TSAIB tutorial

Mathias Høxbro Juel Vendt

2022-04-30

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
- ggpub
- forecast

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 a R package aimed at getting statistical insight, correct for intermission bias 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 assigned a name, e.g. "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.

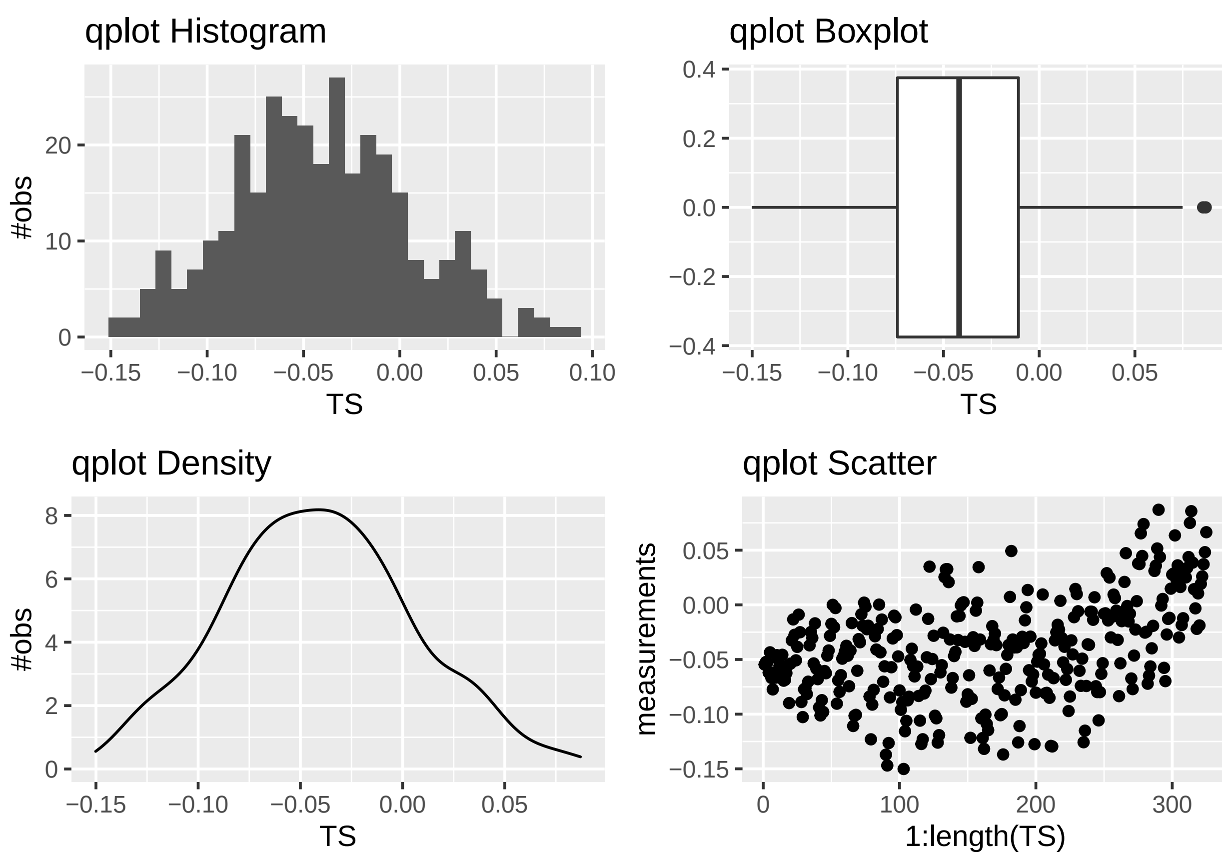
The user can also import their own data from a NetCDF file using the `GridTSExtract` function, and more info on how to use the function on a specific NetCDF file is shown by:

```
?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. For this example, a meaned 30 point grid area (`TS_area`) is analyzed, which is centered around (lon,lat)=(0,73.25). 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 if the extracted data contains a few of those, the `nanrem` option in the `TSdiagnostics` function is set to `"TRUE"`, which will omit any NaN elements for the analysis.

```
TSdiagnostics(TS_area)
```




```
#> Number of NaN's in the data set
#> [1] 0
#> Number of objects in the data set
#> [1] 325
#> Fraction of the data set which is NaN's
#> [1] 0
#> mean
#> [1] -0.04076835
#> standard deviation
#> [1] 0.04648211
#> median
```

