MINDSET household module

Max Boehringer

Jun 14, 2023

CONTENTS:

CHAPTER ONE

INTRODUCTION

This repository contains the code for linking MINDSET results for sectoral price changes with country- and decile specific household expenditure data to assess incidence across expenditure deciles. Further it contains code to introduce decile-specific price and income reactions into the main model

CHAPTER

TWO

DATA PREPARATION

These functions take and merge the different microdatasets containing budget shares and elasticities and returns the xlsx used in the other functions.

Additionally a GLORIA- consumption concordance table is built.

2.1 Data preparation

`dataprep.`**`concordance_GLORIA_CPAT`**`()`

Creates concordance table between GLORIA Sectors and CPAT consumption categories using GTAP 10 codes as a bridge

Inputs:

- CPAT_GTAP.xlsx
- GTAPtoGLORIA.xlsx

Returns: CPAT_GLORIA(df) : Concordance between GLORIA and CPAT

`dataprep.`**`prepare_Microdata`**`()`

Merges expenditure data with price and income elasticity data on country and decile

Inputs:

- HH_Data_CPAT_ALL.dta
- HH_Elasticities.xlsx
- Income Elasticities_CPAT.xlsx

CHAPTER

THREE

FUNCTIONS

Results tables and plots are produced by running MASTER_household_results.py.

This script uses following functions:

3.1 Auxiliary functions

`auxiliary.calc_pc_exp_dg`(*country*, *HH_data*, *MS_q*, *concordance*, *pop_data*)

Calculates per capita expenditures per decile and consumption category g based on expenditure shares and shares of total expenditures per decile from household survey data and final HH_demand per cons. category G from MINDSET to have absolute tax burdens which are consistent with MINDSET revenue results.

Inputs:

- country (str): 3-digit iso code
- HH_data (df): Microdata
- **MS_q(df): MINDSET final household demand vector of country of interest:** Has to include columns "PROD_COMM" and "q_hh_base"
- concordance : concordance table between GLORIA and expenditure categories

Outputs:

- HH_cdf(df): HH survey dataframe with added decile shares of total consumption

for each category, total consumption per category/decile (in 2019 US$) and per capita expenditures per decile/category (in 2019 US$).

`auxiliary.calc_price_changes`(*MS_q*, *MS_p*, *concordance*)

Calculate the price changes per CPAT consumption category for a given country. The function calls calculate_sectorshares() which returns a dataframe including mapped CPAT cons. categories , GLORIA Sectors and their corresponding MINDSET price changes.

Inputs:

- country(str): the iso-3 code of the country for which to calculate sector shares.
- MS_q(df): MINDSET final demand vector (120 rows x 2 columns)

  need columns PROD_COMM and q_hh_base (if those are labelled differently change in code)
- Ms_p(df): MINDSET price vector 120 rows x 2 columns

  need columns TRAD_COMM and delta_p_base (change accordingly in code if labelled differently)
- concordance(df): concordance table between GLORIA and expenditure categories

Outputs:

delta_p_CPAT(dict): A dictionary containing the price changes as values and CPAT consumption categories as keys

auxiliary.**calc_tot_demand_g**(*country*, *HH_data*, *MS_q*, *concordance*)

Returns the total household demand per consumption category G based on MINDSETS final household demand per GLORIA sector.

Inputs:

- country (str): 3-digit iso code
- HH_data (df): Microdata
- **MS_q(df): MINDSET final demand vector (120 rows x 2 columns)** needs columns PROD_COMM and q_hh_base (if those are labelled differently change in code)
- concordance(df): concordance table between GLORIA and expenditure categories

Output:

- **total_hh_demand_g (dict): dictionary containing total hh demand (values)** in 2019 US$ for each consumption category (G)

auxiliary.**calculate_sectorshares**(*MS_q*, *concordance*)

Calculate sector shares of final demand per CPAT category for a given country using MINDSET final household demand vector and concordance table prodced by concordance_GLORIA_CPAT(). Finally it merges the dataframe with MINDSET price changes for the relevant GLORIA sectors.

