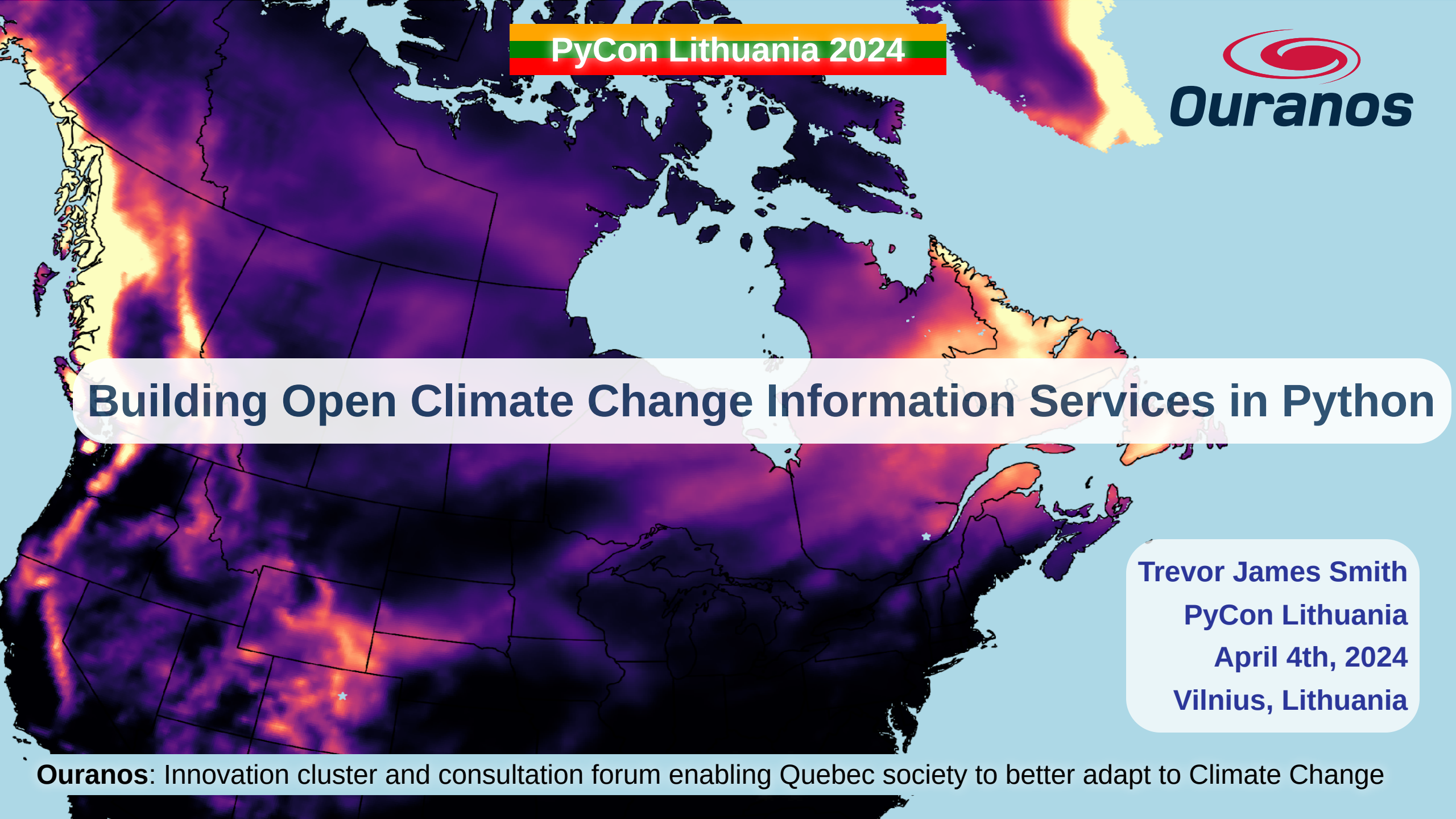


Building Open Climate Change Information Services in Python

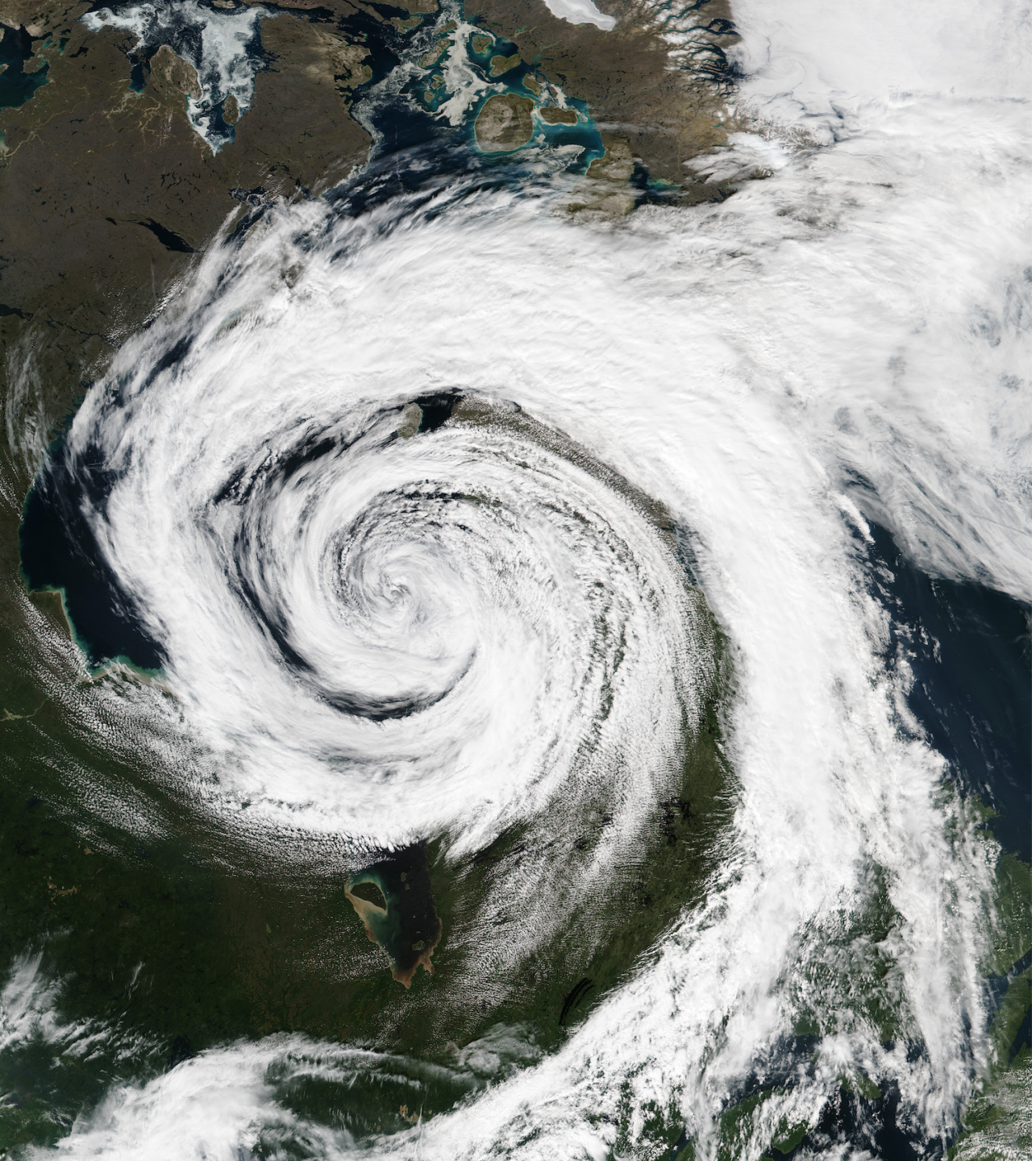
Trevor James Smith

PyCon Lithuania

April 4th, 2024

Vilnius, Lithuania

Ouranos: Innovation cluster and consultation forum enabling Quebec society to better adapt to Climate Change



Presentation Outline

- Who am I? / What is Ouranos?
- What's our context?
- Climate Services?
- `xclim`: climate operations
- `finch`: `xclim` as a Service
- Climate WPS Frontends
- What's next for us?
- Acknowledgements

Photo: Extratropical Cyclone over Hudson Bay, Canada, August 2016. Credit: NASA Earth Observatory.

Who am I?