```
#> [1] -0.04184615
#> quantile
#>          0%         25%         50%         75%        100%
#> -0.15015656 -0.07411409 -0.04184615 -0.01085852  0.08699045
#> sum
#> [1] -13.24971
#>
#>  One Sample t-test
#>
#> data:  TS
#> t = -15.812, df = 324, p-value < 2.2e-16
#> alternative hypothesis: true mean is not equal to 0
#> 95 percent confidence interval:
#>  -0.0458408 -0.0356959
#> sample estimates:
#>   mean of x
#> -0.04076835
```

Removing NaN

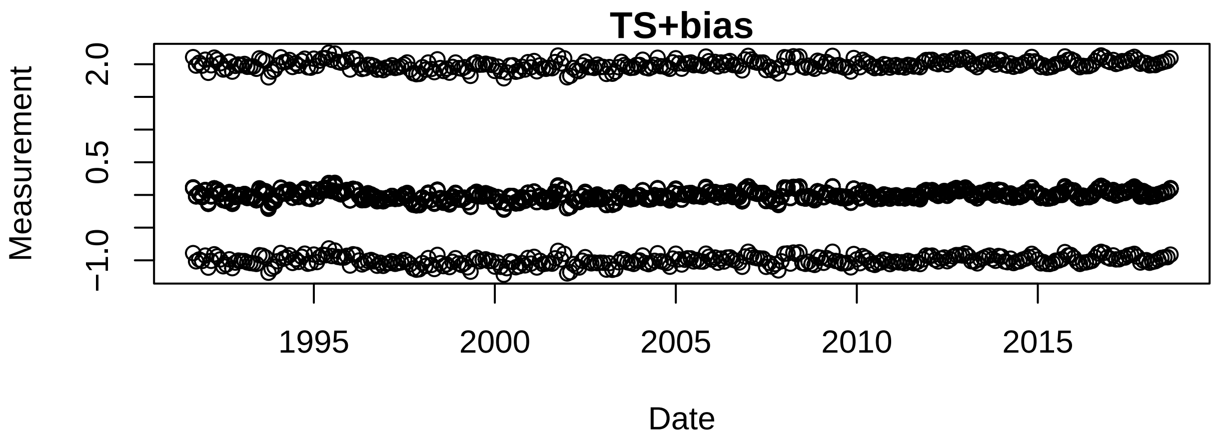
If it is decided to remove NaN objects, the function `TSnanrem` can be used to do this, with a variety of options of how to deal with NaN. For this example, the NaN's of `TSmatrix` are to be replaced with the mean of the time series:

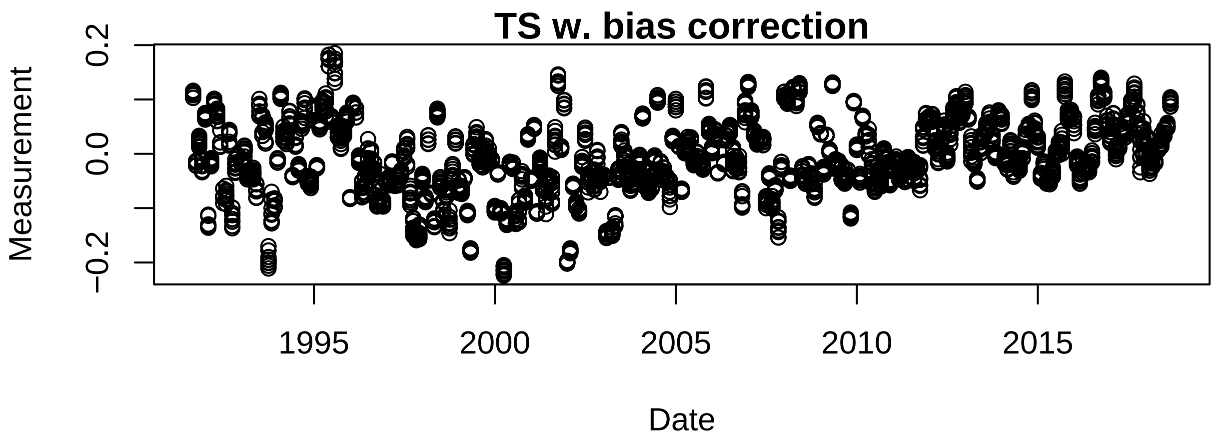
```
TSnanrem(TSdata$TSmatrix, method = "mean")
```

Intermission bias correction

Some times, the data consists of measurements from different satellite missions, and these measurements will have different biases and need to be aligned. This calls for correction of intermission biases, and if the bias is known, it can be accounted for using the `IB` function. The input to this function is a time series struct `TSM`, a date vector, a bias vector containing up to two different biases `c(bias1,bias2)` and lastly, a bias index struct, with the same dimensions as `TSM`, containing zeros for the restrained time series, and integers 1 & 2, for bias1 and bias2 respectively. In the following example, the same 3x3x325 time series as used before, has been added a bias of `c(1,-2)`, and is called `TimeSeriesMatrixBias`, with the corresponding bias index struct `biasid`.

```
IB(TimeSeriesMatrixBias,TSdata$date,c(1,-2),biasid)
```

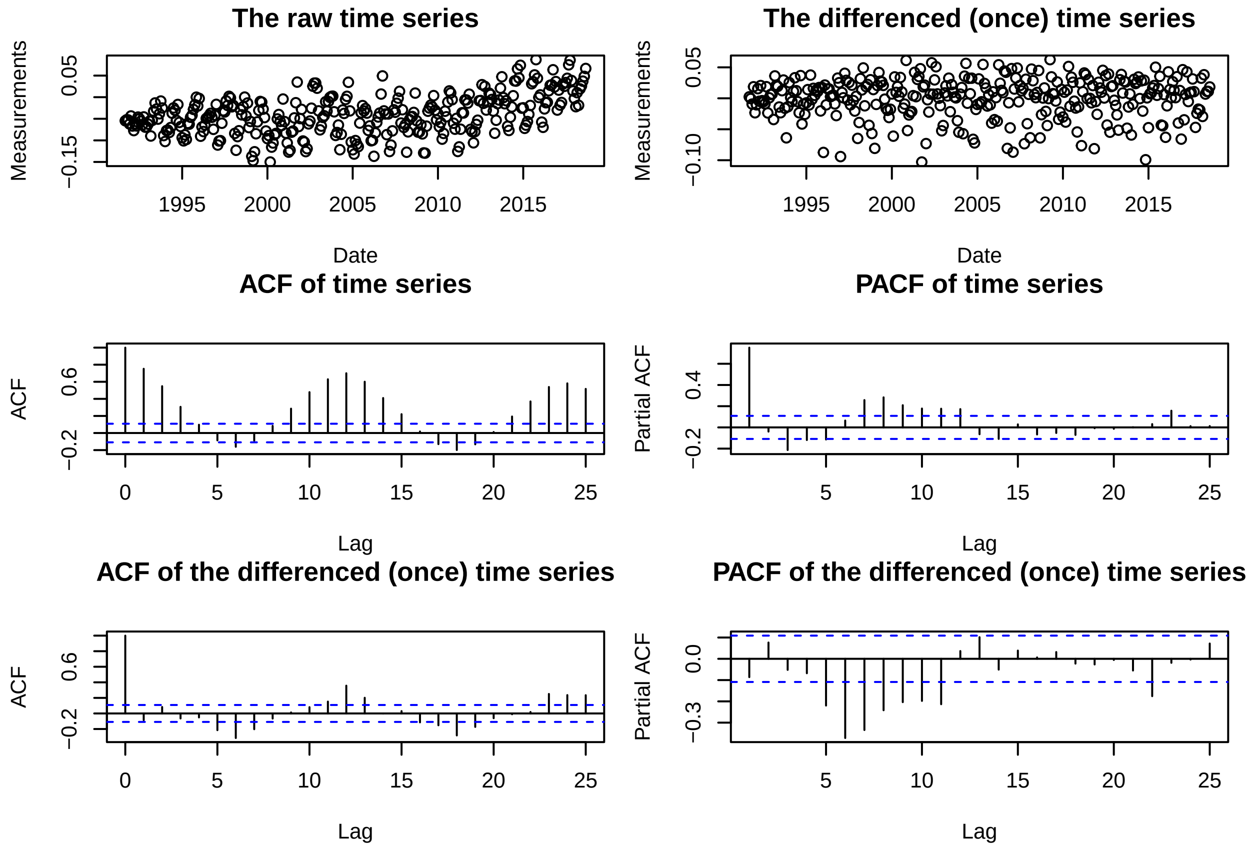




Time series analysis plots

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. The default settings is used, which doesn't transform the data or save the plots:

```
TSAplots(TS_area,TSdata$date)
```



A “tail-off” decay is observed in “ACF of time series”, indicating an AR(p) part, and a cut-off is observed at lag 3 on “PACF of time series”, indicating an AR(p=3) part could be reasonably assumed. The second row of plots show the ACF and PACF of the differenced time series (differenced once!), and the slow decay in the first ACF plot is clearly removed. Differencing once (d=1) is then a reasonable assumption. After differencing, a positive spike is seen at lag 12 on the “ACF of the differenced (once) time series”-plot, which could indicate a seasonality of 12 (S=12). This assumption is likely to be true, as the data used for this example is monthly observations of sea-level anomalies.

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:

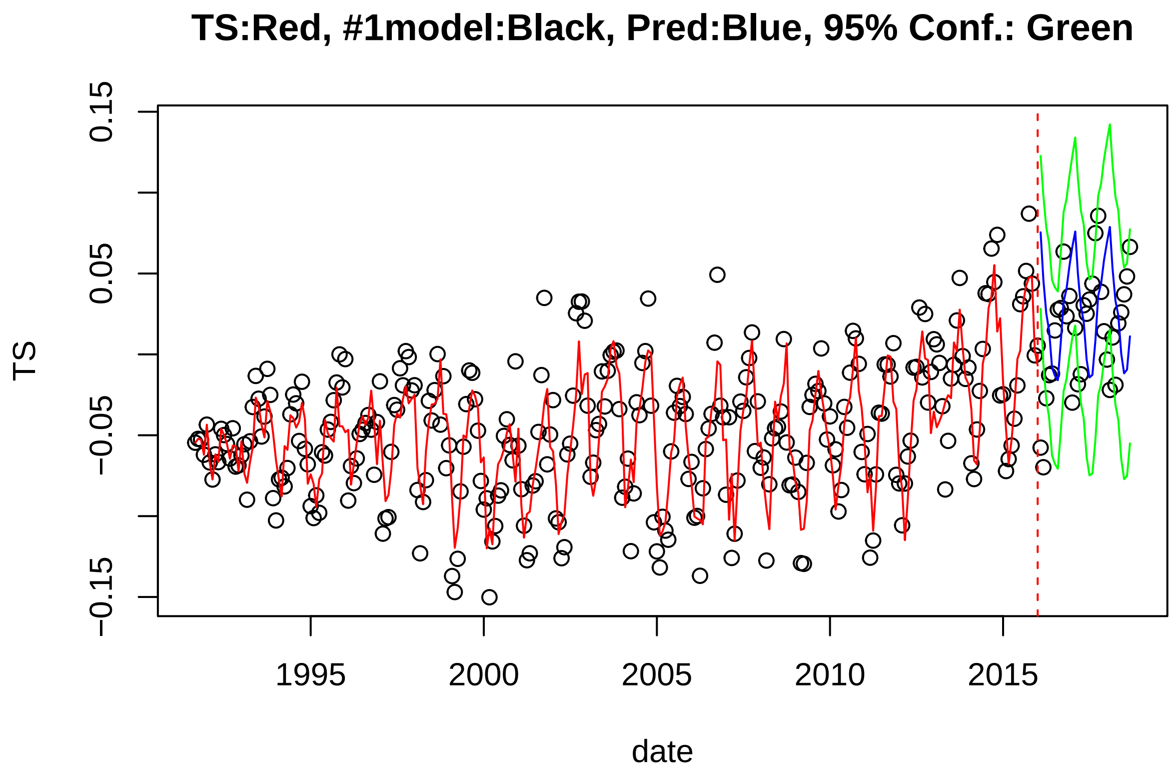
Iterative scanning of ARIMA(p,d,q)x(P,D,Q)s models

From the time series analysis functions, some indications of adequate parameter estimates for ARIMA 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 2008), yields a consistent estimate of the model order, and is given by:

$$\mathrm{BIC} = N \log \hat{\sigma}_\epsilon^2 + \log(N)(p+q+P+Q+d+D) \tag{1}$$

with N being the number of observations in the data set. By default, all parameters are set to 1, with seasonality set to 12 (i.e.: ARIMA(1,1,1)x(1,1,1)_12). For this example, however, the reasonable parameter guesses from `TSAplots` are used: ARIMA(p=3,d=1,q=?)x(P=?,D=?,Q=?)_S=12. Estimating ARIMA models is not an exact science, and ACF and PACF plots on real time series data is sometimes ambiguous. Hence the missing parameters are to be set to a reasonable guess of the maximum parameter value. The higher the parameter range, the more processing is needed to run `ARIMAbuilder`. To make the processing relatively fast, the remaining parameter estimates are set to 1: ARIMA(3,1,1,1,1,1)__12

```
ARIMAbuilder(TS_area,TSdata$date,3,1,1,1,1,1,12)
#> [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] 2 1 0 0 1 0
#> [1] 2 1 0 0 1 1
#> [1] 2 1 0 1 1 0
#> [1] 2 1 0 1 1 1
#> [1] 2 1 1 0 1 0
#> [1] 2 1 1 0 1 1
#> [1] 2 1 1 1 1 0
#> [1] 2 1 1 1 1 1
#> [1] 3 1 0 0 1 0
#> [1] 3 1 0 0 1 1
#> [1] 3 1 0 1 1 0
#> [1] 3 1 0 1 1 1
#> [1] 3 1 1 0 1 0
#> [1] 3 1 1 0 1 1
#> [1] 3 1 1 1 1 0
#> [1] 3 1 1 1 1 1
```




