Ethiopia Poverty Measurement Training

Day 5: Workflow Management, Documentation and Quality Control

Collaboration Tools

Github and shared folders

Example

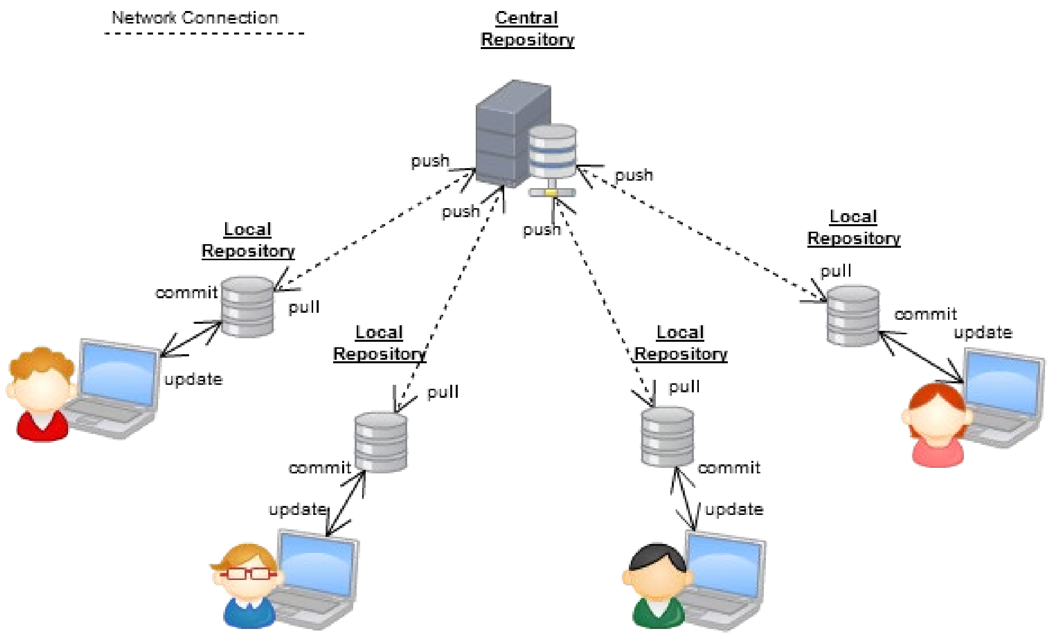
- This way of using Github and shared folders on OneDrive was developed by the the team (WB+NBS)
- Implemented successfully in a number of countries since
- Provides high level of transparency
- Allows for any team member to access and run the code at any point
- Tracks changes to the code in a very easy to use way

GitHub

- Github is a software that helps tracking and managing changes to code
- Allows large teams to efficiently work together on code for data processing and analysis
- It ensures reproducibility
- Convenient tool for effective coding on any analytical project
- Can access code via the website, or a desktop application [or integrated into other coding / software development platforms]

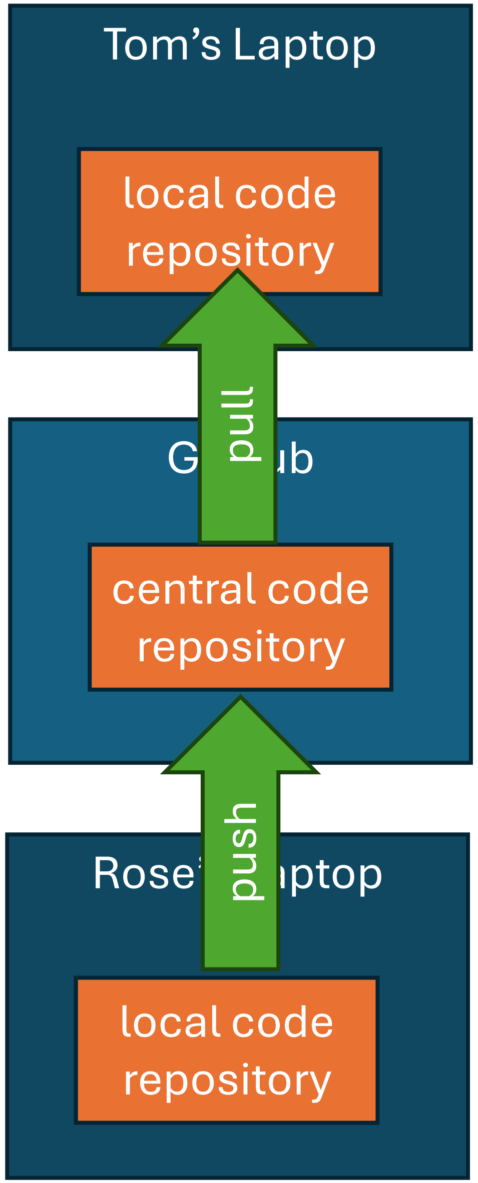
GitHub Model

- There is a central code repository
- Every user also has a local code repository (just in a local folder on your own laptop)
- When a user has made changes / additions they want to share, they can “push” their code to the central repository
- Every user can “pull” and get the most recent code submitted by all the others
- [show example]



Shared Folders on OneDrive

- Store all data (inputs, temporary files and output) in shared folder on OneDrive
- One input folder “read only”
  - contains the data from the survey (updated only by one person when new survey data becomes available)
  - contains any auxiliary data (NSU conversion factors, calorie/food composition tables, CPI data etc)
  - should only have anonymized data for sharing with WB staff/consultants
- Individual folders for each person working on the code
- Each individual folder has both **temp** and **output** folders
- If everyone pulls the most recent code from the central repository, will all get the same outputs



