User guide to the **ST**atistical **A**nalogue **R**esampling **S**cheme – Version 2.4

Boris Orlowsky, Julia Lutz

June 14, 2013

Contents

This program is open source – you may use and adapt it whenever it appears to be useful. However, it comes without any warranty. Climate simulations obtained by this model should be considered as an estimation of possible future development, not as predictions. Even though the program has been tested and evaluated carefully, no guarantee can be given with respect to its bug-freeness. So if you use it, use it at your own risk. And cite its published foundations: Orlowsky [2007] and Orlowsky et al. [2008]. Further model applications can be found in Orlowsky and Fraedrich [2009], Orlowsky et al. [2010] and Lutz et al. [2013].

This manual is organised as follows: After a brief description of the steps within the model and their corresponding objects in the implementation, the next two sections will describe how to install and use the program. Finally, the single source files are summarised. The model works with station as well as with gridded data, e. g. from GCM or RCM simulations.

The program consists of two (three) parts: **stars_2.4.exe**, which generates a date-to-date-mapping (see Section 1), and **post_process.exe**, which generates the final simulation files from the date-to-date-mapping, with optional corrections for inter annual variability and a smoothing of the series between observation and simulation period. Optionally, **pre_process.exe** can be used in order to derive a meaningful set of representative stations (see Section 1.2)

1 The model, a brief description

This model generates simulated time series of climate data on a regional scale, based on station or gridded data. One of its most important features is that the simulated series are constrained only by the parameters of a linear regression line. The annual means of the generated time series of a chosen climate variable have to match these parameters. Inspired by Werner and Gerstengarbe [1997], the simulated series are assembled from segments of the observed series: The approach generates a mapping from dates of a simulation period to dates of the observation period, hence resampling the observation time series. This resampling is constructed such that the corresponding series yield annual means of the chosen climate variable, which feature the prescribed regression line. Furthermore, a set of heuristic rules makes sure that the resulting series exhibit realistic properties such as annual cycles or persistence. See Figure 1 for an illustration of this resampling scheme, for which temperature is chosen as the characteristic variable.

Here, only a brief description of this construction is given. For a detailed description, the reader is referred to Orlowsky et al. [2008].

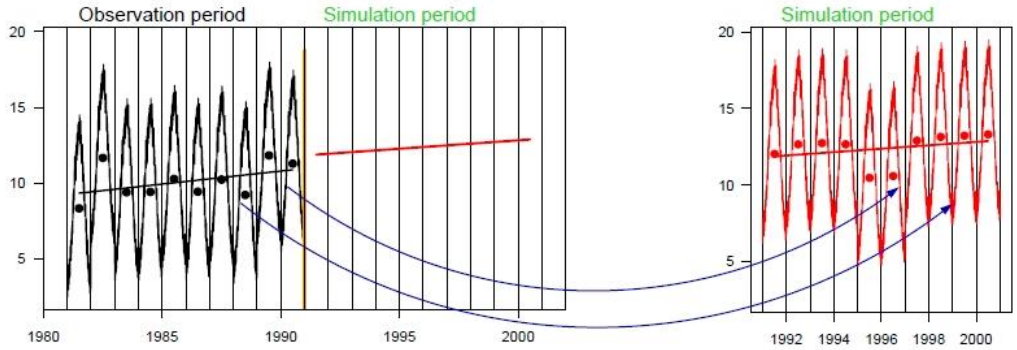

Figure 1: Assembling simulated series from segments of observed series, corresponding to a prescribed regression line. Left: observation series and prescribed regression line for temperature. Right: simulated series with annual means (dots), featuring the prescribed regression line.

1.1 Outline for a single location (station or grid point)

If only a series for a specific location, meteorological station or grid point, is needed, no spatial dependencies have to be taken into account. Figure 2 summarises the generation of the date-to-date-mapping for this case. It comprises two steps operating at different time scales.

