Hands-on session “users”

Introduction

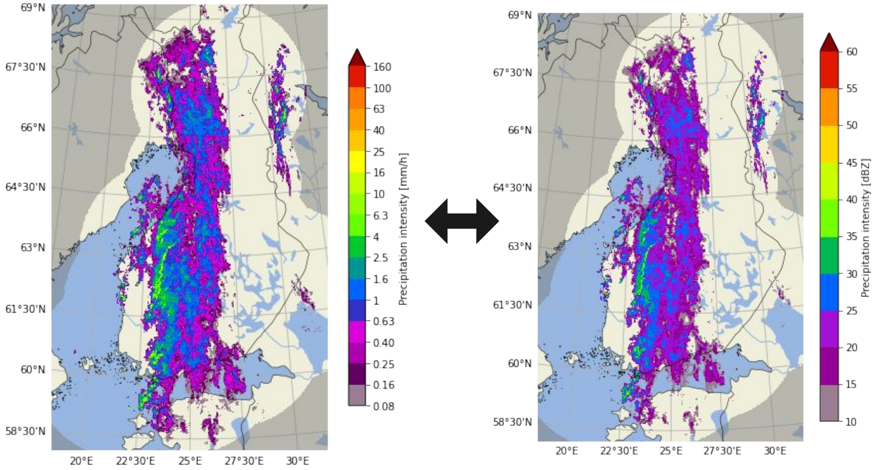
Outline

- The purpose of this session is to give a “hands-on” experience of using pysteps
- The session is divided into 5 exercise blocks + bonus exercises
- In the following slides, we give an introduction to the basic pysteps workflow

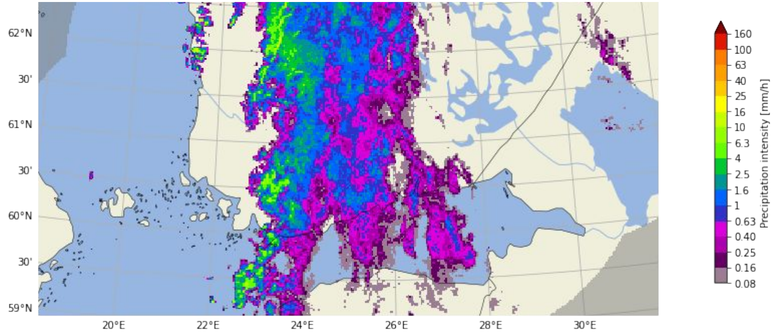
Input data

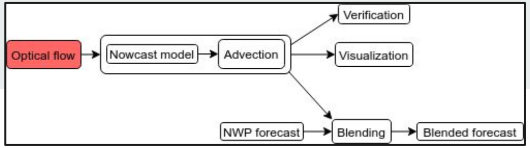
- pysteps uses radar composites as the main input data source
- Reading input data from various source has been implemented in the **io.importers** module
- To implement your own importers, you can use cookiecutter: https://pysteps.readthedocs.io/en/stable/developer_guide/importer_plugins.html
- The **utils** module contains different methods for transforming and clipping the input data

