



Development of Panelification for cloud verification

Virág Lovász (HungaroMet Hungarian Meteorological Service)

In collaboration with : Phillip Scheffknecht (Geosphere Austria)

Technical report about LACE Stay

January 12th – 23rd 2026, GeoSphere Austria, Vienna

Abstract

We have upgraded the Python based Panelification [1] with satellite-based cloud verification capacity. Module additions and adaptations were necessary to extend the previous precipitation-focused functionality of the program. We have also made Panelification more configurable using parameters and settings, facilitating further changes. Some caveats and possible solutions resulting from the previous focus on precipitation are outlined toward the end of this report.

1. Motivation

Cloud verification is highly important for HungaroMet, as it indirectly affects the quality and accuracy of multiple meteorological domains. Such verification is challenging; point-wise verification based on human observations or ceilometer measurements is greatly limited in its representativeness, while new spatial methodologies are just emerging [2]. The use of satellite data is a better option than ceilometers with very limited horizontal coverage, but inconsistencies with forecasts should be minimized [3]. We reviewed the spatial verification methods recommended by NCAR and ACCORD, such as Image Warping, MODE, and SAL, [4] [5] and ultimately decided to implement Panelification (as described in the August 2025 verification report).

Panelification is originally a Python-based development from Phillip Scheffknecht [1], also available to the wider meteorological community in a more simplified R-based Harp version [6]. The power of Panelification lies in its multi-aspect assessment, with a particular focus on spatial verification via Fraction Skill Score. The use of a spatial verification supplemented with conventional scores is recommended in the case of clouds as well as in general [2]. The main challenge when adapting

Panelification for clouds is its original focus on Precipitation. Cloud and rain fields are inherently different: rain has a more or less continuous distribution from zero (or the minimum value in the verification domain) to the maximum value. In contrast, (summed) clouds masks are a field of discrete values [7], such that large fractions of the grid points may have the same value, to the extent of making some of the metrics that work for precipitation meaningless or even undefined.

2. Data and methods

Our developments were carried out in Python and were ultimately merged to the original, Python based repository [1]. The time series analysis represented is also Phillip Scheffknecht's original work [8] and is carried out in Jupyter Notebook.

2.1 SAF product

Our current solution uses the binary **Cloud Mask** (CMA) from the NWC SAF's MTG products as observations, which are available every 15 minutes [7] [9], but the new feature can also use MSG data. The Cloud Mask image is in **Geostationary Satellite Projection (GEOS)**; there is no uniform spatial resolution within the domain, the grid is irregular. Over Europe, this irregularity varies between $\sim 0.035^\circ$ – 0.055° (latitude) and $\sim 0.045^\circ$ – 0.075° (longitude), which is analogous to an effective pixel size range of approximately ~ 4 – 6 km [9] [10]. Since both forecasts tested are available in hourly temporal resolution, the program is designed for per hour timestep verification, and the SAF observations are paired with the nearest (in time) available forecast.

The Cloud Mask files are stored in NetCDF-4 format, as binary image, with a value of 0 representing a cloud-free pixel and a value of 1 representing a cloudy pixel, under the "cma" variable [7]. Our filenames encode the date and time up to minute precision, but this metadata is also available in the files themselves.

2.2 Forecast data

During the test phase, AROME and IFS forecasts available at HungaroMet were tested, using Total Cloud Cover (TCC) as a base field (to be processed further). The 3-hourly AROME Hungary (AROME/HU) forecast has a maximum lead time of 48 hours, they provide hourly outputs for forecasters and 15-minute outputs for the energy sector data (here we work with hourly outputs). IFS forecasts are available 4 times a day and have a maximum lead time of 240 hours. Both forecasts are stored in NetCDF file format, the results are represented in a lat-lon grid, with $0,025^\circ \times 0,025^\circ$ spatial resolution for AROME and $0,1^\circ \times 0,1^\circ$ for IFS. Georeferencing and spatial resolution is only indirectly coded via the NetCDF dimensions: 'maxLatPoints', 'maxLonPoints' and variables 'La1', 'Lo1', (top left coordinates) and 'dx', 'dy' (domain extension). Since we did not terminate the development with the end of the stay, further models have been also tested ever since. Among them, AROME-RUC Hungary (AROME-RUC/HU) is also presented in this report (in section 3.2.4.). AROME-RUC-HU has $0,015^\circ \times 0,015^\circ$ spatial resolution and a maximum lead time of 30 hours, with hourly outputs. Model initializations are available every 3 hours.

2.3 Statistical verification scores in Panelification

We have employed the previously implemented verification scores of the original Panelification and the Harp version [1] [6], which are as follows: **bias**, **Mean Absolute Error (MAE)**, **Root Mean Squared Error (RMSE)**, **Pearson Correlation Coefficient**, **Skillful Spatial Scale** (called d90 in Panelification) [14], and the **Fraction Skill Score** [11].

