TSAIB tutorial

Mathias Høxbro Juel Vendt

2022-03-19

Contents

Preface

The software package TSAIB was developed as the main deliverable for a synthesis project regarding the study-line: Earth and Space Physics and Engineering, with a specialization in Earth Observation.

Installation

The software is implemented in the open source language "R" which can be found at https://www.r-project.org/. The source code to the "R" software package TSAIB is located at GitHub at https://github.com/MathiasVendt/TSAIB

From GitHub

The easiest way to install the package is directly from Github by using the R devtools package. The R-version should be 2.10 or higher.

1. Open R
2. Install the devtools library by typing

```
install.packages('devtools')
```

3. Install the TSAIB package

```
devtools::install_github("MathiasVendt/TSAIB")
```

Dependencies

The packages depends on the following R libraries which can be installed from R with the function `install.packages`. Hence, to install the package "ncdf4"use the following command for the R window:

```
install.packages("ncdf4")
```

- ncdf4
- ggplot2
- ggpubr

Getting started with R

The section gives a short introduction to R, which is useful to new R users.

R tutorials can be found at the r-project web side https://cran.r-project.org/manuals.html

Help pages can be accessed by typing "?" in front of a given function. If we want to access the help for the function `sum` we write

```
?sum
```

To start the web based help interface

```
help.start()
```

To exit R write

```
q()
```

Introduction to the package "TSAIB"

TSAIB is an R package aimed at getting statistical insight and performing time series analysis of satellite observations consisting of sea-level anomalies measured in the arctic ocean. A variety of functions is provided in order to determine a time series model based on the analyses and insights from the analysis functions. In this introduction, examples are given to illustrate how to use the package.

To load the package simply write:

```
library(TSAIB)
```

Using "TSAIB" for analysis, visualization and manipulation of data

This section gives a step by step guide on how to utilize the package for analyzing the data, and the order in which the functions are used are recommended.

Load data

The test data set is called `TSdata`, and is included when the package is downloaded. When the TSAIB library is loaded, info about the data set can be found by the command:

```
?TSdata
```

In order to load the data into the R environment, the data set is assign a name "TSdata" like this:

```
TSdata=TSdata
```

The included data "TSdata" is a list containing 5 elements: Longitudes, latitudes, dates, measurements and a 3x3x325 construct, containing 325 measurements from a 3x3 data point grid. The data is extracted with the `GridTSExtract` function from the TSAIB library, and is done as:

```
TSdata=GridTSExtract(nco)
```

Where `nco` is the NetCDF data from which TSdata is extracted, and documentation on the data is listed in the Data appendix.

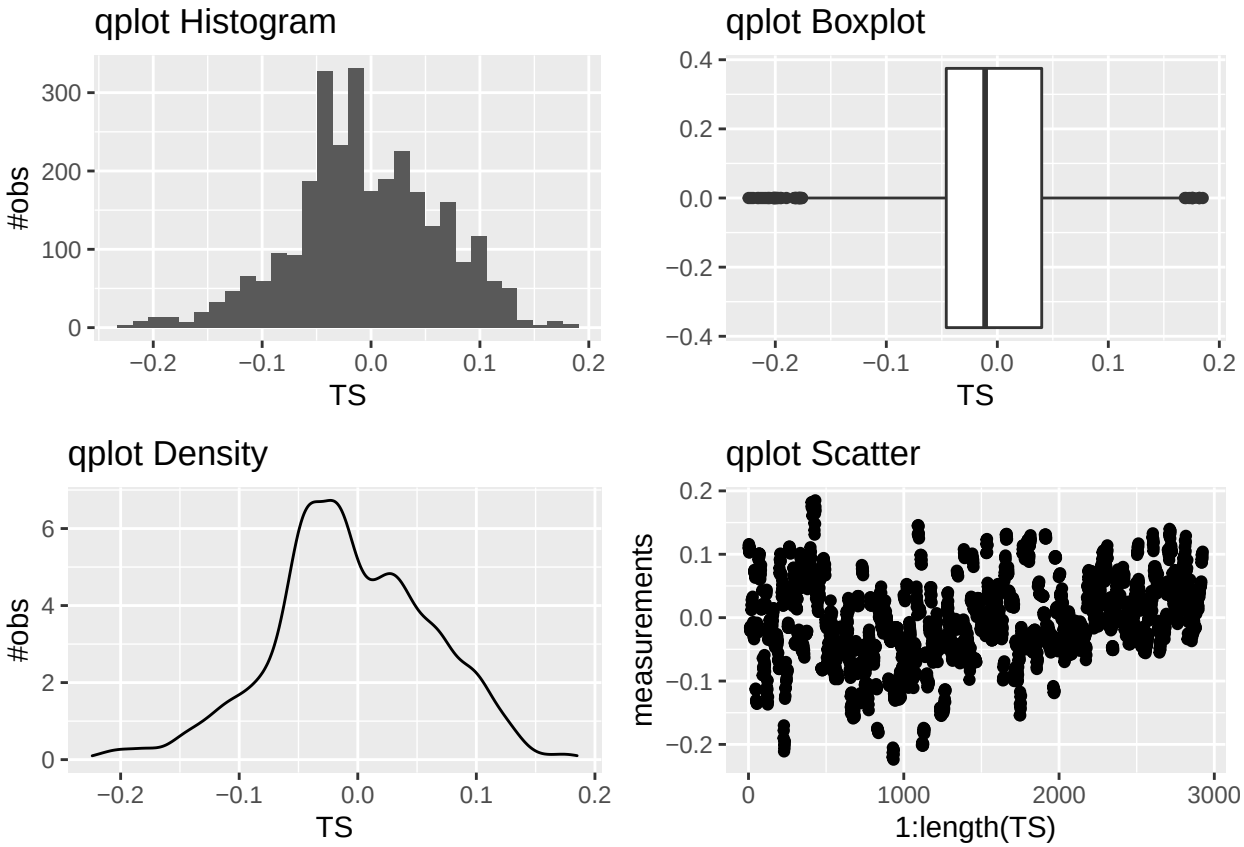
The user can also import their own data from a NetCDF file using the `GridTSExtract` function, and more info on how to use the function on a specific NetCDF file is shown by:

```
?GridTSExtract
```