Tom’s Laptop

local code repository

pull

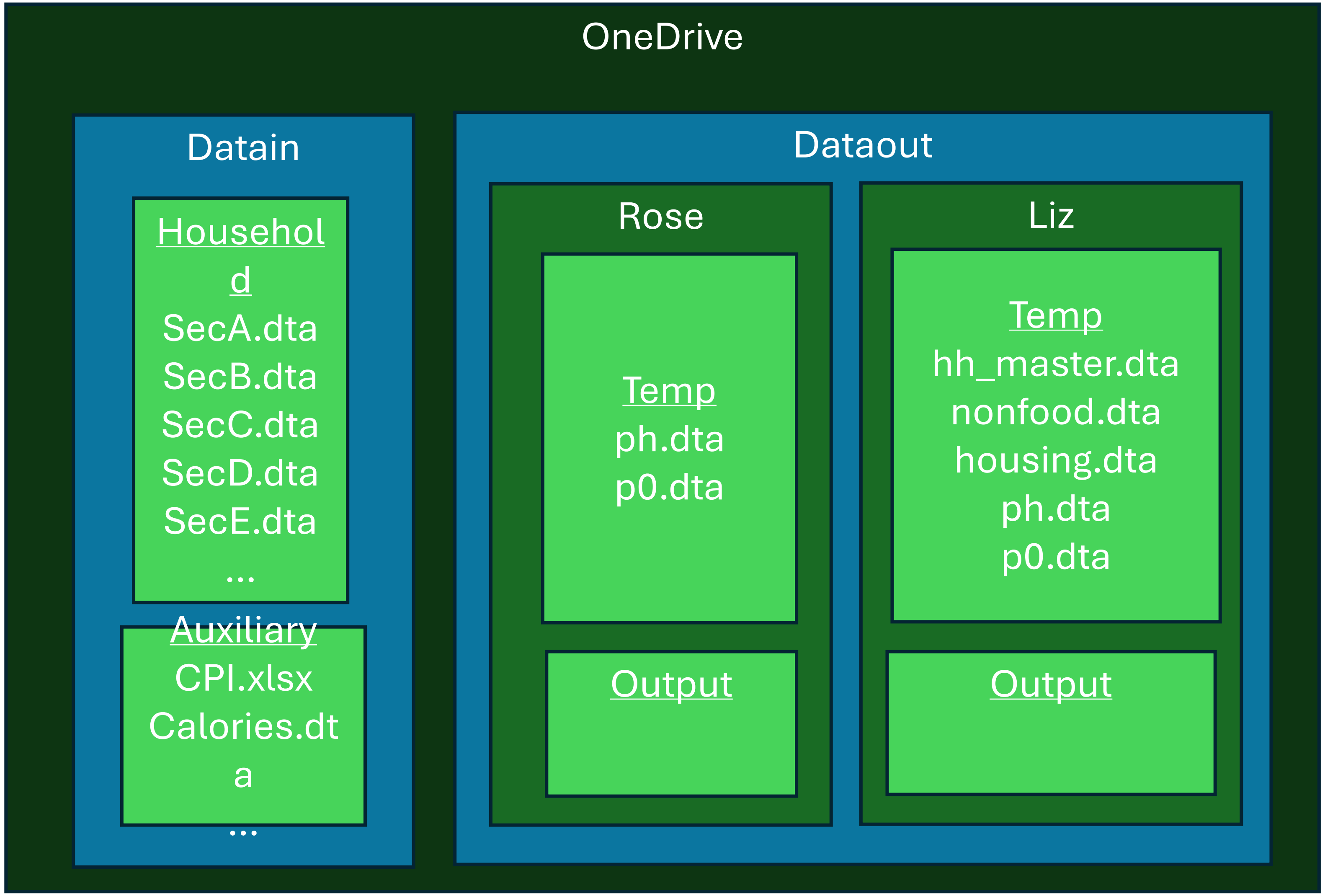
G ub

central code repository

push

Rose aptop

local code repository



OneDrive

Datain

Household

SecA.dta

SecB.dta

SecC.dta

SecD.dta

SecE.dta

…

Auxiliary

CPI.xlsx

Calories.dta

…

Dataout

Rose

Temp

ph.dta

p0.dta

Output

Liz

Temp

hh_master.dta

nonfood.dta

housing.dta

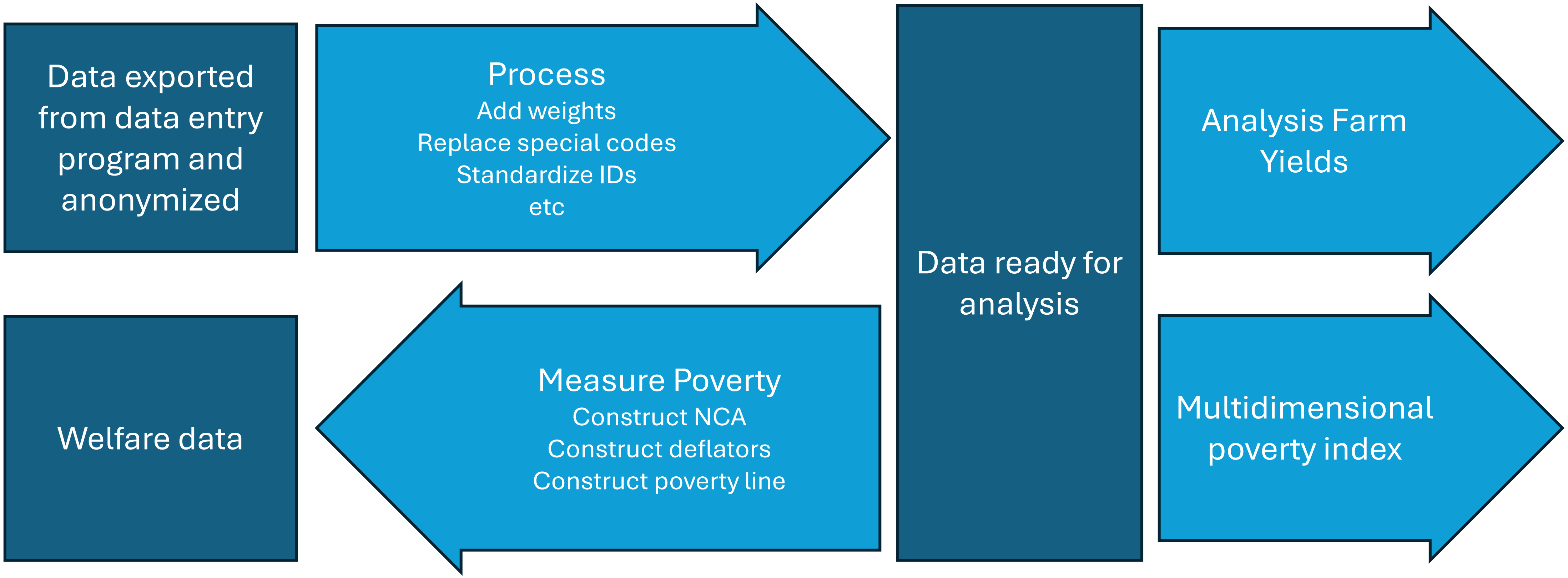
ph.dta

p0.dta

Output

Organizing Workflow

Separate Initial Data Processing from Poverty Measurement



Data exported from data entry program and anonymized

Process
Add weights
Replace special codes
Standardize IDs
etc

Data ready for analysis

Analysis Farm Yields

Measure Poverty
Construct NCA
Construct deflators
Construct poverty line

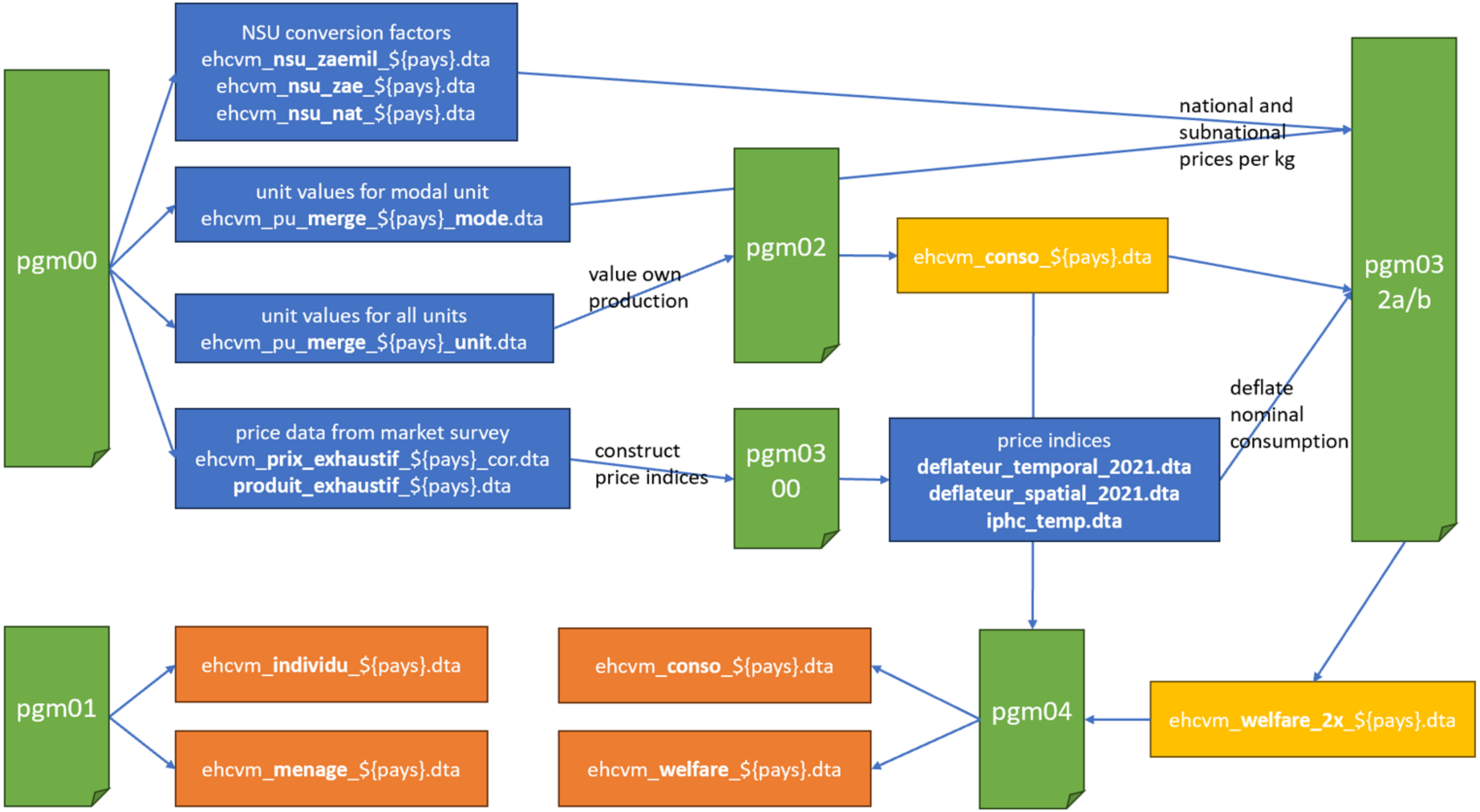
Welfare data

Multidimensional poverty index

Organizing Code

- Have “initialize” do file that
  - sets global macros for file locations
  - set global macros for parameters and methodology choices [show GNQ example]
  - defines version of Stata used
  - provides links to install any nonstandard programs used
  - external values (inflation, interest rate etc) with sources
- First step is to construct a master hh dataset households completing the survey, with household size, weights, urban/rural, admin1, admin2 etc
- Then programs for each step of the analysis
  - For each program, clearly specify what input/temp datasets it uses, what it does, and what output/temp datasets it constructs
- Can have a master program that erases all output and temporary datasets, and runs all the steps in order

Figure 1: Flowchart of Programs and Main Intermediate Datasets


pgm00
NSU conversion factors
ehcvm_nsu_zaemil_${pays}.dta
ehcvm_nsu_zae_${pays}.dta
ehcvm_nsu_nat_${pays}.dta
unit values for modal unit
ehcvm_pu_merge_${pays}_mode.dta
unit values for all units
ehcvm_pu_merge_${pays}_unit.dta
price data from market survey
ehcvm_prix_exhaustif_${pays}_cor.dta
produit_exhaustif_${pays}.dta
value own production
pgm02
ehcvm_conso_${pays}.dta
national and subnational prices per kg
pgm03 2a/b
construct price indices
pgm03 00
price indices
deflateur_temporal_2021.dta
deflateur_spatial_2021.dta
iphc_temp.dta
deflate nominal consumption
pgm01
ehcvm_individu_${pays}.dta
ehcvm_menage_${pays}.dta
ehcvm_conso_${pays}.dta
ehcvm_welfare_${pays}.dta
pgm04
ehcvm_welfare_2x_${pays}.dta

Organizing Each Program

- Clearly specify inputs, purpose and outputs at the top
- Structure your code using outline or other format to make it easier to refer to lines of code
    - See the way templates for exercise are structure
- Comment!
    - Aim for one comment for ever 2-3 lines of code
    - Assume reader knows how Stata works but wants to understand what you are doing and why
- Do not change the values of the original variables that correspond to questions on the questionnaire. Construct new variables, and then refine those (to deal with outliers, other mistakes etc) as needed.
