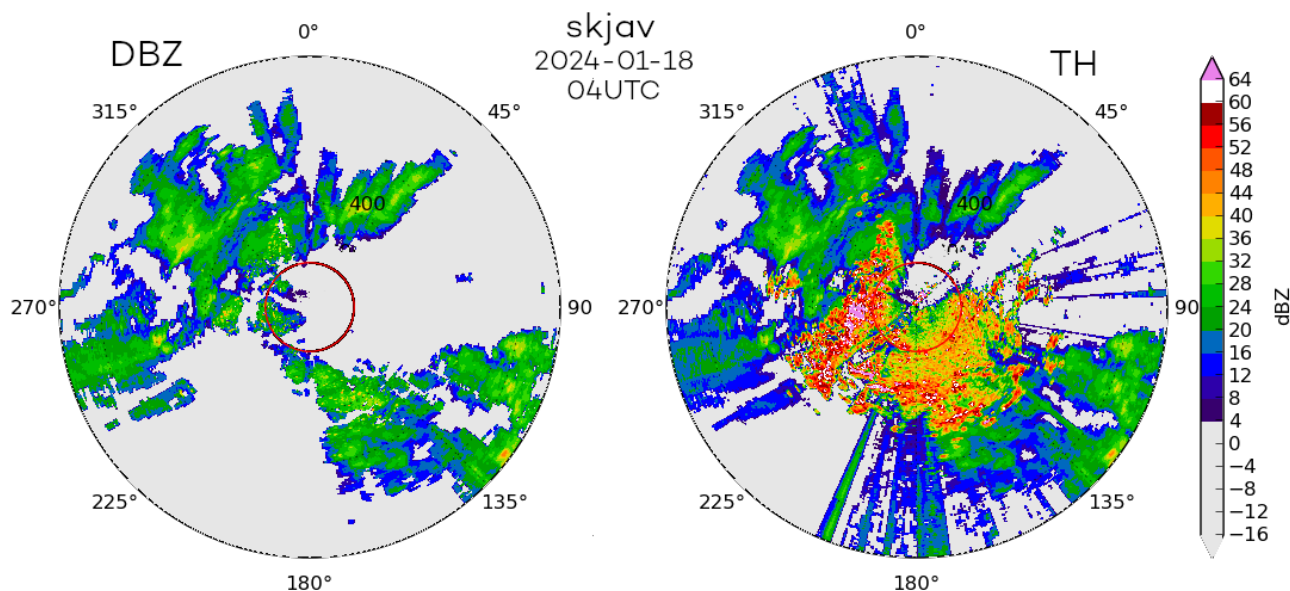


Testing of radar data from the new OPERA NIMBUS production line



Author: Michal Nestiak (SHMU)
Host: Antonín Bučánek, Alena Trojáková (CHMI)
Purpose: Report from RC LACE stay
Place: CHMI, The Czech Republic, Prague
Date(s): 22. 1. – 2. 2. 2024

Table of contents

Table of contents	1
1. Introduction	1
2. Comparison of OIFS and Nimbus radar data availability	2
2.1. Missing radars	2
2.2. Number of files per radar	2
2.3. Number of datasets and elevations for each radar station	3
2.4. 2D Histograms of reflectivity data	5
2.5. Content comparison of OIFS vs Nimbus data	7
3. Processing by HOOF	9
4. Processing by BATOR	9
5. Conclusions	12
6. References	13
7. Appendix	13
7.1. Assimilation experiments paths	13
7.2. Preparation of testing data for OPERA hubs	13
7.3. Mix of notes	14
7.4. Technical notes	14

1. Introduction

The provisional data centre Odyssey (denoted as OIFS) of the OPERA program will be replaced this year by three new production lines capable of better serving the diverse needs of various user groups. For the specific purpose of NWP assimilation, a dedicated Nimbus OPERA radar hub is established and is operated by Geosphere Austria (formerly known as ZAMG). Upon gaining access to the Nimbus Hub in October 2023, we observed differences not only in the number of radars but also in the processed content. This prompted our decision to independently compare the outputs of these two OPERA hubs. The control period was from 2024-01-01 to 2024-01-20. Data files were downloaded from OIFS and Nimbus data hub once a day for the previous day. Only 15 minutes data valid at every hour and 0 minute has been downloaded. A quick overview analysis of OIFS and Nimbus data was performed. The number of received files from OIFS and Nimbus were compared. The number of datasets and the number of elevations present in each file were compared. Heat maps of the number of observations in the DBZ and TH quantity were produced. Two experiments with a 3-hour assimilation cycle were set to test OIFS and Nimbus data for a shortened period from 2024-01-10 to 2024-01-20, due to excessive crashes of HOOF version 1.9 on Nimbus data.

2. Comparison of OIFS and Nimbus radar data availability

It is very important to ensure a smooth transition to the new Nimbus processing line as the shutdown of the OIFS hub is approaching. The new Nimbus hub should guarantee the availability of data from the same radars but also should ensure that the data are of comparable or better quality. Initially, we focus on monitoring the number of individual radars by country and identifying any missing records. The selected period includes the newest data available till the beginning of the stay. Data are processed from 2024-01-01 to 2024-01-20.