Gather simple statistics about the data

It is now desired to get a brief overview of the data, including statistics, distributions and plotting. This can be achieved using the `TSdiagnostics` function on the extracted 3x3 grid of measurements. To access sub elements of a construct in R, use `$` as a seperator between the parent object, and it's sub-directory. The function wont work for NaN elements, and since the extracted data contains a few of those, the `nanrem` option in the `TSdiagnostics` function is set to `"TRUE"` as shown below, which will omit any NaN elements for the analysis.

```
TSdiagnostics(TSdata$TSmatrix,nanrem="TRUE")
#> `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
#> Warning: Removed 4 rows containing non-finite values (stat_bin).
#> Warning: Removed 4 rows containing non-finite values (stat_boxplot).
#> Warning: Removed 4 rows containing non-finite values (stat_density).
#> Warning: Removed 4 rows containing missing values (geom_point).
```




```
#> Number of NaN's in the data set
#> [1] 4
#> Number of objects in the data set
#> [1] 2925
#> Fraction of the data set which is NaN's
#> [1] 0.001367521
#> mean
#> [1] -0.005382746
#> standard deviation
```

```
#> [1] 0.06690633
#> median
#> [1] -0.011
#> quantile
#>     0%    25%    50%    75%   100%
#> -0.224 -0.046 -0.011  0.040  0.185
#> sum
#> [1] -15.723
#>
#>  One Sample t-test
#>
#> data:  TS
#> t = -4.3481, df = 2920, p-value = 1.42e-05
#> alternative hypothesis: true mean is not equal to 0
#> 95 percent confidence interval:
#>  -0.007810080 -0.002955411
#> sample estimates:
#>    mean of x
#> -0.005382746
```