- Use tempfiles (rather than saving to the temp folder) for datasets that have little intrinsic value and are created and then used quickly once
- Avoid lines of code that don't do anything
- To check things, use assert statements, including with merges

Plan Ahead

program	input dataset	temp dataset produced	values of item
2 food.do	Section C	food.dta	1-20, same as initial coding of food items and 21-25, 20 plus initial code of FAFH
3 nonfood general.do	Section B	nonfood.dta	1001-1010, 1000 + initial code of nonfood items
4 health and education.do	Section A	healtheduc.dta	1101-1120
5 durable goods.do	Section D	durables.do	1201-1210, 1200 + initial code of durable goods
6 housing.do	Section E	housing.do	1300

- Organization of code: what do files will you have and what will they do
- Intermediate datasets: what are they, what are they called, what variables do they have
- Unique item code system

Suggested File Structure

datain: the final versions of the main datasets that will be made publicly available. This can also include any other auxiliary data that the NSO is willing to make public alongside, such as a dataset of NSU conversion factors, calorie contents, CPI data etc. The .do files should not change any of these datasets

auxiliary: any other data needed that will NOT be made publicly available. Try to minimize this

programs: the other .do files which will be called by the master .do file

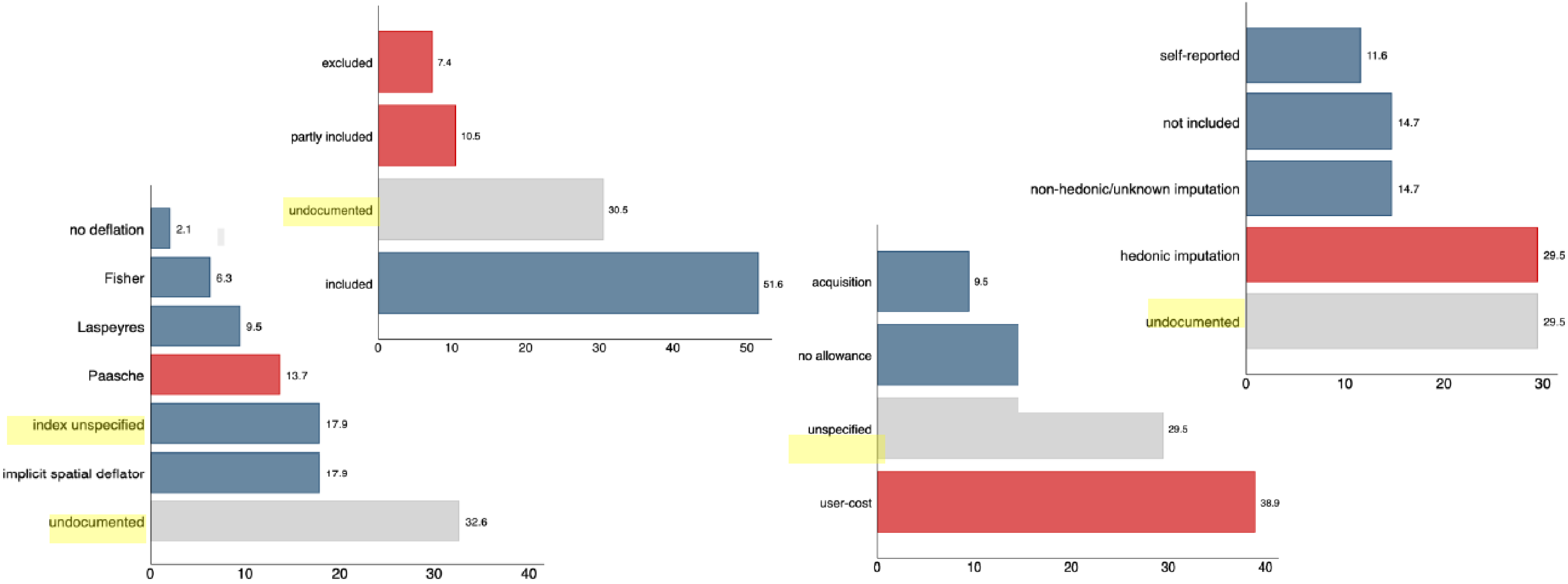
temp: temporary datasets

logs: log files

dataout: final results. This should include a household-level dataset with the welfare aggregate and poverty line as well as various other characteristics of the household. It can include other datasets that are considered final output (such as household-item level consumption data)

Documentation

Let’s End “undocumented” and “unspecified”



Poverty Measurement Database

- Project of D4G, led by me with Gabriel and Sergio, STC Rostand
- Aims to be database of key methodological choices for poverty measurement.
- Have draft database currently in Excel, working on building web interface
- For now, STC(s) are filling this in for the most recent survey for each country, using existing documentation
  - Many unanswered questions
- Better documentation as discussed total / using the templates the global unit has will make this much easier
- PMD is not just for documentation after the fact
- Can be used as a tool to organize information on what was done for the last survey for the country, and discuss before what you want to do (with your TTL/lead economist, NSO etc)

Methodology Note

- Narrative of construction of consumption aggregate and poverty line.
- Include information on basically all the questions we've discussed this week
  - Components of aggregate
  - Adjustments for household size and prices
  - Construction of poverty line
- Document any major issues with the data, significant use of imputation

