Documentation for the Evaluation Suite (EvaSuite v1.0)

Christoph Braun, Evgenii Churiulin, Beate Geyer, and Ronny Petrik

August 14, 2025

This work is licensed under a Creative Commons “Attribution 4.0 International” license.



Contents

1 General introduction to the Evaluation Suite **4**

2 Quick start of EvaSuite on Levante **5**

3 Which datasets can be analyzed? **6**
3.1 Requirements on the datasets . . . 6
3.2 The regular rotated grid . . . 6
3.3 The regular lon-lat grid . . . 6
3.4 The Lambert Conic conformal grid . . . 7
3.5 Single locations - regular rotated or lon-lat grid . . . 7
3.6 Time dimension . . . 7
3.7 NetCDF data variables . . . 8

4 How does it work? - Internal workflow of EvaSuite **9**

5 How to obtain and install EvaSuite **10**
5.1 Requirements on the software . . . 10
5.2 Load default version of EvaSuite on Levante . . . 10
5.3 Installing the stand-alone version of EvaSuite . . . 10
5.3.1 Obtain program code . . . 10
5.3.2 Installation on Levante via pip . . . 11
5.3.3 Update EvaSuite . . . 11
5.3.4 Uninstall EvaSuite . . . 11

6 How to configure EvaSuite **12**
6.1 Controlling the evaluation - main config file . . . 12
6.1.1 General configuration . . . 12
6.1.2 Data mining . . . 15
6.1.3 Professional settings for EvaSuite and netcdf files . . . 19
6.2 Settings for the datasets - model and proof data files . . . 19
6.3 Controlling the visualisation . . . 20
6.3.1 Launch GUI configuration forms . . . 21

7 How to run EvaSuite **22**
7.1 Running the stand-alone version of EvaSuite . . . 22
7.1.1 Configure and run EvaSuite via config-files and sbatch . . . 22
7.1.2 Automation: Run EvaSuite via a wrapper script . . . 23
7.1.3 Examples . . . 23

8 Development **24**
8.1 Technical documentation . . . 24
8.2 Pipelines and tests . . . 24
8.3 Debugging mode . . . 24

9 Support **25**
9.1 Gitlab project 25
9.2 Contact 25

10 Supplementary tools **26**
10.1 Adding pre-defined and custom subregion shapes 26
10.1.1 Command-line based geopackage exploration and modification 26
10.1.2 Jupyter notebook-based geopackage exploration and modification 27
10.2 PlotSmart 27
10.3 Create Score points of evidence plots (ScoPi plots) 27

A Code examples for the general configuration file **29**
A.1 General configuration, dictionary datamining, subdictionary target_EvalFrequency . . 29
A.2 General configuration, dictionary datamining, subdictionary subregions_BoxCorner_files 29

B Tables - overview about possible options **30**
B.1 General configuration, dictionary datamining, supported resample frequencies 30
B.2 General configuration, dictionary datamining, supported resample methods 30
B.3 General configuration, dictionary datamining, supported temporal aggregation methods 31
B.4 General configuration, dataset settings, Correction of Parameter 31

C Overview of available subregions in the geopackage file **32**

D Jupyter notebook to visualize and add custom subregions **42**

E Available metrics in EvaSuite **46**

Chapter 1

General introduction to the Evaluation Suite

The Evaluation Suite is a software tool developed for the evaluation of climate model simulations by comparing model data against gridded reference datasets, which can be either observation or model-based. The recent version of Evaluation Suite is v1.0. Within this document we refer to the Evaluation Suite as EvaSuite.

EvaSuite enables the application of various metrics over user-defined temporal selections, providing flexibility for diverse research applications. EvaSuite can operate as a fully automated component, for instance, as part of the Starter Package for ICON-CLM Experiments (SPICE), or as a stand-alone application.

In this documentation you will find guidance, how to quickly get started with EvaSuite in chapter 2. We describe, which datasets can be analyzed using EvaSuite in chapter 3 and the internal workflow in chapter 4. In the subsequent chapters you will find how to obtain and install (chapter 5), configure (chapter 6), and run EvaSuite (chapter 7). Chapter 7 also points you to hands-on examples to familiarize with EvaSuite. Furthermore, we provide information on development (chapter 8) and support (chapter 9). Here, you can also find contact information to the EvaSuite team. Finally, we provide supplementary tools related to using and defining subregions in chapter 10.

In the appendices you will find further helpful information regarding configurations (appendices A and B), subregions (appendices C and D), as well as the metrics included in EvaSuite (appendix E).

Chapter 2

Quick start of EvaSuite on Levante

In this chapter we describe how to quickly get started using EvaSuite on the computing cluster Levante hosted by the German Scientific Computing Center (DKRZ) in Hamburg.

First, obtain the EvaSuite code, which includes example evaluations (see section 5.3.1). Second, obtain the example data as described in section 7.1.3. Third, load the module EvaSuite as described in section 5.2. Fourth, adjust one of the example evaluations and run it (section 7.1.3). The provided examples allow you to familiarize yourself with running EvaSuite in different configurations and via different pathways.

Chapter 3

Which datasets can be analyzed?

3.1 Requirements on the datasets

The data format which is supported by EvaSuite is 'netCDF'. You can explore examples of files which can be analyzed with EvaSuite by looking at the provided example data (see section 7.1.3).

Due to different operations (like spatial interpolation), it is needed to use EvaSuite only with datasets, which fulfill minimum requirements:

- The horizontal grid should be a regular lon-lat grid, a regular rotated grid, a single geographical location (as in the case of measurement data), a Gaussian grid, or an irregular grid containing lon-lat information for each grid point.
- Each variable in a dataset should possess a variable name without special characters except '_' and '-' and '.'.
- Each variable in a dataset should hold the attributes 'long_name', 'units' and, for the sake of perfectness 'cell_methods'.
- The time dimension should be always contained. The time bounds are very important to make the program understand the 'cell_methods'. It is not allowed to leave out the time coordinate and to code the temporal information in the filename.
- As in EvaSuite all data files related to a specific dataset have to be located in such a way, that a regular expression can be used to catch all of them, i.e. for each dataset only a specifc regular expression is searched for data for the evaluation.

3.2 The regular rotated grid

This grid is often used by regional models (COSMO, REMO, and ICON as output option). All files relevant for evaluation have to fulfill the following requirements on metadata

- The variables 'rlon' and 'rlat' should be contained in each file. Other names should be replaced by the standard names using h nco's. 'rlon' and 'rlat' have to be only one-dimensional.
- Each variable should contain the attribute 'grid_mapping' having the value 'rotated_pole'. Otherwise interpolations cannot be treated in a correct way.
- It is good practice for netcdf-conventions to also put lon's and lat's in a file and to deliver a coordinates attribute for each data variable. If this is not the case, the program could fail w.r.t. horizontal interpolations.

3.3 The regular lon-lat grid

This grid is ofen used by regional models or global models. All files relevant for evaluation have to fulfill the following requirements on metadata

- The coordinate variables 'lon' and 'lat' should be contained in each file. Other names should be replaced by the standard names using nco's. 'lon' and 'lat' have to be only one-dimensional.
- 'grid_mapping' or 'coordinate' attributes are superfluous because the grid is already meant to be located on the sphere with geographical locations.

3.4 The Lambert Conic conformal grid

This grid is sometimes used by regional models like WRF or by providers of measurement data. Starting with version Evaluation Suite Version 0.23.0 , this grid type is allowed but only centre grid points can be examined. The support for the grid is realized by setting it to a curvilinear grid, i.e. the cdo's are only evaluating the longitudes and latitudes contained in the file. All files relevant for evaluation have to fulfill the following requirements on metadata

- The coordinate variables 'lon' and 'lat' should be contained in each file. Other names should be replaced by the standard names using nco's.
- With respect to each data variable, 'grid_mapping' attribute is set to 'lambert_conformal_conic', and the 'coordinate' attribute is set to 'lon lat'.

In addition, a variable 'lambert_conformal_conic' is contained with the following attributes having exemplary values:

```
int lambert_conformal_conic ;
  lambert_conformal_conic:grid_mapping_name = "lambert_conformal_conic" ;
  lambert_conformal_conic:standard_parallel = 46L, 49L ;
  lambert_conformal_conic:latitude_of_projection_origin = 47.5 ;
  lambert_conformal_conic:longitude_of_central_meridian = 13.3333333333333
     ;
  lambert_conformal_conic:false_easting = 400000L ;
  lambert_conformal_conic:false_northing = 400000L ;
  lambert_conformal_conic:earth_radius = 6367470.f ;
  lambert_conformal_conic:radius = 6367470.f ;
```

To use the LCC2 grid, one has to tell the EvaSuite that during data mining with cdo's the grid-describing variable above-mentioned should be repaired, i.e. the string 'lambert_conformal_conic' should be added to the list 'RepairNetcdf_variables' given in the file 'config_SUITEgeneral.py'. Otherwise, the variable is thrown away by cdo's.

3.5 Single locations - regular rotated or lon-lat grid

This grid is often used for point measurements. All files relevant for evaluation have to fulfill one of the following requirements on metadata

1. The coordinate variables 'lon' and 'lat' (singleton dimension) are contained in each file.
2. The coordinate variables 'rlon' and 'rlat' are contained in each file. A 'grid_mapping' and 'coordinates' attribute have to be contained in each file. Moreover, the rotated pole has to be given as netCDF variable.

3.6 Time dimension

The time variable has to have at least a dimension of one. It is not allowed to have no time variable / dimension in a netCDF file. The time units attribute has to be contained as well as the calendar attribute. If the latter is not given then the calendar 'proleptic_gregorian' is taken. Time bounds are not required but it is good practice to code them for each data variable in addition to the time coordinate. Otherwise the 'cell_methods' are not interpreted correctly.

Up to version Evaluation Suite v 1.0, only a relative time coordinate is allowed. The format 'days as ...' leads to conflicts when interpreting the date stamps. One needs to use cdo's to set a relative time axis prior to the evaluation run.

3.7 NetCDF data variables

The best option is that data follow the NetCDF Climate and Forecast (CF) Metadata Conventions. It is a must that the data variables (excluding the coordinate variables) have the attribute standard_name. In addition, the attributes 'long_name' and 'units' should be contained in the netCDF. Otherwise, the program provides the results of evaluations (nc-files and figures) with some strange units and variable descriptions. Moreover, the evaluation can crash if 'units' are missing.

Chapter 4

How does it work? - Internal workflow of EvaSuite

Basically, EvaSuite provides two evaluation modes: CompareRAW and EvalRAW. In the mode CompareRAW the data of all datasets is compared to each other in terms of absolute values, without calculation of any further metrics. In the mode EvalRAW one or multiple datasets are evaluated against a reference dataset via a list of metrics specified by the user for the comparison (see further details in the table in section 6.1).

The general workflow of the Evaluation Suite is as follows (this is described here for the mode EvalRAW):

1. Assemble all necessary input datasets for the evaluation of the indicated variables and periods (corresponds to CDO selvar and seltime)
2. Remap data to target grid and region (corresponds to CDO remap...).
3. Sample/Select time steps that will provide the basis for the evaluation (corresponds to e.g. CDO monmean for resampling time series to monthly mean values)
4. Calculation of the specified evaluation statistics (this can involve spatial or further temporal averaging). This step can be repeated without going through the 3 prior steps by using the option `eval_StartingPoint = 'StartEvaCompute'`.

The data produced during the above steps can be found in the output directory DataMining, which contains the following 4 subdirectories:

1. WP01_Preperation
2. WP02_UniqueGrid
3. WP03_EvalProcess
4. WP04_EvalFinal

If the evaluation mode CompareRAW is used, the last step of calculating evaluation statistics is skipped. Therefore, the resulting data of this comparison is located in the subdirectory WP03_EvalProcess and the subdirectory WP04_EvalFinal is empty.

For both evaluation modes the final step includes the optional creation of figures based on the calculated results. This mode can be turned off or be run separately (see the namelist switches for the visualization in section 6.1).

Chapter 5

How to obtain and install EvaSuite

We here describe different ways to make EvaSuite available on the machine you are working on.

5.1 Requirements on the software

As of Evaluation Suite v 1.0, the following packages or software must be installed:

- The climate data operators 'cdo',
- The ncar netCDF Operator 'nco'.
- The netcdf-Fortran libraries ncdump + nccopy.
- The required Python environment will be installed automatically using the instructions described below.

If needed a description how to install the required software packages and EvaSuite itself can be found in section 5.3.

5.2 Load default version of EvaSuite on Levante

In case you are working on the computing cluster Levante at the Deutsche Klimarechenzentrum, Hamburg, the easiest way to make EvaSuite available is to simply load the module EvaSuite:
`module load evasuite/` You can check the loaded version of EvaSuite via
`module list`.
In case you want to use this version of EvaSuite you do not need to take care of any other steps in order to run EvaSuite.

5.3 Installing the stand-alone version of EvaSuite

There are several options to install EvaSuite. In the following sections we describe how to install EvaSuite on the high performance computing cluster Levante using pip. Although the description is cluster specific, it may provide helpful information for the installation on other clusters, too. More detailed and other options how to install EvaSuite, such as installation via conda or the installation without internet access, are provided in DOCS/Installation.md.

