demand-model-delta

August 9, 2024

In this Jupyter notebook, the role of temperature delta-shift is explored in the context of modelling infrastructure electricity load.

First, a 'standard' detla-shift is calculated by adding constant offsets uniformly to the temperature timeseries thereby simulating conditions of a globally uniform temperature rise of 1.5C, 2.0C, etc.

Second, a 'ppe' delta-shift is performed. This novel approach relies on pre-calculated values of regional climate response to global climate change, taken from UKCP18 RCM and GCM simulations respectively. For each perturbed physics ensemble (ppe) member of UKCP18, the delta-shift is performed to re-analysis data using the regional climate response (eg. 1.443C shift with respect to 1980-2017 baseline at 2.0C global warming, 2.29C shift at 3.0C, etc.).

A weather-driven model of infrastructure electricity load is taken from **Fallon et al. (2023)** and accompanying dataset **Fallon et al. (2024)**.

In this notebook you will:
* Read the reanalysis temperature datasets
* Define climate threshold scenarios and perform delta-shift
* Create a demand timeseries and save outputs

If you benefit from using these reanalysis-driven models of infrastructure electricity demand, please consider adding a citation to our research paper **Fallon et al. (in preparation)**.

1 Jupyter notebook setup and configuration

Import required python libraries

[1]:
```
# Access system functions (used to import local modules)
import sys
from os.path import exists
from datetime import datetime

# Maths
import numpy as np

# Stats / read CSV files
import pandas as pd

# Multi-dimensional data structures
import xarray as xr
```

```
# Hide Warning Messages
import warnings
warnings.filterwarnings('ignore')
```

Import local python files from `modules` folder:

[2]:
```
sys.path.append("modules")

# global temperature threshold scenarios (organise data into convenient␣
 ↪structures for manipulating within the scenario windows)
from scenarios import calculate_hadcrut_reference,␣
 ↪calculate_reanalysis_detrended, gen_stoch_series
```

Compression to use when writing datasets

[3]:
```
encoding_options = dict(zlib=True, complevel=5)
```

2 Datasets

`CSV` files are read using the `pandas` library.

Input timeseries data is stored in `temperature/` and model coefficients are stored in `models/`.

[4]:
```
# read CSV files
merra2_df = pd.read_csv("temperature/merra2-GB-Land-daily.csv",␣
 ↪index_col="time", parse_dates=["time"])
merra2G_df = pd.read_csv("temperature/merra2-Global-Land-monthly.csv",␣
 ↪index_col="time", parse_dates=["time"])

# save indexes for later use
merra2_cal = merra2_df.index
merra2G_cal = merra2G_df.index
```

2.0.1 Convert to `xarray`

Among other features, datasets can be configured with new co-ordinates by using `xarray`.

[5]:
```
# convert reanalysis to xarray
merra2 = merra2_df.to_xarray().temperature
merra2G = merra2G_df.to_xarray().temperature
```

[6]:
```
# create new "index_year" dimension (December to November year)

# merra2 GB (daily)
idx_december = (merra2_cal.month > 11).astype(int)
idx = merra2_cal.year + idx_december
merra2 = merra2.assign_coords(index_year=("time", idx))
```

```
# merra2 global (monthly)
idx_december = (merra2G_cal.month > 11).astype(int)
idx = merra2G_cal.year + idx_december
merra2G = merra2G.assign_coords(index_year=("time", idx))
```

3 Climate Threshold Scenarios

Perform de-trending (remove climate change signal from start/end of timeseries window). Additionally, using HadCRUT5 as a reference for pre-industrial temperatures, calculate the periods that achieve relative warming of 1, 1.5, 2, 3, 4C.

[7]:
```
# calculate reference warming from pre-industrial levels to period 1980-2018
hadcrut_ref_period = slice("1980", "2018")
hadcrut_ref = calculate_hadcrut_reference(hadcrut_ref_period)

# define global deltas (present and future warming amounts)
deltas = np.array([hadcrut_ref, 1., 1.5, 2., 3., 4.])
deltas_relative = deltas - deltas[0]
```

3.1 De-trended reanalysis

Timeseries detrended (to remove ~0.2C/decade climate change signal from 1980-2018).

[8]:
```
# load from file if available
# WARNING: If you require a new copy of merra2_detrended
#          (eg. because you changed some of the inputs above)
#          then please delete/rename "merra2_detrended.nc"
fpath_merra2_detrended = "outputs/temperature/merra2_detrended.nc"
if not exists(fpath_merra2_detrended):
    merra2_detrended = calculate_reanalysis_detrended(merra2, merra2G,␣
 ↪hadcrut_ref_period)
    merra2_detrended.encoding.update(encoding_options)
    #merra2_detrended.attrs["source"] = "MERRA-2"
    merra2_detrended.to_netcdf(fpath_merra2_detrended)
merra2_detrended = xr.open_dataarray(fpath_merra2_detrended)
```

3.2 Standard Δ-shift

[9]:
```
threshold_delta_relative = pd.Series(deltas_relative, index=pd.Index(np.
 ↪arange(len(deltas)), name="threshold")).to_xarray().assign_coords(
    threshold_delta=("threshold", deltas),
    threshold_delta_relative=("threshold", deltas_relative))
```