Conversions between units



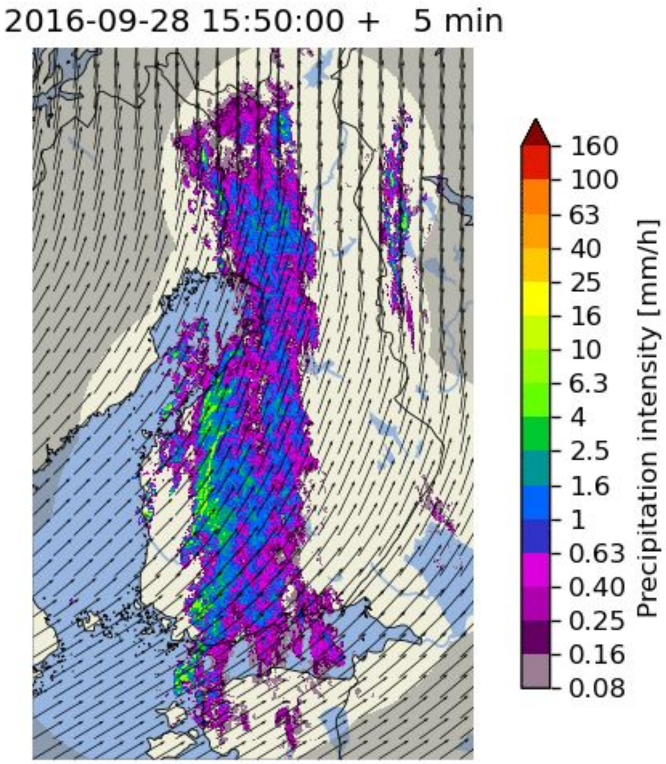
Clipping

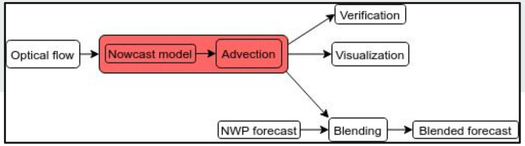




Optical Flow and Extrapolation

- Advection (optical flow & extrapolation) is the key component of all pysteps nowcasting methods
- All methods are based on the "Lagrangian persistence" nowcast shown on the right
- Three different types of optical flow methods have been implemented in the **motion** module:
  - feature tracking: Lucas-Kanade
  - variational: VET and Proesmans
  - spectral: DARTS
- For advection, pysteps implements the backward semi-Lagrangian scheme in the **extrapolation** module

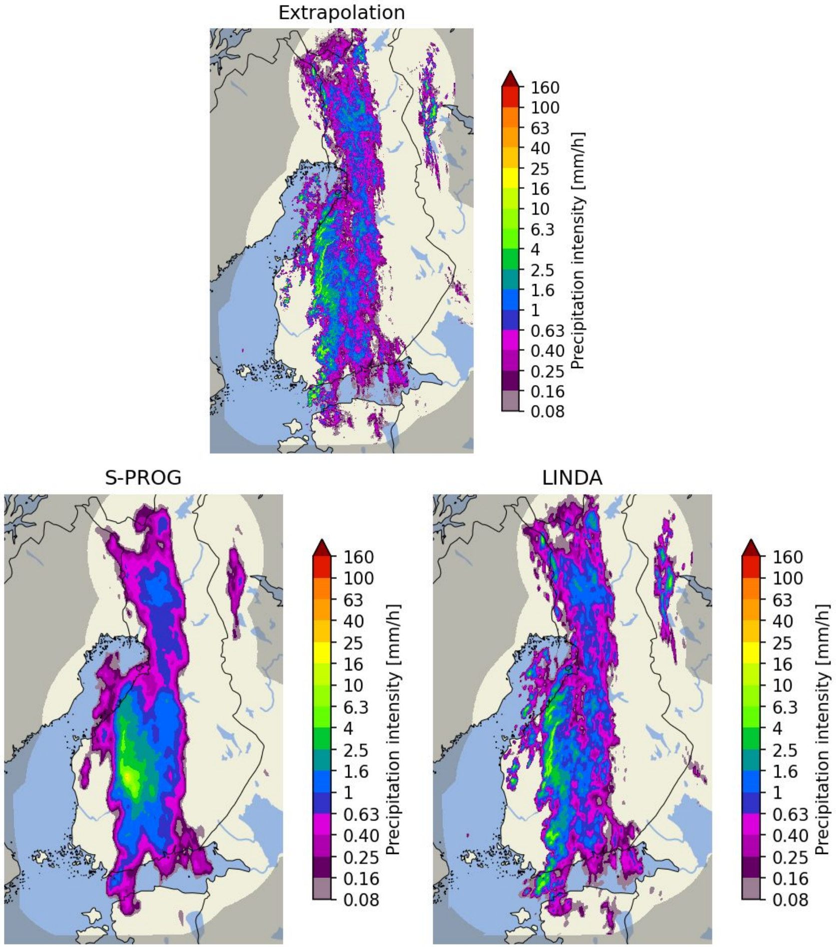




Deterministic Nowcasts

The main methods implemented in the **nowcasts** module:

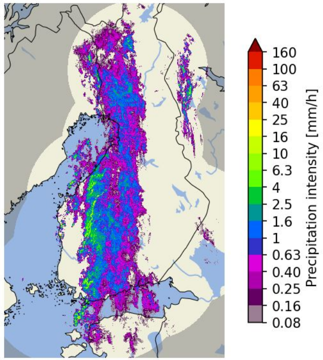
Method	Pros	Cons	Typical computation time
Extrapolation	very fast	no prediction of growth or decay of precipitation	< 10 seconds
S-PROG	• for low-intensity precipitation (< 1-2 mm/h) has generally the best skill • choose for stratiform events	inability to preserve the spatial structure of rainfall fields, and particularly convective cells	< 20 seconds
LINDA-D	• the most accurate method for intense precipitation (> 1-2 mm/h) • choose for convective events	slow to compute	might take several minutes



Ensemble Nowcasts

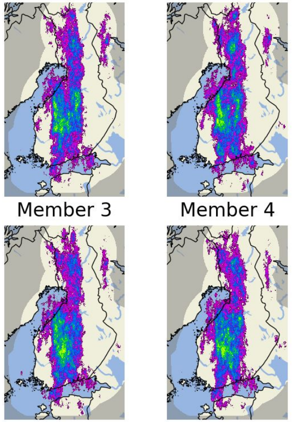
- The main ensemble methods implemented in the **nowcasts** module are STEPS and LINDA-P
- They model two sources of uncertainty: advection field estimation and Lagrangian growth and decay
- The basic rule for choosing the method:
  - stratiform events: STEPS
  - convective events: LINDA-P
- LINDA-P generally produces more realistic ensemble members
- Computation times for the 4-member ensembles shown on the right:
  - STEPS: ~20 seconds
  - LINDA-P: ~5 minutes

Observations at
2016-09-28 15:50 UTC



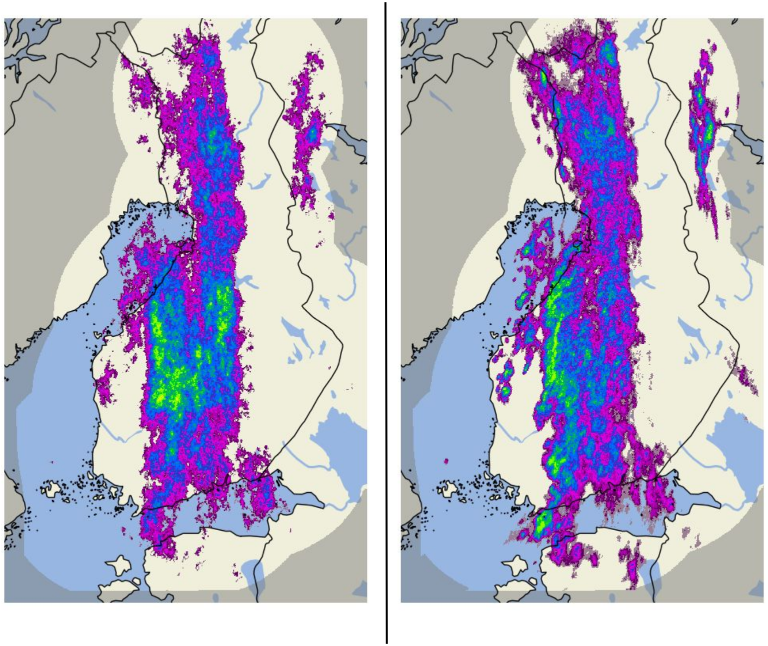
Nowcast ensemble

Member 1 Member 2



First ensemble members

STEPS LINDA-P

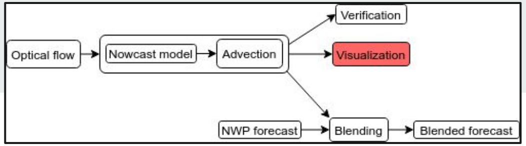


Benchmarking other methods

- When selecting correct method to benchmark against e.g. machine learning models, consider what do you want to achieve with the model
  - preserve variance
  - minimize error
  - represent uncertainty?
- To get comparable forecasts, use methods that try to achieve similar objectives
- For a more detailed discussion, see [the documentation]

From the documentation:

Nowcast type	Machine learning	pysteps	Verification
Deterministic (variance-preserving)	SRGAN, Others?	pysteps.nowcasts.extrapolation (any optical flow method)	MSE, RMSE, MAE, ETS, etc
Deterministic (error-minimization)	Classical ANNs, (deep) CNNs, random forests, AdaBoost, etc	pysteps.nowcasts.sprog, pysteps.nowcasts.anvil or ensemble mean of pysteps.nowcasts.steps/linda	MSE, RMSE, MAE, ETS, etc or better normalized scores, etc
Probabilistic (quantile-based)	Quantile ANN, quantile random forests, quantile regression	pysteps.nowcasts.lagrangian_probability or probabilities derived from pysteps.nowcasts.steps/linda	Reliability diagram (predicted vs observed quantile), probability integral transform (PIT) histogram
Probabilistic (ensemble-based)	GANs, VAEs, etc	Ensemble and probabilities derived from pysteps.nowcasts.steps/linda	Probabilistic verification: reliability diagrams, continuous ranked probability scores (CRPS), etc. Ensemble verification: rank histograms, spread-error relationships, etc

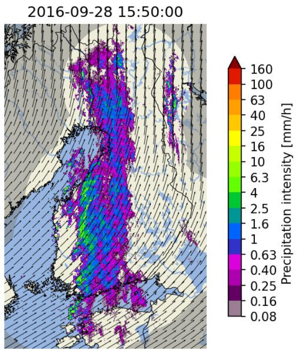


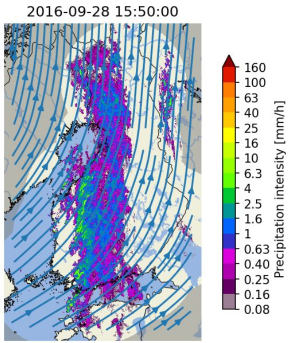
Colorbars with several pre-configured scales and for different data units

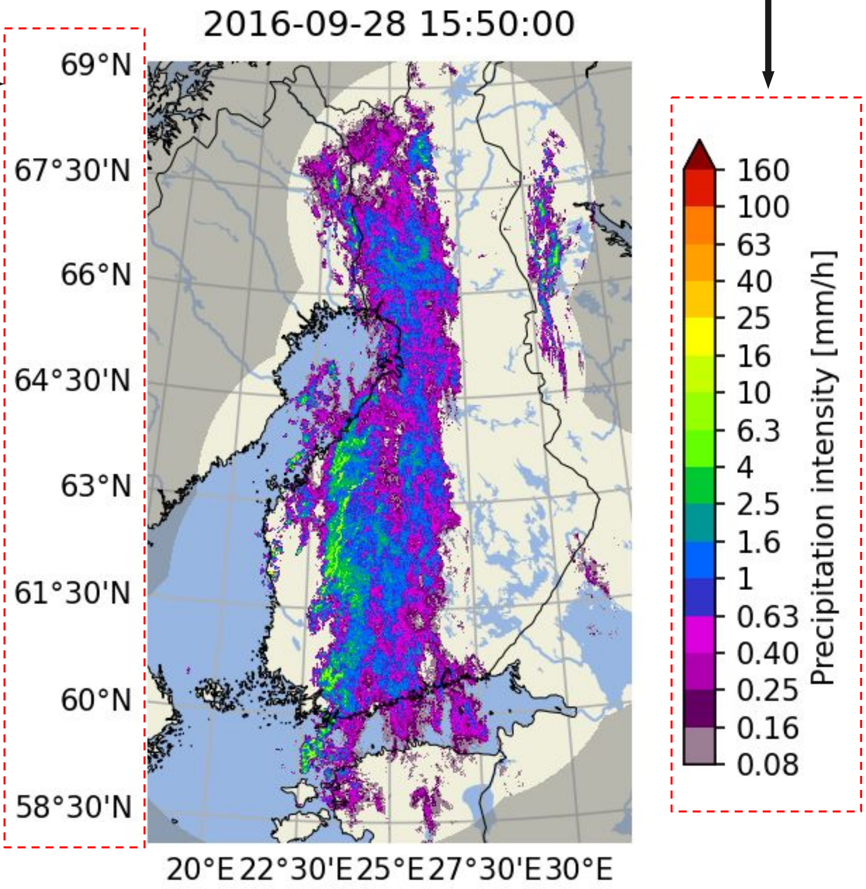
Longitude-latitude lines with labels

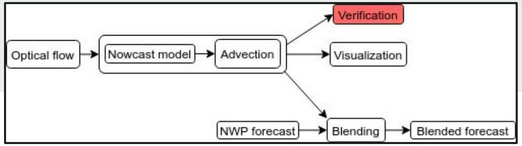
Visualization tools

- Extensive set of visualization tools has been implemented in the **visualization** module
- Support for multiple layers: basemap, precipitation and motion field:
  - plotting of basemaps by using cartopy
  - quivers and streamlines for advection fields