- Results of hedonic regression(s) for housing, including number of observations, R2
- Depreciation rates for durable goods
- Values of the spatial and/or temporal deflators
- Composition of the basket
- Value of food poverty line
- Value of total poverty line

Reproducibility

Replicability

- Archive data and code to be able to fully replicate household level data (final welfare aggregate and poverty status) and any published results
    - Include any programs (`outdetect`, `egenmore` etc) that are not part of standard Stata distribution
- Share input data and do files and see if someone else can fully replicate the results
- Stata command `cf` (for compare file) is very useful to check whether the output files generated as exactly the same
    - Note: datasets need to be sorted the same way before they can be compared
- This is a great task for an STC who is not a Stata or poverty measurement expert, the results should be replicable by any one with a basic knowledge of Stata
- Submit final package to Reproducible Research Repository
    - https://reproducibility.worldbank.org/index.php/home

QER

Check your own work (or your STC’s work, or a colleague’s work)

Self QER Check #0: Check for Domain Mistakes

- Check your own work!
  - constructed temporary and final datasets
- Are the idson each dataset correct?
  - use command `isid which` checks for duplicates, missing values
- Are all the constructed variables nonmissing / nonzero / positive as they should be?
  - use command `assert`
- Check that the various identifiers follow the correct pattern
  - every household in a PSU should have the same value of *rural* and *region* etc.

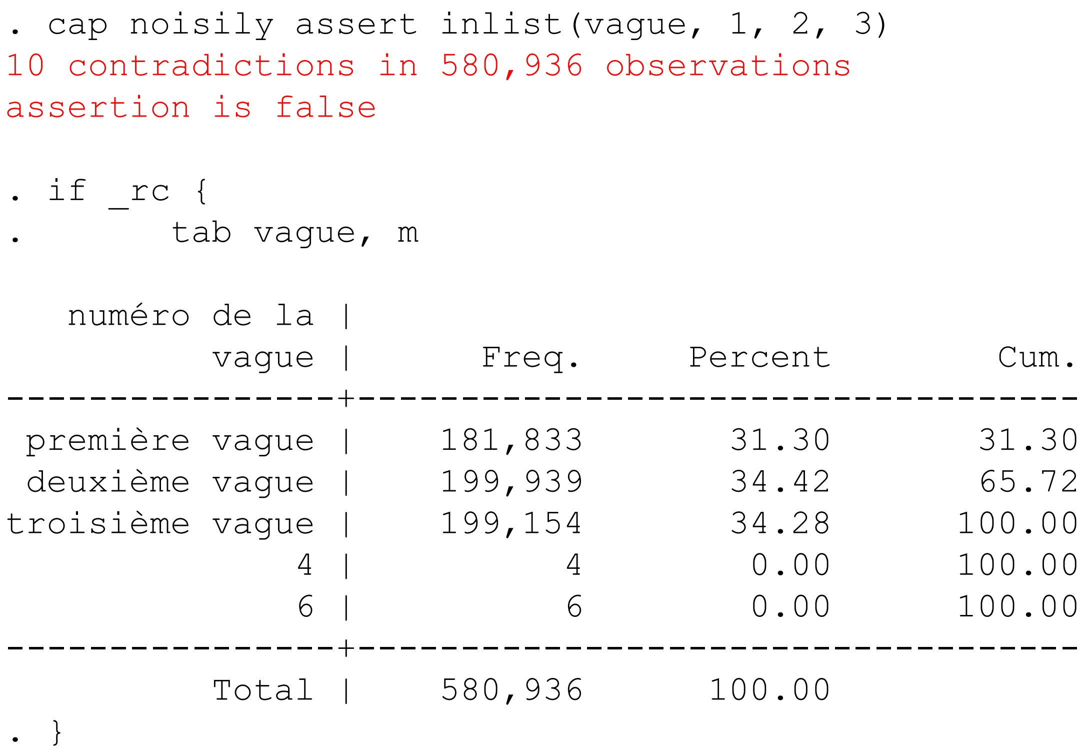
Examples

```
/* ---- 0. Very basic checks ---------------------------
--------------------- */


//  a. identifiers

isid hhid

isid grappe menage


//  c. check key variables

assert year == 2022

assert country == "CMR"

assert inlist(vague,  1, 2, 3)

assert inlist(milieu, 1, 2)

cap noisily assert hhweight < . & hhweight > 0

cap noisily assert inrange(hhsize, 1, 100)
```

```
cap noisily assert dali < . & dali > 0

cap noisily assert dnal < . & dnal > 0

cap noisily assert dtot < . & dtot > 0

cap noisily assert pcexp < . & pcexp > 0


//  d. consistency

bys grappe: assert zae == zae[1]

bys grappe: assert region == region[1]

bys grappe: assert milieu == milieu[1]
```

Capture, noisily and _rc

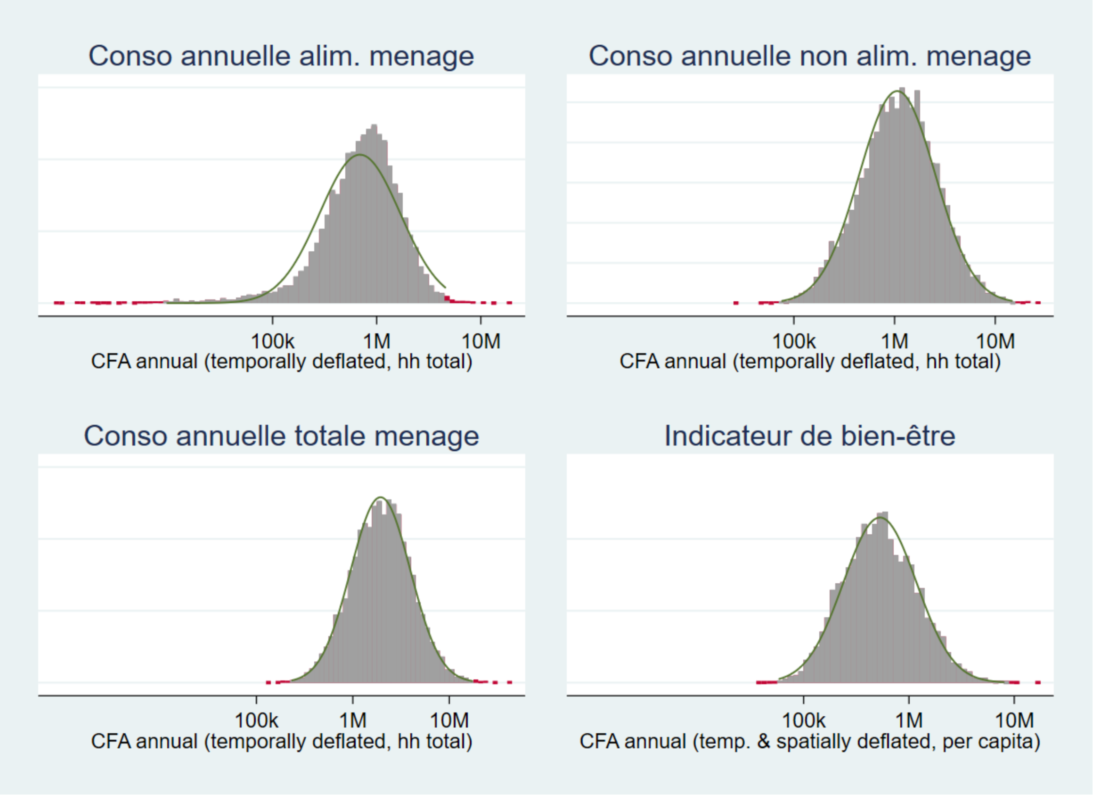
- The prefix `capture` can be placed in front of commands (such as `isid` and `assert`) to make the do file keep running even if those command fail
- Adding `noisily` makes the command still display the error messages
- When I do QERs, I use them a lot so I can run all the checks
- When checking your own work, use them sparingly, it’s best to correct any mistakes they identify as you identify them



```
. cap noisily assert inlist(vague, 1, 2, 3)
10 contradictions in 580,936 observations
assertion is false

. if _rc {
.         tab vague, m

   numéro de la |
          vague |      Freq.     Percent        Cum.
----------------+-----------------------------------
 première vague |    181,833       31.30       31.30
 deuxième vague |    199,939       34.42       65.72
troisième vague |    199,154       34.28      100.00
              4 |          4        0.00      100.00
              6 |          6        0.00      100.00
----------------+-----------------------------------
          Total |    580,936      100.00
. }
```