2.1. Missing radars

During the investigation period, we identified several radar stations that were missing from the OIFS or Nimbus production line. The list of the missing stations is shown in Table 1. The table shows that neither of the production lines is perfect but the Nimbus production line had more radar stations overall.

Table 1: Radars missing in OIFS or Nimbus data sets in the period 1-20 January 2024

OIFS missing radars	Nimbus missing radars
dksin	rsfrg
dkste	eszar
fraja	
isbjo	
isska	

2.2. Number of files per radar

The number of files for each radar station was compared between OIFS and Nimbus production line. Any radar station with more than a one percent difference in the number of files between production lines was included in Table 2. As can be seen, the Nimbus production line is missing almost all files for certain stations. The Nimbus production line has problems mainly with Polish, Romanian, Spanish and Greek radars but also German, Danish and French radar stations are in Table 2.

Table 2: Radars with a significant difference in the number of files between OIFS and Nimbus data sets in the period 1-20 January 2024 (the full sample should have 480 files)

Radar	OIFS file count	Nimbus file count	Difference [%]
plpas	475	27	94.3
grand	326	96	70.6
grlar	335	112	66.6
plgda	478	356	25.5
robar	250	194	22.4
plpoz	479	372	22.3
romed	474	376	20.7
plrze	476	387	18.7
roora	478	410	14.2
rotim	478	418	12.6
plswi	479	441	7.9
plleg	479	445	7.1
robov	124	116	6.5
deneu	476	469	1.5
essev	479	472	1.5
eslid	452	446	1.3
dkvir	457	451	1.3
eslpa	478	472	1.3
esbad	479	473	1.3
esse	436	431	1.1
frtre	479	474	1.0
espma	479	474	1.0
esmad	479	474	1.0
esbar	479	474	1.0

2.3. Number of datasets and elevations for each radar station

To better understand the differences between OIFS and Nimbus data, the number of datasets and elevations were compared between both hubs. Only data valid on 2024-01-15 00:00 UTC are compared. As can be seen from Table 3 the structure of OIFS and Nimbus files differ significantly. A given file may contain multiple elevation scans. Differences were found in the start times of the included elevation scans, e.g. the radar bejab contains three VRAD scans of VRAD starting at minute 49, 54 and 59 on elevation 0.5, while the OIFS contains three VRAD scans at minute 54, 59 and 04 (see files T_PAZZyy_C_EUON_20240118120000_bejab.hdf and T_PAZZ42_C_EUOC_20240118120000_bejab.h5).

Differences in the number of datasets are further analysed later, see section 2.5.

Table 3: Discrepancies in number of datasets and elevations between OIFS and Nimbus data on 2024-01-15 00:00 UTC. Possible errors in this table!

radar	OIFS Datasets	Nimbus Datasets	Common Elevations	Missing Elevations in OIFS	Missing Elevations in Nimbus
behel	111	39	12	0	0
bejab	156	63	15	2	0
bewid	84	48	15	0	0
czbrd	111	39	12	0	0
czska	111	39	12	0	0
deasb	93	33	10	0	0
deboo	93	33	10	0	0
dedrs	63	33	10	0	0
deeis	93	33	10	0	0
deess	93	33	10	0	0
defbg	93	33	10	0	0
defld	93	33	10	0	0
dehnr	93	33	10	0	0
deisn	93	33	10	0	0
demem	93	33	10	0	0
deneu	93	33	10	0	0
denhb	93	33	10	0	0
deoft	93	33	10	0	0
depro	93	33	10	0	0
deros	93	33	10	0	0
detur	93	33	10	0	0
deumd	93	33	10	0	0
dkrom	43	33	26	0	8
dkvir	37	29	24	0	7
esalm	11	13	3	0	0
esbad	11	13	3	0	0
esbar	11	13	3	0	0
eslid	11	13	3	0	0
eslpa	11	13	3	0	0
esmad	11	13	3	0	0
esmur	11	13	3	0	0
espma	11	13	3	0	0
essan	11	13	3	0	0
essev	11	13	3	0	0

esse	11	13	3	0	0
hrbil	39	30	9	0	0
hrdeb	39	30	9	0	0
hrgol	39	30	9	0	0
hrgra	39	30	9	0	0
hubud	38	38	29	4	4
huhar	48	38	11	2	1
hunap	38	38	10	2	2
hupog	38	38	14	2	1
husze	38	38	11	2	1
mtgud	113	36	11	0	0
skjav	111	39	12	0	0
skkoj	111	39	12	0	0
sklaz	111	39	12	0	0

2.4. 2D Histograms of reflectivity data

The 2D histogram analysis provides a comprehensive visual representation of the data. By plotting the counts of measured reflectivity for each radar station, we could quickly identify any significant variations or patterns across different stations. We chose for simplification observations collected over 24 hours on 2024-01-18. For several radar stations we saw fewer observations in the Nimbus data compared to OIFS, for instance, see the hubud station in Figure 1.