Removing NaN

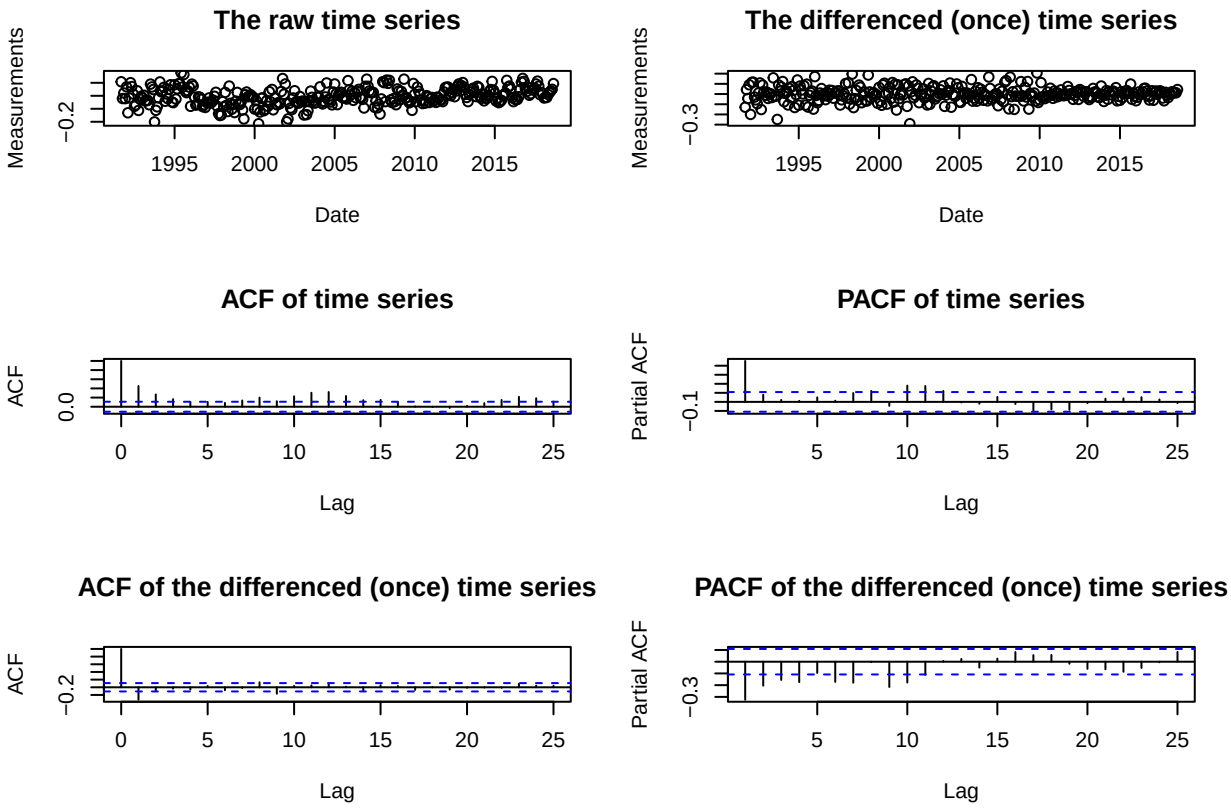
If it is decided to remove NaN objects, the function `TSnanrem` can be used to do this, with a variety of options of how to deal with NaN. For this example, the NaN's are to replaced with the mean of the time series:

```
TSnanrem(TSdata$TSmatrix, method = "mean")
```

Time series analysis plots

It is possible to get an insight of the time series with the `TSAplots` function. This function utilizes R's inbuilt differencing function `diff`, auto correlation function `acf` and partial auto correlation function `pacf`, to display a collection of plots to determine the time series characteristics. It is possible to choose a transformation of the data, being either `sqrt` or `log` transformations, and even to save the plots in a variety of formats. For this example, the [2,2] grid point of `TSdata$TSmatrix` is analyzed, and the default settings is used, which doesn't transform the data or save the plots:

```
TSAplots(TSdata$TSmatrix[2,2,],TSdata$date)
```



Using "TSAIB" for model estimation

To estimate a model which is describing the time series at hand, the `TSAIB` package containes several functions with different approches:

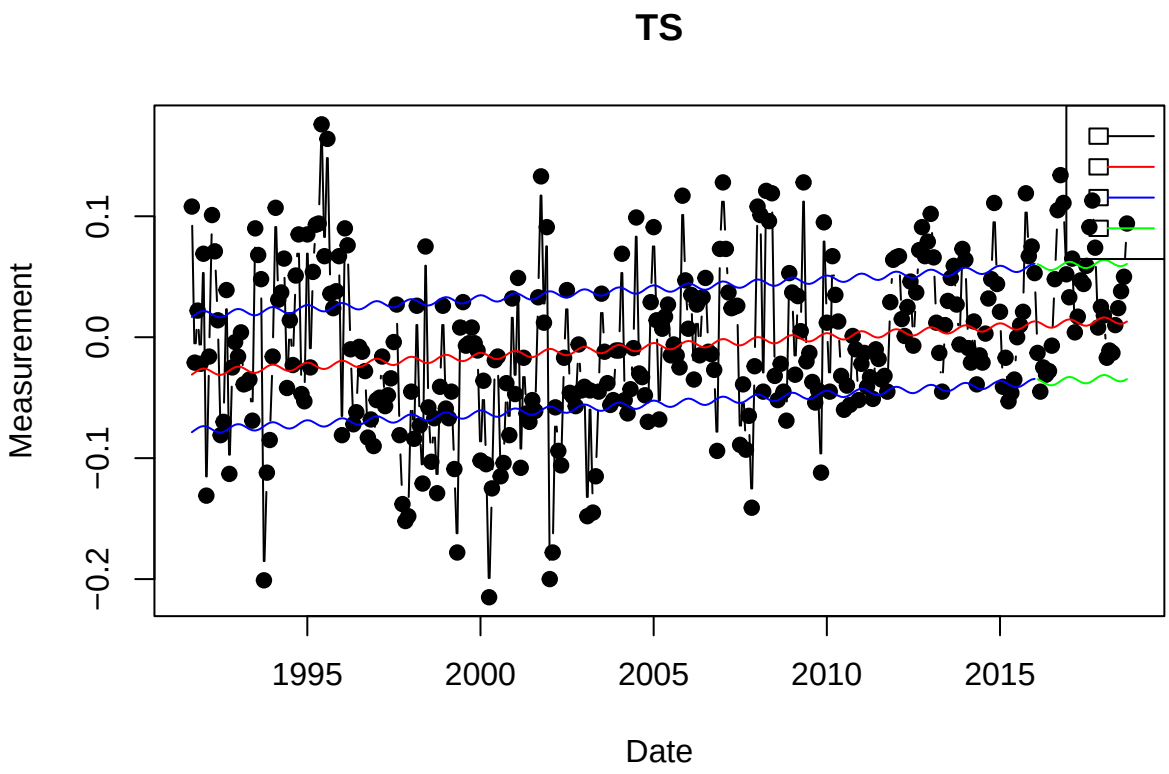
Simple time series model estimation, using OLS