```
#> $`#1(p,q,P,Q)`
#>      dim1 dim2 dim3 dim4
#> [1,]    1    1    1    1
#>
#> $BIC1
#> [1] -2301.147
#>
#> $`#2(p,q,P,Q)`
#>      dim1 dim2 dim3 dim4
#> [1,]    2    1    1    1
#>
#> $BIC2
#> [1] -2297.367
#>
#> $`#3(p,q,P,Q)`
#>      dim1 dim2 dim3 dim4
#> [1,]    3    1    1    1
#>
#> $BIC3
#> [1] -2295.258
#>
#> $`#4(p,q,P,Q)`
#>      dim1 dim2 dim3 dim4
#> [1,]    1    1    0    1
#>
#> $BIC4
```

```
#> [1] -2284.659
#>
#> $`#5(p,q,P,Q)`
#>      dim1 dim2 dim3 dim4
#> [1,]    2    1    0    1
#>
#> $BIC5
#> [1] -2281.368
```

It is seen that ARIMA(3,1,1)x(1,1,1)_12 is on third place, ARIMA(2,1,1)x(1,1,1)_12 is on second place, and the best model based on BIC is: ARIMA(1,1,1)x(1,1,1)_12, and that is shown in the figure above.

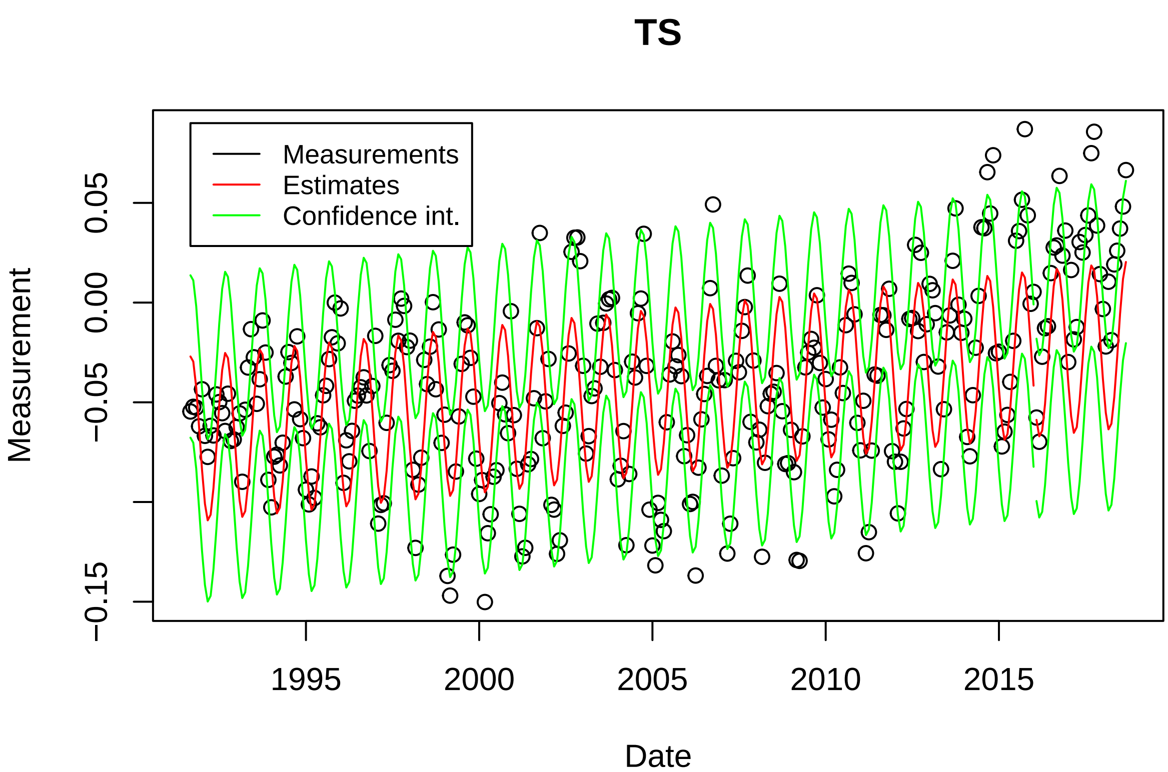
Simple time series model estimation, using OLS

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

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

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

```
TSSmodel(TS_area,TSdata$date)
```




```
#> [1] "Parameter estimates:"
```

```
#> [1] "alpha"
#> [1] -0.06869383
#> [1] "beta_t (trend)"
#> [1] 0.0001462904
#> [1] "beta_s (sinus)"
#> [1] 0.02571606
#> [1] "beta_c (cosinus)"
#> [1] 0.03311466
```