5.3.1 Obtain program code

If you have access to the EvaSuite gitlab repository (`https://gitlab.dkrz.de/clm-community/evasuite/HZG_Evaluation_Suite.git`) you can obtain the available versions of EvaSuite by cloning from there. If you wish to gain access to the gitlab repository, please contact the EvaSuite development team.
With ssh access to the DKRZ gitlab:
`git clone --recurse-submodules`

```
git@gitlab.dkrz.de:clm-community/evasuite/HZG_Evaluation_Suite.git EvaSuite_v1.0
```

Without ssh access to the DKRZ gitlab, using https:
```
git clone --recurse-submodules
https://gitlab.dkrz.de/clm-community/evasuite/HZG_Evaluation_Suite.git EvaSuite_v1.0
```

On Levante, we recommend that you clone the repository into your home directory or a subdirectory of it. In case you obtained the code of EvaSuite via zenodo, we analogously recommend to untar in your home directory or a subdirectory of it. (Exceptions from this are mentioned in section 7.1.3.)

5.3.2 Installation on Levante via pip

1. Generate a Python virtual environment with the name evasuite) or any other name you prefer to use): `python -m venv $HOME/path_to_environments/evasuite`

2. Activate the virtual environment: `source $HOME/path_to_environments/evasuite/bin/activate`

3. Install the required libraries: `cd EvaSuite_v1.0.0`
and
`pip install -r requirements.txt`

4. Install EvaSuite itself: `pip install .`

You can test whether the installation was successful by running the command `evasuite --help`.

Optionally: Add the path, where EvaSuite was installed to, to your $PATH variable by adding the line `PATH=$PATH:$HOME/.local/bin` to the file $HOME/.profile .

Note: In order to later execute EvaSuite, you need to activate the environment created above.

5.3.3 Update EvaSuite

If you want to make use of the new features of EvaSuite provided via gitlab, use the following steps.

1. Navigate to the directory that contains the source code of EvaSuite.
   `cd EvaSuite_v1.0.0`
2. Download a new version via Git
   `git pull`
3. Make sure that you activate the virtual environment created during the installation of EvaSuite
   `source $HOME/path_to_environments/evasuite/bin/activate`
4. Remove the old version
   `pip uninstall hereon_evasuite`
5. Install the new version:
   `pip install .`

5.3.4 Uninstall EvaSuite

Make sure to activate the environment, within which you installed EvaSuite. Afterwards, you can uninstall EvaSuite via pip: `pip uninstall hereon_evasuite`. If required, confirmation with Y should yield ‘Successfully uninstalled hereon-evasuite‘.

Chapter 6

How to configure EvaSuite

The behavior of EvaSuite is controlled by configuration files. The general configuration file (section 6.1) contains parameters for the general configuration of EvaSuite (section 6.1.1) and the data mining (section 6.1.2). The general configuration file also contains a section with advanced user settings to specify all evaluation quantities and evaluation modes (see section 6.1.3). The interface to the data to be analyzed is provided by two configuration files for model and reference data (section 6.2). Furthermore, there are optional configuration files to specify the visualization settings (see section 6.3).

EvaSuite reads these files as dictionaries and extracts all the information needed to make an evaluation run.

6.1 Controlling the evaluation - main config file

The general configuration options of Evaluation Suite are explained in the following table.

6.1.1 General configuration

Option (reject hyphen)	Type	Description + Example
eval_StartingPoint	String	Define where to start with the evaluation. Possible values are 1. 'cleanall': Everything located in the directory 'eval_workdir' is cleaned and the Evaluation run starts from scratch 2. 'StartEvaCompute': The computation starts from all datasets being interpolated to the same grid. This is useful for changing something like data aggregation frequency without doing the full evaluation run. `eval_StartingPoint = 'StartEvaCompute'`
eval_workdir	String	The directory where the evaluation process takes place including all operations on file and intermediate results are stored. The directory contains a subdir 'Datamining' and a subdir 'Visualizations'. Also, the configuration files are written in the workdir. `eval_workdir = "/scratch/m/m221054/HZG_WORK/Suite-Out/"`
eval_backup	Boolean	Configurations, final results and visualizations are backuped after evaluation run into a hashtagged directory named as the 'eval_workdir' but adding a description (datasets, evaluation parameters, evaluation period, temporal aggregation) and a hashtag. The hashtag is computed from the content of the three configuration files. `eval_backup = True`
eval_backup_dir	String	The 'eval_backup' switch activates the backup of results into a hashtagged directory but it can be overwritten by setting explicitly here a directory name.

data2proof_acronyms	List of strings	A list of datasets which are proved by the evaluation tool (often model data). One has to use acronyms without special characters except '_' and '-'. The datasets given here have to be specified in more detail by the settings in 'data2proof_settings', i.e. all acronyms given here have to appear also in 'data2proof_settings'. In the case of skill score related evaluation measures the first dataset in the list is taken as the 'skill-score-reference' where all other datasets are then the 'skill-score' test data. `data2proof_acronyms = ["CEL202","CEL204"]`
evaldata_acronyms	List of strings	A list of datasets which are the reference data for the evaluation tool (often observation, but also a reference model is possible). One has to use acronyms without special characters except '_' and '-'. The datasets given here have to be specified in more detail by the settings in 'data2proof_settings', i.e. all acronyms given here have to appear also in 'data2proof_settings'. `evaldata_acronyms = ["TRY_v001"]`
evaldata_combine_method	String	Specify the treatment of multiple reference data sets ('evaldata_acronyms'). The user can choose to combine the reference data sets during the evaluation process by specifying one of the following options 1. 'mean' : The reference data sets are averaged to a new datasetroot mean square error in temporal space. 2. 'minmax' : The range of the reference datasets is determined and as a result two reference data sets are calculated (a minimum and a maximum data. However, as of Evaluation Suite v 1.0 this feature does not work with the evaluation mode 'EvalRAW'. 3. '-' : The reference datasets are taken as they are. Thus, for multiple reference data sets the evaluation is done for each of them. `evaldata_combine_method = "-"`
eval_Mode	String	Define the type of evaluation and the related visualization. As of Evaluation Suite v 1.0 two types are implemented, i.e. the user can choose between 1. 'CompareRAW': All the datasets are compared to each other w.r.t. the raw data. 2. 'EvalRAW' : The datasets specified in 'data2proof_acronyms' are evaluated against the datasets specified in 'evaldata_acronyms'. The user is free in specifying the list of metric quantities (see 'eval_metrics') `eval_Mode= "EvalRAW"`
eval_RecompVar_type	String	Allow for the complex manipulation of data prior evaluation or comparison. This setting is predefined with 'False', i.e. the user does not has to define it in the configuration. As of Evaluation Suite v 1.0 only one setting is possible 1. 'grad_z': The gradient in the vertical direction is calculated but only in case of levels with a height coordinate. 2. 'setvalidrange;min;max' : All values in the datasets outside the range 'min' and 'max' are set to nan (e. g. radiation at night not considered) 3. 'setmaskedrange;min;max': All values in the datasets within the range 'min' and 'max' are set 1 otherwise to nan. 4. False: No manipulation of the raw data is done. `eval_RecompVar_type = 'grad_z'`
eval_parameters	List of one string	Define which parameters to evaluate. As of Evaluation Suite v 1.0 only one parameter is allowed but the format has to be a list. The user is free in defining a name for a parameter, but for each dataset the user has to specify which variable name is associated with the parameter. `eval_parameters = ["temperature"]`

eval_metrics	List of strings	The metric that should be considered for evaluation. If the 'eval_Mode' is set to 'CompareRAW', this settings in nullified. If the 'eval_Mode' is set to 'EvalRAW', the user is free to choose a list of evaluation measures. At the moment, the following metrics are supported 1. 'BIAS_temporal' : the mean error (bias) for each location over time, 2. 'BIASnorm_temporal' : the mean error (bias) for each location over time, but normalized by the temporal mean, 3. 'BIAS_spatial' : the mean error (bias) for each time over the 2D field, but normalized by the spatial mean, 4. 'MAE_temporal' : the mean absolute error for each location over time, 5. 'MAE_spatial' : the mean absolute error for each time over the 2D field, 6. 'RMSE_temporal' : the root mean square error for each location over time, 7. 'RMSE_spatial' : the root mean square error for each time over the 2D field, 8. 'MAD_temporal' : the mean absolute deviation for each location over time, $MAE(x,y) = MEDIAN(MOD(x,y,t) - MEDIAN(OBS(x,y,t)))$, 9. 'MAD_spatial' : the mean absolute deviation for each time over the 2D field, 10. 'LinCorr_temporal' : the linear correlation for each location over time, 11. 'LinCorr_spatial' : the Pearson correlation coefficient over the 2D field and for each time, 12. 'KlingGupta_temporal': the Kling-Gupta efficiency for each location over time 13. 'quantile_bias_temporal' : the bias of quantiles for each location, computed from the PDF of all values in time. 14. 'quantile_mae_temporal' : as for 'quantile_bias_temporal', but the mean absolute error of quantiles for each location, computed from the PDF of all values in time. 15. 'quantile_iqd_temporal' : as for 'quantile_bias_temporal', but the inverse quadratic distance of quantiles for each location, computed from the PDF of all values in time. 16. 'pdf_pss_temporal' : the Person-Skill-score for each location, computed from temporal distribution. 17. 'AMSESS_temporal' : the advanced mean squared error skill score for each location over time, which is a normalization of the original mean squared error skill score. 18. 'AMSESSsigm_temporal' : the significance for RMSE-based skill score (AMSESS) of model A to be better than those of model B (significance level of 0.10). 19. 'AMSESSsigs_temporal' : the significance for RMSE-based skill score (AMSESS) of model A to be better than those of model B (significance level of 0.05). 20. 'BIASsigBSm_temporal' : the significance for the BIAS of model A to be smaller than those of model B (significance level of 0.10). 21. 'BIASsigBSs_temporal' : the significance for the BIAS of model A to be smaller than those of model B (significance level of 0.05). `eval_metrics = ["RMSE_temporal","AME_temporal"]`

eval_standalone_visual	Boolean	Run only the visualization module. Be aware that if you run EvaSuite in this mode the content of the working directories WP01_Preperation and WP02_UniqueGrid will be deleted (see chapter 4). `eval_standalone_visual = true`

eval_path	List	Indicating the full paths to the results calculated in a previous run of EvaSuite. Required if standalone visualization shall be performed.
		`eval_path = [/scratch/.../CCLM-MIROC6_Ex_vs_Eobs_v25_Ex-T,`
		`/scratch/.../CCLM-ECE3-Veg_Ex_vs_Eobs_v25_Ex-T]`

eval_create_1D_plots	Boolean	Turn on generation of 1d visualizations. `eval_create_1D_plots = true`
eval_create_2D_plots	Boolean	Turn on generation of 2d visualizations, i.e. maps. `eval_create_2D_plots = true`
eval_create_Taylor_plots	Boolean	Turn on the generation of Taylor plots. In order to use this feature, the Taylor GUI must be used to generate a settings YAML for taylor plot generation. See section 6.3 for futher details. `eval_create_Taylor_plots = true`
eval_cmor_table	String	Path to and name of a CMOR table. This table will be created automatically. It is used in the visualization module to map all original variable names (as defined in the netcdf input files) to the common variable name defined via paramdict (see section 6.2). `eval_cmor_table = /work/.../evasuite_testdata/cmor_table.csv`
eval_plot_type	String	Specify the visualization for the 1d plots. Possible options are line, scatter, bar, hist, qqplot, qqplot_percentiles, qqplot_scatter, and percentiles_diff. `eval_plot_type = line`
eval_proj_type	String	Specify the visualization for the 2d plots. Possible options are global_PlateCarree, copat2_LambertConformal, and RotatedPole. `eval_proj_type = copat2_LambertConformal`

6.1.2 Data mining

The option 'datamining' is a huge dictionary controlling the whole data processing. The dictionary contains many settings explained below in the table. It is most important to mention that the settings can be subdivided into two groups.

One group contains all the settings up to 'interpol_method' and they are fixed for all evaluation runs using the option `eval_StartingPoint = 'StartEvaCompute'`. In other words, if the user wants to evaluate another period of interest or wants to redefine the reference spatial grid, he has to rerun the full EvaSuite using `eval_StartingPoint = 'cleanall'`.

The second group contains all settings starting at 'target_EvalFrequency' and they can be changed for rerunning the evaluation using the option `eval_StartingPoint = 'StartEvaCompute'`. Thus, many evaluations can be performed without always treating a huge amount of data processing steps.

