How to Use TimeSeries.OBeu Package

A Short Guide in R and OpenCPU environments

Kleanthis Koupidis

November 7, 2016

This document describes the use of the functions implemented in TimeSeries.OBeu package both in R and OpenCPU environments.

Install:

Load *devtools* library or install it if not already:

Then install *TimeSeries.OBeu* from Github

And load the library

```r
library(TimeSeries.OBeu)
```

Use:

The basic function is:

```r
ts.analysis(tsdata,x.order=NULL,h=1)
```

where *tsdata*: The input univariate time series data *x.order*: An integer vector of length 3 specifying the order of the Arima model and *h*: The number of prediction steps

R Example

The package includes the following time series data: Athens_draft_ts, Athens_revised_ts, Athens_reserved_ts, Athens_approved_ts and Athens_executed_ts.

```r
Athens_draft_ts
```

```
## Time Series:
## Start = 2004
## End = 2015
## Frequency = 1
##  [1] 720895000 628937000 618550000 724830000 858942000 919508000 977488000
##  [8] 931607000 866517393 667108000 773422555 759559284
```

We select for example the approved budget phase of Athens and we want to predict 4 years ahead.

```r
ts.analysis(tsdata = Athens_approved_ts, h=4)
```

```
## {"acf.param":{"acf.parameters":{"acf":[1,0.427,0.2297,0.0089,-0.3902,-0.4655,-0.4154,-0.2643,0.0666,-
```

If we can set a specific order to fit the model for the same prediction steps. We select for example a three-length vector of p=2 (*AR order*) d=1 (*first differences*) and q=1 (*MA order*).

```
ts.analysis(tsdata = Athens_approved_ts, x.order=c(2,1,1), h=4)
```

```
## {"acf.param":{"acf.parameters":{"acf":[1,0.427,0.2297,0.0089,-0.3902,-0.4655,-0.4154,-0.2643,0.0666,
```

OpenCPU Short Guide - TimeSeries.OBeu

Go to: http://okfnrg.math.auth.gr/ocpu/test/

How to use functions:

Type to the endpoint:

```
 ../library/ {name of the library} /R/ {function}
```

If you want to see the function parameters you should:

- Select Method:

```
Get
```

and in order to run a function you should:

- Select Method:

```
Post
```

Example #1:

1. Go to http://okfnrg.math.auth.gr/ocpu/test/
2. Copy and paste the following function to the endpoint

```
../library/TimeSeries.OBeu/R/ts.analysis
```

3. *Select Method*:

```
 Post
```

4. **Add parameters** and set:

Define the input time series data:

- *Param Name*:

```
tsdata
```

- *Param Value* one of the following:

```
Athens_draft_ts

Athens_revised_ts

Athens_reserved_ts

Athens_approved_ts

Athens_executed_ts
```

Define the order of the model fits and forecasts (*optional*):

- *Param Name*:

```
x.order
```

- *Param Value* -for example:

```
c(2,1,1)
```

Define the prediction steps (*default is 1 prediction step*):

- *Param Name*:

```
h
```

- *Param Value* -for example:

```
4 # (or another number, default h=1)
```

5. Ready! Click on **Ajax request**!
6. To see the results:

copy the */ocpu/tmp/{this}/R/.val* (the first choice on the right panel)

7. and paste http://okfnrg.math.auth.gr/ocpu/tmp/ {this} /R/.val on a new tab.

Example #2 - Rudolf/Open Spending Time Series

1. Go to http://okfnrg.math.auth.gr/ocpu/test/
2. Copy and paste the following function to the endpoint

```
../library/TimeSeries.OBeu/R/open_spending.ts
```

3. *Select Method*:

```
Post
```

4. **Add parameters** and set:

Define the input time series data:

