Spatial verification tool

Documentation

El directorio de trabajo se encuentra en /home/esp0754/PERM/deode\_spatial\_verif.

La estructura que proponemos de la herramienta se encuentra en el PowerPoint de este mismo directorio. Los programas que componen la herramienta de verificación espacial se dividen dos categorías: (i) los programas necesarios para realizar la verificación espacial propiamente dicha y (ii) los programas adicionales que pueden ser útiles a la hora de usar la herramienta.

**Legend:**

(i) OBS: combination of the database name of the observations (<obs\_db>) along with the variable to be verified (<var\_verif>) joined with the character "\_", e.g. SEVIRI\_bt, IMERG\_pcp; (ii) Case: name of the extreme event to be verified, e.g. IANOS, VALENCIA; (iii) exp/s (expLowRes and/or expHighRes): name of the low and high spatial resolution experiments to be verified and compared, e.g. AIB\_46h1\_de2 (low res.) VAL500m\_46h1\_de2 (high res.).

# Configs

The **parameters to be configured** in order to perform the spatial verification are detailed in this section. This setup is divided into 3 types of configuration files:

## Observation config file

The templates of these files are located in the config/templates/ folder. Modified configuration files must be placed into config/ directory.

config\_<obs\_db>.yaml

* format. Name and extension of the files to be used as observations for spatial verification. The format of the files may not match the originals. This is due to the necessity for some databases to be processed before use. For instance: SEVIRI provides irradiance values from channel 9. Since post-processing of the raw data is required to convert to brightness temperature, the raw data are processed and saved in netCDF format for simplicity.
  + **filename**. [str]. Name of the observation file with extension, if any. It must contain the required characters to indicate the date format used by datetime.
  + **extension**. [str]. File format. Possible values are: netCDF, Grib2.
* vars. Information regarding the variable to be used for verification.
  + **<var\_verif>**. [str]. Variable name to be verified. Possible values are: pcp (precipitation), bt (brightness temperature), refl (maximum reflectivity).
    - **var\_raw**. [str]. Variable name to get from the raw files.
    - **postprocess**. [bool]. Boolean value. It indicates whether the raw files of the observations have been processed or not. This control parameter is used to tell the verification process which variable to get: 'var' or <var\_verif>.
    - **res**. [str]. Spatial resolution of the observations. IMPORTANT: there must be a single space between value and unit.
    - **description**. [str]. Description of <var\_verif>. Used as a colorbar label in plots.
    - **units**. [str]. Units of <var\_verif>.
    - FSS. Threshold and scales for FSS.
      * **thresholds**. [list]. List of the different thresholds (float values) to be verified in FSS.
      * **scales**. [list]. List of the different scales (int values) to be verified in FSS. Numbers relate to pixels, not length units.
    - SAL. Parameters to be used in SAL. These parameters are only used for object detection, which is required to compute structure and location values.
      * **f**. [float]. Multiplicative factor to apply in the SAL methodology following (Wernli et al., 2008). The value they set is 1/15.
      * **q**. [float]. Quartile of the observations/experiments to be estimated to set an object detection threshold. A value of 0.95 is recommended to avoid outliers (Gilleland et al., 2009).
      * **tstorm\_kwargs**. [dict]. Additional object detection parameters (see: https://github.com/pySTEPS/pysteps/blob/master/pysteps/feature/tstorm.py).

## Case config file

config\_<Case>.yaml

* dates. Time period where the extreme event occurs.
  + **ini**. [str]. Time (UTC) at which the event starts in the format "%Y%m%d%H".
  + **end**. [str]. Time (UTC) at which the event ends in the format "%Y%m%d%H".
* location. Spatial domain covered by the extreme event.
  + **zoom**. [list]. List of plotting domain coordinates for a higher zoom of the extrem event [<longitud\_min>, <longitud\_max>, <latitud\_min>, <latitud\_max>].
  + **NOzoom**. [list]. List of plotting domain coordinates [<lon\_min>, <lon\_max>, <lat\_min>, <lat\_max>].
* verif\_domain. A sequence of verification domains to be used as a common domain for the spatial verification of the experiments. If at the initial timestep where the event starts (see "dates": "ini") there is no verification domain defined, a cropping of the domain of the experiment to be verified is performed. This is not recommended since it does not ensure a common verification domain between experiments. The sequence of different verification domains at different time instants allows the tracking of a moving event. The Ianos medicane is an example.
  + **<%Y%m%d%H>([str])**. [list] List of verification domain coordinates [<lon\_min>, <lon\_max>, <lat\_min>, <lat\_max>].