Dictionary key (reject hyphen)	Type	Description + Example
targettime_fullspan	List of two strings	Define the time period to examine. Specifying a list of two comma-separated strings containing the beginning and the end in format "YYYY-MM-DD HH:MM:SS". If the user wants only a specific date, he has to give '[begin,end]'. `["2007-01-01 00:00:00", "2007-12-31 23:00:00"]`

targettime_season	List with one String	Define which period of the year to examine, i.e. cut out from 'targettime_fullspan' only a specific time of the year. The following settings are allowed 1. 'complete' : All data are taken between the two dates given by 'targettime_fullspan'. 2. 'Month-XX': A specific month is cut out, where 'XX' has to be replaced by two two-digit month number. 3. 'winter', 'autumn', 'summer', 'spring' : A specific season is cutted out (this option is buggy and should not be used `"targettime_season": ["complete"]`
targetgrid_horizontal	String	Indicates the reference horizontal spatial grid on which the evaluation is based. This value has to be consistent with one of the 'data2proof_acronyms' and 'evaldata_acronyms'. All datasets given in 'data2proof_acronyms' and 'evaldata_acronyms' are interpolated to that grid, which is contained by the dataset given here. The interpolation method is specified with the option 'interpol_method' of the dictionary 'datamining'. The program recognizes automatically equal grids, i.e. if the data are already on the reference grid, no interpolation is done. If the user gives a '-' then nothing is done. `"targetgrid_horizontal": "CEL202"`
addDS_on_targrid_hori	List of strings	Indicates which datasets given in 'data2proof_acronyms' and 'evaldata_acronyms' are on the same grid as the reference horizontal spatial grid. This value has to be consistent with one of the 'data2proof_acronyms' and 'evaldata_acronyms'. This option allows to control which datasets are finally interpolated to the reference horizontal grid. It is required for those cases, where a lot of datasets are on the same grid as the reference grid. There is no automatic detection to find out about equal horizontal grids in the various datasets. In case that no dataset has the same grid as the dataset with the reference grid, set to 'None'. `"addDS_on_targrid_hori": ["CEL203","CEL204","CEL205"]`
targetgrid_vertical	String	The same as for the option 'targetgrid_horizontal', but now for a reference vertical grid. Only height levels are supported in EvaSuite. The type of vertical levels has to be given for each dataset using the switch 'levtype' (see dataset dictionaries). The interpolation to height levels is done linearly. `"targetgrid_vertical": "-"`
interpol_method	String	The method used for horizontal interpolation, but as of Evaluation Suite v 1.0 stuck to 'linear'. `"interpol_method": "bilinear"`
targetdom_horizontal	Dictionary	Cut out only a part of the whole domain covered by the data set with the horizontal reference grid (given in 'targetgrid_horizontal'). Thus, the settings are related to the horizontal reference grid and only the related data set is cut according to the user-setting and prior interpolation. The setting requires the key 'False' or a dictionary with items corresponding to the dimension to cut along and a list of two values indicating the start and end grid index. For instance, in the case of two horizontal dimensions the first item can be 'x-dimension:[10,20]'. The switch is predefined with 'False', i.e. the user does not have to define it in the configuration. `"targetdom_horizontal": 'rlon': [100,300], 'rlat': [50,250]`

targetdom_vertical	Dictionary	The same as for the option 'targetdom_horizontal' but with respect to the vertical coordinates. Thus, some vertical slices are cut out of the dataset holding the vertical reference grid (given in 'targetgrid_vertical'). The setting requires the key 'False' or a dictionary with the item corresponding to the level to cut along and a list of two values indicating the start and end grid index. The switch is predefined with 'False', i.e. the user does not has to define it in the configuration. `"targetdom_vertical": 'height': [2,4]`
target_EvalFrequency	Dictionary	Define the temporal resolution on which the evaluation measures or the comparison is based on, i.e. the raw data are recomputed to the frequency given here and afterward the evaluation starts. There are 3 possible options for the dictionary: 1. 'method' : '-': Nothing is done with the raw data. 2. 'method' : 'reselect','targetgrid': gridtime, 'tolerance': tol: The date stamps in all raw data are reselected consistent to the date stamps in the dataset specified by 'gridtime'. The value 'gridtime' has to be consistent with one of the acronyms given in 'data2proof_acronyms' and 'evaldata_acronyms'. The reselection process searches with the nearest neighbour method (no interpolation is done and the final time stamps in the netcdf's are the same as in gridtime). The tolerance for the reselection can be specified using option tol (e.g. '1H' means one hour, '30M' means thirty minutes). This option is usually utilized in case of measurement data with data gaps in time. 3. 'method' : 'resample','freq' : freqv, 'how' : howv: Define for each data set the new temporal frequency and the method how to get there. Details about the supported keywords for 'freqv' and 'howv' can be found in the tables here and here An example for the three possibilites is given here.
target_RollingMean	Integer	Defines the number of consecutive time steps over which the reference data and the model data are averaged in a temporal window shifted through the time dimension. The number of time steps is related to the setting in 'target EvalFrequency'. If the frequency for evaluation is daily values, then a setting of 'target_RollingMean=5' leads to 5 day rolling means for evaluations or comparisons. If no rolling mean should be computed, set to 'None'. `"target_Rollingmean": 5`
aggregate_temporal	String	Indicates how to aggregate the evaluation statistics calculated. Please do not mix it up with the setting for 'target_EvalFrequency'. As an example: You have one dataset with hourly data and two datasets B and C with 10-minute data. Using the option 'target_EvalFrequency' to select for all datasets only the time stamps of dataset A, i.e. all data are now hourly data. Now you can use `"aggregate_temporal": "weekly"` to determine the BIAS of hourly data on a weekly aggregation period, i.e. if the raw data contained one year of data, you get a timeseries with 53 values for the BIAS. Possible values for the setting 'aggregate_temporal' can be found here. A special keyword is 'fullperiod' which means to aggregate the statistics over the whole time (in the afore-mentioned case only one BIAS for the whole year). `"aggregate_temporal": "3-daily"`

subregion_acronyms	List of strings / None	Defines the sub-domains of interest or one gives 'False' as a keyword. If this setting is not equal to 'False' then evaluation statistics are prepared for each sub-domain given here in the list. The names given here must be consistent to the header line of the files given in the setting 'subregions_BoxCorner_files'. If the program find a keyword in the sub-domain list also in the header line of a specific file, then the polygon in this file is associated with the subdomain. If not, the subdomain is automatically removed from further evaluation. `"subregions_acronyms": ["NI"]`
subregions_BoxCorner_files	Dictionary	The key 'polyfile' is a string and defines the location of the polygone file (format geopackage) which contains the polygons given in the list 'subregion_acronyms'. The key 'attribregion' is a string and defines the column of the attribute table in the polygon geopackage file which contains the names of the polygons. Thus, the names of polygons has to match with the 'subregions_acronyms' `"subregions_BoxCorner_files": {"polyfile": "polygons.gpkg", "attribregion": "NAME_0"}`
mask_Orography	Dictionary	Allows to analyze regions dependent on their altitude (height above sea level). Key 'method' can be set to 'threshold' or 'ranges'. If 'threshold' is chosen, each of the subregions given in 'subregion_acronyms' is divided further into two subsubregions dependent on the altitude given by the key 'value'. I. e. all points located below the altitude of 'value' are considered as subsubregion1 and all values above are considered as subsubregion2. If the key 'method' is set to 'ranges', each of the subregions is devided further into many subsubregions dependent on the altitudes given by the key 'limits', which is a list of real or integer numbers. I. e. all values in the list define an altitude range for which data points are filtered and analyzed: '[200,1000,2000]' separates between data points located between 0 and 200 m altitude, 200 and 1000 m altitude, 1000 and 2000 m altitude and points located higher than 2000 m altitude. Set this parameter to 'None' or 'False' to deactivate the orographic based masking. `"mask_Orography": {"method": "ranges", "value": False, "limits": [0,200,1000,2000]}`
mask_dataf_Orography	String	Defines the location of the data with orography in order to analyze subsubregions based on orography. `"mask_dataf_Orography": "orography.nc"`
mask_landsea	XXX	Not yet tested, please report to EvaSuite team in case you have experience with this parameter!
mask_dataf_landsea	XXX	Not yet tested, please report to EvaSuite team in case you have experience with this parameter!
operator_time	String	Considered only in case of `eval_Mode= "CompareRAW"`. For simple comparison, this setting defines with which method the temporal aggregation given in 'aggregate_temporal' is done. For an evaluation with measure 'RMSE' it is predefined but in the case of a simple comparison, the temporal aggregation is free of choice. Possible values are 'mean', 'sum', 'max', and 'std' and '-'. `"operator_time": "mean"`
operator_spatial_x	String	Considered only in case of `eval_Mode= "CompareRAW"`. As for 'operator_spatial_t' but for the spatial aggregation along the x-axis. `"operator_spatial_x": "mean"`

operator_spatial_y	String	Considered only in case of `eval_Mode= "CompareRAW"`. As for 'operator_spatial_x' but for the spatial aggregation along the y-axis. Only in case of no aggregation along the x-axis, the user is free of choice. If there is already an operation along the x-axis, both operators are coupled. `"operator_spatial_y": "mean"`
operator_spatial_z	String	Not implemented as of Evaluation Suite v 1.0.

6.1.3 Professional settings for EvaSuite and netcdf files

At the end of the general configuration file there is a section where the names of directories created by the program can be manipulated: 'directory_visualize', 'directory_datamining', 'directory_maskregions', 'directoryDM_WPprepare' and 'directoryDM_WPcompeval' (the last are lists with two entries).

Moreover, the user has to define which variables and attributes have to be preserved during data processing: 'RepairNetcdf_variables' and 'RepairNetcdf_attributes' are lists of strings. The preserved variables and attributes are searched in the raw data and if they get lost due to data processing, an automatic repair process is called prior interpolation, grid masking and visualization. 'RepairNetcdf_targetgrid' is of particular interest because here the variable names are given which contain coordinate information of the target grid.

In addition, 'PossibleDimNames' specifies which coordinate variables have to be associated with the spatial, temporal, and vertical dimensions. Otherwise the program, for instance, does not know how to compute a mean on the spatial field.

6.2 Settings for the datasets - model and proof data files

The two dictionaries 'data2proof_settings' and 'evaldata_settings' contain all information about the raw data of these datasets defined by the acronyms in the settings 'data2proof_acronyms' and 'evaldata_acronyms'. That is all acronyms given have to be specified in the dictionaries 'data2proof_settings' and 'evaldata_settings'. Both dictionaries contain the following settings:

Dictionary key (reject hyphen)	Type	Description + Example
format_data	String	Defines in which format the raw data are given. Only 'netcdf' is supported. `"format_data": "netcdf"`
gridtype	String	Defines the type of grid for the raw data. As of Evaluation Suite v 1.0 possible values are 'gridded' and 'station'. The last option is for the point observations. `"gridtype": "gridded"`
levtype	String	The vertical coordinate type of the raw data. As of Evaluation Suite v 1.0 only 'height is supported. This setting only matters if it comes to a vertical interpolation. In that case complex 3D-height coordinates or pressure levels are not allowed in the recent version. `"levtype": "height"`

paramcorr	Dictionary	Defines for each variable name given in 'paramdict' the possibility to correct the data w.r.t. to units. If the key 'unitcor' is true, one can specify with key 'corr_vals : [oper, number, newunit]' how to manipulate the data using '*', '-', '+' or '/' as a mathematical operator and 'number' as value as well as 'newunit' as new unit. The 'paramcorr' dictionary also allows for time shifts. If the key 'timeshift' is true, one can specify with the key 'timcor_val : [oper,number,unit]' how to shift the time axis using '+' or '-' as operator. 'number' as well as 'unit' define how much hours (unit='h') or minutes (unit='m') the shift should be. Please be aware that the variable name, which is being used for the definition of correction factors, has to be the variable name which is used in the netcdf file, that contains the data. An example for the setting is given here.
paramdict	Dictionary	Defines which eval parameter belongs to which variable name in the netcdf file. Using this setting the evaluation program knows how to associate the 'eval_parameters' with netcdf data. `"paramdict": {"temperature": "T_2M", "precip": "TOT_PREC"}`
storagedir	String	The location where the data are stored. If the data are distributed over many subdirectories, one should only give the root directory here. Everything else is handled by the option 'storagefil' `"storagedir": "/pool/data/CCLM-CRCS/output_CEL/"`
storagefil	String	The name of the files. One can set single files or multiple files using regular expressions not only for files but also subdirectories. As of Evaluation Suite v 1.0 only one evaluation parameter is allowed. This parameter can be distributed over many files (e.g. monthly timeseries). `"storagefil": "CEL*/CEL202.*.2007.nc"`

6.3 Controlling the visualisation

For the default visualization output of Evaluation Suite the user simply needs to set the appropriate switches for 1d and 2d visualization in the general configuration as described in section 6.1.

For custom configuration of the visualization the user can provide additional configuration files for

- 2d visualization via `--map-config=MAP_GUI.yaml`
- 1d visualization via `--plot-config=PLOT_GUI.yaml`
- visualization of Taylor diagrams via `--taylor-config=TAYLOR_GUI.yaml`

.

See section 7.1.1 for further details regarding the usage of configuration files in order to run Evaluation Suite .

For map and plot configurations, you only need to specify settings that differ from the default values. Missing parameters will be auto-filled with defaults. A detailed description of all available parameters for the visualization can be found in "DOCS" in the files

- Maps_user_settings.md
- Plots_user_settings.md
- Taylor_user_settings.md.

For Taylor diagrams, a complete YAML configuration file has to be provided. It is recommended to generate this YAML file using the corresponding GUI form as described in section 6.3.1. Example GUI configuration files are available in "EXAMPLE_CONFIGS".