Inputs:

- country(str): the iso-3 code of the country for which to calculate sector shares.
- MS_q(df): MINDSET final demand vector (120 rows x 2 columns) of coubtry of interest

  need columns PROD_COMM and q_hh_base (if those are labelled differently change in code)
- concordance(df): concordance table between GLORIA and expenditure categories

Outputs:

df_prices [pandas.DataFrame] A DataFrame with the following columns: - 'CPAT Variable': The CPAT consumption category - 'GLORIASector': The GLORIA sector - 'sector_share': Share of final HH demand of GLORIA sectors within the CPAT category

auxiliary.**get_pop**(*country*, *pop_data*)

Reads and returns official 2019 population of country from https://data.worldbank.org/indicator/SP.POP.TOTL

Inputs:

- country (str) : 3-digit iso code

Outputs:

- pop2019 (int) : population in 2019

auxiliary.**petroleum_coke_frs_shares**(*country*, *HH_data*)

Calculates expenditure shares for refined petroleum and coke oven products within the respective GLORIA sectors: Multiple consumption categories mapped to two GLORIA sectors so final GLORIA demand has to be split up accordingly

Inputs:

- country : country of interest
- HH_data : household expenditure data containing budget shares

Outputs:

- shares_cp (dict): expenditure categories (keys) , shares (values)

3.2 Main functions for calculating tax incidence

`tax_burden_scaled.`**`save_results`**(*country*, *HH_data*, *MS_q*, *MS_p*, *concordance*, *pop_data*)

Saves sector shares , price changes by consumption categories and absolute and relative incidence by decile as xlsx to be used for further analysis or plots.

`tax_burden_scaled.`**`tax_burden_MS`**(*country*, *HH_data*, *MS_q*, *MS_p*, *concordance*, *pop_data*)

Calculates and returns tax burdens per expenditure decile for plots and returns:

1. absolute and scaled to MINDSET hh demand
2. relative tax burden (scaled to MINDSET hh demand) in percent of pretax (scaled to MINDSET expenditures)

This function calls on multiple other functions, so all of the input data specified at the beginning of the script is needed

Inputs:

- country (str): 3-digit iso code
- HH_data (df): Microdata
- **MS_q(df): MINDSET final household demand vector of country of interest:** Has to include columns “PROD_COMM” and “q_hh_base”
- **Ms_p(df): MINDSET sectoral price changes in country of interest:** Has to include columns “TRAD_COMM” and “delta_p” “delta_p” depends on scenario (base, techn, trade) - column is renamed when specifying scenario in main file
- concordance : concordance table between GLORIA and expenditure categories
- pop_data : Population data

Returns:

- **HH_data_country_all (Pd.Dataframe): Dataframe containing following columns:**
  - “iso3” : 3-digit iso code
  - “quant_cons” : expenditure decile

  -“abs_inc_MS” : absolute tax burden -“rel_inc_MS”: tax burden relative to pretax expenditures -“abs_inc_ela_MS” : absolute tax burden with price-reaction -“rel_inc_ela_MS”: absolute tax burden with price-reaction relative to pretax expenditures -“cons_pc_MS” : total consumption per capita per decile pre-policy -“price_reaction” : price reaction only -> for tests

3.3 Main functions for calculating tax incidence with revenue recycling

`transfers.`**`get_other_investment`**(*country*, *countrynames*, *public_inv*)

Gets total bulk, debt or else financed public investment per infrastructure category

Inputs:

- country(str): 3 digit iso code of country
- **countrynames(df): Dataframe with countrynames and iso codes.** From last sheet of GTAPtoGLORIA.xlsx.
- **public_inv(df): Dataframe containing public investment in 1000 US$ per**

  GLORIA sector and country. From Public_inv.csv(first row has to be excluded

)

Outputs:

- public_inv_dict(dict): Dictionary with other public investment per infrastructure category in $

`transfers.`**`get_publ_inv_shares`**(*country*, *shares*)

Retrieves shares of revenue recycled into government spending used for each infrastructure category from tax template. GLORIA sectors are mapped as follows:

“wtr” : 60% of total investment into water sector [95] “sani” : [95] * 40% + [96] * 20% “ely” = [93] , “ICT” : [110,111] “transpub” = [101,102,104,106]

Inputs: -country(str): 3 digit iso code of country -shares(df): Dataframe with shares of revenue recycled into government spending

per GLORIA sector. From Templates_tax_BTA_{country}_GLORIA.xlsx, sheet “govt_spending”

Outputs: -pi_share_dict(dict): Dictionary with shares of revenue recycled [value] per consumption category i [key]

