Introduction to flagr

Background

A flag is an attribute of a cell in a data set that provides additional qualitative information about the statistical value of that cell. They can indicate, for example, that a given value is estimated, confidential or represents a break in the time series.

Currently different sets of flags are in use in the European Statistical System (ESS). Some statistical domains use the SDMX code list for observation status and confidentiality status. Eurostat uses a simplified list of flags for dissemination, and other domains apply different sets of flags defined in regulations or in other agreements.

While in most cases it is well defined how a flag shall be assigned to an individual value, it is not straightforward to decide what flag shall be propagated to aggregated values like a sum, an average, quintiles, etc. This topic is important for Eurostat as the European aggregates are derived from national data points. Thus the information contained in the individual flags need to be summarized in a flag for the aggregate. This issue is not unique to Eurostat, but can occur for any aggregated data. For example, a national statistical institute may derive the national aggregate from regional data sets. In addition, the dissemination process provides further peculiarity: only a limited set of flags, compared to the set of flags used in the production process, can be applied in order to make it easily understandable to the users.

Eurostat tested various approaches with a view to well balance transparency and clarity of the information made available to users in a flag. The resulting 3 methods are implemented in an R package for assigning a flag to an aggregate based on the underlying flags and values. Since the topic has relevance outside of Eurostat as well, it was decided to publish the respective package (*flagr*) with a view to foster re-use within the European Statistical System and to stimulate discussion, including width the user community.

The *flagr* package

The objective of the *flagr* package is to promote a standardised and rational approach within and across statistical domains to assign flags to an aggregate based on the flags and values of the underlying components.

The package was developed with minimal dependencies on other packages, keeping in mind that in many work places there can be limitations to install additional packages to the standard R installation.

The package contains a fictive test data set, a wrapping function (`propagate_flag`) calling the different methods and 3 methods (`flag_hierarchy`, `flag_frequency` and `flag_weighted`) to derive flags for aggregates.

Test data

The test data is based on a Eurostat data set adding random values and flags to the original data table of *spr_exp_sum*[1]. The data has a melted structure and follows the data configuration how the data sets from the Eurostat database can be retrieved. It contains a filtered set of the last four columns of the data table retrieved by the `get_eurostat("spr_exp_sum",keepFlags = T)` command of the *eurostat* package.

```
head(test_data)
```

```
##   geo flags       time   values
## 1  AT  <NA> 2009-01-01  9873.04
## 2  AT  <NA> 2010-01-01 11229.27
```

[1] The *spr_exp_sum* data table contains data about social protection expenditures in Euro per inhabitant.

```
## 3  AT  <NA> 2011-01-01 10791.54
## 4  AT    de 2012-01-01 10887.16
## 5  AT  <NA> 2013-01-01 11418.25
## 6  AT  <NA> 2014-01-01 11278.88
```

```
summary(test_data)
```

```
##      geo                flags               time
##  Length:195         Length:195         Length:195
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
##
##
##
##      values
##  Min.   :  463.1
##  1st Qu.: 3110.9
##  Median : 5510.4
##  Mean   : 7071.4
##  3rd Qu.:10624.3
##  Max.   :20875.1
```

The methods

1. `flag_hierarchy` method

The first method is based on the hierarchy described in the SDMX guidelines for implementing the observation status code list. The method contains two predefined hierarchy list of flags as can be seen in the table below, but it can be used with any own list of flags.

Table 1: Predefined hierarchy of flags implemented in the `flag_hierarchy` method

SDMX flags	corresponding Eurostat flags
B / time series break (highest importance)	b
O / missing value	:
M / missing value; data cannot exist	z
L / missing value; data exist but were not collected	
H / missing value; holiday or weekend	
Q / missing value; suppressed	c
J / derogation	
S / strike and other special events	
D / definition differs	d
I / imputed value	
F / forecast value	f
E / estimated value	e; s
P / provisional value	p
N / not significant	n
U / low reliability	u
V / unvalidated value	
G / experimental value	
A / normal value	

In this simplest method, the aggregate inherits all the flags of the underlying data. In case there are several flags for individual values, it takes into account all the flags separately. For example, in the case of the 4^{th} observation the flags *"de"* are taken into account as *"d"* and as *"e"* too. The function first creates a unique list of all the flags for a given time period and then the hierarchy, like the one in Table 1, is applied in order to end up with a single flag for the aggregate.

