Naval Research Laboratory
Washington, DC 20375-5320

NRL/7530/MR—2024/1

Observation Uncertainty Estimation: Desrozier Diagnostic with Python

WILLIAM F. CAMPBELL

HUI W. CHRISTOPHERSEN

Atmospheric Dynamics and Prediction Branch
Marine Meteorology Division

June 26, 2024

REPORT DOCUMENTATION PAGE

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ORGANIZATION

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED	
		START DATE	END DATE
26-06-2024	NRL Memorandum Report	10/01/2022	07/01/2024

4. TITLE AND SUBTITLE
Observation Uncertainty Estimation: Desrozier Diagnostic with Python

5a. CONTRACT NUMBER	5b. GRANT NUMBER	5c. PROGRAM ELEMENT NUMBER
5d. PROJECT NUMBER	**5e. TASK NUMBER**	**5f. WORK UNIT NUMBER**
		992B10

6. AUTHOR(S)
William F. Campbell and Hui W. Christophersen

7. PERFORMING ORGANIZATION / AFFILIATION NAME(S) AND ADDRESS(ES)
Naval Research Laboratory
7 Grace Hopper Ave
Monterey, CA 93943

8. PERFORMING ORGANIZATION REPORT NUMBER
NRL/7530/MR—2024/1

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Naval Research Laboratory
4555 Overlook Ave SW
Washington, DC 20375-5320

10. SPONSOR / MONITOR'S ACRONYM(S) NUMBER
NRL

11. SPONSOR / MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT
DISTRIBUTION STATEMENT A: Approved for public release; distribution is unlimited.

13. SUPPLEMENTAL NOTES

14. ABSTRACT
This report introduces a software tool designed to compute observation uncertainty utilizing the Desroziers method. Notably, this software enables users to flexibly calculate observation error covariance across user-defined regions for specific periods. The outputs generated by this software hold potential applications in other methodologies, including the three-cornered hat method and machine learning-based uncertainty estimation frameworks.

15. SUBJECT TERMS
Desroziers diagnostic, correlated observation error, data assimilation, numerical weather prediction

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES
a. REPORT	b. ABSTRACT	c. THIS PAGE		
U	U	U	SAR	23

19a. NAME OF RESPONSIBLE PERSON	19b. PHONE NUMBER *(Include area code)*
William F. Campbell	(831) 656-5161

This page intentionally left blank.

Observation Uncertainty Estimation: Desroziers Diagnostic with Python

William Campbell and Hui Christophersen

Atmospheric Dynamics and Prediction Branch
Marine Meteorology Division

Abstract:

At most operational NWP centers, observation error covariance matrices are typically presumed to remain constant in space and time. Our study has revealed that even observation uncertainty for longstanding, stable satellite sensors displays spatial and temporal variability. This report introduces a Python software tool designed to compute observation uncertainty via the well-known Desroziers method. Notably, this software enables users to flexibly calculate observation error covariance across user-defined regions and time periods. The outputs generated by this software hold potential applications in other methodologies, including the three-cornered hat method and machine learning-based uncertainty estimation frameworks.

1. Introduction

Data assimilation (DA) methods combine model forecasts (aka backgrounds) with observations, weighting them based on their respective uncertainties to produce the most reliable estimate of the atmospheric state (aka the analysis). Precise characterization of these error matrices is essential for accurate state estimation. Historically, there has been a primary focus on estimating and characterizing the covariance matrix of background errors (e.g., Bannister, 2008). Observation errors were assumed to be uncorrelated, and techniques such as thinning or 'superobbing' data have been used to fulfill this assumption (Lorenc, 1981). Recent research has explicitly addressed inter-channel correlations for microwave (Campbell et al., 2017) and hyperspectral infrared sensors (Geer 2019), demonstrating improvements in model analyses and forecasts. Given the need to utilize observations more effectively, especially for high-resolution forecasting, there is a pressing need to better understand and represent these statistical properties.

Observation errors may arise from various sources, some of which may vary with the atmospheric state and model resolution. A proven method for estimating observation uncertainties is the one proposed by Desroziers et al. (2005). This approach has been successfully employed to estimate observation uncertainty at several operational centers such as ECMWF (Bormann and Bauer, 2010; Bormann et al., 2010, 2016), the Met Office (Weston et al., 2014; Stewart et al., 2014), NCEP (Kleist et al., 2021), and FNMOC-NRL (Campbell et al., 2017). This method utilizes a large number of residuals between observations and their model equivalents to estimate observation uncertainty.

Currently, observation errors for any operational satellite sensors are represented by a single, globally constant standard deviation for each channel, along with the fixed global error covariance from a one-time Desroziers estimate using 2-3 months of residuals. This study aims to refine the Desroziers estimate by calculating it flexibly over a region for a period of time to determine if there is spatial and temporal variability in observation error. The report is organized into the following sections: section 2 describes

Manuscript approved June 25, 2024.

the dataset and steps for calculating uncertainty estimation, section 3 demonstrates the spatial and temporal variabilities for a few operational satellite sensors, followed by the application of the software and concluding remarks.