6.3.1 Launch GUI configuration forms

There are GUIs, which provide assistance to create configuration files for visualizing the data via maps, 1d plots, and Taylor diagrams. Please note that in order to run the GUI, you need to enable a graphical interface while connecting to the cluster, i.e. connect via `ssh -X user@host` for linux based systems. See DOCS/X11_user_guide.md for further information and other systems.

Use these commands to launch the GUI forms for interactive configuration:

- 2d visualization via `evasuite --gui_map_configuration`
- 1d visualization via `evasuite --gui_plt_configuration`
- visualization of Taylor diagrams via `evasuite --gui_taylor_configuration`

We strongly recommend to use the GUI form for the generation of Taylor diagrams.

Chapter 7

How to run EvaSuite

7.1 Running the stand-alone version of EvaSuite

We here describe two options how to run EvaSuite. The first option directly operates on the configuration files of EvaSuite. The second option uses a wrapper script to set the configuration parameters of the EvaSuite and allows to execute multiple evaluations in a loop (e.g., for different variables or metrics). Furthermore, we provide an overview of example evaluations, which showcase how to use these different options. The scripts for the example evaluations are provided along with the EvaSuite code.

To run EvaSuite we generally recommend the following usage of home, scratch, and work directories (given a similar cluster architecture as for Levante) for storing the data produced at different steps while conducting an evaluation:

- home: Put the scripts to execute the evaluation here. Log files should also be written to home.
- scratch: Configure EvaSuite to write all the (intermediate) results here. This is done via the namelist switch eval_workdir (see 6.1.1).
- work: Configure EvaSuite to write the final results here. This is done via the namelist switch eval_backupdir (see 6.1.1).

7.1.1 Configure and run EvaSuite via config-files and sbatch

You can directly configure EvaSuite via the configuration file for general settings. To do so, you can create and fill copies of the file "config_SUITEgeneral_TEMPLATE.yaml", which you find in the subdirectory "configuration" in the EvaSuite code directory that you created (section 5.3.1).

Furthermore, you need to make sure that the model and/or reference data that you want to analyze is contained in the model and reference configuration files you refer to in the general configuration file. To do so, you can copy and extend the model and reference configuration files "model-data_settings_COPAT2_ex.yaml" and "reference_datasets_v2.yaml" in the subdirectory "inputs" in the EvaSuite code directory. We provide examples containing the three filled configuration files in section 7.1.3.

The command to run EvaSuite is (you need to replace the path to and the specific names of the configuration files):

```
evasuite --cfggen=path_to_EvaSuite_code_dir/configuration/your_GENERAL_namelist.yaml
--inprefdat=path_to_EvaSuite_code_dir/inputs/your_Reference_datasets.yaml
--inpproofdat=path_to_EvaSuite_code_dir/inputs/your_Modeldata_settings.yaml .
```

Note: Make sure to activate the Python or conda environment in which you installed EvaSuite (see 5.3).

Important note: The evaluation carried out by EvaSuite can be quite comprehensive. Therefore, we recommend running EvaSuite in an interactive session on a compute node or via submitting a batch script via sbatch as presented in the examples in section 7.1.3.

7.1.2 Automation: Run EvaSuite via a wrapper script

You can also use a wrapper script to run EvaSuite. This provides the option to submit multiple evaluations at the same time. In this case the template configuration file GENERAL_namelist.yaml in the subdirectory "configuration" in the EvaSuite code directory is filled during the execution of the wrapper script. Afterwards, EvaSuite is executed using the resulting configuration files. We recommend to run the wrapper script as a batch script and submit it via sbatch as presented in the examples in the section 7.1.3. If you use and adapt one of the wrapper scripts, the output of multiple variables or metrics will be stored in a common output directory, which you need to specify in the script.

7.1.3 Examples

In this section we list the examples currently available for EvaSuite. Please refer to the readme files in EXAMPLE/EXAMPLE_RUNNERS and the corresponding subdirectories for more detailed information and guidance.

In the subdirectory EXAMPLE/EXAMPLE_RUNNERS you can find 6 examples for running EvaSuite directly via sbatch. Furthermore, there are 2 examples that showcase the execution on an interactive node on Levante. The results of these examples should be identical to the examples submitted via sbatch with analogue names. Finally, there are 2 examples that showcase the execution of EvaSuite using a wrapper script. For these examples we recommend to (also) place a copy of the EvaSuite code directory in your work directory (on an architecture like Levante) and execute EvaSuite from there. This will facilitate the adjustment of the scripts to run the evaluation. As a result all the data will be located on work. You can of course adjust the scripts to incorporate a similar separation of work, scratch and home as described above.

Please note that in order to successfully run the examples you need to adjust the paths provided in the example scripts to your personal directory structure.

In the next section, we describe how to obtain the data to run the example evaluations.

Obtain data for example evaluations

If you are working on Levante you can use the shell script "get_data.sh" in "EXAMPLE/EXAMPLE_DATA" in order to copy the example data to your work, i.e. "TEST" should point to a location on your work:

```
./get_data.sh TEST
```

If you do not have access to the data on Levante, the data will be automatically downloaded from `https://swift.dkrz.de/v1/dkrz_475f8922-045d-43f4-b661-0b053074407e/EvaSuite_test_data/evasuite_demofiles.tar.gz?temp_url_sig=a5116b494b620580c479d67984c61547e6a1476f&temp_url_expires=2027-07-14T10:03:53Z`.

After execution you will find 3 shapefiles with subregion shapes (.gpkg) and 15 netcdf files within TEST. The total size of the data is 20 Gb.

The netcdf files provide model data from the projects NUKLEUS and COPAT2 as well as reanalysis data from Eobs v25 and v28. The model data is based on simulations conducted by Hendrik Feldmann, KIT, Karlsruhe, and Andreas Will, BTU, Cottbus-Senftenberg.

Chapter 8

Development

In case you want to contribute to the development of EvaSuite please contact the EvaSuite team. The joint place for development is the gitlab repository hosted by DKRZ: `https://gitlab.dkrz.de/clm-community/evasuite/HZG_Evaluation_Suite`.

8.1 Technical documentation

The technical documentation is organized as a block scheme. A detailed description of the block scheme can be found in DOCS/BLOCKSHEMES.

The block scheme of the entire EvaSuite code can be found in the readme in DOCS/BLOCKSHEMES/MODULE_DOCS. From here you can navigate through the code. If provided in the code, the docstring of each function can be viewed. The three squares next to each function name indicate the status of the documentation in the code. Please have a look at the readme in DOCS/BLOCKSHEMES for further details and quickly assess, which docstrings are available via the visual indicator (three squares).

8.2 Pipelines and tests

Automated tests via a pipeline are conducted after pushing to git. There are also basic license checks applied.

8.3 Debugging mode

EvaSuite can be executed in debugging mode. For information how to do this please see DOCS/Debug_mode.md.

Chapter 9

Support

9.1 Gitlab project

Please report your issues with EvaSuite in the gitlab project `https://gitlab.dkrz.de/clm-community/evasuite/HZG_Evaluation_Suite` or by contacting the EvaSuite team.

9.2 Contact

You can contact the EvaSuite team via beate.geyer@hereon.de or imktro-climate-modelling@lists.kit.edu.

Chapter 10

Supplementary tools

We here describe some helpful supplementary tools that are provided along with EvaSuite. The tools that require with additional code are located in the subdirectory "external".

10.1 Adding pre-defined and custom subregion shapes

EvaSuite allows for calculation of different statistics in different subregions. The shape of this subregions is treated by the EvaSuite using so-called geopackage files which is an extension of the former known shape files. If you want to execute EvaSuite, you need to specify a geopackage file and the ID of the attribute used to identify the names of the shapes. Both settings need to be configured in the general configuration file.

We provide a geopackage file with an long list of predefined subregions, which can easily be used with EvaSuite. You can download the geopackage file here:
`https://swift.dkrz.de/v1/dkrz_c0725fe8741c474b97f291aac57f268f/clmcom/spice/EvaSuite_allpolygones.v1.gpkg`. The list of the available subregions is given in C.

We here describe two options of checking the content of existing geopackage files and adding further subregions. The first option is command-line based and suitable for adding already existing polygons to the geopackage file. This option is ready to be used after installing EvaSuite. The second option is based on a jupyter notebook, which allows the visualization of subregions. Based on the second option, we also demonstrate how to add custom subregions with simple geometries. The jupyter notebook can be obtained from the EvaSuite git repository as described in the corresponding subsection.

10.1.1 Command-line based geopackage exploration and modification

So, starting with a geopackage-file one has e.g. received with the installation of SPICE, type

`evasuite --showpolygons ${geopackage-file}`

to receive a list of the content. Please investigate the header of the table shown and identify the attribute name for the region names (e.g. 'NAME_0').

To get the whole list of regions stored in the geopackage file type

`evasuite --showpolygons ${geopackage-file} NAME_0` .

This are regions which can be handled by the EvaSuite. If you want to handle more regions, you need to extend the geopackage file. There are two possibilities.

Add further subregions using a geopackage file

This can be done using the following syntax

`evasuite --addpolygondb ${newgeopackage} ${newattrib} ${newregionnames} ${oldgeopackage}${oldattrib} ${result}` ,

where the old geopackage file with regions stored under the attributes ${oldattrib} is extended by the content of the new geopackage file. Regarding the new geopackage file one has to specify the attribute ${newattrib} and the values of the attribute ${newregionnames} (the names of the regions stored in the new geopackage file). The final file is stored in ${result} and the final attribute is the same as the old attribute (${oldattrib}).

Add further subregions from an ascii file

The ascii file needs to be in the so-called wkt-format. This can be done using the following syntax
`evasuite --addpolygonascii ${newwktfile} ${newattrib} ${oldgeopackage} ${oldattrib} ${result}`.

It is the nearly the same procedure as for case A), but the final file is extended by ALL regions stored in the wkt file (with names given by the attribute $newattrib).

10.1.2 Jupyter notebook-based geopackage exploration and modification

The jupyter notebook can be found here:
external/geopackage/Plot_subregions_from_gpkg_file_and_add_subregions.ipynb.

A pdf copy of the jupyter notebook is provided in chapter D. Descriptions are included within the notebook itself.

Please see the DKRZ resources on the usage of jupyterhub and jupyter notebooks here: `https://docs.dkrz.de/doc/software%26services/jupyterhub/index.html`.

10.2 PlotSmart

PlotSmart was developed by Sergei Petrov, Helmholtz-Zentrum Hereon, as an alternative to the built-in visualization of EvaSuite. The PlotSmart package allows you to create tables of figures and easily control the main settings to produce high-quality figures for publications. To use PlotSmart, you will need to work within a Jupyter notebook on the Levante HPC system.

PlotSmart can be found here: external/plotsmart. For further details and a user guide see the documentation there.

10.3 Create Score points of evidence plots (ScoPi plots)

The scripts to create ScoPi plots are located here: external/plot_scopi.

For further information on ScoPi plots see: COSMO Technical Report No. 51 CLM Community WG EVAL, COordinated PArameter Testing project 2 (COPAT2): COSMO-CLM 6.0 clm1 recommended model configuration; `https://www.dwd.de/EN/ourservices/cosmo_technical_reports/pdf_einzelbaende/2023_51.pdf?__blob=publicationFile&v=2`.

Acknowledgements

The authors thank Sergei Petrov for contributing to this documentation and providing the tool PlotSmart, Shweta Singh for providing the basic code for the ScoPi plots, Divyaja Lawand and Eleonora Cusinato for providing example evaluations, and Hendrik Feldmann and Andreas Will for providing model data for the example evaluations.

Appendix A

Code examples for the general configuration file

A.1 General configuration, dictionary datamining, subdictionary target_EvalFrequency

```
"target_EvalFrequency" : {
    "method" : "resample",
    "freq" : "daily",
    "how" : "mean"
}

"target_EvalFrequency" : {
    "method" : "reselect",
    "targetgrid" : "PAULAT"
}

"target_EvalFrequency" : {"-"}
```

A.2 General configuration, dictionary datamining, subdictionary subregions_BoxCorner_files

```
"subregions_BoxCorner_files" : {
    "directory" : "/pf/m/m221054/HZG-WORK/project_OpenFRED/EvaSuite/
       Subregion_BoxCorners/BeateGeyer_Lowlands/",
    "filepattern" : "poly*.dat"
},
```

Appendix B

Tables - overview about possible options

B.1 General configuration, dictionary datamining, supported resample frequencies

keyword	Description
'yearly'	yearly aggregation
'seasonal'	seasonal aggregation (starting in December)
'monthly'	monthly aggregation
'weekly'	weekly aggregation (starting on Mondays)
'daily'	daily aggregation
'hourly'	hourly aggregation

Table B.1: Supported resample frequencies

EvaSuite also allows for aggregation periods that are a multiple of the above periods, e.g., "2-monthly" will perform an aggregation over 2 months.