Bias, also referred to as systematic error, provides information about the tendency of a model to over- or underestimate clouds. MAE is the average magnitude of the error. Root Mean Squared Error penalizes large errors more strongly. These conventional point-wise verification scores supplement spatial verification with additional information [11].

The spatial metric in Panelification is the Fraction Skill Score (**FSS**) [12], which can either be used to verify a single instantaneous field, in which case there is no threshold other than distinguishing between 0 and 1. Alternatively, one can use the sum of multiple hourly cloud mask fields as a proxy for the cloud cover duration, which allows the use of multiple thresholds.

Cloud Cover verification in hourly breakdown is more practical for aviation meteorology, whilst Cloud Cover *Duration* (CCD; during daytime) can aid solar energy forecasts.

In addition to ranking based on the condensed weighted FSS (see Appendix A), we can preserve outcome from all the windows (for neighbourhood areas) and threshold combinations in a Pandas Dataframe. This serves further analysis. Moreover, visual evaluation of the panels can also provide in-depth information. There can be cases of poor FSS performance of different hourly thresholds, but *useful* FSS values when using percentile thresholds. Such a case indicates that the forecast has localized the cloud phenomenon well and the error stems from the duration dimension (or, e.g. intensity such as for precipitation, etc.). This happens because percentiles ensure that the same number of grid points are in the binary fields for observations and models, such that for a sufficiently large window size the score will approach 1. However, this is generally not the case for discrete distributions [6].

Pearson (Pearson Correlation Coefficient) is used as a linear correlation measurement between forecast and observed values, ranging from -1 (perfect negative correlation) to +1 (perfect positive correlation) [11]. The Pearson correlation coefficient reacts to similar features as the percentile-based FSS. Even though it is not explicitly a metric for spatial verification, it is still an indicator for the correctness of the shape (extension) of the verified field. This is because a correlation of 1 only happens if observed and forecast fields have the same shape, except for some constant factor and offset, which is equivalent to having a linear relationship between the two. However, unlike precipitation, cloud forecasts can contain clouds in every grid point or every hour, or no clouds at all, resulting in a constant field. In such cases, the Pearson correlation coefficient is undefined due to a division by 0. Its application to discrete values is questionable, but it can be used for longer CCDs with sufficient spatial variation.

Moreover, Panelification also calculates the Skillful Spatial Scale, (named 'd90' in the program) [4], which is the smallest window size of getting useful ($0.5 <$) FSSs [5]. This is done in **scoring.py** [with the function **fss_d90**] and follows the formula of Roberts & Lean [13] and Skok & Roberts [14]. However, the 90th percentile does not always have meaningful value. If, for instance, more than 10% of the forecast is cloudy at all times, the 90th percentile and the maximum have the same value. We do not encounter the same problem with precipitation as rain fields are more heterogenous and do not contain large areas with the same value. However, the opposite problem, i.e. less than 10% of the field containing rain, can still arise, in which case the 90th percentile equals to zero.

After the program runs, scores are written to csv files. Average and quantile values in them can be an indicator for how representative the d90 is for that given event. However, we welcome suggestions for specific threshold values in the matter.

Time series analysis happens outside of Panelification, and uses the obtained FSS scores, stored in binary Python pickle files [15]. The Fraction Skill Scores should not be averaged for multiple events, instead it should be aggregated over the selected time period to ensure a consistent normalization, thereby avoiding distortions in the denominator that can arise from temporal variability in event occurrence [12].

These verification scores, visualisation panels, and the aggregation analysis are all part of the original Panelification program. Our contribution during the stay was to add data reading and preprocessing for satellite data and generally for the HungaroMet file format, and to add or adapt functions for cloud verification and visualization. Other than that, the two cloud-specific verification parameters for FSS are:

- Cloud Cover (dependent on window size)
- Cloud Cover Duration (dependent on threshold (of duration) and window size)

Logging was refined to help the user with messages in case of any error.

3. Results

3.1 Newly added modules

Since image acquisition with the MTG Seviri sensor takes place some time after the nominal time of scanning [9], SAF images have a systematic offset compared to the forecast timesteps. The temporal resolution (15 mins) of available SAF images has also a mismatch to our hourly forecast fields. Embedded functions from **read_SAF.py** handle the necessary data checking, reading, and matching observation to forecast, minimizing the lag, but the current version is optimized for the longitudinal position of Hungary and Austria. Since this approximate time offset may differ between countries (depending on their location and when exactly the region is scanned [9]), the corresponding time offset for any observation data in **check_path** in **read_SAF.py** should be adapted accordingly. This solution assumes that the names of satellite images encode image acquisition time and not the nominal time.