Trevor James Smith

github.com/Zeitsperre

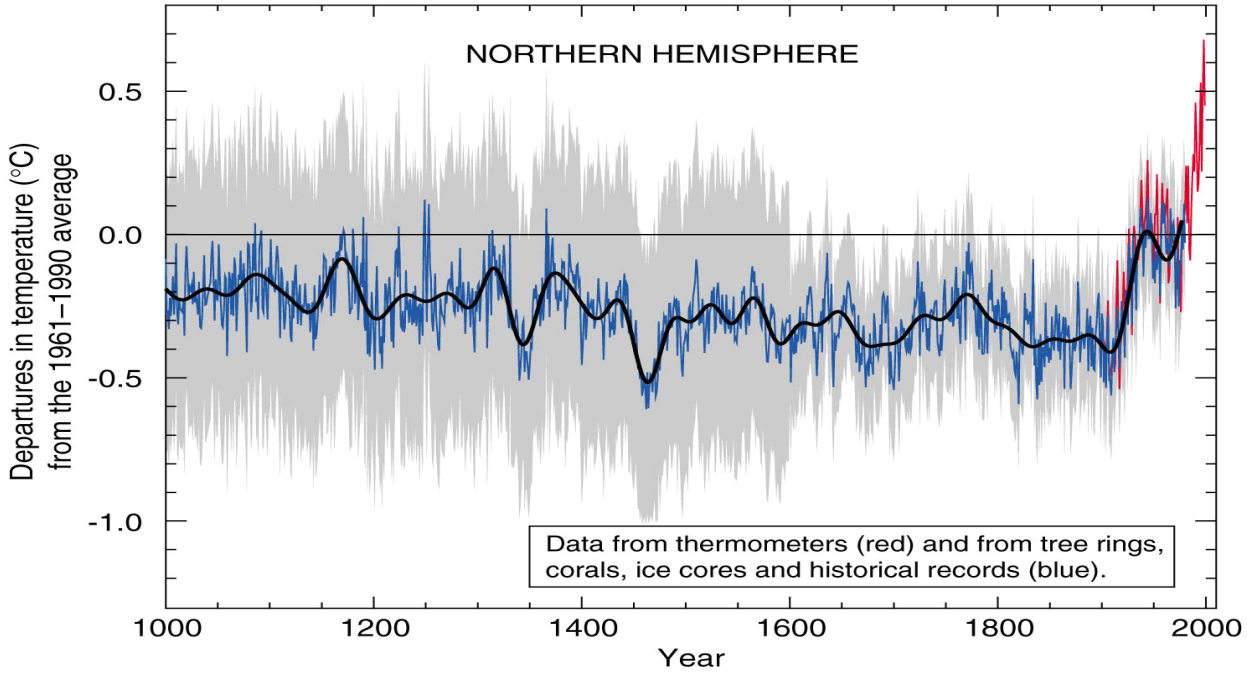
Zeit@techhub.social

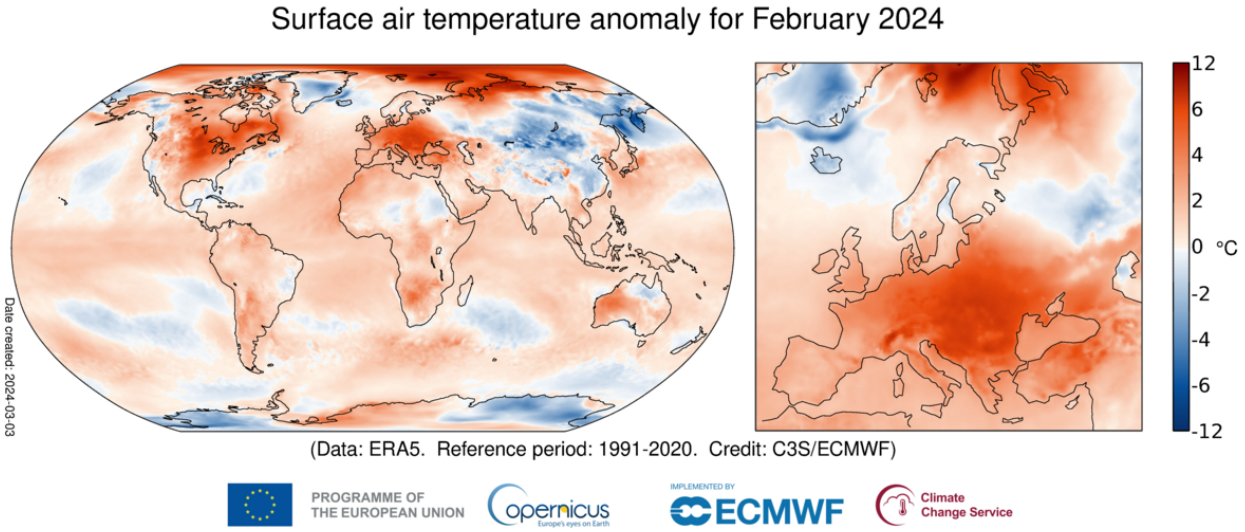
- Research software developer/maintainer from Montréal, Canada
- Studied climate change impacts on wine viticulture in Southern Québec
- Making stuff with Python for ~6 years
- 僕は日本語を勉強しています!

What is Ouranos?

- Not-for-profit climate research consortium established in 2003 in Montréal, Québec, Canada
    - Created in response to the January 1998 North American Ice Storm
- Climate change adaptation, climate modelling, and **climate information services**
- Regional Climate Projection Data Provider

Photo credit: https://www.communitystories.ca/v2/grand-verglas-saint-jean-sur-richelieu_ice-storm/





What's the climate situation?

- Climate Change is having major impacts on Earth's environmental systems
- IPCC: **Global average temperature has increased > 1.1 °C** over pre-industrial normals

