Reproducibility in Action

My experiences writing a reproducible paper

Rasp, Selz and Craig, 2017. Variability and clustering of mid-latitude summertime convection: […]. Submitted to JAS

Stephan Rasp

- LMU Munich -

Computation section in the manuscript

e. Computational details and reproducibility

This subsection closely follows the guidelines on publishing computational results proposed by Irving (2016). The analysis and plotting of model and observation data was done using Python. The Python libraries NumPy (Numerical Python; van der Walt et al. (2011)) and SciPy (Jones et al. 2001–) were used heavily. The raw data were read with the Python module cosmo_utils (code available upon request). The figures were plotted using the Python module Matplotlib (Hunter 2007). Plotting colors were chosen according to the Hue-Chroma-Luminance color space (Stauffer et al. 2015). Some plots were post-processed using the vector graphics program Inkscape.

To enable reproducibility of the results, this paper is accompanied by a version-controlled code repository (https://github.com/raspstephan/convective_variability_analysis) and a Figshare repository (Rasp 2017), which contains a snapshot of the code repository at the time of submission and supplementary log files for each figure. These log files contain information about the computational steps taken from the raw data to the generation of the plots. While the model code and initial data is not openly available, a detailed technical description of the model simulations can be found in the `cosmo_runscripts` directory of the code repository. The Jupyter notebooks (Kluyver et al. 2016) mentioned in the text are stored in the directory `jupyter_notebooks` of the repository. Links to non-interactive versions of the notebooks can be found on the front page of the Github repository; rendered PDF versions are also added to the supplement of this paper.

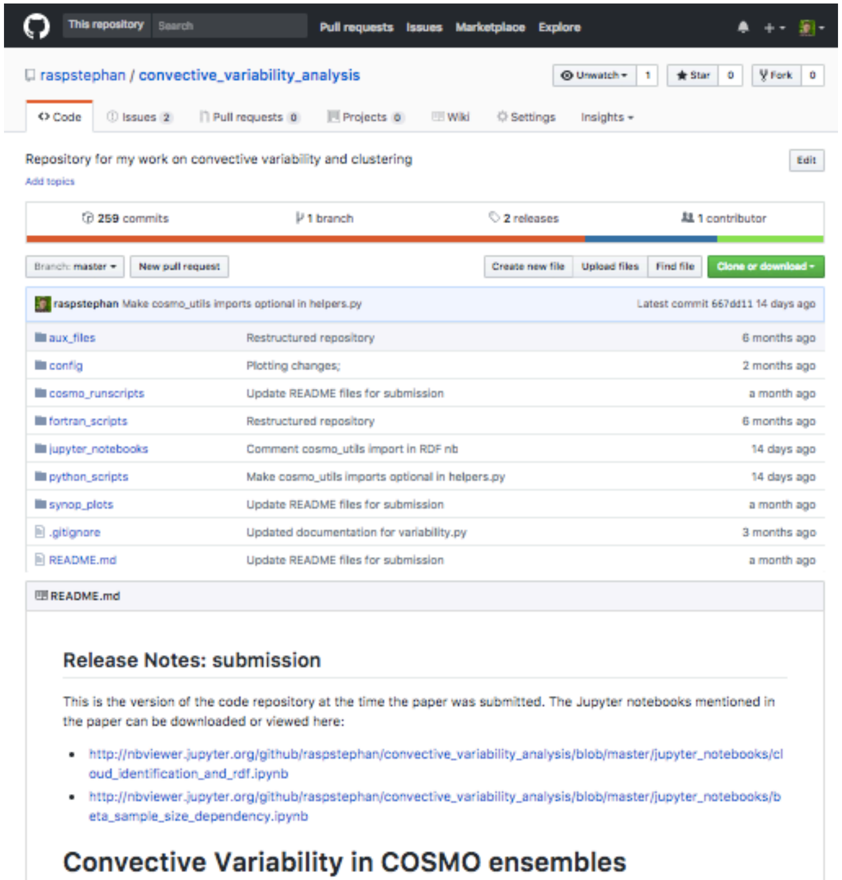
Citation of software tools

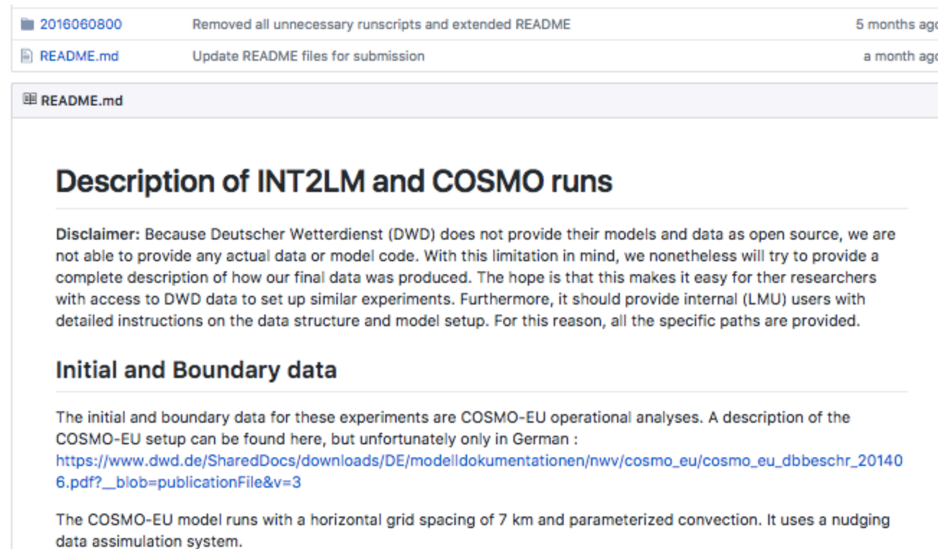
Github repository

Figshare repository with log files

Jupyter notebooks

The **Github repository** contains all code used from the raw data to the final figures with adequate documentation.

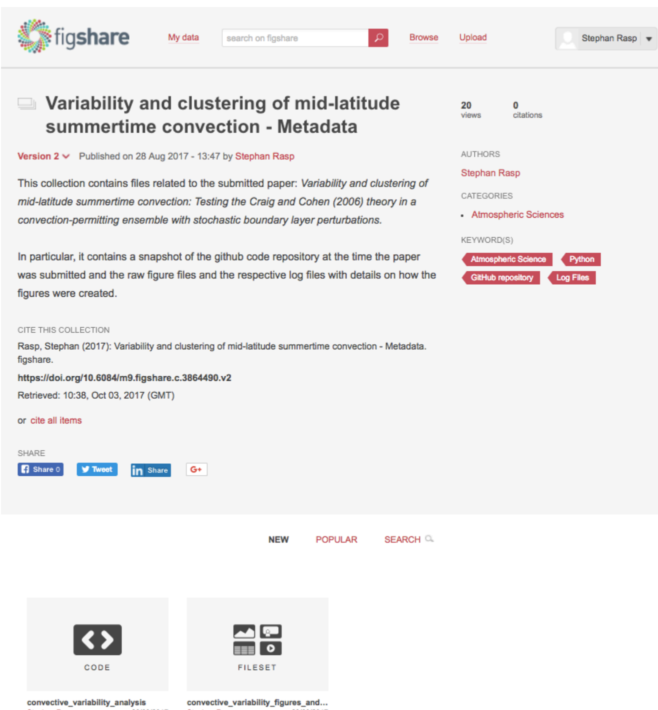




Since driving data and model source code are **not open-source** as much information as possible about the model version and how to obtain the data are given. Additionally, the location of data and model are listed for internal users.

A **Figshare repository**, which is referenced in the paper, contains a snapshot of the Github repository and the all raw figures and log files.

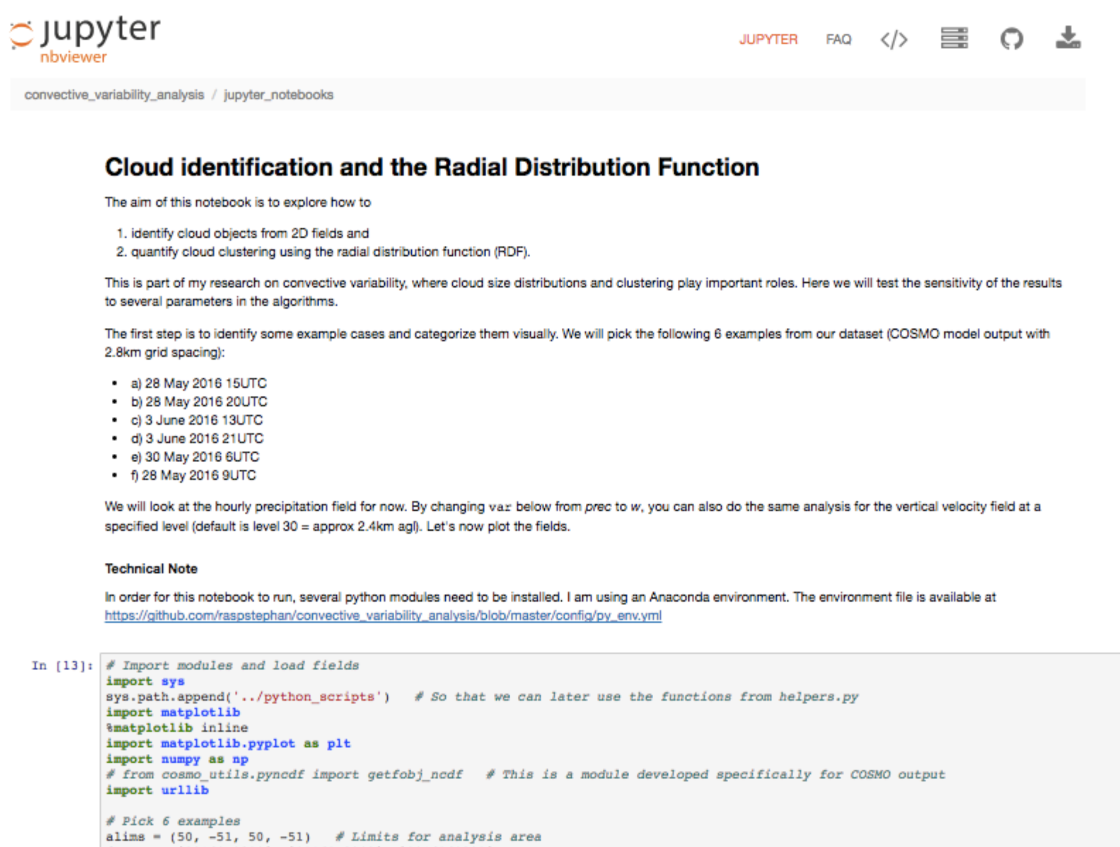
Rasp, S., 2017: Variability and clustering of mid-latitude summertime convection - metadata. Figshare, accessed 28 August 2017, doi:10.6084/m9.figshare.c.3864490.




```
Plotting log

##########################################################
Time: 2017-08-01T12:41:32

Executed command

----------------

python weather_time_series.py --date_start 2016052800 --date_end 2016060800 --time_start 1 --time_end 24 --nens 50
--pp_name all_days_50mem --plot_name all_days_50mem --radar_mask day --plot_type prec_cape_comp


in directory: /home/s/S.Rasp/repositories/convective_variability_analysis/python_scripts



Git hash: f20fb9f16b9a0a32569d1eceae8db271400e3c96



Full argparse parameters

------------------------

--plot_type prec_cape_comp
--recompute False
--time_start 1
```

The log file lists the exact command executed and the computational environment. For an example of a simple log file creation in Python: https://raspstephan.github.io/2017/08/24/reproducibility-hack.html

Jupyter notebooks allow for **literate programming**. Great for data exploration, code examples and simple analysis. In combination with a VCS literate programming can make decisions more reproducible. For R: R Markdown



jupyter nbviewer — JUPYTER FAQ

convective_variability_analysis / jupyter_notebooks

Cloud identification and the Radial Distribution Function

The aim of this notebook is to explore how to

1. identify cloud objects from 2D fields and
2. quantify cloud clustering using the radial distribution function (RDF).

This is part of my research on convective variability, where cloud size distributions and clustering play important roles. Here we will test the sensitivity of the results to several parameters in the algorithms.

The first step is to identify some example cases and categorize them visually. We will pick the following 6 examples from our dataset (COSMO model output with 2.8km grid spacing):

- a) 28 May 2016 15UTC
- b) 28 May 2016 20UTC
- c) 3 June 2016 13UTC
- d) 3 June 2016 21UTC
- e) 30 May 2016 6UTC
- f) 28 May 2016 9UTC

We will look at the hourly precipitation field for now. By changing `var` below from *prec* to *w*, you can also do the same analysis for the vertical velocity field at a specified level (default is level 30 = approx 2.4km agl). Let's now plot the fields.

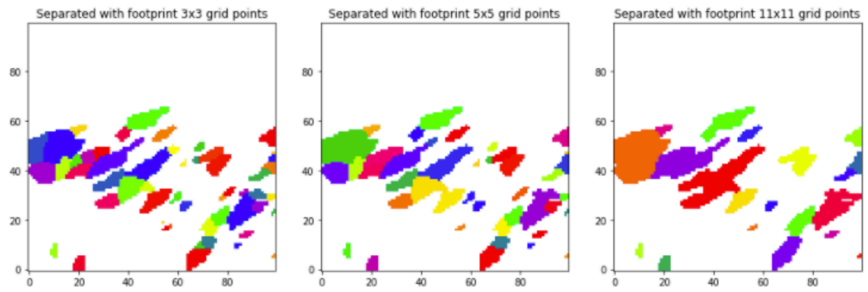
Technical Note

In order for this notebook to run, several python modules need to be installed. I am using an Anaconda environment. The environment file is available at https://github.com/raspstephan/convective_variability_analysis/blob/master/config/py_env.yml

In [13]:

```
# Import modules and load fields
import sys
sys.path.append('../python_scripts')   # So that we can later use the functions from helpers.py
import matplotlib
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
# from cosmo_utils.pyncdf import getfobj_ncdf   # This is a module developed specifically for COSMO output
import urllib

# Pick 6 examples
alims = (50, -51, 50, -51)   # Limits for analysis area
dates = ['2016052800', '2016052800', '2016060300',
```

Jupyter notebooks allow for **literate programming**. Great for data exploration, code examples and simple analysis. In combination with a VCS literate programming can make decisions more reproducible. For R: R Markdown



As we said before the unseparated objects seem too large. Using the smallest footprint (cross), however, results in many very small objects, which doesn't seem to be very realistic. Another issue is that for the larger footprints (11x11) some objects are lost even if they are not connected to other objects. This is due to the minimum distance between the local maxima set by the footprint.

Feel free to chose any of the other examples. For the scattered cases, the differences are not quite as large.

For now we choose the 3x3 footprint as a good compromise and explore the sensitivity of the RDF below.

[16]:
```
# Do the calculation with a 3x3 footprint for all examples
labels_sep_list = []
for field in field_list:
    labels_sep_list.append(helpers.identify_clouds(field, threshold, water=True, neighborhood=3)[0])
```

2. The Radial Distribution Function

The radial distribution function (RDF) measures the clustering of objects as a function of distance. It is commonly used in statistical mechanics to describe how the density of particles varies as a function of distance from a reference particle. See https://en.wikipedia.org/wiki/Radial_distribution_function

2.1 Mathematical formulation

We are computing a discrete version of the RDF in 2D. Each object is defined by its center of mass.

The normalized RDF $g(r)$ is given by

$$g(r) = \frac{\langle N(r \pm 0.5\Delta r)\rangle}{A(r \pm 0.5\Delta r)} \frac{1}{\rho}$$

where $N(r \pm 0.5\Delta r)$ is the number of objects in the interval $r \pm 0.5\Delta r$ and the angled brackets indicate the ensemble mean over all objects. $A(r \pm 0.5\Delta r)$ is the area of the annulus, which is given by $2\pi r\Delta r$. ρ is the domain mean object density.

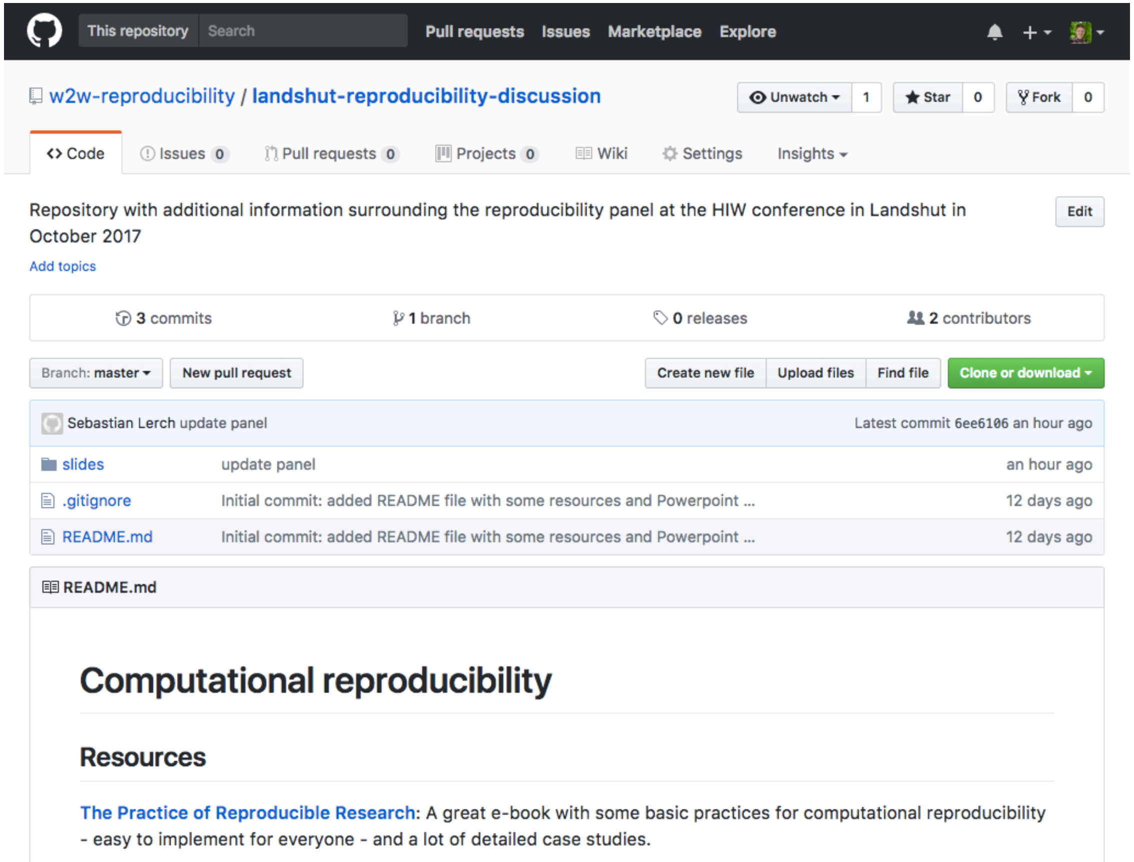
The RDF which is normalized by the domain mean density is therefore unitless and gives the following information: "How much more likely is an object located within a distance interval $[r - \Delta r, r + \Delta r]$ to another object, compared to a completely random distribution of objects in the domain?" A completely random distribution, therefore, has a value of 1 at all radii.

Final thoughts from my side:

- Few examples of reproducible research in climate and weather science.
- Initial (re)structuring of code was time-consuming.
- Striving to write reproducible code made me really think about my analysis.
- In the long run, reproducible code probably saves time, personally and for the community.
- Uncertainty how to deal with non-open-source data and model code.

Resources available at https://github.com/w2w-reproducibility/landshut-reproducibility-discussion

Link on HIW Webpage: https://hiw2017.wavestoweather.de



Panel and open discussion:

Hannah Christensen, NCAR - Julia Keller, WMO - Linus Magnusson, ECMWF - Jenny Sun, NCAR

- Why is so little research currently computationally reproducible in the weather and climate sciences?
- What are the biggest problems related to reproducibility at the moment?
- How can these problems be tackled by the community?
- Should there be strict guidelines and requirements regarding reproducibility? Who enforces them (funding agencies, journals, universities, …)?