SCXMPIP User Manual

# Introduction

## This document is intended for the users of SCXMPIP. It gives an overview of the program and the methodology used. It also articulates how to invoke the program and describes the reporting capabilities. It is assumed that the user is familiar with data concepts and terminology used in the R programming language and has an understanding of the subject matter of international merchandise trade and price or volume indices.

# Overview

## The SCXMPIP is a collection of functions and control variables offered in the R programming language that provide the user with the functionality to aggregate international merchandise trade values from disaggregated data into a collection of formatted reports, and to create a fixed price basket of products to use in the calculation of common price indices.

## This R based package was originally created for the PRASC project. The package is currently in beta 1.0, it contains all intended functionalities but there may be some inaccuracies or bugs.

## The price index calculation follows methods and concepts from the UN “Calculation of External Trade Indices Based on Unit Values” document and the CEPII “International Trade Price Indices” document. Both documents are provided in the package.

# **Inputs and outputs**

## The following diagram illustrates the data flow in the SCXMPIP:

Valid data

Data validation

Raw Data Input

Basket creator

(Optional)

Data reports

### 

Price Basket

### 

Index Calculation

(laaspeyres or Paasche)

Price Index

### Functions

### Csv file output

### (Data flow schematic)

### Data Flow Description:

### Input into the SCXMPIP is taken from raw data saved in comma separated value (.csv) format or tab delimited txt files (.txt). The files are specified upon invocation of the raw\_data\_input function. A validation process filters out any invalid data, ensuring that only suitable data are kept for processing.

### Trade values are aggregated by type of trade flow (import or export) via the hierarchy of the Harmonized Commodity Description and Coding System (HS). Detailed trade reports can be created using the frequency specified by the user.

### Using the price basket creator function you can then select an annual basket of goods for the reference year specified by ranking the commodities by value and HS Section and selecting those commodities with highest proportions. Using said basket you’re able to calculate a price index using the Laspeyres or Paashe index calculator.

## **Control variables**

|  |  |
| --- | --- |
| Variable name | Description |
| positions | Positions of the columns of your raw data |
| base\_year | Reference base year for your index |
| imports/exports | Import and Export codes used in your data |
| multiple\_files | Set to TRUE if you want to input multiple files for your raw data |
| quarterly\_monthly | Set to TRUE for quarterly reports and FALSE for monthly reports |
| header | If your raw data already has column names then set to TRUE |

## **Data requirements**

### The minimum set of required variables includes:

* Reference month
* Reference year
* Type of trade
* HS code
* Partner country
* Value
* Weight
* Unit of measure (optional)
* Quantity

### In cases where the quantity variable is reported, it is used in place of weight in price estimation calculations.

### The raw data is validated at the 6-digit HS (HS6) level. The HS6 code is obtained by truncating the reported HS code found in the raw data to the first six digits. These HS6 codes are compared to a list of HS6 codes maintained by the UNSD. Any invalid HS6 codes found in the raw data are removed from the main data set and saved in an output file for inspection.

### If missing or negative values are reported for either the trade value or for both quantity and weight variables then they are removed from the main data set and saved in an output file for inspection.

# Usage

## Main functions

### You must first input your raw data. This will be done using the raw\_data\_input function. Before calling the function in the init.R file make sure that your variables have been changed relative to your file and to your preferences. Once your variables are set you can call the raw\_data\_input function or simply uncomment the already prepared function call in the init.R file. To call raw\_data\_input (must be in init.R):

### raw\_data\_input(positions=c(month,year,trade\_type,hs,partner\_code,value,weight,unit\_of\_measure,quantity),multiple\_files,header,base\_year,quarterly\_monthly,imports,exports)

### After calling the raw\_data\_input function there will be a total of 5 new files in your directory where you called your function. In the reports folder you will see 4 new files: “Country by Value report.csv”-> Creates a report that looks at your trade partners “Invalid.csv” -> Stores any data that had an invalid HS code or any invalid numerical value “HS2 report.csv” -> Creates a report at the HS2 aggregation level “HS4 report.csv” -> Creates a report at the HS4 aggregation level You will also find a new file in your main directory “Valid\_raw\_data.csv”, this is valid data that can be used to calculate a price index.

### You can now use the “Valid\_raw\_data.csv” to calculate a price index. To use the laspeyres or paasches functions you simply have to call one of either functions. Once called you will see one new file appear in your directory: “Paasches\_index\_result.csv” or “Laspeyres\_index\_result.csv”. To call the price index functions (must be in init.R):

### paasche\_index(base\_year)

### laspeyres\_index(base\_year)

## Additional features

### Before calling one of the price index functions you can call the basket\_creator function that will look at your data and create a file that shows you your most important commodities in each chosen HS sections. You can then change your valid\_raw\_data accordingly. To call the function (must be in init.R):

### basket\_creator(positions=c(month,year,trade\_type,hs,partner\_code,value,weight,unit\_of\_measure,quantity),header,hs\_sections=c(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22),base\_year)

### If you already have a valid-raw\_data.csv file and you simply want to add data you can call the add\_data function which will add your chosen file to the valid\_raw\_data.csv file. To call the function (must be in init.R):

### add\_data(positions=c(month,year,trade\_type,hs,partner\_code,value,weight,unit\_of\_measure,quantity),header)