hotspell

Release 0.1.4

Ilias Agathangelidis

Jul 04, 2023

CONTENTS:

CHAPTER

ONE

ABOUT

Hotspell is a Python package that detects past heat wave events using daily weather station data of minimum and maximum air temperature. The user can choose between a range of predefined threshold-based and percentile-based heat wave indices or alternatively can define a full customizable index.

The main output of hotspell are the dates and characteristics of heat waves found within the study period, stored in a pandas DataFrame. If selected by the user, summary statistics (i.e. annual metrics) of the heat wave events are also computed.

CHAPTER

TWO

QUICK START

1. Import the hotspell package

```
import hotspell
```

2. Choose the heat wave index CTX90PCT

```
index_name = "ctx90pct"
hw_index = hotspell.index(name=index_name)
```

3. Set your data path of your CSV file

```
mydata = "my_data/my_file.csv"
```

The CSV file should include the following columns

- Year
- Month
- Day
- Tmin
- Tmax

in the above order, **without** a header line. Each day should be in a seperate line; missing days/lines are allowed.

For example:

1999	8	29	23.2	37.1
1999	8	31	24.1	37.7
…	…	…	…	…

4. Find the heat wave events

```
hw = hotspell.get_heatwaves(filename=mydata, hw_index=hw_index)
heatwaves_events = hw.events
heatwaves_metrics = hw.metrics
```

CHAPTER

THREE

ACKNOWLEDGEMENTS

Hotspell is developed during research under the Greek project *National Network for Climate Change and its Impact*, CLIMPACT.

CHAPTER

FOUR

LICENSE

Hotspell is licensed under the BSD 3-clause license.

GitHub: http://github.com/agathangelidis/hotspell

4.1 Installing

4.1.1 Dependencies

The required dependencies of hotspell are:

- NumPy
- pandas

It is recommended to use the Anaconda Python distribution to ensure that the above dependencies are properly installed, before installing hotspell.

If a new conda environment or Miniconda is used, the dependencies can be easily installed using the conda package manager:

```
conda install numpy pandas
```

4.1.2 Installing with pip

Hotspell is available on PyPI and can be installed using the pip package manager:

```
pip install hotspell
```

4.2 Tutorial

4.2.1 Preprocessing of daily observations

First, we need to obtain some daily weather observations that we will use to detect heat wave events.

We will use a CSV file with daily historical climatic data (1901 - 2020) from the Thissio station (Athens, Greece) of the National Observatory of Athens (Founda, 2011, Founda et al., 2013). The dataset is puclic under CC-BY-SA 4.0.

After downloading our file, we should inspect its data and preprocess it so that it can be used with hotspell.

[1]:
```
import pandas as pd

raw_data = "/my_path/hcd_noa.csv"  # Replace with your path

df = pd.read_csv(raw_data)
df.head()
```

[1]:

	YEAR	MONTH	DAY	Tmax (oC)	Tmin (oC)	RH (%)	Rain (mm)
0	1901	1	1	14.3	5.9	67.0	0.2
1	1901	1	2	15.0	8.9	80.0	5.8
2	1901	1	3	10.3	4.8	76.0	5.0
3	1901	1	4	7.1	4.8	78.0	3.2
4	1901	1	5	10.4	5.2	83.0	6.6

As we can see, the dataset includes 7 columns.

The first 3 columns correspond to the year, month and day of the observations, columns 4 and 5 to the daily maximum (Tmax) and daily minimum (Tmin) air temperature (in °C), column 6 to to relative humidity (RH, expressed as percent) and the last column to precipitation (Rain, in mm).

We need to drop humidity and rain and rearrange Tmax and Tmin:

[2]:
```
df = df[["YEAR", "MONTH", "DAY", "Tmin (oC)", "Tmax (oC)"]]
df.head()
```

[2]:

	YEAR	MONTH	DAY	Tmin (oC)	Tmax (oC)
0	1901	1	1	5.9	14.3
1	1901	1	2	8.9	15.0
2	1901	1	3	4.8	10.3
3	1901	1	4	4.8	7.1
4	1901	1	5	5.2	10.4

Suppose that there are nodata values included in the timeseries which have been set as -9999.

