Statistics/Data analysis

```
/*
use "${data}\Example_FiscalSim_raw_data.dta" , clear
merge 1:1 hhID memberID using "${data}\proc\Example_FiscalSim_dem_data.dta", nogen u
> pdate replace
merge 1:1 hhID memberID using "${data}\proc\Example_FiscalSim_market_income_data.dta
> ", nogen update replace
merge 1:1 hhID memberID using "${data}\proc\Example_FiscalSim_SSC_direct_taxes_data.
> dta", nogen update replace
*/

use "${simulationData}\01_${countryName}_${simulationName}_${dem_inc_SY}.dta" , clea
> r
merge 1:1 hhID memberID using "${simulationData}\06_${countryName}_${simulationName}
> _${mkt_inc_PY}.dta", nogen assert(match)
merge 1:1 hhID memberID using "${simulationData}\07_${countryName}_${simulationName}
> _${ssc_dir_tax_PY}.dta", nogen assert(match)

egen double pens_other_trans_orig = rowtotal(${pensions} /*other_ben*/)


* UPRATING TO THE POLICY YEAR THOSE PROGRAMS THAT ARE NOT SIMULATED
foreach var in $pensions /*soc_pens unem_ben other_ben*/ {
        replace `var' = `var' * ${pensions_uprating}
}


* UPRATING AGE TO THE POLICY YEAR                                          // Misha - l
> et me know if this is OK
foreach var in age {
        replace `var' = `var' + ${`var'_uprating}
}


* child benefits
gen child = (age <= 16)
>     // define children
bysort hhID: egen n_child = total(child)                                  // count num
> ber of children in the household
gen n_elig_child = min(n_child, ${max_child_elig})              //restrict number of
>  children using the information from the parameter sheet
gen double child_ben = ${child_benefit} * n_elig_child * 12 // Calculate the amount
> of the benefit. Do not forget to convert monthly paramenetrs to annual values if r
> elevant

gen n_elig_child_orig = min(n_child, 3)
gen double child_ben_orig = 100 * n_elig_child_orig * 12 // We calculate the origina
> l amount of benefits for the baseline separately


* simulating increase in enrollment in unemployemnt benefits
gen unem_potent = (unem_ben == 0 & wage == 0 & entr_inc == 0 & inrange(age,17,60)) /
> / we define who are potential recepient of unemplyment benefits
gen double unem_ben_orig = unem_ben

su unem_ben [aw = ind_weight] if unem_ben > 0
        global unem_ben = r(mean) // we will impute average value of unemployment ot
>  the new recipients
```

```
gen unem_weight = ind_weight
        replace unem_weight = 0 if unem_ben > 0 // we exclude those who are already
> in the program

        set seed 1000
        gen rank = runiform() if unem_potent == 1 // instead of random allocation, w
> e may use predicted probabilities if there is information

        gen all = 1
        bysort all (rank hhID memberID): gen double cum_unem_weight = sum(unem_weigh
> t)
        replace unem_ben = ${unem_ben} if cum_unem_weight <= ${unempl_coverage_incre
> ase}
        replace unem_ben_orig = ${unem_ben} if cum_unem_weight <= 1000


* GMI program (may depend on other programs)
gen double GMI = 0
egen double pre_GMI_income = rowtotal(${market_income} ${SSC} ${direct_taxes} ${pens
> ions} ${direct_transfers}) // include only those that are used to for GMI administ
> rative income (GMI is put as zero for time-being)
bysort hhID: egen double pre_GMI_income_hh = total(pre_GMI_income) // count number o
> f children in the household
gen double pre_GMI_income_pc = pre_GMI_income_hh / hh_size
replace GMI = max(0, ${GMI_threshold} * 12 - pre_GMI_income_pc) // This programs cov
> er the income upto the threshold

* GMI program for the basline
egen double pre_GMI_income_orig = rowtotal(net_market_income_orig /*pens_other_trans
> _orig unem_ben_orig child_ben_orig*/)
bysort hhID: egen double pre_GMI_income_hh_orig = total(pre_GMI_income_orig) // coun
> t number of children in the household
gen double pre_GMI_income_pc_orig = pre_GMI_income_hh_orig / hh_size
gen double  GMI_orig = max(0, 1200 * 12 - pre_GMI_income_pc_orig) // This programs c
> over the income upto the threshold


/*
if $income_consistency_check ==1 {
>
        foreach var in child_ben unem_ben GMI {
            assert abs(`var' - `var'_orig) < 10 ^ (-10)
        }
}
*/


egen double pens_trans_orig = rowtotal(child_ben_orig unem_ben_orig GMI_orig pens_ot
> her_trans_orig)

keep hhID memberID ${pensions} ${direct_transfers} pens_trans_orig
order hhID memberID ${pensions} ${direct_transfers} pens_trans_orig

mvencode ${pensions} ${direct_transfers} pens_trans_orig, mv(0) override

isid hhID memberID

save "${simulationData}\08_${countryName}_${simulationName}_${pens_dir_trans_PY}.dta
> ", replace
```