Contents

1 Installation 3
2 Using docker-compose 5
3 Configuration 7
4 Developer Guide 9
5 Processes 11
6 Tutorials 33
7 Notebooks 35
8 Releasenotes 37
9 Sphinx AutoAPI Index 43
10 Further reading: 45
Bibliography 47
Flyingpigeon is a Web Processing Service with a collection of processes for the climate community as a compartment of the Birdhouse.

**Note:** It is one of the early birds from birdhouse and currently declared as testbed or sandbox to test new processes. Once the processes and functions are in a mature state, they move into a dedicated other or new bird. Flyingpigeon is currently the test-suite.
CHAPTER 1

Installation

The installation uses the Python distribution system Anaconda to maintain software dependencies. If Anaconda is not available, then a minimal Anaconda will be installed during the installation processes in your home directory `~/.anaconda`.

The installation process setups a conda environment named `flyingpigeon`. All additional packages and configuration files are put into this conda environment. The location is `~/.conda/envs/birdhouse`.

Now, check out the Flying Pigeon code from github and start the installation:

```
$ git clone https://github.com/bird-house/flyingpigeon.git
$ cd flyingpigeon
$ make clean install
```

After successful installation, you need to start the services. All installed files (config etc ...) are below the conda environment `birdhouse` which is by default in your home directory `~/.conda/envs/birdhouse`. Now, start the services:

```
$ make start   # starts supervisor services
$ make status # shows supervisor status
```

The deployed WPS service is available on `http://localhost:8093/wps?service=WPS&version=1.0.0&request=GetCapabilities`.

Check the log files for errors:

```
$ tail -f ~/birdhouse/var/log/pywps/flyingpigeon.log
$ tail -f ~/birdhouse/var/log/supervisor/flyingpigeon.log
$ tail -f ~/birdhouse/var/log/nginx/error.log
```

One source of error is having another server listening on the same port as flyingpigeon. Use `netstat -listen` to check if port 8093 is already in use.

For other install options, run `make help` and read the documentation for the Makefile.
Using docker-compose

Start flyingpigeon with docker-compose (docker-compose version > 1.7):

$ docker-compose up

By default the WPS is available on port 8080: http://localhost:8080/wps?service=WPS&version=1.0.0&request=GetCapabilities.

You can change the ports and hostname with environment variables:

$ HOSTNAME=flyingpigeon HTTP_PORT=8093 SUPERVISOR_PORT=48093 docker-compose up

Now the WPS is available on port 8093: http://flyingpigeon:8093/wps?service=WPS&version=1.0.0&request=GetCapabilities.
If you want to run on a different hostname or port then change the default values in custom.cfg:

```
$ cd malleefowl
$ vim custom.cfg
$ cat custom.cfg
[settings]
hostname = localhost
http-port = 8091
```

After any change to your custom.cfg you **need** to run `make update` again and restart the supervisor service:

```
$ make update  # or install
$ make restart
$ make status
```
4.1 Running WPS service in test environment

For development purposes you can run the WPS service without nginx and supervisor. Use the following instructions:

```bash
# get the source code
$ git clone https://github.com/bird-house/flyingpigeon.git
$ cd flyingpigeon

# create conda environment
$ conda env create -f environment.yml

# activate conda environment
$ source activate flyingpigeon

# install flyingpigeon code into conda environment
$ python setup.py develop

# start the WPS service
$ flyingpigeon

# open your browser on the default service url
$ firefox http://localhost:5000/wps

# ... and service capabilities url
$ firefox http://localhost:5000/wps?service=WPS&request=GetCapabilities
```

The `flyingpigeon` service command-line has more options:
For example you can start the WPS with enabled debug logging mode:

```
$ flyingpigeon --debug
```

Or you can overwrite the default PyWPS configuration by providing your own PyWPS configuration file (just modify the options you want to change):

```
# edit your local pywps configuration file
$ cat mydev.cfg
[logging]
level = WARN
file = /tmp/mydev.log

# start the service with this configuration
$ flyingpigeon -c mydev.cfg
```

Flyingpigeon provides processes for climate model data analysis, climate impact studies and investigations of extremes. A combination of processes will be called a ‘workflow’.

**Note:** Flyingpigeon is a bird to develop and test processes and functions. Once they are in a major status they move to another service.
CHAPTER 5

Processes

Warning: Due to dependency issues most of the processes are currently disabled. We are working on enabling them again. Contact a developer if you urgently need one of the following processes.

5.1 Climate indices

Climate indices are values that describe the state the climate system for a certain parameter. Climate indices as timeseries can be used to describe or estimate the change in climate over time.

The climate indices processes in flyingpigeon are based on the python package icclim. They are subclassed to:

5.1.1 Process identifiers:

- Simple indices Simple indices are based on a single input variable, and with and a simple calculation algorithm.
- Percentile indices Percentile-based indices are calculated based on an given percentile of a reference period.
  The calculation of percentile-based indices is done in two steps:
  Calculation of the percentile value for a given reference period Cumulative sum of the number of days beyond the threshold
- Multivariable indices
- User defined indices

5.1.2 Input Parameter:

for simple indices

Indice Dailypercentile
For a given percentil value (default=90) the according value will be calculated for each day in the year. The result is an annual cycle of corresponding values.