In case there is a flag which is not mentioned in the predefined hierarchy list, then the function returns an error mentioning the missing flag(s). The hierarchical order can be defined as a string or a character vector. The first character of the string/vector has the highest importance and the last character has the lowest significance. The below examples show these different options and how the results change for a different order.

```
library(tidyr)
flags <- spread(test_data[, c(1:3)], key = time, value = flags)
propagate_flag(flags[, c(2:ncol(flags))],"hierarchy","puebscd")
```

```
## $`2009-01-01`
## [1] "p"
##
## $`2010-01-01`
## [1] "p"
##
## $`2011-01-01`
## [1] "p"
##
## $`2012-01-01`
## [1] "p"
##
## $`2013-01-01`
## [1] "p"
##
## $`2014-01-01`
## [1] "p"
##
## $`2015-01-01`
## [1] "p"
```

```
propagate_flag(flags[, c(2:ncol(flags))],"hierarchy",c("b","c","d","e","p","s","u"))
```

```
## $`2009-01-01`
## [1] "e"
##
## $`2010-01-01`
## [1] "b"
##
## $`2011-01-01`
## [1] "c"
##
## $`2012-01-01`
## [1] "b"
##
## $`2013-01-01`
## [1] "b"
##
## $`2014-01-01`
## [1] "b"
```

```
##
## $`2015-01-01`
## [1] "b"
```

The advantage of this option is that it is very simple to implement. The disadvantage is that each flag is treated as it had the same importance, independently form the frequency of a given flag in the underlying data set. This negative side effect is shown for the last period in the second example where 1 *"b"* flag overrules 10 *"p"* flags.

2. `flag_frequency` method

In this method the flag of the aggregate is the most common flag in the underlying data. In case there are multiple flags for a single value, all the appearances are counted. For example the *"de"* flag of the 4^{th} observation increases the count of the *"d"* and *"e"* flag by one. Finally, if the most frequent flag is not unique (e.g. there are several flags with the same frequency count) then all the flags are listed as a result, as it is the case in the below example.

```
library(tidyr)
flags <- spread(test_data[, c(1:3)], key = time, value = flags)
propagate_flag(flags[, c(2:ncol(flags))],"frequency")
```

```
## $`2009-01-01`
## [1] "p"
##
## $`2010-01-01`
## [1] "p"
##
## $`2011-01-01`
## [1] "e"
##
## $`2012-01-01`
## [1] "e" "u"
##
## $`2013-01-01`
## [1] "u"
##
## $`2014-01-01`
## [1] "p"
##
## $`2015-01-01`
## [1] "p"
```

This technique corrects for the disadvantage of the first method. The downside of this option is that in many cases the frequency count is not in line with the contribution of the underlying values to the total (e.g. flags assigned to values for small countries have a bigger influence on the flag of the aggregate than the countries' contribution to the total).

3. `flag_weighted` method

To address the deficiency of the previous method, the weighted frequency method replaces the simple frequency count with a weighted frequency of flags, where the weight is the contribution of the individual values to the aggregate. After calculating the total values of the weights for each different flags, the aggregate would receive a single flag which has the highest total value if at least a given proportion of the observations have flags. If the aggregate is the sum of the individual values then in mathematical terms this option can be written as follows:

$$F_{agg} = \underset{\hat{F} \in F_i}{\arg\max} \sum_{F_i = \hat{F}} X_i \quad , \qquad \text{if} \qquad \frac{\sum_{X_i \in D_{\hat{F}}} X_i}{\sum_{X_i \in D} X_i} \geq T \tag{1}$$

Where F_{agg} is the derived flag for the aggregate, X_i the underlying values in the data set of D, F_i are the corresponding flag(s) for each individual data point; $D_{\hat{F}}$ is a the subset of D with flag value $\hat{F}$, and T is the threshold value. As an initial threshold a value of 0.5 is suggested, meaning that the sum of the weights contributed to the aggregate for the given flag is more than half of the total weights.