```
TMBTSA(TimeSeriesMatrixBias,TSdata$date,biasid)
#> Optimizing tape... Done
#> iter: 1  value: 3777.837 mgc: 2.569 ustep: 1
#> iter: 2  mgc: 8.604228e-16
#> iter: 1  mgc: 8.604228e-16
#> Matching hessian patterns... Done
#> outer mgc:  1061.659
#> iter: 1  mgc: 8.604228e-16
#> iter: 1  mgc: 8.604228e-16
#> outer mgc:  1061.659
#> iter: 1  value: 3018.259 mgc: 9.275589 ustep: 1
#> iter: 2  mgc: 7.431555e-15
#> iter: 1  mgc: 7.431555e-15
#> outer mgc:  2017.715
#> iter: 1  value: 3475.424 mgc: 2.775149 ustep: 1
#> iter: 2  mgc: 1.831868e-15
#> iter: 1  value: 2906.777 mgc: 0.6846696 ustep: 1
#> iter: 2  mgc: 1.812786e-15
#> iter: 1  mgc: 1.812786e-15
#> outer mgc:  655.4666
#> iter: 1  value: 2704.404 mgc: 1.041508 ustep: 1
#> iter: 2  mgc: 2.331468e-15
#> iter: 1  mgc: 2.331468e-15
#> outer mgc:  302.2869
#> iter: 1  value: 2698.871 mgc: 3.279067 ustep: 1
#> iter: 2  mgc: 3.727053e-15
#> iter: 1  value: 2653.696 mgc: 1.348685 ustep: 1
#> iter: 2  mgc: 2.803313e-15
#> iter: 1  mgc: 2.803313e-15
#> outer mgc:  239.4027
#> iter: 1  value: 2609.935 mgc: 1.282868 ustep: 1
#> iter: 2  mgc: 2.709638e-15
#> iter: 1  mgc: 2.709638e-15
#> outer mgc:  221.2975
#> iter: 1  value: 2502.296 mgc: 2.328355 ustep: 1
#> iter: 2  mgc: 3.219647e-15
#> iter: 1  mgc: 3.219647e-15
#> outer mgc:  336.5034
#> iter: 1  value: 2402.44 mgc: 3.075157 ustep: 1
#> iter: 2  mgc: 3.330669e-15
#> iter: 1  mgc: 3.330669e-15
#> outer mgc:  288.362
#> iter: 1  value: 2286.802 mgc: 1.625555 ustep: 1
#> iter: 2  mgc: 4.940492e-15
#> iter: 1  mgc: 4.940492e-15
```

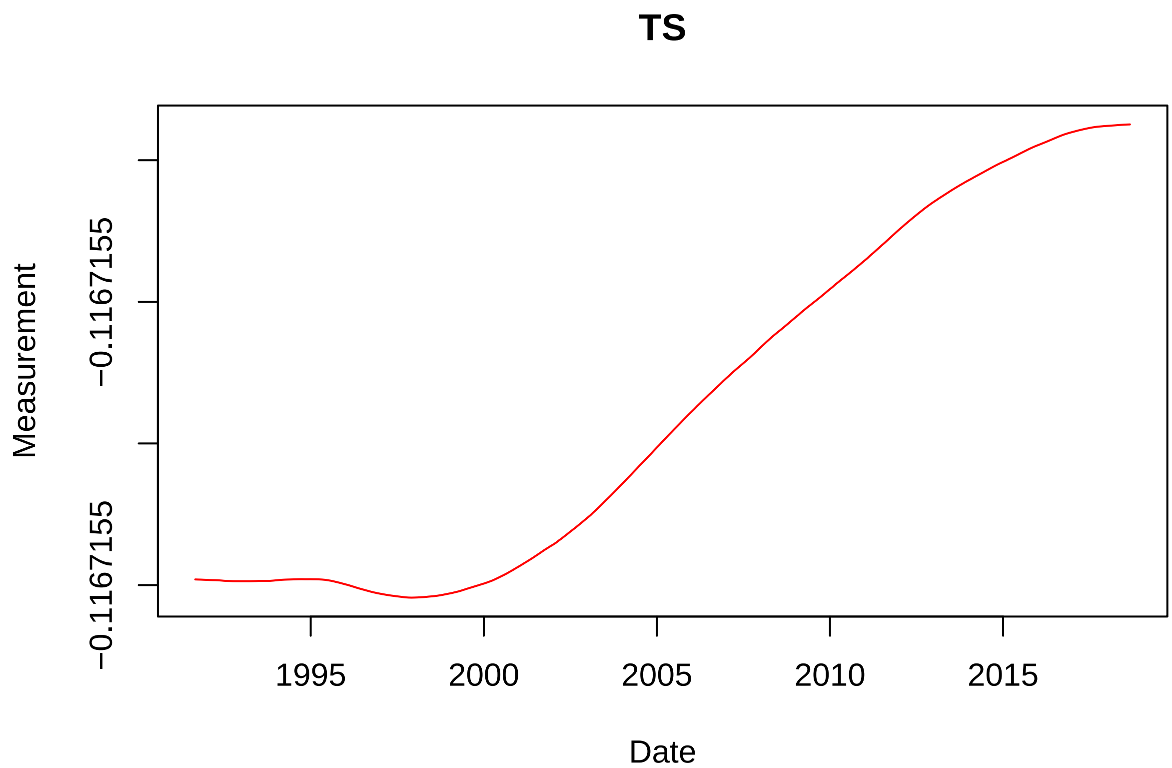
```
#> outer mgc:  558.2035
#> iter: 1  value: 2183.58 mgc: 4.823007 ustep: 1
#> iter: 2  mgc: 6.050715e-15
#> iter: 1  mgc: 6.050715e-15
#> outer mgc:  278.0973
#> iter: 1  value: 2058.785 mgc: 8.859986 ustep: 1
#> iter: 2  mgc: 1.376677e-14
#> iter: 1  mgc: 1.376677e-14
#> outer mgc:  188.9604
#> iter: 1  value: 1967.97 mgc: 27.32878 ustep: 1
#> iter: 2  mgc: 3.158585e-14
#> iter: 1  value: 2025.023 mgc: 4.1573 ustep: 1
#> iter: 2  mgc: 1.554312e-14
#> iter: 1  mgc: 1.554312e-14
#> outer mgc:  259.1656
#> iter: 1  value: 2010.176 mgc: 1.368793 ustep: 1
#> iter: 2  mgc: 1.471046e-14
#> iter: 1  mgc: 1.471046e-14
#> outer mgc:  112.1497
#> iter: 1  value: 1984.034 mgc: 1.487697 ustep: 1
#> iter: 2  mgc: 1.465494e-14
#> iter: 1  mgc: 1.465494e-14
#> outer mgc:  99.83354
#> iter: 1  value: 1957.996 mgc: 1.315417 ustep: 1
#> iter: 2  mgc: 1.731948e-14
#> iter: 1  mgc: 1.731948e-14
#> outer mgc:  40.84696
#> iter: 1  value: 1903.95 mgc: 1.511563 ustep: 1
#> iter: 2  mgc: 2.131628e-14
#> iter: 1  value: 1826.688 mgc: 2.681886 ustep: 1
#> iter: 2  mgc: 3.064216e-14
#> iter: 1  value: 1659.139 mgc: 8.274488 ustep: 1
#> iter: 2  mgc: 1.071088e-13
#> iter: 1  mgc: 1.071088e-13
#> outer mgc:  95.39767
#> iter: 1  value: 1684.289 mgc: 3264.03 ustep: 1
#> iter: 2  mgc: 1.307399e-12
#> iter: 1  value: 1649.838 mgc: 199.4227 ustep: 1
#> iter: 2  mgc: 3.055334e-13
#> iter: 1  mgc: 3.055334e-13
#> outer mgc:  15.0494
#> iter: 1  value: 1634.393 mgc: 186.1128 ustep: 1
#> iter: 2  mgc: 1.368974e-13
#> iter: 1  value: 1640.611 mgc: 63.3561 ustep: 1
#> iter: 2  mgc: 1.199041e-13
#> iter: 1  mgc: 1.199041e-13
#> outer mgc:  61.65922
#> iter: 1  value: 1639.047 mgc: 91.18057 ustep: 1
#> iter: 2  mgc: 1.048051e-13
#> iter: 1  mgc: 1.048051e-13
#> outer mgc:  46.89791
#> iter: 1  value: 1629.213 mgc: 69.43221 ustep: 1
#> iter: 2  mgc: 1.181277e-13
```