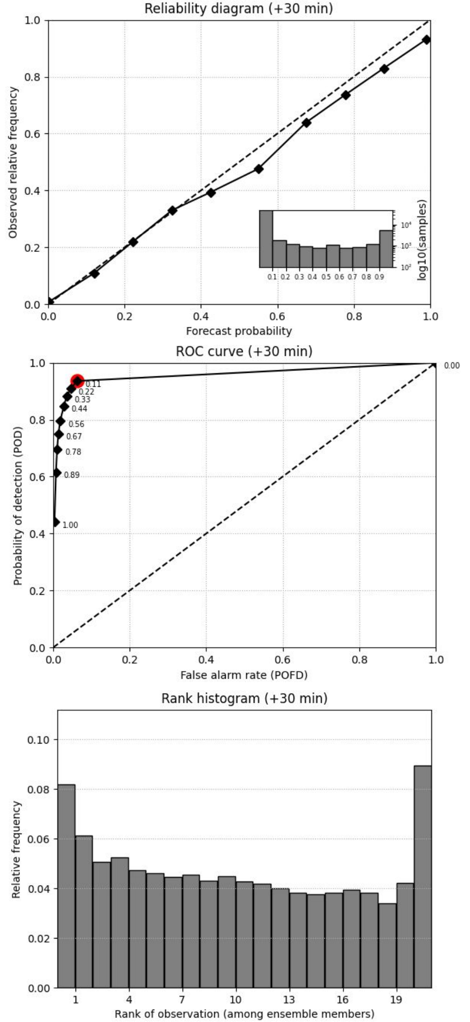
Verification tools / metrics

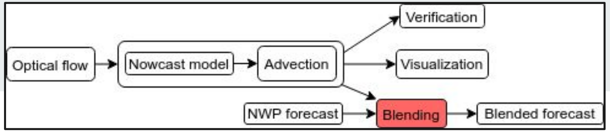
- A large number of verification utilities and metrics have been implemented in the **verification** module
- Functionality
  - creation of verification objects and aggregation from multiple nowcasts
  - plotting of verification results

Metrics

- Deterministic
  - categorical: CSI, ETS, POD, FAR
  - continuous: MAE, ME
  - scale/intensity-based metrics: FSS, intensity-scale
  - radially averaged power spectral density (RAPSD)
- Probabilistic
  - CRPS
  - reliability diagram
- Ensemble
  - spread
  - rank histogram