The Swiss stations in Nimbus data have significantly fewer observations and also have an oddly placed minimum that makes them very difficult to use in the assimilation of the ALADIN model (see Figure 2). BATOR calculates the minimum detectable radar reflectivity as a function of distance from the radar (here denoted by detection threshold). This function is used to assign a value to undetected observations. We used the same algorithm as in BATOR and plotted the radar detection threshold with a red line. As you can see in Figure 2, the radar detection threshold is quite wrong for the chlem radar station.

Note: chlem station has the coded value for undetect=1 in hdf file but DBZ data contain zeros for undetected observation instead!

The British stations in Nimbus data have similar issues as the Swiss stations, see Figure 3.

Figure 1: 2D histogram of the number of observations in the DBZ quantity for the radar “hubud” on 2024-01-18, red line indicates the radar detection threshold, pink and turquoise lines indicate the radar detection threshold with the minimum reflectivity factor of -30dBZ and -40dBZ, respectively.

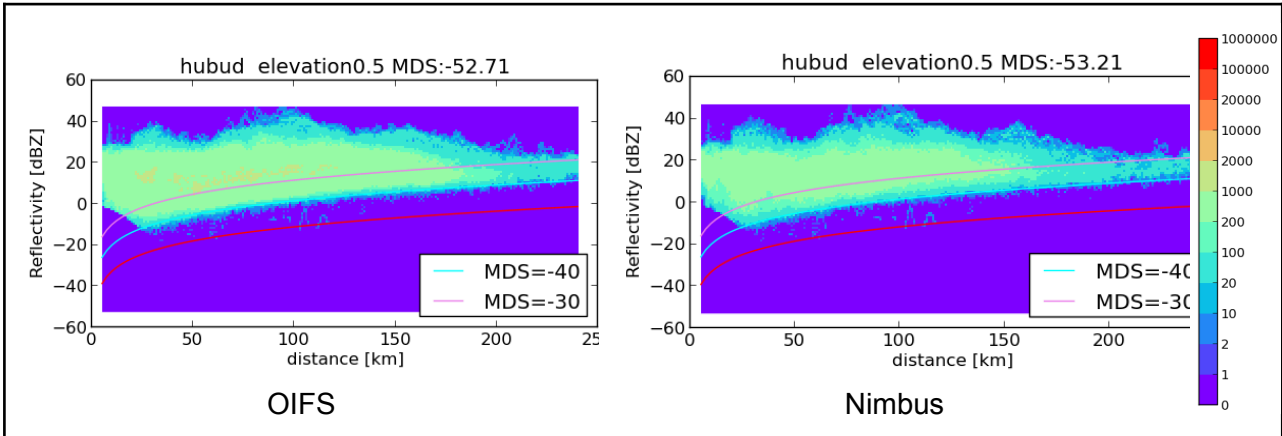


Figure 2: 2D histogram of the number of observations in the DBZ quantity for the radar “chlem” on 2024-01-18, red line indicates the radar detection threshold, pink and turquoise lines indicate the radar detection threshold with the minimum reflectivity factor of -30dBZ and -40dBZ, respectively.

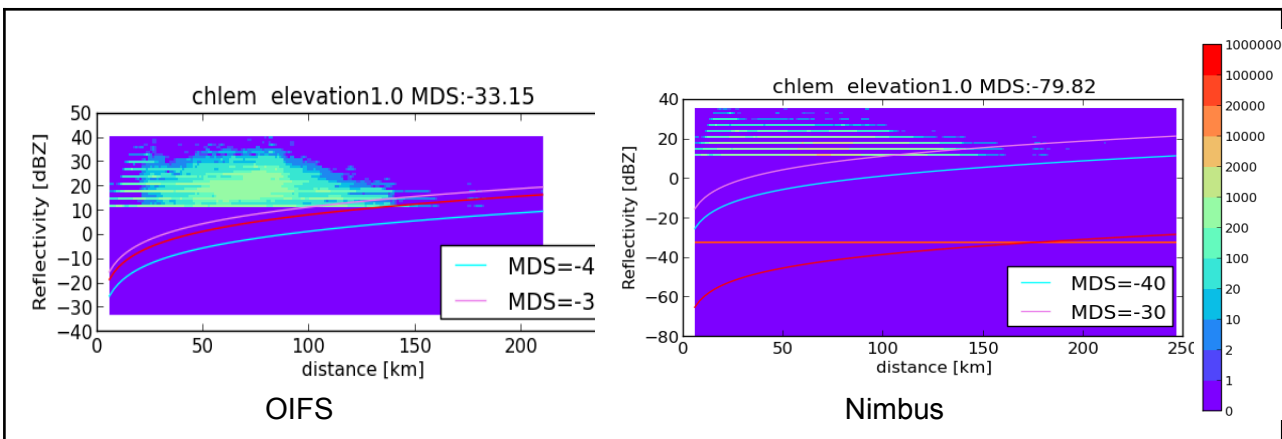
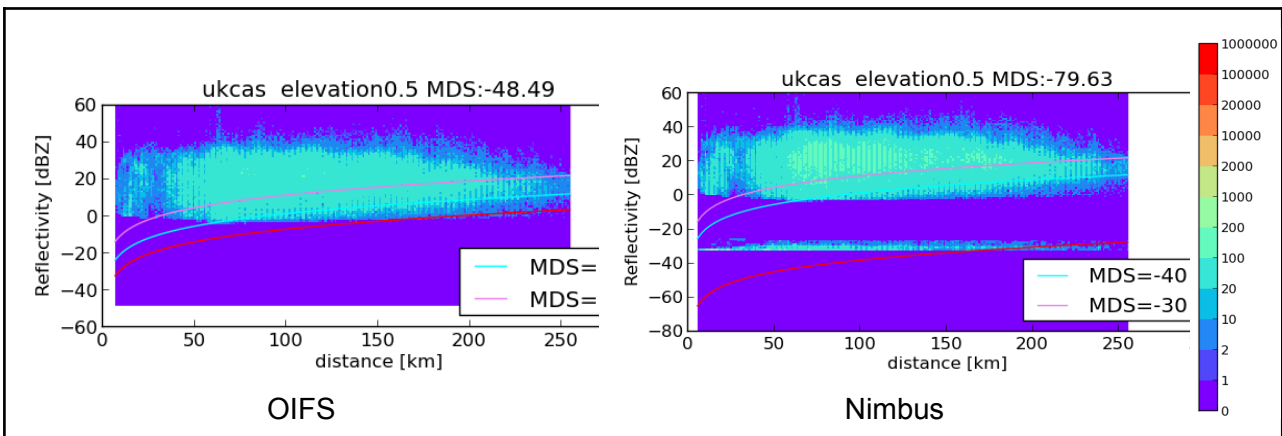