`transfers.`**`public_investment`**(*country*, *HH_data*, *MS_rev_govt*, *shares*, *countrynames*, *public_inv*, *pop_data*)

Distribution of public investment in infrastructure access spending - proxied as cash transfers towards deciles with non-existing access. Proxied transfers are received by households without pre-existing access to infrastructure type i:

Inputs:

- country(str): 3 digit iso code of country
- HH_data (df): Microdata
- MS_rev_govt (float) : total tax revenue to be recycled into government spending
- **shares(df): Dataframe with shares of revenue recycled into government spending** per GLORIA sector. From Templates_tax_BTA_{country}_GLORIA.xlsx, sheet “govt_spending”
- **countrynames(df): Dataframe mapping countrynames and ISO3 codes.** From “REGIONS” sheet of GTAPtoGLORIA.xlsx.
- public_inv(df): Dataframe with other investment into infrastructure categories
- pop_data(df): Population data

Returns:

- HH_data_country(df): Dataframe with per proxied per capita transfers when investing in public infrastructure

`transfers.`**`save_results_public`**(*country*, *HH_data*, *MS_rev_govt*, *shares*, *countrynames*, *public_inv*, *pop_data*)

Function to save results of public investment in infrastructure access spending as xlsx. Change working directory to folder where you want to save the results.

Inputs:

- country(str): 3 digit iso code of country
- HH_data (df): Microdata
- MS_rev_govt (float) : total tax revenue to be recycled into government spending
- **shares(df): Dataframe with shares of revenue recycled into government spending** per GLORIA sector. From Templates_tax_BTA_{country}_GLORIA.xlsx, sheet "govt_spending"
- **countrynames(df): Dataframe mapping countrynames and ISO3 codes.** From "REGIONS" sheet of GTAPtoGLORIA.xlsx.
- public_inv(df): Dataframe containing public investment per country /sector
- pop_data(df) : Dataframe containing population per country

Saves per proxied per capita transfers when investing in public infrastructure as xlsx to be used for further analysis or plots.

`transfers.`**`save_results_target`**(*country*, *HH_data*, *MS_q*, *MS_p*, *MS_rev_inc*, *concordance*, *pop_data*, *decile_target=10*)

Saves sector shares , price changes by consumption categories and absolute and relative incidence by decile as csv to be used for further analysis or plots.

Parameters:

- country (str): 3-digit iso code
- HH_data (df): Microdata
- **MS_q(df): MINDSET final household demand vector of country of interest:** Has to include columns "PROD_COMM" and "q_hh_base"
- **Ms_p(df): MINDSET sectoral price changes in country of interest:** Has to include columns "TRAD_COMM" and "delta_p" "delta_p" depends on scenario (base, techn, trade) - column is renamed when specifying scenario in main file
- MS_rev_inc(float) : total tax revenue to be recycled via direct transfers
- concordance (df): concordance table between GLORIA and expenditure categories
- pop_data (df): population data
- decile_target (int): OPTIONAL-targeted deciles for per capita transfers (default = 10)

`transfers.`**`targeted_transfer`**(*country*, *HH_data*, *MS_q*, *MS_p*, *MS_rev_inc*, *concordance*, *pop_data*, *decile_target*)

Calculates tax burden after revenue recycling via direct targeted per capita transfers Absolute tax burden after revenue recycling is calculated as pre-revenue recycling tax burden calculated by tax_burden_MS() minus the received per capita transfer.

Inputs:

- country (str): 3-digit iso code
- HH_data (df): Microdata

- **MS_q(df): MINDSET final household demand vector of country of interest:** Has to include columns "PROD_COMM" and "q_hh_base"
- **Ms_p(df): MINDSET sectoral price changes in country of interest:** Has to include columns "TRAD_COMM" and "delta_p" "delta_p" depends on scenario (base, techn, trade) - column is renamed when specifying scenario in main file
- MS_rev_inc(float) : total tax revenue to be recycled via direct transfers
- concordance (df): concordance table between GLORIA and expenditure categories
- pop_data (df): Population data
- decile_target (int): OPTIONAL-targeted deciles for per capita transfers (default = 10)

Output;

- **tb (df): Dataframe containing absolute and relative tax burdens for each decile** after revenue recycling in form of direct per capita transfers

3.4 Plots

The R-script for plots plots the incidence results using xlsx files. A pdf containing the graphs is also produced when running MASTER_household_results.py.