B.2 General configuration, dictionary datamining, supported resample methods

keyword	Description
'mean'	calculating an aggregated mean (e.g. daily mean)
'sum'	calculating an aggregated sum (Bug 1: NaN values are not considered, i.e. a daily sum is '1' if 23 hourly values are NaN and one value is '1'; Bug 2: sum over NaN-values is 0)
'min'	calculating a minimum during aggregation (e.g. daily minimum)
'max'	calculating a maximum during aggregation (e.g. daily maximum)

Table B.2: Supported resample methods

B.3 General configuration, dictionary datamining, supported temporal aggregation methods

keyword	Description
'fullperiod'	aggregation in total
'yearly'	yearly aggregation
'seasonal'	seasonal aggregation (starting in December)
'monthly'	monthly aggregation
'weekly'	weekly aggregation (starting on Mondays)
'daily'	daily aggregation
'hourly'	hourly aggregation
'monthlyoverperiod'	grouping all data to single months
'weeklyoverperiod'	grouping all data to single business weeks
'dailyoverperiod'	grouping all data to single days of the year
'hourlyoverperiod'	grouping all data to single hours of the day

Table B.3: Supported temporal aggregation methods

B.4 General configuration, dataset settings, Correction of Parameter

```
"paramcorr"   : {"T_2M" :
{ "unitcor"    : True, "corr_vals"   : ["+","273.15","K"],
  "timeshift" : True, "timcor_val" : ["-","6","h"]
}}
```

Appendix C

Overview of available subregions in the geopackage file

We provide a geopackage file with an long list of predefined subregions, which can easily be used with EvaSuite. You can download the geopackage file here: `https://swift.dkrz.de/v1/dkrz_c0725fe8741c474b97f291aac57f268f/clmcom/spice/EvaSuite_allpolygones.v1.gpkg`. The list of the available subregions is given in C.

No	NAME_0	category
1	S_Mediterranean_Sea	Sea
2	S_Black_Sea	Sea
3	S_Baltic_Sea	Sea
4	S_North_Sea	Sea
5	S_Marmara_Sea	Sea
6	S_North_Sea_extended	Sea
7	S_North_Atlantic_Ocean	Sea
8	S_South_Atlantic_Ocean	Sea
9	P_Alps	Prudence box
10	P_BritishIslands	Prudence box
11	P_East-Europe	Prudence box
12	P_France	Prudence box
13	P_IberianPeninsula	Prudence box
14	P_Mediterranean	Prudence box
15	P_Mid-Europe	Prudence box
16	P_Scandinavia	Prudence box
17	P_North-West-Africa	Prudence box
18	Albania	country
19	Austria	country
20	Belgium	country
21	Bulgaria	country
22	Bosnia and Herzegovina	country
23	Belarus	country
24	Switzerland	country
25	Cyprus	country
26	Czech Republic	country
27	Germany	country
28	Denmark	country
29	Spain	country
30	Estonia	country
31	Finland	country
32	France	country
33	United Kingdom	country
34	Georgia	country

35	Greece	country
36	Croatia	country
37	Hungary	country
38	Ireland	country
39	Iceland	country
40	Italy	country
41	Kosovo	country
42	Lithuania	country
43	Luxembourg	country
44	Latvia	country
45	Moldova	country
46	Macedonia	country
47	Montenegro	country
48	Netherlands	country
49	Norway	country
50	Poland	country
51	Portugal	country
52	Romania	country
53	Russia	country
54	Serbia	country
55	Slovakia	country
56	Slovenia	country
57	Sweden	country
58	Turkey	country
59	Ukraine	country
60	AustriaCzech	summarized countries
61	Baltics	summarized countries
62	Benelux	summarized countries
63	HungarySlovakia	summarized countries
64	MiddleBalkan	summarized countries
65	NorthWesternBalkan	summarized countries
66	SouthernBalkan	summarized countries
67	EZG_EU_Rhein	catchment
68	EZG_EU_Rhein_N	catchment
69	EZG_EU_Rhein_S	catchment
70	EZG_G_Donau	catchment German part
71	EZG_G_Elbe	catchment German part
72	EZG_G_Ems	catchment German part
73	EZG_G_Havel-Spree	catchment German part
74	EZG_G_Issel	catchment German part
75	EZG_G_Maas	catchment German part
76	EZG_G_Oder	catchment German part
77	EZG_G_Rhein	catchment German part
78	EZG_G_Schwarze-Elster	catchment German part
79	EZG_G_Spree	catchment German part
80	EZG_G_Weser	catchment German part

No	NAME_0	category
81	Stuttgart, Stadtkreis	Landkreis Deutschland
82	Böblingen	Landkreis Deutschland
83	Esslingen	Landkreis Deutschland
84	Göppingen	Landkreis Deutschland
85	Ludwigsburg	Landkreis Deutschland
86	Rems-Murr-Kreis	Landkreis Deutschland
87	Heilbronn, Stadtkreis	Landkreis Deutschland

88	Heilbronn, Landkreis	Landkreis Deutschland
89	Hohenlohekreis	Landkreis Deutschland
90	Schwäbisch Hall	Landkreis Deutschland
91	Main-Tauber-Kreis	Landkreis Deutschland
92	Heidenheim	Landkreis Deutschland
93	Ostalbkreis	Landkreis Deutschland
94	Baden-Baden, Stadtkreis	Landkreis Deutschland
95	Karlsruhe, Stadtkreis	Landkreis Deutschland
96	Karlsruhe, Landkreis	Landkreis Deutschland
97	Rastatt	Landkreis Deutschland
98	Heidelberg, Stadtkreis	Landkreis Deutschland
99	Mannheim, Stadtkreis	Landkreis Deutschland
100	Neckar-Odenwald-Kreis	Landkreis Deutschland
101	Rhein-Neckar-Kreis	Landkreis Deutschland
102	Pforzheim, Stadtkreis	Landkreis Deutschland
103	Calw	Landkreis Deutschland
104	Enzkreis	Landkreis Deutschland
105	Freudenstadt	Landkreis Deutschland
106	Freiburg im Breisgau, Stadtkreis	Landkreis Deutschland
107	Breisgau-Hochschwarzwald	Landkreis Deutschland
108	Emmendingen	Landkreis Deutschland
109	Ortenaukreis	Landkreis Deutschland
110	Rottweil	Landkreis Deutschland
111	Schwarzwald-Baar-Kreis	Landkreis Deutschland
112	Tuttlingen	Landkreis Deutschland
113	Konstanz	Landkreis Deutschland
114	Lörrach	Landkreis Deutschland
115	Waldshut	Landkreis Deutschland
116	Reutlingen	Landkreis Deutschland
117	Tübingen, Landkreis	Landkreis Deutschland
118	Zollernalbkreis	Landkreis Deutschland
119	Ulm, Stadtkreis	Landkreis Deutschland
120	Alb-Donau-Kreis	Landkreis Deutschland
121	Biberach	Landkreis Deutschland
122	Bodenseekreis	Landkreis Deutschland
123	Ravensburg	Landkreis Deutschland
124	Sigmaringen	Landkreis Deutschland
125	Ingolstadt, Kreisfreie Stadt	Landkreis Deutschland
126	München, Kreisfreie Stadt	Landkreis Deutschland
127	Rosenheim, Kreisfreie Stadt	Landkreis Deutschland
128	Altötting	Landkreis Deutschland
129	Berchtesgadener Land	Landkreis Deutschland
130	Bad Tölz-Wolfratshausen	Landkreis Deutschland
131	Dachau	Landkreis Deutschland
132	Ebersberg	Landkreis Deutschland
133	Eichstätt	Landkreis Deutschland
134	Erding	Landkreis Deutschland
135	Freising	Landkreis Deutschland
136	Fürstenfeldbruck	Landkreis Deutschland
137	Garmisch-Partenkirchen	Landkreis Deutschland
138	Landsberg am Lech	Landkreis Deutschland
139	Miesbach	Landkreis Deutschland
140	Mühldorf a. Inn	Landkreis Deutschland
141	München, Landkreis	Landkreis Deutschland
142	Neuburg-Schrobenhausen	Landkreis Deutschland
143	Pfaffenhofen a.d. Ilm	Landkreis Deutschland

144	Rosenheim, Landkreis	Landkreis Deutschland
145	Starnberg	Landkreis Deutschland
146	Traunstein	Landkreis Deutschland
147	Weilheim-Schongau	Landkreis Deutschland
148	Landshut, Kreisfreie Stadt	Landkreis Deutschland
149	Passau, Kreisfreie Stadt	Landkreis Deutschland
150	Straubing, Kreisfreie Stadt	Landkreis Deutschland
151	Deggendorf	Landkreis Deutschland
152	Freyung-Grafenau	Landkreis Deutschland
153	Kelheim	Landkreis Deutschland
154	Landshut, Landkreis	Landkreis Deutschland
155	Passau, Landkreis	Landkreis Deutschland
156	Regen	Landkreis Deutschland
157	Rottal-Inn	Landkreis Deutschland
158	Straubing-Bogen	Landkreis Deutschland
159	Dingolfing-Landau	Landkreis Deutschland
160	Amberg, Kreisfreie Stadt	Landkreis Deutschland
161	Regensburg, Kreisfreie Stadt	Landkreis Deutschland
162	Weiden i.d. Opf, Kreisfreie Stadt	Landkreis Deutschland
163	Amberg-Sulzbach	Landkreis Deutschland
164	Cham	Landkreis Deutschland
165	Neumarkt i.d. OPf.	Landkreis Deutschland
166	Neustadt a.d. Waldnaab	Landkreis Deutschland
167	Regensburg, Landkreis	Landkreis Deutschland
168	Schwandorf	Landkreis Deutschland
169	Tirschenreuth	Landkreis Deutschland
170	Bamberg, Kreisfreie Stadt	Landkreis Deutschland
171	Bayreuth, Kreisfreie Stadt	Landkreis Deutschland
172	Coburg, Kreisfreie Stadt	Landkreis Deutschland
173	Hof, Kreisfreie Stadt	Landkreis Deutschland
174	Bamberg, Landkreis	Landkreis Deutschland
175	Bayreuth, Landkreis	Landkreis Deutschland
176	Coburg, Landkreis	Landkreis Deutschland
177	Forchheim	Landkreis Deutschland
178	Hof, Landkreis	Landkreis Deutschland
179	Kronach	Landkreis Deutschland
180	Kulmbach	Landkreis Deutschland
181	Lichtenfels	Landkreis Deutschland
182	Wunsiedel i. Fichtelgebirge	Landkreis Deutschland
183	Ansbach, Kreisfreie Stadt	Landkreis Deutschland
184	Erlangen, Kreisfreie Stadt	Landkreis Deutschland
185	Fürth, Kreisfreie Stadt	Landkreis Deutschland
186	Nürnberg, Kreisfreie Stadt	Landkreis Deutschland
187	Schwabach, Kreisfreie Stadt	Landkreis Deutschland
188	Ansbach, Landkreis	Landkreis Deutschland
189	Erlangen-Höchstadt	Landkreis Deutschland
190	Fürth, Landkreis	Landkreis Deutschland
191	Nürnberger Land	Landkreis Deutschland
192	Neustadt a.d. Aisch-Bad Windsheim	Landkreis Deutschland
193	Roth	Landkreis Deutschland
194	Weißenburg-Gunzenhausen	Landkreis Deutschland
195	Aschaffenburg, Kreisfreie Stadt	Landkreis Deutschland
196	Schweinfurt, Kreisfreie Stadt	Landkreis Deutschland
197	Würzburg, Kreisfreie Stadt	Landkreis Deutschland
198	Aschaffenburg, Landkreis	Landkreis Deutschland
199	Bad Kissingen	Landkreis Deutschland

200	Rhön-Grabfeld	Landkreis Deutschland
201	Haßberge	Landkreis Deutschland
202	Kitzingen	Landkreis Deutschland
203	Miltenberg	Landkreis Deutschland
204	Main-Spessart	Landkreis Deutschland
205	Schweinfurt, Landkreis	Landkreis Deutschland
206	Würzburg, Landkreis	Landkreis Deutschland
207	Augsburg, Kreisfreie Stadt	Landkreis Deutschland
208	Kaufbeuren, Kreisfreie Stadt	Landkreis Deutschland
209	Kempten (Allgäu), Kreisfreie Stadt	Landkreis Deutschland
210	Memmingen, Kreisfreie Stadt	Landkreis Deutschland
211	Aichach-Friedberg	Landkreis Deutschland
212	Augsburg, Landkreis	Landkreis Deutschland
213	Dillingen a.d. Donau	Landkreis Deutschland
214	Günzburg	Landkreis Deutschland
215	Neu-Ulm	Landkreis Deutschland
216	Lindau (Bodensee)	Landkreis Deutschland
217	Ostallgäu	Landkreis Deutschland
218	Unterallgäu	Landkreis Deutschland
219	Donau-Ries	Landkreis Deutschland
220	Oberallgäu	Landkreis Deutschland
221	Berlin	Landkreis Deutschland
222	Brandenburg an der Havel, Kreisfreie Stadt	Landkreis Deutschland
223	Cottbus, Kreisfreie Stadt	Landkreis Deutschland
224	Frankfurt (Oder), Kreisfreie Stadt	Landkreis Deutschland
225	Potsdam, Kreisfreie Stadt	Landkreis Deutschland
226	Barnim	Landkreis Deutschland
227	Dahme-Spreewald	Landkreis Deutschland
228	Elbe-Elster	Landkreis Deutschland
229	Havelland	Landkreis Deutschland
230	Märkisch-Oderland	Landkreis Deutschland
231	Oberhavel	Landkreis Deutschland
232	Oberspreewald-Lausitz	Landkreis Deutschland
233	Oder-Spree	Landkreis Deutschland
234	Ostprignitz-Ruppin	Landkreis Deutschland
235	Potsdam-Mittelmark	Landkreis Deutschland
236	Prignitz	Landkreis Deutschland
237	Spree-Neiße	Landkreis Deutschland
238	Teltow-Fläming	Landkreis Deutschland
239	Uckermark	Landkreis Deutschland
240	Bremen, Kreisfreie Stadt	Landkreis Deutschland
241	Bremerhaven, Kreisfreie Stadt	Landkreis Deutschland
242	Hamburg	Landkreis Deutschland
243	Darmstadt, Kreisfreie Stadt	Landkreis Deutschland
244	Frankfurt am Main, Kreisfreie Stadt	Landkreis Deutschland
245	Offenbach am Main, Kreisfreie Stadt	Landkreis Deutschland
246	Wiesbaden, Kreisfreie Stadt	Landkreis Deutschland
247	Bergstraße	Landkreis Deutschland
248	Darmstadt-Dieburg	Landkreis Deutschland
249	Groß-Gerau	Landkreis Deutschland
250	Hochtaunuskreis	Landkreis Deutschland
251	Main-Kinzig-Kreis	Landkreis Deutschland
252	Main-Taunus-Kreis	Landkreis Deutschland
253	Odenwaldkreis	Landkreis Deutschland
254	Offenbach, Landkreis	Landkreis Deutschland
255	Rheingau-Taunus-Kreis	Landkreis Deutschland