Figure 3: 2D histogram of the number of observations in the DBZ quantity for the radar “ukcas” on 2024-01-18, red line indicates the radar detection threshold, pink and turquoise lines indicate the radar detection threshold with the minimum reflectivity factor of -30dBZ and -40dBZ, respectively.



2.5. Content comparison of OIFS vs Nimbus data

The content of OIFS vs Nimbus files was compared with the local tool `ls_H5all.py` (uses `h5py` library) which lists a basic overview of the content of the given file. The Nimbus files are more compact in terms of a number of datasets, as can be seen in Table 3. Table 4 shows the difference in structure for the same variables, time, elevation between the OIFS file and the Nimbus file.

Table 4: comparison structure of OIFS vs Nimbus file for czbrd 2024-01-18 11UTC

OIFS	Nimbus
/dataset1: elevation 0.100000: startdate 20240118: starttime 105345: data1 VRAD: /dataset25: elevation 0.100000: startdate 20240118: starttime 105345: data1 DBZH: quality1 fi.fmi.ropo.detector.classification: quality2 se.smhi.detector.beamblockage: quality3 pl.imgw.quality.qi_total: /dataset97: elevation 0.100000: startdate 20240118: starttime 105345: data1 TH:	/dataset24: elevation 0.100000: startdate 20240118: starttime 105345: data1 DBZH: data2 VRADH: data3 TH: quality1 fi.fmi.ropo.detector.classification: quality2 se.smhi.detector.beamblockage: quality3 pl.imgw.quality.qi_total:
After Hoof /dataset12: elevation 0.100000: startdate 20240118: starttime 105345: data1 DBZH: data2 TH: quality1 fi.fmi.ropo.detector.classification: quality2 se.smhi.detector.beamblockage: quality3 pl.imgw.quality.qi_total:	After Hoof /dataset12: elevation 0.100000: startdate 20240118: starttime 105345: data1 DBZH: data2 TH: quality1 fi.fmi.ropo.detector.classification: quality2 se.smhi.detector.beamblockage: quality3 pl.imgw.quality.qi_total:

To be able to assimilate the OPERA radar data it is necessary to have all of the following parameters and necessary metadata in the dataset (e.g. NI for VRAD) in a dataset:

- For assimilation of reflectivity: DBZ (DBZH), TH, QC;
- For assimilation of radial wind: VRAD, DBZ, QC.

We observed several typical issues in the provided data from both Nimbus and OIFS hubs:

- Some datasets contain only DBZ, QC but it missing corresponding TH;
- Some datasets contain only VRAD but without corresponding DBZ and QC;
- Some datasets contain only VRAD and DBZ without corresponding QC;
- Some radars write precise values of elevation measured by radar in the datasets instead of preset/requested elevation angle for the radar. This means that one concrete elevation is

changing the elevation angle from one scan to the other. The angle difference is less than 0.1 degree (e.g. hubud).







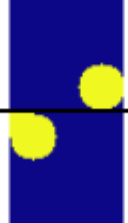





It would be beneficial to have a tool to compare the contents of the radar data files from both hubs since the data are stored in a different number of datasets, which may also contain different starttime. We would like to compare not only the attributes of the files but also the structural similarity of the dataset from both production lines because simple root mean square error is not a good indicator of similarity of the data due to double penalty in small shifts of the data.

Structural similarity

We decided to use opencv (cv2) and skimage python libraries (example of usage <https://www.kaggle.com/code/jsrshivam/structural-similarity>) for addressing the structural similarity between data from OIFS and Nimbus hub. This automated method identifies field disparities and allows for straightforward evaluation.