CHAPTER

FOUR

ADJUSTMENTS (MINDSET)

In its current version MINDSET models households demand reaction to tax-induced price changes of consumer goods with USDA-based own- and cross price elasticities of demand (estimated in Muhammad et al. 2011) for one representative household. Since a countries aggregate demand adjustment in response to price shocks depends both on the distribution of total expenditures and differing substitution opportunities across the income/expenditure spectrum, we can make use of pre-harmonized Microdata containing information on decile- and country specific expenditure patterns and (own)-price elasticities of demand.

4.1 Price and Income adjustments

`Price_and_Income_Elas.sector_adj_factors.`**`HHdemand_adjustments_income_GLORIA`**(*country*, *HH_data*, *MS_q*, *MS_rev_inc*, *concordance*, *decile_target=10*)

Converts adjustment factors per consumption category into GLORIA sectoral demand adjustment factor Returns dictionary with GLORIA sectors as keys and adjustment factors as values. Adjustment factors are calculated based on decile specific price-elasticities of demand and shares of overall expenditures per decile and consumption category

Inputs:

- country(str): ISO-3 code
- HH_data(df): household data containing elasticities and expenditure shares
- MS_q(df): final household demand (GLORIA sectors x 1) : Mix between imports and exports
- MS_rev_inc(float): revenue recycled into income tax cuts
- concordance(df): concordance table between GLORIA and expenditure categories
- decile_target(int): decile under and including which the tax revenue is distributed (default: 10))

Returns:

- adj_factors_g(df): pandas Dataframe with GLORIA sectors as the index column and income adjustment factors as value column (“adj_factor”)

Price_and_Income_Elas.sector_adj_factors.**HHdemand_adjustments_price_GLORIA**(*country*, *HH_data*, *MS_q*, *MS_p*, *concordance*)

Converts adjustment factors per consumption category into GLORIA sectoral demand adjustment factor Returns dictionary with GLORIA sectors as keys and adjustment factors as values. Adjustment factors are calculated based on decile specific price-elasticities of demand and shares of overall expenditures per decile and consumption category

Inputs:

- country(str): ISO-3 code
- HH_data(df): prepared Microdata
- MS_q(df): MINDSET final demand vector (120 rows x 2 columns) of coubtry of interest

  needs columns PROD_COMM and q_hh_base (if those are labelled differently change in code)
- **Ms_p(df): MINDSET price vector 120 rows x 2 columns of country of interest** needs columns TRAD_COMM and delta_p_base (change accordingly in code if labelled differently)
- concordance (df) : concordance table between GLORIA sectors and expenditure categories

Returns:

- adj_factors_price(df) : Dataframe with GLORIA sectors as the index column and price adjustment factors as the value column ("adj_factor")

Price_and_Income_Elas.sector_adj_factors.**calc_price_changes**(*MS_q*, *MS_p*, *concordance*)

Calculate the price changes per CPAT consumption category for a given country. The function calls calculate_sectorshares() which returns a dataframe including mapped CPAT cons. categories , GLORIA Sectors and their corresponding MINDSET price changes.

Inputs:

- country(str): the iso-3 code of the country for which to calculate sector shares.
- MS_q(df): MINDSET final demand vector (120 rows x 2 columns)

  need columns PROD_COMM and q_hh_base (if those are labelled differently change in code)
- Ms_p(df): MINDSET price vector 120 rows x 2 columns

  need columns TRAD_COMM and delta_p_base (change accordingly in code if labelled differently)
- concordance(df): concordance table between GLORIA and expenditure categories

Returns:

delta_p_CPAT(dict): A dictionary containing the price changes as values and CPAT consumption categories as keys

Price_and_Income_Elas.sector_adj_factors.**calc_tot_demand_g**(*country*, *HH_data*, *MS_q*, *concordance*)

Returns the total household demand per consumption category G based on MINDSETS final household demand per GLORIA sector.