"Since systematic scientific assessments began in the 1970s, the influence of human activities on the warming of the climate system has evolved from theory to established fact"

- IPCC Sixth Assessment Report Technical Summary

What's the climate data situation?

- Climate data is growing exponentially in size
  - Climate models being developed every year
  - Simulations being produced every day
  - Higher resolution input **and** output datasets
  - Specialized analyses and user needs

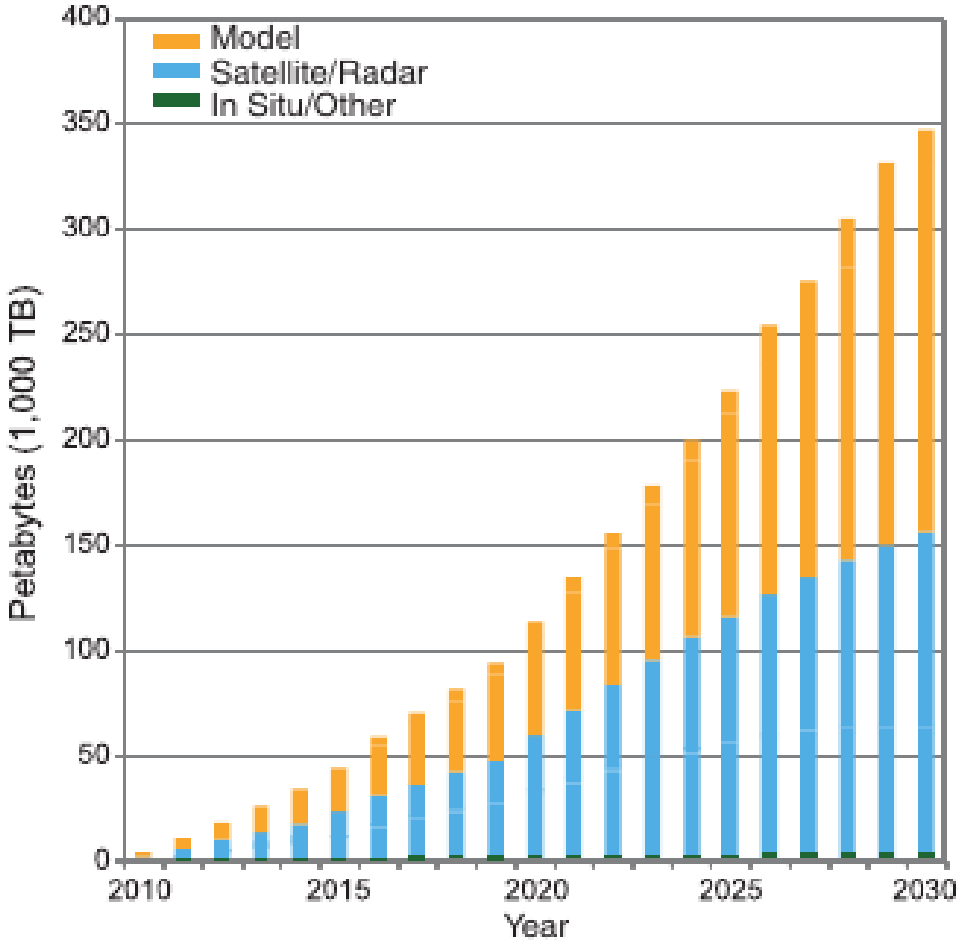


Fig. 2. The volume of worldwide climate data is expanding rapidly, creating challenges for both physical archiving and sharing, as well as for ease of access and finding what's needed, particularly if you are not a climate scientist. The figure shows the projected increase in global climate data holdings for climate models, remotely sensed data, and in situ instrumental/proxy data.

