Technical documentation

Version v2.4

Table of Contents

R project guidelines

All the analysis script R should be placed in the main folder. Other resources must be positioned according to the structure of the subfolder that follows.

- **R folder** contains the main function file (functions.R) and the others custom functions (one per file)
- **data folder** contains all the raw data (CSV is preferred), maps and other input files
- **output folder** contains output data (.Rdata and .csv)

functions.R

The functions.R file is useful to load all packages needed for the analysis. If a packages isn't installed on the local machine, the script will install it avoiding the errors. Moreover, it loads all the additional functions placed in the R folder. To make sure that a new function is loaded and "propagated" in all analyses files, simply put a new R file named as the name of the function in the R subfolder.

code.R

The code.R file contains all the scripts for running the automatic analysis in the server. In the fist part of the script, there is the parameter definition. Secondly, there is the generation datasets from step 0 to step 3.

In order to correct for non-sampling error in data submissions, such as measurement errors or possible fraudulent activities, we run a pre-processing routine. The pre-processing routine consisted of extracting and validating the raw data reaching the crowdsourcing platform from the mobile app in real time. This phase consists of four steps: (1) the automatic data retrieved from the digital platform through the API and conversion of the json into structured data, (2) data transformation (e.g. standardisation of measurement units), (3) data geo-location to different levels of administrative sub-division and finally, (4) outlier detection. First three steps are executed through the function dataload(), while the outlier detection is performed using the function outlier_detection().

R functions

dataload()

Data loading via API and conversion from unstructured JSON to structured dataframe (step 0, 1).

```
dataload(end.day=NA,
      end.month=NA,
      end.year=NA,
      start.day = NA,
      start.month= NA,
      start.year = NA)
```

Usage

Arguments

start.day an integer. The starting day of the observations subset.
start.month an integer. The starting month of the observations subset.
start.year an integer. The starting year of the observations subset.
end.day an integer. The ending day of the observations subset.
end.month an integer. The ending month of the observations subset.
end.year an integer. The ending year of the observations subset.

Details

The arguments are used in the request via the ona API. It is not guaranteed to work in the api v1 version due to incompatibilities of the ONA platform. To ensure error-free data download, it is preferable to download the entire dataset with this function and then subset later in the R environment.

Output

List with step 0 and step 1 dataframes

outlier_detection()

Procedures for spatial clustering using DBSCAN, outlier detection and isolated point relocation.

```
outlier_detection(data,
              h = 2,
              method =
              "sigma",
              graph=FALSE,
              eps,
              minPts = 3,
              maxd = 0.1,
              outlier.rm =
              TRUE,
              isolated.rm =
              FALSE)
```

Usage

Arguments

data a dataframe. Input dataframe must be generated from subset (rolling week of only one commodity) of step 1

h an integer. The parameter for outlier detection. (Default is 2)

method a string. Method for outlier detection. If “sigma”, the outlier detection is based on standard deviation method. If “iqr”, the outlier detection is performed using the IQR interquartile range method. (Default is sigma)

graph a logical. If TRUE, additional graph are plotted. This can slow down the function so use it only when it is actually needed. (Default is FALSE)

eps a float. In DBSCAN algorithm is the size of the epsilon neighborhood

minPts an integer. In DBSCAN algorithm is the number of minimum points in the eps region (for core points). (Default is 3)

maxd a float. Maximum distance of relocation. (Defailt is 0.1)

outlier.rm a logical. If TRUE, the outlier points are removed from the output dataframe. (Default TRUE)

isolated.rm a logical. If TRUE, the isolated points are redmoved from the output dataframe. (Default FALSE)

Details

Oulier detection algorithm is divided in three sections: 1) DBSCAN clustering algorithm 2) Price outlier for clustered points 3) Isolated points relocation

