Ulike måter å sette sammen familiedata på

1) Aggregere fra persondata til familiedata

a) Lag to datasett:
   - Datasett A: Persondatasett
   - Datasett B: Persondatasett bestående av familieid og familieopplysninger

b) Datasett B aggregeres til familienivå via familieid (= personid til eldste person i familien)

c) Datasett B kobles mot A via **personid**

d) Datasett A inneholder nå familieopplysninger for personene som er eldst i familien

e) Personer i datasett A som ikke har påkoblet familieopplysninger fjernes

f) Datasett A inneholder nå data på familienivå (samt personopplysninger knyttet til eldste person i familien).
   - Populasjon = alle familier gitt ved uttrekkskriteriene for datasett A, representert ved eldste person i hver familie

2) Konstruere familiedata og koble på persondatasett som kontekstuell opplysning

a) Lag to datasett:
   - Datasett A: Persondatasett bestående av familieid
   - Datasett B: Persondatasett bestående av familieid og familieopplysninger

b) Datasett B aggregeres til familienivå via familieid

c) Datasett B kobles mot A **via familieid**

d) Datasett A inneholder nå persondata og familiedata på personnivå. Alle personer tilhørende samme familie (samme verdi på familieid) vil ha samme familieopplysninger.
   - Populasjon = alle individer gitt ved uttrekkskriteriene for datasett A (uendret)

3) Koble foreldredata på persondata

a) Lag to datasett:
   - Datasett A: Persondatasett bestående av farid og morid
   - Datasett B: Persondatasett bestående av opplysninger knyttet til hhv. far og mor

b) Koble hhv. far- og mor-opplysninger fra datasett B mot datasett A **via hhv. farid og morid**

c) Datasett A inneholder nå persondata og data knyttet til hhv. far og mor
   - Populasjon = alle individer gitt ved uttrekkskriteriene for datasett A (uendret)

4) Koble data om foreldre og besteforeldre på persondata (flere generasjoner)

a) Lager først to datasett:
   - Datasett A: Persondatasett bestående av farid og morid
   - Datasett B: Persondatasett bestående av opplysninger knyttet til hhv. far og mor, inkludert fars farid og morid, samt mors farid og morid

b) Datasett B kobles mot datasett A **via hhv. farid og morid**

c) Datasett A inneholder nå persondata og data knyttet til hhv. far og mor, inkludert fars farid (=farfarid) og morid (=farmorid) samt mors farid (=morfarid) og morid (=mormorid)

d) Lag datasett C: Persondatasett med opplysninger knyttet til hhv. farfar, farmor, morfar og mormor

e) Datasett C kobles mot datasett A **via hhv. farfarid, farmorid, morfarid og mormorid**

f) Datasett A inneholder nå persondata og data knyttet til hhv. far, mor, mormor, morfar, farmor og farfar
   - Populasjon = alle individer gitt ved uttrekkskriteriene for datasett A (uendret)

5) Konstruere persondatasett bestående av ektefeller

a) Lag to datasett:
   - Datasett A: Persondatasett bestående av personer i en familie som er **eldst og som ikke er barn** (keep if personkode == '1'), inkludert variablene familieid, personkode, parstatus og andre opplysninger som f.eks. yrke
   - Datasett B: Persondatasett med de samme variablene som i datasett A, bestående av personer i en familie som er **yngst og som ikke er barn** (keep if personkode == '2')

b) Flytt deg over til datasett A (eldst i familien) med kommandoen «use»

c) Variablene i datasett A kobles på datasett B (yngst i familien) **via familieid**
   - Grunnen til at du må koble data fra A til B og ikke motsatt, er at datasett B sin familieid-variabel peker til eldste person i familien (som er dem som befinner seg i datasett A)

d) Flytt deg over til datasett B (yngst i familien) med kommandoen «use»

e) Datasettet B vil nå inneholde alle opplysningene du trenger for å lage statistikk for ektefeller (hvem som er mann og kone kommer an på hvem av dem som er yngst/eldst, men dette kan en se ved å inkludere variabelen «kjønn» i datasettet):
   - Personkode_yngst
   - Parstatus_yngst
   - Yrke_yngst
   - Personkode_eldst
   - Parstatus_eldst
   - Yrke_eldst

Kontekstuelle kommunedata

6) Koble kontekstuelle (aggregerte) kommunedata på et persondatasett

- Prinsippet er det samme som for 2) der vi kobler familiedata på et persondatasett, men her bruker vi en vanlig kommunevariabel som koblingsnøkkel i stedet for familieid.