256	Wetteraukreis	Landkreis Deutschland
257	Gießen, Landkreis	Landkreis Deutschland
258	Lahn-Dill-Kreis	Landkreis Deutschland
259	Limburg-Weilburg	Landkreis Deutschland
260	Marburg-Biedenkopf	Landkreis Deutschland
261	Vogelsbergkreis	Landkreis Deutschland
262	Kassel, Kreisfreie Stadt	Landkreis Deutschland
263	Fulda	Landkreis Deutschland
264	Hersfeld-Rotenburg	Landkreis Deutschland
265	Kassel, Landkreis	Landkreis Deutschland
266	Schwalm-Eder-Kreis	Landkreis Deutschland
267	Waldeck-Frankenberg	Landkreis Deutschland
268	Werra-Meißner-Kreis	Landkreis Deutschland
269	Rostock, Kreisfreie Stadt	Landkreis Deutschland
270	Schwerin, Kreisfreie Stadt	Landkreis Deutschland
271	Mecklenburgische Seenplatte	Landkreis Deutschland
272	Landkreis Rostock	Landkreis Deutschland
273	Vorpommern-Rügen	Landkreis Deutschland
274	Nordwestmecklenburg	Landkreis Deutschland
275	Vorpommern-Greifswald	Landkreis Deutschland
276	Ludwigslust-Parchim	Landkreis Deutschland
277	Braunschweig, Kreisfreie Stadt	Landkreis Deutschland
278	Salzgitter, Kreisfreie Stadt	Landkreis Deutschland
279	Wolfsburg, Kreisfreie Stadt	Landkreis Deutschland
280	Gifhorn	Landkreis Deutschland
281	Goslar	Landkreis Deutschland
282	Helmstedt	Landkreis Deutschland
283	Northeim	Landkreis Deutschland
284	Peine	Landkreis Deutschland
285	Wolfenbüttel	Landkreis Deutschland
286	Göttingen	Landkreis Deutschland
287	Diepholz	Landkreis Deutschland
288	Hameln-Pyrmont	Landkreis Deutschland
289	Hildesheim	Landkreis Deutschland
290	Holzminden	Landkreis Deutschland
291	Nienburg (Weser)	Landkreis Deutschland
292	Schaumburg	Landkreis Deutschland
293	Region Hannover	Landkreis Deutschland
294	Celle	Landkreis Deutschland
295	Cuxhaven	Landkreis Deutschland
296	Harburg	Landkreis Deutschland
297	Lüchow-Dannenberg	Landkreis Deutschland
298	Lüneburg, Landkreis	Landkreis Deutschland
299	Osterholz	Landkreis Deutschland
300	Rotenburg (Wümme)	Landkreis Deutschland
301	Heidekreis	Landkreis Deutschland
302	Stade	Landkreis Deutschland
303	Uelzen	Landkreis Deutschland
304	Verden	Landkreis Deutschland
305	Delmenhorst, Kreisfreie Stadt	Landkreis Deutschland
306	Emden, Kreisfreie Stadt	Landkreis Deutschland
307	Oldenburg (Oldenburg), Kreisfreie Stadt	Landkreis Deutschland
308	Osnabrück, Kreisfreie Stadt	Landkreis Deutschland
309	Wilhelmshaven, Kreisfreie Stadt	Landkreis Deutschland
310	Ammerland	Landkreis Deutschland
311	Aurich	Landkreis Deutschland

312	Cloppenburg	Landkreis Deutschland
313	Emsland	Landkreis Deutschland
314	Friesland (DE)	Landkreis Deutschland
315	Grafschaft Bentheim	Landkreis Deutschland
316	Leer	Landkreis Deutschland
317	Oldenburg, Landkreis	Landkreis Deutschland
318	Osnabrück, Landkreis	Landkreis Deutschland
319	Vechta	Landkreis Deutschland
320	Wesermarsch	Landkreis Deutschland
321	Wittmund	Landkreis Deutschland
322	Düsseldorf, Kreisfreie Stadt	Landkreis Deutschland
323	Duisburg, Kreisfreie Stadt	Landkreis Deutschland
324	Essen, Kreisfreie Stadt	Landkreis Deutschland
325	Krefeld, Kreisfreie Stadt	Landkreis Deutschland
326	Mönchengladbach, Kreisfreie Stadt	Landkreis Deutschland
327	Mülheim an der Ruhr, Kreisfreie Stadt	Landkreis Deutschland
328	Oberhausen, Kreisfreie Stadt	Landkreis Deutschland
329	Remscheid, Kreisfreie Stadt	Landkreis Deutschland
330	Solingen, Kreisfreie Stadt	Landkreis Deutschland
331	Wuppertal, Kreisfreie Stadt	Landkreis Deutschland
332	Kleve	Landkreis Deutschland
333	Mettmann	Landkreis Deutschland
334	Rhein-Kreis Neuss	Landkreis Deutschland
335	Viersen	Landkreis Deutschland
336	Wesel	Landkreis Deutschland
337	Bonn, Kreisfreie Stadt	Landkreis Deutschland
338	Köln, Kreisfreie Stadt	Landkreis Deutschland
339	Leverkusen, Kreisfreie Stadt	Landkreis Deutschland
340	Düren	Landkreis Deutschland
341	Rhein-Erft-Kreis	Landkreis Deutschland
342	Euskirchen	Landkreis Deutschland
343	Heinsberg	Landkreis Deutschland
344	Oberbergischer Kreis	Landkreis Deutschland
345	Rheinisch-Bergischer-Kreis	Landkreis Deutschland
346	Rhein-Sieg-Kreis	Landkreis Deutschland
347	Städteregion Aachen	Landkreis Deutschland
348	Bottrop, Kreisfreie Stadt	Landkreis Deutschland
349	Gelsenkirchen, Kreisfreie Stadt	Landkreis Deutschland
350	Münster, Kreisfreie Stadt	Landkreis Deutschland
351	Borken	Landkreis Deutschland
352	Coesfeld	Landkreis Deutschland
353	Recklinghausen	Landkreis Deutschland
354	Steinfurt	Landkreis Deutschland
355	Warendorf	Landkreis Deutschland
356	Bielefeld, Kreisfreie Stadt	Landkreis Deutschland
357	Gütersloh	Landkreis Deutschland
358	Herford	Landkreis Deutschland
359	Höxter	Landkreis Deutschland
360	Lippe	Landkreis Deutschland
361	Minden-Lübbecke	Landkreis Deutschland
362	Paderborn	Landkreis Deutschland
363	Bochum, Kreisfreie Stadt	Landkreis Deutschland
364	Dortmund, Kreisfreie Stadt	Landkreis Deutschland
365	Hagen, Kreisfreie Stadt	Landkreis Deutschland
366	Hamm, Kreisfreie Stadt	Landkreis Deutschland
367	Herne, Kreisfreie Stadt	Landkreis Deutschland

368	Ennepe-Ruhr-Kreis	Landkreis Deutschland
369	Hochsauerlandkreis	Landkreis Deutschland
370	Märkischer Kreis	Landkreis Deutschland
371	Olpe	Landkreis Deutschland
372	Siegen-Wittgenstein	Landkreis Deutschland
373	Soest	Landkreis Deutschland
374	Unna	Landkreis Deutschland
375	Koblenz, Kreisfreie Stadt	Landkreis Deutschland
376	Ahrweiler	Landkreis Deutschland
377	Altenkirchen (Westerwald)	Landkreis Deutschland
378	Bad Kreuznach	Landkreis Deutschland
379	Birkenfeld	Landkreis Deutschland
380	Mayen-Koblenz	Landkreis Deutschland
381	Neuwied	Landkreis Deutschland
382	Rhein-Lahn-Kreis	Landkreis Deutschland
383	Westerwaldkreis	Landkreis Deutschland
384	Cochem-Zell	Landkreis Deutschland
385	Rhein-Hunsrück-Kreis	Landkreis Deutschland
386	Trier, Kreisfreie Stadt	Landkreis Deutschland
387	Bernkastel-Wittlich	Landkreis Deutschland
388	Eifelkreis Bitburg-Prüm	Landkreis Deutschland
389	Vulkaneifel	Landkreis Deutschland
390	Trier-Saarburg	Landkreis Deutschland
391	Frankenthal (Pfalz), Kreisfreie Stadt	Landkreis Deutschland
392	Kaiserslautern, Kreisfreie Stadt	Landkreis Deutschland
393	Landau in der Pfalz, Kreisfreie Stadt	Landkreis Deutschland
394	Ludwigshafen am Rhein, Kreisfreie Stadt	Landkreis Deutschland
395	Mainz, Kreisfreie Stadt	Landkreis Deutschland
396	Neustadt an der Weinstraße, Kreisfreie Stadt	Landkreis Deutschland
397	Pirmasens, Kreisfreie Stadt	Landkreis Deutschland
398	Speyer, Kreisfreie Stadt	Landkreis Deutschland
399	Worms, Kreisfreie Stadt	Landkreis Deutschland
400	Zweibrücken, Kreisfreie Stadt	Landkreis Deutschland
401	Alzey-Worms	Landkreis Deutschland
402	Bad Dürkheim	Landkreis Deutschland
403	Donnersbergkreis	Landkreis Deutschland
404	Germersheim	Landkreis Deutschland
405	Kaiserslautern, Landkreis	Landkreis Deutschland
406	Kusel	Landkreis Deutschland
407	Südliche Weinstraße	Landkreis Deutschland
408	Rhein-Pfalz-Kreis	Landkreis Deutschland
409	Mainz-Bingen	Landkreis Deutschland
410	Südwestpfalz	Landkreis Deutschland
411	Regionalverband Saarbrücken	Landkreis Deutschland
412	Merzig-Wadern	Landkreis Deutschland
413	Neunkirchen	Landkreis Deutschland
414	Saarlouis	Landkreis Deutschland
415	Saarpfalz-Kreis	Landkreis Deutschland
416	St. Wendel	Landkreis Deutschland
417	Dresden, Kreisfreie Stadt	Landkreis Deutschland
418	Bautzen	Landkreis Deutschland
419	Görlitz	Landkreis Deutschland
420	Meißen	Landkreis Deutschland
421	Sächsische Schweiz-Osterzgebirge	Landkreis Deutschland
422	Chemnitz, Kreisfreie Stadt	Landkreis Deutschland
423	Erzgebirgskreis	Landkreis Deutschland