Functions within the newly added **read_SAF.py** module:

Check_paths:

Path of the SAF images is defined here. The function decodes the data string from the filenames, and checks availability according to the date defined in the commandline. This function also handles the 5-minute offset between the round UTCs of the forecasts and the SAF images, and selects the relevant files accordingly.

lonlat_from_nwc_geos:

Produces the lat, lon coordinates from the GEOS projected x,y coordinates of SAF images. The function is embedded into the `read_saf_data` function.

read_saf_data:

Reads the SAF data from the selected netCDFs under the variable 'cma'. Reading is done

according to the start and end arguments (file identification), as well as 'duration' arguments also received through commandline arguments. Reads instantaneous fields if duration is 1, otherwise sums them up within the defined verification window. Returns lat, lon, and SAF data arrays. The function is embedded into read_SAF_obs.

read_SAF_obs :

Prepares the Python dictionary [16] named 'datalist', containing configuration (SAF), type (obs), name (SAF cma) lat, lon coordinates and the processed SAF cma data (array) itself, in key-value pairs.

3.2 Modified modules

3.2.1. The use of command line arguments

In Panelification, the basic verification settings can be specified using the command line options. These options (can be automated with a loop in a bash-scripted .sh file) initialize and control multiple aspects of the verification from the top. The use of a few options worth mentioning as a contrast to the original precipitation-based Panelification:

-p Verification parameter. Should always be set as **cma** for binary cloudmask verification, nearly all the developments described in this report are activated through this argument. Default setting is left unchanged ('precip').

-s Start date and time, mandatory option. Previously, the original main code was optimized for precipitation, where an interval is inseparable from the amount variable, thus setting the start time one hour before the timestep to be verified was pre-requisites. In our current update, **when setting the verification parameter to cma** it is no longer expected, because **SAF_reading** functions were configured to fit the context. Start date should be defined as a date string, such as YYYYMMDDHH.

-d Duration, mandatory option. The duration is key and sets the cloud verification logic of Panelification.

A value of one enables instantaneous (hourly) cloud verification, whilst higher values the verification of Cloud Cover Duration, by accumulating the mask fields over the length of duration. The specific length of Cloud Cover Duration is elective, we have tested and we recommend 12 hours (daytime) or 24 hours. It is required to set the number in **integer**. We've left the fault unchanged (1).

Newly added command line options:

--d_windows Window levels, optional. Users can, if needed, define window levels of choice. Note that cloud-specific window levels in kilometers (5,10,20,50,100) are now also conditioned to **args.parameter** cma, and by that, performed automatically. But even this can be overruled for testing purposes or other individual causes, with **d_windows options**. Users may enter desired level(s) directly in the commandline, in the form of **integers**, separated by space.

Other newly added /modified functionalities in main.py:

If the verification parameter is cma, the `read_SAF_obs` function from the `read_SAF` module is used.

3.2.2. Newly modified functions

`custom_experiments_test.py:`

The module has been complemented to handle the IFS and AROME forecasts, specifically for the Total Cloud Cover variable, as discussed earlier. Their defining templates were added to individual Python dictionaries [16] in `custom_experiment_test.py`, containing path template, variable name, init interval, max lead time in key-value pairs. Note that you should set the max_lead time 1 hour longer than the actual maximum, to comply with the cloud specific modifications set in start option (3.2.1).

`io_netcdf:`

`read_HungaroMet_netcdf:`

The module `io_netcdf` reads and preprocesses forecast TCC fields in a flexible way, where the resolution of the model field is not hard-coded but automatically detected from metadata included in the netCDF files.

Takes the path of the folder and netCDF variable names as function arguments from templates defined in `custom_experiments`. Perform thresholding on the field with the minimum set to 0.2, for the original Total Cloud Cover field to be converted to a binary Cloud Mask. We choose this threshold to perform the conversion following the recommendation of Frogner et al [17]. The function reads the “La1”, “Lo1”, “Dx”, “Dy” variables and the “maxLatPoints”, “maxLonPoints” dimensions from the netCDF files and consequently produces the full latitude and longitude grid from them.

Same as in `read_SAF`, data reading is depending on `args.duration`, if duration is greater than 1, sums up the (at this point, already binary) grid point values, otherwise reads the instantaneous binary value. The function returns the full lat, lon arrays and cma data.

`io_main:`

`ModelConfiguration` (class):

`get_data` function has been modified: if the model experiment name contains “hun”, the program uses `read_HungaroMet_netcdf` for reading the forecast fields.

regions.py:

The boundary box of Hungary has been added to the existing geographical regions, defined in a Python dictionary. Currently, the only active subdomain corresponds to the same (to entire Hungary). Note that regions.py defines regions, transforms both observations and forecast fields into a common projection grid and then into a cutout (corresponding to the subdomain), but also default and region-specific thresholds. They were originally meant for precipitation verification, subject to a minimum amount of rainfall/hail/snow before the panels can be drawn at all.