2. Materials and Methodology

2.1 Datasets

The Navy's Global Environment Model (NAVGEM; Hogan et al., 2014), along with its data assimilation system, the Naval Research Laboratory (NRL) Atmospheric Variational Data Assimilation System-Accelerated Representer (NAVDAS-AR; Daley & Barker, 2000; Kuhl et al., 2013; Rosmond & Xu, 2006; Xu et al., 2005), generates a vast array of model outputs useful for various applications. The current operational configuration of NAVGEM entails a horizontal resolution of 19 km (T681) with 60 vertical levels extending up to 0.04 hPa. NAVDAS-AR is a dual-resolution strong constraint hybrid 4DVar system, wherein the hybrid background error covariances currently consist of a linear combination: 75% derived from the static background error covariance and 25% derived from an 80-member flow-dependent ensemble (J. McLay et al., 2010; J. G. McLay et al., 2008).

The data assimilation (DA) system processes approximately 4 million observations every 6 hours, encompassing conventional observations, global navigation satellite system (GNSS) radio occultation bending angles and ground-based zenith total delay, satellite-derived atmospheric motion vectors (AMVs), satellite radiances, and trace gas retrievals. Notably, the DA system generates the model background, or first guess, and model analysis at the observation location and time, facilitating the derivation of Desroziers estimates in the observation space. Historical outputs from the operational NAVDAS-AR were stored in ASCII format; recent output is in HDF5 format.

2.2 Desroziers estimate

The Desroziers estimate uses the residuals of observation-minus-background ($O - B$) and observation-minus-analysis ($O - A$) to derive observation error covariance (R) for all assimilated observations.

$$R = < (O - B)(O - A)^T > \quad (1)$$

Note that we are only computing a block diagonal R. For satellite radiances, typically a profile is observed by a given instrument, consisting of measurements at different frequencies (channels) at the same location in space and time. Then each block of R contains a rank-one estimate of the interchannel error covariance for that satellite and instrument at a particular latitude, longitude, and time. (Spatially correlated observation error would be represented by off-diagonal blocks, which are set to zero in this report. This is the current standard practice at all operational centers.) For a global estimate of R, we simply average all of the available rank-one estimates, accounting for the possibility that some profiles may not have all channels available, and symmetrize the result (covariances must be symmetric, but sampling error, analysis imperfections, and observation bias can introduce asymmetry into the estimate). It is then straightforward to derive estimates for arbitrary spatial regions and time periods by restricting which rank-one estimates are averaged.

2.3 Steps to compute spatio-temporal-varying Desroziers estimate

The software predefines a set of regions as dictionary keys: Eastern Hemisphere, Western Hemisphere, Tropics (20°S to 20°N), Northern Mid-latitude (20°-60°N), Southern Mid-latitude (20°-60°N), South Pole

(>60°S), and North Pole (>60°N). The time component is primarily controlled as an input argument of the software, where the files are filtered by time (e.g., using wild card matching to identify files corresponding to the month of January for a given year).

The estimation process typically focuses on a specific observational type (e.g., AMVs, radiances for a particular sensor). For the current demonstration, we opt to include only satellite radiances. The initial step involves converting the innovation file from ASCII format to CSV format (using csvCreate.bash), which is then transformed into a Python dataframe with selected rows (representing instruments) and columns (containing meta information). Subsequently, this dataframe is saved in a Python pickle file (using csv2pandas.bash).

The Desroziers estimation module then reads these pickle files for a specified instrument and period, extracts data from the predefined region dictionary, and computes the covariance, partitions it into correlation and standard deviation, and plots them as a function of channels (dfr_desroz.bash).

To streamline these processes, three bash shell scripts are employed: csvCreate.bash, csv2pandas.bash, and dfr_desroz.bash. The computed covariances are then output for a given region, time period, and instrument.

Python and bash routines, along with documentation webpages autogenerated with Sphinx for v1.0, are available at https://github.nrlmry.navy.mil/metobillc/desroziers.git. The appendix A demonstrates the three step calculations with a day of innovation files.

3. Demonstration

Figure 1 illustrates the standard deviation and correlation of the covariance matrix pertaining to the Defense Meteorological Satellite Program (DMSP) Special Sensor Microwave Imager/Sounder (SSMI/S) across the Southern (20°-60° S) and Northern (20°-60° N) mid-latitude regions in January 2022. Both metrics, the standard deviation, and correlation matrix, exhibit noticeable spatial variability. Specifically, SSMIS data from surface-sensitive channels (channels 13-16) indicates lower standard deviations and higher cross-channel correlations within the Southern mid-latitude region compared to the Northern mid-latitude region.

Likewise, Figure 2 indicates temporal variations in the standard deviation of the error covariance, particularly for the high-altitude sounding channels (channels 22-24).