The first step was to write a python program that randomly generates two 2D arrays (data_oifs, data_nimbus) evolving in time and compares them. This program is made for testing the functionality of the cv and skimage libraries themselves. The program is located on kazi2 machine at CHMI: /home/mma271/app/compare_opera/bin/ss/oifs_vs_nimbus/fake_circle_compare_ssim.py

Figure 4: Structural similarity of randomly generated data of size 50x50 pixels.

Oifs				Nimbus				Oifs	Nimbus	Difference	field [50x50]
c1x	c1y	c2x	c2y	c1x	c1y	c2x	c2y				Image similarity
14	10	40	40	15	10	40	40				0.921
10	10	40	40	15	10	40	40				0.805
40	40	40	40	15	10	40	40				0.782
10	10	0	0	15	10	40	40				0.649

After conducting preliminary tests with generated data, I attempted to proceed with real data experiments. However, there were unresolved crashes in the h5py library of my miniconda user installation on the CHMI:kazi (python3.11) system. I tried my miniconda user installation on the SHMU:HPC3 (python3.9.18), but the same problem was observed. This was a blocking point in

the moment of stay. I have therefore postponed further investigation of this method until my return to SHMU, where I plan to conduct a thorough examination.

3. Processing by HOOF

A Python based tool called HOOF (Homogenization of OPERA OIFS Files,) was used to cope with inhomogeneity in the OPERA dataset in terms of structure and metadata. This tool allows the processing of the entire OPERA dataset with BATOR. We started testing with version 1.9 of HOOF, which was used during previous stays and studies at CHMI. We first tried to process all data on the selected period (from 2024-01-01 to 2024-01-20). Unfortunately, HOOF 1.9 cannot handle incomplete quality flags due to missing the what/gain section for the new quality flag “eu.opera.odc.hac”. Depending on radar scanning strategy it leads to no data or a partial hdf5-file after the HOOF. According to personal communication with Ladislav Meri from the SHMU radar department, the eu.opera.odc.hac quality flag is coded without the gain attribute and should represent ground clutter. The problem with eu.opera.odc.hac stopped on 10 January. Due to this reason, we shortened the testing period for assimilation experiments, see section 4.

Following the implementation of HOOF2, which requires Python 3, it was observed that the software demonstrated enhanced capability in managing incomplete quality flags. This improvement was attributed to its selective writing of requested quality flags, effectively circumventing the problematic flag “eu.opera.odc.hac”.

List of typical error/warning messages:

- Warning: The DBZ group /dataset1/data1 has no corresponding TH group
- Warning: A TH quantity from /dataset1/data1 has no matching DBZ, omitting the TH dataset
- DBZ quantity in /dataset1/data1 does not have the required quality groups, this error message often appears when VRAD and DBZ are in the same dataset.

VRAD datasets sometimes have different starttime than DBZ, TH and do not contain quality flags, typically for Hungarian and Spanish radars. Polish radars have missing TH for most levels in Nimbus data.

4. Processing by BATOR

To compare the OIFS and Nimbus data we set up two assimilation experiments and we shortened the period for experiments due to the problem with the eu.opera.odc.hac quality flag in the HOOF version 1.9 (HOOF version 2 was not installed at the moment of preparation). The period was from 2024-01-10 to 2023-01-20. We made a subset of files that were in both data sources to exclude differences from missing files. We used the ALARO NH-v1B model on cycle cy43t2ag_op2. We run a 3-hour assimilation cycle with the default setup of radar assimilation, see Appendix for more details. The data from selected radars were first homogenised by HOOF and then read by BATOR and finally used in assimilation. We had no more technical issues with processing Nimbus data.

A comparison of the number of observations retrieved by BATOR from OIFS and Nimbus data is shown in Figure 4. The counts of observations are split according to the country of the radar station. OIFS data are denoted by a turquoise line, Nimbus data are in red. As can be seen, some countries have the same number of processed observations from OIFS and Nimbus, such as the Czech Republic, Croatia and France. However, other countries have a significant decrease in Nimbus data compared to OIFS data, such as Denmark, Spain, Hungary and Germany. This data drop is very dependent on the station itself. We selected the three stations with the largest drop in Nimbus data compared to OIFS data, see Figure 5. German radar deneu has a very similar number of observations between Nimbus and OIFS data the first part of the period. However, from 14 January there is a very significant decrease in the number of available observations in Nimbus data. Danish radar dkrom always has less observations in Nimbus data compared to OIFS data. Hungarian radar hubud is missing observation in Nimbus data very often, but there are also times when Nimbus and OIFS data have the same number of observations, this could indicate some processing issues in the Nimbus hub.

Figure 5: Number of observations per state processed by BATOR during the period from 2024-01-10 to 2024-01-20.

