Statistics/Data analysis

```
//  Program:    04-dataPrep-income-transfersAndPensions.do
//  Task:       cleans hh-level private/govt transfers and pension incomes using Ken
> yan survey data (2015/16 KIHBS)
//  Project:    Kenya Fical MicroSim
//  Author:     Yared Seid - 2023Jan15


// #1
// Loading the data
use "${dataSources}\Kenya_2015_16_KIHBS\HH_Information", clear


// #2
// Generating vars

* IDs
codebook county clid hhid

qui tostring clid,      gen(clusterID) format(%04.0f)                    // cluster i
> d
qui tostring hhid,      gen(hhid_STR) format(%02.0f)                     // houdehold
>  id, within county and cluster

label var clusterID "Cluster ID"

gen hhID = clusterID + hhid_STR
label var hhID "Household ID"
isid hhID

// TRANSFERS
des o*
clonevar any_transfer = o01
label var any_transfer "=1 if any transfer"
tab any_transfer

// Transfer recieved in cash
* private transfer
egen priv_trans_cash = rowtotal(o02_a o02_g o02_b o02_e), missing
>     // a= from ind transfer (within Kenya transfer), g= from outside Kenya transfe
> r AND b= from non-profit institution, e= from corporate sector (both b&e are from
> within Kenya transfer)  // annual
label var priv_trans_cash "Private transfer in cash"

* government transfer
egen govt_trans_cash = rowtotal(o02_c o02_d), missing                    // c= nation
> al govt and d= county govt // annual
label var govt_trans_cash "Government transfer in cash"

des priv_trans_cash govt_trans_cash
sum priv_trans_cash govt_trans_cash

// Transfer recieved in kind
* private transfer
egen priv_trans_kind = rowtotal(o10_a o10_e o02_b o02_d), missing
>     // a= from ind transfer (within Kenya transfer), e= from outside Kenya transfe
> r AND b= from non-profit institution, d= from corporate sector (both b&d are from
> within Kenya transfer)        // annual
label var priv_trans_kind "Private transfer in kind"

* government transfer
egen govt_trans_kind = rowtotal(o10_c), missing                 // c= govt   // annu
> al
label var govt_trans_kind "Government transfer in kind"
```

```
des priv_trans_kind govt_trans_kind
sum priv_trans_kind govt_trans_kind

// Total transfers (in cash + in kind) by govt and private transfer
egen priv_trans = rowtotal(priv_trans_cash priv_trans_kind), missing
label var priv_trans "Private transfer: cash + in kind"

egen govt_trans = rowtotal(govt_trans_cash govt_trans_kind), missing
label var govt_trans "Government transfer: cash + in kind"

local transferVars priv_trans_cash priv_trans_kind priv_trans ///
        govt_trans_cash govt_trans_kind govt_trans

des `transferVars'
sum `transferVars'

*******************************************************************************
/* Final touch of key variables - order matters! First annualize, then cap upper out
> liers */
local finalTouchVars priv_trans_cash priv_trans_kind priv_trans ///
        govt_trans_cash govt_trans_kind govt_trans

sum `finalTouchVars'
/*
foreach z of local finalTouchVars {
ys_annualizing_values   `z', recall_in_days()           // values are reported annua
> lly in the original data
}
*
*/

sum `finalTouchVars'
ys_upper_outliers               `finalTouchVars'                                    /
> / setting upper outliers to MEAN+3SD
sum `finalTouchVars'

        /* Graph it */
        foreach z of local finalTouchVars{
        sum `z'
        local `z'_Mean = round(`r(mean)')
        histogram `z', fraction normal ti("`z' - Annual (Mean = ``z'_Mean')")  xline
> (``z'_Mean')
        }
*******************************************************************************


// OTHER INCOMES
des p*

* capital income
clonevar cap_income_d = p02
>     // capital income dummy
tab cap_income_d

gen cap_inc = p03
>     // annual
label var cap_inc "Capital income"

* pensions income
clonevar pension_d = p04
>     // dummy for any pension income
```

```
gen lab_pens_d = [(p05_1=="A") | (p05_1=="B") | (p05_1=="C")] & !missing(p05_1)
>     // dummy for labor pension: A=CIVIL SERVANTS PENSION PLAN, B=PRIVATE PENSION,
> C=PERSONAL PENSION PLAN
gen oth_pens_d = (p05_1=="X")  & !missing(p05_1)                // dummy for other p
> ension

gen lab_pens = p06 if lab_pens_d==1                                                 // m
> onthly
label var lab_pens "Pension income: from labor"

gen oth_pens = p06 if oth_pens_d==1                                               // monthl
> y
label var oth_pens "Pension income: other"

* renal income
clonevar rent_inc_d = p07
>     // dummy for rental income
egen rent_inc = rowtotal(p08_re p08_co p08_la p08_ma p08_su p08_ot), missing  // fro
> m residential, commercial, land, machinery, sub soil assets, and others  // monthl
> y
labe var rent_inc "Rental income"

* other income - regural
clonevar other_inc_reg_d = p09                                                      // d
> ummy for other regural income
egen other_inc_reg = rowtotal(p10_1 p10_2), missing              // items 1 and 2 no
> t specified  // monthly
label var other_inc_reg "Other income - regural"


* other income - irregural