The first step operates on the time scale of years (parts of the flow chart concerned with this time scale are highlighted in grey). It generates a first approximation to the mapping, which consists of a simple rearrangement of entire calendar years from the observation period. This rearrangement is chosen out of a large random sample of shuffled calendar years, such that its corresponding temperature series is as close as possible to the prescribed regression line and to the observed inter annual variance. This series is guaranteed to exhibit realistic annual cycles and weather sequences within the single years, as they are simply copied from the observed series. Choosing the rearrangement which is the closest to both the prescribed temperature regime and the inter annual variance from the observation period helps to prevent a loss of long-term variability to which resampling schemes like STARS are prone. [See in Sec. 4.2 **prepare_years.cpp** and in Sec. 4.3: **generate_fa.cpp**]

The second step iteratively alters this first approximation in order to find a series which matches the prescribed regression parameters exactly. This step replaces segments of a given number of consecutive days, according to

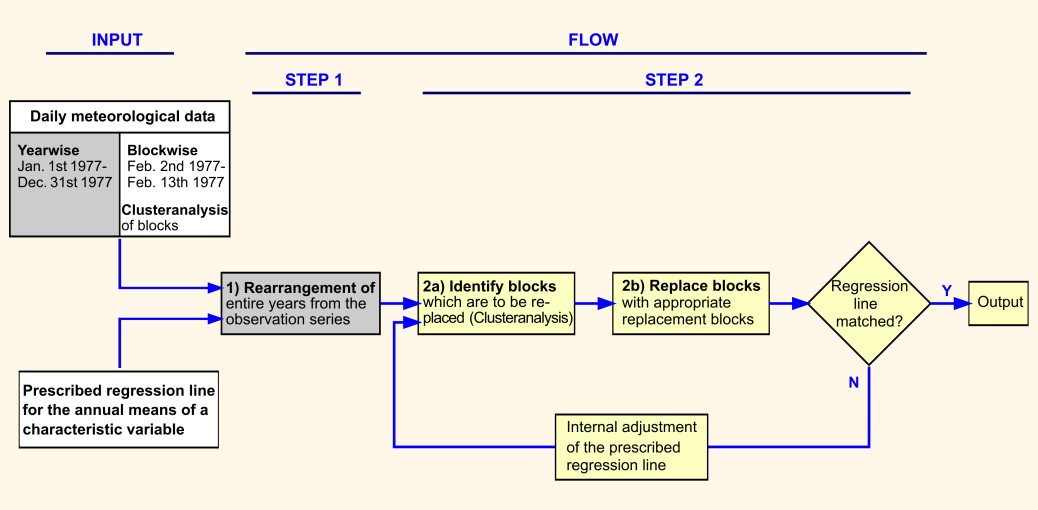

Figure 2: Summary of the generation of the date-to-date-mapping.

the parameter blocklength (see Section 3.2.3).

Replacing blocks instead of single days helps to obtain simulated series with realistic persistences, as the weather sequences within the blocks are entirely copied from the observed series. Experiments with station data in Central Europe have shown that a block length of 12 days essentially captures the persistence of the observed time series. This number might be different for other regions in the world. Thus, this should be tested and the number should be adjusted if necessary. [see in Sec. 4.2: **prepare_blocks.cpp**]

The blocks which are to be replaced are identified using a heuristic criterion, which decides, loosely speaking, whether any given block in the first approximation series contributes too much to the mismatch between the prescribed and the first approximation regression line. For this, a cluster analysis of the blocks is required, which classifies blocks into groups consisting of similar blocks, where "similar" refers to the temperature observations of the blocks. [See in Sec. 4.4: **identify.cpp**]

The replacing blocks are selected from the blocks of the observation period according to several heuristic rules. Besides temperature observations, these rules make use of the calendar dates of the blocks, in order to select blocks from appropriate seasons. They define a set of potential replacements, which, firstly, bring the regression line of the resulting series closer to the prescribed one and, secondly, make sure that the inserted replacement fits well into the parts of the series which have been set already. From this set of potential replacements, a block is chosen randomly. After the replacement of all "bad" blocks, the date-to-date-mapping is defined for the current iteration. [See in