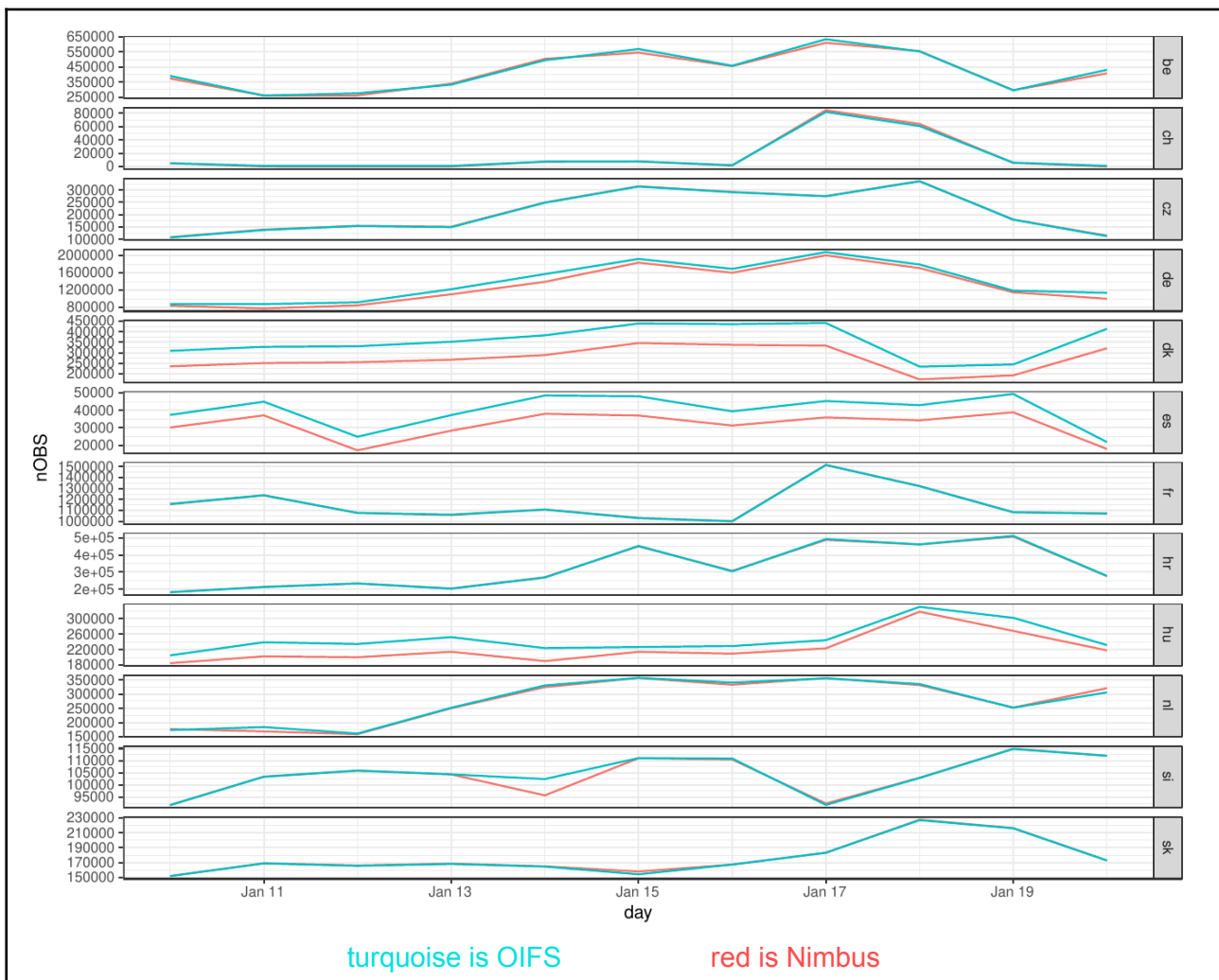
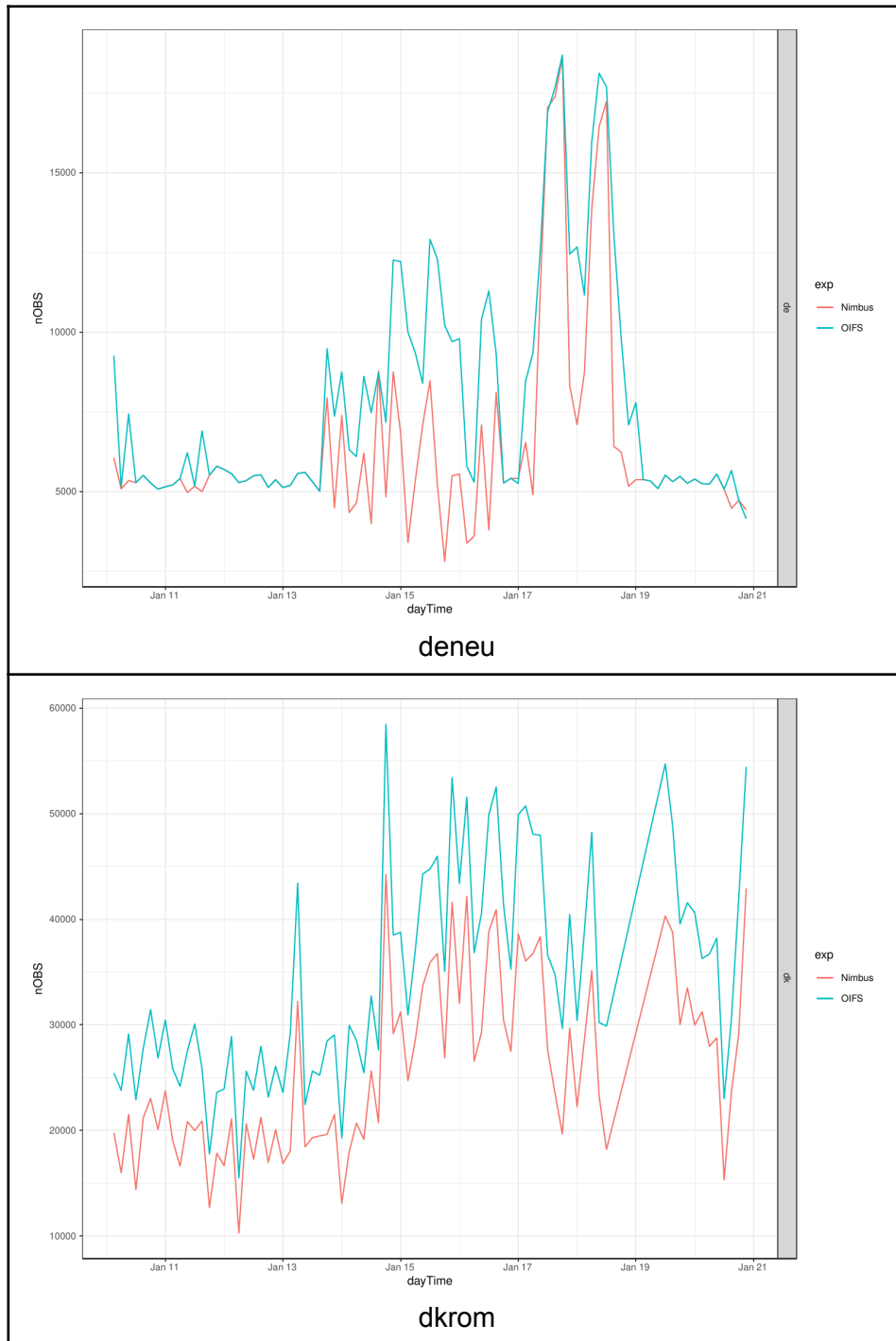