424	Mittelsachsen	Landkreis Deutschland
425	Vogtlandkreis	Landkreis Deutschland
426	Zwickau	Landkreis Deutschland
427	Leipzig, Kreisfreie Stadt	Landkreis Deutschland
428	Leipzig	Landkreis Deutschland
429	Nordsachsen	Landkreis Deutschland
430	Dessau-Roßlau, Kreisfreie Stadt	Landkreis Deutschland
431	Halle (Saale), Kreisfreie Stadt	Landkreis Deutschland
432	Magdeburg, Kreisfreie Stadt	Landkreis Deutschland
433	Altmarkkreis Salzwedel	Landkreis Deutschland
434	Anhalt-Bitterfeld	Landkreis Deutschland
435	Jerichower Land	Landkreis Deutschland
436	Börde	Landkreis Deutschland
437	Burgenlandkreis	Landkreis Deutschland
438	Harz	Landkreis Deutschland
439	Mansfeld-Südharz	Landkreis Deutschland
440	Saalekreis	Landkreis Deutschland
441	Salzlandkreis	Landkreis Deutschland
442	Stendal	Landkreis Deutschland
443	Wittenberg	Landkreis Deutschland
444	Flensburg, Kreisfreie Stadt	Landkreis Deutschland
445	Kiel, Kreisfreie Stadt	Landkreis Deutschland
446	Lübeck, Kreisfreie Stadt	Landkreis Deutschland
447	Neumünster, Kreisfreie Stadt	Landkreis Deutschland
448	Dithmarschen	Landkreis Deutschland
449	Herzogtum Lauenburg	Landkreis Deutschland
450	Nordfriesland	Landkreis Deutschland
451	Ostholstein	Landkreis Deutschland
452	Pinneberg	Landkreis Deutschland
453	Plön	Landkreis Deutschland
454	Rendsburg-Eckernförde	Landkreis Deutschland
455	Schleswig-Flensburg	Landkreis Deutschland
456	Segeberg	Landkreis Deutschland
457	Steinburg	Landkreis Deutschland
458	Stormarn	Landkreis Deutschland
459	Erfurt, Kreisfreie Stadt	Landkreis Deutschland
460	Gera, Kreisfreie Stadt	Landkreis Deutschland
461	Jena, Kreisfreie Stadt	Landkreis Deutschland
462	Suhl, Kreisfreie Stadt	Landkreis Deutschland
463	Weimar, Kreisfreie Stadt	Landkreis Deutschland
464	Eichsfeld	Landkreis Deutschland
465	Nordhausen	Landkreis Deutschland
466	Unstrut-Hainich-Kreis	Landkreis Deutschland
467	Kyffhäuserkreis	Landkreis Deutschland
468	Schmalkalden-Meiningen	Landkreis Deutschland
469	Gotha	Landkreis Deutschland
470	Sömmerda	Landkreis Deutschland
471	Hildburghausen	Landkreis Deutschland
472	Ilm-Kreis	Landkreis Deutschland
473	Weimarer Land	Landkreis Deutschland
474	Sonneberg	Landkreis Deutschland
475	Saalfeld-Rudolstadt	Landkreis Deutschland
476	Saale-Holzland-Kreis	Landkreis Deutschland
477	Saale-Orla-Kreis	Landkreis Deutschland
478	Greiz	Landkreis Deutschland
479	Altenburger Land	Landkreis Deutschland

480	Eisenach, Kreisfreie Stadt	Landkreis Deutschland
481	Wartburgkreis	Landkreis Deutschland
482	Germany_Baden-Wuerttemberg	Bundesland Deutschland
483	Germany_Bayern	Bundesland Deutschland
484	Germany_Berlin	Bundesland Deutschland
485	Germany_Brandenburg	Bundesland Deutschland
486	Germany_Bremen	Bundesland Deutschland
487	Germany_Hamburg	Bundesland Deutschland
488	Germany_Hessen	Bundesland Deutschland
489	Germany_Mecklenburg-Vorpommern	Bundesland Deutschland
490	Germany_Niedersachsen	Bundesland Deutschland
491	Germany_Nordrhein-Westfalen	Bundesland Deutschland
492	Germany_Rheinland-Pfalz	Bundesland Deutschland
493	Germany_Saarland	Bundesland Deutschland
494	Germany_Sachsen-Anhalt	Bundesland Deutschland
495	Germany_Sachsen	Bundesland Deutschland
496	Germany_Schleswig-Holstein	Bundesland Deutschland
497	Germany_Thueringen	Bundesland Deutschland
498	Germany_East	Region Deutschland
499	Germany_North	Region Deutschland
500	Germany_West	Region Deutschland
501	Germany_South	Region Deutschland
502	IAWAK-EE	Region NUKLEUS-project
503	KlimaKonform	Region NUKLEUS-project
504	KlimaKonform+EZG_WE	Region NUKLEUS-project
505	EZG_WE	Region NUKLEUS-project
506	KlimaKonform_BBOX	Region NUKLEUS-project
507	KARE	Region NUKLEUS-project
508	ISAP	Region NUKLEUS-project
509	WAKOS_all	Region NUKLEUS-project
510	WAKOS_Festland	Region NUKLEUS-project
511	WAKOS_Inseln	Region NUKLEUS-project
512	Duisburg	Region NUKLEUS-project
513	Duisburg_3x3	Region NUKLEUS-project

Appendix D

Jupyter notebook to visualize and add custom subregions

We here provide a pdf copy of a jupyter notebook to visualize subregions and add custom subregions. In this example we show how to add a rectangular subregion, which covers entire Europe. The jupyter notebook can be found here:
external/geopackage/Plot_subregions_from_gpkg_file_and_add_subregions.ipynb.

Example: Plotting regions from gpkg file and adding polygons

Christoph Braun, KIT, July 2024

Load required modules

In [1]:
```
import geopandas as gpd
from shapely.geometry import Polygon
```

Load data

In [2]:
```
proj = "PROJECT"
user = "USER"
```

In [3]:
```
path_to_example_data = "/work/"+proj+"/"+user+ \
                       "/evasuite_demofiles/example_data/"
```

In [4]:
```
input_file = path_to_example_data+"EvaSuite_subregion_shapes.gpkg"
```

In [5]:
```
data = gpd.read_file(input_file)
```

Explore data

In [6]:
```
data
```

Out[6]:

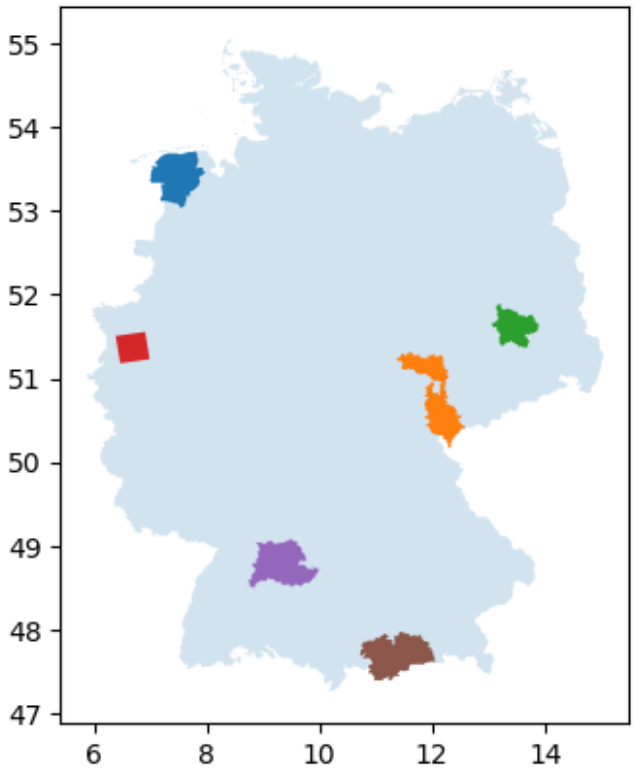
	ID	NAME_0	ID_0	ID_1	geometry
0	1.0	P_Alps	NaN	NaN	POLYGON ((5.00000 44.00000, 5.00000 48.00000, ...
1	2.0	P_BritishIslands	NaN	NaN	POLYGON ((-10.00000 50.00000, -10.00000 59.000...
2	3.0	P_East-Europe	NaN	NaN	POLYGON ((16.00000 44.00000, 16.00000 55.00000...
3	4.0	P_France	NaN	NaN	POLYGON ((-5.00000 44.00000, -5.00000 50.00000...
4	5.0	P_IberianPeninsula	NaN	NaN	POLYGON ((-10.00000 36.00000, -10.00000 44.000...
...	...	...	...	...	...
89	NaN	Duisburg_3x3	NaN	NaN	MULTIPOLYGON (((6.46907 51.17857, 6.98961 51.2...
90	NaN	EZG_Rhein	NaN	NaN	MULTIPOLYGON (((7.36470 51.96069, 7.37073 51.9...
91	NaN	EZG_Rhein_N	NaN	NaN	MULTIPOLYGON (((7.36470 51.96069, 7.37073 51.9...
92	NaN	EZG_Rhein_S	NaN	NaN	MULTIPOLYGON (((6.39623 49.77450, 6.39775 49.7...
93	NaN	DTL	NaN	NaN	MULTIPOLYGON (((9.12593 47.66864, 9.12577 47.6...

94 rows × 5 columns

... e.g. plot NUKLEUS regions

In [7]:
```
ax = data[data['NAME_0'] == 'Germany'].plot(alpha=0.2)
data[data['NAME_0'] == 'WAKOS_Festland'].plot(ax=ax)
data[data['NAME_0'] == 'KlimaKonform'].plot(ax=ax,color='C1')
data[data['NAME_0'] == 'IAWAK-EE'].plot(ax=ax,color='C2')
data[data['NAME_0'] == 'Duisburg_3x3'].plot(ax=ax,color='C3')
data[data['NAME_0'] == 'ISAP'].plot(ax=ax,color='C4')
data[data['NAME_0'] == 'KARE'].plot(ax=ax,color='C5')
```

Out[7]:
```
<AxesSubplot:>
```



Adding a new subdomain and writing to file

Create new polygon covering entire European region

In [8]:
```
lat_point_list = [35, 35, 70, 70, 35]
lon_point_list = [-10, 25, 25, -10, -10]

polygon_geom = Polygon(zip(lon_point_list, lat_point_list))
```

Add new polygon to data

In [9]:
```
# get last position of dataframe
entry_position = len(data)
```

In [10]:
```
data.loc[entry_position] = [entry_position,
                            "EUROPE_RECT",
                            "NaN",
                            "NaN",
                            polygon_geom]
```

Check resulting dataframe

In [11]:
```
data
```

Out[11]:

	ID	NAME_0	ID_0	ID_1	geometry
0	1.0	P_Alps	NaN	NaN	POLYGON ((5.00000 44.00000, 5.00000 48.00000, ...
1	2.0	P_BritishIslands	NaN	NaN	POLYGON ((-10.00000 50.00000, -10.00000 59.000...
2	3.0	P_East-Europe	NaN	NaN	POLYGON ((16.00000 44.00000, 16.00000 55.00000...
3	4.0	P_France	NaN	NaN	POLYGON ((-5.00000 44.00000, -5.00000 50.00000...
4	5.0	P_IberianPeninsula	NaN	NaN	POLYGON ((-10.00000 36.00000, -10.00000 44.000...
...	...	...	...	...	...
90	NaN	EZG_Rhein	NaN	NaN	MULTIPOLYGON (((7.36470 51.96069, 7.37073 51.9...
91	NaN	EZG_Rhein_N	NaN	NaN	MULTIPOLYGON (((7.36470 51.96069, 7.37073 51.9...
92	NaN	EZG_Rhein_S	NaN	NaN	MULTIPOLYGON (((6.39623 49.77450, 6.39775 49.7...
93	NaN	DTL	NaN	NaN	MULTIPOLYGON (((9.12593 47.66864, 9.12577 47.6...
94	94.0	EUROPE_RECT	NaN	NaN	POLYGON ((-10.00000 35.00000, 25.00000 35.0000...

95 rows × 5 columns

Write dataframe to gkpg file

In [12]:
```
data.to_file(path_to_example_data+"EvaSuite_subregion_shapes_extended.gpkg",
             driver="GPKG")
```

Appendix E

Available metrics in EvaSuite

This chapter was written by Sergei Petrov, Ronny Petrik, and Beate Geyer.

1 General Information

1.1 One model evaluation

Let **OBS** and **MOD** be some observational data and the model output for a specific variable, which is stored as a time series at the moments $t_k, k = \overline{1, N_T}$ on a structured 2D-grid $(x_i, y_j),\ i = \overline{1, N_x},\ j = \overline{1, N_y}$

$$\mathrm{OBS}(x_i, y_j, t_k),\ \mathrm{MOD}(x_i, y_j, t_k). \tag{1}$$

The list of available metrics is shown in the following table.

Eva-Suite name	Brief description	Output
'BIAS_temporal'	Temporal bias & Mean over time temporal bias for each location	$\mathbf{e_t}(x_i, y_j, t_k)$, $\mathbf{me_t}(x_i, y_j)$
'BIASnorm_temporal'	Normalized bias over time & Mean over time bias normalized by the temporally averaged observation values for each location	$\mathbf{e_{t,n}}(x_i, y_j, t_k)$, $\mathbf{me_{t,n}}(x_i, y_j)$
'BIAS_spatial'	Spatial bias & Mean over time spatial bias	$\mathbf{e_s}(t_k)$, $\mathbf{me_s}$
'BIASnorm_spatial'	Normalized spatial bias & Mean spatial bias normalized by spatially and temporally averaged observation values	$\mathbf{e_{s,n}}(t_k)$, $\mathbf{me_{s,n}}$
'MAE_temporal'	Temporal absolute bias & Mean over time temporal absolute bias for each location	$\mathbf{ae_t}(x_i, y_j, t_k)$, $\mathbf{mae_t}(x_i, y_j)$
'MAE_spatial'	Spatial absolute bias & Mean over time spatial absolute bias	$\mathbf{ae_s}(t_k)$, $\mathbf{mae_s}$
'MAD_temporal'	Temporal absolute bias & Median absolute deviation for each location over time	$\mathbf{ae_t}(x_i, y_j, t_k)$, $\mathbf{mad_t}(x_i, y_j)$
'MAD_spatial'	Spatial median absolute deviation & Mean over time spatial median absolute deviation	$\mathbf{mad_s}(t_k)$, $\mathbf{mmad_s}$
'RMSE_temporal'	The root of mean square error for each location over time	$\mathbf{rmse_t}(x_i, y_j)$
'RMSEbcn_temporal'	Normalized by mean over time observations unbiased root mean square error	$\mathbf{rmse_{t,n}}(x_i, y_j)$
'RMSE_spatial'	The spatial root mean square error & Mean over time spatial root mean square error	$\mathbf{rmse_s}(t_k)$, $\mathbf{mrmse_s}$
'LinCorr_temporal'	the linear correlation for each location over time	-
'LinCorr_spatial'	the Pearson correlation coefficient over in 2D space for each time step and temporal aggregation according to user-settings	-
'KlingGupta_temporal'	the Kling-Gupta efficiency for each location over time (composition of Linear correlation, BIAS and variability)	-
'quantile_bias_temporal'	the bias of quantiles for each location, computed from the PDF of all values in time	-
'quantile_mae_temporal'	the absolute mean error of quantiles for each location, computed from the PDF of all values in time	-
'quantile_iqd_temporal'	the inverse quadratic distance of quantiles for each location, computed from the PDF of all values in time	-
'pdf_pss_temporal'	The Persson-Skill-score for each location, computed from the PDF of all values in time	-

1.2 Evaluation of two models

Let **OBS** and **MOD_A** as well as **MOD_B** be some observational data and the output of two different models for a specific variable, which is stored as a time series at the moments $t_k, k = \overline{1, N_T}$ on a structured 2D-grid $(x_i, y_j),\ i = \overline{1, N_x},\ j = \overline{1, N_y}$

$$\mathrm{OBS}(x_i, y_j, t_k),\ \mathrm{MOD_A}(x_i, y_j, t_k),\ \mathrm{MOD_B}(x_i, y_j, t_k). \tag{2}$$

Then, a so-called skill score compares the performance of model A and model B against the reference data set (OBS). The following table lists the available skill scores.