## Experiment config file

config\_<exp>.yaml

* model.
  + **name**. [str]. Model name. Used for generated file names and plot titles.
* format. Name and extension of the files to be used as predictions for spatial verification.
  + **filepaths**. [list]. List with the different directories where the experiments are located. It is IMPORTANT that, even if there is only one directory, the path is preceded by a ",". If the path contains the name of the experiment, replace this name by "%exp". The use of several paths makes it possible to use, for the same experiment, high resolution simulations that have been generated with different domains. UNDER TESTING.
  + **filename**. [str]. Name of the experiment file with extension, if any. It must contain the required characters to indicate the date format used by datetime. The lead time must be indicated as “%LLL”.
  + **extension**. [str]. File format. Possible values are: netCDF, Grib2.
* inits. Initializations of the experiment to be used in the spatial verification.
  + <**init\_time**>. [str]. Format: "%Y%m%d%d%H".
    - **path**. [int]. Element of the "format": "filepaths" list to be used for this initialization. It is specified with an integer, where 0 indicates the first element. This allows selecting the appropriate domain in the case that two different domains of the same experiment include this initialization.
    - **fcast\_horiz**. [str]. Time (UTC) when the verification of this initialization is to be finished. It must be in the format "%Y%m%d%d%H". This may be because the event ends at that timestep or the experiment’s forecast horizon do not reach that timestep. It also allows to exclude timesteps where the event has moved and does not fall within the domain of the experiment.
* vars. Information regarding the variable to be used for verification.
  + <**var\_verif**>. [str]. Variable name to be verified. Possible values are: pcp (precipitation), bt (brightness temperature), refl (maximum reflectivity).
    - **var**. [int or str]. Variable name to get from the experiment files. If "format": "extension" is “Grib2”, "var" must be an integer with the message associated to the verification variable.
    - **accum**. [bool]. Boolean value. It indicates whether "var" is cumulative or not. If True, the previous instant is subtracted from the current instant to compute the current hourly value.
    - **postprocess**. [str]. If not "None", the code looks for this key in the postprocess\_function dictionary (from /scripts/libs/dicts.py) which will get the function with which to perform the postprocess transformation “var” to <var\_verif>. This function must be developed in /scripts/libs/PostProcess.py (note differences with "vars": <var\_verif>: "postprocess" from observation config file).
    - **negative\_values**. [bool]. Boolean value. It indicates if the variable should be analyzed with negative sign. This argument allows applying the FSS and SAL metrics when the variable contains mainly negative values and minimum values should be analyzed. For example, brightness temperature in units of ºC.
  + lat
    - **var**. [str]. Latitude name to get from the experiment files. If values are 1D, a 2D meshgrid is created.
    - **description**. [str]. Used as a description of this variable for regrided experiments in netCDF format.
  + lon
    - **var**. [str]. Longitude name to get from the experiment files. If values are 1D, a 2D meshgrid is created.
    - **description**. [str]. Used as a description of this variable for regrided experiments in netCDF format.

# Before start…

In order to perform the spatial verification, it is necessary to have observational databases. In addition, these databases generally need to be processed before they can be used for verification. Some scripts have been developed to perform this task. These scripts must be launched from the same directory where they are located. Before running a script, the configuration files of the observations and case study must have been properly created. In particular:

* config\_<obs\_db>.yaml:
  + format
    - filename
  + vars
    - <var\_verif>
      * var
* config\_<Case>.yaml:
  + dates
    - ini
    - end
  + location
    - NOzoom

## scripts/downloads/download\_<OBS>.py

To execute these scripts, username and password in nasa (for IMERG) and access credentials (key + password, for SEVIRI) are required. They can be executed via terminal:

python3 download\_<OBS>.py <Case> <user/key> <password/token>

These files perform the following tasks:

*IMERG*

1. Download 30-minutal files for the time span of the extreme event in the case study.
2. Average two intra-hourly HDF5 files to obtain the cumulative precipitation for that hour (the original variable (precipitationCal) is the 1-hour precipitation rate).
3. Extract 1D arrays of latitudes and longitudes and build a two-dimensional meshgrid with both.
4. Save the information (variable name: pcp) in netCDF format in the OBSERVATIONS/data\_IMERG\_pcp/<Case>/ directory. The computed values correspond to the accumulated precipitation at the previous hour.