a) Lag to datasett:
   - Datasett A: Persondatasett bestående av kommunevariabel
   - Datasett B: Persondatasett bestående av kommunevariabel samt lønn og formue

b) Datasett B aggregeres til kommunenivå via kommunevariabelen. Bruker collapse(mean) ……, by(kommune) som beregner gjennomsnittsverdi av lønn og formue over alle kommuner

c) Datasett B kobles mot datasett A **via kommunevariabelen**

d) Datasett A inneholder nå persondata og gjennomsnittlig lønn og formue per kommune (verdiene knyttes til kommunen hver enkelt er bosatt i)

e) Vi ønsker også å lage data som viser andel arbeidsledige på kommunenivå, og oppretter et tredje datasett:
   - Datasett C: Persondatasett bestående av kommunevariabel og variabel for arbeidssøkerstatus

f) Lag en variabel «ant_ledig» som tar verdien 1 for alle med arbeidssøkerstatus == ‘1’ (helt ledig)
g) Lag en variabel «ant_bosatt» som tar verdien 1 for alle individer (=alle bosatte)
h) Datasett C aggregeres til kommunenivå via kommunevariabelen. Bruker collapse(sum) ……, by(kommune) som beregner summen av antall ledige og bosatte over alle kommuner
i) Datasett C kobles mot datasett A **via kommunevariabelen**
j) Flytt deg over til datasett A ved kommandoen «use»
k) Datasett A inneholder nå også antallet ledige og antall bosatte per kommune
l) Gjør om til andel ledige per kommune ved hjelp av tallene for antall ledige og antall bosatte per kommune: generate ledig_pst = (ant_ledig / ant_bosatt) * 100