Overpeck, Jonathan T., Gerald A. Meehl, Sandrine Bony, and David R. Easterling. "Climate Data Challenges in the 21st Century." Science 331, no. 6018 (February 11, 2011): 700–702. https://doi.org/10.1126/science.1197869

Climate Services

What do they provide?

- Tailoring objectives and information to different user needs
- Providing access to **climate information**
- Building local mitigation/adaptation capacity
- Offering training and support
- Making sense of **Big *climate* Data**

Infographic from The Canadian Centre for Climate Services (CCCS)

What information do Climate Services provide?

Climate indicators, e.g.:

- **Hot Days** (Days with temperature >= 22 deg Celsius);
- **Beginning / End / Length of the growing season**;
- **Average seasonal rainfall** (3-Month moving average precipitation);
- **Daily temperature range**;
  - etc.

Planning tools

- Maps
- Point estimates at geographic locations
- Time series estimates
- Gridded values
- Raw data (for experts)
- **Not really sure what they want/need?**
  Guidance from experts!

Why build a Climate Services library in Python?

- Robust and fast scientific Python libraries
- Growing demand for climate services/products
  - Provide access to the community so they can help themselves
- *The timing was right*
  - Internal and external demand for common tools
- Less time writing code, more time spent doing research

What are the requirements?

What does it need to perform?

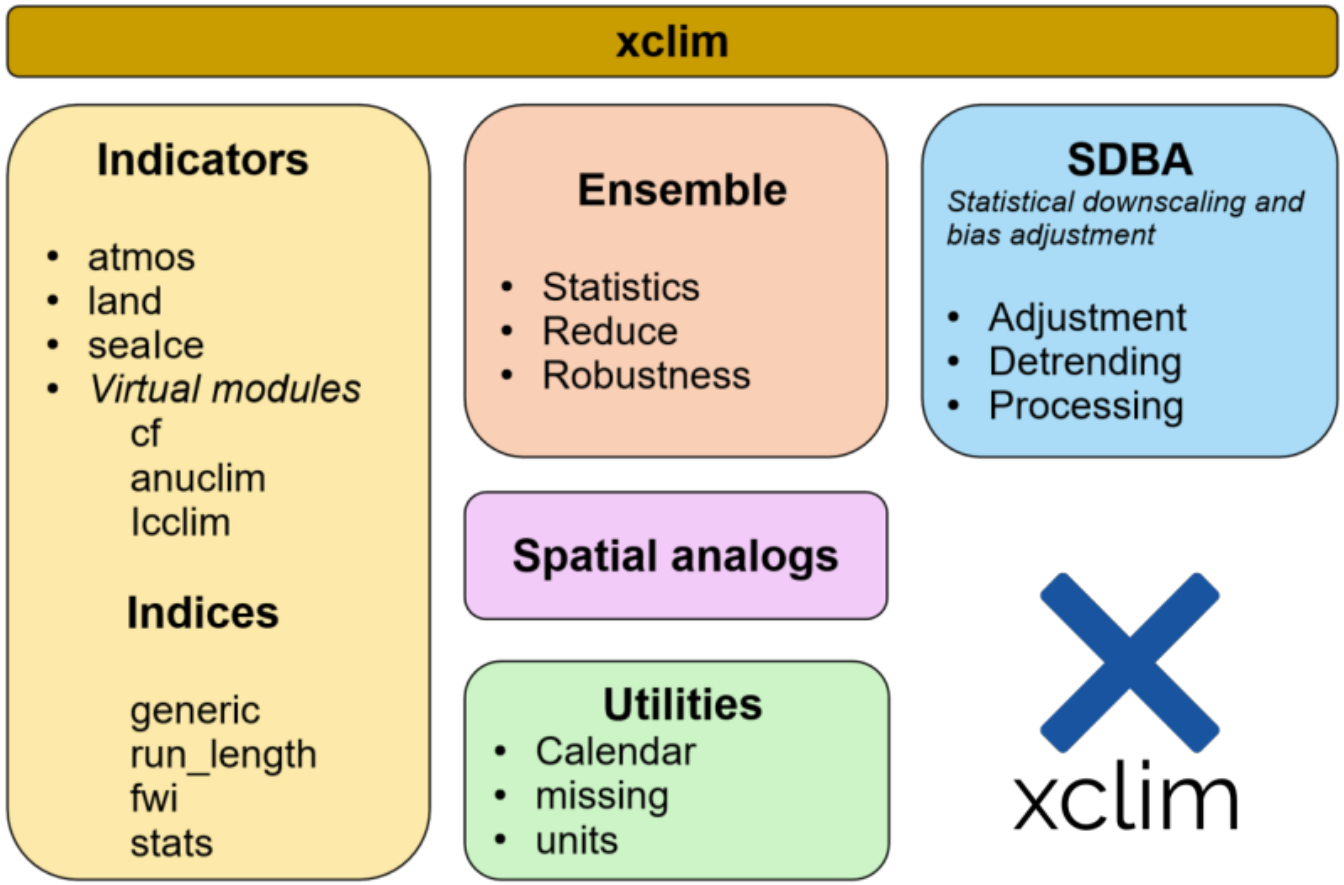
- **Climate Indicators**
  - Units management
  - Metadata management
- **Ensemble statistics**;
- **Bias Adjustment**;
- **Data Quality Assurance Checks**

Implementation goals?