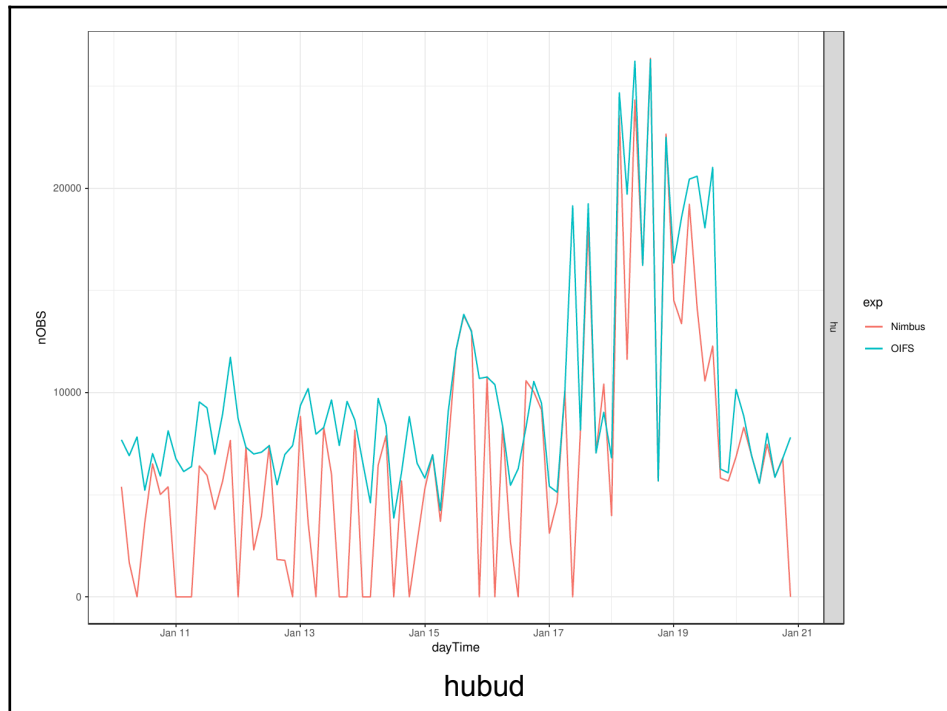