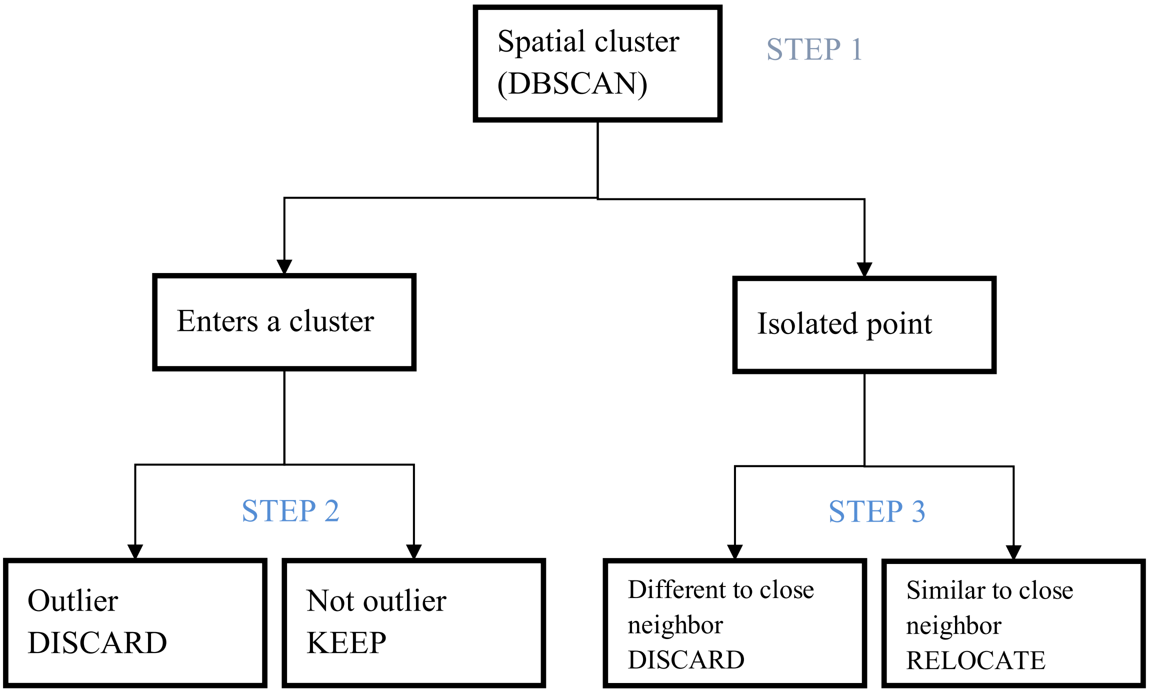

Figure 1. Diagrammatic representation of the three steps of outlier detection

Outlier detection starts by flagging up outliers based purely on spatial proximity (i.e. isolated points), without reference to the valued observed. The method used for cluster detection is the density-based spatial clustering of applications with noise (or DBSCAN) (Ester et al. 1996). Compared to other algorithms, the DBSCAN can group points that are close together (points with many points nearby), discarding isolated points in low-density regions. Therefore, it is possible to define different spatio-temporal markets. Based on local competition between selling points, prices are expected to be distributed over space without significant discontinuities within market boundaries. The competitive pressure from a selling point is relevant for other selling points or stores within a few kilometres (eps), and diminishes with increasing distance (Zhu and Singh 2009). Moreover, a minimum number of observations (minPts) is required in each cluster, allowing multiple contributions to be compared leveraging the idea of the 'wisdom of the crowd' (Surowiecki 2005).

In the second step, we considered the points that enter a cluster from the DBSCAN algorithm, and use two methods to detect price outliers. The first method consists of the classical removal of exceptional values, without considering the spatial distribution of the observation. This involves the detection and removal of all values that exceed k times the standard deviation from the mean. Alternatively, a more robust method responds to the classical right skew distribution of prices relies on the median of the price instead of mean average, and the interquartile range instead of their standard deviation. The second method relies on the idea that it is possible to detect outliers more precisely by introducing a spatial component and comparing only nearby points of sale within the same market. When this is done, unusual data (possibly generated by non-sampling errors) can be detected by looking at price values in the vicinity of the commodity in question. The idea is to define as neighbours all the points that are closer together than an arbitrary threshold* (eps). A spatial outlier is an observation that is statistically different from the values observed in the neighbourhood and is intuitively defined as the value that exceeds *r* times the variance from the average price (1) represented by the spatial lag (2)**.** All the observations marked as outliers are removed from the dataset.