Sec. 4.4: **replace.cpp**, **used_red.cpp**, **date_red.cpp**, **bridge_red.cpp**]

If despite these replacements the regression line of the simulated annual temperature means does not match the prescribed parameters (within the given tolerance, see Section 3.2.3), this second step is iteratively repeated. Therefore, in order to compensate for a potential bias from the previous iteration, the following iteration is prescribed with exaggerated regression parameters (see Orlowsky et al. [2008], for details). [See in Sec. 4.5: **check_regression_parms.cpp**]

Both at step 1 and at the end of step 2b, years or blocks are drawn randomly. This makes any simulation a stochastic realization of the population of possible simulations, given the prescribed regression parameters and the set of heuristic rules. Its range can be estimated by generating large ensembles of simulations.

1.2 Extension to several locations

The approach works pretty much along the same lines for multi-location simulations, except for that it takes place in a parameter space of higher dimensionality. The ultimate goal however, a date-to-date-mapping, is the same. As the simulated series are generated by rearranging observations from the observation period according to this mapping, and the order of the rearrangement is the same at all stations or grid points, the series of simulated fields consist of spatial fields that were observed during the observation period.

In order to limit the dimensionality of the task, in a preparatory step climatologically similar locations are classified using cluster analysis. The stations are characterised by selected climate statistics from the observation period: e. g., mean temperature and precipitation, standard deviation of temperature and precipitation and the difference between the mean of the second half and the mean of the first half of the observation period. They thus roughly describe climatological level, variability and temporal development at the stations.

For each of the clusters, a representative station is identified by looking for the station most similar to the cluster centre of mass. Linear regression parameters for the characteristic variable are prescribed at each of the representative stations, thereby allowing for the simulation of spatially differentiated developments.

Suggestions of sets of representative stations are made by the program **pre_process.exe**, which determines useful sets of 1 to 10 out of all available stations [see in Section 4.9: **pre_processing/pre_processing.cpp**]. It uses a simple kmeans-clustering algorithm with random initialisation from

the SiMath-library (see below).

After all simulations are run, their most important characteristics are written to a file [see Section 4.7].

2 Installation

The program is implemented in C++. It makes use of parts of the open source C++ library SiMath 1.0[1], which were partly edited and corrected by Boris Orlowsky. They are integrated directly into the source code, so for this no extra library compilation and linking is necessary. However, the program requires a part of the *boost*-library[2], namely the *filesystem*-part.

2.1 Linux/Unix

A Makefile for Unix and Linux environments, which allows for an easy build, is provided. It has been tested in SuSE Linux and Debian Linux environments with the gcc (4.1.0 and 4.1.2) compiler. Note that in this Makefile, the boost-library is supposed to be located in the directory, where the program shall be compiled. If the boost-library is installed elsewhere, adapt the boost-related variables in the Makefile.

Once the boost-library is installed (please note, that, although boost is mainly a header-only-library, the *filesystem*-part needed for this application requires some compilation[3]), just place the archive **stars_2.4.tar.gz** into a suitable directory, unpack it by typing

```
$> tar zxf stars_2.4.tar.gz
```

and then, after editing the Makefile compile it by typing

```
$> make all
```

This should give you the **stars_2.4.exe**, the **pre_process.exe** and the **post_process.exe** files. Alternatively, you can compile only one of them by typing

```
$> make star
```

or

```
$> make post_process,
```

or

[1] See http://www.silicos.com/simath.html
[2] See http://www.boost.org/

```
$> make pre_process.

$> make clean
```

removes the object files. If the build is dynamically linking the boost-library (as it happens with the provided Makefile), an

```
$> export LD_LIBRARY_PATH=<path_to_stars2.4/boost/boost_1_34_1/lib:$LD_LIBRARY_PATH
```

or equivalent will be necessary.

3 Usage

As you can see from Section 1, in order to generate regional climate simulations, a small set of representative stations or grid points has to be identified in a preparatory step. Suggestions for these stations or grid points are given by the program **pre_process.exe**, but the user is free to identify the stations or grid points which in his expertise optimally represent the climatological variability of the region of interest.

