Statistics/Data analysis

```
//  Program:    03-dataPrep-incomeVars.do
//  Task:       cleans individual and household incomes using Kenyan survey data (20
> 15/16 KIHBS)
//  Project:    Kenya Fical MicroSim
//  Author:     Yared Seid - 2023Jan15


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


// #2
// Generating vars

* Generating IDs
codebook 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 a cluster
qui tostring b01,       gen(b01_STR) format(%02.0f)                              // h
> oudehold member id, within a household

label var clusterID "Cluster ID"

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

gen memberID = hhID + b01_STR
label var memberID "Personal ID"
isid memberID


// Earnings
merge 1:1 memberID using "${DataCleaningPath}\02-dataPrep-hhMemberDemogVars", keepus
> ing(age female)       // importing age and gender
drop _merge

********************************************************************************
/* * Here is the label for whether a hh member is working or not on both primary and
>  secondary activity
                   1 Paid employee (outside hh )
           2 Paid employee ( within hh )
           3 Working employer...
           4 Own-account worker
           5 Members of producers? cooperatives
           6 Contributing family worker
           7 Apprentice
           8 Volunteer
          96 Other (specify)
*/

* calculate earning only for working age?
gen working_age = (age>=17.5 & age<=65.4) if !mi(age)           // assuming retireme
> nt age = 65
label var working_age "=1 if a hh member's age is within working age range, i.e., [1
> 8-65]"
```

```
/* Paid worker status on primary activity*/
gen earning_member_pri = .
replace earning_member_pri = 1 if [d10_p==1 | d10_p==2 | d10_p==3 | d10_p==4]
replace earning_member_pri = 0 if [d10_p==5 | d10_p==6 | d10_p==7 | d10_p==8]
replace earning_member_pri = .o if [d10_p==96]

/* Paid worker status on secondary activity*/
gen earning_member_sec = .
replace earning_member_sec = 1 if [d10_s==1 | d10_s==2 | d10_s==3 | d10_s==4]
replace earning_member_sec = 0 if [d10_s==5 | d10_s==6 | d10_s==7 | d10_s==8]
replace earning_member_sec = .o if [d10_s==96]

/* Piad worker status for a hh member*/
egen earning_member = rowtotal(earning_member_pri earning_member_sec), missing
replace earning_member =1 if earning_member==2 // =2 when a memeber is earning both
> in primary and secondary activities
replace earning_member =.o if missing(earning_member) & [earning_member_pri ==.o | e
> arning_member_sec ==.o]

assert earning_member==0 | earning_member==1 if !missing(earning_member)
*********************************************************************************

* earning type
gen earner_wage = 1 if [d10_p==1 | d10_p==2 | d10_p==3] | [d10_s==1 | d10_s==2 | d10
> _s==3]
gen earner_entr_inc = 1 if [d10_p==4] |  [d10_s==4]

foreach var in earner_wage earner_entr_inc {
        replace `var' = .o if missing(`var') & earning_member ==.o  // other/please
> specify indicator
}

tempvar checkVar
egen `checkVar' = rowtotal(earner_wage earner_entr_inc), missing
assert `checkVar'==1 | `checkVar'==2 if !missing(`checkVar')
assert earning_member == 1 if [`checkVar'>0 & `checkVar'<=2]


* earning type by gender
gen earning_member_female = earning_member*female
gen earner_wage_female = earner_wage*female
gen earner_entr_inc_female = earner_entr_inc*female

label var earning_member "=1 if hh member earns any income"
label var earning_member_female "=1 if FEMALE hh member earns any income"
label var earner_wage "=1 if hh member is wage earner"
label var earner_wage_female "=1 if FEMALE hh member is wage earner"
label var earner_entr_inc "=1 if hh member is enterpreneur income earner"
label var earner_entr_inc_female "=1 if hh FEMALE member is enterpreneur income earn
> er"

* top coding? possibly in d26 - replace with maximum non-top coded value
sum d26
*local max = r(max)
sum d26 if !inrange(d26,999998,999999)
replace d26 = r(max) if inrange(d26,999998,999999)

gen inc_pri = d26
>       // monthly primary income
gen inc_sec = d43
>       // monthly secondary income
```

```
* wage
egen wage = rowtotal(inc_pri inc_sec) if earner_wage==1, missing                // m
> onthly wage

* business/enterpreneur income
egen entr_inc_ind = rowtotal (inc_pri inc_sec) if earner_entr_inc==1, missing   // m
> onthly business income

keep clid clusterID hhID memberID wage entr_inc_ind working_age earning_member earni
> ng_member_female earner_wage earner_wage_female earner_entr_inc earner_entr_inc_fe
> male
isid hhID memberID

order earning_member earning_member_female earner_wage earner_wage_female earner_ent
> r_inc earner_entr_inc_female, last

tempfile income1
save `income1', replace


// Loading the data
use "${dataSources}\Kenya_2015_16_KIHBS\Household_Enterprises", clear           // h
> ousehold-level data for hh-level enterpreneur income

// Generating vars

* Generating IDs
codebook 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 a cluster

label var clusterID "Cluster ID"

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

codebook hhID

* hh-level enterpreneur income
clonevar inc_type = n03_en                     // income generating activity type a
> s in beauty shop, etc

bysort hhID: egen entr_inc_hh = total(n07_ks), missing  // n07_ks = inc by activity
> type

keep clid clusterID hhID  entr_inc_hh
bysort hhID: keep if _n==1
isid hhID

tempfile income2
