f.eks. `res`)

c) Studér residualene ved å bruke grafiske verktøy som f.eks. `histogram`. Du bruker navnet på variabelen med residualene som du lagde i b) som input til den grafiske fremstillingen.

d) For å sjekke om residualene er normalfordelte, bruk kommandoen `histogram` (med residual-variabelen som input) med opsjonen `normal`.

11) Ekstraoppgave: Hente ut opplysninger basert på hendelser over tid - forløpsuttrekk

I tillegg til vanlige tverrsnittsuttrekk (ett måletidspunkt) som en gjør ved bruk av kommandoen `import`, kan en også hente ut hendelser som skjer over et tidsspenn. Dette kan brukes til å identifisere enheter der en bruker tid som ekstra dimensjon.

Vi skal i denne oppgaven prøve oss på å identifisere individer i vår populasjon som skilte seg i løpet av året før, dvs. i 2018.

a) Bruk kommandoen `create-dataset` og opprett et nytt datasett med et passende navn, f.eks. `forløp`.
b) Bruk kommandoen `import-event` til å hente ut alle records for variabelen ds/SIVSTANDFDT_SIVSTAND som dekkes av tidsintervallet 2018-01-01 til 2018-12-31. Variabelen kan gis et passende navn, f.eks. `sivforløp`
c) Bruk kommandoen `keep if` til å beholde kun records som inneholder verdien for tilstanden “skilt”, dvs. verdien ‘4’.
d) Du må nå aggregere opp dataene til personnivå med én record per individ. Du bruker da kommandoen `collapse(count)` og spesifiserer forløpsvariabelen `sivforløp` etterfulgt av et komma for å angi opsjon, og til slutt betingelsen `by(PERSONID_1)`. Målestatistikken `count` teller opp antall records (altså med statusverdien “skilt” siden vi har fjernet øvrige records med andre statuser), og erstatter opprinnelig verdi med dette antallet for variabelen `sivforløp`.
e) Bruk kommandoen `rename` til å døpe om `sivforløp` til f.eks. `ant_ganger_skilt`.
f) Bruk kommandoen `merge` til å koble variabelen `ant_ganger_skilt` med ditt persondatasett (`totalpop`).
g) Bruk kommandoen `use` til å flytte deg over til persondatasettet `totalpop`. Lag en dummyvariabel som tar verdien 1 for personer med verdier på `ant_ganger_skilt` >= 1. Øvrige gis verdien 0. Bruk kommandoene `generate` og `replace` i likhet med fremgangsmåten tidligere i oppgavesettet.
h) Sjekk hvor mange i din populasjon som har status “skilt” i løpet av 2018 ved å bruke kommandoen `tabulate`. Bruk dummyvariabelen fra g) som input.

Fasit (syntax):

1)
require no.ssb.fdb:12 as ds

2)
create-dataset totalpop

3a)
import ds/BEFOLKNING_FOEDSELS_AAR_MND as faarmnd
generate alder = 2019 - int(faarmnd/100)

3b)
histogram alder, discrete

3c)
import ds/BEFOLKNING_STATUSKODE 2019-01-01 as regstat
keep if regstat == '1'
histogram alder, discrete
summarize alder

3d)
keep if alder > 30 & alder < 50

4)
import ds/BEFOLKNING_KJOENN as kjønn
import ds/BEFOLKNING_FODELAND as land
import ds/NUDB_BU 2019-07-31 as utd
import ds/BEFOLKNING_KOMMNR_FAKTISK 2019-01-01 as bosted
import ds/INNTEKT_WLONN 2015-12-31 as lønn15
import ds/INNTEKT_WLONN 2016-12-31 as lønn16
import ds/INNTEKT_WLONN 2017-12-31 as lønn17
import ds/INNTEKT_WLONN 2018-12-31 as lønn18
import ds/INNTEKT_WLONN 2019-12-31 as lønn19