$P_j > lag(P_j) + rsd(P_j)$ or $P_j < lag(P_j) - rsd(P_j)$ (1)

where $lag(P_i) = \sum_{i=1}^{n} w_{ij}\, P_j$and$w_{ij} = \begin{cases} 1\ if\ i\ and\ j\ are\ neighbours \\ \quad 0\ otherwise \end{cases}$ (2)

In the third step, we consider the points that are not part of any cluster produced by the DBSCAN algorithm, classified as isolated points. If the value observed in that point is similar the mean of a cluster, the point is associated with that cluster even if the points are distant in space (maxd). The underlying idea is to minimise the loss of information by connecting the isolated point to a cluster instead of deleting it. If, conversely, the isolated point is very different from the mean of any other cluster, then the point is discarded.

Output

Dataframe. The variable outlier is set to TRUE if the observation is marked as outlier, FALSE if the value is not an outlier, NA if an error occours during the execution of the function (usually is related to the choice of the parameters).

spatial_postsampling()

Procedures for spatial post sampling.

```
spatial_postsampling(data)
```

Usage

Arguments

data a dataframe. Input dataframe must be generated from step2

Details

In a nutshell, the s*patial* post-sampling method can be described as follows. Suppose that a set of *N* observations is collected by crowdsourcing on a set of *L* given geographical sub-areas into which the entire study area is partitioned (e.g. counties or regions). To implement the strategy, we then compare the location of the observed data with that of a set of points selected using a reference formal sample design of equivalent sample size. While in principle any design can be used, it is reasonable to assume a stratified random sample with geographical stratification and probability proportional to size (pps) or one of the optimal spatial sample designs described in the literature as a reference sample design (see Arbia 1993; Grafström, Lundström, and Schelin 2012).

In each of the L sub-areas considered, the N observations are then reweighted to resemble the formal sampling scheme by following these operational steps:

In Step 1, we count the number of observations available at a given geographical level. We will assume that in the *l-th* location (*l=1,…, L*), we have a total number of n_l crowdsourced observations, with $n_l = \sum_m n_{m,l}$, m being the internal index of location l. The total number of observations in the whole crowdsourced exercise is N = $\sum_{l=1}^{L} n_l$. We also define$X_{m.l}$as the *m-th* observation of the variable of interest X in the sub-area l.

In Step 2, the observations in each of the L locations are averaged with a simple unweighted mean $\bar{X}_l \;\; = \frac{\sum_m x_{m,l}}{n_l}$.

In Step 3, we count the number of data points needed to satisfy a formal sampling procedure in each of the L locations. Using, for instance, a random stratified sample with geographical stratification and probability proportional to the population size, we can identify a sample of data points exactly equal to those observed. We define m_l as the number of observations which should be required by the formal design in each location, with $N = \sum_{l=1}^{L} m_l$.

In Step 4 we build up a *spatial post-sampling ratio,* defined as the ratio between the number of observations required by the reference sampling plan and the number of observations available with crowdsourcing in each location, that is: $PS_l = \frac{m_l}{n_l}$.

Finally, in Step 5 the mean of the target variable X is calculated as a weighted average of X using the post-sampling ratio as weights. So, formally, we have:

$$\bar{X}^{ps} = \frac{\sum_{l=1}^{L} PS_l * X_l}{\sum_{l=1}^{L} PS_l} \qquad (3)$$

Thus, if in each location $l, PS_l = 1$, then the number of observations available in location 1 is precisely that required by the reference sampling plan, and no adjustment is needed. Conversely, if in location $l, PS_l \neq 1$, then the number of observations available in location l is different from that required by the reference sampling plan, and the observations need to be reweighted. If no observations are available in location l $(n_l = 0)$, then the location is not considered in the averaging process; if no observations are required in location l $(m_l = 0)$, then the observations collected in location l will also not contribute to the calculation of the global mean.

In its essence, our method falls within the class of post-stratification methods which share the idea of re-weighting observations to correct for under- or over-representation and differ only in the way the weights are derived. In our case, the reference to the geographical space of collection is essential to the method.

Our framework can also be used to measure the reliability of a crowdsourcing exercise, by comparing the available dataset with that required by a reference sample design.

A possible reliability measure is the following Crowdsourcing Reliability Index:

$$CRI = 1 - \frac{\sum_{l=1}^{L}(m_l - n_l)^2}{\sum_{l=1}^{L} n_l^2 - 2N \min_l(n_l) + N^2} \qquad (4)$$

with all symbols already introduced. Expression (4) is a measure of reliability that ranges between 0 and 1. In fact, in the case of low reliability, we are in the worst-case scenario when all crowdsourced data is concentrated in one single spatial sub-area where, following a formal sample design, we needed the minimum number of points. In this case $n_l = N, if l = \min_l(m_l)$ and $n_l = 0,$ otherwise and *CRI* = 0. Conversely, in the case of maximum reliability (when the crowdsourced data and the formal design perfectly coincide and we do not need any post-sampling correction), we have$(m_l = n_l), \forall l$ so that $\sum_{l=1}^{L}(m_l - n_l)^2 = 0$ and *CRI* = 1.