save `income2', replace


// Loading the data - crop income
use "${dataSources}\Kenya_2015_16_KIHBS\Agriculture output (L1_L20)", clear
>       // Crop data


// 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"                   // not unique ID since hhs may produ
> ce multiple crops

* Crop income
clonevar crop_type = l02_cr                                     // types of crop pro
> duced by hhs
clonevar agri_inc_croptype = l12                        // income from each crop typ
> e

bysort hhID: egen agri_inc_crop = total(agri_inc_croptype), missing             // s
> um of income from each crop type // annual


tempfile income_crop
save `income_crop', replace


// Loading the data - Livestock income
use "${dataSources}\Kenya_2015_16_KIHBS\Livestock (M1_M15)", clear
>     // Livestock data


// 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"                   // not unique ID since hhs may produ
> ce multiple livestock

* Livestock income
bysort hhID: egen agri_inc_animal = total(m06), missing                 // sum of in
> come from selling livestock // annual

tempfile income_livestock
save `income_livestock', replace

* merging crop income and livestock income
use `income_crop', replace

merge m:m hhID using `income_livestock', keepusing(agri_inc_animal)
drop _merge

egen agri_inc_hh = rowtotal(agri_inc_crop agri_inc_animal), missing  // annual
```

```
order hhID /*agri_inc_crop agri_inc_animal*/ agri_inc_hh
sort hhID

/* For agricultural income: I believe missing value means the hh didn't produce crop
> , livestock, etc (may be because its an urban hh). Zero values, however, may denot
> e that it is an agricultural rural hh but didn't produce (some or all) crops, live
> stocks, etc at all. */

keep clid clusterID hhID agri_inc_hh
bysort hhID: keep if _n==1
isid hhID

tempfile income3
save `income3', replace                          // agricultutal income data


// Merge all cleaned data
use `income1'

merge m:1 hhID using `income2', keepusing(entr_inc_hh)          // importing hh-leve
> l entr_inc
drop _merge

merge m:m hhID using `income3', keepusing(agri_inc)             // importing hh-leve
> l angricultural income
drop _merge

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

sort hhID memberID

order working_age earning_member earning_member_female earner_wage earner_wage_femal
> e earner_entr_inc earner_entr_inc_female, last


********************************************************************************
                * Final touch
********************************************************************************
/* Assigning hh-lvel incomes (such as enterpreneur income, agricultural income, etc
> to individual hh members depending on their different income earning status) */

* 1. Enterpreneur income
* Number of enterpreneur income earners in the hh
by hhID: egen hh_entr_inc_devider = sum(earner_entr_inc) if earner_entr_inc==1

/* entr_inc is sum of entr_inc_ind (which is calculated earlier) and entr_inc_ind2 (
> which is calculated below by assigning hh-lvel income (entr_inc_hh) to enterpreneu
> r income earning members )  */
gen entr_inc_ind2 = entr_inc_hh/hh_entr_inc_devider                     // hh entr_i
> nc is assigned to members
egen entr_inc = rowtotal(entr_inc_ind entr_inc_ind2), missing   // member ind income
>  + assigned income to member

* 2. Agricultural income
* Number of income earners in the hh  -- since ther is no indicator for agi inc earn
> er
by hhID: egen hh_inc_devider = sum(earning_member) if earning_member==1

gen agri_inc = agri_inc_hh/hh_inc_devider  // member agri inc = hh agri inc devided
> # of earning members
```

```
* Labelling variables
label var wage "Wage"
label var entr_inc "Enterpreneur income"
label var agri_inc "Agricultural income"

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

/* One more final touch of key variables - order matters! First annualize, then cap
> upper outliers */

local finalTouchVars1 wage entr_inc                     // monthly income
local finalTouchVars2 agri_inc                          // annual income
local finalTouchVars `finalTouchVars1' `finalTouchVars2'

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

sum `finalTouchVars'
ys_upper_outliers       `finalTouchVars'             // setting upper outliers to ME
> AN+3SD
ys_lower_outliers       entr_inc                                // setting lower out
> liers to MEAN-3SD since entr_inc_hh has -ve values
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')
        }
********************************************************************************
********************************************************************************


// #3
// Saving the data
local keepVars /*county*/ clid clusterID hhID memberID wage entr_inc agri_inc ///
working_age earning_member earning_member_female earner_wage earner_wage_female earn
> er_entr_inc earner_entr_inc_female

/* To assign Yes/No value lable to dummy variables */
local yesnoVars          working_age earning_member earning_member_female earner_wag
> e earner_wage_female earner_entr_inc earner_entr_inc_female


order `keepVars'
keep `keepVars'

sort  hhID memberID
codebook hhID memberID


* saving details
local dta_name           03-dataPrep-incomeVars
local dta_note          "Income variables"
```

```
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

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