- **Operational** : Capable of handling very large ensembles of climate data
- **Foolproof** : Automatic verification of data and metadata validity by default
- **Extensible** : Flexibility of use and able to easily provide custom indicators, as needed

Xclim: Climate Services library

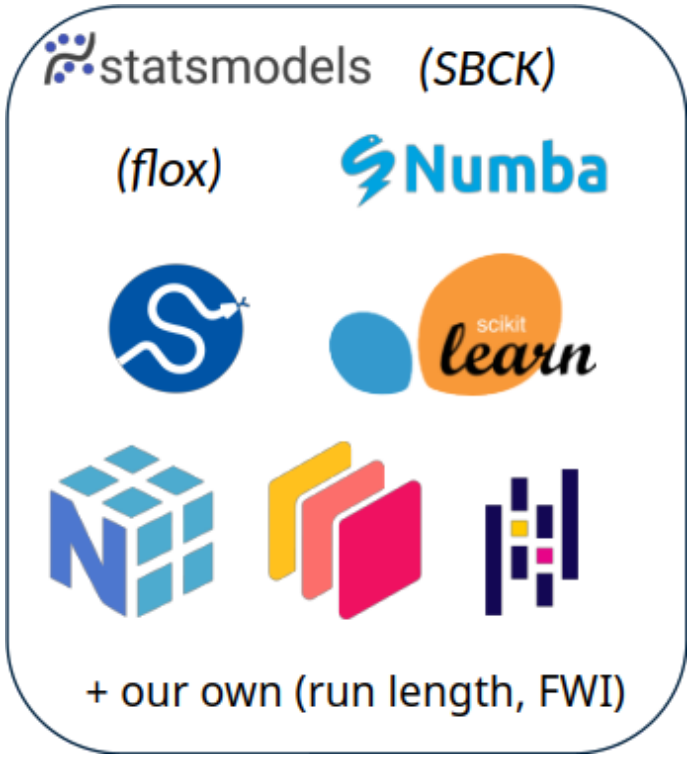
- **Asynchonous IO** and **fast**
- **Open Source** design
- **standards-compliant** metadata
- **Extensible** (modular)
- **Operational**



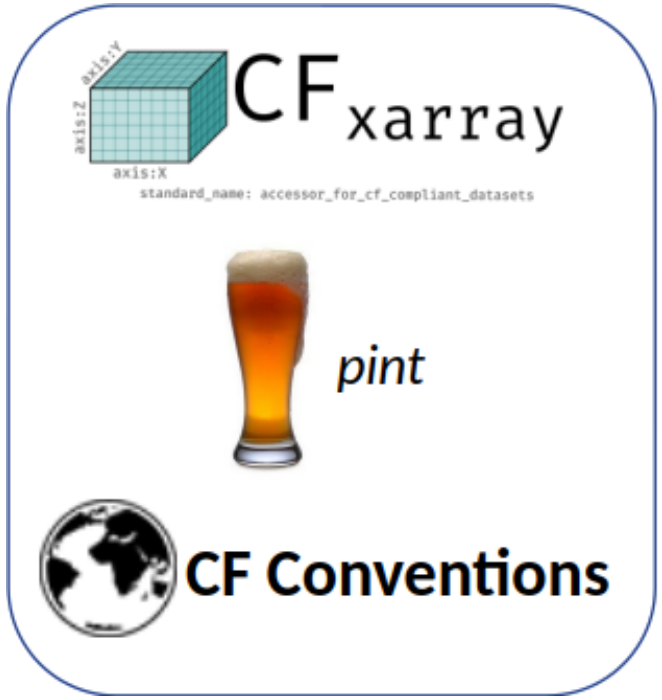
https://github.com/Ouranosinc/xclim

How did we build Xclim?

- **Data Structure**



- **Algorithms**



- **Data and Metdata Conventions**

Upstream contributions from Xclim

- Non-standard calendar (`cftime`) support in `xarray.groupby`
- Quantile methods in `xarray.groupby`
- Non-standard calendar conversion migrated from `xclim` to `xarray`
- Climate and Forecasting (CF) unit definitions inspired from `MetPy`
  - Inspiring work in `cf-xarray`
- Weighted variance, standard deviations and quantiles in `xarray` (for ensemble statistics)
- Faster **NaN**-aware quantiles in `numpy`
- Initial polyfit function in `xarray`
- Not to forget mentioning work done by the team in `xESMF`, `intake-esm`, `cf-xarray`, `xncml`, and others for `xclim`-related downstream tools and workflows