Self QER Check #1: Overall Distribution

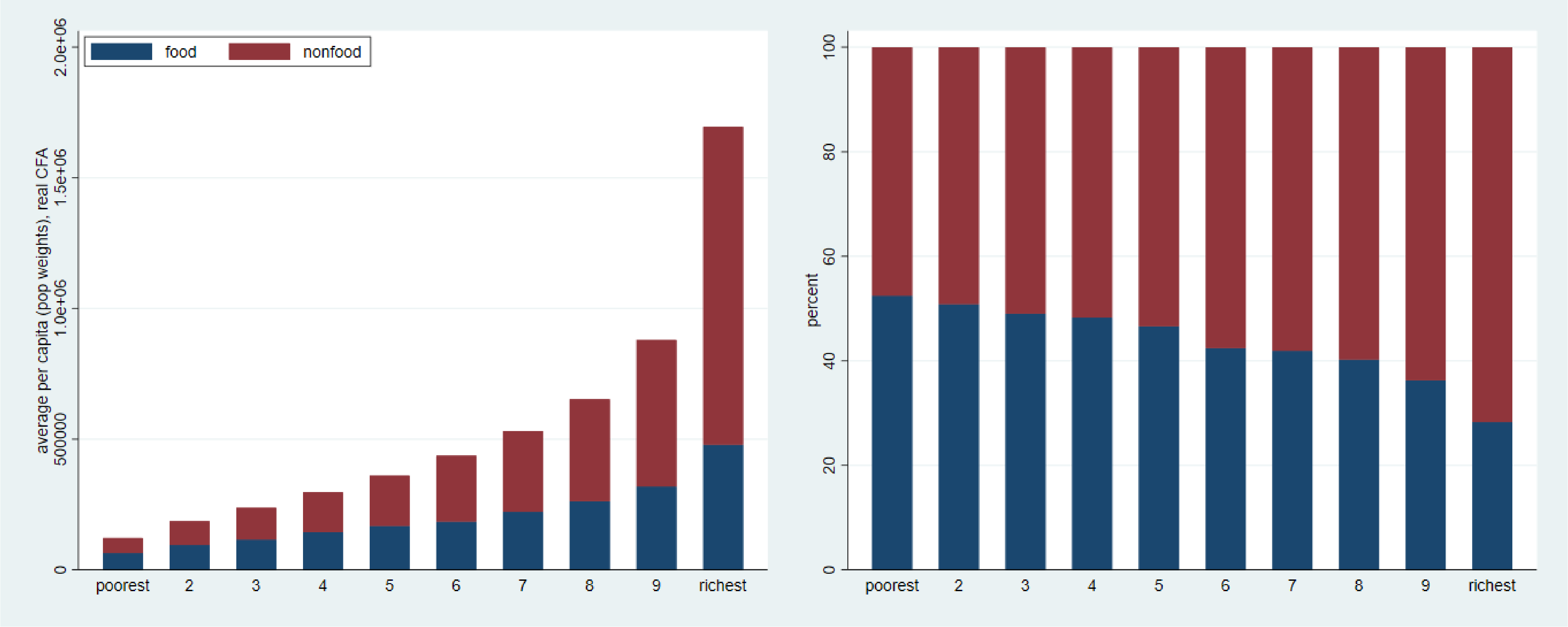
- Overall distribution of per capita (or per adult equivalent) real consumption, total, food and nonfood
- All should be basically log normal
- If they are not:
  - do not make any corrections at this level
  - try to figure out where outliers or weirdness in the distribution is coming from and adjust there
  - if you can't, you may need to drop outlier households and reweight



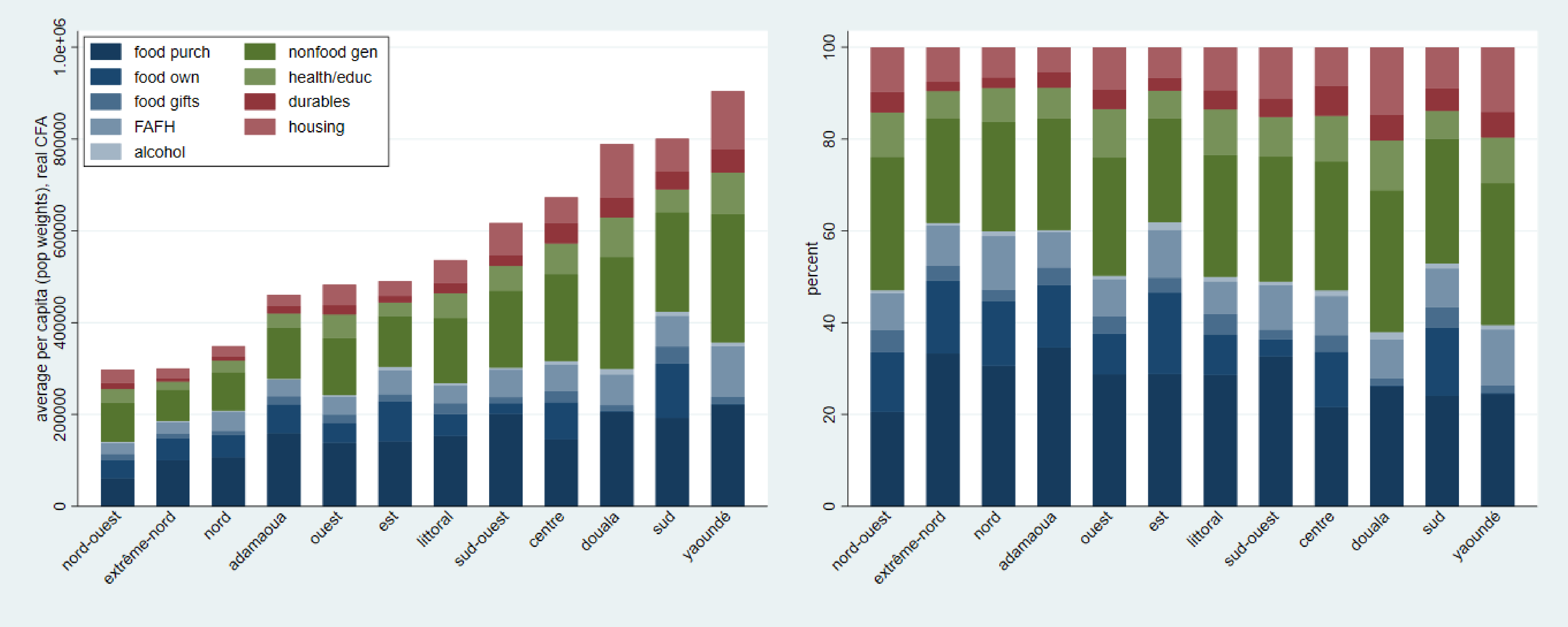
Self QER Check #2: Breakdown of consumption

- Breakdown of consumption (at least food and nonfood, ideally more detailed) by decile, in totals and shares
- Does share of food in total consumption decrease as consumption increases?
- Break down by section of questionnaire / means of construction (most likely to help find coding errors)
  - food from own production
  - food from purchases
  - food from gifts/other
  - FAFH, general nonfood
  - education and health services
  - durable goods use value
  - Housing
- Breakdown by first digit COICOP classification
- Can also look at by urban/rural and/or admin 1 etc