clonevar other_inc_irreg_d = p11
>       // dummy for other irregural income
egen other_inc_irreg = rowtotal(p12_1 p12_2 p12_3), missing  // items 1 and 2 not sp
> ecified  // annual
label var other_inc_irreg "Other income - irregural"


*******************************************************************************
                * Final touch
*******************************************************************************
/* Assigning hh-lvel incomes (such as capital, pension, rental, other income  etc to
>  individual hh members depending on different criteria) */

/* Importing household size */
merge 1:1 hhID using "${DataCleaningPath}\01-dataPrep-hhDemogVars", keepusing(hh_siz
> e)
drop _merge
merge 1:m hhID using "${DataCleaningPath}\02-dataPrep-hhMemberDemogVars", keepusing(
> memberID age) // importing memberID
drop _merge

* 1. Capital income: assignment to hh member is achieved by dividing it to hh size
rename cap_inc hh_cap_inc
gen cap_inc = hh_cap_inc/hh_size
drop hh_cap_inc
```

```
* 2. Pensions income
rename oth_pens hh_oth_pens                     // assignment to hh member is achiev
> ed by dividing it to hh size
gen oth_pens = hh_oth_pens/hh_size
drop hh_oth_pens

/* Counting the number of pensioners in the hh -- above age 60*/
bysort hhID: egen double num_pensioners = count(memberID) if age>59.5 & !missing(age
> )
        assert hh_size >= num_pensioners if !missing(num_pensioners)

rename lab_pens hh_lab_pens                     // assignment to hh member is achiev
> ed by dividing it number of pensioers
gen lab_pens = hh_lab_pens/num_pensioners
drop hh_lab_pens  num_pensioners


* 3. Rental income
rename rent_inc hh_rent_inc                     // assignment to hh member is achiev
> ed by dividing it to hh size
gen rent_inc = hh_rent_inc/hh_size
drop hh_rent_inc


* 4. Other regural and irregural incomes
rename other_inc_reg hh_other_inc_reg   // assignment to hh member is achieved by di
> viding it to hh size
gen other_inc_reg = hh_other_inc_reg/hh_size
drop hh_other_inc_reg

rename other_inc_irreg hh_other_inc_irreg       // assignment to hh member is achiev
> ed by dividing it to hh size
gen other_inc_irreg = hh_other_inc_irreg/hh_size
drop hh_other_inc_irreg


* Labelling variables
label var cap_inc "Capital income"
label var lab_pens "Pension income: from labor"
label var oth_pens "Pension income: other"
labe var rent_inc "Rental income"

                        ************************************************

/*  One more final touch of key variables - order matters! First annualize, then cap
>  upper outliers */
local finalTouchVars1 lab_pens oth_pens rent_inc other_inc_reg                  // m
> onthly income
local finalTouchVars2 cap_inc other_inc_irreg
>       // annual income

foreach z of local finalTouchVars1 {
ys_annualizing_values   `z', recall_in_days(30)         // annualizing monthly value
> s
}
*

****************************************
* other income total - regural plus irregural - is generated after annualizing bc th
> e recall periods for other regural and irregural incomes are  different
egen other_inc = rowtotal(other_inc_reg other_inc_irreg)
label var other_inc "Other income - regural and irregural"
****************************************
```

```
local finalTouchVars lab_pens oth_pens rent_inc cap_inc other_inc

sum `finalTouchVars'
ys_upper_outliers               `finalTouchVars'                                    /
> / setting upper outliers to MEAN+3SD
sum `finalTouchVars'

        /* Graph it */
        foreach z of local finalTouchVars{
        qui sum `z'
        local `z'_Mean = round(`r(mean)')
        histogram `z', fraction normal ti("`z' - Annual (Mean = ``z'_Mean')")  xline
> (``z'_Mean')
        }
*******************************************************************************
*******************************************************************************

local otherIncomeVars cap_inc lab_pens oth_pens rent_inc other_inc  // other_inc_reg
>  other_inc_irreg

local keepVars `transferVars' `otherIncomeVars'

des `otherIncomeVars', full
sum `otherIncomeVars'

// #3
// Saving the data
local keepVars county clid clusterID hhID memberID `transferVars' `otherIncomeVars'

/* To assign Yes/No value lable to dummy variables */
local yesnoVars

order `keepVars'
keep `keepVars'

sort  hhID
codebook hhID

* saving details
local dta_name           04-dataPrep-otherIncomes-transfersAndPensions
local dta_note          "Income: transfers and pensions"

include "${DataCleaningPath}\i-dta-savingDetails.do"    // saving the data with addi
> tional info on notes, labels, etc
*************************


*******************************************************************************
// #4
// Post-data-saving checking
/* Checking for the hh size (we count only those who are present) */
bysort hhID: egen double hh_size_test = count(memberID)
        assert hh_size == hh_size_test // if !mi(ind_weight)

order hhID memberID hh_size hh_size_test

*******************************************************************************
```