```
#> iter: 1  mgc: 1.181277e-13
#> outer mgc:  18.8698
#> iter: 1  value: 1619.753 mgc: 75.12839 ustep: 1
#> iter: 2  mgc: 1.29674e-13
#> iter: 1  mgc: 1.29674e-13
#> outer mgc:  26.11239
#> iter: 1  value: 1607.02 mgc: 28.89981 ustep: 1
#> iter: 2  mgc: 1.287859e-13
#> iter: 1  mgc: 1.287859e-13
#> outer mgc:  28.65113
#> iter: 1  value: 1593.829 mgc: 23.59686 ustep: 1
#> iter: 2  mgc: 1.301181e-13
#> iter: 1  mgc: 1.301181e-13
#> outer mgc:  12.73748
#> iter: 1  value: 1580.519 mgc: 21.01104 ustep: 1
#> iter: 2  mgc: 1.607603e-13
#> iter: 1  value: 1540.458 mgc: 86.07908 ustep: 1
#> iter: 2  mgc: 2.033929e-13
#> iter: 1  value: 1473.161 mgc: 221.1544 ustep: 1
#> iter: 2  mgc: 3.419487e-13
#> iter: 1  mgc: 3.419487e-13
#> outer mgc:  86.12042
#> iter: 1  value: 1348.931 mgc: 610.4088 ustep: 1
#> iter: 2  mgc: 6.252776e-13
#> iter: 1  value: 1195.837 mgc: 1967.385 ustep: 1
#> iter: 2  mgc: 1.756817e-12
#> iter: 1  mgc: 1.756817e-12
#> outer mgc:  48.6914
#> iter: 1  value: 1108.181 mgc: 6911.485 ustep: 1
#> iter: 2  mgc: 6.458833e-12
#> iter: 1  value: 1151.394 mgc: 2567.432 ustep: 1
#> iter: 2  mgc: 2.071232e-12
#> iter: 1  mgc: 2.071232e-12
#> outer mgc:  30.62496
#> iter: 1  value: 1104.875 mgc: 415.5129 ustep: 1
#> iter: 2  mgc: 2.913225e-12
#> iter: 1  value: 1048.243 mgc: 714.5264 ustep: 1
#> iter: 2  mgc: 5.10525e-12
#> iter: 1  mgc: 5.10525e-12
#> outer mgc:  17.59704
#> iter: 1  value: 956.8339 mgc: 1580.036 ustep: 1
#> iter: 2  mgc: 8.615331e-12
#> iter: 1  mgc: 8.615331e-12
#> outer mgc:  10.06241
#> iter: 1  value: 865.418 mgc: 208.5247 ustep: 1
#> iter: 2  mgc: 1.461986e-11
#> iter: 1  mgc: 1.461986e-11
#> outer mgc:  3.617258
#> iter: 1  value: 750.5326 mgc: 1854.784 ustep: 1
#> iter: 2  mgc: 1.59428e-11
#> iter: 1  mgc: 1.59428e-11
#> outer mgc:  1.249794
#> iter: 1  value: 635.5525 mgc: 221.5711 ustep: 1
```

```
#> iter: 2  mgc: 3.183231e-11
#> iter: 1  mgc: 3.183231e-11
#> outer mgc:  8.343409
#> iter: 1  value: 520.4755 mgc: 8358.4 ustep: 1
#> iter: 2  mgc: 6.188827e-11
#> iter: 1  value: 368.6541 mgc: 28094.79 ustep: 1
#> iter: 2  mgc: 1.638085e-10
#> iter: 1  mgc: 6.188827e-11
#> outer mgc:  2.243337
#> iter: 1  value: 405.3424 mgc: 7694.568 ustep: 1
#> iter: 2  mgc: 1.223555e-10
#> iter: 1  mgc: 1.223555e-10
#> outer mgc:  7.5777
#> iter: 1  value: 290.1654 mgc: 834.0145 ustep: 1
#> iter: 2  mgc: 2.427312e-10
#> iter: 1  mgc: 2.427312e-10
#> outer mgc:  5.283105
#> iter: 1  value: 163.2211 mgc: 7834.292 ustep: 1
#> iter: 2  mgc: 5.706298e-10
#> iter: 1  mgc: 5.706298e-10
#> outer mgc:  0.7389044
#> iter: 1  value: 42.39956 mgc: 8539.729 ustep: 1
#> iter: 2  mgc: 1.071477e-09
#> iter: 1  mgc: 1.071477e-09
#> outer mgc:  1.370749
#> iter: 1  value: -72.46146 mgc: 9764.334 ustep: 1
#> iter: 2  mgc: 2.301029e-09
#> iter: 1  mgc: 2.301029e-09
#> outer mgc:  0.07443649
#> iter: 1  value: -187.3247 mgc: 11839.2 ustep: 1
#> iter: 2  mgc: 4.484746e-09
#> iter: 1  mgc: 4.484746e-09
#> outer mgc:  0.3770449
#> iter: 1  value: -300.6353 mgc: 11540.72 ustep: 1
#> iter: 2  mgc: 9.394114e-09
#> iter: 1  mgc: 9.394114e-09
#> outer mgc:  0.01980595
#> iter: 1  value: -413.9001 mgc: 11370.44 ustep: 1
#> iter: 2  value: -413.9001 mgc: 2.012898e-08 ustep: 1
#> mgc: 1.856786e-08
#> iter: 1  value: -413.9001 mgc: 1.856786e-08 ustep: 1
#> mgc: 1.856786e-08
#> outer mgc:  0.09085644
#> iter: 1  value: -526.6577 mgc: 10322.76 ustep: 1
#> iter: 2  value: -526.6577 mgc: 4.040994e-08 ustep: 1
#> mgc: 3.786153e-08
#> iter: 1  value: -526.6577 mgc: 3.786153e-08 ustep: 1
#> mgc: 3.786153e-08
#> outer mgc:  0.008316976
#> iter: 1  value: -639.4154 mgc: 9584.881 ustep: 1
#> iter: 2  value: -639.4154 mgc: 7.421365e-08 ustep: 1
#> mgc: 7.287803e-08
#> iter: 1  value: -639.4154 mgc: 7.287803e-08 ustep: 1
```