Examples of verification plots for 30-minute STEPS nowcasts

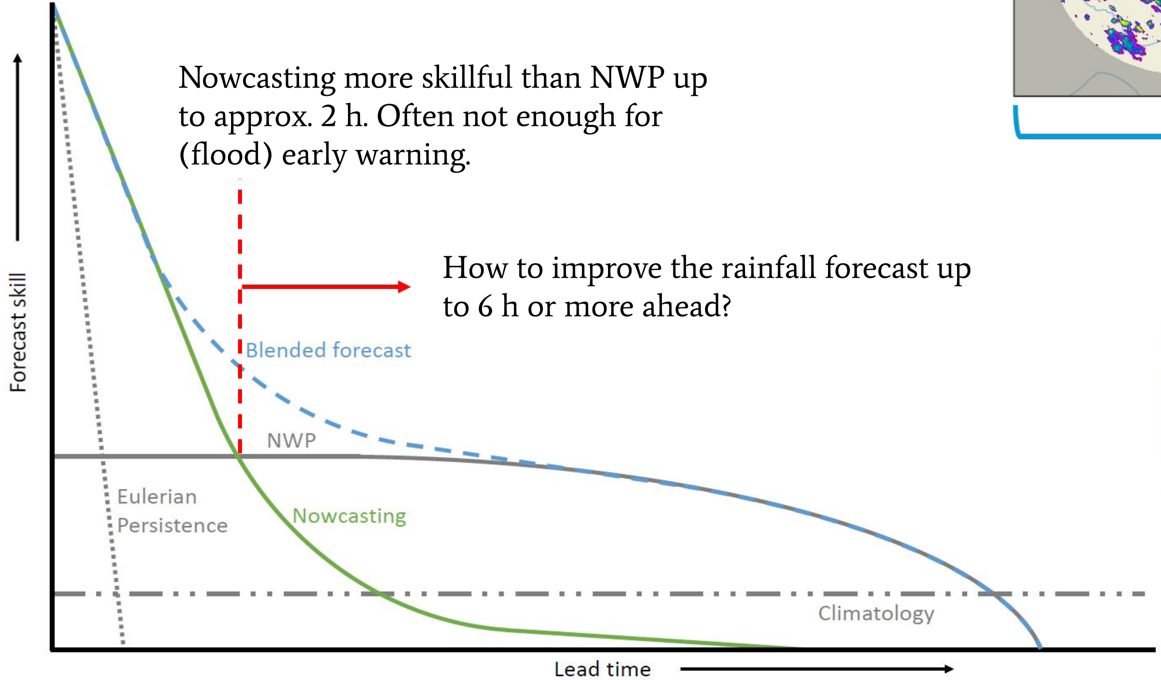


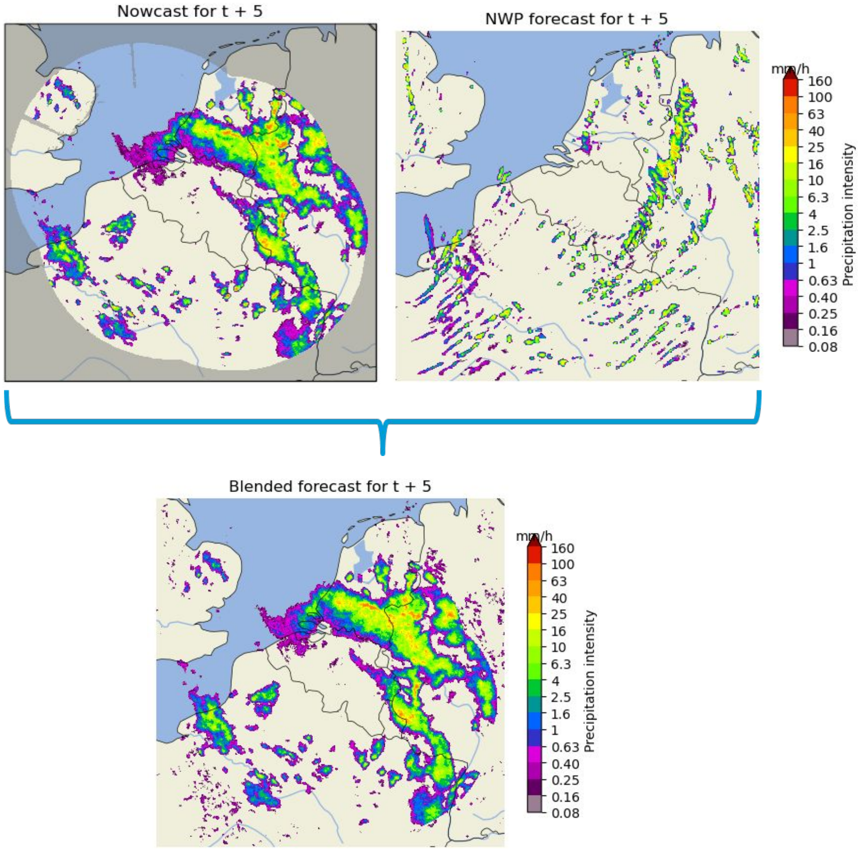


Blending with NWP

Nowcasting more skillful than NWP up to approx. 2 h. Often not enough for (flood) early warning.

How to improve the rainfall forecast up to 6 h or more ahead?