Xclim algoritm design

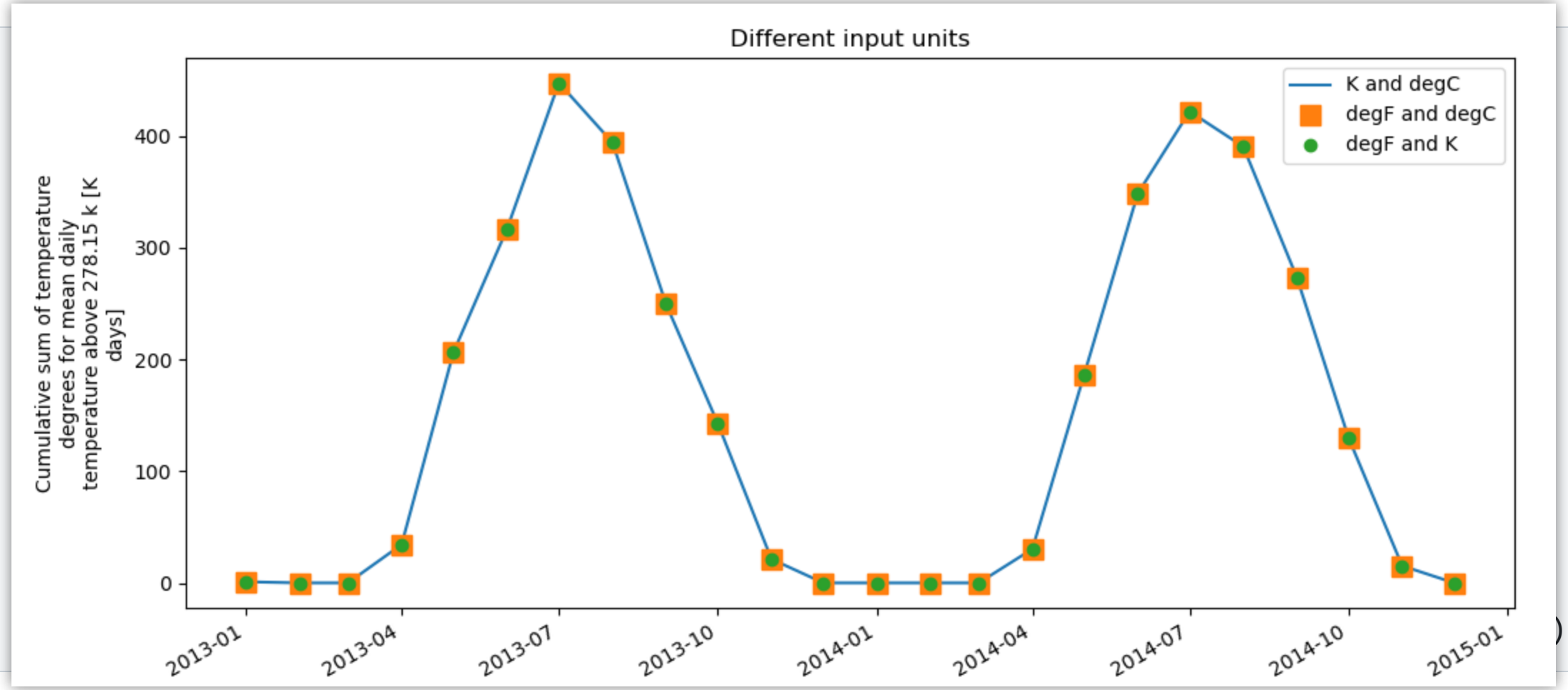
Two ways of calculating indicators

- `indice` (**Core algorithms**)
  - For users that don't care for the standards and quality checks
- `indicators` (**End-User API**)
  - Metadata standards checks
  - Data quality checks
  - Time frequency checks
  - Missing data-compliance
  - Calendar-compliance

What does Xclim do? 🤔 Units Management

```python
import xclim
from clisops.core import subset

# Data is in Kelvin, threshold is in Celsius, and other combinations

# Extract a single point location for the example
ds_pt = subset.subset_gridpoint(ds, lon=-73, lat=44)

# Calculate indicators with different units

# Kelvin and Celsius
out1 = xclim.atmos.growing_degree_days(tas=ds_pt.tas, thresh="5 degC", freq="MS")

# Fahrenheit and Celsius
out2 = xclim.atmos.growing_degree_days(tas=ds_pt.tas_F, thresh="5 degC", freq="MS")

# Fahrenheit and Kelvin
out3 = xclim.atmos.growing_degree_days(tas=ds_pt.tas_F, thresh="278.15 K", freq="MS")
```

What does Xclim do? □ Units Management



What does Xclim do? □ Metadata Locales

```python
import xarray as xr
import xclim


ds = xr.open_dataset("my_dataset.nc")

with xclim.set_options(
    # Drop timesteps with more than 5% of missing data
    set_missing="pct", missing_options=dict(pct={"tolerance": 0.05}),

    metadata_locales=["fr"] # Add French language metadata
):
    # Calculate Annual Frost Days (days with min temperature < 0 °C)
    FD = xclim.atmos.frost_days(ds.tas, freq="YS")
```

What does Xclim do? □ Metadata Locales

```
import
import


ds = xr

with xc
    # D
    set                                                    ),

    met
):
    # C                                                    )
    FD
```

xarray.DataArray 'fd' (**time**: 2, **lat**: 25, **lon**: 53)

302.0 306.0 308.0 314.0 317.0 322.0 323.0 ... 0.0 0.0 0.0 0.0 0.0 0.0

▾ Coordinates:

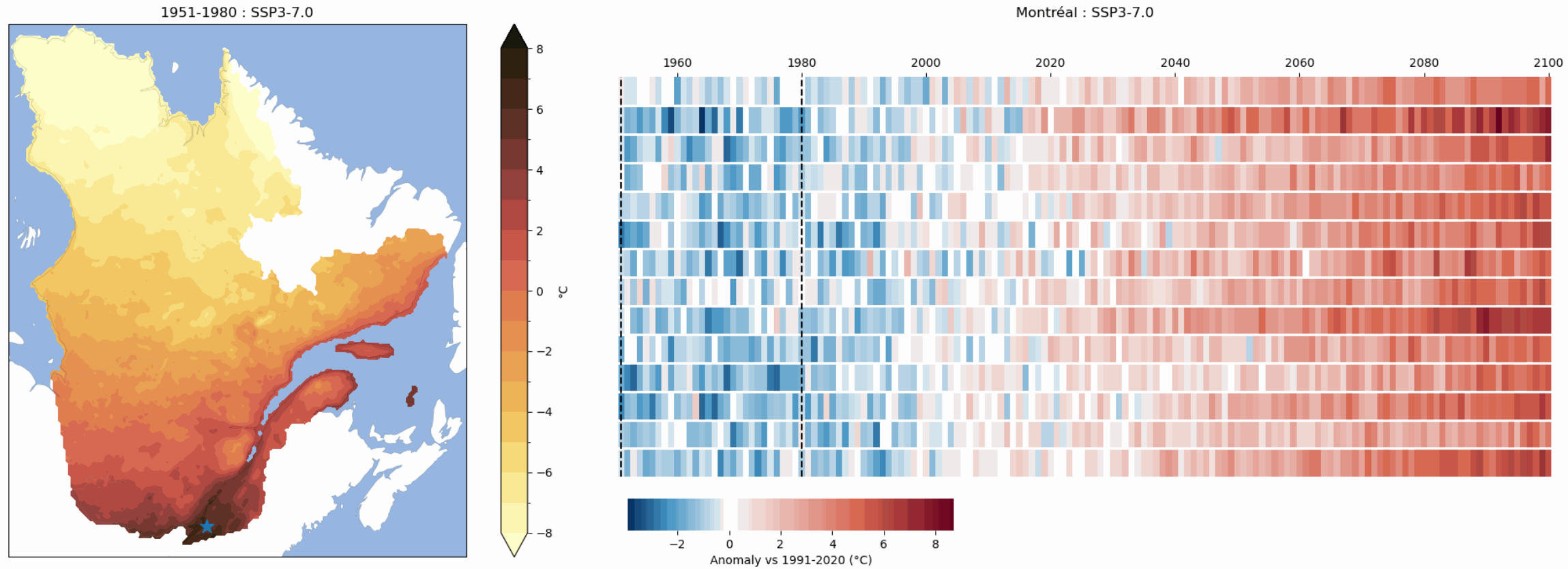
lat	(lat)		float32	75.0 72.5 70.0 ... 20.0 17.5 15.0
lon	(lon)		float32	200.0 202.5 205.0 ... 327.5 330.0
time	(time)		datetime64[ns]	2013-01-01 2014-01-01

