Package StQT

Luis Sanguiao Sande

13 de abril de 2016

Introduction

The StQT package is an attempt to standardize the representation of data transformations, in the same sense that StQ standardizes the representation of data. In fact, it stands for StQ Transformation. But, what do we understand by data transformation? In this context, it means a process that calculates a new data set from an old data set. For example, an imputation, a summary or even a forecast are transformations. But a data merge **is not** a transformation since we are using a second data set (except for a self data merge which will be explained later). If we would happen to need a transformation that uses information from two data sets, we would have to join them into a single data set before.

The

The representation

It is strongly inspired in the syntax of operator := from data.table package. The main idea is to apply a set of rules according to the sentence `data[<domain>,<output>:=<fun>(<input>),by=<by>]` where each <> surrounded item is an element of the rule. As long as function definition is included in the object, it is obvious that we can represent this way almost everything. Almost, because the number of rows of data.table data, would not be changed. The consequence was to allow a direct assignation in `<output>` field which when present, will create new rows. For a more detailed view of the syntax of the rules read next section.

Note that if we remove `<domain>` and `<by>` and put `<fun>` and `<input>` as the only elements of the rule we still can get any transformation (even with one rule). But this rules representation would be useless in that case, since you have a function (an R function) that does all the work. So, generally speaking: keep the functions as simple as possible and try to use the other fields of the rule to achieve that. And be conscious that a lower number of rules **is not** necessarily better.

Detailed syntax

There are three types of rules in StQT:

- Horizontal: These are by far the more common rules and they are applied as we had said with a command like `data[<domain>,<output>:=<fun>(<input>),by=<by>]`. This creates new or override `<output>` columns with one element per row applying the usual recycling.
- Vertical: Similar to horizontal rules, but output includes some assignments (one or more). For each `<by>` group an additional row is created which copies the variables in `<by>`, sets the assigned variables and calculates the unassigned in a similar way to an horizontal rule.
- Internal: These are not real functions but *special* codes in `<fun>` field that trigger concrete behaviors different to usual horizontal or vertical rules. They are *FunDelVar*, *FunDelRow* and *FunAutoLink* so far. Except for *FunDelRow*, we could write functions to get the same behavior (at least when applied to data.table objects), but they are anyways useful because implement common tasks in a standard way.

There is also an `<order>` field which if not empty, orders the data by the variables specified.

Note that `<key>` field is ignored unless we are transforming an StQ object (see next section).

How does TApply work on StQ objects

Other methods