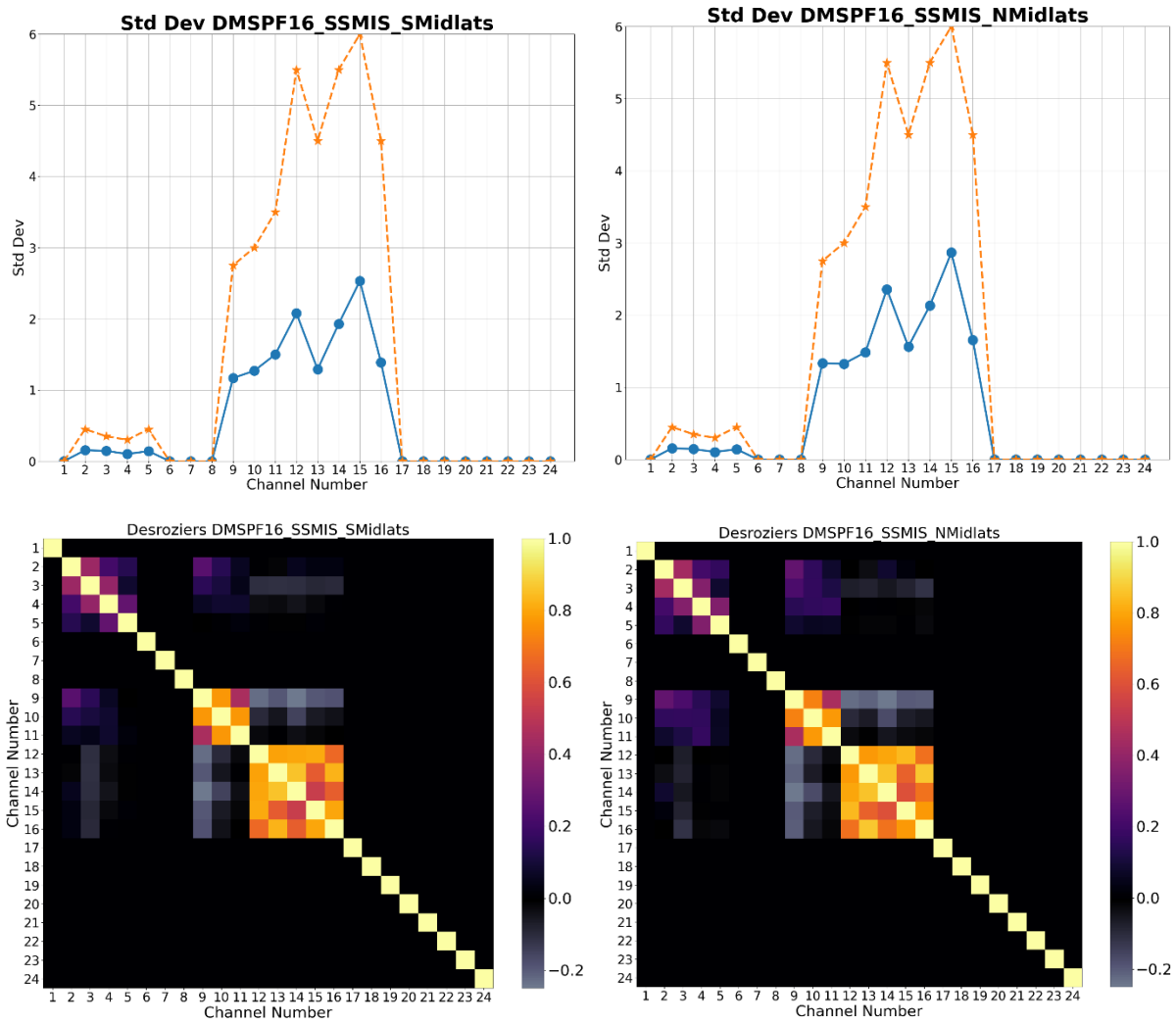

Figure 1. (Top) Standard deviation and (bottom) correlation of the error covariance matrix defined in equation (1) for Southern (20°-60° S) and Northern (20°-60° N) mid-latitude regions for DMSP F16 SSMIS during January 2022.

The orange dashed lines are those used in the operational NAVGEM DA system, while the blue solid lines are the ones diagnosed by the Desroziers estimate.