We should delete these measurements:

[3]:
```
nodata_value = -9999
df = df.loc[
    (df["Tmin (oC)"] != nodata_value ) & (df["Tmax (oC)"] != nodata_value)
]
```

Now, we are ready to save our processed file; we must not write out the header and the index of the DataFrame.

[4]:
```
processed_data = "/my_path/hcd_noa_processed.csv"  # Replace with your path

df.to_csv(processed_data, header=False, index=False)
```

4.2.2 Detect heat waves

We have finished the preprocessing of our data and we are now ready to use hotspell.

Create a heat index

First we must initialize the heat wave index we want to use. For this first example we are going to use the index CTX90PCT.

[5]:
```
import hotspell

index_name = "ctx90pct"
ctx90pct = hotspell.index(name=index_name)
```

This index uses as a threshold the calendar day 90th percentile value of the maximum temperature based on a 15-day moving window. A heat wave occurs when the threshold is exceeded for at least 3 consecutive days.

For the complete list of the available heat wave indices to use see *here*.

Find heat wave events and compute annual metrics

Using our heat wave index and the output CSV from the first part of the tutorial we will find the heat wave events for the Thissio station.

[6]:
```
hw = hotspell.get_heatwaves(filename=processed_data, hw_index=ctx90pct)
```

Above we used the default arguments for the parameters of get_heatwaves:

- the base period, used to calculate the percentile values, was 1961 to 1990
- we limited our interest only to months June to August
- we chose to compute the annual metrics and to export our results in csv files

For a list of available choises see the documentation for *hotspell.get_heatwaves*.

Let's examine our results.

The `hw.events` attribute is a DataFrame that contains the dates of detected heat wave events, as well as their basic characteristics (duration and temperature statistics).

[7]:
```
hw.events.head()
```

[7]:
```
           begin_date   end_date  duration  avg_tmax  std_tmax  max_tmax
index
1901-08-01 1901-08-01 1901-08-03         3      37.9       0.8      38.8
1902-07-22 1902-07-22 1902-07-24         3      38.2       1.8      40.3
1903-08-14 1903-08-14 1903-08-16         3      36.0       0.2      36.2
1904-08-09 1904-08-09 1904-08-12         4      36.5       0.3      36.8
1905-08-26 1905-08-26 1905-08-31         6      36.5       0.9      38.0
```

[8]:
```
hw.events.describe()
```

[8]:
```
         duration    avg_tmax    std_tmax    max_tmax
count  198.000000  198.000000  198.000000  198.000000
mean     4.717172   36.745455    1.121212   38.145455
```

(continues on next page)

(continued from previous page)

```
std      2.172932    1.538327    0.637774    2.020794
min      3.000000   32.500000    0.100000   32.800000
25%      3.000000   36.000000    0.700000   36.900000
50%      4.000000   36.950000    1.000000   38.000000
75%      6.000000   37.600000    1.400000   39.200000
max     13.000000   41.000000    3.600000   44.800000
```

From the above we see that the CTX90PCT index resulted in 198 heat wave events between 1901 and 2020, with an average duration of nearly 5 days and an average temperature of 36.7 °C.

The `hw.metrics` attribute is a DataFrame with the annual heat waves properties.

- hwn: number of events
- hwf: number of days
- hwd: duration of longest event
- hwdm: mean duration of events
- hwm: mean normalized magnitude
- hwma: mean absolute magnitude
- hwa: normalized magnitude of hottest day
- hwaa: absolute magnitude of hottest day

For a more detailed description of heat waves metrics see the *documentation*.

[9]:
```
hw.metrics.head()
```

[9]:

year	hwn	hwf	hwd	hwdm	hwm	hwma	hwa	hwaa
1901	1	3	3.0	3.0	7.2	38.8	7.2	38.8
1902	1	3	3.0	3.0	8.7	40.3	8.7	40.3
1903	1	3	3.0	3.0	4.6	36.2	4.6	36.2
1904	1	4	4.0	4.0	5.2	36.8	5.2	36.8
1905	1	6	6.0	6.0	6.4	38.0	6.4	38.0

Detect heat waves using a custom index

Let's repeat the procedure devising a custom index, that we will call *extreme*, that aims to capture only the most severe cases of heat. We can define the heat waves under this index as the period of at least 4 concecutive days with maximum temperatures above 40 °C.