Additionally to the data, prescribed regression parameters of the selected climate variable must be provided. This can be done in two ways: by prescribing a general trend for the entire region, which is understood as starting from the level of the end of the observation period. It is rescaled for each representative station or grid point according to the evolution of the observation period (see Section 3.2.3 for more details). Alternatively, a file for each representative station or grid point can be provided, which contains the regression parameters (see Section 3.2.2).

The archive **files_for_testing.tar.gz** contains (not necessarily meaningful) example files, which can be used for a test run (under Windows, for example WinZip is able to open this archive).

Unless otherwise stated, files are expected to be located in the working directory, i.e. the directory, from which **stars_2.4.exe** is called and into which its results are written.

3.1 Optional pre-processing

The program **pre_process.exe** can help identifying meaningful sets of representative stations or grid points. It is invoked as follows:

```
$> pre_process.exe basz_path work_path
```

where

[3] See http://www.boost.org/more/getting_started/unix-variants.html

- **basz_path** is the path where the available basis scenarios are stored and
- **work_path** denotes the directory, where the result is to be stored.

This result is stored in **suggested_reference_stations.dat** and contains suggested lists of up to 10 representative stations. They can be inserted into **simparm.dat** (see below in Section 3.2.3).

3.2 Preparation of the input data

3.2.1 Observation data files

The files containing the observation data need to be named according to the following scheme: **?????basz.dat**, where ????? stands for a 5-digit number id of the station or grid point and *basz* stands for "Basis Szenario" (basis scenario). They need to contain an arbitrary header line, and have to consist of columns containing the following elements: day, month, year, meteorological variables...
e. g. the PIK internal format,

day	mon	year	tmax	tmean	tmin	prec	hum	airp	vapor	sunsh	cloud	rad	wind
1	1	1951	-1.5	-3.8	-12.6	1.5	83.0	987.7	3.8	1.5	7.3	297.0	4.3
2	1	1951	-.3	-2.5	-5.1	.0	92.0	983.8	4.9	.0	5.0	208.0	4.3
3	1	1951	1.2	-.2	-5.0	.0	85.0	991.2	5.2	.0	7.7	141.0	2.9
4	1	1951	3.4	.6	-1.0	.0	82.0	1003.6	5.4	2.2	4.0	198.0	5.8
5	1	1951	4.1	2.4	-1.1	4.8	92.0	999.0	6.6	.0	7.0	158.0	5.8
6	1	1951	6.0	4.4	3.0	1.9	99.0	996.7	8.7	.0	8.0	35.0	4.8
7	1	1951	6.9	5.4	1.8	1.1	75.0	989.6	7.0	.3	7.3	48.0	8.3
8	1	1951	6.1	3.9	2.9	.7	85.0	997.9	6.9	.4	8.0	93.0	6.6
9	1	1951	5.6	3.5	2.0	5.2	83.0	995.3	6.7	.0	7.7	133.0	9.2

The selected characteristic climate variable must be located in one of those columns (for the introductory description of the model this variable was mean temperature). NA values should be entered as -999.9.

It is important to know that all kind of variables in any order can be written in the input files. However, some processes in the model (e. g. preprocessing, post-processing, simulation characterisations) demand the meteorological variables in the order shown above, otherwise the results are not reasonable. It is therefore recommended to use the specific order and to add other variables in additional columns. Obviously, some of the processes also can not deal with NA values. Hence, they should be avoided, if a proper simulation characterisation is desired.

3.2.2 Regression parameter files

The regression parameters characterise the development of the selected characteristic climate variable. More specifically, the simulation series are generated such that the annual means of the selected variable feature a regression

line, which corresponds to the prescribed parameters. If not calculated from a general prescribed trend (see Section 1.2 and 3.2.3) the files containing them have to be named **?????.trends** in the root directory and consist of two values each, the starting level of the regression line for the simulation period and the final level. E. g., if you want to simulate a climate of which the temperature starts at 8.7 °C and arrives at 9.9 °C (in terms of the linear trend of mean annual temperature), then the corresponding file would look like

```
8.7  9.9

```

Note that the second number has to be followed by a blank line.