**Percentile-based indices**

<table>
<thead>
<tr>
<th>Index</th>
<th>Value</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>TG10p</td>
<td>tas</td>
<td>Days with TG &lt; 10th percentile of daily mean temperature (cold days)</td>
</tr>
<tr>
<td>TX10p</td>
<td>tasmax</td>
<td>Days with TX &lt; 10th percentile of daily maximum temperature (cold day-times)</td>
</tr>
<tr>
<td>TN10p</td>
<td>tasmin</td>
<td>Days with TN &lt; 10th percentile of daily minimum temperature (cold nights)</td>
</tr>
<tr>
<td>TG90p</td>
<td>tas</td>
<td>Days with TG &gt; 90th percentile of daily mean temperature (warm days)</td>
</tr>
<tr>
<td>TX90p</td>
<td>tasmax</td>
<td>Days with TX &gt; 90th percentile of daily maximum temperature (warm day-times)</td>
</tr>
<tr>
<td>TN90p</td>
<td>tasmin</td>
<td>Days with TN &gt; 90th percentile of daily minimum temperature (warm nights)</td>
</tr>
<tr>
<td>WSDI</td>
<td>tasmax</td>
<td>Warm-spell duration index</td>
</tr>
<tr>
<td>CSDI</td>
<td>tasmin</td>
<td>Cold-spell duration index</td>
</tr>
<tr>
<td>R75p</td>
<td>pr</td>
<td>Days with PRCPTOT &gt; 75th percentile of daily amounts (moderate wet days)</td>
</tr>
<tr>
<td>R75pTOT</td>
<td>pr</td>
<td>Precipitation fraction due to moderate wet days (&gt;75th percentile)</td>
</tr>
<tr>
<td>R95p</td>
<td>pr</td>
<td>Days with PRCPTOT &gt; 95th percentile of daily amounts (very wet days)</td>
</tr>
<tr>
<td>R95pTOT</td>
<td>pr</td>
<td>Precipitation fraction due to very wet days (&gt;95th percentile) (%)</td>
</tr>
<tr>
<td>R99p</td>
<td>pr</td>
<td>Days with PRCPTOT &gt; 99th percentile of daily amounts (extremely wet days)</td>
</tr>
<tr>
<td>R99pTOT</td>
<td>pr</td>
<td>Precipitation fraction due to extremely wet days (&gt;99th percentile)(%)</td>
</tr>
</tbody>
</table>

**for Multivariable Indices**

<table>
<thead>
<tr>
<th>Index</th>
<th>Value</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>CD</td>
<td>tas , pr</td>
<td>Days with TG &lt; 25th percentile of daily mean temperature and PRCPTOT &lt; 25th percentile of daily precipitation sum (cold/dry days)</td>
</tr>
<tr>
<td>CW</td>
<td>tas , pr</td>
<td>Days with TG &lt; 25th percentile of daily mean temperature and PRCPTOT &gt; 75th percentile of daily precipitation sum (cold/wet days)</td>
</tr>
<tr>
<td>WD</td>
<td>tas , pr</td>
<td>Days with TG &gt; 75th percentile of daily mean temperature and PRCPTOT &lt; 25th percentile of daily precipitation sum (warm/dry days)</td>
</tr>
<tr>
<td>WW</td>
<td>tas , pr</td>
<td>Days with TG &gt; 75th percentile of daily mean temperature and PRCPTOT &gt; 75th percentile of daily precipitation sum (warm/wet days)</td>
</tr>
</tbody>
</table>

Climate indices have to be calculated for a time aggregation.
<table>
<thead>
<tr>
<th>Time aggregation</th>
<th>Description</th>
<th>values per year</th>
</tr>
</thead>
<tbody>
<tr>
<td>mon</td>
<td>monthly</td>
<td>12</td>
</tr>
<tr>
<td>sem</td>
<td>seasonal</td>
<td>4</td>
</tr>
<tr>
<td>yr</td>
<td>yearly</td>
<td>1</td>
</tr>
<tr>
<td>ONDJFM</td>
<td>winter half</td>
<td>1</td>
</tr>
<tr>
<td>AMJJAS</td>
<td>summer half</td>
<td>1</td>
</tr>
<tr>
<td>DJF</td>
<td>winter</td>
<td>1</td>
</tr>
<tr>
<td>MAM</td>
<td>Spring</td>
<td>1</td>
</tr>
<tr>
<td>JJA</td>
<td>Summer</td>
<td>1</td>
</tr>
<tr>
<td>SON</td>
<td>Autumn</td>
<td>1</td>
</tr>
<tr>
<td>Jan</td>
<td>Januar</td>
<td>1</td>
</tr>
<tr>
<td>Feb</td>
<td>Februar</td>
<td>1</td>
</tr>
<tr>
<td>Mar</td>
<td>March</td>
<td>1</td>
</tr>
<tr>
<td>April</td>
<td>April</td>
<td>1</td>
</tr>
<tr>
<td>May</td>
<td>May</td>
<td>1</td>
</tr>
<tr>
<td>Jun</td>
<td>June</td>
<td>1</td>
</tr>
<tr>
<td>Jul</td>
<td>July</td>
<td>1</td>
</tr>
<tr>
<td>Aug</td>
<td>August</td>
<td>1</td>
</tr>
<tr>
<td>Sep</td>
<td>September</td>
<td>1</td>
</tr>
<tr>
<td>Oct</td>
<td>October</td>
<td>1</td>
</tr>
<tr>
<td>Nov</td>
<td>November</td>
<td>1</td>
</tr>
<tr>
<td>Dec</td>
<td>December</td>
<td>1</td>
</tr>
</tbody>
</table>