Inputs:

- country (str): 3-digit iso code
- HH_data (df): Microdata containing expenditure shares
- **MS_q(df): MINDSET final demand vector (120 rows x 2 columns)** needs columns PROD_COMM and q_hh_base (if those are labelled differently change in code)
- concordance(df): concordance table between GLORIA and expenditure categories

Returns:

- **total_hh_demand_g (dict): dictionary containing total hh demand (values)** in 2019 US$ for each consumption category (G)

`Price_and_Income_Elas.sector_adj_factors.`**`calculate_sectorshares`**(*MS_q*, *concordance*)

Calculate sector shares of final demand per CPAT category for a given country using MINDSET final household demand vector and concordance table prodced by concordance_GLORIA_CPAT(). Finally it merges the dataframe with MINDSET price changes for the relevant GLORIA sectors.

Inputs:

- country(str): the iso-3 code of the country for which to calculate sector shares.
- MS_q(df): MINDSET final demand vector (120 rows x 2 columns) of coubtry of interest

  need columns PROD_COMM and q_hh_base (if those are labelled differently change in code)
- concordance(df): concordance table between GLORIA and expenditure categories

Outputs:

df_prices [pandas.DataFrame] A DataFrame with the following columns: - 'CPAT Variable': The CPAT consumption category - 'GLORIASector': The GLORIA sector - 'sector_share': Share of final HH demand of GLORIA sectors within the CPAT category - 'delta_p_base': The change in price for the given GLORIA sector

`Price_and_Income_Elas.sector_adj_factors.`**`get_weighted_income_adj_factors`**(*country*, *HH_data*, *MS_q*, *MS_rev_inc*, *concordance*, *decile_target=10*)

Calculates decile specific income adjustment factors based on the decile specific income elasticities and expenditure shares. The user has the option to target a specific decile such that the tax revenue is distributed equally among the deciles (default: 10)

Inputs:

- country(str): ISO-3 code
- HH_data(df): household data
- MS_q(df): MINDSET final demand vector (120 rows x 2 columns) of coubtry of interest

  needs columns PROD_COMM and q_hh_base (if those are labelled differently change in code)
- MS_rev_inc(float): Tax revenue to be recycled via income tax cut (in 1000 $)
- concordance(df): concordance between GLORIA and expenditures
- decile_target(int): decile to target

Returns:

- **sum_weighted_adj(dict): dictionary with deciles as keys and** adjustment factors as values

`Price_and_Income_Elas.sector_adj_factors.`**`get_weighted_price_adj_factors`**(*country*, *HH_data*, *MS_q*, *MS_p*, *concordance*)

Returns a dictionary of weighted adjustment factors per consumption category to be reused in the MINDSET Household Module. First country-specific HH and elasticity data, and MINDSET price changes per consumption category are used to calculate the adjustment factor of each expenditure category oer decile. Those are then weighted by the share of total expenditure per decile per category.

Inputs:

- country (str): The ISO 3 country code for the country of interest
- HH_data (pd.DataFrame) : Household microdata
- MS_q(df): MINDSET final demand vector (120 rows x 2 columns) of coubtry of interest

  need columns PROD_COMM and q_hh_base (if those are labelled differently change in code)
- Ms_p(df): MINDSET price vector 120 rows x 2 columns of country of interest

  need columns TRAD_COMM and delta_p_base (change accordingly in code if labelled differently)
- concordance (pd.DataFrame) : concordance table between GLORIA sectors and expenditure categories

Returns:

- sum_weighted_adj (dict) [0] : wtd. adj. factors per expenditure category
- sum_old (dict) [1] : for checks: sum of old expenditures per category - for tests

`Price_and_Income_Elas.sector_adj_factors.`**`petroleum_coke_frs_shares`**(*country*, *HH_data*)

Calculates shares for refined petroleum and coke oven products: Multiple consumption categories mapped to two GLORIA sectors so final GLORIA demand has to be split up accordingly

Inputs:

- country : country of interest
- HH_data : household expenditure data

Returns:

- shares_cp (dict): expenditure categories (keys) , shares (values)

CHAPTER

FIVE

DATA PREPARATION

These functions take and merge the different microdatasets containing budget shares and elasticities and returns the xlsx used in the other functions.

Additionally a GLORIA- consumption concordance table is built.