Vedlegg A: Skriptsyntax for familiedata (1-5)

```
//Kobler til databank

require no.ssb.fdb:15 as db


textblock

1) Aggregere fra persondata til familiedata

----------------------------------------

endblock


//Oppretter først et persondatasett for personer i familier bestående av
ektepar med små barn

create-dataset persondata1

import db/BEFOLKNING_REGSTAT_FAMTYP 2019-01-01 as famtype

tabulate famtype

keep if famtype == '2.1.1'


//Legger til demografiske opplysninger

import db/BEFOLKNING_FOEDSELS_AAR_MND as faarmnd

generate alder = 2019 - int(faarmnd/100)

import db/BEFOLKNING_BARN_I_REGSTAT_FAMNR 2019-01-01 as antbarn


//Oppretter datasett for generering av total yrkesinntekt per familie =>
enhet = familie

create-dataset familiedata1

import db/BEFOLKNING_REGSTAT_FAMNR 2019-01-01 as famnr

import db/INNTEKT_WYRKINNT 2019-01-01 as yrkesinnt

collapse (sum) yrkesinnt, by(famnr)

rename yrkesinnt familieinnt


//Kobler familieinntekt på persondatasettet (enhet = personer)

merge familieinnt into persondata1 on PERSONID_1


//Lager familiestatistikk. Familienummeret består av person-id til eldste
person i familien, så når en fjerner individer med manglende familieinntekt
```

```
sitter en igjen med et datasett med familie som enhet. Alle
personopplysninger vil da gjelde for eldste person i familien

use persondata1

drop if sysmiss(familieinnt)

rename alder alder_eldst


histogram alder_eldst, discrete

histogram antbarn, discrete percent

tabulate antbarn


summarize familieinnt

barchart (mean) familieinnt, by(antbarn)

histogram familieinnt, freq

histogram familieinnt, by(antbarn) percent



textblock

2) Konstruere familiedata og koble på persondatasett

-----------------------------------------------------

endblock


//Oppretter først et persondatasett for personer i familier bestående av
ektepar med små barn

create-dataset persondata2

import db/BEFOLKNING_REGSTAT_FAMTYP 2019-01-01 as famtype

import db/BEFOLKNING_REGSTAT_FAMNR 2019-01-01 as famnr

tabulate famtype

keep if famtype == '2.1.1'


//Legger til diverse demografiske opplysninger

import db/BEFOLKNING_FOEDSELS_AAR_MND as faarmnd

generate alder = 2019 - int(faarmnd/100)

import db/BEFOLKNING_BARN_I_REGSTAT_FAMNR 2019-01-01 as antbarn
```

```
//Oppretter datasett for generering av total yrkesinntekt per familie =>
enhet = familie

create-dataset familiedata2

import db/BEFOLKNING_REGSTAT_FAMNR 2019-01-01 as famnr

import db/INNTEKT_WYRKINNT 2019-01-01 as yrkesinnt

collapse (sum) yrkesinnt, by(famnr)

rename yrkesinnt familieinnt


//Kobler familieinntekt på persondatasettet (enhet = personer)

merge familieinnt into persondata2 on famnr


//Lager person- og familiestatistikk

use persondata2

histogram alder, discrete

histogram antbarn, discrete percent

tabulate antbarn


summarize familieinnt

barchart (mean) familieinnt, by(antbarn)

histogram familieinnt, freq

histogram familieinnt, by(antbarn) percent



textblock

3) Koble foreldredata på persondata

-----------------------------------

endblock


//Lager et persondatasett med lenker til far og mor

create-dataset persondata3

import db/INNTEKT_WYRKINNT 2019-01-01 as inntekt

import db/BEFOLKNING_KJOENN as kjønn

import db/NUDB_BU 2019-01-01 as utd

import db/BEFOLKNING_FAR_FNR as fnr_far

import db/BEFOLKNING_MOR_FNR as fnr_mor
```

```
//Henter opplysninger om foreldre og kobler på persondatasett
create-dataset foreldredata3
import db/INNTEKT_WYRKINNT 2019-01-01 as inntekt_far
import db/NUDB_BU 2019-01-01 as utd_far
clone-variables inntekt_far -> inntekt_mor
clone-variables utd_far -> utd_mor

merge inntekt_far utd_far into persondata3 on fnr_far
merge inntekt_mor utd_mor into persondata3 on fnr_mor

//Kjører en enkel lineær regresjon for å teste sammenheng mellom egen og
foreldres inntekt
use persondata3
generate mann = 0
replace mann = 1 if kjønn == '1'

destring utd
generate høyutd = 0
replace høyutd = 1 if utd >= 700000 & utd < 900000
replace høyutd = utd if sysmiss(utd)

destring utd_far
generate høyutd_far = 0
replace høyutd_far = 1 if utd_far >= 700000 & utd_far < 900000
replace høyutd_far = utd_far if sysmiss(utd_far)

destring utd_mor
generate høyutd_mor = 0
replace høyutd_mor = 1 if utd_mor >= 700000 & utd_mor < 900000
replace høyutd_mor = utd_mor if sysmiss(utd_mor)

summarize inntekt inntekt_far inntekt_mor
histogram inntekt_far, percent
histogram inntekt_mor, percent
```

```
correlate inntekt_far inntekt_mor
tabulate høyutd_far høyutd_mor

regress inntekt mann inntekt_far inntekt_mor høyutd høyutd_far høyutd_mor



textblock
4) Koble data om foreldre og besteforeldre på persondatasett
------------------------------------------------------------
endblock

//Lager et persondatasett med lenker til far og mor
create-dataset persondata4
import db/INNTEKT_WYRKINNT 2019-01-01 as inntekt
import db/BEFOLKNING_KJOENN as kjønn
import db/NUDB_BU 2019-01-01 as utd
import db/BEFOLKNING_FAR_FNR as fnr_far
import db/BEFOLKNING_MOR_FNR as fnr_mor

//Henter opplysninger om foreldre og kobler på persondatasett
create-dataset foreldre4
import db/INNTEKT_WYRKINNT 2019-01-01 as inntekt_far
import db/NUDB_BU 2019-01-01 as utd_far
clone-variables inntekt_far -> inntekt_mor
clone-variables utd_far -> utd_mor

import db/BEFOLKNING_FAR_FNR as fnr_farfar
import db/BEFOLKNING_FAR_FNR as fnr_morfar
import db/BEFOLKNING_MOR_FNR as fnr_farmor
import db/BEFOLKNING_MOR_FNR as fnr_mormor

merge inntekt_far utd_far fnr_farfar fnr_farmor into persondata4 on fnr_far
merge inntekt_mor utd_mor fnr_morfar fnr_mormor into persondata4 on fnr_mor

create-dataset besteforeldre4
```

```
import db/INNTEKT_WYRKINNT 2019-01-01 as inntekt_farfar

import db/NUDB_BU 2019-01-01 as utd_farfar

clone-variables inntekt_farfar -> inntekt_farmor

clone-variables inntekt_farfar -> inntekt_morfar

clone-variables inntekt_farfar -> inntekt_mormor

clone-variables utd_farfar -> utd_farmor

clone-variables utd_farfar -> utd_morfar

clone-variables utd_farfar -> utd_mormor


merge inntekt_farfar utd_farfar into persondata4 on fnr_farfar

merge inntekt_farmor utd_farmor into persondata4 on fnr_farmor

merge inntekt_morfar utd_morfar into persondata4 on fnr_morfar

merge inntekt_mormor utd_mormor into persondata4 on fnr_mormor


use persondata4

summarize inntekt inntekt_far inntekt_mor inntekt_farfar inntekt_farmor
inntekt_morfar inntekt_mormor

barchart(mean) inntekt inntekt_far inntekt_mor inntekt_farfar
inntekt_farmor inntekt_morfar inntekt_mormor




textblock

5) Konstruere persondatasett bestående av ektefeller

-----------------------------------------------------

endblock


//Lager datasett bestående av eldste person i hver familie

create-dataset eldst5

import db/BEFOLKNING_REGSTAT_FAMNR 2021-01-01 as famnr

import db/BEFOLKNING_REGSTAT_PERSONKODE 2021-01-01 as personkode_eldst

import db/BEFOLKNING_PARSTATUS 2021-01-01 as parstatus_eldst

import db/REGSYS_ARB_YRKE_STYRK08 2020-11-16 as yrke_eldst

keep if personkode_eldst == '1'
```

```
//Lager datasett med yngste person i hver familie, og som ikke er barn

create-dataset yngst5

import db/BEFOLKNING_REGSTAT_FAMNR 2021-01-01 as famnr

import db/BEFOLKNING_REGSTAT_PERSONKODE 2021-01-01 as personkode_yngst

import db/BEFOLKNING_PARSTATUS 2021-01-01 as parstatus_yngst

import db/REGSYS_ARB_YRKE_STYRK08 2020-11-16 as yrke_yngst

keep if personkode_yngst == '2'


//Bruker datasettet med eldste personer og kobler tilhørende variabler inn
i datasett med yngste personer via familienummer (familienummer = eldste
persons id-nummer)

use eldst5

merge personkode_eldst parstatus_eldst yrke_eldst into yngst5 on famnr


//Datasettet yngst inneholder nå data om begge ektefeller/parmedlemmer for
alle par i befolkningen. Kjører kontrolltabeller

use yngst5

tabulate personkode_yngst, missing

tabulate personkode_eldst, missing

tabulate parstatus_yngst, missing

tabulate parstatus_eldst, missing

tabulate yrke_eldst, missing

tabulate yrke_yngst, missing


//Sjekke om begge, en eller ingen av ektefellene/parmedlemmene er i jobb

generate jobb_yngst = 1 if sysmiss(yrke_yngst) == 0

generate jobb_eldst = 1 if sysmiss(yrke_eldst) == 0

tabulate jobb_eldst jobb_yngst, missing
```