**Mosaic**

To be checked if multiple polygons should be merged into one polygon.

### 5.1.3 Output:

**tar archive** Tar archive containing all netCDF files. Subsetting is performed for each input dataset.

**netCDF example** One netCDF file is picked out as an example file to be displayed on the web mapping service.

### 5.2 Spatial Analogues

Spatial analogues are maps showing which areas have a climate today that is analogous to the future climate of a given place. This type of map can be useful for climate adaptation to see which places are coping today with specific climate conditions. For example, officials from a city located in a temperate region could learn from the experience of another city where heatwaves are a common occurrence and intervention plans have been improved over time.

Spatial analogues are estimated by comparing the distribution of climate indices computed at the target location over the future period with the distribution of the same climate indices computed over a reference period for multiple candidate regions. A number of methodological choices thus enter the computation:

- the climate indices,
- the metric measuring the difference between both distributions,
- the reference data from which to compute the indices,
- the future climate scenario to compute the target indices.

The climate indices chosen to compute the spatial analogs are usually annual values of indices relevant to the intended audience of these maps. For example, if the maps are made for the grape industry, then the climate indices used
could be the length of the frost-free season, growing degree-days, annual winter minimum temperature and annual number of very cold days \cite{Roy2017}.

The \texttt{flyingpigeon.processes.SpatialAnalogProcess} offers six distance metrics: standard euclidean distance, nearest neighbor, Zech-Aslan energy distance, Kolmogorov-Smirnov statistic, Friedman-Rafsky runs statistics and the Kullback-Leibler divergence. A description and reference for each distance metric is given in \texttt{flyingpigeon.dissimilarity} and based on \cite{Grenier2013}.

The reference data set should cover the target site in order to perform validation tests, and a large area around it. Global or continental scale datasets are generally used, but the spatial resolution should be high enough for users to be able to recognize climate features they are familiar with.

Different future climate scenarios from climate models can be used to compute the target distribution over the future period. Usually the raw model outputs are bias-corrected with the observation dataset. This is done to avoid discrepancies that would be introduced by systematic model errors. One way to validate the results is to compute the spatial analog using the simulation over the historical period. The best analog region should thus cover the target site.

The WPS process automatically extracts the target series from a netCDF file using geographical coordinates and the names of the climate indices (the name of the climate indices should be the same for both netCDF files). It also allows users to specify the period over which the distributions should be compared, for both the target and candidate datasets.

An accompanying process \texttt{flyingpigeon.processes.MapSpatialAnalogProcess} can then be called to create a graphic displaying the dissimilarity value. An example of such graphic is shown below, with the target location indicated by a white marker.

![Spatial Analog Example]

\textbf{Fig. 1}: A map of the dissimilarity metric computed from mean annual precipitation and temperature values in Montreal over the period 1970-1990.
5.2.1 References

5.3 Species Distribution Model

The processes related to species distribution models (SDM) produce data showing the growth favourability for single tree species related to climate conditions. Generally, the input data consist of tree observations (coordinate points of tree occurrences) and climate indices based on temperature at surface and precipitation. With statistical methods (general additive models [GAM]), the spatial favourability is calculated for each year of the input timeseries. The SDM processes are able to handle ensemble datasets and output a favourability dataset for each ensemble member.

5.3.1 Processing steps in an SDM experiment

- fetching tree occurrences of a specific tree species from the GBIF database
- calculation of climate indices timeseries based on climate model raw data
- generation of a presence/absence mask based on the GBIF data coordinates and the grid resolution of the climate indices data
- calculation of time mean values for the climate indices data for the given reference period
- statistical training (GAM) based on presence/absence mask and climate indices of a reference period
- calculation of favourability as yearly timeseries for each dataset based on the statistically-trained GAM

5.3.2 SDM-related processes

Besides a big ‘all-in-one’ process which contains all the analysis steps from fetching of raw data to the final output, the processes can also be run as parts. The following processes are available:

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sdm_getgbif</td>
<td>Only fetching GBIF tree occurrence data</td>
</tr>
<tr>
<td>sdm_getindices</td>
<td>Only calculation of climate indices</td>
</tr>
<tr>
<td>sdm_csvindices</td>
<td>output of sdm_getgbif and sdm_getindices as input to run an SDM experiment</td>
</tr>
<tr>
<td>sdm_csv</td>
<td>output of sdm_getgbif and raw climate model data as input to run an SDM experiment</td>
</tr>
<tr>
<td>sdm_allinone</td>
<td>All required steps are performed in this process to run an SDM process (!! time consuming !!)</td>
</tr>
</tbody>
</table>

### 5.3.3 Climate Indices for SDM:

<table>
<thead>
<tr>
<th>Index</th>
<th>Input Variable</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>TG_yr</td>
<td>tas</td>
<td>Mean of mean temperature per year</td>
</tr>
<tr>
<td>TG_AMJJAS</td>
<td>tas</td>
<td>Mean of mean temperature from April to September</td>
</tr>
<tr>
<td>TG_ONDJFM</td>
<td>tas</td>
<td>Mean of mean temperature from October to March</td>
</tr>
<tr>
<td>TG_JJA</td>
<td>tas</td>
<td>Mean of mean temperature from June to August</td>
</tr>
<tr>
<td>GD4_yr</td>
<td>tas</td>
<td>Growing degree days [sum of TG &gt;= 4 degrees] per year</td>
</tr>
<tr>
<td>TNn_yr</td>
<td>tasmin</td>
<td>Minimum of minimum temperature per year</td>
</tr>
<tr>
<td>TNn_AMJJAS</td>
<td>tasmin</td>
<td>Minimum of minimum temperature from April to September</td>
</tr>
<tr>
<td>TNn_ONDJFM</td>
<td>tasmin</td>
<td>Minimum of minimum temperature from October to March</td>
</tr>
<tr>
<td>TNn_Jan</td>
<td>tasmin</td>
<td>Minimum of minimum temperature in January</td>
</tr>
<tr>
<td>FD_ONDJFM</td>
<td>tasmin</td>
<td>Nr of frost days [tasmin &lt; 0°C] from October to March</td>
</tr>
<tr>
<td>FD_Apr</td>
<td>tasmin</td>
<td>Nr of frost days [tasmin &lt; 0°C] in April</td>
</tr>
<tr>
<td>FD_May</td>
<td>tasmin</td>
<td>Nr of frost days [tasmin &lt; 0°C] in May</td>
</tr>
<tr>
<td>FD_Jun</td>
<td>tasmin</td>
<td>Nr of frost days [tasmin &lt; 0°C] in June</td>
</tr>
<tr>
<td>CFD_ONDJFM</td>
<td>tasmin</td>
<td>Longest period of consecutive frost days from October to March</td>
</tr>
<tr>
<td>ID_yr</td>
<td>tasmax</td>
<td>Nr of ice days [tasmax &lt; 0°C] per year</td>
</tr>
<tr>
<td>SU_yr</td>
<td>tasmax</td>
<td>Summer days [tasmax &gt;= 25°C] per year</td>
</tr>
<tr>
<td>PRCPTOT_yr</td>
<td>pr</td>
<td>Precipitation total amount [sum] per year</td>
</tr>
<tr>
<td>PRCPTOT_ONDJFM</td>
<td>pr</td>
<td>Precipitation total amount [sum] in winter half</td>
</tr>
<tr>
<td>PRCPTOT_ONDJFM</td>
<td>pr</td>
<td>Precipitation total amount [sum] in winter half</td>
</tr>
<tr>
<td>PRCPTOT_MAM</td>
<td>pr</td>
<td>Precipitation total amount [sum] in spring (March to May)</td>
</tr>
<tr>
<td>PRCPTOT_JJA</td>
<td>pr</td>
<td>Precipitation total amount [sum] in summer (June to August)</td>
</tr>
<tr>
<td>RR1_yr</td>
<td>pr</td>
<td>Nr of days with precipitation &gt; 1 mm per year</td>
</tr>
<tr>
<td>CDD_AMJJAS</td>
<td>pr</td>
<td>Consecutive dry days precipitation &lt; 1 mm in summer half</td>
</tr>
</tbody>
</table>

### 5.3.4 Examples

**GBIF Data fetch**

1. Login to a Phoenix GUI (e.g. Compute provider mouflon at DKRZ)
2. Menu option: Processes
3. Choose a Web Processing Service: Flyingpigeon
4. Choose WPS Process of Flyingpigeon: SDM - Get GBIF only
5. Set the taxonomic name of the tree species of your choice and optionally reduce the region to fetch data (recommended to reduce the process time)

**Calculate climate indices**

1. Login to a Phoenix GUI (e.g. Compute provider mouflon at DKRZ)
2. Menu option: Wizard
3. Choose a Favorite: No Favorite
4. Choose WPS Process of Flyingpigeon: SDM - Get indices only
5. Choose the climate indices to be taken as input for the SDM experiment ( optionally change the archive format)
6. Choose Input Parameter of species distribution model: netCDF
7. Choose Data Source: Earth System Grid Federation (ESGF)

The next window displays the data search interface to the data available in the ESGF archive. The input files are used to calculate the climate conditions. Depending on the selection of indices, appropriate variables have to be provided (e.g. an index based on precipitation needs ‘pr’ as an input variable). Multiple selection can be done while pressing the Ctrl button. With other search options, the data selection looks like this example:

8. Follow the 'next' buttons and submit the job with ‘Done’.

Running an SDM experiment

With the outputs of the first two examples (fetching GBIF data and climate indices calculation), it is possible to run an SDM experiment.

1. Login to a Phoenix GUI (e.g. Compute provider mouflon at DKRZ)
2. Menu option: Processes
3. Choose a Web Processing Service: Flyingpigeon
4. Choose WPS Process of Flyingpigeon: SDM - csvindices
5. Resources is an archive file containing the precalculated climate indices, and GBIF data is the csv table output by the GBIF data fetch process. You can use the basket buttons to put/take the appropriate urls for the data an easy way.
6. Execute to submit the job and follow the process in the monitor.

5.4 Subsets

Generates a polygon subset of input netCDF files. Based on an ocgis call, several pre-defined polygons (e.g. world countries) can be used to generate an appropriate subset of input netCDF files.
5.4.1 Method:

Integrated ocgis performs the subsetting.

5.4.2 Process identifiers:

- `subset_continents` subsets continents
- `subset_countries` subsets countries
- `subset_europeanregions` subsets European regions
- `subset_points` extracts timeseries for given coordinate subset_points

5.4.3 Input parameter:

**Polygons** Abbreviation of the appropriate polygon.

**Mosaic** The option ‘MOSAIC’ as a checkbox allows you to decide, in the case of multiple polygon selection, if the polygons should be stitched together into one polygon (e.g. shape of Germany and France as one polygon) or calculated as separate output files.

5.4.4 Shapefile optimisation:

For optimisation of the subset process, the appropriate shapefiles are prepared as follows:

**Shapefile preparation**

This text describes how to prepare, simplify and customize shapefiles from the GADM database. We used GADM version 2.7.

Start by downloading gadm26_levels.gdb, the ESRI shapefile for the whole world that contains all six administration levels. The resulting file is a directory that can be read with qgis 2.8. (Note: for Fedora users, you must have Fedora 22 in order to upgrade to qgis 2.8).

**Open shapefile in qgis**

To open gadm26_levels.gdb in qgis, follow the steps below:

Add Vector Layer ->

--------> Source type = Directory

--------> Source:

---------------------- Type: OpenFileGDB

After selecting the directory, select “Open” and a window will appear listing the six levels. Select the level you would like. We used “adm1” which corresponds to the region level (i.e. one level smaller than country level).
Filter the countries you wish to retain

You can use qgis to select countries you wish to extract from the world shapefile. Use the “Advanced Filter (Expression)” option in the Attribute Table. For example, for a selection of EU countries, you can use this expression:

```
```

The Attribute Table will then be updated and you can choose all the rows. This selection will be displayed on the map in the main window of qgis:

You can save this selection in the format of an ESRI Shapefile:
Layer -> Save as -> Save only selected features

**Simplify using mapshaper (command line)**

Note that the resulting map is very highly resolved, and it is often necessary to simplify the lines. There is an online tool called mapshaper that we found to be very effective. It can be used both on the command line and as a web GUI.
On the command line, the default options are: Visvalingam Weighted Area, no Snap Vertices. Choose the simplification level, the input and output shapefiles. Here is an example for 1% simplification:

```bash
$ mapshaper -i adm1-EU.shp -simplify 1% -o adm1-EU-mapshaped1.shp
=> Repaired 98 intersections; unable to repair 1 intersection.
```

This produced a simplified map, shown here (purple) superimposed on the original map (blue), zoomed on the coastline of Norway:

![Simplified Map Example](image)

**Simplify using mapshaper (GUI)**

You can test the different simplify options using the mapshaper GUI instead of the command line version. Namely:

- Visvalingam Weighted Area | Effective Area
- Snap Vertices ON | OFF

Also, the GUI seems to be more successful at repairing all intersections.

The figure below shows the original (cyan), NoSnapVertices-WeightedArea (magenta), and NoSnapVertices-EffectiveArea (purple):
(There were very tiny differences between Snap Vertices vs. No Snap Vertices. Too hard to say from this example if one was better than the other).

**Customize shapefile**

The shapefile produced from the adm1 level of the ESRI shapefile as described above shows all regions of the selected countries, but when displayed on the screen, some regions were too small both visually and for the resolution of our models (~100 km):
Another issue with using the ESRI file containing all six admin levels is that there is no unique identifier column such as “HASC_1”. For example, for the region Homyel’ (with a ‘ at the end) in Belarus, “HASC_1” = “BY.HO”. Without this field, one would be forced to use “NAME_1” = “Homyel’” to identify this region, but the special character ‘ may cause problems in the python script that reads the file.

So, we need to merge the small country regions together while leaving the larger regions alone.

Steps:

1. We downloaded the GADM shapefile as a single layer (here), and the EU countries were selected as before.

   Note: HASC_1” was NULL for some countries, e.g. FRA, ITA, GBR, BEL. We manually replaced the NULLs with a unique identifier following the convention in the file.

2. We then decided which regions to merge together, and then we formed the following qgis filter expression:
3. In the main screen of QGIS, these selected countries were regrouped with respect to field “ISO” (i.e. country level):

   Vector -> Geoprocessing tools -> Dissolve -> Dissolve field = ISO

4. The other countries (whose regions are large enough to be resolved) were selected in the Attribute Table in the same way, but using ID_1 (corresponding to level adm1) as the identifier.

5. Finally, the two shapefiles were fused together:

   Vector -> Data Management Tools -> Merge shapefiles to one

6. The resulting shapefile was simplified with the mapshaper GUI at 0.1%, which can then be read into the flying-pigeon python scripts.

7. To display in the browser, the shapefile was converted to GeoJSON using ogr2ogr:

   $ ogr2ogr -overwrite -f GeoJSON output.geojson input.shp

Here is the resulting file containing region-level and country-level areas:
5.5 Mask Land / Sea

Maskes the Land (or Sea) Area of a netCDF file. The process is searching the data cache for a land_area_fraction file fitting to the file to be masked.
5.6 Regridding Operations

5.7 Climate Fact Sheet Generator
Generates PDF document with statistical values for one or multiple selected countries.

5.8 Download Resources
Downloads resources (limited to 50GB) to the local file system of the birdhouse computer provider.

5.9 Segetal flora
Species biodiversity of segetal flora. Input files: variable:tas, domain: EUR-11 or EUR-44.

5.10 Visualisation
Time series visualisation of netCDF files. Creates a spaghetti plot and an uncertainty plot.

5.11 Earth Observation Processes
Earth Observation Data (Satellite images) are stored in COPERNICUS data archive. To use the following processes an appropriate user account at scihub.copernicus.eu is required.

- Search Products With an given bounding box, time and cloud cover threshold, products will be searched. Output is a list of products as well as a graphic visualising the spatial extent of the tiles
- Fetch Products Basically like the search process but including the fetch of appropriate data into the disk system of the birdhouse provider

5.12 Moved Processes
5.12.1 Analogues of circulation
Analogues of circulation provide a versatile tool to investigate the relation between climate variables (such as temperature or precipitation) and large-scale atmospheric circulation patterns (SLP or Zg (h)). The deployed software in the analogues processes are the circulation analogue simulation tool (CASTf90).

Method:
Here should come a bit of a method description.
Process identifiers:

- **Analogues_reanalyses**: Includes a preselection of reanalyses pressure data (sea surface or geopotential height). This process fetches the data based on the selected reanalyses dataset and variable, so no input file has to be provided by the user.

- **Analogues_model**: Designed to analyse a climate model data set. The location of the input files has to be provided by the user (using the ESGF search interface is possible).

- **Analogues_compare**: To search analogue days in a climate model dataset for given days in a reanalyses dataset.

- **Analogues_viewer**: Analogues data output as a text file can be visualized in an html page with interactive graphics.

Input parameter:

- **Data experiment**: Input Data
- **Region**: Region to be analysed
- **Start data of analysis period**: Starting date of the period to be analysed
- **End date of analysis period**: End date of the period to be analyzed
- **Start reference period**: Start date of period where analogue days will be picked
- **End reference period**: End date of period where analogue days will be picked
- **Nr of analogues**: Number of analogues to be detected and written out as results. They are ordered with increasing distance from the original pressure pattern.
- **Seasonal window**: Number of days defining the calendar proximity, i.e., the time window around the given date in all years except the year of the given day.
- **normalisation**: Pressure values can be normalized (subtraction of average value over the whole period). Possible options are:
  - **None**: No normalisation
  - **based**: normalisation based on reference period
  - **sim**: normalisation based on analysis period
  - **own**: normalisation of reference and analysis data values by their own average
- **Distance**: Methods to calculate the distance:
  - **euclidean**
  - **mahalanobis**
  - **cosine**
  - **of**
- **output file format**
  - **netCDF**: output values will be provided as a netCDF file
  - **ascii**: output values will be provided as an ascii file
- **Time window**: Values of the analysis period can be smoothed by averaging with the values of the following days given in 'time window' (default = 1).
Outputs:

**Config File**  configuration file used by the CASTf90 software

**Analogs File**  list of analogue days

**Target netCDF**  pressure values as input for CASTf90 for analogue days to be picked

**prepared netCDF**  pressure values as input for CASTf90 for analogue days to be searched

**html viewer**  output of the analogues_viewer process for data visualisation

**modified analogues txt file**  output analogue days formatted for the analogues_viewer process

Examples:

### GUI for analogues_reanalyses

Picks dates of analog pressure patterns in a reference period for a specific time slice.

- Login to a Phoenix GUI (e.g. Compute provider DKRZ)
  
  (Detailed descriptions for login options)
  
  - Menu option: Processes
  - Web Processing Services: Flyingpigeon
  - Processes: Days with analog pressure pattern
  - choose the appropriate input parameter

(a resource is not necessary, NCEP data are fetched within the process)
5.12.2 Weather Regimes

Calculation of weather regimes based on pressure patterns (k-means clustering method). The processes performs a pattern classification for observations data (NCEP) as well as model data. Both results are compared.

Method:

- fetching observation data
- fetching model data
- subset the selected geographical region
- selection of month to be analyzed
- unit conversion to hPa (if necessary)
- regridding (bilinear) to the grid of the observation data (if necessary)
- computing of principal components for dimension reduction

Process identifiers:

- `weatherregimes_reanalyses` to perform weather regime cluster analyses on pre-defined datasets of weather regimes
- `weatherregimes_models` to perform weather regime cluster analyses on climate model data
- `weatherregimes_projection` project trained weather regime clusters on a second dataset

Input Parameter:

- `resources` (links to netCDF sea surface pressure data) or search with phoenix
- `dataset` NCEP slp data (automatic fetch)
- `Region` Region for weather regimes classification specified by coordinate bounding box
- `Nr. of clusters` defines the number of weather regimes to be detected

Outputs:

- pressure pattern graphic
- R workspace
- PCA data
- Frequency

Post-processing:

The weather regime process provides a map of the detected pressure patterns according to the centroid. This is a default graphic and might not satisfy the needs of the user. But the process also provides the entire workspace containing all required values to generate an individual plot according to the needs of the user. Download the R-Workspace (output of the weather regime process) to your local machine by clicking the download button or run e.g.:

This can be read with R syntax. Here is an example to generate an individual plot:

```r
library(maps)
load("output_classification-e49f2d78-8563-11e6-bf14-fbeae168c26e.Rdat")
#save as pdf
pdf(file="output_graphics.pdf", width=14, height=7)
#save as eps
#postscript(file="output_graphics.eps",width=1400,height=700)
# colorscale
bluered=colorRampPalette(c("darkblue","blue","lightblue","white","white","pink","red ...
"darkred"))
zcol=c(round(min(dat.class$reg.var/100)),round(max(dat.class$reg.var/100)))
zlev=seq(zcol[1],zcol[2])
for(i in 1:nreg){
  champ=dat.class$reg.var[,i]/100
  par(mar=c(3,3,2,1.5))
  dum=t(matrix(champ, length(lat), length(lon)))
  lat.sort=sort(lat,index.return=TRUE)
  titleplot=paste("WR: ",i," (",format(dat.class$perc.r[i],digits=3), ")")
  filled.contour(lon,sort(lat),dum[,lat.sort$ix],color.palette =bluered,
                asp = 0,nlevels=length(zlev),levels=zlev,
                plot.title = title(main = titleplot, xlab = "lon", ylab = "lat"),
                plot.axes={axis(1); axis(2);
                            contour(lon,sort(lat),dum[,lat.sort$ix],add=T, nlevels=6,lwd=2);
                            library(fields)
                            world(xlim=range(lon),ylim=range(lat),interior=FALSE,add=TRUE)
                }
}
# end i
dev.off() #eps or pdf
```

Examples:

GUI for Weather regimes

Weather regimes comparison between NCEP and CMIP5

- Login to a Phoenix GUI (e.g. Compute provider mouflon at DKRZ)
- Menu option: Wizard
- Choose a Favorite: No Favorite
- Choose a Web Processing Service: Flyingpigeon
- Choose WPS Process of Flyingpigeon: Weather Regimes
- Literal inputs of Weather Regimes: defaults given, feel free to change ;-)  
- Choose Input Parameter of Weather Regimes: netCDF
- Choose Data Source: Earth System Grid Federation (ESGF)

The next window is the data search interface to the data available in the ESGF archive. Weather regimes are computed based on sea surface pressure values. The appropriate variable for CMIP5 data is ‘psl’. With other search options, the datas election look like this example:
You can save your settings as a favorite. And submit the job. Done!!!

You can follow the log file in the monitor (click on the job ID e.g. a4aa98de-ffde-11e5-b50a-bb0d01b14483). Manual reload of your browser site is necessary.

5.12. Moved Processes
Processes in Flyingpigeon can be executed in various ways:

- with terminal command line using birdy
- within a script in a language such as python
- ref via the web browser GUI Phoenix:
  ref Tutorials to get familiar with general usage of Phoenix
6.1 Tutorial: Ensemble Robustness

Processing files of the local (compute providers) disc system with solr - search:

Note for administrators: To index files run: birdfeeder from-walker –start-dir /path/to/local/folder/

- Login to a Phoenix GUI (e.g. Compute provider DKRZ) (Detailed descriptions for login options)
- Menu option: Wizard
- Choose a Favorite: No Favorite
- Choose a Web Processing Service: Flyingpigeon
- Choose WPS Process of Flyingpigeon: Weather Regimes
- Literal inputs of Weather Regimes: defaults given
- Choose Input Parameter of Weather Regimes: netCDF
- Choose Data Source: Birdhouse Solr Search

The open window is the data search interface to the available local data of the compute provider. Weather regimes are computed based on sea surface pressure values. Appropriate variables are ‘psl’ or ‘slp’. In this example, search for ‘psl’:

![solr_search_psl.png](tutorials/pics/solr_search_psl.png)

You can save your settings as a favorite. And submit the job. Done!!!

You can follow the log file in the monitor (click on the job ID e.g. a4aa98de-ffde-11e5-b50a-bb0d01b14483). Manual reload of your browser site is necessary.
tutorials/pics/monitor_log_weatherregimes.png
CHAPTER 7

Notebooks

- WPS Ensemble Robustness with local files
- Weatherregimes
8.1 1.2.1 (2018-09-14)

Bug-fix release:
- disabled many processes due to conda dependency conflicts (#261).
- simplified buildout.cfg (#245).
- tests for subset_countries added (#237).
- numerous others fixes.

8.2 1.2.0 (2018-04-04)

Issues:
- Fixed abstract for CSV files output in pointinspection process: #216
- snappy installation is optional: #229
- Disabled sphinx buildout configuration: #227
- Fixed test failures: #210 and #224
- Fixed codacy report: #211
- Fixed readthedocs build: #207

8.3 1.1.0 (2017-12-22)

- disabled analogs processes (using castf90) … moved to black-swan.
- added new spatial analogs process.
• added initial version of satellite processes using scihub.coperniucs data.
• updated weatherregimes processes.

8.4 1.0.3 (2017-12-21)
• fixed sphinx build.

8.5 1.0.2 (2017-12-20)
• updated conda environment.
• fixed pytest configuration.
• updated travis link in Readme.

8.6 1.0.1 (2017-11-14)
• fixed version number
• fixed changes formatting
• display version number in service title

8.7 1.0.0 (2017-11-01)
• code adapted to pywps4
• ocgis v2 deployed
• Tests for components
• Version published in Computers & Geosciences

8.7.1 Set of processes

Base processes:
• Fetch resources
• Fetch GBIF Species Coordination
• Subset Polygons
• Point Inspection
• Timeseries visualisation
• Climate Indices Calculation

Climate Impact:
• Species Distribution Model
• Segetal Flora Calculation
Extreme Weather Events Assessment:

- Analogs of Circulation for reanalyzes Datasets
- Analogs of Circulation for model data
- Analogs of Circulation Comparison between reanalyzes and climate model data
- Analogs output data visualisation
- Weather regime Determination for reanalyzes Datasets
- Weather regime Determination for model datasets
- Weather regime projections (based on previous analyses)

**8.8 0.11.0 (2017-07-11)**

converted processes to pywps-4 from next:

- subsetting countries, continents and european regions
- climate indices (daily percentiles, single variable)
- species distribution model
- land-sea mask
- point inspection
- fetch resources

**8.9 0.10.1 (2017-07-11)**

- disabled bbox parameter . . . needs to be fixed in OWSLib.
- updated titles of analogs processes.
- updated version in docs.
- disabled wps_gbiffetch test . . . was stalled.

**8.10 0.10.0 (2017-07-10)**

- Translate code pywps4 conform
- Climate indices dailypercentile
- Climate Fact sheet Generator
- R plot for SDM response cuvres running under CentOS
- Species distribution model Processes modularized in five processes
- Direction switch for analogs comparison process
8.11 0.9.1 (2016-11-16)

- modularisation of segetalflora process
- docker update

8.12 0.9.0 (2016-09-08)

- Subset points
- Subset European regions
- Subset world countries
- Subset continents
- Analogues for reanalyses datasets
- Analogues for model datasets
- Analogues for comparison model to reanalyses datasets
- Species Distribution Model based on GBIF CSV file
- Species Distribution Model with GBIF search included
- Weather regimes for reanalyses datasets
- Weather regimes for model datasets
- Weather regimes for model datasets with centroids trained on reanalyses datasets
- Segetalflora
- Initial spatial analogues process
- Climate indices (simple)
- Climate indices (percentile-based)
- Download resources
- Initial ensembles robustness
- Plots for time series

8.13 0.2.0 (2016-07-15)

- analogs detection and viewer.
- timeseries plot.
- indices calculation with iclim.
- subsetting for countries and regions.
- weather regimes.
- SDM: species distribution model for tree species based on GBIF.
- species biodiversity of segetal flora.
8.14 0.1.0 (2014-09-04)

Paris Release

• moved code to github.
• Initial Release.
CHAPTER 9

Sphinx AutoAPI Index

This page is the top-level of your generated API documentation. Below is a list of all items that are documented here.
Further reading:

**Flying Pigeon (the bird)**  The pigeon finds its way home over extremely long distances. [...]. (Wikipedia).

**Flying Pigeon (the bike)**  Flying Pigeon is a Chinese bicycle company [...]. The Flying Pigeon is the most popular vehicle ever. (Wikipedia)