The same procedure can be used if the aggregate is a weighted average, index, or quantile. In this case in equation 1 the individual values (X_i) are replaced by the individual weights used for the calculation of the aggregates.

```
library(tidyr)
flags <- spread(test_data[, c(1:3)], key = time, value = flags)
weights <- spread(test_data[, c(1, 3:4)], key = time, value = values)

flags<-flags[, c(2:ncol(flags))]
weights<-weights[, c(2:ncol(weights))]
propagate_flag(flags,"weighted",flag_weights=weights)
```

```
## $`2009-01-01`
## $`2009-01-01`[[1]]
## [1] NA
##
## $`2009-01-01`[[2]]
## [1] NA
##
##
## $`2010-01-01`
## $`2010-01-01`[[1]]
## [1] NA
##
## $`2010-01-01`[[2]]
## [1] NA
##
##
## $`2011-01-01`
## $`2011-01-01`[[1]]
## [1] NA
##
## $`2011-01-01`[[2]]
## [1] NA
##
##
## $`2012-01-01`
## $`2012-01-01`[[1]]
## [1] NA
##
## $`2012-01-01`[[2]]
## [1] NA
##
##
## $`2013-01-01`
```

```
## $`2013-01-01`[[1]]
## [1] NA
##
## $`2013-01-01`[[2]]
## [1] NA
##
##
## $`2014-01-01`
## $`2014-01-01`[[1]]
## [1] NA
##
## $`2014-01-01`[[2]]
## [1] NA
##
##
## $`2015-01-01`
## $`2015-01-01`[[1]]
## [1] NA
##
## $`2015-01-01`[[2]]
## [1] NA
```

```
propagate_flag(flags,"weighted",flag_weights=weights,threshold=0.1)
```

```
## $`2009-01-01`
## $`2009-01-01`[[1]]
## [1] NA
##
## $`2009-01-01`[[2]]
## [1] NA
##
##
## $`2010-01-01`
## $`2010-01-01`[[1]]
## [1] NA
##
## $`2010-01-01`[[2]]
## [1] NA
##
##
## $`2011-01-01`
## $`2011-01-01`[[1]]
## [1] "e"
##
## $`2011-01-01`[[2]]
## [1] "0.133211654011488"
##
##
## $`2012-01-01`
## $`2012-01-01`[[1]]
## [1] "u"
##
## $`2012-01-01`[[2]]
## [1] "0.124148695828608"
##
```

```
##
## $`2013-01-01`
## $`2013-01-01`[[1]]
## [1] NA
##
## $`2013-01-01`[[2]]
## [1] NA
##
##
## $`2014-01-01`
## $`2014-01-01`[[1]]
## [1] "p"
##
## $`2014-01-01`[[2]]
## [1] "0.174097815337322"
##
##
## $`2015-01-01`
## $`2015-01-01`[[1]]
## [1] "p"
##
## $`2015-01-01`[[2]]
## [1] "0.335586943761588"
```

As the above examples show this method results in missing value (`NA`) if the sum of the weights for all the flag is below the limit. When the total is above the threshold, then the maximum of the sum of weights is returned in the results next to the derived flag.

Summary

This paper shows how the `propagate_flag` function of the *flagr* package can call different methods to derive flags for aggregates, provides examples and discusses its use cases. For the *hierarchy* method the ordered list of flags is required as a parameter. For the *frequency* method no other information is needed outside the individual flags. Finally for the *weighted* frequencies method the weights are required as input next to the compulsory flags.

Possible extensions of the package in the future:

- return not just the first flag, but provide a table with further flags in the hierarchy
- create a shiny app, which can provide the flags for an uploaded csv file