3.2.3 Simulation parameter files

The program will look for a file called **simparm.dat** in the working directory. In this, a number of options for the simulations have to be set, always in the form OPTION=VALUE. The options are:

- **basz_path** (string) – Contains the absolute path of the observations (basis scenario files). Defaults to ".", the current directory.
- **char_var** (integer number) – The column number, in which the characteristic variable is stored. Must range between 4 and the number of columns in the **?????basz.dat** files.
- **blocklength** (integer number) – The length of the blocks in days. For Central Europe, 12 days has proven to be a reasonable length, which is also the default.
- **normalise** (y/n) – Whether or not the used data should be normalised. This is important, if the variability varies much across the studied region. If normalise=y, the data is normalised to have a mean of 0 and an unit standard deviation.
- **output_block** (y/n) – As you can see from Section 1.1 and Figure 2, the input data is organised in blocks of **blocklength** days (sliding blocks). These blocks are classified using a cluster analysis. If you want to use an own clustering algorithm, set **output block**=y. The program then only builds the sliding blocks, outputs them to **block.dat** (which can be used for the external cluster analysis) and exits. Otherwise, the blocks are built and the simulation is run.

- **internal_cluster** (y/n) – If yes, the data is clustered by the program internally, using a kmeans clustering and a random seed. Alternatively (if internal_cluster=n), the user is expected to provide an external clustering in the file **cluster.dat**, which contains a class id for each block (see Section 1). **cluster.dat** therefore contains as many entries as there are rows in **block.dat**. The cluster ids have to range in $0 \ldots (no\_clusters - 1)$.

- **no_clusters** (integer number) – The number of clusters. This has to be provided also if the clustering is done externally (defaults to 50).

- **no_sims** (integer number in 1... many) – The number of simulations to be run. For each simulation, a directory is generated which will keep its results.

- **stats** – A list of ????? station ids (the representative stations or grid points), separated by blanks. The program works with any number of stations, however, convergence to the prescribed regression lines of each station can become very hard to achieve for numbers larger than approximately 6. In that case, tolerances need to be very generous (see below).

- **delta_temp** (floating point number) – General trend of the characteristic climate variable for the entire region, starting from the level of the end of the observation period. This number will be rescaled for each representative station or grid point by assuming constant ratios of the observed trends at these stations for the simulation period.

- **calc trends** (y/n) – Whether regression parameters are to be calculated from the general trend in **delta_temp**. If not, they are read from the regression parameter files, **?????.trends**, one file per representative station or grid point.

- **tolerances** – A list of floating point numbers, giving the tolerated deviations from the prescribed regression line. One tolerance per station, separated by blanks. Units are the same as the unit of the chosen characteristic variable. We recommend values of 10%-20%.

- **tol_jd** (integer number) – The allowed difference of the Julian day when replacing blocks, see Section 1.1. Defaults to 20.

- **no_conca** (integer number) – Number of rearrangements of entire calendar years, the best of which is used as the first approximation. A

reasonable number is 50000. However, for tests one can use a lesser number (for example 5000) to save computation time.

simparm.dat could look like:

```
basz_path=/data/elbe_data
char_var=5
blocklength=12
normalise=n
output_block=n
internal_cluster=n
no_clusters=50
no_sims=3
stats=17007  21005
delta_temp=1.2
calc_trends=y
tolerances=0.2  0.19
tol_jd=20
```

3.2.4 The calendar for the simulation period

The program also looks for a text file containing the calendar dates of the simulation period in the working directory. It has to be named **datum_sim.dat** and contains the dates in the format
day month year
e.g.,

```
1  1  1976
2  1  1976
3  1  1976
4  1  1976
5  1  1976
6  1  1976
7  1  1976
```

The simulation period must not consist of more years than the observation period.

3.3 Run – generating the date-to-date-mapping

If everything is prepared, the simulations can be started by typing

```
$> stars_2.4.exe  number
```

where number is an optional integer number which serves as seed for the random number generator. This is useful if several runs of the program are started simultaneously as it is thereby possible to make sure that every single run starts with a different random seed.