These need to be overridden, as they are not applicable for binary cloud masks. We do that during cma verification by activating the `forcescore` and `forceddraw` options in the command line (which were already existing arguments in the original version of Panelification).

parameter_settings.py (Functions edited):

get_fss_threshold(args) :

When the verification parameter is cma, it generates a list of thresholds, in range of the duration (e.g.: for 24 hour Cloud Cover Duration, produces 24 different thresholds, whilst for Cloud Cover, the standalone threshold is 0.5, given that instantaneous field is binary).

title_parts(args) :

This function takes the commandline arguments as arguments. Has been supplemented: when args.parameter (from commandline) is cma, title is subject to the value of args.duration, leading to either Cloud Cover or Cloud Cover Duration, as previously discussed in 2.2. Any number of duration greater than 1 leads to Cloud Cover Duration, as opposed to instantaneous Cloud Cover verification.

Colorbar_label(args) :

Colorbar labels for both Cloud Cover and Cloud duration are subject to the same parameter and duration condition as in title_parts .

3.2.3. Panels

The figures (Fig. 1, Fig.2) below demonstrate the output of the cloud Panelification which basically has two possible running modes, already explained in 2.2. (of course for CCD, the exact duration can be still fine-tuned as desired). Note that duration is always 1 hour with Cloud Cover mode, as we are talking about instantaneous fields. Model labels contain the time of initialization and (in parenthesis) the ranking.

The golden and silver colors (also bronze, if there are enough models in comparison) also illustrate the rankings of FSS performance. Keep in mind that the FSS ranking on the panels refers to the ***cumulated-weighted FSS***, always valid for that given weather event alone, and thus this type of FSS cannot be normalized over longer periods, nor to be aggregated. The same color scheme is also used to mark point-based verification scores and supplement the specific FSS values. (Fig.2, bottom right AROME with 00 UTC initialization, first threshold)[6].

In case of `args.duration > 1`, Panelification verifies “Cloud Cover Duration” instead, where the threshold axis refers to hours of duration of Cloud Cover. (Fig.2).

Clearly, with instantaneous Cloud Cover verification, the binary image data cannot be organised into real percentile categories, which is also evident in the image (Figure 1.). For that reason percentiles could be switched off (in `scores.py`) when verifying instantaneous fields. Note that the time step to which the predictions apply is the first time step of the interval shown in the title (Figure 1). The interval in the title is just a remnant of the program's orientation to precipitation fields (see 3.2.1). In a 12 hours Cloud Cover Duration verification (Figure 2), percentiles do serve with some value but use it cautiously as it is still not a continuous field. In the first example (Figure 1) the IFS model outperforming AROME in all regards is pretty straightforward.

Figure 1. was produced via the following commandline arguments:

```
python main.py --loglevel debug --parameter cma --
verif_dataset SAF_cma --region Hungary --
zoom_to_subdomain True -s 2026061120 -d 1 --
custom_experiments arome_hun ecmwf_hun -l 24 --
custom_experiment_file custom_experiments_test --
save_full_fss True --name "cm_test" --forcedraw True --
forcescore True
```

In this case, in order to limit the forecast init to 00 UTC exclusively, as seen on Fig.1., the `init_intervals` in `custom_experiment` file was also set to 24 (hours).

Cloud Cover over Hungary (Default)
20260611 20 - 20260611 21 UTC (1 hours)