[10]:
```
extreme = hotspell.index(
    name="extreme",
    var="tmax",
    fixed_thres=40,
    min_duration=4
)

hw_extreme = hotspell.get_heatwaves(filename=processed_data, hw_index=extreme)
```

We see that at this extreme case study only two events satisfied the heat wave criteria.

[11]:
```
hw_extreme.events
```

[11]:
```
           begin_date   end_date  duration  avg_tmax  std_tmax  max_tmax
index
1987-07-21 1987-07-21 1987-07-27         7      41.6       0.7      42.8
2007-07-22 2007-07-22 2007-07-25         4      41.4       0.5      41.9
```

4.3 Heat wave indices

Metric	Name in hotspell	Description	Reference
CTN90PCT	ctn90pct	Tmin > calendar day 90th pt, 15-day window, at least 3 days	Perkins & Alexander 2013
CTN95PCT	ctn95pct	Tmin > calendar day 95th pt, 15-day window, at least 3 days	
CTX90PCT	ctx90pct	Tmax > calendar day 90th pt, 15-day window, at least 3 days	Perkins & Alexander 2013
CTX95PCT	ctx95pct	Tmax > calendar day 95th pt, 15-day window, at least 3 days	
Hot days	hot_days	Tmax > 35 °C	Collins et al. 2000
Hot events (day)	hot_events_daytime	Tmax > 35 °C, at least 3 to 5 days (default 3 days in hotspell)	Collins et al. 2000
Hot events (night)	hot_events_nighttime	Tmin > 35 °C, at least 3 to 5 days (default 3 days in hotspell)	Collins et al. 2000
SU (summer days)	summer_days	Tmax > 25 °C	Alexander et al. 2006
TN90P	tn90p	Tmin > calendar day 90th pt, 5-day window	Alexander et al. 2006
TR (tropical nights)	tropical_nights	Tmin > 20 °C	Alexander et al. 2006
TX90P	tx90p	Tmax > calendar day 90th pt, 5-day window	Alexander et al. 2006
WSDI	wsdi	Tmax > calendar day 90th pt, 5-day window, at least 6 days	Alexander et al. 2006

4.4 Heat wave metrics

Metric	Description	Reference
HWN (Heat wave number)	The annual number of heat wave events	Fischer & Schär 2010
HWF (Heat wave frequency)	The sum of participating heat wave days per year	Fischer & Schär 2010
HWD (Heat wave duration)	The length (in days) of the longest yearly event	Fischer & Schär 2010
HWDM (Heat wave duration mean)	The average length (in days) of heat wave events per year	Fischer & Schär 2010
HWM (Heat wave magnitude)	The average daily magnitude across all heat wave events within a year, expressed as the anomaly against the mean Tmin or Tmax	Perkins & Alexander 2013
HWMA (Heat wave magnitude absolute)	The average daily magnitude across all heat wave events within a year	
HWA (Heat wave amplitude)	The hottest day of the hottest yearly event, expressed as the anomaly against the mean Tmin or Tmax	Fischer & Schär 2010
HWAA (Heat wave amplitude absolute)	The hottest day of the hottest yearly event	

4.5 hotspell

4.5.1 hotspell package

Submodules

hotspell.heatwaves module

class hotspell.heatwaves.**HeatWaves**(*events*, *metrics*)

Bases: object

Class designed for storing heat wave events.

It is the holder for the output of *get_heatwaves*.

Parameters

- **events** (*DataFrame*) – It contains the dates of detected heat wave events, as well as their basic characteristics (duration and temperature statistics).
- **metrics** (*DataFrame*) – It contains the summary of heat waves per year via standard metrics. Years with no heat waves are distinguished from years with missing data.