For each simulation, a directory is created, labelled with integer numbers starting from 0. All output for these simulations is stored in the respective directories.

The output of the model is written to the directory of the current simulation. It consists of four files:

- **order_years.dat** – The order of the rearrangement of the yearwise segments from the observation period, which constitutes the first approximation (see Section 1.1). Its elements range from 1 to the number of years of the observation period.

- **final_order_dates.dat** – As the ultimate goal of the simulation consists of the generation of a date-to-date-mapping of calendar dates of the observation period to dates of the simulation period, this mapping is written to file **final_order_dates.dat**, which looks like:

  ```
  11  3  1976 <--  21  2  1961
  12  3  1976 <--  22  2  1961
  13  3  1976 <--  13  3  1974
  14  3  1976 <--  14  3  1974
  15  3  1976 <--  15  3  1974
  16  3  1976 <--  16  3  1974
  17  3  1976 <--  17  3  1974
  18  3  1976 <--  18  3  1974
  19  3  1976 <--  19  3  1974
  20  3  1976 <--  20  3  1974
  21  3  1976 <--  21  3  1974
  22  3  1976 <--  22  3  1974
  23  3  1976 <--  23  3  1974
  24  3  1976 <--  24  3  1974
  25  3  1976 <--  30  3  1972
  26  3  1976 <--  31  3  1972
  27  3  1976 <--   1  4  1972
  28  3  1976 <--   2  4  1972
  ```

  where e. g. the first row means that March 11th 1976 from the simulation period is assigned February 21st 1961 from the observation period and so on.

- **final_order_days.dat** – The output in file **final_order_days.dat** is a single list of the row indices corresponding to the assigned dates. This will in general be the preferred starting point for building and analysing the corresponding simulation. Row indices start at 1.

  For example, if the first few lines in **final_order_days.dat** look like

  ```
  4
  5
  6
  7
  8
  9
  10
  11
  12
  13
  14
  15
  382
  383
  384
  ```

  then the first day of the simulation period is assigned the fourth day of the observation period, the second the fifth, ..., the 13th the 382nd and so on.

- **statistics.dat** – Several statistics concerning the current simulation.

3.4 Characterising the simulations

Once all simulations are run, a file called **sim_characteristics.dat** is written to the working directory. It contains a summarising line for each simulation, indicating mean and trend with respect to annual means of all meteorological variables and the climatological water balance, averaged over all stations and grid points. This information is intended to help choosing simulations for post-processing and further analysis. Note that this statistics can be carried out reasonable only if the variables in the input files are placed like in the example file in Section 3.2.1.

Also given is the rank of each simulation on a scale from wet to dry. This rank is determined as follows: Mean and trend of the climatological water balance are calculated for each representative station and each simulation. E. g., if 4 representative stations are used, an 8-tuple ($8 = 4 * 2$, 2 because of

mean and trend) characterises each simulation. These tuples are written into the rows of a matrix with as many rows as simulations. Each column of the resulting matrix is then sorted individually, such that the first row contains the minimal means and trends and the last row the corresponding maxima. The rank of a given simulation can then be determined by looking for the matrix row, which is most similar to the 8-tuple of the simulation (similarity measured by the Euclidean distance).

At the end of this file, the most prominent quantiles and their simulations are given. They give a reasonable starting point, if one is interested in analysing the whole spectrum of the ensemble without analysing every single simulation (which eventually can be quite a lot of data).

3.5 Post-processing – generating the final simulation files

To generate the final simulation files from the date-to-date-mapping files, the program **post_process.exe** is used. It can simply rearrange the observations according to final **order_days.dat**, or do so plus extra corrections (see below). If the corrections are to be applied, the input data MUST comply with the following file format:

```
header line
day month year v1 v2 v3 v4 v5 v6 v7 v8 v9 v10 v11
```

where

v1	maximum of temperature (°C)
v2	mean of temperature (°C)
v3	minimum of temperature (°C)
v4	precipitation (mm)
v5	relative humidity (%)
v6	air pressure (hPa)
v7	vapour pressure (hPa)
v8	sunshine duration (h)
v9	cloud cover (1/8)
v10	radiation (daily sum, J/cm^2)
v11	mean wind speed (m/s).