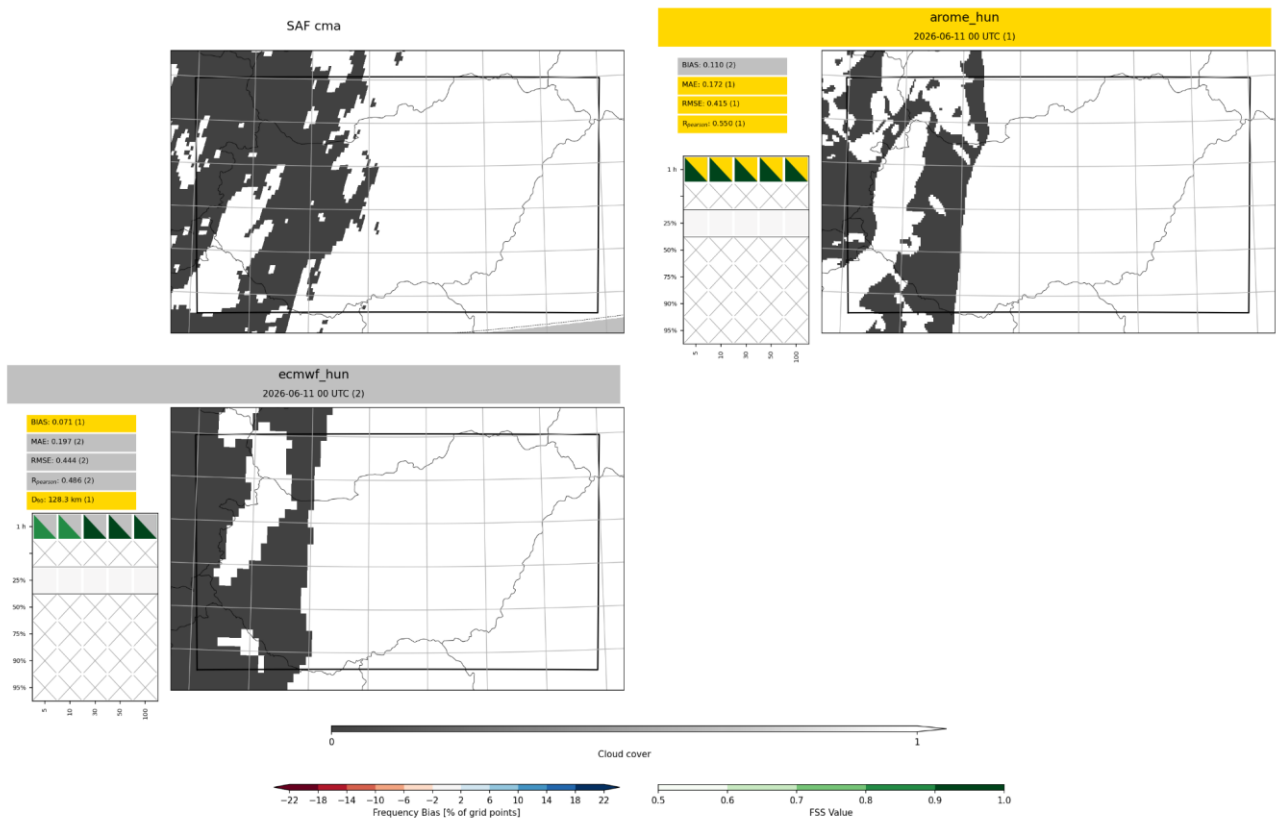


Fig.1. Verification of (instantaneous) Cloud Cover for AROME/HU and IFS model runs initialized at **00 UTC** , valid at **2026 06.11 20 UTC**.

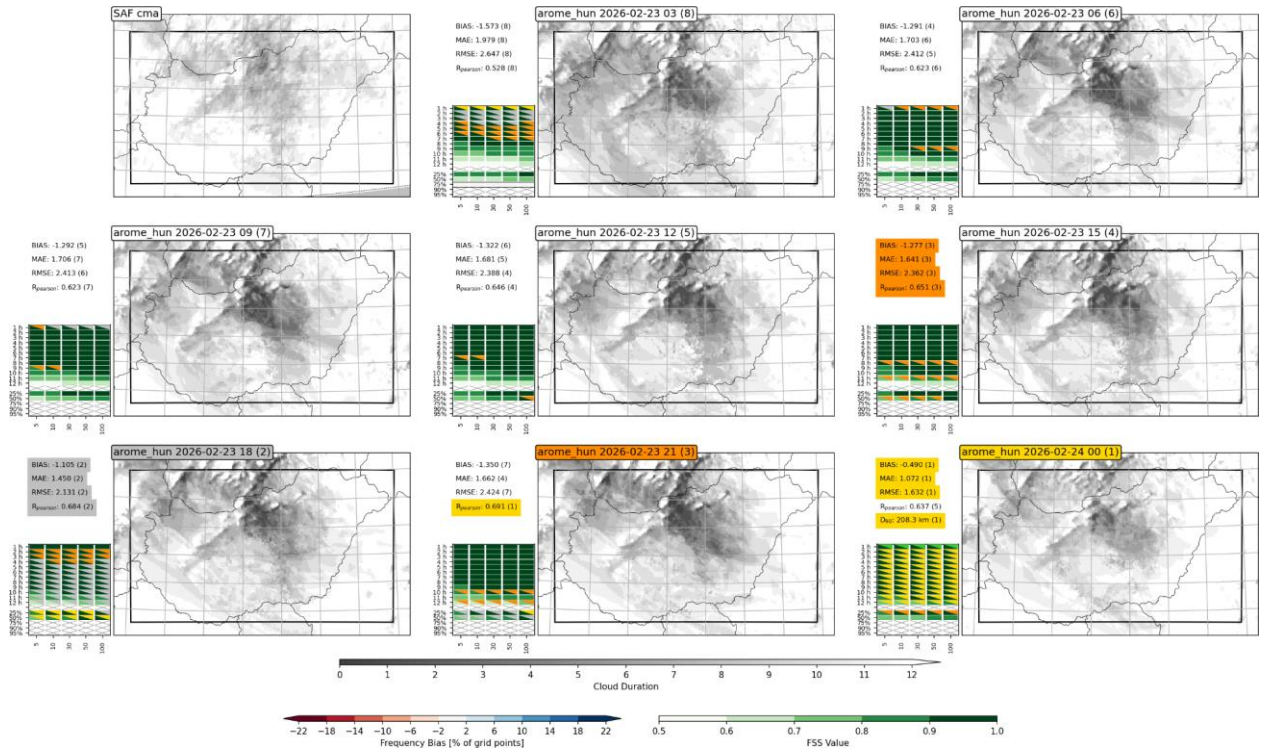


Fig.2 Verification of Cloud Cover Duration, (12 h) of multiple AROME/HU runs, for 12 h (01 UTC - 12 UTC) Cloud Cover Duration on 24 February 2026.