Figure 6: Number of observations for the deneu, dkrom, hubud radar processed by BATOR during the period from 2024-01-10 to 2024-01-20.





5. Conclusions

A brief comparison of data collected from the recently started Nimbus production line with the existing OPERA OIFS production line was made during this stay. Nimbus data contained more radar stations over the examined period (from 2024-01-01 to 2024-01-20) compared to OIFS. On the other hand, more often there were missing files in Nimbus data compared to OIFS, see Table 2. Two-dimensional histograms gave the impression that the Nimbus data contained fewer observations compared to the OIFS data. And also showed an oddly placed minimum of Nimbus data for Swiss and British radar stations. There was an attempt to explore the structural similarity between the data from both production lines. Unfortunately, it was postponed due to technical problems with python libraries.

The new Nimbus data were homogenised by HOOF. Unfortunately, HOOF version 1.9 cannot handle incomplete quality flags (due to missing what/gain for the new quality flag “eu.opera.odc.hac”). Depending on radar scanning strategy it leads to no data or a partial hdf5-file after the HOOF. The HOOF version 2 can handle incomplete quality flags as it writes only requested ones (without the problematic flag “eu.opera.odc.hac”).

At the last step, two assimilation experiments were prepared to compare the amount of observations processed by BATOR. We observed a significant decrease in the number of observations in Nimbus data compared to OIFS data, especially for radar stations from Denmark, Spain, Hungary and Germany. The positive aspect is that so far, we haven't encountered any technical problems related to assimilation. However, it is important to note that a comparison of assimilated observations was not done yet.

6. References

<https://www.eumetnet.eu/activities/observations-programme/current-activities/opera/>

https://www.eumetnet.eu/wp-content/themes/aeron-child/observations-programme/current-activities/opera/database/OPERA_Database/index.html

7. Appendix

7.1. Assimilation experiments paths

experiment oifs data (kazi2): /home/mma204/SX/scr/exp/zkd
work directory: /work/mma204/exp/zkd

experiment nimbus data (kazi2): /home/mma204/SX/scr/exp/zke
work directory: /work/mma204/exp/zke

plots: /work/mma204/exp/zkd

7.2. Preparation of testing data for OPERA hubs

For the purpose of expediting testing procedures, we required a concise set of OPERA hubs data. This need led to the creation of the program `create_tar_selected_cc.py`, which can be accessed at the following location:

kazi2: /home/mma271/app/compare_opera/bin/create_tar_selected_cc/

It is possible to define which radars from specific countries will be included in the testing tar. The relevant parameters in the script are:

```
hubs = ["oifs", "nimbus"]  
cc = ["sk", "cz", "hu", "hr"]
```

Example output:

```
/home/mma271/app/compare_opera/bin/test_output_nimbus.tar [sk,cz, hu, hr]  
/home/mma271/app/compare_opera/bin/test_output_oifs.tar [sk, cz,hu,si]
```

This tool was customised for future use in ongoing work, particularly not only for comparing radar files from each hub but also in subsequent model data assimilation experiments. However, caution is advised, as files with the same radar names may not necessarily contain identical input data. It will be better explained later with an example, but shortly in file with the same name may be a different time period of scans.

For preparing pure file statistic we write python program which class Radar:

```
def __init__(self, filename, lat=None, lon=None):
    self.filename = filename
    self.lat = lat
    self.lon = lon
    self.parse_filename()
```

```
def generate_tarfile_paths(unixtime_N, array_names, file_name_masks, file_name_ext):
def extract_main_filename(file_path):
def write_main_filenames_to_file(main_filenames, output_file):
def compare_files(file1, file2):
def compare_tar_files(tarfile1_path, tarfile2_path, output_file=None):
def list_files_in_tar(tarfile_path):
def print_radar_info(radar_array, array_name):
def print_radars_by_country(radar_array, country_code):
def print_unique_names_by_country(radar_array):
def print_unique_names_and_sum_by_country(radar_array):
def count_radars_by_country(radar_array):
def print_count_radars_by_country(radar_array):
def print_unique_names_and_sum_by_country(*args):
def print_unique_names_and_sum_table(*args, array_names=None):
def main(year_S, month_S, day_S, hour_S, minute_S, year_E, month_E, day_E, hour_E,
minute_E):
```

/home/mma271/app/compare_opera/bin/compare_tar_v010_single_day.py

/home/mma271/app/compare_opera/bin/compare_tar_v011.py

7.3. Mix of notes

Tu som docasne dal vsetok ten balast. Urobim si v tom poriadok, len nech to tu nerobi velky pocet stran

https://docs.google.com/document/d/1jh43spm9UitBqoYoRtaC_yYuPKulwxPUZwZShA-Rs_wM/edit#heading=h.tdeyemyb6wo4

7.4. Technical notes

Install of miniconda on chmu:kazi

For develop purpose we installed on CHMU:kazi miniconda3 (Python 3.11.0)


```
import geopandas as gpd
import pandas as pd
import matplotlib.pyplot as plt

source

# Assuming you have a GeoDataFrame with country boundaries (world) - you may need to download this or use a different
world = gpd.read_file(gpd.datasets.get_path('naturalearth_lowres'))

# Assuming you have a DataFrame with country data (data)
# Example data
data = pd.DataFrame({
    'cc': ['be', 'ch', 'de', 'fr', 'uk'],
    'value': [1, 2, 3, 4, 5]
})

# Merge the country data with your data
merged_data = world.merge(data, how='left', left_on='iso_a2', right_on='cc')

# Create a figure and axes
fig, ax = plt.subplots(1, 1, figsize=(15, 10))

# Plot the map and fill by the 'value' column
merged_data.plot(column='value', ax=ax, legend=True, cmap='Blues', legend_kwds={'label': 'Value'})

# Save the figure
plt.savefig('filled_map.png', bbox_inches='tight')

# Show the plot
plt.show()
```