*SEVIRI*

1. Download netCDF files with the closest “pasada” to the hours to be verified of the event (%H:57) for a region 5º larger than the plotting domain of the case study.
2. Compute the brightness temperature from the irradiance of channel 9 (see scripts/libs/PostProcess.IrradianceToBrightnessTemperature).
3. Build the latitude and longitude meshgrid.
4. Save the information (variable name: bt) in netCDF format in the OBSERVATIONS/data\_SEVIRI\_bt/<Case>/ directory. The computed values correspond to the approximate instantaneous brightness temperature of the date.

## scripts/utils/PostProccess\_OPERA\_<var\_verif>.py

OPERA raw files have been downloaded manually. They are located in the ATOS directory: /perm/esp0754/databases/. These files are represented in a given projection. A two-dimensional grid building procedure has been carried out. Therefore, the PostProccess\_OPERA\_<var\_verif> scripts:

1. Get the variable to be verified.
2. Build the two-dimensional grid of OPERA radar.
3. Save the values in netCDF format.

They can be executed via terminal:

python3 PostProccess\_OPERA\_<var\_verif>.py <path\_OPERA\_raw> <Case>

# Run spatial verification

The main.sh file runs the spatial verification scripts developed for the spatial verification exercise. The comparison between two experiments consists of 4 main steps and 3 optional steps. All scripts run with 3 arguments (see Legend at the beginning of this document). These arguments must be set in the preamble of the shell-script.

## Main scripts

The products generated by these scripts are required to perform the following steps. Therefore, if the program is not completed successfully, successive programs cannot be used.

1. The set\_environment.py script creates the required folders (if they do not exist) to save the generated products.
2. The regrid.py script links the simulations of an experiment to the directory (created in the previous step) and performs a linear interpolation of the data to the grid of the observations to be verified. Additionally, a map with the experiment data in its original grid is generated. This plot allows to check that the variable selected in the experiment is the correct one.
3. The verification.py script gets the arrays of observations and regridded experiments (generated in the previous step), crops a common verification domain and computes the FSS and SAL metrics. Results are saved in several plots splitted by initializations. Additionally, plots with the FSS and SAL verification for each time step are generated. The original SAL (from module pysteps.salscores.sal) has been modified and a new object detection figure is now generated. In addition, object detection is further configurable (see "vars": <var\_verif>: "SAL": "tstorm\_kwargs" from the observations configuration file). Default values for detection objects are shown in Appendix. For more detailed information on this, two jupyter notebooks are available upon request.
4. Finally, the compExps\_stats.py and compExps\_maps.py scripts generate summary plots that allow comparison of the verifications conducted in the previous step for the low and high spatial resolution experiments. This comparison is performed on the timesteps verified in both experiments (initializations and lead times).
   1. The comExps\_metrics.py script generates the mean FSS comparison, the FSS distributions for each threshold and scale, the SAL scatter-plot of both experiments, the SAL distributions, and a summary plot with boxplots and medians of FSS and SAL, respectively, for each initialization separately.
   2. The compExps\_maps.py script allows the comparison in a subjective way with a gif plot (and the frames that compose it) of the fields to be verified for each of the experiments against the observational values. In addition, an additional plot is generated with the total accumulated value during the extreme event if the variable to be verified is precipitation or with the maximum (minimum) of the event for each point if the variable to be verified is reflectivity (brightness temperature).

## Helpful scripts

The scripts detailed below require the successful execution of some of the main scripts.

1. The script plot\_regrid.py generates plots with maps for each timestep to be verified. Observational values are plotted in the left panel while the regridded experiments are plotted in the right panel. This script allows to check that the interpolation to the grid of observations is consistent. It is necessary to have previously executed the regrid.py script.
2. The program create\_panels.py combines all the plots generated from the verification for a timestep. To run this file, the main scripts 1-3 must have been run successfully previously. This script generates a 2x3 figure. The observations, the regridded experiment and the experiment with its original resolution are displayed in the top row. The bottom row shows the results of FSS, SAL and the objects found by the SAL, both the observations and the experiment.

# Appendix

Default values for SAL methodology following the pysteps module.

|  | f | q | Max\_num\_features | Minref | Maxref | Mindiff | Minmax |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Default | 1/15 | 0.95 | None | f\*Rq | f\*Rq+1·10-5 | 1·10-5 | f\*Rq |

where Rq is the value corresponding to q-quartile from input.

# References

Wernli, H., Paulat, M., Hagen, M., & Frei, C. (2008). SAL—A novel quality measure for the verification of quantitative precipitation forecasts. *Monthly Weather Review*, *136*(11), 4470-4487.

Gilleland, E., Ahijevych, D., & Ebert, B. (2009). Spatial forecast verification methods intercomparison. *Wea. Forecasting*.