Vedlegg B: Skriptsyntax for kommunedata

```
textblock
Hvordan koble kontekstuelle data på et persondatasett
-----------------------------------------------------
endblock

//Kobler på database
require no.ssb.fdb:15 as db

//Lager persondatasett
create-dataset personer
import db/BEFOLKNING_KOMMNR_FAKTISK 2019-01-01 as kommune
import db/BEFOLKNING_KJOENN as kjønn
import db/INNTEKT_WLONN 2019-01-01 as lønn
import db/INNTEKT_BER_BRFORM 2019-01-01 as formue
summarize lønn formue
histogram lønn, freq
histogram formue, freq

//Lager tall for snittlønn og snittformue per kommune og kobler på
persondatasett
create-dataset kommunedata_lønn_formue
import db/BEFOLKNING_KOMMNR_FAKTISK 2019-01-01 as kommune
import db/INNTEKT_WLONN 2019-01-01 as lønn
import db/INNTEKT_BER_BRFORM 2019-01-01 as formue
collapse (mean) lønn formue, by(kommune)
rename lønn snittlønn_kommune
rename formue snittformue_kommune
summarize snittlønn_kommune snittformue_kommune
merge snittlønn_kommune snittformue_kommune into personer on kommune

//Lager tall for antall arbeidsledige og bosatte per kommune og kobler på
persondatasett
create-dataset kommunedata_ledig_bosatt
```

```
import db/BEFOLKNING_KOMMNR_FAKTISK 2019-01-01 as kommune

import db/ARBSOEK2001FDT_HOVED 2019-01-01 as as_status

generate ant_ledig = 1 if as_status == '1'

generate ant_bosatt = 1

collapse (sum) ant_ledig ant_bosatt, by(kommune)

summarize ant_ledig ant_bosatt

merge ant_ledig ant_bosatt into personer on kommune


//Bruker til slutt persondatasettet til å foreta en enkelt
regresjonsanalyse

use personer

generate ledig_pst = (ant_ledig / ant_bosatt) * 100


generate mann = 0

replace mann = 1 if kjønn == '1'


generate oslo = 0

replace oslo = 1 if kommune == '0301'


generate storkommune = 0

replace storkommune = 1 if ant_bosatt > 80000


generate rik_kommune = 0

replace rik_kommune = 1 if snittformue_kommune > 2000000


regress lønn mann oslo formue snittformue_kommune snittlønn_kommune
ledig_pst ant_bosatt

regress lønn mann oslo formue rik_kommune snittlønn_kommune ledig_pst
storkommune
```