Notes

Column names of metrics correspond to:

hwn
: [Heat wave number] The annual total sum of heat wave events

hwf
: [Heat wave day frequency] The annual total sum of heat wave days

hwd
: [Heat wave duration] The length of the longest heat wave per year

hwdm
: [Heat wave duration (mean)] The average length of heat waves per year

hwm
: [Heat wave magnitude] The average magnitude of all events (anomaly against seasonal mean)

hwma
: [Heat wave magnitude (absolute value)] The average magnitude of all events

hwa
: [Heat wave amplitude] The hottest day of hottest event per year (anomaly against seasonal mean)

hwaa
: [Heat wave amplitude (absolute value)] The hottest day of hottest event per year

`hotspell.heatwaves.`**`get_heatwaves`**(*filename*, *hw_index*, *ref_years=('1961-01-01', '1990-12-31')*, *summer_months=(6, 7, 8)*, *max_missing_days_pct=10*, *export=True*, *metrics=True*)

Detect heat wave events from weather station data.

Parameters

- **filename** (*str or path object*) – The path of the csv file that contains the weather data. It requires specific columns to be included in the csv file in a specific order.
- **hw_index** (HeatWaveIndex) – An HeatWaveIndex object created using the *index* function.
- **ref_years** (*tuple of str, default ("1961-01-01", "1990-12-31")*) – The first and the last year of the reference period. It should be set using the "YYYY-MM-DD" format.
- **summer_months** (*tuple of int or None, default (6, 7, 8)*) – A tuple with all months of the summer period. For the southern hemisphere it should be set as (12, 1, 2) or similar variants.
- **max_missing_days_pct** (*int, default 10*) – The percentage of maximum missing days for a year to be considered valid and be included in the metrics. If a summer period has been defined the percentage corresponds only to this period.
- **export** (*bool, default True*) – If True, output is exported as csv files in the same folder as the input data.
- **metrics** (*bool, default True*) – If True, annual metrics are computed and are exported if *export=True*.

Return type
HeatWaves object

hotspell.indices module

class hotspell.indices.**HeatWaveIndex**(*name*, *var*, *pct*, *fixed_thres*, *min_duration*, *window_length*)

Bases: object

A class used to represent a heat wave index.

name

The name of the index. For predefined indices it follows the naming conventions of Perkins & Alexander (2013)

Type
str

var

The meteorological variable.

Type
str, one of "tmin" or "tmax"

pct

The percentile used as a threshold.

Type
int

fixed_thres

The absolute threshold of the meteorological value. If both pct and fixed_thres are set, pct has precedence over fixed_thres.

Type
int or float

min_duration

The minimum number of consecutive days should last so that a warm event is considered a heat wave.

Type
int

window_length

The total number of days that a moving window has when computing the percentile value for each day.

Type
int

hotspell.indices.**index**(*name=None*, *var=None*, *pct=None*, *fixed_thres=None*, *min_duration=None*, *window_length=None*)

Create a predefined or custom HeatWaveIndex object.

Parameters

- **name** (*str*) – The name of the index. For predefined indices it follows the naming conventions of Perkins & Alexander (2013)
- **var** (*str*) – The meteorological variable.
- **pct** (*int*) – The percentile used as a threshold.
- **fixed_thres** (*int or float*) – The absolute threshold of the meteorological value. If both pct and fixed_thres are set, pct has precedence over fixed_thres.

- **min_duration** (*int*) – The minimum number of consecutive days should last so that a warm event is considered a heat wave.
- **window_length** (*int*) – The total number of days that a moving window has when computing the percentile value for each day.

Return type
HeatWaveIndex object

Module contents

4.6 References

Alexander, L. V., Zhang, X., Peterson, T. C., Caesar, J., Gleason, B., Klein Tank, A. M. G., … & Vazquez-Aguirre, J. L. (2006). Global observed changes in daily climate extremes of temperature and precipitation. Journal of Geophysical Research: Atmospheres, 111(D5).

Collins, D. A., Della-Marta, P. M., Plummer, N., & Trewin, B. C. (2000). Trends in annual frequencies of extreme temperature events in Australia. Australian Meteorological Magazine, 49(4), 277-292.

Fischer, E. M., & Schär, C. (2010). Consistent geographical patterns of changes in high-impact European heatwaves. Nature geoscience, 3(6), 398-403.

Perkins, S. E., & Alexander, L. V. (2013). On the measurement of heat waves. Journal of climate, 26(13), 4500-4517.

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

h

INDEX