5)
summarize lønn15 lønn16 lønn17 lønn18 lønn19

```
barchart (mean) lønn15 lønn16 lønn17 lønn18 lønn19
barchart (count) lønn15 lønn16 lønn17 lønn18 lønn19
barchart (median) lønn15 lønn16 lønn17 lønn18 lønn19

histogram lønn19, freq
histogram lønn19, freq normal


6a)
generate norsk = 0
replace norsk = 1 if land == '000'
tabulate norsk
tabulate norsk, cellpct
piechart norsk

6b)
generate utdnivå = substr(utd,1,1)
tabulate utdnivå, cellpct
destring utdnivå

6c)
summarize lønn15 lønn16 lønn17 lønn18 lønn19 if utdnivå < 2
summarize lønn15 lønn16 lønn17 lønn18 lønn19 if utdnivå > 6

tabulate kjønn, summarize(lønn19)
tabulate norsk, summarize(lønn19)
tabulate utdnivå, summarize(lønn19)

barchart (mean) lønn15 lønn16 lønn17 lønn18 lønn19, over(kjønn)
barchart (mean) lønn15 lønn16 lønn17 lønn18 lønn19, over(norsk)
barchart (mean) lønn15 lønn16 lønn17 lønn18 lønn19, over(utdnivå)


8a)
import ds/REGSYS_ARB_YRKE_STYRK08 2019-11-16 as yrke19

generate yrkesgr = 9
replace yrkesgr = 1 if substr(yrke19,1,1) == '1'
replace yrkesgr = 2 if yrke19 == '2211'
replace yrkesgr = 3 if yrke19 == '2212'
replace yrkesgr = 4 if yrke19 == '2223'
replace yrkesgr = 5 if substr(yrke19,1,2) == '23'
replace yrkesgr = 6 if substr(yrke19,1,2) == '25'
```

```
replace yrkesgr = 7 if substr(yrke19,1,2) == '61'
replace yrkesgr = 999 if sysmiss(yrke19)

8b)
define-labels yrkelabel 1 Ledere 2 Allmennleger 3 Legespesialister 4 Sykepleiere 5 Lærere 6
'IT-utviklere' 7 Landbruk 9 Annet 999 'Uten jobb'
assign-labels yrkesgr yrkelabel

8c)
tabulate yrkesgr
tabulate yrkesgr, cellpct
tabulate yrkesgr, summarize(lønn19)
barchart (mean) lønn19, over(yrkesgr)
barchart (mean) lønn15 lønn16 lønn17 lønn18 lønn19, over(yrkesgr)


9a)
import ds/BEFOLKNING_FAR_FNR as far_fnr
import ds/BEFOLKNING_MOR_FNR as mor_fnr

9b)
create-dataset foreldre

9c)
import ds/INNTEKT_WLONN 2019-12-31 as lønn19_far
import ds/REGSYS_ARB_YRKE_STYRK08 2019-11-16  as yrke19_far
import ds/NUDB_BU 2019-07-31 as utd_far
generate utdnivå_far = substr(utd_far,1,1)
destring utdnivå_far

9d)
clone-variables lønn19_far -> lønn19_mor
clone-variables yrke19_far -> yrke19_mor
clone-variables utdnivå_far -> utdnivå_mor

9e)
merge lønn19_far yrke19_far utdnivå_far into totalpop on far_fnr
merge lønn19_mor yrke19_mor utdnivå_mor into totalpop on mor_fnr

9f)
use totalpop
summarize lønn19 lønn19_far lønn19_mor
correlate lønn19 lønn19_far
```

```
correlate lønn19 lønn19_mor
correlate lønn19_far lønn19_mor
correlate utdnivå utdnivå_far
correlate utdnivå utdnivå_mor
correlate utdnivå_far utdnivå_mor

correlate lønn19 lønn19_far if kjønn == '1'
correlate lønn19 lønn19_mor if kjønn == '2'
correlate utdnivå utdnivå_far if kjønn == '1'
correlate utdnivå utdnivå_mor if kjønn == '2'
```

9g)

```
generate yrkesgr_far = 9
replace yrkesgr_far = 1 if substr(yrke19_far,1,1) == '1'
replace yrkesgr_far = 2 if yrke19_far == '2211'
replace yrkesgr_far = 3 if yrke19_far == '2212'
replace yrkesgr_far = 4 if yrke19_far == '2223'
replace yrkesgr_far = 5 if substr(yrke19_far,1,2) == '23'
replace yrkesgr_far = 6 if substr(yrke19_far,1,2) == '25'
replace yrkesgr_far = 7 if substr(yrke19_far,1,2) == '61'
replace yrkesgr_far = 999 if sysmiss(yrke19_far)

assign-labels yrkesgr_far yrkelabel
```

9h)

```
tabulate yrkesgr_far yrkesgr if yrkesgr_far < 9, rowpct
tabulate yrkesgr_far yrkesgr if yrkesgr_far < 9 & kjønn == '1', rowpct
```

10a)

```
generate mann = 0
replace mann = 1 if kjønn == '1'

generate oslo = 1 if bosted == '0301'
replace oslo = 0 if bosted != '0301'
tabulate oslo, cellpct

generate høy_utd = 1 if utdnivå >= 7
replace høy_utd = 0 if utdnivå >= 0 & utdnivå < 7

regress lønn19 alder norsk oslo mann høy_utd lønn19_far
```

10b)

regress-predict lønn19 alder norsk oslo mann høy_utd lønn19_far, residuals(res)

10c)
histogram res

10d)
histogram res, normal

11a)
create-dataset forløp

11b)
import-event ds/SIVSTANDFDT_SIVSTAND 2018-01-01 to 2018-12-31 as sivforløp

11c)
keep if sivforløp == '4'

11d)
collapse (count) sivforløp , by(PERSONID_1)

11e)
rename sivforløp ant_ganger_skilt

11f)
merge ant_ganger_skilt into totalpop

11g)
use totalpop
generate skilt2018 = 0
replace skilt2018 = 1 if ant_ganger_skilt >= 1

11h)
tabulate skilt2018