If the **?????basz.dat**-files contain more columns, the last column is expected to contain a categorical information about circulation pattern types.

The final outcome of the program is written to files **?????simsz.dat** in the simulation directories, which have the same format as the input file **?????basz.dat**.

Besides, the post-processing creates a file called **post_statistics.dat** if the level (see below) is greater than or equal to 1. It is intended to monitor the correction of the inter-annual variability and/or correction of breaks between observation and simulation time series (which are likely if level ≥ 2). The STARS post-processing routine uses an iterative adaptation algorithm to achieve these corrections. Especially in arid regions this can result into anomalous precipitation values for particular days. This affects mostly humidity and radiation related variables, i. e. precipitation, relative humidity, vapour pressure, sunshine duration, cloudiness, global radiation and wind speed. Therefore for each station the following statistics will be monitored for each affected variable and stored in **post_statistics.dat**:

- number of years with false variance/break adaption
- number of years which needs multiple sweeps for variance/break adaption
- maximum number of sweeps for one year
- maximum amount changed for a single day
- percentage of variance/break adapted days

The syntax of the post-processing is the following:

```
$> post_process.exe basz_path work_directory simulation level
```

where

- **basz path** – the path of the basis scenarios
- **work directory** – the path from where the simulations were invoked. It must contain a file called **stations.tab**, which in its first column has the 5 digit station id and in its second column the latitude. This matters to possible corrections of the sunshine duration (necessary for level 1 and 2, see below).
- **simulation** – The simulation number which is to be post-processed. The number corresponds to the directory – an integer from 0 to $n_{sim}-1$ and can be taken from, e. g., the **sim_characteristics.dat**-file.
- **level** – One of the following

  0 – Just rearrange. This works for all kind of input data according to Sec. 3.2.1.

1 – Rearrange and correct each variable year by year such that the interannual variance of the simulated series is the same as during the observation period. As resampling schemes like STARS sometimes tend to underestimate long term variabilities, this may be a welcome post-tuning. As this means an individual alteration of the meteorological variables and these are subject to differing constraints – e. g., precipitation must not be negative –, the format of the observation files as described above is mandatory for a correct post-processing. If there are more columns than in the list above, the last column is expected to contain categorical information, which is not altered. Any additional columns up to this categorical column are modified without any considerations for meaningful ranges.

2 – Rearrange, correct for interannual variance and shift the simulated series, such that there is no break between the end of the observation series and the begin of the simulated series. This can be useful if a period directly following the observation period is to be simulated. "No break" in this setting means, that the regression lines of the annual means from the observation and from the simulation period connect smoothly. For additional columns in the list above applies the same as for level 1.

3 – As level 2 but with modified treatment of sunshine duration (sunsh) and global radiation (rad). For post-processing level 0 and 1 the absolute values of these variables are rearranged. This can cause a break of the physical limitations due to the obliquity of the ecliptic. For example a day with maximum attainable sunshine duration in summer can be reallocated to a day in autumn with a lower possible sunshine duration. To avoid this effect, post-processing level 3 applies the reordering to the relative values of sunshine duration and global radiation. Therefore, in a first step the absolute observations of each day are transformed to values relative to the multi-year average of that day. Then the rearrangement is applied to the transformed values. Finally the time series is transformed back into absolute values using the multi-year average of the observations.

Example:

```
$> ../post_process.exe /data/elbe_data . 23 2
```

4 The program files

Files are organised in several directories, which correspond to different units in the program. Each directory contains a **.hpp**-header file, which defines one or several classes or just functions, see below.

main.h and **main.cpp** – The files which bundle it all... There are contributions from the following directories, organised along the boxes in Figure 2.

4.1 Input

This directory corresponds to the boxes on the left side of Figure 2.