[10]:
```
# load from file if available
# WARNING: If you require a new copy of merra2_detrended
#          (eg. because you changed some of the inputs above)
#          then please delete/rename "merra2_detrended_delta.nc"
```

```
fpath_merra2_detrended_delta = "outputs/temperature/merra2_detrended_delta.nc"
if not exists(fpath_merra2_detrended_delta):
    merra2_detrended_delta = merra2_detrended + threshold_delta_relative
    merra2_detrended_delta.encoding.update(encoding_options)
    merra2_detrended_delta.name = "temperature"
    merra2_detrended_delta.attrs = merra2_detrended.attrs
    merra2_detrended_delta.attrs["source"] = "MERRA-2, HadCRUT5"
    merra2_detrended_delta.attrs["title"] = "MERRA-2 Detrended, with␣
 ↪Delta-Shift"
    merra2_detrended_delta.attrs["production time"] = datetime.now().
 ↪strftime("%Y-%m-%d %H:%M:%S")
    merra2_detrended_delta.to_netcdf(fpath_merra2_detrended_delta)
merra2_detrended_delta = xr.open_dataarray(fpath_merra2_detrended_delta)
```

3.3 PPE Δ-shift

In addition to using a 'standard' delta-shift of temperature, we can look at the impacts of regional variability in delta-shift. The GB warming is calculated in the UKCP18 RCM, with respect to the time-period where the UKCP18 GCM matches the desired threhsold (1.5, 2.0, etc). Therefore, this 'ppe delta-shift' employs the delta-shift method to represent different rate of warming that could be experienced locally with respect to global average temperature change

[11]:
```
ukcp18_ppe_rcm_deltas = pd.read_csv("temperature/ukcp18_ppe_rcm_deltas.csv",␣
 ↪index_col=0).set_index(pd.Index(np.arange(len(deltas)), name="threshold")).
 ↪to_xarray().to_array("ppe")
```

[12]:
```
# load from file if available
# WARNING: If you require a new copy of merra2_delta
#          (eg. because you changed some of the inputs above)
#          then please delete/rename "merra2_detrended_ppe_delta.nc"
fpath_merra2_ppe_delta = "outputs/temperature/merra2_detrended_ppe_delta.nc"
if not exists(fpath_merra2_ppe_delta):
    # calculate relative delta-shift to apply, constructing variability of each␣
 ↪UKCP18 ppe
    merra2_ppe_delta = merra2_detrended + ukcp18_ppe_rcm_deltas
    merra2_ppe_delta = merra2_ppe_delta.assign_coords(
        threshold_delta=("threshold", deltas),
        threshold_delta_relative=("threshold", deltas_relative))
    merra2_ppe_delta.encoding.update(encoding_options)
    merra2_ppe_delta.name = "temperature"
    merra2_ppe_delta.attrs = merra2_detrended.attrs
    merra2_ppe_delta.attrs["source"] = "MERRA-2, UKCP18, HadCRUT5"
    merra2_ppe_delta.attrs["title"] = "MERRA-2 Detrended, with PPE Delta-Shift"
    merra2_ppe_delta.attrs["production time"] = datetime.now().
 ↪strftime("%Y-%m-%d %H:%M:%S")
    merra2_ppe_delta.to_netcdf(fpath_merra2_ppe_delta)
merra2_ppe_delta = xr.open_dataarray(fpath_merra2_ppe_delta)
```

4 Infrastructure electricity demand model

Calculate the infrastructure electricity demand separately for each version of the temperature time-series ('standard' delta-shift and 'ppe' delta-shift).

[13]:
```
realisations = 210
region = "GB"
```

[14]:
```
# demand timeseries from merra2_detrended
ds = gen_stoch_series(merra2_detrended_delta.expand_dims("ppe"),␣
 ↪realisations_size=realisations, region=region, rng=np.random.
 ↪default_rng(971218), title=f"merra2_detrended_delta derived infrastructure␣
 ↪electricity demand")
ds.encoding.update(encoding_options)
ds.sel(ppe=0).drop("ppe").to_netcdf("outputs/demand/merra2_detrended_delta.nc")
```

[15]:
```
# demand timeseries from merra2_ppe_delta
# load from file if available
# WARNING: If you require a new copy of a given demand timeseries
#          (eg. because you changed some of the inputs above)
#          then please rename or delete the existing .nc file
fpath = "outputs/demand/merra2_detrended_ppe_delta_{col}.nc"
# random number generator (used in stochastic modelling)
rng_list = [580297, 139715, 427556, 802644, 741087, 546205, 244996, 986182,␣
 ↪445011, 456272, 228297, 431855]
for col, seed in enumerate(rng_list):
    # get random number generator from unique seed
    rng = np.random.default_rng(seed)
    # get ppe member name
    ppe = str(ukcp18_ppe_rcm_deltas.ppe[col].values).zfill(2)
    # set output filename
    fname = fpath.format(col=ppe)
    # if file not already present, generate timeseries and save to fname
    if not exists(fname):
            print(f"Generating stochastic time series␣
 ↪merra2_ppe_delta_{str(ppe)}.nc")
            ds = gen_stoch_series(merra2_ppe_delta.isel(ppe=[col], drop=False),␣
 ↪realisations_size=realisations, region=region, rng=rng,␣
 ↪title=f"merra2_detrended_ppe_delta derived infrastructure electricity␣
 ↪demand")
            ds = ds.rename({"threshold_delta_relative":␣
 ↪"threshold_delta_relative_global"}).
 ↪assign_coords(threshold_delta_relative_regional=("threshold",␣
 ↪ukcp18_ppe_rcm_deltas.isel(ppe=col).values))
            ds.encoding.update(encoding_options)
            ds.to_netcdf(fname)
```