```
#> mgc: 7.287803e-08
#> outer mgc:  0.02123904
#> iter: 1  value: -752.0866 mgc: 8675.487 ustep: 1
#> iter: 2  value: -752.0866 mgc: 1.371002e-07 ustep: 1
#> mgc: 1.600472e-07
#> iter: 1  value: -752.0866 mgc: 1.600472e-07 ustep: 1
#> mgc: 1.600472e-07
#> outer mgc:  0.002857104
#> iter: 1  value: -864.7539 mgc: 7946.556 ustep: 1
#> iter: 2  value: -864.7539 mgc: 3.151571e-07 ustep: 1
#> mgc: 2.75915e-07
#> iter: 1  value: -864.7539 mgc: 2.75915e-07 ustep: 1
#> mgc: 2.75915e-07
#> outer mgc:  0.004975685
#> iter: 1  value: -977.3971 mgc: 7226.838 ustep: 1
#> iter: 2  value: -977.3971 mgc: 6.635843e-07 ustep: 1
#> mgc: 6.327755e-07
#> iter: 1  value: -977.3971 mgc: 6.327755e-07 ustep: 1
#> mgc: 6.327755e-07
#> outer mgc:  0.0008734387
#> iter: 1  value: -1090.04 mgc: 6595.521 ustep: 1
#> iter: 2  value: -1090.04 mgc: 1.072883e-06 ustep: 1
#> mgc: 1.072883e-06
#> iter: 1  value: -1090.04 mgc: 1.072883e-06 ustep: 1
#> mgc: 1.072883e-06
#> outer mgc:  0.001173319
#> iter: 1  value: -1202.678 mgc: 6012.828 ustep: 1
#> iter: 2  value: -1202.678 mgc: 2.348753e-06 ustep: 1
#> mgc: 2.348753e-06
#> iter: 1  value: -1202.678 mgc: 2.348753e-06 ustep: 1
#> mgc: 2.348753e-06
#> outer mgc:  0.00025101
#> iter: 1  value: -1315.315 mgc: 5480.747 ustep: 1
#> iter: 2  value: -1315.315 mgc: 5.1008e-06 ustep: 1
#> mgc: 5.1008e-06
#> iter: 1  value: -1315.315 mgc: 5.1008e-06 ustep: 1
#> mgc: 5.1008e-06
#> outer mgc:  0.0002785884
#> iter: 1  value: -1427.947 mgc: 5001.982 ustep: 1
#> iter: 2  value: -1427.947 mgc: 1.035171e-05 ustep: 1
#> mgc: 1.035171e-05
#> iter: 1  value: -1427.947 mgc: 1.035171e-05 ustep: 1
#> mgc: 1.035171e-05
#> outer mgc:  6.950157e-05
#> iter: 1  value: -1540.58 mgc: 4557.258 ustep: 1
#> iter: 2  value: -1540.58 mgc: 1.984985e-05 ustep: 1
#> mgc: 1.984985e-05
#> iter: 1  value: -1540.58 mgc: 1.984985e-05 ustep: 1
#> mgc: 1.984985e-05
#> outer mgc:  6.658116e-05
#> iter: 1  value: -1540.58 mgc: 1.984985e-05 ustep: 1
#> mgc: 1.984985e-05
#> iter: 1  value: -1540.58 mgc: 1.984985e-05 ustep: 1
```

```
#> mgc: 1.984985e-05
#> outer mgc:  6.658116e-05
#> iter: 1  value: -1540.255 mgc: 0.01072662 ustep: 1
#> mgc: 1.88616e-05
#> outer mgc:  6.668407e-05
#> iter: 1  value: -1540.905 mgc: 0.01074604 ustep: 1
#> mgc: 1.72667e-05
#> outer mgc:  6.647845e-05
#> iter: 1  value: -1540.577 mgc: 0.01072457 ustep: 1
#> mgc: 1.72322e-05
#> outer mgc:  5.836162
#> iter: 1  value: -1540.577 mgc: 0.0107481 ustep: 1
#> mgc: 1.889936e-05
#> outer mgc:  5.847845
#> iter: 1  value: -1540.575 mgc: 793709783 ustep: 1
#> iter: 2  value: -1540.575 mgc: 1.838257e-05 ustep: 1
#> mgc: 1.767625e-05
#> outer mgc:  8.335443
#> iter: 1  value: -1540.575 mgc: 793709783 ustep: 1
#> iter: 2  value: -1540.575 mgc: 1.903041e-05 ustep: 1
#> mgc: 1.908728e-05
#> outer mgc:  8.33531
#> iter: 1  value: -1540.577 mgc: 0.008580671 ustep: 1
#> mgc: 2.036405e-05
#> outer mgc:  4.637133
#> iter: 1  value: -1540.577 mgc: 0.008577463 ustep: 1
#> mgc: 2.169021e-05
#> outer mgc:  4.637088
```



Appendices

Data set used for demonstration of the TSAIB package:

TSdata