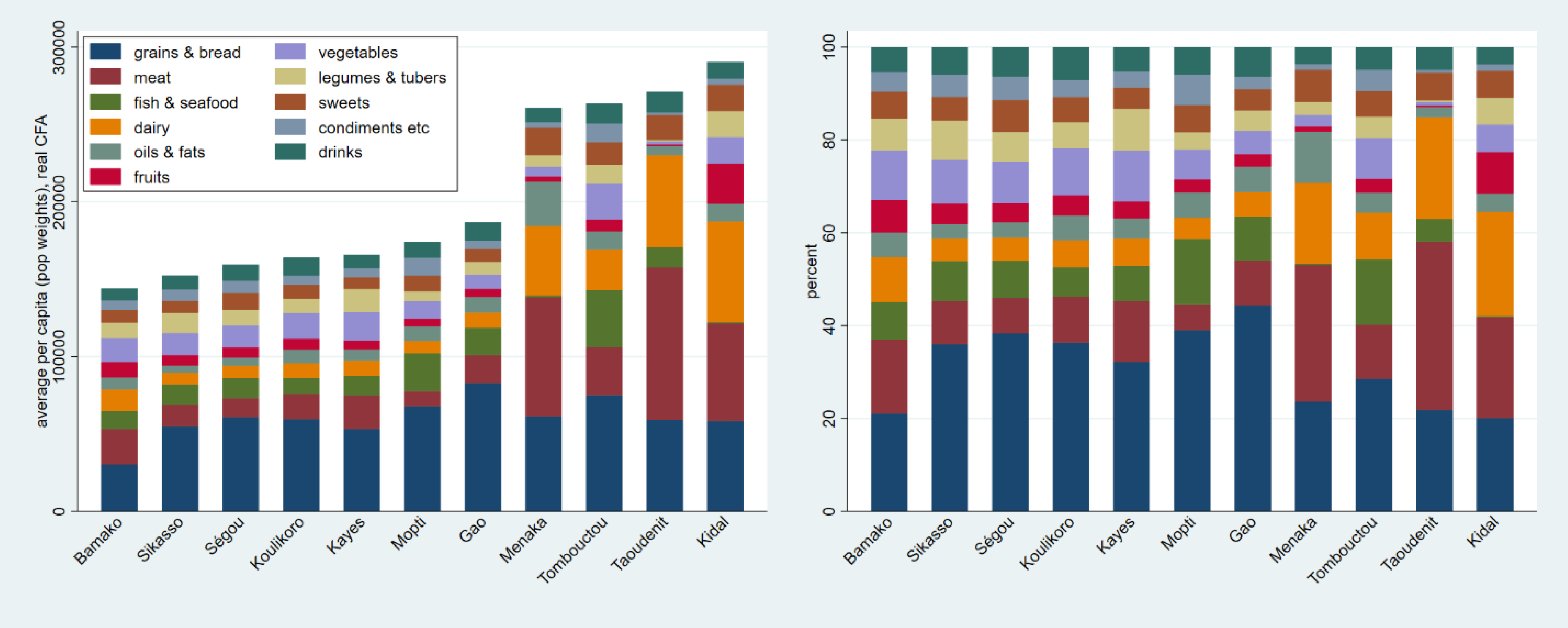
Self QER Check #2



Self QER Check #2



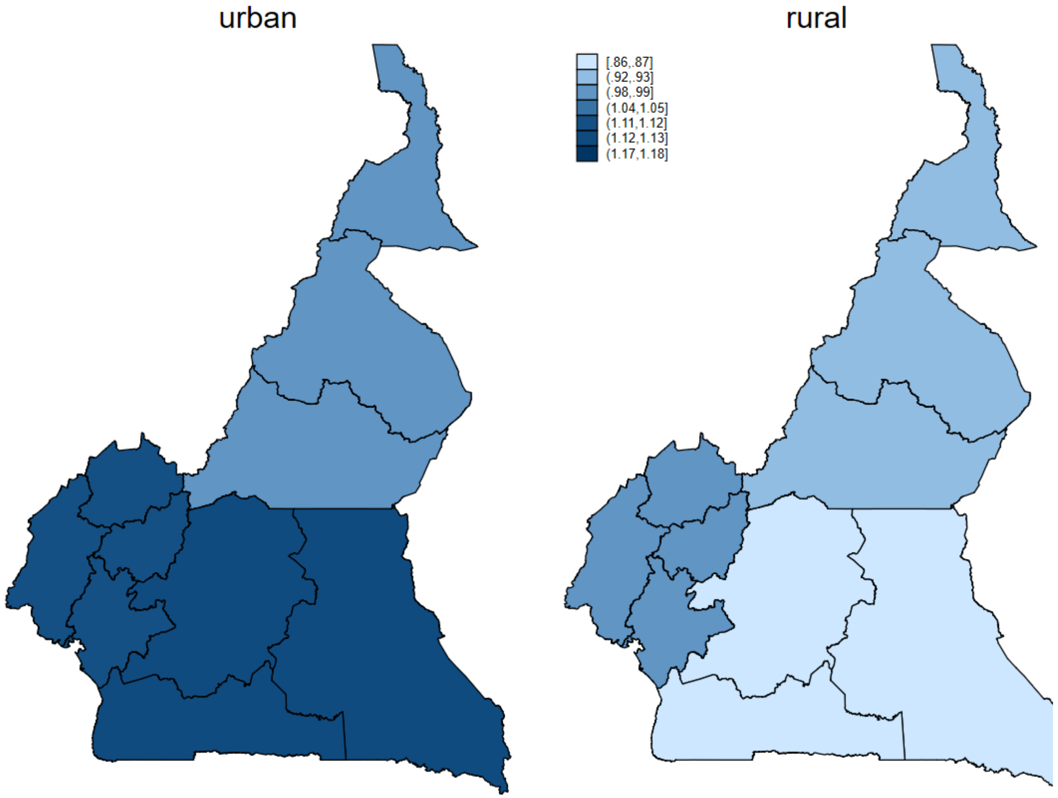
Self QER Check #2



Self QER Check #3: Deflators

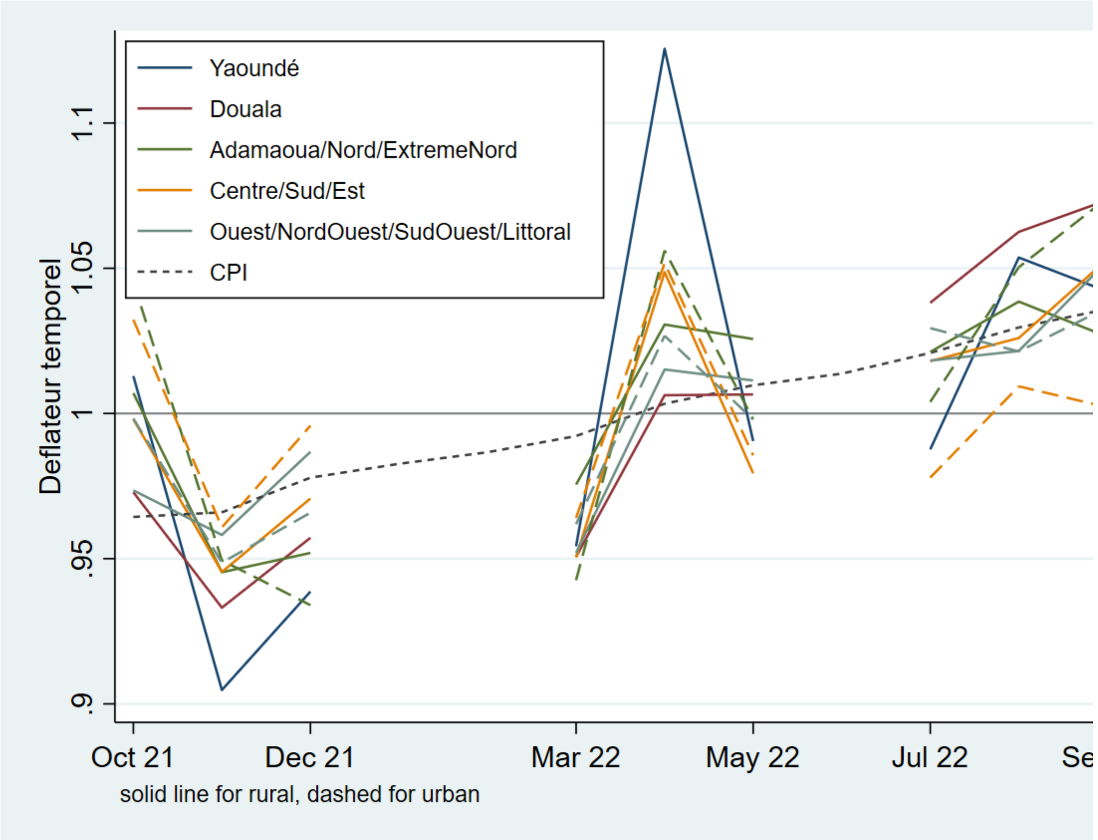
Values of temporal and spatial price deflators, at the level at which they were constructed.