Output

Dataframe

global.postsampling()

Procedures for global post sampling.

```
global.postsampling(data)
```

Usage

Arguments

data a dataframe. Input dataframe must be generated from step2

Details

Post sampling is based on LGA geographical level population. The function computes the global CRI index. The output is provided at LGA level.

In a nutshell, the *global* post-sampling method can be described as follows. Suppose that a set of *N* observations is collected by crowdsourcing on a set of *L* given geographical sub-areas (LGA) into which the entire study area is partitioned (e.g. counties or regions). To implement the strategy, we then compare the location of the observed data with that of a set of points selected using a sampling design based on probability proportional to size (pps) balanced on the population of the sub-areas.

In each of the L sub-areas considered, the N observations are then reweighted to resemble the proportion with the population by following these operational steps:

In Step 1, we count the number of observations available at a given geographical level. We will assume that in the *l-th* location (*l=1,…, L*), we have a total number of n_l crowdsourced observations, with $n_l = \sum_m n_{m,l}$, m being the internal index of location l. The total number of observations in the whole crowdsourced exercise is N = $\sum_{l=1}^{L} n_l$. We also define$X_{m.l}$as the *m-th* observation of the variable of interest X in the sub-area l.

In Step 2, the observations in each of the L locations are averaged with a simple unweighted mean $\bar{X}_l \quad = \frac{\sum_m X_{m,l}}{n_l}$.

In Step 3, we count the number of data points needed to satisfy the random stratified sample with geographical stratification and probability proportional to the population size. We define m_l as the number of observations which should be required by the formal design in each location, with $N = \sum_{l=1}^{L} m_l$.

In Step 4 we build up a *global post-sampling ratio,* defined as the ratio between the number of observations required by the reference sampling plan and the number of observations available with crowdsourcing in each location, that is: $PS_l = \frac{m_l}{n_l}$.

Finally, in Step 5 the mean of the target variable X is calculated as a weighted average of X using the post-sampling ratio as weights. So, formally, we have:

$$\bar{X}^{ps} = \frac{\sum_{l=1}^{L} PS_l * X_l}{\sum_{l=1}^{L} PS_l} \qquad (5)$$

Thus, if in each location $l, PS_l = 1$, then the number of observations available in location 1 is precisely that required by the reference sampling plan, and no adjustment is needed. Conversely, if in location $l, PS_l \neq 1$, then the number of observations available in location l is different from that required by the reference sampling plan, and the observations need to be reweighted. If no observations are available in location l $(n_l = 0)$, then the location is not considered in the averaging process; if no observations are required in location l $(m_l = 0)$, then the observations collected in location l will also not contribute to the calculation of the global mean.

In its essence, our method falls within the class of post-stratification methods which share the idea of re-weighting observations to correct for under- or over-representation and differ only in the way the weights are derived. In our case, the reference to the population of the area is essential to the method.

Our framework can also be used to measure the reliability of a crowdsourcing exercise, by comparing the available dataset with that required by a reference sample design.

A possible reliability measure is the following Crowdsourcing Reliability Index:

$$CRI = 1 - \frac{\sum_{l=1}^{L}(m_l - n_l)^2}{\sum_{l=1}^{L} n_l^2 - 2N \min_l(n_l) + N^2} \qquad (6)$$

with all symbols already introduced. Expression (4) is a measure of reliability that ranges between 0 and 1. In fact, in the case of low reliability, we are in the worst case scenario when all crowdsourced data is concentrated in the sub-area with the smallest number of population. In this case $n_l = N, if l = \min_l(m_l)$ and $n_l = 0,$ otherwise and *CRI* = 0. Conversely, in the case of maximum reliability (when the crowdsourced data and the population distribution coincide and we do not need any post-sampling correction), we have$(m_l = n_l), \forall l$ so that $\sum_{l=1}^{L}(m_l - n_l)^2 = 0$ and *CRI* = 1.

Output

Dataframe