5.1 Tests

`test_consumption.`**`test_calc_pc_exp_dc`**(*HH_data*, *MS_q*, *concordance*, *pop_data*)

Tests whether sum of per capita expenditures per consumption category per decile adds up to total demand per consumption category

`test_consumption.`**`test_calc_pricechanges`**(*MS_q*, *MS_p*, *concordance*)

Asserts that price changes are calculated correctly:

Took appliances category and calculated price changes by hand

– If price vector changes, fixtures have to be changed too

`test_consumption.`**`test_calc_total_demand_g`**(*HH_data*, *MS_q*, *concordance*)

Tests whether total final hh demand for category g is calculated correctly and whether sum of total decile expenditures add up to sum of q_hh_base of the corresponding categories

`test_consumption.`**`test_conspc`**(*HH_data*, *MS_q*, *MS_p*, *concordance*)

Tests weighted adj. factor function: Sum of decile total expenditures should be roughly similar (sum of budget shares not always sums to 1) to the sum of expenditures

`test_consumption.`**`test_decile_shares`**(*HH_data*, *MS_q*, *concordance*, *pop_data*)

Tests whether shares of expenditures on category g for decile d add up to 1 for each category.

`test_consumption.`**`test_income_adj_factors`**(*HH_data*, *MS_q*, *MS_rev_inc*, *concordance*)

Tests whether income adjustment factors *q_hh_base at sectoral level add up to the same change as at cons category level

`test_consumption.`**`test_pc_transfer`**(*HH_data*, *MS_q*, *MS_p*, *MS_rev_inc*, *concordance*, *pop_data*)

Tests whether sum of per capita transfers sum up to the total revenue used for income tax cuts

`test_consumption.`**`test_price_adj_factors`**(*HH_data*, *MS_q*, *MS_p*, *concordance*)

Tests whether adjustment factors *q_hh_base at sectoral level add up to same change as at the cons category level

`test_consumption.`**`test_price_adj_factors_taxburden`**(*HH_data*, *MS_q*, *MS_p*, *concordance*, *pop_data*)

Test whether individual demand after price elasticites elasticities (New expenditures) adds up to adjustment factors per category * total demand per expenditure category `test_consumption.`**`test_price_change_g`**(*HH_data*, *MS_q*, *MS_p*, *concordance*)

Tests whether price changes per GLORIA sector are the same as price changes per cons category

with 1 % tolerance

`test_consumption.`**`test_public_transfer`**(*HH_data*, *MS_rev_govt*, *shares*, *countrynames*, *public_inv*, *pop_data*)

tests whether transfers add up to share * revnue recycled into government spending

THIS test only is suitable when all of the government spending goes into mapped infrastructure sectors (apart from waste)- as in running BGR example data- otherwise the test will fail as the expected transfer is the total government revenue recycling

`test_consumption.`**`test_sector_mapping`**()

Asserts 5 test GLORIA sectors are correctly mapped into their corresponding consumption category by concordance_GLORIA_CPAT.

`test_consumption.`**`test_sector_shares`**(*MS_q*, *concordance*)

Check whether sector shares are correctly calculated by function: fixtures self calculated for paper cons. category. (GLORIA sectors 59,60,61)

`test_consumption.`**`test_sector_shares_add_up_to_one`**(*MS_q*, *concordance*)

Asserts that sector shares within a consumption category add up to one.

`test_consumption.`**`test_tax_burden`**(*HH_data*, *MS_q*, *MS_p*, *concordance*, *pop_data*)

Tests whether price changes * HH demand in GLORIA final demand vector is roughly equal to the sum of all decile tax burdens

`test_consumption.`**`test_total_shares`**(*HH_data*, *MS_q*, *concordance*, *pop_data*)

Tests whether decile shares for all categories add up to 1

CHAPTER

SIX

SURVEY CHECK

The function returns an xlsx to comparing MINDSET vs HH Survey per capita consumption in each country

6.1 Check

`Survey_MINDSET_check.`**`pcc_check`**(*er*, *pop_wb*, *gdp_defl*, *MS_q*, *HH_data*)

Creates xlsx to compare total per capita consumption from MINDSET (sum of q_hh_base divided by 2010 population) with inflation and population adjusted total per capita consumption from household survey

Run this file for the output table - with files from base_data in Github repo and check that all of the data in base_data is accessible

Inputs:

- er (df): exchange rates
- pop_wb (df): population
- gdp_defl (df): gdp deflator
- MS_q (df): MINDSET final household demand vector of country of interest from results_BGR.xlsx
- HH_data (df): Microdata with elasticities

Returns:

- xlsx file with total per capita consumption from MINDSET and household survey

CHAPTER

SEVEN

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX