Consortium for Small-Scale Modelling

Technical Report No. 33

V.A.S.T.
(Versus Additional Statistical Techniques)
User Manual (v2.0)

September 2017

DOI: 10.5676/DWD_pub/nwv/cosmo-tr_33

Deutscher Wetterdienst
MeteoSwiss
Ufficio Generale Spazio Aereo e Meteorologia
ΕΘΝΙΚΗ ΜΕΤΕΩΡΟΛΟΓΙΚΗ ΥΠΗΡΕΣΙΑ
Instytucie Meteorogii i Gospodarki Wodnej
Administratia Nationala de Meteorologie
ROSHYDROMET
Agenzia Regionale Protezione Ambiente Piemonte
Agenzia Regionale Prevenzione Ambiente Energia Emilia Romagna
Centro Italiano Ricerche Aerospaziali
Amt für GeoInformationswesen der Bundeswehr
Israel Meteorological Service

www.cosmo-model.org

Editor: Massimo Milelli, ARPA Piemonte

V.A.S.T.

(Versus Additional Statistical Techniques)

User Manual (v2.0)

N. Vela [*]

[*] ARPA Piemonte, University of Turin

Contents

Contents

1 Introduction

This document explains how to use the VAST software provided by Naima Vela (Arpa Piemonte/University of Turin) developed according to VERSUS2 priority project, task 4: "Introduction of additional statistical techniques in VERSUS".
The software can perform the following tasks:

- Open and read txt files of forecast and observation produced by any external software following an example scheme
- Produce spatial verification of the precipitation field (main goal), wind speed and total cloud cover.
- If the user has the possibility to use the LIBSIM[1] software, VAST can produce its own input files starting from GRIB (observation and forecast) of BUFR (observation)

1.1 Fuzzy verification

Fuzzy verification answers the question:"Which is the link between spatial forecast and a combination of the intensity of the precipitation (or wind speed or total cloud cover) and the scale of the event?"

The scale decomposition methods allow us to diagnose the model errors and performances according to different scales. The scale-intensity approach links the traditional bi-dimensional verification categories: it returns the model skills according to different precipitation intensities (or wind speed or total cloud cover) and spatial scales. It verifies the model over the whole domain. It is useful in the spatial verification of discontinuous fields. It supplies information for both single case studies and forecast systems evaluated over a longer time. Using a neighbourhood verification method an exact correspondence between forecast and observation is not needed.

It is important to understand that this type of verification, for what concerns precipitation, gives better information when performed for case studies on limited areas, so that the patterns can be compared without being overhung by long time or large areas of no rain condition.

2 Software description

The architecture of the VAST software has been produced after a deep study of the IDL code of the Ebert software for fuzzy verification.

2.1 Components

The VAST software is composed by the following files:

- `install.sh`: Shell script that creates the directory tree and places the other scripts in the correct directory

[1]The "`extra software`" for VERSUS users

- `use_libsim.sh`[2]: Shell script that allows LIBSIM users to produce the VAST input files (txt) starting from GRIB and BUFR
- `vast.sh`[3]: Shell script containing all the instructions to execute the different parts of the procedure
- `input_libsim.conf`: Configuration file used by the shell script `use_libsim.sh` to set all the variables that point to directories and name of the files
- `directories.conf`: Configuration file used by the shell script `vast.sh` to set all the variables that point to directories and name of the files
- `input_csv2txt.nml`: Namelist read by `csv2txt.f90` Fortran code containing all the instructions regarding the directories, the name of the files, the internal variables and the parameters
- `input_fuzzy.nml`: namelist read by `read_txt.f90` and `fuzzy_verify.f90` Fortran codes containing all the instructions regarding the directories, the name of the files, the internal variables and the parameters
- `csv2txt.f90`: Fortran code that reads the output files from LIBSIM and transform them into txt files read by VAST
- `read_txt.f90`: Fortran code able to process the input TXT files (observation and forecast) and produce the input for the second and main Fortran code
- `fuzzy_verify.f90`: Fortran code that uses the outputs of the first one to perform the verification.

2.2 Directories' structure

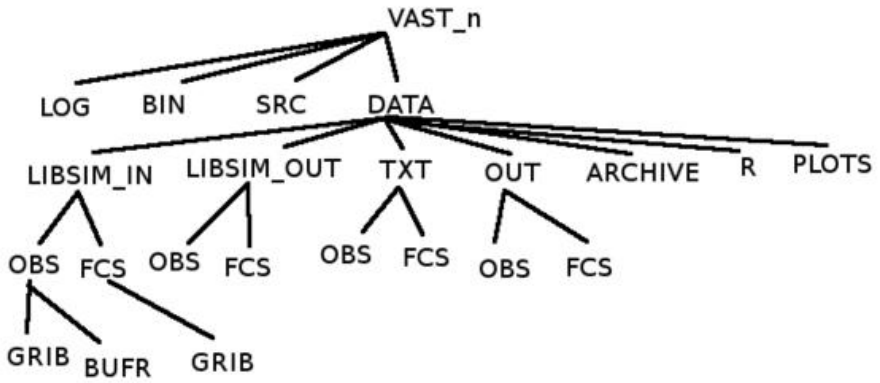

Figure 1: Default structure of the directories of the VAST software

[2]The file will contain numbers to identify the version of VAST to whom it belongs. i.e. `use_libsim_2_0.sh` will be referred to the VAST version 2.0

[3]The file will contain numbers to identify the version of VAST to whom it belongs. i.e. `vast_2_0.sh` will be referred to the VAST version 2.0

In Figure 1 it is possible to see the default structure of the directories in the software. If the presented structure conflicts with the users needs (particular displacement of the storage, or similar problems), it will be necessary to modify the configuration files accordingly.This has been made particularly hard by the requirement of having a common path that leads to the subdirectories (reducing the degrees of freedom of the structure of the software). In particular the user should not modify the relative path that links the `VAST/src` and `VAST/bin` directories.
The shell scripts "`use_libsim.sh`" and "`vast.sh`" will be automatically inserted in the directory "`bin`". The configuration files, the namelists and the Fortran codes will be automatically inserted in the directory "`src`".
If these files need to be displaced differently, the user will have to modify the scripts accordingly.
An archive file containing all the contingency tables calculated by the main Fortran code (`fuzzy_verify.f90`) during the verification will be created every time a verification is performed. It can be found in the directory `VAST/data/ARCHIVE` and it can be recognized by the date and time of the launch of the verification (`vast.sh`).

3 Installation and compilation

The user needs to install the Fortran compiler `gfortran` to use the Fortran codes contained in the package.
The user needs to install `R Project for Statistical Computing` to produce the plots of the needed indexes. In particular, the following libraries (and all their dependencies) need to be installed:

- `fields`
- `gplots`

To install the R packages, the user needs to log in as root, download the packages in the extension `package.tar.gz` and type the following command line:

- `R CMD INSTALL package.tar.gz`

The user needs then to unpack and copy the folder `vast_install` into the desired location by using the command line:

- `tar -xzf vast_install.tar.gz`

The user then needs to move into the `vast_install` directory to change the permissions for the shell script `install.sh` by giving the following command:

- `chmod 777 install.sh`

(The numbers could be different if the user wants to keep a particular category from executing the script)
Then the user needs to run the executable by giving the following command:

- `./install.sh`

The script will create all the directories needed by the software and will copy the scripts in the correct place. If the user needs a different structure for the directories, than it will be necessary to manually modify them and the codes.

Once the directories have been created, the user needs to place the input files in the correct place. If the user is using LIBSIM to pre-process the input files, than they will be put in the `obs` (`GRIB` or `BUFR`) and `fcs/GRIB` folders contained in the directory `VAST/data/LIBSIM_in`. If the user has a different software to produce the input files for VAST, than the preprocessed files must be inserted in the `obs` and `fcs` folders contained in the directory `VAST/data/TXT`.

All the shell scripts (`use_libsim.sh` if needed and `vast.sh`) have to be converted into executable files to be correctly used to produce the verification. To do so the user needs to move to the directory containing the executable files (`bin`) and use the following instruction from command line:

- `chmod 777` *`your_script.sh`*

(The numbers could be different if the user wants to keep a particular category from executing the script)
The compilation of the Fortran codes is automatically performed while launching the executables (`vast.sh` and `use_libsim.sh`).

3.1 NOTE FOR PEOPLE WHO HAVE ALREADY INSTALLED PREVIOUS VERSIONS OF VAST

If the user doesn't want to delete and recreate the directory's structure of VAST, he/she will have to delete all the files contained in the folders `VAST/bin` and `VAST/src` and substitute them with the files contained in the folder `vast_install_2_0` (with the exception of `install.sh`).

3.2 NOTE for users working on a VERSUS machine

If the user is running VAST on a VERSUS machine, it is possible to find all the needed extra packages for R in the directory `vast_install/documentation/R_packages_versus`
It is mandatory to use the given packages to be installed on the VERSUS version of the R software. Nothing changes for the installation instruction.

4 Activities performed by the software

4.1 For LIBSIM users

The LIBSIM users have the possibility to perform their verification starting from standard input files (GRIB or BUFR for the observation, GRIB for the forecast).

4.1.1 The input files

Input observation files: The observation files must be BUFR or GRIB. For a better behaviour of the software (faster and less memory demanding), it would be better to use files (BUFR or GRIB) that only contain the total precipitation variable. The files must be placed in the folder `VAST/data/LIBSIM_in/obs/BUFR` or `GRIB`. Since the files will be collected together by the command "`cat`", it is important that the GRIB files have a `.grb` extension, the BUFR files must have a `.bfr` extension.
Example files can be found in the folder `vast_install/documentation/data/LIBSIM_in/obs`.

Input forecast files: The forecast files must be GRIB. For a better behaviour of the software (faster and less memory demanding), it would be better to use files that only contain the total precipitation variable. The files must be placed in the folder `VAST/data/LIBSIM_in/fcs/GRIB`. Since the files will be collected together by the command "`cat`", it is important that they have a `.grb` extension.
Example files can be found in the folder `vast_install/documentation/data/LIBSIM_in/fcs`.

The characteristics of the input files can be deduced by analysing the example GRIB and BUFR given as examples with the package. Such details can be investigated by using the following commands:

- For GRIB:
  - `wgrib -V filename.grb`
- or
  - `grib_dump filename.grb`
- For BUFR:
  - the user can upload an example file to the url `http://apps.ecmwf.int/codes/bufr/validator/`

In Figure 2 it is possible to read the output of `wgrib -V` *`your_file.grb`* command for an observation GRIB.

```
rec 1:0:date 2014101321 APCP kpds5=61 kpds6=1 kpds7=0 levels=(0,0) grid=255 sfc nudge ana 0hr: bitmap: 23606 undef
  APCP=Total precipitation [kg/m^2]
  timerange 13 P1 0 P2 1 TimeU 1  nx 526 ny 518 GDS grid 0 num_in_ave 0 missing 0
  center 200 subcenter 255 process 1 Table 2 scan: WE:SN winds(grid)
  latlon: lat  42.950000 to 46.667000 by 0.007000  nxny 272468
          long 6.008000 to 11.310000 by 0.010000, (526 x 518) scan 64 mode 136 bdsgrid 1
  min/max data -2.96906e-06 11.4665  num bits 16  BDS_Ref -2.96906e-06  DecScale 0 BinScale -12
```

Figure 2: Output of `wgrib -V filename.grib` for observation

The most important parameter is the `timeRangeIndicator` which must be set to **13**

In Figure 3 it is possible to read the output of `wgrib` command for a forecast GRIB.

```
rec 1:0:date 2014101400 APCP kpds5=61 kpds6=1 kpds7=0 levels=(0,0) grid=255 sfc 0-10hr acc:
  APCP=Total precipitation [kg/m^2]
  timerange 4 P1 0 P2 10 TimeU 1  nx 447 ny 532 GDS grid 10 num_in_ave 0 missing 0
  center 200 subcenter 255 process 58 Table 2 scan: WE:SN winds(N/S)
  rotated LatLon grid  lat -21.925000 to -8.650000  lon -3.500000 to 7.650000
    nxny 237804  (447 x 532)  dx 25 dy 25  scan 64  mode 128
    transform: south pole lat -32.500000 lon 10.000000  rot angle 0.000000
  min/max data 0 78.7188  num bits 16  BDS_Ref 0  DecScale 0 BinScale -9
```

Figure 3: Output of `wgrib -V filename.grib` for forecast

Regular output files coming from COSMO models fit to this standard.

Thanks to the LIBSIM capabilities, the software works both with GRIB1 and GRIB2 files.

It is important that the **BUFR** files contain **only the hourly cumulate of the precipitation, without any duplicate**.

4.1.2 The configuration files

The elaboration of the input files for LIBSIM is performed thanks to two files: `use_libsim.sh` (the executable) and `input_libsim.conf` (its configuration file).

4.1.3 input_libsim.conf

The first part of the configuration file (Figure 4) contains the list of the directories that must be set for the correct functioning of the software:

```
#DIRECTORIES
#If any change has been made to the structure of VAST, than modify with the actual path
#
#COMMON PATH (up to the directory containing VAST)
common_path='/versus/'
#
#INPUT FILES
#Observation:
#IMPORTANT: comment the path not needed
#If GRIB files (comment if not):
dir_in_obs='VAST/data/LIBSIM_in/obs/GRIB/'
#If BUFR files (comment if not):
#dir_in_obs='VAST/data/LIBSIM_in/obs/BUFR/'
#Forecast
dir_in_fcs='VAST/data/LIBSIM_in/fcs/GRIB/'
#OUTPUT FILES
#Observation
dir_out_obs='VAST/data/LIBSIM_out/obs/'
#Forecast
dir_out_fcs='VAST/data/LIBSIM_out/fcs/'
#Fortran code
dir_src='VAST/src/'
#Executable
dir_bin='VAST/bin/'
#Log
dir_log='VAST/log/'
#
#
#OBSERVATION FORMAT: 'grib' or 'bufr'
#
format='grib'
```

Figure 4: Directories to be set to make the `use_libsim.sh` script work

It is important to notice that the user needs to set the type of input for the observation in two different ways (both need to be done):

- by uncomment the corresponding input path (and commenting the other)
- by setting the variable `format` to grib or bufr

The "`common_path`" variable consists in the absolute path where the `VAST` directory has been located in your machine, and it will be used to complete all the other relative paths (that **should not be modified**).

The second part of the configuration file (Figure 5) is dedicated to the specification of the name of the verification (that will be used to chose the correct output files in the second part of the verification), to specify the run of the model to be verified, to (optionally) select the input files by the leading part of their name (up to 10 carachters) and to select the cumulation period for the verification. The user must be aware of the value of the cumulation time of the input files and set the variable `cum` accordingly (the value must be equal or multiple of the input cumulation time).
Example: If the user have hourly cumulation for the observed precipitation and wants to re-cumulate it on the intervals 0-6, 6-12, 12-18, 18-24 (`cum='00 06'`), the first GRIB must contain the cumulated rain for the hour **one** (which means from 0 to 1).
If the input files are BUFR files, they must contain all the hourly cumulate of the precipitation OR the precipitation already cumulated on the desired period and no other cumulation periods.

```
#NAME OF THE VERIFICATION (string with no gaps)
name='201606_decade1_7km_12h'
#
#RUN (for forecast only, use string like '00', '06', '12', etc)
use_run='00'
#
#
#Identification of the files ('y'/'n')
#If 'y' then add string
add_id='y'
id_file='LAMIN0063'
#
#
#CUMULATION TIME
#The cumulation time must be equal
#or bigger (ONLY for precipitation) than the one
#of the input data
#The cumulation time must be given as follows.
#Ex: '00 03' means three hours cumulation
cum='00 12'
```

Figure 5: Name and cumulate parameters needed by LIBSIM

The third part of the configuration file (Figure 6) is dedicated to the configuration of the grid on which the verification will be performed. It is necessary for the grid to have lower resolution and a smaller area if compared to the forecast input, so that no missing point appear in the final result. The user will be able to check if the forecast grid contains any "holes" by plotting and visualising the CSV (directory `VAST/data/LIBSIM_out`) resulting files (an easy and free way to do that is to use the QGIS software. The user needs to see a perfect LON/LAT regular grid).

```
#GRID PARAMETERS
#The grid borders must be strictly smaller than the original area.
#The grid step must be strictly bigger than the original grid step.
#If the obs and fcs output files don't have the same grid
#it can mean that the selected area is too close to the original one
#(obs or fcs) or that the grid step is too close to the original one
#(obs or fcs) and holes have been left in the new grid.
#If the problem is the area, the user needs to reduce it.
#If the problem is the grid step, the user needs to enlarge it.
#So try with different parameters untill at least the forecast grid
#is regular and contains no holes
#Grid limits
x_min='6.315'
x_max='9.987'
y_min='43.1'
y_max='46.585'
#Number of grid points in each direction (x=lon, y=lat)
nx='39'
ny='40'
```

Figure 6: Grid parameters needed by LIBSIM

The data can include more than one day of precipitation but the grib messages must NOT overlap (i.e. day 01/01/2016 from hour 01 to hour 24 and then day 02/01/2016 from hour 01. **NOT** day 01/01/2016 from hour 01 to hour 72 and then day 02/01/2016 from hour 01.). The author suggests to use the software to verify single case studies (few days) over limited areas (i.e. the Piedmont region), since this kind of verification performs best when precipitation patterns are present and occupy the most part of the verified area. It is important that in the forecast files every date and time appear only once. If more files contain the same date and time (i.e. overlapping forecast runs) the resulting CSV will be overwritten.
The last part of the configuration file (Figure 7) contains the names of the files involved in the preprocessing of the files. The names of the files should be left as default.

```
#
#FORTRAN PROGRAM (should be left as default)
csv2txt_f90='csv2txt.f90'
#EXECUTABLE
csv2txt_exe='csv2txt'
csv2txt_log='csv2txt.log'
csv2txt_stderr='csv2txt_stderr.log'
#
#LISTS (should be left as default)
list_obs='list_obs.dat'
list_fcs='list_fcs.dat'
#
#NAMELIST (should be left as default)
namelist='input_csv2txt.nml'
```

Figure 7: Names of the files used by the preprocessing part of the software with LIBSIM

When all the variables have been set, the script `use_libsim.sh` will perform the following tasks:

- Collect all the input forecast files with the "`.grb`" extention AND the given specifications (run and, if chosen, name of the file) into a single grib called `forecast.grib`
  - it is VERY IMPORTANT that none of the forecast input files are called "`forecast.grib`". If that happens the procedure will cause a conflict and abort the execution
- Use the LIBSIM functionality "`vg6d_transform`" to re-grid the input files
- Count the number of the input messages

- Produce CSV files and place them in the correct directory
- Create a copy of a re-gridded forecast message to use it as a template to re-grid the observation
- Proceed to repeat the same operation for the observation files (with the extention ".grb" or "`.bfr`")
  - in case of grib files, the produced total input file is called "`observation.grib`"
  - in case of bufr files, the produced total input file is called "`observation.bufr`"
  - it is VERY IMPORTANT that none of the observation input files have one of these names. If that happens the procedure will cause a conflict and abort the execution

When all the CSV files are ready and displaced in the correct directory, the script creates lists of these files, and proceeds to compile and run the Fortran code (default is `csv2txt.f90`) that transforms the output of LIBSIM (`.csv`) into the input for VAST (`.txt`). So, to execute this code, the user must **first** complete the namelist file called `input_csv2txt.nml`. The user needs to substitute the example paths with the actual path of the half processed files(this needs to be done before launching the `use_libsim.sh` executable). The patterns to be completed are shown in Figure 8

```
&directories
!Fill in with the actual path (up to the directory containing VAST)
common_path='/versus/'
!Directories of input and output for csv2txt.f90
!(to be modified only if the user made modifications to the structure of VAST)
dirin_obs='VAST/data/LIBSIM_out/obs/'
dirin_fcs='VAST/data/LIBSIM_out/fcs/'
dirout_obs='VAST/data/TXT/obs/'
dirout_fcs='VAST/data/TXT/fcs/'
dir_log='VAST/log/'
&end

&filenames
!Lists of csv files (should be left as default)
filelist_obs='list_obs.dat'
filelist_fcs='list_fcs.dat'
!Name of the verification with no gaps
ver_name='201606_decade1_7km_12h'
&end
```

Figure 8: Configuration file for the Fortran code `csv2txt.f90`

This ends the first part of the procedure, executed through the file `use_libsim.sh`, dedicated to the LIBSIM users only.

4.2 For all users

4.2.1 The input files

The VAST input files must be `txt` files. Their internal structure is very strict but simple to reproduce, as shown in Figure 7

Longitude	Latitude	Value
6.315	43.010	-999.000
6.352	43.010	-999.000
6.389	43.010	-999.000
6.426	43.010	-999.000
6.463	43.010	-999.000
6.500	43.010	-999.000
6.538	43.010	-999.000
6.575	43.010	-999.000
6.612	43.010	0.000
6.649	43.010	0.000
6.686	43.010	0.000
6.723	43.010	0.000
6.760	43.010	0.000
6.797	43.010	0.000
6.834	43.010	0.000
6.871	43.010	0.000
6.908	43.010	0.000
6.946	43.010	0.000
6.983	43.010	0.000
7.020	43.010	0.000
7.057	43.010	0.000
7.094	43.010	0.471
7.131	43.010	1.525
7.168	43.010	1.434
7.205	43.010	0.805

Figure 9: Structure of the `txt` input file (observation or forecast)

The file is composed by:

- One header line containing the **exact** string: " `Longitude Latitude Value`"
- One column containing the longitude values
- One column containing the latitude values
- One column containing the precipitation, wind speed or total cloud cover values (-999 if no data)

The order of the coordinates is important. The first couple of coordinates must be the lower left corner and the longitude is the parameter that increases first. The coordinate values must be the same for the forecast and the observation files. In case of missing data, the "value" column must contain the number -999.
Also the name of the files is important and must follow the scheme:

- `obs_YYYYMMDDHHmm_`*`NameOfTheVerification`*`.txt`
- `fcs_YYYYMMDDHHmm_`*`NameOfTheVerification`*`.txt`

where YYYY is the year, MM is the month, DD is the day, HH is the hour and mm is the minute. The "*NameOfTheVerification*" must be assigned to the variable `ver_name` in the file `input_fuzzy.nml`
From version *2.0* it is possible to verify three different variables with VAST (NOT starting from LIBSIM): precipitation (mm), total cloud cover (%), wind speed (m/s). Once the input files have been loaded the user should modify the "`input_fuzzy.nml`" configuration file.

4.2.2 The configuration files

Once the input files are placed in the correct directory (`VAST/date/TXT/obs` or `fcs`), all the activities will be driven by the executable file `vast.sh`. So, the software needs to read

in the configuration files all the specifications related to this part of the execution. The two configuration files governing this part of the software are `directories.conf` (read by `vast.sh`) and `input_fuzzy.nml` (read by `read_txt` and `fuzzy_verify`). Figure 10 and Figure 11 show the information required by the configuration file `directories.conf`

```
#DIRECTORIES
#
#COMMON PATH (up to the directory containing VAST)
common_path='/versus/'
#
#If any change has been made to the structure of VAST, than modify with the actual path
#
#INPUT TXT files
dirin_obs='VAST/data/TXT/obs/'
dirin_fcs='/VAST/data/TXT/fcs/'
#Files processed by read_txt.f90 and ready to be used by fuzzy_verify.f90
dirout_obs='VAST/data/OUT/obs/'
dirout_fcs='VAST/data/OUT/fcs/'
#Backup of input and output
dirin_obs_backup='VAST/data/TXT/obs/backup/'
dirin_fcs_backup='VAST/data/TXT/fcs/backup/'
dirout_obs_backup='VAST/data/OUT/obs/backup/'
dirout_fcs_backup='VAST/data/OUT/fcs/backup/'
#R scripts (output of fuzzy_verify.f90)
dirout_r='VAST/data/R/'
#Backup of R scripts
dirout_r_backup='VAST/data/R/backup/'
#Plots
dirout_plots='VAST/data/PLOTS/'
#Fortran codes
dir_src='VAST/src/'
#Executables
dir_bin='VAST/bin/'
#Log files
dir_log='VAST/log/'
#
```

Figure 10: Directories to be configured for the execution of `vast.sh`

```
#NAME OF THE VERIFICATION (string with no gaps)
name="201606_decade1_7km_12h"
#
#FILE NAMES (If the user doesn't have a particular reason for changing these names,
#they should be left in the default version)
#Fortran files
read_txt_f90='read_txt.f90'
fuzzy_f90='fuzzy_verify.f90'
#Executables
read_txt_exe='read_txt'
fuzzy_exe='fuzzy_verify'
#Configuration files
namelist='input_fuzzy.nml'
#Lists
input_obs_list='list_obs.dat'
input_fcs_list='list_fcs.dat'
output_obs_list='list_obs.txt'
output_fcs_list='list_fcs.txt'
r_list='list.r'
#Logs
txt_log='txt.log'
fuzzy_log='fuzzy.log'
txt_stderr='txt_stderr.log'
fuzzy_stderr='fuzzy_stderr.log'
out_txt_log='out_txt.log'
```

Figure 11: File names to be configured for the execution of `vast.sh` (should be left as default with the exception of "`name`")

It is very important for the user to set the variable "`name`" according to the naming of the input files, so that the `vast.sh` executable can chose the correct data to be verified.

Figure 12 and Figure 13 show the information contained in the file `input_fuzzy.nml` and needed by the executable `read_txt`.

```
&common_dir
!Fill in with the actual path (up to the directory containing VAST)
common_path='/versus/'
&end

&directories_txt
!Directories of input and output for txt files
dirin_obs='VAST/data/TXT/obs/'
dirin_fcs='VAST/data/TXT/fcs/'
dirout_obs='VAST/data/OUT/obs/'
dirout_fcs='VAST/data/OUT/fcs/'
dir_log='VAST/log/'
&end
```

Figure 12: Common path and directories to be configured for the execution of `read_txt`. Directories should be left as default.

```
!Names of the list of output of read_txt.f90
!(should be left as default)
&filenames_txt
filelist_obs='list_obs.dat'
filelist_fcs='list_fcs.dat'
&end
```

Figure 13: File names to be configured for the execution of `read_txt` (should be left as default)

Figure 14 and Figure 15 show the information contained in the file `input_fuzzy.nml` and needed by the executable `fuzzy_verify`. The `filenames` group in the `input_fuzzy.nml` namelist also contains information about the model, the grid and the period of the verification. All these last specifications are read as character strings. Since the period, grid and model variables are used to produce the name of the output files, **they must contain no blanks**.

```
&directories_fuzzy
!Directory observation
dirin_obs='VAST/data/OUT/obs/',
!Directory forecast
dirin_fcs='VAST/data/OUT/fcs/',
!Directory output R )
dir_out='VAST/data/R/',
!Directory output plots
dir_plot='VAST/data/PLOTS/',
!Directory log
dir_log='VAST/log/',
!Directory archive
dir_arch='VAST/data/ARCHIVE/'
&end
```

Figure 14: Directories to be configured for the execution of `fuzzy_verify` (should be left as default).

```
&filenames
!List obs (should be left as default)
list_obs='list_obs.txt',
!List fcs (should be left as default)
list_fcs='list_fcs.txt',
!Model
model='COSMO-I7',
!Grid
grid='COSMO-I7'
!Period
period='201606_1-10'
!File list R (should be left as default)
file_list='list.r'
&end
```

Figure 15: File names to be configured for the execution of `fuzzy_verify` (should be left as default).

In Figure 16 the variables relative to the spatial boxes and the thresholds are shown. The user can choose up to ten different thresholds on which the verification will be performed (the thresholds can be different from those shown in the example). If the user needs less thresholds, the value of the unused thresholds must be -999 (like thresh(10) in Figure 16). All the used thresholds must come before the unused ones (i.e. thresh(1)=0.2, thresh(2)=0.5, thresh(3)=1, thresh(4)=2, thresh(5)=5, thresh(6)=10, thresh(7)=-999, thresh(8)=-999, thresh(9)=-999,thresh(10)=-999. **NOT** thresh(1)=0.2, thresh(2)=-999, thresh(3)=1, thresh(4)=2, thresh(5)=-999, thresh(6)=10, thresh(7)=-999, thresh(8)=20 ,thresh(9)=-999, thresh(10)=-999.)
The user has to indicate the resolution of the grid (in Km) by modifying the variable "grid_step". This will be used to produce the plots (this will be not used if the user choses the "grid points" for the space scale in the plots).
From version *2.0* it is possible to verify 3D boxes of observation and forecast by introducing the **time** dimension. The user should set the variable `n_dimension=1` if a standard 2D verification is required (or whenever there are missing timesteps in observation and/or forecast). The user should use a variable greater than 1 to elaborate 3D boxes.
Example1: n_timesteps=3 $\Longrightarrow$ previous, current, following timestep will be used for the verification
Example2: n_timesteps=5 $\Longrightarrow$ previous two, current, following two timesteps.
Another new feature of version *2.0* is the possibility to verify three differnt variables with VAST (NOT using LIBSIM): the variable is chosen by assigning the value '`pre`' (precipitation, mm), '`tcc` (total cloud cover, %)' or '`win`' (wind speed, m/s) to the `variable` line.

```
&dimensions
!Number of windows (1,3,5,9,17,33,65,...,(2^n)+1)
n_windows=5,
!NEW! Number of different timesteps for 3D box
!Example: n_timesteps=3 => previous one, current, following one
!Example: n_timesteps=5 => previous two, current, following two
!ATTENTION: the higher the number, the slower the process
n_timesteps=3,
!Grid resolution (km) used in plot definition
grid_step=8
!Number of valid thresholds
n_thresh=10,
!Thresholds (array of size 10. Fill with -999 if needed)
thresh(1)=0.2,
thresh(2)=0.5,
thresh(3)=1.0,
thresh(4)=1.5,
thresh(5)=2.5,
thresh(6)=5.0,
thresh(7)=7.5,
thresh(8)=10.0,
thresh(9)=15.0,
thresh(10)=20.0,
!Indicate the variable the user wants to verify:
!'pre' for precipitation (mm)
!'tcc' for total cloud cover (%)
!'win' for wind (m/s)
variable='pre'
&end
```

Figure 16: Spatial box and thresholds to be configured for the execution of `fuzzy_verify`

Figure 17 shows all the details needed by the code regarding the actual verification: methods and scores.
For type one, the methods can be different or the same one repeated (up to six times). A score must be associated to each method (ie: `method1(1)='UP'`, `score1(1)='BIAS'`; `method1(2)='MC'`,`score1(2)='FAR'`. This means that the `BIAS` will be calculated using the `Upscaling` method, while the `FAR` will be calculated using the `Minimum Coverage` method). For type two instead, the score and the method are striclty associated (ie: if `method2(1)='FB'`, then `score2(1)='FSS'`; if `method2(2)='PG'`, then `score2(2)='BSS'`; if `method2(3)='PP'`, then `score2(3)='ETSr'`). Type three only contains one score. The number of the used methods for each type must be indicated through the variables `nm1`, `nm2` and `nm3`. The unused methods and scores must be commented (the indication of the methods and scores must follow the same principle of the threshold indication: unused methods and scores must be placed at the end of the array).

```
&methods_scores
!Number of methods of type 1 (max 6)
nm1=5,
!Methods
!Choose between: UP, MC, YN, FZ, JP, ME
method1(1)='UP',
method1(2)='MC',
method1(3)='YN',
method1(4)='FZ',
method1(5)='ME',
!Scores
!Chose between: BIAS, POD, FAR, ETS (if method=UP),
!               POD, FAR, ETS (if method=MC, YN, FZ, JP)
!               HK (if method=ME)
score1(1)='BIAS',
score1(2)='POD',
score1(3)='FAR',
score1(4)='ETS',
score1(5)='HK ',
!Number of methods of type 2 (max 3)
nm2=3,
!Methods
!These methods are strictly coupled with the scores
!The decision is to calculate them or not
!(comment the line and adjust vector numbers)
method2(1)='FB',
method2(2)='PG',
method2(3)='PP',
!Scores
score2(1)='FSS',
score2(2)='BSS',
score2(3)='ETSr',
!Number of methods of type 3
nm3=1,
!Methods
method3(1)='RM',
!Scores
score3(1)='RMSE'
&end
```

Figure 17: Verification methods to be configured for the execution of `fuzzy_verify`

Examples for method 1:

- The user wants to obtain the following scores: bias, pod, far
- The user wants to use the following method for all the scores: upscaling
- Result:
  - method1(1)='UP'
  - method1(2)='UP'
  - method1(3)='UP'
  - !method1(4)=*whatever, commented*
  - !method1(5)=*whatever, commented*
  - score1(1)='BIAS'
  - score1(2)='POD'
  - score1(3)='FAR'
  - !score1(4)=*whatever, comented*
  - !score1(5)=*whatever, commented*
- The user wants to obtain the following scores: bias, pod, far

- The user wants to use the following methods: upscaling for bias, minimum coverage for pod, yes/no for far
- Result:
  - method1(1)='UP'
  - method1(2)='MC'
  - method1(3)='YN'
  - !method1(4)=*whatever, commented*
  - !method1(5)=*whatever, commented*
  - score1(1)='BIAS'
  - score1(2)='POD'
  - score1(3)='FAR'
  - !score1(4)=*whatever, commented*
  - !score1(5)=*whatever, commented*
- The user wants to obtain the following score: far
- The user wants to calculate FAR with three different methods: upscaling, minimum coverage, yes/no
- Result:
  - method1(1)='UP'
  - method1(2)='MC'
  - method1(3)='YN'
  - !method1(4)=*whatever, commented*
  - !method1(5)=*whatever, commented*
  - score1(1)='FAR'
  - score1(2)='FAR'
  - score1(3)='FAR'
  - !score1(4)=*whatever, commented*
  - !score1(5)=*whatever, commented*

Figure 18 shows the last specifications needed by `fuzzy_verify` to chose the correct data (`ver_name`), set the spatial scale in the plots (`km_points`) and to chose the format of the output (`png_pdf`). The `png_pdf` option must be set to **pdf** if the user is working on a VERSUS machine or with old versions of software R. Version *2.0* asks the user to indicate if their R version is newer ('y') or older ('n') than 3.0.

```
&plot_format
!Is your R version >=3.0? (y/n)
r_version='n'
!'png' or 'pdf'
png_pdf='pdf',
!'km' for kilometers, 'gp' for grid points
km_points='gp',
!NAME OF THE VERIFICATION (string with no gaps)
ver_name='?????'
&end
```

Figure 18: Format information for the making of the plots. Used by `fuzzy_verify`

4.2.3 Performance of the verification

Once that the `vast.sh` executable has been launched, the software performs the verification. These are the main steps of the execution:

- The software uploads the information contained in the configuration file `directories.conf`
- Compiles the Fortran code `read_txt.f90`
- Compiles the Fortran code `fuzzy_verify.f90`
- Creates the list of the input TXT files (forecast and observation)
- Runs the executable `read_txt` using the information stored in the namelist `input_fuzzy.nml`
- Produces the half processed DAT files in the directories `VAST/data/OUT/fcs` and `obs`
- Produces the namelist `VAST/src/output_txt.nml` that contains the information related to the data grid dimensions
- Produces the list of the half processed files used by `fuzzy_verify`
- Runs the executable `fuzzy_verify` using the information stored in the namelists `input_fuzzy.nml` and `output_txt.nml` (that produces the R scripts containing the instructions to create the plots)
- Runs the R command that compiles all the scripts containing the instructions to plot the results
- Moves the half processed data files and the R scripts in the backup directories

At the end of the procedure, the user will find all the desired plots in the folder `VAST/data/PLOTS`. If the user needs to perform small changes in the plots (labels, title, colours, legend displacement, etc.), it will be possible to modify the R scripts (located in the `backup` folder inside the `VAST/data/R` directory) without rerunning the verification.

5 Use, recap for LIBSIM users

Input files: The user needs to place the input files in the correct folders. Observation can be bufr or grib, forecast must be grib.

Configuration file: The user needs to configure the file `input_libsim.conf` by completing all the directories' paths, by commenting the unused path for observation input (BUFR if using GRIB and vice versa), by choosing the type of the input files (BUFR or GRIB), by setting the grid parameters (limits and steps), by choosing the cumulation period (`cum`) and by assigning a name to the verification. The configuration file is contained in the directory `VAST/src/`

Namelist: The user needs to complete the directories' paths.

Executable file: The user can now run the executable file `use_libsim.sh` from the `bin` directory by typing:
`./use_libsim.sh`

The executable will now produce the TXT files needed by VAST and place them in the directories `VAST/data/TXT/obs` and `fcs`.

5.1 Use, recap for non LIBSIM users only

Input observation files: The observation files must be txt files with the same characteristics of the ones contained in the folder `vast_install/documentation/data/TXT/obs` and described before. Their naming must contain the date and the name of the verification, as shown before.
Input forecast files: The forecast files must be txt files with the same characteristics of the ones contained in the folder `vast_install/documentation/data/TXT/fcs` and described before. Their naming must contain the date and the name of the verification, as shown before.

The observation and forecast grid must be exactly the same. The coordinates start from the bottom-left corner of the area and proceed by letting the longitude grow first. It is important that the header corresponds exactly to the one shown in the example files.

5.2 Use, recap for both users

Configuration file `directories.conf`: Before the first use of the software, the user will find question marks in the directories common path to be set. The user should fill in the complete path corresponding to the real position of the directory. The information inside this configuration file need to be inserted in the following way (the part within the quotation marks):

```
Variable='my_variable_name'
Directory='my_directory_path'
```

The comment lines need to be inserted in the following way:

```
# Comment
```

Configuration file `input_fuzzy.nml`:

Before the first use of the software the user will find question marks in the directories common path to be set. The user should fill in the complete path corresponding to the real position of the directory. The information inside this configuration file need to be inserted in the following way (the part within the quotation marks):

```
Variable='my_variable_name'
Directory='my_directory_path'
```

The comment lines need to be inserted in the following way:

```
!  Comment line
```

The user needs then to decide the extension of the larger spatial scale needed for the verification to be performed to set the corresponding parameter. This decision must be related to the total dimension of the grid and the resolution of forecast and observation data. The user also needs to decide which type (types) of verification are the most correct to evaluate the situation to be verified.
The program will perform the verification by increasing the extent of the boxes (if the method requires it) starting from the single grid point up to the desired one following the sequence (1, 3, 5, 7, 9, 17, 33, ..). For each box and for each scale the software will calculate the required indexes.

`Dimensions`:

- `n_windows`: Number of window dimensions (up to 10)
- `n_thresh`: Number of valid thresholds (up to 10)
- `thresh`: Value of each threshold (10 values, blanks at the end to be filled with -999)

`Methods_scores`:

- `nm1`: Number of methods of type one (UP, MC, YN, FZ, JP, ME) (up to 6)
- `method1`: List of methods of type one
- `score1`: List of scores to be calculated associated to each method (BIAS, FAR, POD, POFD, ETS, HK)
- `nm2`: Number of methods of type two (FB, PG, PP) (up tp three)
- `method2`: List of methods of type two
- `score2`: List of scores to be calculated associated to each method (FSS, BSS, ETSr)
- `nm3`: Number of methods of type three (RM) (only one)
- `method3`: List of methods of type three
- `score3`: List of scores to be calculated associated to each method (RMSE)

In the namelist files advises are given on how to combine the different methods and scores for type one. Type two and type three are fixed, so for example if the user chooses `method2(1)` **FB**, the `score2(1)` must be **FSS** and so on.

5.3 Perform the verification

Once everything has been set, the user only has to move to the directory associated to the variable "`dir_bin`" in the file `directories.conf` (i.e. "`VAST/bin`") and type:

- `./vast.sh`

The elaboration of the files will start followed by the verification process.

At the end of the verification process all the files contained in the sub directories of TXT and OUT will be moved to their backup folder.

6 Verification methods

Upscaling, anywhere in window, minimum coverage, fuzzy logic, joint probability and multi-event contingency table belong to the verification methods of type 1 (use contingency table and subdivide by thresholds). Brier skill score, fraction skill score and practically perfect hindcast belong to the verification methods of type 2 (don't use contingency table, subdivide by thresholds). Area related root mean square error belongs to the verification methods of type 3 (doesn't use a contingency table and doesn't subdivide by thresholds).

6.1 Upscaling (UP)

- Gets average values in window
- Averaged observation (forecast) $\geq$ threshold $\Rightarrow$ Ob (Fc) array = 1
- Averaged observation (forecast) $<$ threshold $\Rightarrow$ Ob (Fc) array = 0
- Possible scores: BIAS, FAR, POD, POFD, ETS

Hit	Ob = 1	Fc = 1
Miss	Ob = 1	Fc = 0
False Alarm	Ob = 0	Fc = 1
Correct Negative	Ob = 0	Fc = 0

Figure 19: Contingency table for "Upscaling" method

6.2 Anywhere in window (YN)

- Event (threshold exceeding) occurs anywhere in the window
- The portion of window where the event occurs is calculated
- Po (Pf) = Portion observed (Portion forecast)
- Possible scores: FAR, POD, POFD, ETS

Hit	Po $>$ 0	Pf $>$ 0
Miss	Po $>$ 0	Pf = 0
False Alarm	Po = 0	Pf $>$ 0
Correct Negative	Po = 0	Pf = 0

Figure 20: Contingency table for "Anywhere in window" method

6.3 Minimum coverage (MC)

- The portion of window where the event occurs is calculated
- Po (Pf) = Portion observed (Portion forecast)
- Scoring using proportional decision rule (min. cov. = 0.1, Marsigli criterion for 90th percentile)
- Possible scores: FAR, POD, POFD, ETS

Hit	$\text{Po} \geq 0.1 \quad \text{Pf} \geq 0.1$
Miss	$\text{Po} \geq 0.1 \quad \text{Pf} < 0.1$
False Alarm	$\text{Po} < 0.1 \quad \text{Pf} \geq 0.1$
Correct Negative	$\text{Po} < 0.1 \quad \text{Pf} < 0.1$

Figure 21: Contingency table for "Minimum coverage" method

6.4 Fuzzy logic (FZ)

- The portion of window where the event occurs is calculated
- Po (Pf) = Portion observed (Portion forecast)
- Scoring using fuzzy logic
- Possible scores: FAR, POD, POFD, ETS

Hit	$\min(Po, Pf)$
Miss	$\min(Po, (1 - Pf))$
False Alarm	$\min((1 - Po), Pf)$
Correct Negative	$\min((1 - Po), (1 - Pf))$

Figure 22: Contingency table for "Fuzzy logic" method

6.5 Joint probability (JP)

- The portion of window where the event occurs is calculated
- Po (Pf) = Portion observed (Portion forecast)
- Scoring using joint probabilities
- Possible scores: FAR, POD, POFD, ETS

Hit	(Po $*$ Pf)
Miss	(Po $*$ (1-Pf))
False Alarm	((1-Po) $*$ Pf)
Correct Negative	((1-Po) $*$ (1-Pf))

Figure 23: Contingency table for "Joint probability" method

6.6 Multi-event contingency table (ME)

- Square windows are used instead of radii
- Observation occurrence is calculated for each point and for each threshold
- Observation exceeding threshold $\Rightarrow$ Oy = 1
- Observation NOT exceeding threshold $\Rightarrow$ Oy = 0
- The portion of window where the event occurs is calculated only for the forecast
- Pf = Portion forecast
- Hanssen and Kuipers (HK) score is calculated

Hit	Oy = 1	Pf > 0
Miss	Oy = 1	Pf = 0
False Alarm	Oy = 0	Pf > 0
Correct Negative	Oy = 0	Pf = 0

Figure 24: Contingency table for "Multi-event contingency table" method

6.7 Brier Skill Score (Method=PG, Score=BSS)

Pragmatic approach: single observation, neighbourhood forecast. BSS: Measures the improvement of the probabilistic forecast relative to a reference forecast (usually the long-term or sample climatology), thus taking climatological frequency into account. Not strictly proper. Unstable when applied to small data sets: the rarer the event, the larger the number of samples needed.
Answers the question: What is the magnitude of the probability forecast errors?

$$\textit{Brier skill score} - \quad BSS = \frac{BS - BS_{reference}}{0 - BS_{reference}} = 1 - \frac{BS}{BS_{reference}}$$

Figure 25: Brier Skill Score formulation

$$\textit{Brier score} - \quad BS = \frac{1}{N}\sum_{i=1}^{N}(p_i - o_i)^2 = \frac{1}{N}\sum_{k=1}^{K} n_k(p_k - \bar{o}_k)^2 - \frac{1}{N}\sum_{k=1}^{K} n_k(\bar{o}_k - \bar{o})^2 + \bar{o}(1-\bar{o})$$

Figure 26: Brier Score formulation

6.8 Fractions Brier Skill Score (Method=FB, Score=FSS)

The FSS ranges from 0 to 1. A score of 1 is attained for a perfect forecast and a score of 0 indicates no skill. As r expands and the number of grid boxes in the neighbourhood increases, the FSS improves as the observed and model probability fields are smoothed and overlap increases.

$$FSS = 1 - \frac{\frac{1}{N}\sum_N (P_f - P_o)^2}{\frac{1}{N}\left[\sum_N P_f^2 + \sum_N P_o^2\right]}$$

Figure 27: Fraction Skill Score formulation

Where Pf (Po) is the portion of spatial box of the forecast (observation) grid covered by rain exceeding a certain threshold and N is the number of boxes.

6.9 ETS ratio (Method=Practically Perfect Hindcast, Score=ETSr)

Useful forecast resembles one that would have been issued by a forecaster given perfect knowledge of the observations beforehand. Finds maximum ETS for practically perfect hindcast, which is based on the threshold probabilities, and the probability threshold at which it occurs.

6.10 Area related root mean square error (Method= RM, Score=RMSE)

The Root mean square error is calculated over each box. It is calculated for the ordered arrays.

$$RMSE = \sqrt{\left(\frac{1}{n}\right)\sum\left(f_i - o_i\right)^2}$$

Figure 28: Root Mean Square Error formulation

7 Output

7.1 Methods that belong to type 1

A list of the graphics produced by VAST software is given here. The pictures are examples of the possible outputs. The graphics produced for the verification of scores calculated through methods of type one are:

- Scale-intensity

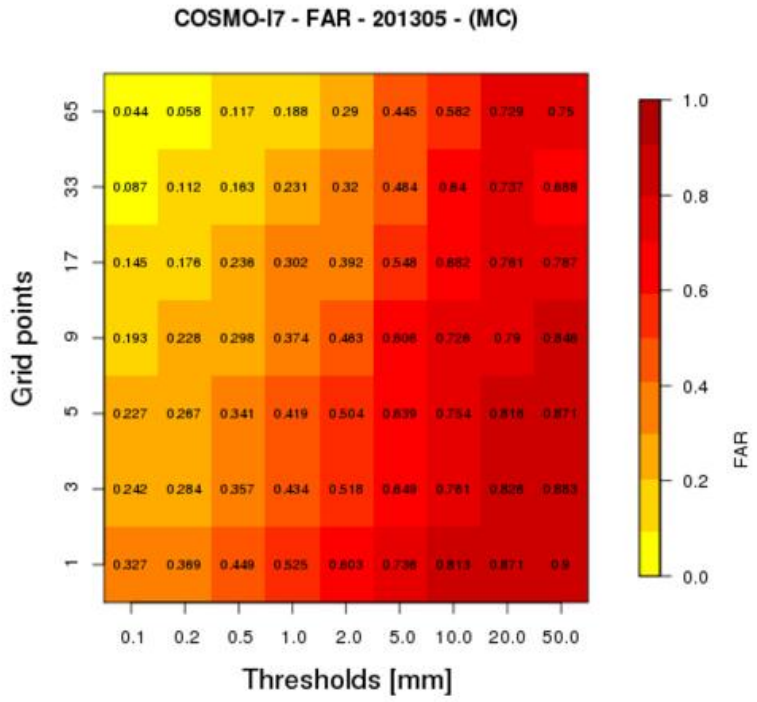

Figure 29: Example of a scale-intensity plot

- Score vs scale

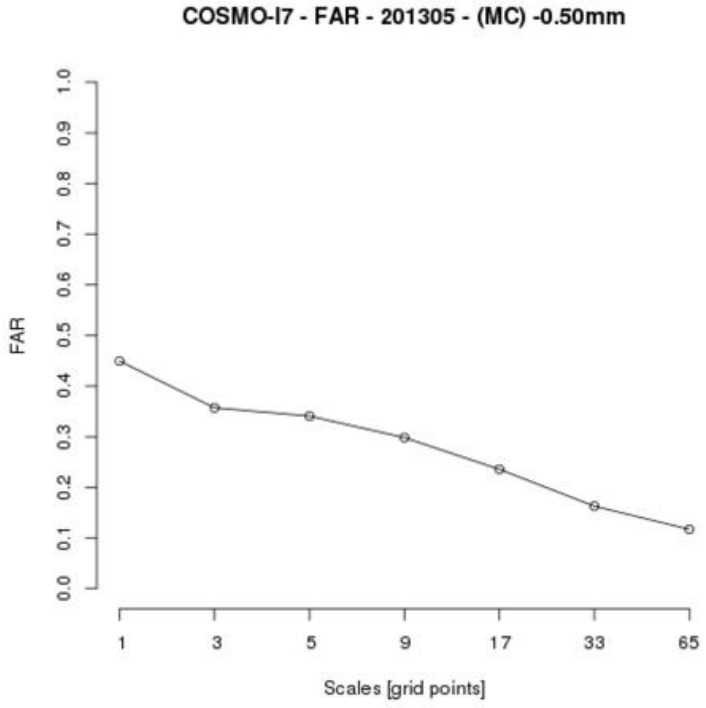

Figure 30: Example of a score vs scale plot

- Score vs scale (all together)

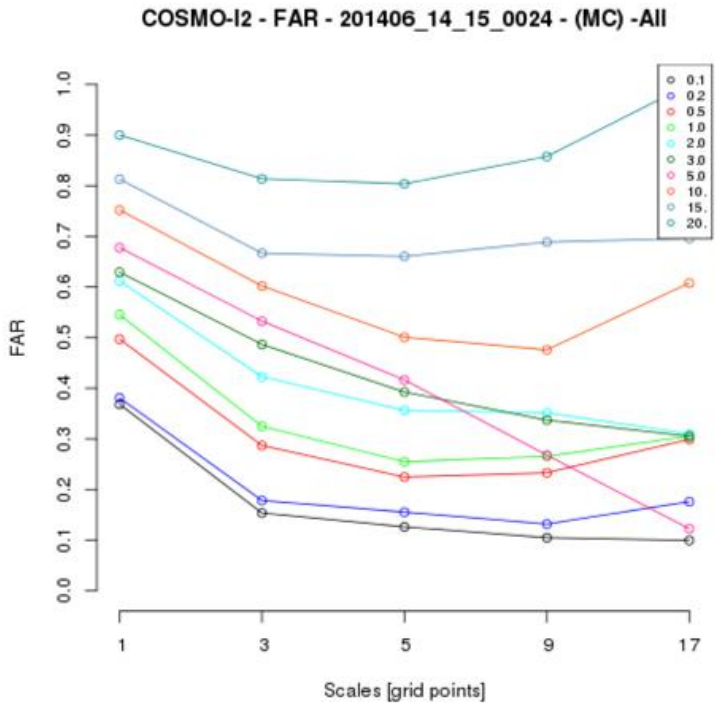

Figure 31: Example of a score vs scale plot (all together)

- Score vs intensity (at minimum scale)

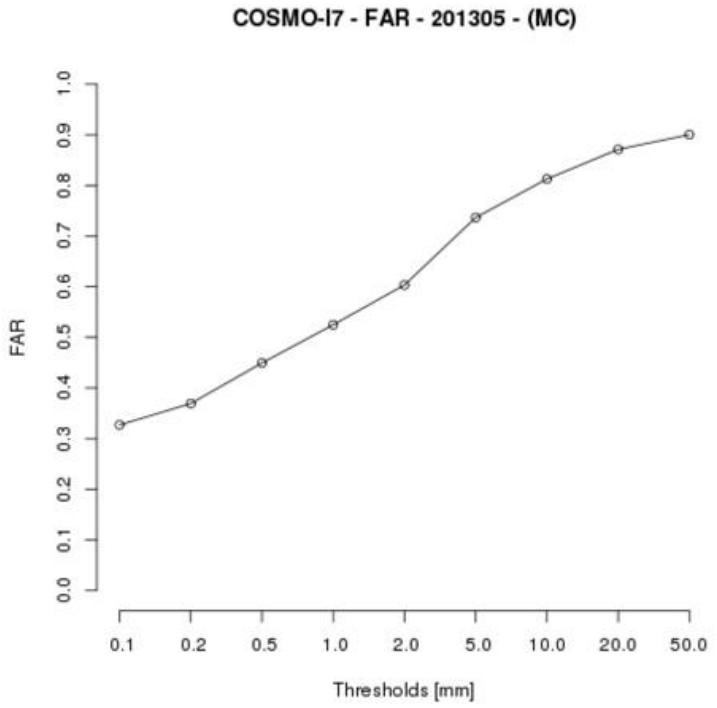

Figure 32: Example of a score vs intensity plot

7.2 Methods that belong to type 2

The graphics produced for the verification of scores calculated through methods of type one are:

- Scale-intensity

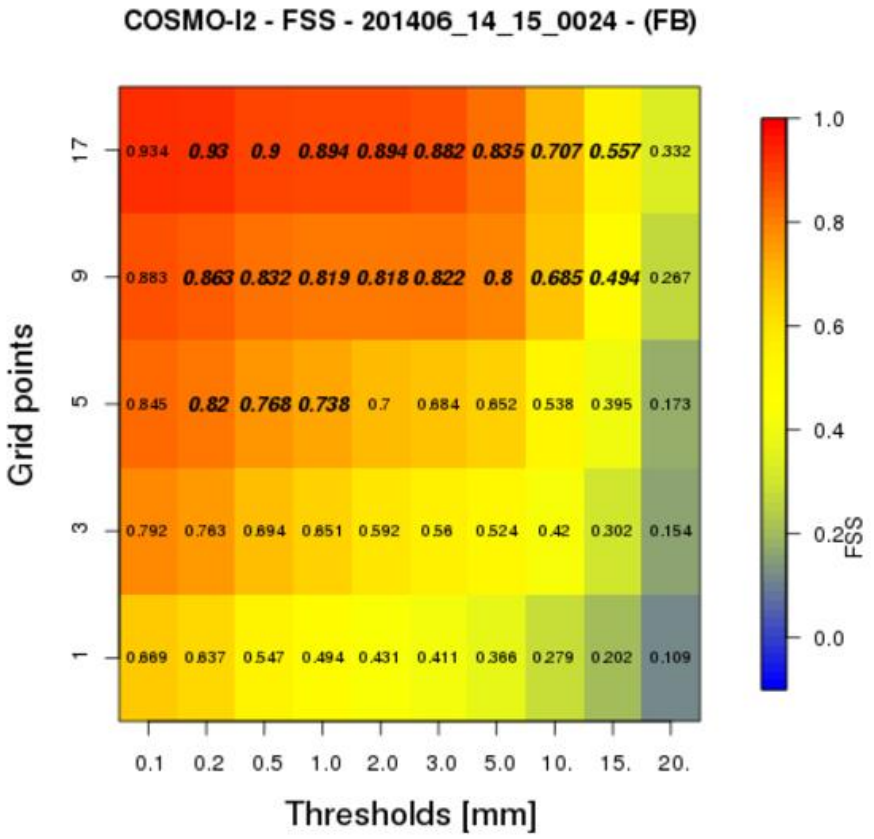

Figure 33: Example of a scale-intensity plot

7.3 Methods that belong to type 3

The graphics produced for the verification of scores calculated through methods of type one are:

- Score vs scale

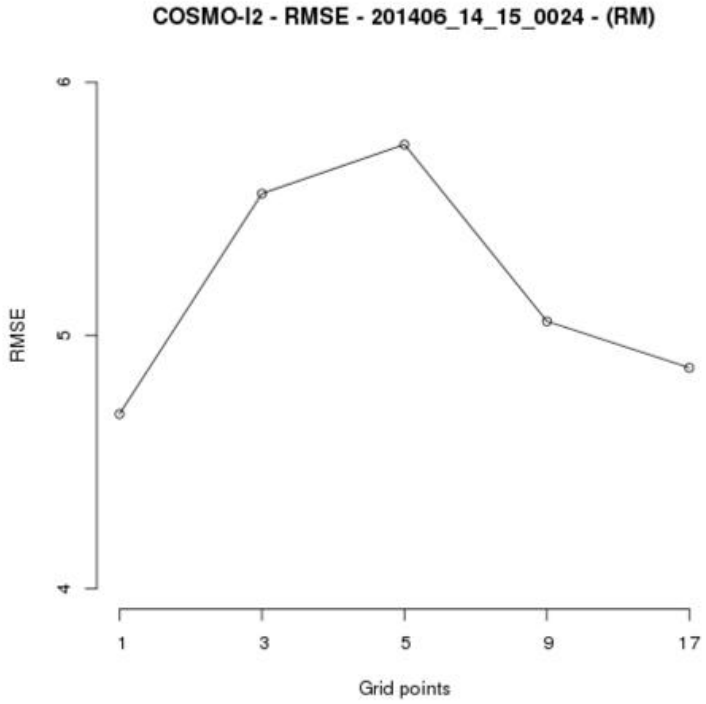

Figure 34: Example of a score vs scale plot

8 Conclusions

The software is designed to give the user the possibility to perform a wide range of spatial verification in a very handy way.
The LIBSIM users can perform their verification starting from GRIB (for both observation and forecast) and BUFR (for observation) files. The non-LIBSIM users must produce txt files according to the given instructions. In the future it might be useful to try to make VAST software independent from LIBSIM and able to use standard input files itself.

9 Acknowledgments

Thanks to all the Working Group 5 for the support.

10 References

Gilleland, E., Ahijevych, D.A., Brown, B.G. and Ebert, E.E., 2010. Verifying forecasts spatially. *Bull. Amer. Met. Soc.* **91**, 1365–1373.

Gilleland, E., Ahijevych, D., Brown, B.G., Casati, B. and Ebert, E.E., 2009. Inter-comparison of spatial verification methods. *Wea. Forecasting* **24**, 1416–1430.

Ahijevych, D.A., Gilleland, E., Brown, B.G. and Ebert, E.E., 2009. Application of spatial verification methods to gridded precipitation forecasts. *Wea. Forecasting* **24**, 1485–1497.

Ebert, E.E., 2009. Neighborhood verification of high resolution precipitation forecasts. *Wea. Forecasting* **24**, 1498–1510.

Casati, B., Wilson, L.J., Stephenson, D.B, Nurmi, P., Ghelli, A., Pocernich, M., Damrath, U., Ebert, E.E., Brown, B.G. and Mason, S., 2008. Forecast verification: current status and future directions. *Meteorol. Appl.*,**15**, 3–18.

Ebert, E.E., 2008. Fuzzy verification of high resolution gridded forecasts: A review and proposed framework. *Meteorol. Appl.*,**15**, 51–64.

Bowler, N.E., Arribas, A., Mylne, K.R., Robertson, K.B. and Beare, S.E., 2008. The MOGREPS short–range ensemble prediction system. *Q. J. Roy. Meteorol. Soc.* **134**, 703–722.

Buizza, R., Hollingsworth, A., Lalaurette, F. and Ghelli, A., 1999. Probabilistic predictions of precipitation using the ECMWF Ensemble Prediction System. *Wea. Forecasting* **14**, 168–189.

Buizza, R., 2005. The ECMWF Ensemble Prediction System. In: *Predictability of weather and climate,* (eds. T. Palmer and R. Hagedorn), Cambridge University Press, Cambridge, 459–488.

List of COSMO Newsletters and Technical Reports

(available for download from the COSMO Website: www.cosmo-model.org)

COSMO Newsletters

No. 1: February 2001.

No. 2: February 2002.

No. 3: February 2003.

No. 4: February 2004.

No. 5: April 2005.

No. 6: July 2006.

No. 7: April 2008; Proceedings from the 8th COSMO General Meeting in Bucharest, 2006.

No. 8: September 2008; Proceedings from the 9th COSMO General Meeting in Athens, 2007.

No. 9: December 2008.

No. 10: March 2010.

No. 11: April 2011.

No. 12: April 2012.

No. 13: April 2013.

No. 14: April 2014.

No. 15: July 2015.

No. 16: July 2016.

No. 17: July 2017.

COSMO Technical Reports

No. 1: Dmitrii Mironov and Matthias Raschendorfer (2001):
Evaluation of Empirical Parameters of the New LM Surface-Layer Parameterization Scheme. Results from Numerical Experiments Including the Soil Moisture Analysis.

No. 2: Reinhold Schrodin and Erdmann Heise (2001):
The Multi-Layer Version of the DWD Soil Model TERRA_LM.

No. 3: Günther Doms (2001):
A Scheme for Monotonic Numerical Diffusion in the LM.

No. 4: Hans-Joachim Herzog, Ursula Schubert, Gerd Vogel, Adelheid Fiedler and Roswitha Kirchner (2002):
LLM ¯ the High-Resolving Nonhydrostatic Simulation Model in the DWD-Project LITFASS.
Part I: Modelling Technique and Simulation Method.

No. 5: Jean-Marie Bettems (2002):
EUCOS Impact Study Using the Limited-Area Non-Hydrostatic NWP Model in Operational Use at MeteoSwiss.

No. 6: Heinz-Werner Bitzer and Jürgen Steppeler (2004):
Documentation of the Z-Coordinate Dynamical Core of LM.

No. 7: Hans-Joachim Herzog, Almut Gassmann (2005):
Lorenz- and Charney-Phillips vertical grid experimentation using a compressible nonhydrostatic toy-model relevant to the fast-mode part of the 'Lokal-Modell'.

No. 8: Chiara Marsigli, Andrea Montani, Tiziana Paccagnella, Davide Sacchetti, André Walser, Marco Arpagaus, Thomas Schumann (2005):
Evaluation of the Performance of the COSMO-LEPS System.

No. 9: Erdmann Heise, Bodo Ritter, Reinhold Schrodin (2006):
Operational Implementation of the Multilayer Soil Model.

No. 10: M.D. Tsyrulnikov (2007):
Is the particle filtering approach appropriate for meso-scale data assimilation ?

No. 11: Dmitrii V. Mironov (2008):
Parameterization of Lakes in Numerical Weather Prediction. Description of a Lake Model.

No. 12: Adriano Raspanti (2009):
COSMO Priority Project "VERification System Unified Survey" (VERSUS): Final Report.

No. 13: Chiara Marsigli (2009):
COSMO Priority Project "Short Range Ensemble Prediction System" (SREPS): Final Report.

No. 14: Michael Baldauf (2009):
COSMO Priority Project "Further Developments of the Runge-Kutta Time Integration Scheme" (RK): Final Report.

No. 15: Silke Dierer (2009):
COSMO Priority Project "Tackle deficiencies in quantitative precipitation forecast" (QPF): Final Report.

No. 16: Pierre Eckert (2009):
COSMO Priority Project "INTERP": Final Report.

No. 17: D. Leuenberger, M. Stoll and A. Roches (2010):
Description of some convective indices implemented in the COSMO model.

No. 18: Daniel Leuenberger (2010):
Statistical analysis of high-resolution COSMO Ensemble forecasts in view of Data Assimilation.

No. 19: A. Montani, D. Cesari, C. Marsigli, T. Paccagnella (2010):
Seven years of activity in the field of mesoscale ensemble forecasting by the COSMO–LEPS system: main achievements and open challenges.

No. 20: A. Roches, O. Fuhrer (2012):
Tracer module in the COSMO model.

No. 21: Michael Baldauf (2013):
A new fast-waves solver for the Runge-Kutta dynamical core.

No. 22: C. Marsigli, T. Diomede, A. Montani, T. Paccagnella, P. Louka, F. Gofa, A. Corigliano (2013):
The CONSENS Priority Project.

No. 23: M. Baldauf, O. Fuhrer, M. J. Kurowski, G. de Morsier, M. Müllner, Z. P. Piotrowski, B. Rosa, P. L. Vitagliano, D. Wójcik, M. Ziemiański (2013):
The COSMO Priority Project 'Conservative Dynamical Core' Final Report.

No. 24: A. K. Miltenberger, A. Roches, S. Pfahl, H. Wernli (2014):
Online Trajectory Module in COSMO: a short user guide.

No. 25: P. Khain, I. Carmona, A. Voudouri, E. Avgoustoglou, J.-M. Bettems, F. Grazzini (2015):
The Proof of the Parameters Calibration Method: CALMO Progress Report.

No. 26: D. Mironov, E. Machulskaya, B. Szintai, M. Raschendorfer, V. Perov, M. Chumakov, E. Avgoustoglou (2015):
The COSMO Priority Project 'UTCS' Final Report.

No. 27: J-M. Bettems (2015):
The COSMO Priority Project 'COLOBOC': Final Report.

No. 28: Ulrich Blahak (2016):
RADAR_MIE_LM and RADAR_MIELIB - Calculation of Radar Reflectivity from Model Output.

No. 29: M. Tsyrulnikov and D. Gayfulin (2016):
A Stochastic Pattern Generator for ensemble applications.

No. 30: D. Mironov and E. Machulskaya (2017):
A Turbulence Kinetic Energy – Scalar Variance Turbulence Parameterization Scheme.

No. 31: P. Khain, I. Carmona, A. Voudouri, E. Avgoustoglou, J.-M. Bettems, F. Grazzini, P. Kaufmann (2017):
CALMO - Progress Report.

No. 32: A. Voudouri, P. Khain, I. Carmona, E. Avgoustoglou, J.M. Bettems, F. Grazzini, O. Bellprat, P. Kaufmann and E. Bucchignani (2017):
Calibration of COSMO Model, Priority Project CALMO Final report.

COSMO Technical Reports

Issues of the COSMO Technical Reports series are published by the *COnsortium for Small-scale MOdelling* at non-regular intervals. COSMO is a European group for numerical weather prediction with participating meteorological services from Germany (DWD, AWGeophys), Greece (HNMS), Italy (USAM, ARPA-SIMC, ARPA Piemonte), Switzerland (MeteoSwiss), Poland (IMGW), Romania (NMA) and Russia (RHM). The general goal is to develop, improve and maintain a non-hydrostatic limited area modelling system to be used for both operational and research applications by the members of COSMO. This system is initially based on the COSMO-Model (previously known as LM) of DWD with its corresponding data assimilation system.

The Technical Reports are intended

- for scientific contributions and a documentation of research activities,
- to present and discuss results obtained from the model system,
- to present and discuss verification results and interpretation methods,
- for a documentation of technical changes to the model system,
- to give an overview of new components of the model system.

The purpose of these reports is to communicate results, changes and progress related to the LM model system relatively fast within the COSMO consortium, and also to inform other NWP groups on our current research activities. In this way the discussion on a specific topic can be stimulated at an early stage. In order to publish a report very soon after the completion of the manuscript, we have decided to omit a thorough reviewing procedure and only a rough check is done by the editors and a third reviewer. We apologize for typographical and other errors or inconsistencies which may still be present.

At present, the Technical Reports are available for download from the COSMO web site (www.cosmo-model.org). If required, the member meteorological centres can produce hard-copies by their own for distribution within their service. All members of the consortium will be informed about new issues by email.

For any comments and questions, please contact the editor:

Massimo Milelli
Massimo.Milelli@arpa.piemonte.it