▸ Indexes: (3)

▾ Attributes:

units :	days
cell_methods :	time: sum over days
history :	[2022-12-23 11:14:55] frost_days: FROST_DAYS(tasmin=air, thresh='0 degC', freq='YS') with options check_missing=pct, missing_options={'tolerance': 0.05} - xclim version: 0.39.0
long_name :	Number of frost days (tmin < 0°c)
description :	Annual number of days where the daily minimum temperature is below 0 degc.
long_name_fr :	Nombre de jours où la température minimale quotidienne est sous 0 degC
description_fr :	Nombre annuel de jours où la température minimale quotidienne est sous 0 degC.

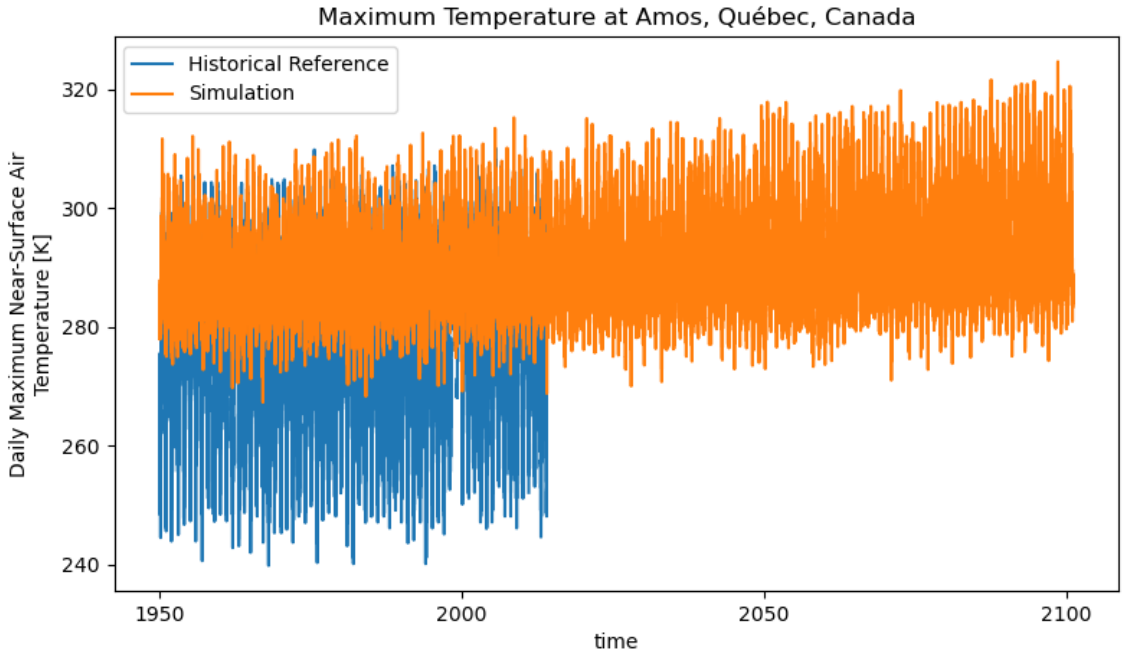
What does Xclim do ? Climate Ensemble Mean Analysis