Eva-Suite name	Brief description	Output
'AMSESS_temporal'	the advanced mean squared error skill score for each location over time, which is a normalization of the original mean squared error skill score.	-
'AMSESSsigm_temporal'	the significance for RMSE-based skill score (AMSESS) of model A to be better than those of model B (significance level of 0.10)	-
'AMSESSsigs_temporal'	the significance for RMSE-based skill score (AMSESS) of model A to be better than those of model B (significance level of 0.05)	-
'BIASsigm_temporal'	the significance for the BIAS of model A to be smaller than those of model B (significance level of 0.10). -	
'BIASsigs_temporal'	the significance for the BIAS of model A to be smaller than those of model B (significance level of 0.05)	-

2 Full descriptions of metrics

2.1 BIASes

2.1.1 Temporal bias

The **temporal bias** is 2D-field of difference between model output and observations for each time step. The **mean temporal bias** is the averaged over time 2D-field of difference between model output and observations

$$\mathbf{e_t}(x_i, y_j, t_k) = \mathrm{MOD}(x_i, y_j, t_k) - \mathrm{OBS}(x_i, y_j, t_k) \tag{3}$$

$$\mathbf{me_t}(x_i, y_j) = \frac{\sum_{k=1}^{N_T} \mathbf{e_t}(x_i, y_j, t_k)}{N_T} \tag{4}$$

2.1.2 Normalized temporal bias

The **normalized temporal bias** is a 2D-field of difference between model output and observations divided by the observations at each time step. The **mean normalized temporal bias** is the averaged over time 2D-field of difference between model output and observations divided by the temporally mean value of observations

$$\mathbf{e_{t,n}}(x_i, y_j, t_k) = \frac{\mathbf{e_t}(x_i, y_j, t_k)}{\mathrm{OBS}(x_i, y_j, t_k)} \tag{5}$$

$$\mathbf{me_{t,n}}(x_i, y_j) = \mathbf{me_t}(x_i, y_j) \cdot \left(\frac{\sum_{k=1}^{N_T} \mathrm{OBS}(x_i, y_j, t_k)}{N_T} \right)^{-1} \tag{6}$$

2.1.3 Spatial bias

The **spatial bias** is the averaged over spatial dimensions 1D-function of difference between model output and observations. The **mean spatial bias** is a single number that represents a mean over time spatial bias

$$\mathbf{e_s}(t_k) = \frac{\sum_{i=1}^{N_x}\sum_{j=1}^{N_y} \mathbf{e_t}(x_i, y_j, t_k)}{N_x \cdot N_y} \tag{7}$$

$$\mathbf{me_s} = \frac{\sum_{k=1}^{N_T} \mathbf{e_s}(t_k)}{N_T} \tag{8}$$

2.1.4 Normalized spatial bias

The **normalized spatial bias** is the spatial bias normalized by spatially averaged observational data. The **mean normalized spatial bias** is the mean spatial bias normalized by spatially and temporally averaged observational data

$$\mathbf{e_{s,n}}(t_k) = \mathbf{e_s}(t_k) \cdot \left(\frac{\sum_{i=1}^{N_x}\sum_{j=1}^{N_y} \mathrm{OBS}(x_i, y_j, t_k)}{N_x \cdot N_y} \right)^{-1} \tag{9}$$

$$\mathbf{me_{s,n}} = \mathbf{me_s} \cdot \left(\frac{\sum_{i=1}^{N_x}\sum_{j=1}^{N_y}\sum_{k=1}^{N_T} \mathrm{OBS}(x_i, y_j, t_k)}{N_x \cdot N_y \cdot N_T} \right)^{-1} \tag{10}$$

2.2 Mean Absolute Errors (MAEs)

2.2.1 Temporal mean absolute error

The **temporal absolute error** is 2D-field of absolute difference between model output and observations. The **temporal mean absolute error** is the averaged over time temporal mean absolute error

$$\mathbf{ae_t}(x_i, y_j, t_k) = \left| \mathbf{e_t}(x_i, y_j, t_k) \right| \tag{11}$$

$$\mathbf{mae_t}(x_i, y_j) = \frac{\sum_{k=1}^{N_T} \mathbf{ae_t}(x_i, y_j, t_k)}{N_T} \tag{12}$$

2.2.2 Spatial mean absolute error

The **spatial absolute error** is the averaged over spatial dimensions 1D-function of the absolute difference between model output and observations. The **spatial mean absolute error** is the single number that represents a mean over time spatial absolute error

$$\mathbf{ae_s}(t_k) = \frac{\sum_{i=1}^{N_x}\sum_{j=1}^{N_y} \mathbf{ae_t}(x_i, y_j, t_k)}{N_x \cdot N_y} \tag{13}$$

$$\mathbf{mae_s} = \frac{\sum_{k=1}^{N_T} \mathbf{ae_s}(t_k)}{N_T} \tag{14}$$

2.3 Median Absolute Deviations (MADs)

We introduce the sample median function **MED** as the central value of the same sample in ascending order. Taking **MED** value over time for 2D time-series means taking the median value over time for each point of the grid. Taking **MED** value over space for 2D time-series means taking the central value (for 2D-sample with fixed time) for each time step.

2.3.1 Temporal median absolute deviation

The **temporal mean absolute deviation** is the median over time for the absolute difference between model output and the median over time observations

$$\mathbf{mad_t}(x_i, y_j) = \underset{t_k}{\mathrm{MED}}\Big(\Big|\mathrm{MOD}(x_i, y_j, t_k) - \underset{t_k}{\mathrm{MED}}\big[\mathrm{OBS}(x_i, y_j, t_k)\big]\Big|\Big) \tag{15}$$

2.3.2 Spatial median absolute deviation

The **spatial median absolute deviation** is the median over space dimensions for the absolute difference between model output and the median over space dimensions of observations. The **mean spatial mean absolute deviation** is the single number that represents the time-averaged spatial median absolute deviation

$$\mathbf{mad_s}(t_k) = \underset{x_i, y_j}{\mathrm{MED}}\Big(\Big|\mathrm{MOD}(x_i, y_j, t_k) - \underset{x_i, y_j}{\mathrm{MED}}\big[\mathrm{OBS}(x_i, y_j, t_k)\big]\Big|\Big) \tag{16}$$

$$\mathbf{mmad_s} = \frac{\sum\limits_{k=1}^{N_T} \mathbf{mad_s}(t_k)}{N_T} \tag{17}$$

2.4 Root mean square errors (RMSEs)

2.4.1 Temporal root mean square error

The **temporal root mean square error** is the root of the averaged over time squared bias for each location

$$\mathbf{rmse_t}(x_i, y_j) = \sqrt{\frac{\sum\limits_{k=1}^{N_T} \mathbf{e}_t(x_i, y_j, t_k)^2}{N_T}} \tag{18}$$

2.4.2 Normalized temporal root mean square error

The **normalized temporal root mean square error** is unbiased temporal root mean square error divided by mean over time observation data

$$\mathbf{rmse_{t,n}}(x_i, y_j) = \sqrt{\frac{\sum\limits_{k=1}^{N_T} \Big(\mathbf{e_t}(x_i, y_j, t_k) - \mathbf{me_t}(x_i, y_j)\Big)^2}{N_T}} \cdot \left(\frac{\sum\limits_{k=1}^{N_T} \mathrm{OBS}(x_i, y_j, t_k)}{N_T}\right)^{-1} \tag{19}$$

2.4.3 Spatial root mean square error

The **spatial root mean square error** is the root of the averaged over space dimensions squared bias for each time step. The **mean spatial root mean square error** is the temporal mean of spatial root mean square errors

$$\mathbf{rmse_s}(t_k) = \sqrt{\frac{\sum\limits_{i=1}^{N_x}\sum\limits_{j=1}^{N_y} \mathbf{e_t}(x_i, y_j, t_k)^2}{N_x \cdot N_y}} \tag{20}$$

$$\mathbf{mrmse_s} = \frac{\sum\limits_{k=1}^{N_T} \mathbf{rmse_s}(t_k)}{N_T} \tag{21}$$

2.5 Linear correlation

2.5.1 The temporal linear correlation

2.5.2 The spatial linear correlation / Pearson correlation

2.6 The Kling-Gupta efficiency

The efficiency is widely used for river discharge and reads

$$\mathrm{KGE} = 1 - \sqrt{(r-1)^2 + (\alpha-1)^2 + (\beta-1)^2} \tag{22}$$

involving the Pearson-Correlation coefficient r and the ratio α between the variability of modelled values and observed values. β is the so-called bias term, i. e. the ratio between the mean of the modelled values and the observed values. A perfect score would be equal to one and a considerably good performance is reached for values above 0.7.

2.7 Metrics based on probability density functions

2.7.1 BIAS of quantiles

The distribution of model and observations are analyzed for the quantiles 0.2, 0.4, 0.6, and 0.8 (standard settings). The bias for each of these quantiles is summed up. There is the option to remove the systematic bias from the calculation.

2.7.2 Mean absolute error of quantiles

The distribution of modelled values and observations are analyzed for the quantiles 0.2, 0.4, 0.6, and 0.8 (standard settings). The mean absolute error of these quantiles is summed up. There is the option to remove the systematic bias from the calculation.

2.7.3 Inverse quadratic distance

The distribution of modelled values and observations are analyzed for the quantiles 0.2, 0.4, 0.6, and 0.8 (standard settings). The inverse quadratic distance of these quantiles is summed up. There is the option to remove the systematic bias from the calculation.

2.7.4 Persson Skill-score

The PDF from modelled values and observed values is computed based on user-defined bins and for each location. Then, both PDFs are compared by computing

2.8 Skill score based on the RMSE, AMSESS

The advanced mean squared error skill score compares the mean squared errors between model A and model B based on a reference data set. It reads A score of 1 means that model A is much better than model B, and -1 means that model B is much better than model A. A score equal to zero means that the performance of models A and B are close to each other.

2.9 Significance of model skill based on the RMSE

2.9.1 Moderate Significance of the AMSESS

The PDF of the advanced mean squared error skill score (see section 2.8) is analyzed using bootstrapping, i. e. 500 hundred samples of observational data and model data (models A and B) are created to compute the AMSESS. If 90 percent of the distribution is below zero, model B is defined to be significantly better than model A and vice versa (i. e. a one-sided significance test with a significance level of 0.10).

2.9.2 Strong Significance of the AMSESS

The PDF of the advanced mean squared error skill score (see section 2.8) is analyzed by means of bootstrapping, i. e. 500 hundred samples of observational data and model data (model A and B) are created to compute the AMSESS. If 95 percent of the distribution is below zero, model B is defined to be significantly better than model A and vice versa (i. e. a one-sided significance test with a significance level of 0.05).

2.10 Significance of model skill based on the BIAS

2.10.1 Moderate Significance of the BIAS

The PDF of the BIAS is analyzed by means of bootstrapping, i. e. 500 hundred samples of observational data and model data are created to compute the BIAS again and again for model A and model B. Then, the distribution of the BIAS in model A and model B are compared by computing the difference between both distributions. If 90 percent of the distribution is below zero, the BIAS of model B is defined to be significantly smaller than those of model A and vice versa (significance test with a significance level of 0.10).

2.10.2 Strong Significance of the BIAS

The PDF of the BIAS is analyzed by means of bootstrapping, i. e. 500 hundred samples of observational data and model data are created to compute the BIAS again and again for model A and model B. Then, the distribution of the BIAS in model A and model B are compared by computing the difference between both distributions. If 95 percent of the distribution is below zero, the BIAS of model B is defined to be significantly smaller than those of model A and vice versa (significance test with a significance level of 0.05).