- *Param Name*:

```
json_data
```

- *Param Value* -the following output from open spending api or you can provide the json URL:

```
'{"page":0,
"page_size":30,
"total_cell_count":15,
"cell":[],
"status":"ok",
"cells":
[{"global__fiscalPeriod__28951.notation":"2002",
"global__amount__0397f.sum":290501420.64,"global__amount__0397f__CZK.sum":9210928544.2325,"_count":4805
{"global__fiscalPeriod__28951.notation":"2003",
"global__amount__0397f.sum":311242291.07,"global__amount__0397f__CZK.sum":9832143974.9013,"_count":4988
{"global__fiscalPeriod__28951.notation":"2004",
"global__amount__0397f.sum":5268500701.1,"global__amount__0397f__CZK.sum":170688885714.24,"_count":1005
{"global__fiscalPeriod__28951.notation":"2005",
"global__amount__0397f.sum":2542887761.01,"global__amount__0397f__CZK.sum":77204615312.025,"_count":203
{"global__fiscalPeriod__28951.notation":"2006",
"global__amount__0397f.sum":14803951786.68,"global__amount__0397f__CZK.sum":429758720367.32,"_count":13
{"global__fiscalPeriod__28951.notation":"2007",
"global__amount__0397f.sum":16188514346.44,"global__amount__0397f__CZK.sum":445588857385.76,"_count":22
{"global__fiscalPeriod__28951.notation":"2008",
"global__amount__0397f.sum":18231035815.89,"global__amount__0397f__CZK.sum":480643028250.12,"_count":24
{"global__fiscalPeriod__28951.notation":"2009",
"global__amount__0397f.sum":19079541164.68,"global__amount__0397f__CZK.sum":511808691742.54,"_count":26
{"global__fiscalPeriod__28951.notation":"2010",
"global__amount__0397f.sum":22738650575.01,"global__amount__0397f__CZK.sum":597685430364.14,"_count":87
{"global__fiscalPeriod__28951.notation":"2011",
"global__amount__0397f.sum":24961375670.57,"global__amount__0397f__CZK.sum":626230992823.26,"_count":13
{"global__fiscalPeriod__28951.notation":"2012",
"global__amount__0397f.sum":261513607691.41,"global__amount__0397f__CZK.sum":7030666436872.5,"_count":1
{"global__fiscalPeriod__28951.notation":"2013",
"global__amount__0397f.sum":268946402299.09,"global__amount__0397f__CZK.sum":7226220232913.8,"_count":1
{"global__fiscalPeriod__28951.notation":"2014",
"global__amount__0397f.sum":255222816704.9,"global__amount__0397f__CZK.sum":6907598086283.4,"_count":17
{"global__fiscalPeriod__28951.notation":"2015",
"global__amount__0397f.sum":22976062973.62,"global__amount__0397f__CZK.sum":636276111928.46,"_count":21
{"global__fiscalPeriod__28951.notation":"2016",
"global__amount__0397f.sum":12051686541.16,"global__amount__0397f__CZK.sum":325672725401.77,"_count":16
"order":[["global__fiscalPeriod__28951.fiscalPeriod","asc"]],
"aggregates":["","_count"],
"summary":{"global__amount__0397f.sum":945126777743.27,"global__amount__0397f__CZK.sum":25485085887878}
"attributes":[""]}'
```

Define the time label of the json input:

- *Param Name*:

```
time
```

- *Param Value* -for example:

```
"global__fiscalPeriod__28951.notation" # or

'global__fiscalPeriod__28951.notation'
```

Define the amount label of the json input:

- *Param Name*:

```
amount
```

- *Param Value* -for example:

```
'global__amount__0397f.sum' # or

"global__amount__0397f.sum"
```

Define the order of the model fits and forecasts (*optional*):

- *Param Name*:

```
order
```

- *Param Value* -for example:

```
c(3,1,1)
```

Define the prediction steps (*default is 1 prediction step*):

- *Param Name*:

```
prediction_steps
```

- *Param Value* -for example:

```
4 # (or another number, default h=1)
```

5. Ready! Click on **Ajax request**!
6. To see the results:

copy the */ocpu/tmp/{this}/R/.val* (the first choice on the right panel)

7. and paste http://okfnrg.math.auth.gr/ocpu/tmp/ {this} /R/.val on a new tab.

Further Details:

- https://www.opencpu.org/help.html
- https://cran.r-project.org/web/packages/opencpu/vignettes/opencpu-server.pdf
- https://www.opencpu.org/jslib.html

Github:

- https://github.com/okgreece/TimeSeries.OBeu