```
?TSdata
 Description
  An extracted list using the "GridTSExtract" function, from the data set:
  Rose, S.K.; Andersen, O.B.; Passaro, M.; Ludwigsen, C.A.; Schwatke,
  C. Arctic Ocean Sea Level Record from the Complete Radar
  Altimetry Era: 1991-2018. Remote Sens. 2019, 11, 1672.
  https://www.mdpi.com/2072-4292/11/14/1672
 Usage
  TSdata
 Format
  A large list containing 5 elements, which are:
  longitude: containing longitudes from -180:180, by increments of 0.5 degrees
  (length: 720)
  latitude: containing latitudes from 65:81.5, by increments of 0.25 degrees (length: 67)
  date: containing years from 1991:2019, by increments of 1/12 (length:325)
  measurements: containing sea_level_anomaly measurements for each increments of:
  longitude, latitude and date. (dimension: [67,720,325])
  TSmatrix: containing a 3x3x325 grid of measurements centered around (lon,lat)=(-165,74)
```

Functions

GridTSExtract

```
?GridTSExtract
 Description
  Extracts a time series from a NetCDF lon&lat coordinate grid
 Usage
GridTSExtract(
  nco,
  lonid = "longitude",
  latid = "latitude",
  timeid = "date",
  measurementsid = "sea_level_anomaly",
  coord = c(-165, 74),
  radius = 0,
  dlon = 2,
  dlat = 4
)
 Arguments
  nco: The NetCDF object (open with "nc_open" from the "ncdf4" library)
  lonid: The variable id for longitude vector. e.g.: "longitude"
  latid: The variable id for latitude vector. e.g.: "latitude"
  timeid: The variable id for the time vector. e.g.: "date"
  measurementsid: The variable id for the measurements vector. e.g.: "sea_level_anomaly"
  coord: The center coordinate for the TS grid area. e.g.: c(lon,lat) i.e. c(-164,74)
  radius: The square radius of the TS grid. e.g: 0=1x1 grid, 1=3x3 grid, 2=5x5 grid etc.
  dlon: Number of data points for each degree longitude
  dlat:Number of data points for each degree latitude
 Value: Time series from grid
```

TSdiagnostics

```
?TSdiagnostics
 Description
  Shows basic statistics and characteristics of the time series data
 Usage
  TSdiagnostics(TS, nanrem = "FALSE")
 Arguments
  TS:The time series to be analyzed
  nanrem: Set NaN to be removed or not. e.g.: nanrem="TRUE"
 Value: Basic statistics and characteristics
```

TSnanrem

```
?TSnanrem
 Description
  Removes or imputes NaN from the time series data set, with a specified method of choice
 Usage
  TSnanrem(TS, method = "mean", value_nr = 0)
 Arguments
  TS: The time series to remove NaN from
  method: The imputation method. e.g.: "omit", "mean", "median" or "value"
```

```
  value_nr: The specific value for the "value" imputing method. e.g.: value_nr=0
 Value: Time series with NaNs removed or replaced
```

IB

```
?IB
 Description
  Corrects a struct of timeseries for up to two intermission biases
 Usage
  IB(TSM, date, biasvec = c(1, -2), biasid)
 Arguments
  TSM: The time series struct with the intermission bias
  date: The time axis in the time series (dates, seconds, intervals)
  biasvec: The vector containging up to 2 different bias. If the data only contains one bias, insert tha
  biasid: A struct with the same dimensions as TSM, with integers indicating the bias: 0 for the restrai
Value: Time series struct with bias correction
```

TSAplots

```
?TSAplots
 Description
  Plots and saves the ACF and PACF for a pre-defined time series, with and without data
  transformations (log, sqrt), and differencing. Input type: Time series
 Usage
  TSAplots(TS, date, trans = "NONE", saveas = "NONE")
 Arguments
  TS: The time series for the TSAplots function
  date: The time axis in the time series (dates, seconds, intervals)
  trans: data transform parameter: trans='log' for log transform, trans='sqrt' for sqrt
  transform, and trans='NONE' for no transform
  saveas: specifies the file format to save the plots in the current working directory.
  Possible formats is: PDF, JPEG, TIFF, BMP and PNG. e.g.: "pdf" or "jpeg".
  If no format is specified, it will not save the plots
 Value: Plots from TSAplots
```

TSSmodel

```
?TSSmodel
 Description
  Estimates a simple time series model

 Usage
  TSSmodel(TS, date, p = 12, t = 1, testsize = round(length(date)/10))
 Arguments
  TS: The time series for the TSSmodel function
  date: The time axis in the time series (dates, seconds, intervals)
  p: The period. e.g.: p=12, for 12 months in a year
  t: The time steps. e.g.: t=1, for monthly time steps, with period 12
  testsize: The number of data points to be used for testing the model
 Value: Simple TS model
```

ARIMAbuilder

```
?ARIMAbuilder
 Description
  Lists the top 5 estimated ARIMA/SARIMA models of the form: ARIMA(p,d,q)x(P,D,Q),
  based on specified parameter values and ranges
 Usage
  ARIMAbuilder(TS, p = 1, d = 1, q = 1, P = 1, D = 1, Q = 1, S = 12)
 Arguments
  TS: The time series to build the models upon (remember to transform before
  using this function!)
  p: The maximum AR value, estimated from plot analysis
  d: The fixed differencing, often 0, 1 or 2
  q:    The maximum MA value, estimated from plot analysis
  P:    The maximum seasonal AR value, estimated from plot analysis
  D:    The fixed seasonal differencing, often 0, 1 or 2
  Q:    The maximum seasonal MA value, estimated from plot analysis
  S:    The seasonality/period of the seasonal differencing
 Value: list of top 5 ARIMA/SARIMA models
```

References

Madsen, Henrik. 2008. "Time Series Analysis." http://www.http://henrikmadsen.org/books/time-series-analysis/.