In Figure 2, all AROME model runs have a negative bias, meaning that they all predicted shorter Cloud Cover Duration, the 00 UTC run has the least evident negative bias, and also the most consistently high FSS among all thresholds. The runs with gold and silver FSS results are consistent with the point scores that received gold and silver rankings, except for the *third* place (bronze): the panel suggests that 15 UTC initialization has much worse Fraction Skill Score at the 12-hour thresholds, making a slight difference in overall ranking to the advantage of the 21 UTC run. Figure 2.

Panel (Fig.2) was produced via the following command line arguments:

```
python main.py --loglevel debug --parameter cma --verif_dataset
SAF_cma --region Hungary --zoom_to_subdomain True -s 2026022401
-d 12 --custom_experiments arome_hun -l 24 --
custom_experiment_file custom_experiments_test --save_full_fss
True --name "cm_test" --forceddraw True --forcescore True
```

3.2.4. Further analysis of FSS

At this point, aggregation is handled outside of Panelification [8]. But the necessary numerator and denominator values to aggregate FSS over a chosen time length are calculated and returned within the program and can be saved to files for later use with the command line argument. `save_full_fss`, as part of the original program.

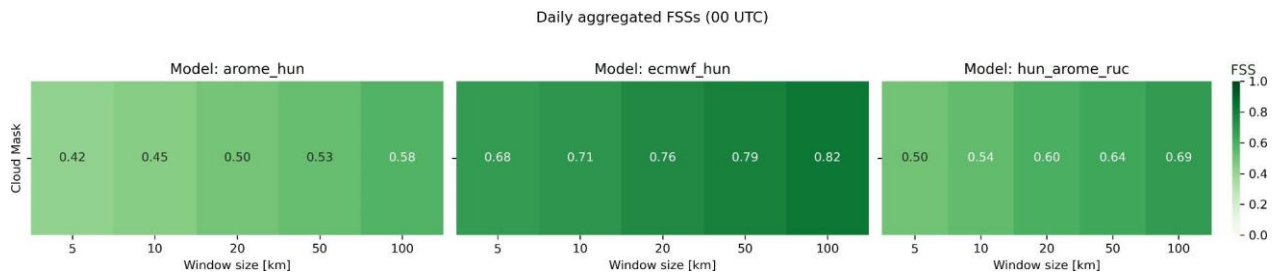
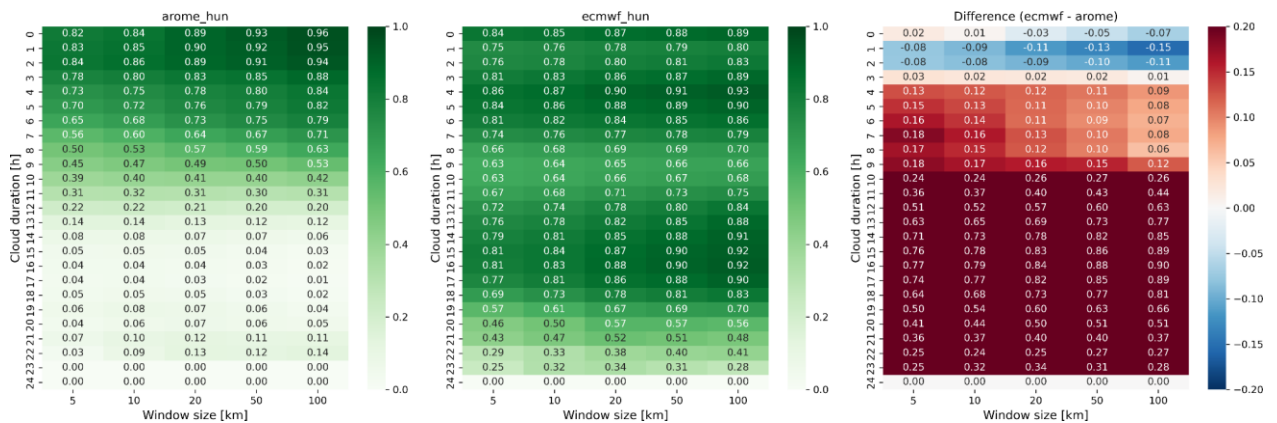


Figure 3. Daily aggregated FSSs of hourly Cloud Cover verification. Models verified against SAF Cloud Mask .