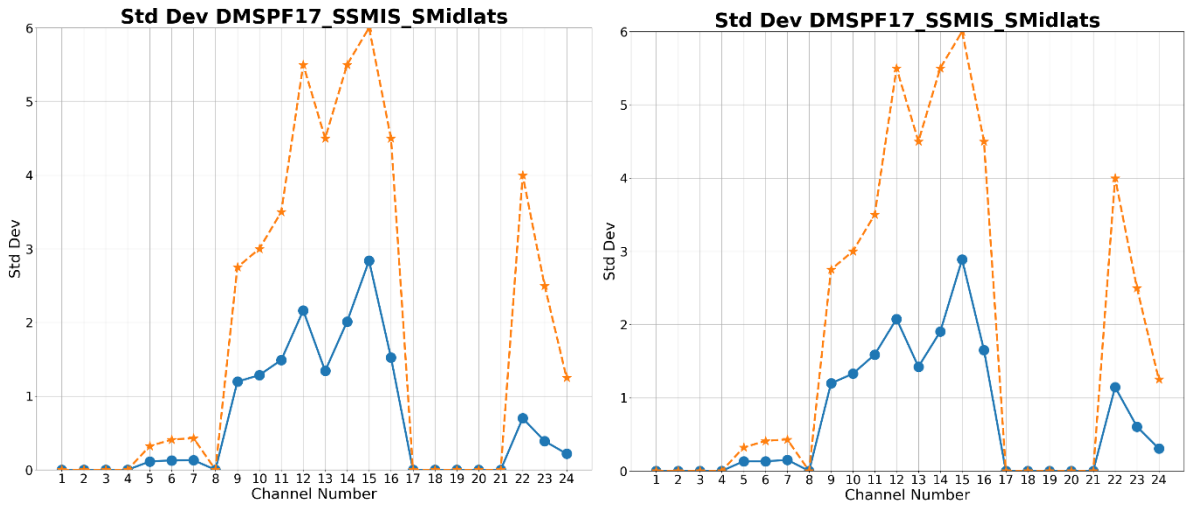

Figure 2. Standard deviation of the error covariance matrix defined in equation (1) for Southern mid-latitude region (20°-60° S) for DMSP F17 SSMIS during (left) January and (right) June 2022. The orange dashed lines are those used in the operational NAVGEM DA system, while the blue solid lines are the ones diagnosed by the Desroziers estimate.

4. Application of the software

The Desroziers estimation software offers versatile error covariance calculation capabilities tailored to specific satellite sensors. While it is primarily designed for satellite radiances featuring various channels, the same approach can be adapted for other observation platforms, such as atmospheric motion vectors and in situ observations. Moreover, the software's integration of observation, background, and analysis data facilitates the implementation of alternative uncertainty estimation methods such as the three-cornered hat method (Semane et al., 2022; Todling et al., 2022).

Recent studies have demonstrated the potential of machine learning (ML) methods in estimating error covariance (Christophersen et al., 2024). In this context, the Desroziers estimation derived from the software can serve as a reference truth for the ML framework, underscoring its utility and versatility in contemporary error estimation methodologies.

5. Concluding remarks

The report provides an overview of the materials and methodology involved in Desroziers estimation, a widely utilized technique within operational centers and the scientific community. Notably, the software enables flexible estimation concerning both region and time. Traditionally, legacy satellite radiances in operational centers have been associated with a fixed uncertainty, usually global and invariant over time.

Our study has revealed that uncertainty associated with legacy sensors, prevalent in most operational centers, exhibits spatial and temporal variability. Assessing the extent to which this variability influences analysis and forecasting necessitates thorough experimentation, which will be detailed in a subsequent study.

References

Bannister, R. N. (2017). A review of operational methods of variational and ensemble-variational data assimilation A review of forecast error covariance statistics in atmospheric variational data assimilation. I: Characteristics and measurements of forecast error covariances. Quart. J. Roy. Meteor. Soc., **134:** 1951–1970.

Campbell, W. F., et al. (2017). Accounting for correlated observation error in a dual-formulation 4D variational data assimilation system. Mon. Wea. Rev., **145**(3): 1019-1032.

Christophersen H., D. Sidoti, W. Campbell, and E. Satterfield, 2024a: Leveraging Machine learning to Exploit SmallSats, 104th American Meteorology Society annual conference, Baltimore, MD.

Desroziers, G., et al. (2005). "Diagnosis of observation, background and analysis-error statistics in observation space." Quart. J. Roy. Meteor. Soc., **131**(613): 3385-3396.

Geer A. (2019): Correlated observation error models for assimilating all-sky infrared radiances, Atmos. Meas. Tech., **12**, 3629–3657.

Lorenc A.C., 1981: A Global Three-Dimensional Multivariate Statistical Interpolation Scheme. Mon. Wea. Rev., **109**: 701–721.

McLay, J. G., et al. (2008). Evaluation of the ensemble transform analysis perturbation scheme at NRL. Mon. Wea. Rev., **136**(3): 1093-1108.

McLay, J., et al. (2010). A local formulation of the ensemble transform (ET) analysis perturbation scheme. Wea. Forecasting, **25**(3): 985-993.

Semane, N., et al. (2022). Comparison of Desroziers and Three-Cornered Hat Methods for Estimating COSMIC-2 Bending Angle Uncertainties. Journal of Atmospheric and Oceanic Technology **39**: 929-939.

Todling, R., et al. (2022). The relationship between two methods for estimating uncertainties in data assimilation. Quart. J. Roy. Meteor. Soc., **148**: 2942–2954.

Waller, J. A., et al. (2017). On diagnosing observation-error statistics with local ensemble data assimilation. Quart. J. Roy. Meteor. Soc., **143**(708): 2677-2686.

Appendix A

The calculation of Desroziers estimate includes three steps as listed in the section 2.3. This appendix provides a simple demonstration with a days of innovations files (four files in total at every 6 hours).

1. $python3 csvCreate.py /path/to/obs/sens/data/*

The input files are those automatically generated by the operational NAVGEM in the names such as "obs_sens_ops_YYYYMMDDHH.txt.bz2". The csvCreate.py reads directly individual bz2 compressed files and parses them into a new directory /path/to/obs/sens/data/innovarCSV/ for each file. Each input file has an csv formatted output with the name as "obs_sens_ops_YYYYMMDDHH.txt.csv".

One example command on DSRC nautilus machine: $ python3 csvCreate.py /p/work1/christoh/desroziers/obs_sens/*

2. $python3 innov2pandas.py /path/to/obs/sens/data/innovarCSV/*.csv -s /output/path/prefix

csvToPandas.py is to create dataframe with all rows and columns from the csv innovation vector and save sensor specific radiances into python pickle files. "prefix" or "suffix" offers additional flexibility for the saved filenames.

The saved filename convention is "{prefix}_{sensor}_YYYYMMDDHH.gz.{suffix}".

On DSRC nautilus, the test command is like this: $python3 innov2pandas.py $WORKDIR/desroziers/obs_sens/innovarCSV/*.csv -s $WORKDIR/desroziers/test_i2p/des

3. $python3 dfr_desroz.py / output/path/prefix/*{sensor}*YYYMMDD* -r {region_dict} -P /output/path/plot_prefix -o /output/path/output_prefix -S /output/path/save_prefix.

The dfr_desroz.py computes sensor specific Desrosiers estimate given the time period (which can be a day, a month, a year or any arbitrary time period matched with wild card) and region dictionary (Eastern Hemisphere, Western Hemisphere, Tropics (20°S to 20°N), Northern Mid-latitude (20°-60°N), Southern Mid-latitude (20°-60°N), South Pole (>60°S), and North Pole (>60°N)). The python script also gives options to customize output filenames with various prefixes.

One example with DSRC nautilus is: $ python3 dfr_desroz.py /p/work1/christoh/desroziers/test_i2p/*ATMS*20221001* -r Global -P ./plots.png -o ./output.txt -S ./full.pkl

Appendix B

Software documentation follows.

CONTENTS:

CHAPTER

ONE

CSVCREATE, (H5)INNOV2PANDAS, AND DFR_DESROZ

1.1 Readme File

Bill Campbell Last updated 6/17/2024 https://github.nrlmry.navy.mil/metobillc/desroziers

develop branch, hash 6caac82 Process of Execution:

0) Initial input is a directory with a set of innovation vectors, e.g. $WORKDIR/obs_sens/obs_sens_ops*.txt.bz2

Bash scripts must be modified to run on your particular machine.

If you plan to qsub csvDriver.bash (must be modified to run on calval3), then qsub calval_i2p.bash, then qsub dfr_desroz.bash, which is the recommended procedure, then you will need to first modify calval_shared.bash for whatever machine you are on (renaming it if you are not on calval). calval_shared.bash is assumed to be in $HOME/desroziers along with the other code,scripts, and this README file. You will need to modify datadir to point to the raw obs_sens data, either in your own location, or in someone elses archive where you have read access. You should also modify outdir, which is where all of your output will go.

1) **First step is to run csvCreate.py**
    a) python csvCreate.py $WORKDIR/obs_sens/obs_sens_ops*.txt.bz2 -p 8 (if supported, can use multiple processors with the -p option)
    b) **This will create a new directory 'innovarCSV' full of csv style files, which are the input to the** next step, either csvToPandas.py or innov2pandas.py
    c) Multiple styles of innovation vectors, such as those from the fsoi output, are handled correctly
    d) Gzipped and bzipped innovation vectors are also handled
    e) **Alternatively, qsub the csvDriver.bash script to spawn multiple copies of the csvCreate.bash script**
        1) csvCreate.bash invokes csvCreate.py with MPI to work on multiple files at once
2) **Second step is to run either a) or b) or c) or d) (d) is preferred):**
    a) **csvToPandas.py to create dataframe with all rows and columns from the csv innovation vectors**
        1) These will be large, but have all the information for e.g. data mining
    b) innov2pandas.py to create dataframe with chosen rows (instruments) and columns, e.g. those needed for the Desroziers calculation.
        1) **Invoked as follows: python innov2pandas.py innovarCSV/inn*20180124*.csv** -i 188 -c Instrument Satellite spacetime channel xiv_ob resid -s pkl_test_results/oneday_Jan24.pkl
        2) Leaving out -i will process all available instruments that are also in the sat_inst_info.py RADIANCE_SATNAMES dict

3) Leaving out -c will retain all columns relevant for Desroziers plus those that might be of interest for machine learning applications

4) These will be smaller, custom files for a specific application, such as the Desroziers calculation

5) Each instrument and DTG (but not satellite) will have its own pickled gzipped dataframe file

6) These files are the input for dfr_desroz.py

c) h5innov2pandas.py to create dataframe with chosen rows (instruments) and columns, e.g. those needed for the Desroziers calculation.

1) **Invoked as follows: python h5innov2pandas.py innovarH5/inn*2020090718*.h5** -i 188 -c Instrument Satellite spacetime channel xiv_ob resid -s pkl_test_results/oneday_Sep7.pkl

2) Leaving out -i will process all available instruments that are also in the sat_inst_info.py RADIANCE_SATNAMES dict

3) Leaving out -c will retain all columns relevant for Desroziers plus those that might be of interest for machine learning applications

4) These will be smaller, custom files for a specific application, such as the Desroziers calculation

5) Each instrument and DTG (but not satellite) will have its own pickled gzipped dataframe file

6) These files are the input for dfr_desroz.py

d) Alternatively, qsub the calval_i2p.bash script

3) **Third step is to run dfr_desroz.py (if second step was running b) or c) or d))**

a) **dfr_desroz.py takes the output from the second step and calculates, prints, and saves the resulting**
Desroziers matrices for each satellite for the selected instrument 1) Invoked as follows: python dfr_desroz.py

-F D:/ensv3_innov/pkl_test_results/twomonth_SSMIS*20160602* -S D:/ensv3_innov/pkl_test_results/ensv3_f5.pkl -P D:/ensv3_innov/pkl_test_results/ensv3f5.png

b) Alternatively, qsub the dfr_desroz.bash script (should work now, but untested)

c) ml_readi2p.py is basically a subset of dfr_desroz.py, that outputs an intermediate dataframe aggregated over time, suitable for further ML processing.

1.2 src

1.2.1 apply_example module

apply_example.**main**()

docstring

apply_example.**matrix_inflate**(*dfr*)

qwer

apply_example.**random_profiles**(*dfr*, *fraction=0.01*)

Each Goehash is a profile for some instrument

apply_example.**rank_one_sparse_desroziers**(*group*)

asdf

apply_example.**read_innov_columns**(*innov_file*)

Read and parse columns of innovation file

apply_example.**sparse_concat_rename**(*sparse*)

Sparsify, concatentate matrices

1.2.2 csvCreate module

Take a list of innovation vector files, and output a list of CSV files

csvCreate.**create_csv**(*saveh*, *fileh*, *write_header=True*)

Read innovation vector, parse, save as csv file.

csvCreate.**create_csv_from_h5**(*inputfile*)

Read innovation vector, parse, save as csv file

csvCreate.**line_parse**(*line*, *fmtstring*)

Decode unpacked line.

csvCreate.**main**()

csvCreate.**parse_args**()

Get args.

csvCreate.**read_h5_data**(*inputfile*)

Modify this to work for Desroziers.

csvCreate.**read_in**(*filename*, *outdir=None*)

read files, parse them, output as CSV

1.2.3 csvToPandas module

Use dasks built in read_table function to read in list of csv files, in the arguemnts we define the column names and data types. Skipinitialspace will skip leading spaces between commas to avoid errors. Na_filter set to false so that dask will not search for NA values, which speeds up our read times. Finally, we set collection to true so that result is a dask dataframe and not delayed data type.

class csvToPandas.**Range**(*start*, *end*)

Bases: object

csvToPandas.**main**()

csvToPandas.**parse_args**()

get args

csvToPandas.**read_in**(*filenames*, *desroziers_only=False*)

does everything, should be broken into smaller, focused methods

1.2.4 des_plots module

Plotting routines for desroziers matrices

des_plots.**cor_colormap**(*cor*)

Construct two-part colormap for correlation matrices Positive correlations should always have the same scale Negative correlation should have the same visual spacing

des_plots.**matrix_compress**(*full*, *tol=1e-08*)

Remove unit vector rows and columns

des_plots.**plot_cormat**(*cor*, *figbase*, *compress_tol=1e-08*)

Plot and save each raw desroziers matrix

des_plots.**plot_error_correlation**(*num_channel*, *C_bc*, *inst*, *sat*, *plot_title*, *figname*)

From desrozierstools/temp/desrozier.py

des_plots.**plot_error_stddev**(*num_channel*, *stddev*, *errstd*, *inst*, *sat*, *plot_title*, *figname*)

From desrozierstools/temp/desrozier.py

des_plots.**plot_multiple_stddev**(*stddev_dict*, *inst*, *sat*, *plot_title*, *figname*)

Plot std deviation as a function of channel for multiple experiments Expects dict that looks like {'ctlname': ctl_dict, 'expname': exp_dict}

des_plots.**plot_raw**(*des*, *inst*, *argplot*, *dtg_range*, *plot_std=True*)

Plot and save each raw desroziers matrix

1.2.5 des_regions module

Current defined regions in RegionsDispatch dict: Tropics, NMidlats, SMidlats, Midlats, NPole, SPole, Polar, WHem, and EHem.

des_regions.**grouped_stats**(*df*, *dtg=None*)

Global and regional stats from a full dataframe. Returns a regions dispatch table with stats dataframes for each region (key).

1.2.6 des_stats module

Convert between correlation and covariance

des_stats.**corr2cov**(*corr*, *std*)

convert correlation matrix to covariance matrix given standard deviation

Parameters

- **corr** (*array_like, 2d*) – correlation matrix, see Notes
- **std** (*array_like, 1d*) – standard deviation

Returns

cov – covariance matrix

Return type

ndarray (subclass)

Notes

This function does not convert subclasses of ndarrays. This requires that multiplication is defined elementwise. np.ma.array are allowed, but not matrices.

des_stats.**cov2corr**(*cov*, *return_std=False*)

convert covariance matrix to correlation matrix

Parameters

cov (*array_like, 2d*) – covariance matrix, see Notes

Returns

- **corr** (*ndarray (subclass)*) – correlation matrix
- **return_std** (*bool*) – If this is true then the standard deviation is also returned. By default only the correlation matrix is returned.

Notes

This function does not convert subclasses of ndarrays. This requires that division is defined elementwise. np.ma.array and np.matrix are allowed.

1.2.7 des_utils module

Utils for saving dicts and dataframes, plus other miscellaneous

des_utils.**get_dtgrange**(*filepaths*)

Do this right – extract basename of filenames, then find a 10-digit DTG string in each. Then find min and max.

des_utils.**save_dataframe**(*dfr*, *argsave*, *dtg_range*)

Save pickled dataframe

des_utils.**save_des**(*des*, *inst*, *argsave*, *dtg_range*, *fstr='_desroziers_'*)

Save desroziers dict

des_utils.**save_raw**(*des*, *inst*, *argsave*, *dtg_range*, *save_std=False*)

Save each raw desroziers matrix in txt files

1.2.8 desroziers module

Desroziers calculations and helper functions

desroziers.**compute_desroziers**(*dfr*)

Driver for calculation of desroziers matrices

desroziers.**ddt_symmetrize**(*array*)

Standard symmetrizing

desroziers.**random_profiles**(*ddf*, *fraction=0.001*)

Random sample of spacetimes Can this be done completely in dask to avoid the .compute()

desroziers.**rank_one_desroziers**(*dfr_inst*, *maxchan*, *pdf=False*)

des

desroziers.**rank_one_sparse_desroziers**(*dfr_inst*, *maxchan*, *pdf=False*)

des

1.2.9 desroziers_innov2pandas module

desroziers_innov2pandas.**add_columns**(*dfr*)

Added needed columns

desroziers_innov2pandas.**main**()

Extract instruments, columns from a set of innovation vectors, and save resulting dataframe From spyder, issue the following command: runfile('M:/Python/desrozierstools/innovarCode/Hodyss_innov2pandas.py',

wdir='M:/Python/desrozierstools/innovarCode', args='innovarCSV/innovar_1_2018012300.csv -i 70 71 72 73 74 75 -c ob lat_ob lon_ob insty c_pf_ob c_db_ob -s Hodyss.pkl')

desroziers_innov2pandas.**parse_args**()

usage: desroziers_innov2pandas.py [-h] [-i [INSTRUMENTS [INSTRUMENTS ...]]]
[-c [COLUMNS [COLUMNS ...]]] [-s SAVENAME] [-t SWITCH] filename [filename ...]

positional arguments:
filename filename(s) or wildcard

optional arguments:

-h, --help show this help message and exit

-i [INSTRUMENTS [INSTRUMENTS ...]], –insty_list [INSTRUMENTS [INSTRUMENTS ...]]

List of instrument codes

-c [COLUMNS [COLUMNS ...]], –columns [COLUMNS [COLUMNS ...]]

List of column names

-s SAVENAME, --savename SAVENAME Save dfr in pickle file

-t, --switch Read from pickle file, no dask.compute()

desroziers_innov2pandas.**rank_one_desroziers**(*dfr*)

The group has a multi-index whos first entry is the instrument

desroziers_innov2pandas.**read_in**(*filenames*)

Read files of innovations into dask dataframe

desroziers_innov2pandas.**select_subset**(*dfr*, *instruments*, *columns*)

Choose select instrument numbers and column names from dataframe

desroziers_innov2pandas.**sparse_concat_rename**(*sparse*)

Sparsify, concatentate matrices

1.2.10 dfr_desroz module

dfr_desroz.**compute_desroziers**(*dfr*)

For each sat/inst combination, compute desroziers and associated statistics.

dfr_desroz.**main**()

dfr_desroz.**parallel_file_read**(*filename*, *nprocs*)

Parallel read list of filenames using nprocs processors. Return a concatenated dataframe. Only a single instrument is allowed (multiple satellites are OK)

dfr_desroz.**parse_args**()

parse_args takes one mandatory argument, a list of pickled dataframe filenames (allows wildcarding). Optional args: -p to choose how many processors for parallel file read -o to choose an output directory for text file versions of correlations and std deviations -S to pickle and save the final dataframe (default='myexp.pkl') -P to plot the correlations and std deviations -r to choose a list of regions from those specified in the RegionsDispatch dict in des_regions.py (default='Global') As of this version, Tropics, NMidlats, SMidlats, Midlats, NPole, SPole, Polar, WHem and EHem are the defined regions Note that the -P, -o, and -S optional args implicitly contain a filenaming convention. For example, if one were to specify -o myoutdir/myprefix.text, then the output would go to the myoutdir directory, and all of the files would look like myoutdir/myprefix_{satellite}_{instrument}_{region}_{start_date}_{end_date}.text. Similarly, -S mypkldir/full.pkl yields a filename of mypkldir/full_desroziers_{inst}_{start_date}_{end_date}.pkl.

dfr_desroz.**read_preprocess**(*filename*)

Read in pickled dataframes (can be compressed, read_pickle should be able to infer the type and uncompress properly). Create and drop columns, change column type of channel to int, then outer merge on location (space-time). Finally create the desroziers column (IxRyProd) and drop redundant columns, returning initial dataframe

dfr_desroz.**select_satellites_and_regions**(*dfr*, *regions=['Global']*)

Loops through satellites and valid regions, computing Desroziers.

dfr_desroz.**txt_save**(*des*, *inst*, *argsave*, *dtg_range*, *save_std=False*)

Save some matrices as text files.

1.2.11 header module

header.**converter**()

header.**create_param**()

header.**insty_def**()

1.2.12 innov2pandas module

innov2pandas.**augment_columns**(*dfr*)

Generate new columns from information already encoded in dataframe

innov2pandas.**main**()

Extract instruments, columns from a set of innovation vectors, and save resulting dataframe From spyder, issue the following command: runfile('D:/desroziers/innov2pandas.py', wdir='D:/desroziers',

args='D:/desroziers_data/control_run/innovarCSV/obs_sens*.csv -i 185 197 198 199 200 -c Instrument Satellite lat lon ob_date_time channel err_ob xiv_ob resid -s 'D:desroziers_data/control_run/i2poutput/test.pkl')

Leaving out -i defaults to all available instruments Leaving out -c defaults to all columns relevant for machine learning

innov2pandas.**parse_args**()

usage: innov2pandas.py [-h]
[-i RADIANCE_SATNAMES.keys()] [-c [COLUMNS ...]] [-s SAVENAME] filename [filename ...]

positional arguments:
filename filename(s) or wildcard

options:

-h, --help show this help message and exit

-i RADIANCE_SATNAMES.keys(), –insty_list RADIANCE_SATNAMES.keys()
List of instrument codes

-c [COLUMNS ...], –columns [COLUMNS ...]
List of column names

-s SAVENAME, --savename SAVENAME Save dfr in pickle file

innov2pandas.**read_in**(*filename*)

Read files of innovations into pandas dataframe, retaining only radiance observations (jvar==13)

innov2pandas.**save_dtg**(*three_args*)

Save individual DTG pickle files. Expecting input tuple to have length 3, and include a dtg, a pandas dataframe, and a string contining a path for file saving

innov2pandas.**save_full_dataframes**(*dfr_dict*, *argsave*, *dtg_range*)

Save pickled dataframes by instrument

1.2.13 ml_readi2p module

ml_readi2p.**get_dtgrange**(*filepaths*)

Do this right – extract basename of filenames, then find a 10-digit DTG string in each. Then find min and max.

ml_readi2p.**main**()

ml_readi2p.**parallel_file_read**(*filename*, *nprocs*)

Parallel read list of filenames using nprocs processors. Return a concatenated dataframe. Only a single instrument is allowed (multiple satellites are OK)

ml_readi2p.**parse_args**()

Get args.

ml_readi2p.**read_preprocess**(*filename*)

Read in pickled dataframes (can be compressed, read_pickle should be able to infer the type and uncompress properly). Create and drop columns, change column type of channel to int, then return dataframe

ml_readi2p.**save_des**(*des*, *inst*, *argsave*, *dtg_range*, *fstr='_ML_preprocessed_'*)

Save ML preprocessed dict

ml_readi2p.**select_satellites_and_regions**(*dfr*, *regions=['Global']*)

Loops through satellites and valid regions, returning a dict of dataframes.

1.2.14 sat_inst_info module

Dicts with channel, satellite, instrument information

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX

R

S

T