The function `TSSmodel` estimates a simple model of the form:

$$Y_t = \alpha + \beta_t t + \beta_s \sin(\frac{2\pi}{p}t) + \beta_c \cos(\frac{2\pi}{p}t) \tag{1}$$

where the alpha and the betas are the estimated parameters, and p and t is set to 12 and 1 respectively by default:

```
TSSmodel(TSdata$TSmatrix[2,2,],TSdata$date)
```




```
#> [1] "Parameter estimates:"
#> [1] "alpha"
#> [1] -0.02962197
#> [1] "beta_t (trend)"
#> [1] 0.0001355023
#> [1] "beta_s (sinus)"
#> [1] 0.001612321
#> [1] "beta_c (cosinus)"
#> [1] -0.0025966
```

Bruteforcing ARIMA(p,d,q)x(P,D,Q)s models

From the time series analysis functions, some indications of adequate parameter estimates for arima/sarima models may be present, and if so, the `ARIMAbuilder` function could be useful. The inputs are the maximum number of p,q,P,Q parts in the arima model, and a fixed level of differencing: d and D. The output will display the top 5 best models, based on the Bayesian Information Criteria(BIC). BIC is chosen as it, according to Madsen 2007, yields a consistent estimate of the model order, and is given by:

$$\text{BIC} = N\log\hat{\sigma}_\epsilon^2 + \log(N)(p+q+P+Q+d+D) \tag{2}$$

with N being the number of observations in the data set. By default, all parameters are set to 1:

```
ARIMAbuilder(TSdata$TSmatrix[2,2,])
#> [1] 0 1 0 0 1 0
#> [1] 0 1 0 0 1 1
```

```
#> [1] 0 1 0 1 1 0
#> [1] 0 1 0 1 1 1
#> [1] 0 1 1 0 1 0
#> [1] 0 1 1 0 1 1
#> [1] 0 1 1 1 1 0
#> [1] 0 1 1 1 1 1
#> [1] 1 1 0 0 1 0
#> [1] 1 1 0 0 1 1
#> [1] 1 1 0 1 1 0
#> [1] 1 1 0 1 1 1
#> [1] 1 1 1 0 1 0
#> [1] 1 1 1 0 1 1
#> [1] 1 1 1 1 1 0
#> [1] 1 1 1 1 1 1
#> $`#1(p,q,P,Q)`
#>      dim1 dim2 dim3 dim4
#> [1,]    1    1    1    1
#>
#> $AIC1
#> [1] -1726.777
#>
#> $`#2(p,q,P,Q)`
#>      dim1 dim2 dim3 dim4
#> [1,]    1    1    0    1
#>
#> $AIC2
#> [1] -1719.734
#>
#> $`#3(p,q,P,Q)`
#>      dim1 dim2 dim3 dim4
#> [1,]    0    1    1    1
#>
#> $AIC3
#> [1] -1702.433
#>
#> $`#4(p,q,P,Q)`
#>      dim1 dim2 dim3 dim4
#> [1,]    0    1    0    1
#>
#> $AIC4
#> [1] -1696.014
#>
#> $`#5(p,q,P,Q)`
#>      dim1 dim2 dim3 dim4
#> [1,]    1    0    1    1
#>
#> $AIC5
#> [1] -1661.88
```