One way of using the saved data when evaluating forecasts over a longer time is to visualize the aggregated Fraction Skill Score, such as in Fig 3. Aggregation period may encompass a day, a week, a month, etc. Figure 3. demonstrates the FSS outcome of AROME/HU, IFS (“ecmwf_hun”), and AROME-RUC/HU, on different window sizes, where IFS also demonstrates the most favourable FSS overall and also on the smallest window size in this particular case. If Panelification was run in the 24 h Cloud Cover *Duration* mode, it’s a valuable time period on its own (Figure 4.), but aggregation can still be extended to week or month.



4. Discussion, Caveats, and further plans

4.1. Discussion and caveats

'**Accumulation**' in the context of hourly cloud masks is more of a count of cloudy moments at full hours than a real accumulation. It assumes each instantaneous value is representative for one full hour and ignores the intermittent (sub-hourly) state of cloud cover. Exploiting the full temporal resolution (15 minutes) of SAF products could enable a more representative CCD evaluation, but this would also require model output every 15 minutes. Sub-hourly verification is not implemented in Panelification at this point and would require some further changes.

One question arises about the representativeness of the 90th percentile during an event where the majority of the field is covered with the maximum possible value for clouds. For precipitation, the D90 [5] should always approach 1 as the window size approaches $2N$, where N is the grid point count along the longer dimension of the verification domain. However, for cloud fields this is not necessarily the case, which can result in a frequency bias for a field that was explicitly designed to remove such a bias. For instance, comparing a model field where 20% of the points have the maximum possible value (cloud coverage every hour) to an observation field where 40% of the points have the maximum possible value will result in calculating the FSS for two fields with a frequency bias of 100% (twice as many points above the threshold in the observations than in the model). In theory, this can lead to the value approaching some limit less than 0.5, which is not possible in continuous fields apart from extremely sparse events.

4.2. Further plans

From a practical perspective, the further beneficiaries of such a cloud verification development are the solar energy sector and aviation meteorology. Sector-specific geographical regions can be defined to serve them. Such requested sub-regions should be added under the country-specific dictionaries in `regions.py`. The delineation precision of countries / regions and the verification itself are limited to rectangular bounding boxes of the defined areas. In the future, this might be expanded to delineate even irregular subregions defined in shape files, like countries, counties or fully custom irregularly shaped regions. However, this requires a modification to the FSS calculation functions to properly handle masked or missing values.

Furthermore, additional SAF products for Cloud verification could be used in Panelification, like for instance Cloud Type (CT), with the verification as follows:

- Running **hourly cloud verification** as previously described, but using the vertical categories in SAF Cloud Type to serve as thresholds for Fraction Skill Score. This would require some preprocessing of forecast and observation data.
- Duration is still the required threshold measure for Cloud Cover Duration. In the CT case, accumulation over time would not yield meaningful results any more, unless verification only covers one Cloud Type field. CT would be used for hourly verification exclusively.

At the moment, further developments are in progress for extending the time series analysis highlighted in Section 3.2. Operative implementation of Panelification for clouds (via SAF cloud products) is on schedule at HungaroMet.

5. Conclusion

We successfully fine-tuned Panelification to read, process, and match SAF data with cloud data extracted from HungaroMet forecast file formats, which was the primary objective outlined in the proposal for the two-week stay. We have enabled the verification of Cloud Cover and Cloud Cover Duration using the demonstrated observation–simulation combinations.

In addition, we made the program’s workflow more flexible and introduced numerous logging messages to support users. Finally, we identified several concerns related to the emerging field of cloud verification and welcome feedback and suggestions from the entire LACE community.

Further notes

Panelification might also challenge the narratives of AI-based forecast models supremacy, AI models are typically trained on point wise error calculation, therefore it is only natural they perform well on point wise verification [18]. Verifying AI-based forecasts with the multi-aspect Panelification, makes it easier to uncover hidden limitations.

Acknowledgements

The presented work is done as a joint cooperation with Phillip Scheffknecht (GeoSphere Austria). We acknowledge casual (supervised) AI assistance in coding.

Appendix A: The FSS-Based Ranking Score (Condensed Weighted FSS)

The overall ranking of the verified simulations within each panel is based on the performance of each simulation for all the FSS windows and thresholds. First, the FSS values are transformed using

$$S_{w,t} = 2(FSS_{w,t} - \frac{1}{2}),$$

Where S represents how far a value is above the “skillful threshold”, w is the window size, and t is the threshold used to calculate the corresponding FSS value. This maps a perfect forecast to 1 and a forecast at the threshold of not being skillful to 0. These values are then summed using the following weights:

$$FSS_{cw} = \sum_w \sum_t S_{w,t} \frac{2w_{max}}{w+w_{max}} \frac{t+t_{max}}{t_{max}}.$$

This means that small spatial scales have approximately twice the weight of large spatial scales, and high thresholds have approximately twice the weight of low thresholds. While the weighting by spatial scale is valid for any forecast quantity, the weighting by threshold is due to the fact that high precipitation values are both more important and more difficult to predict. This is not necessarily the case for cloud cover, but the weighting was not changed for cloud verification.