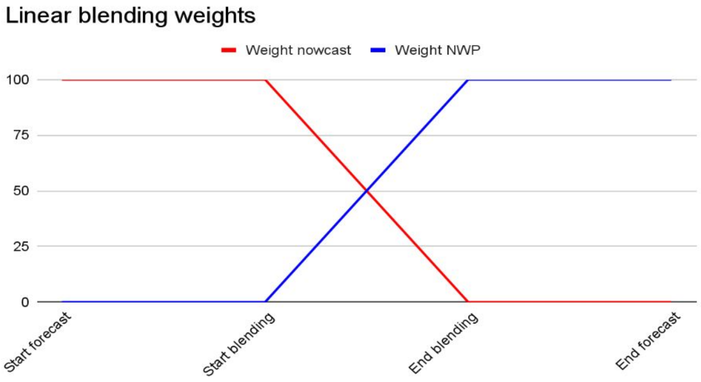




Blending with NWP: methods in pysteps

1. Linear blending

- Fixed start and end point of blending procedure
- Weights go linearly from 1 to 0 and 0 to 1.

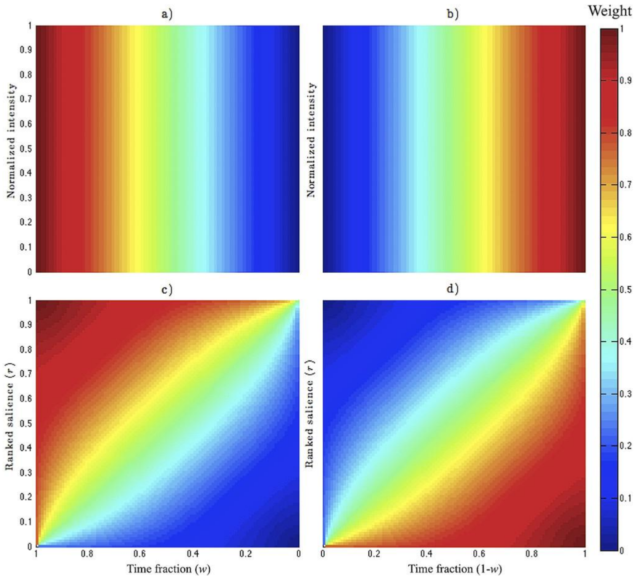


2. Saliency-based blending

- Similar to linear blending, but:
- Preserves pixel intensities over time if they are strong enough according to their ranked salience.

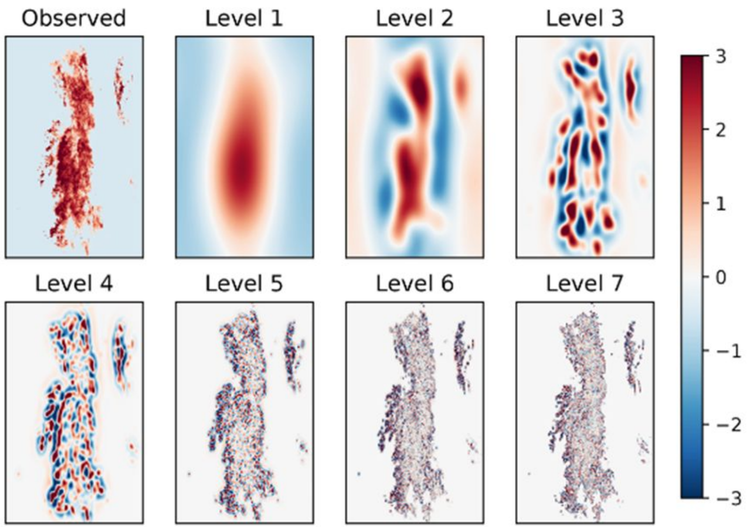
Hwang et al., 2015, Weather and Forecasting