(If they are constructed at a very granular level – PSU or household – show the distribution including extreme outliers, by urban/rural, admin 1 etc).



Self QER Check #3: Deflators

- Concerning

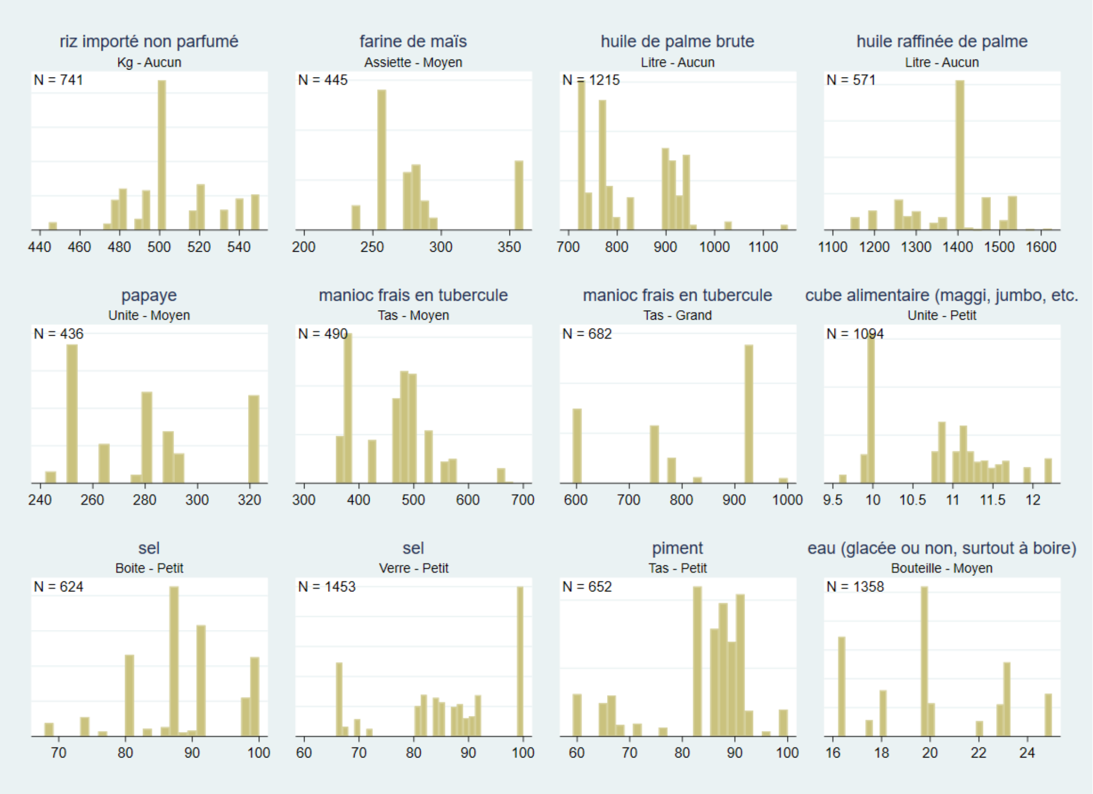


Self QER Check #4: Prices

Analysis of most influential prices for valuing own production.

Take say the 10 food items with the highest total value from own production and look at the price actually used to value them

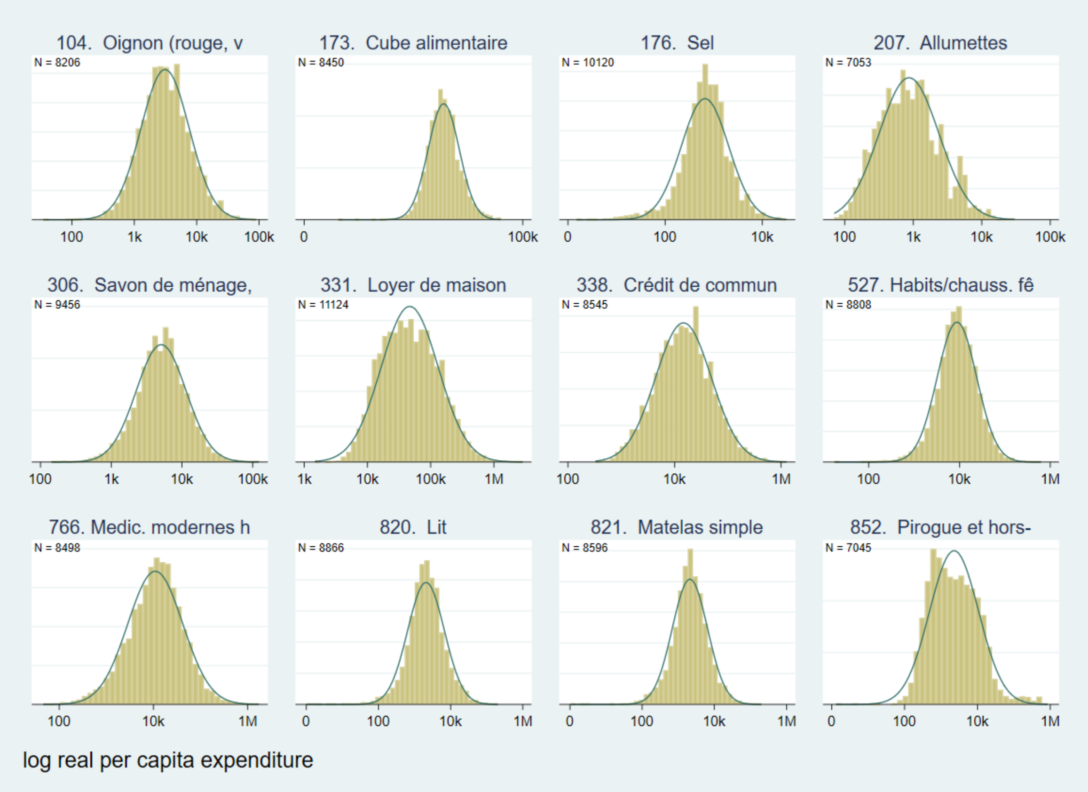
Either show exactly all the prices (if these are constructed at a high enough level), or the distribution of prices, including extremes, by urban/rural, admin 1