References:

- [1] P. Scheffknecht [pscheffknecht-geosphere] Panelification. [Currently a private repository: <https://github.com/pscheffknecht-geosphere/panelification>].
- [2] E. K. Dolinar, J.E. Nachamkin. Perspectives on Cloud Prediction, Postprocessing, and Verification for DoD Applications. Bulletin of the American Meteorological Society, (2024) E1034–E1041 <https://doi.org/10.1175/BAMS-D-24-0077.1>
- [3] M. Costa-Surós, J. Calbó, J.A. González, J. Martin-Vide. Behavior of cloud base height from ceilometer measurements. Atmospheric Research. (2013). Volume 127, June 2013, Pages 64-76
- [4] R. Crocker, M. Mittermaier. Exploratory use of a satellite cloud mask to verify NWP models. Meteorological Applications. (2013). 20: 197–205 DOI: 10.1002/met.1384
- [5] B. Brown, T. Jensen, H., J., Gotway, K., Newman, E. Gilleland, T. Fowlerand, R. Bullock. Methods for Evaluation of Cloud Predictions. 7th International Verification Methods Workshop Berlin, Germany, (2017), 10th of May
- [6] P. Schmederer, C. Peralta, B. Fabrizio. Developments done during Polly Schmederer's ACCORD VS at DMI (June 2024). RC LACE stay report. (2024) Available at: <https://www.rlace.eu/applications-and-verification>
- [7] NWC SAF. Cloud Mask. (last checked: 2026.04.11). https://www.nwcsaf.org/cma_description
- [8] P.Scheffknecht. Panelification, period evaluation. (2026) Available at: https://github.com/pscheffknecht-geosphere/panelification_period_eval
- [9] EUMETSAT MSG Level 1.5 Image Data Format Description. (2017) 12, 9 pages. Available at: https://user.eumetsat.int/s3/eup-strapi-media/pdf_ten_05105_msg_img_data_e7c8b315e6.pdf
- [10] V. S. Gabriel, L. Bugliaro, M. Montag, S. Ries, Z. Wang., K. Widmaier, M. Arico, S. Unterstrasser, J. Mayer, d. Menekay, A. Marsing, E. de la Torre Castro, L. Megill, M. Scheibe, and Ch. Voigt. A manually labeled contrail dataset from MSG/SEVIRI. Earth System Science Data (2026) <https://doi.org/10.5194/essd-18-2397-2026>
- [11] Development Testbed Central. Verification Statistics for continuous Forecasts. Last accessed: 2026.06.11 Available at: <https://dtcenter.org/metplus-practical-session-guide-version-5-0/basic-verification-statistics-review/continuous-forecasts/verification-statistics-continuous-forecasts>
- [12] M. P. Mittermaier. A meta-analysis of fraction skill score: A limiting case and implications for aggregation. Monthly Weather Review, (2021) 149 (11), 3491–3504.

<https://doi.org/10.1175/MWR-D-18-0106.1>

- [13] Roberts, N. M., and H. W. Lean. Scale-selective verification of rainfall accumulations from high-resolution forecasts of convective events. *Mon. Wea. Rev.*, (2008) 136, 78–97, <https://doi.org/10.1175/2007MWR2123.1>.
- [14] G. Skok, N. Roberts. Estimating the displacement in precipitation forecasts using the Fractions Skill Score. *Q.J.R. Meteorol. Soc.*, (2018) 144: 414-425. <https://doi.org/10.1002/qj.3212>
- [15] Python (2001). Pickle - Python object serialization. <https://docs.python.org/3/library/pickle.html>
- [16] Python (last accessed: 2026) Python Dictionaries. https://www.w3schools.com/python/python_dictionaries.asp
- [17] I.E., Frogner, U. Andrae, P. Ollinhao, A. Hally, K. Hamalainen, J. Kauhanen, K-I. Ivarsson, D. Yazgi . Model uncertainty representation in a convection-permitting ensemble - SPP and SPPT in HarmonEPS. *Monthly Weather Review*. (2022). 66 pages. DOI 10.1175/MWR-D-21-0099.1.
- [18] M. Bonavita. On Some Limitations of Current Machine Learning Weather Prediction Models. *Geophysical Research Letters* (2024) 10 pages. <https://agupubs.onlinelibrary.wiley.com/doi/10.1029/2023GL107377>