Input.hpp – The input level consists of three classes:

- **Simparm.cpp** – The Simparm class, where the parameters from **simparm.dat** are stored.
- **Data.cpp** – The Data class, where the observational data from the representative stations and the respective regression line parameters are stored.
- **Dates.cpp** – The Dates class, where calendar dates of simulation and observation period are stored.

read_methods.cpp – Methods for reading matrices and vectors from files.

4.2 Preparation

This directory corresponds to the lower half of the left top box in Figure 2, namely the preparatory organisation of the input data.

Preparation.hpp and **Preparation.cpp** – The Prepared_data class, which contains the data organised as

- **prepare_years.cpp** years and as
- **prepare_blocks.cpp** blocks, including the cluster analysis.

4.3 Firstapprox

This directory corresponds to the second level grey box, the rearrangement of entire years.

Firstapprox.hpp and **Firstapprox.cpp** – The Firstapprox class, which contains the first approximation, i. e. a rearrangement of entire years.
generate_fa.cpp contains the necessary functions for this.

4.4 Mending

This directory corresponds to the yellow boxes of step 2a and 2b in Figure 2.

Mending.hpp and **Mending.cpp** – The Mending class, which contains the mended first approximation, i. e. the first approximation with some selected blocks replaced in order to obtain series which match the regression parameters.
identify.cpp – Functions for identifying blocks which are to be replaced.
replace.cpp – Functions which replace the identified blocks. These functions call code from:

- **used_red.cpp** – A function which selects unused blocks.
- **date_red.cpp** – A function which selects blocks with an appropriate Julian day.
- **bridge_red.cpp** – A function which selects blocks which match the predecessor and/or successor in the simulated series.

4.5 check

This directory contains the functionality for checking whether the Mending object complies with the prescribed regression parameters and, if not, updates the internally prescribed exaggerated regression parameters.

check_regression_parms.cpp – Check the regression parameters achieved so far and update the internally prescribed ones.

4.6 output

output.cpp – Various functions for output.

4.7 Sim_charact

Sim_charact.hpp and **Sim_charact.cpp** – Initialise an object for the characterisation of the simulations.
pre_simulation.cpp – Preparatory functions and data.
post_simulation.cpp – Synopsis of the individual simulation characteristics and output.

4.8 misc

misc.cpp – Various arithmetic functions (linear regression etc.), cluster assignment, building of blocks and type conversions.

4.9 pre_processing

pre_processing.cpp – Program which gives suggestions for meaningful choices for the set of representative stations.

4.10 post_processing

post_processing.cpp – All you need for the post-processing...

4.11 my_SiMath

Parts of the SiMath 1.0-library, edited and corrected in parts by Boris Orlowsky. Provides functionality for matrix and vector operations and cluster analysis.

References

J. Lutz, J. Volkholz, and Gerstengarbe F.-W. Climate projections for southern Africa using complementary methods. *Climate Change Strategies and Management*, 5(2):130–151, 2013.

B. Orlowsky. *Setzkasten Vergangenheit – ein kombinatorischer Ansatz für regionale Klimasimulationen.* PhD thesis, University of Hamburg, Hamburg, Germany, 2007. http://www.sub.uni-hamburg.de/opus/volltexte/2007/3316/.

B. Orlowsky and K. Fraedrich. Upscaling European surface temperatures to North Atlantic circulation-pattern statistics. *International Journal of Climatology*, 29(6):839–849, 2009.

B. Orlowsky, F.-W. Gerstengarbe, and P. C. Werner. A resampling scheme for regional climate simulations and its performance compared to a dynamical RCM. *Theoretical and Applied Climatology*, 92:209–223, 2008.

B. Orlowsky, O. Bothe, K. Fraedrich, F.-W. Gerstengarbe, and X. Zhu. Future climates from bias-bootstrapped weather analogs: An application. *Journal of Climate*, 23(13):3509–3524, 2010.

P. C. Werner and F.-W. Gerstengarbe. Proposal for the development of climate scenarios. *Climate Research*, 8:171–182, 1997.