Self QER Check #5: Household-item level data

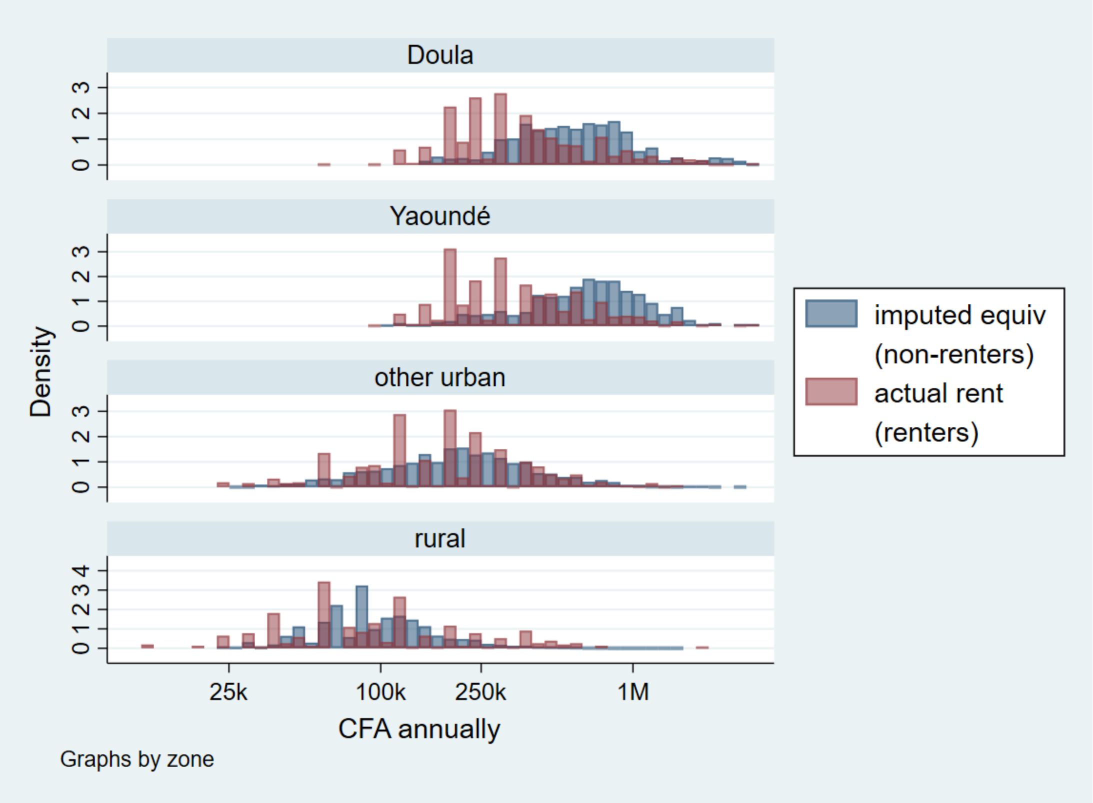
Check item-household-level data does not have any negative values, extremely small values, or extremely large values

Look at top items by total value of consumption and/or number of observations

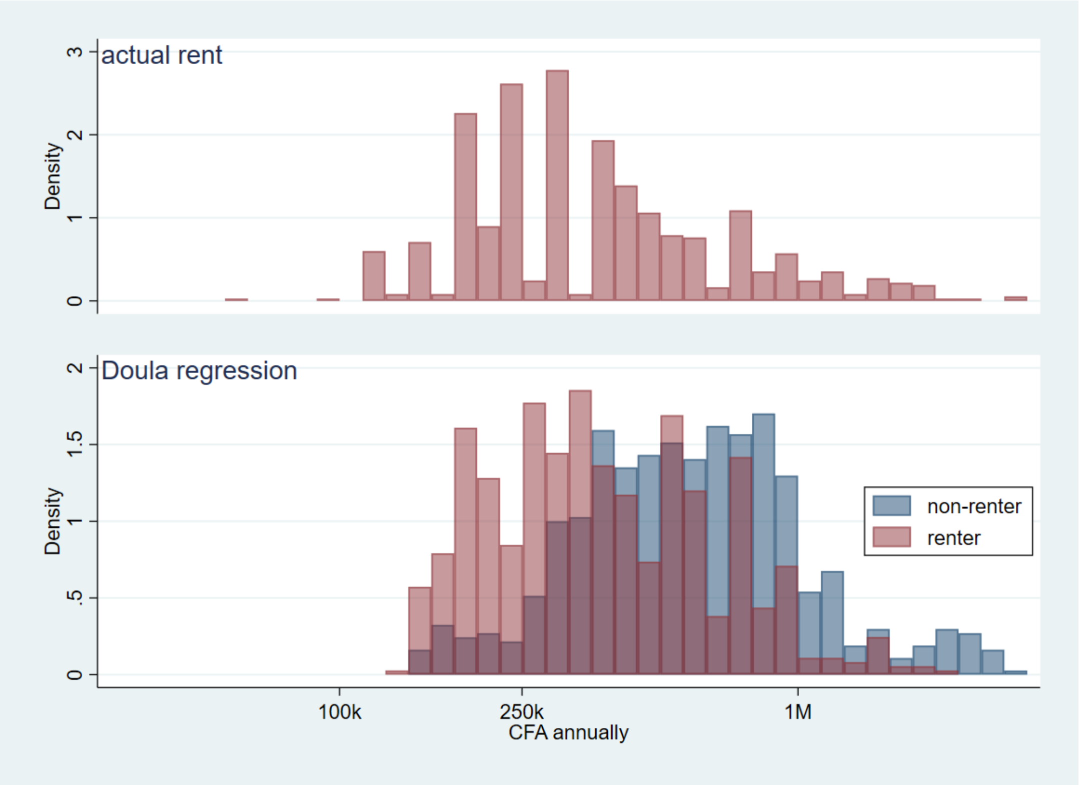


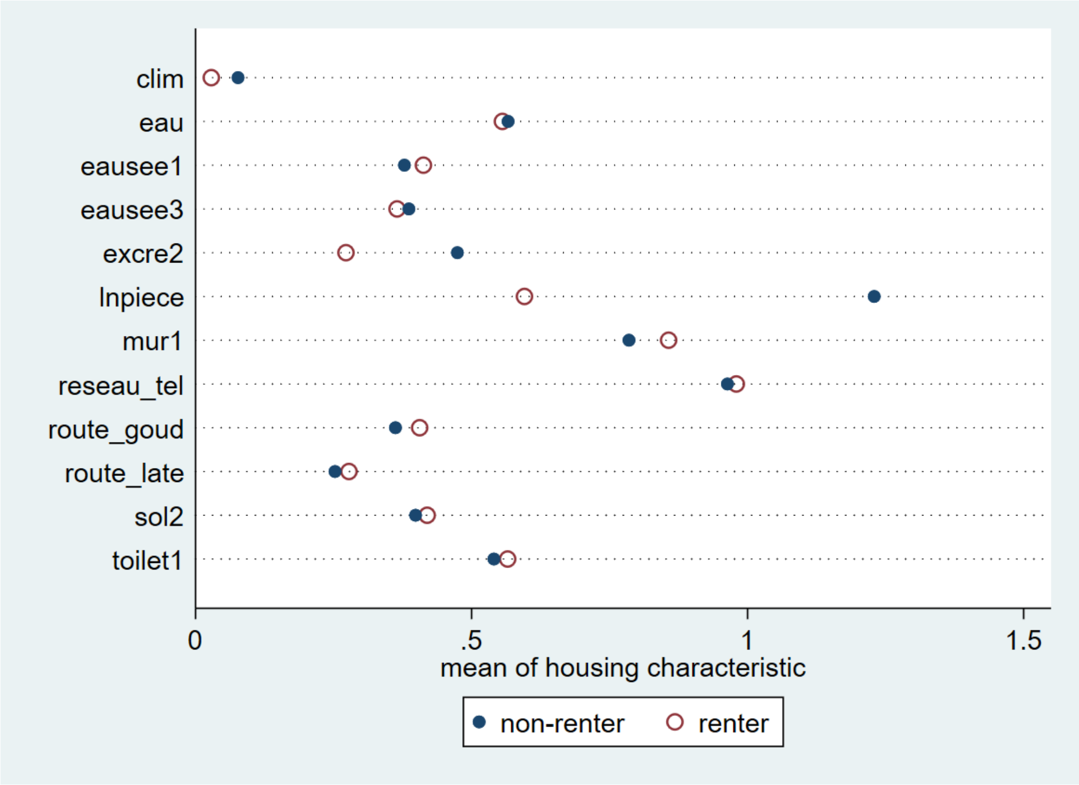
Self QER Check #5: Housing Model

- Check housing model.
- If hedonic regression is used, report for [each] model:
  - percent of rents
  - number of observations
  - R-squared



Self QER Check #5: Housing Model


actual rent
Density
Doula regression
Density
non-renter
renter
100k
250k
1M
CFA annually


clim
eau
eausee1
eausee3
excre2
lnpiece
mur1
reseau_tel
route_goud
route_late
sol2
toilet1
mean of housing characteristic
● non-renter
○ renter