3. STEPS blending (see next slide)



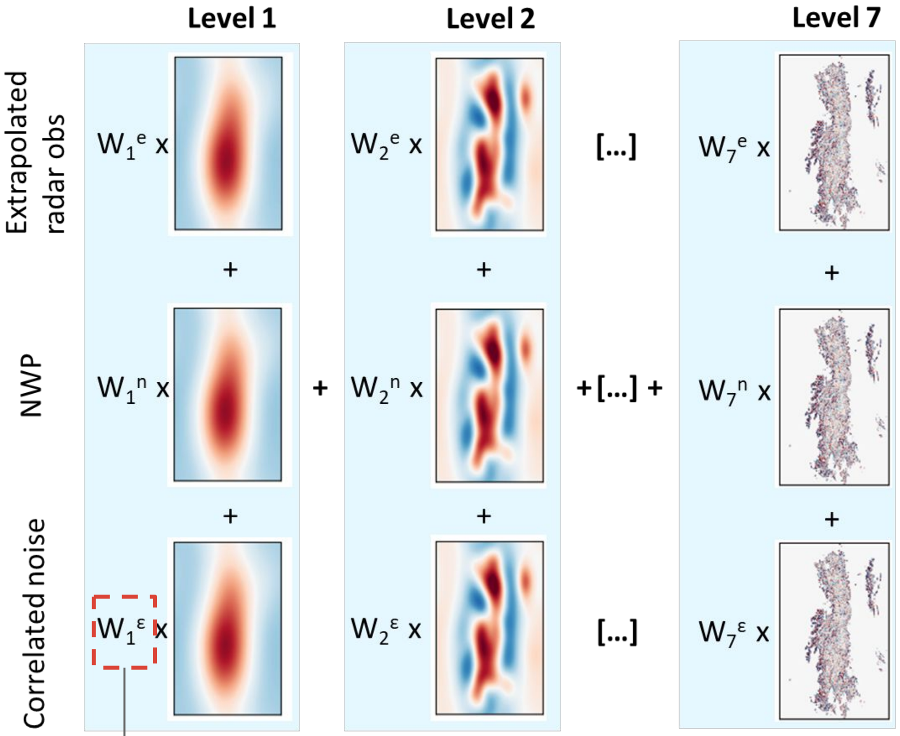
See the QPN session on Monday (16:30 - 18:00) for more information about this method and its' implementation in pysteps!

The STEPS method



Pulkkinen et al., 2019

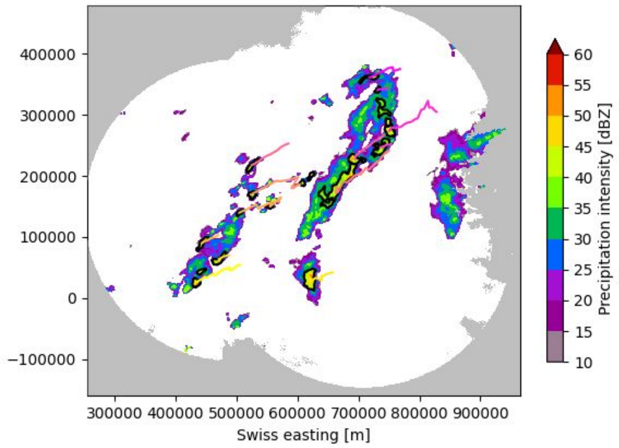
Per ensemble member:

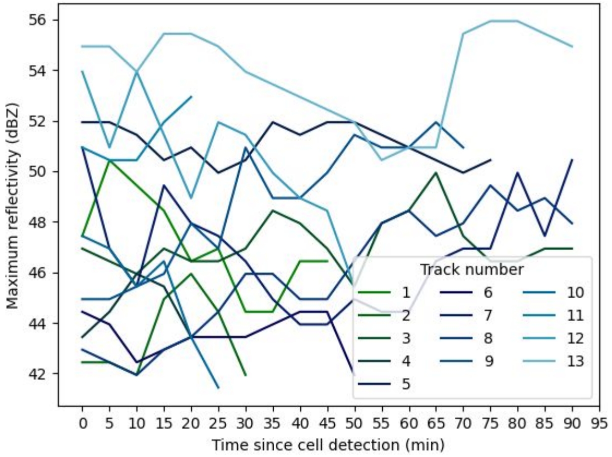


Blending weights depend on initial and expected future skill

T-DaTing module: Thunderstorm Detection and Tracking

- Identify and track thunderstorm cells from radar images
- Visualize cells and tracks in time
- Study properties of the cell tracks
- Tracking algorithm from the Swiss TRT Thunderstorms Radar Tracking algorithm
  - Hering et al., 2004, ERAD 2004
  - Feldmann et al., 2021, Weather Clim. Dynam
- For how to use, see the example in the gallery





Introduction to Google Colab

https://research.google.com/colaboratory

- Colab is a web-based Python environment
- Free of charge to use: you only need to have a Google account
- Support for different types of blocks:
  - *Code*: runnable code with output
  - *Section*: for organizing your notebook
  - *Text*: descriptions of code blocks
- The default environment has:
  - Python 3.7 with a large number of scientific packages pre-installed
  - 1 Intel Xeon CPU core with 2 threads
  - 12 GB memory, 100 GB disk space + Google drive