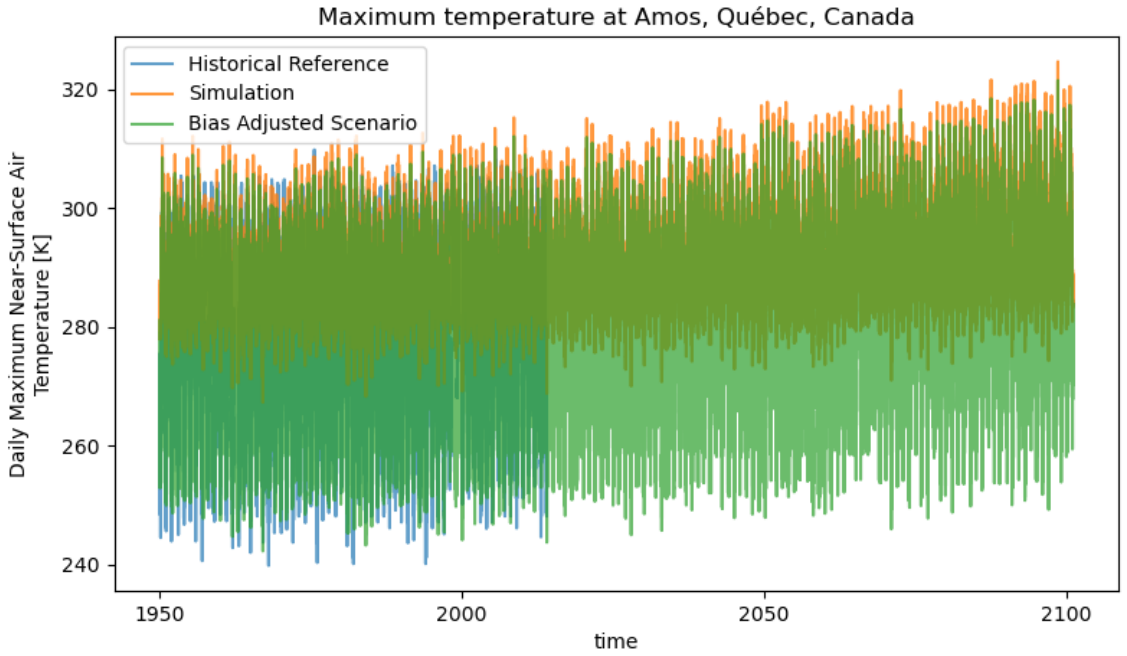


Average temperature from the years 1991-2020 baseline across 14 IPCC climate models at Montréal, Québec (*extreme warming scenario: SSP3-7.0*)

What Does Xclim do? □ Bias Adjustment

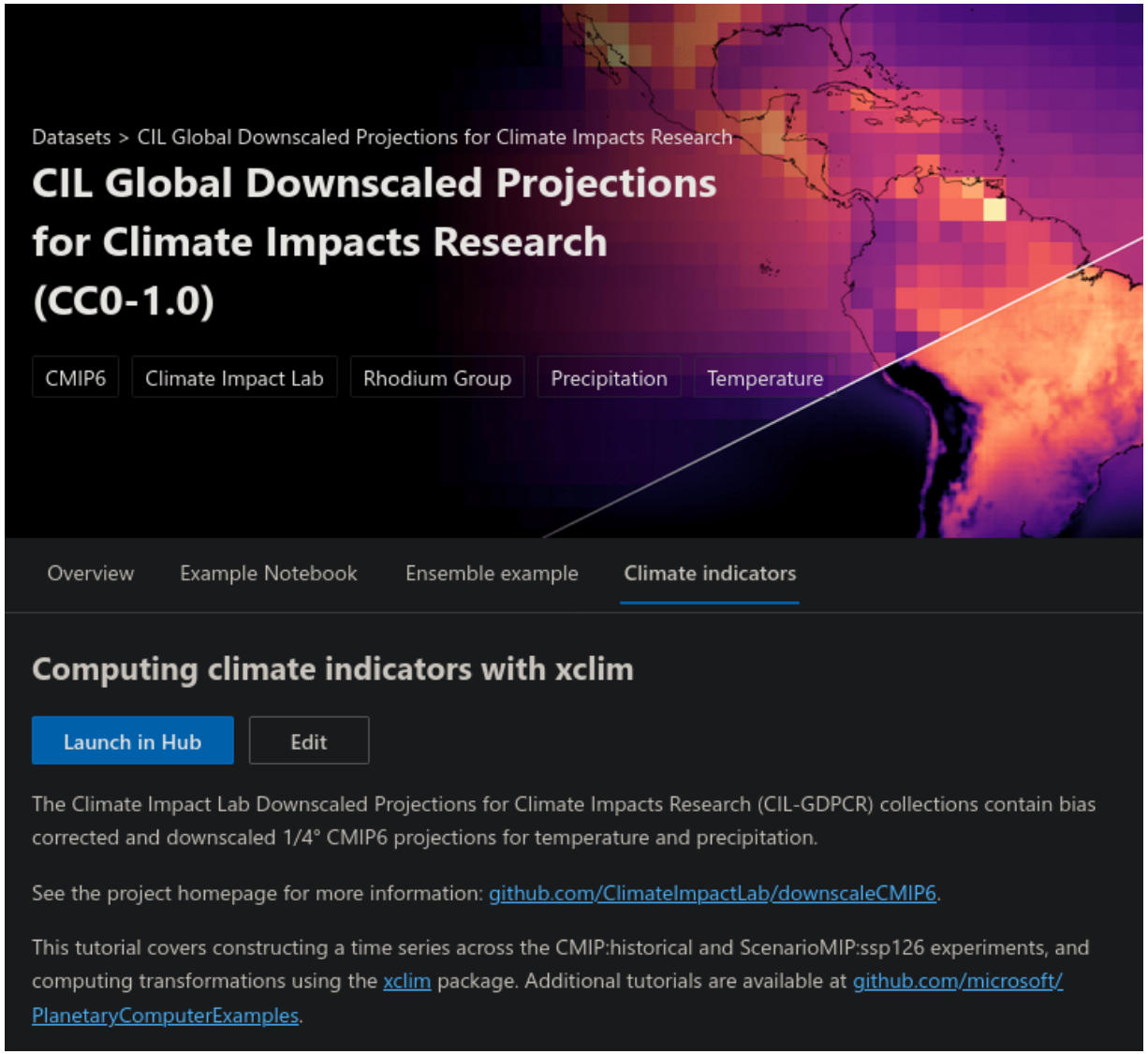
- Adjusts model bias from projected data using a `train` / `adjust` approach
- Several implementations available :
  - Quantile Mapping
  - Principal Components Analysis
  - Multivariate (MBCn)
- Plugin support for Python package **SBCK** (dOTC, CDFt, and other algorithms)





That's great and all, but what if...

- There's just too much data that we need to crunch:
  - The data could be spread across servers globally
  - Local computing power is just not enough for the analysis
- We need to run lots of specific workflows regularly
- The user doesn't know how to write a Python script:
  - A biologist who uses `R` for their work
  - A city planner who just needs a range of estimates for future rainfall
  - Agronomist wondering about average growing conditions in 10 years



Xclim on Compute Platforms

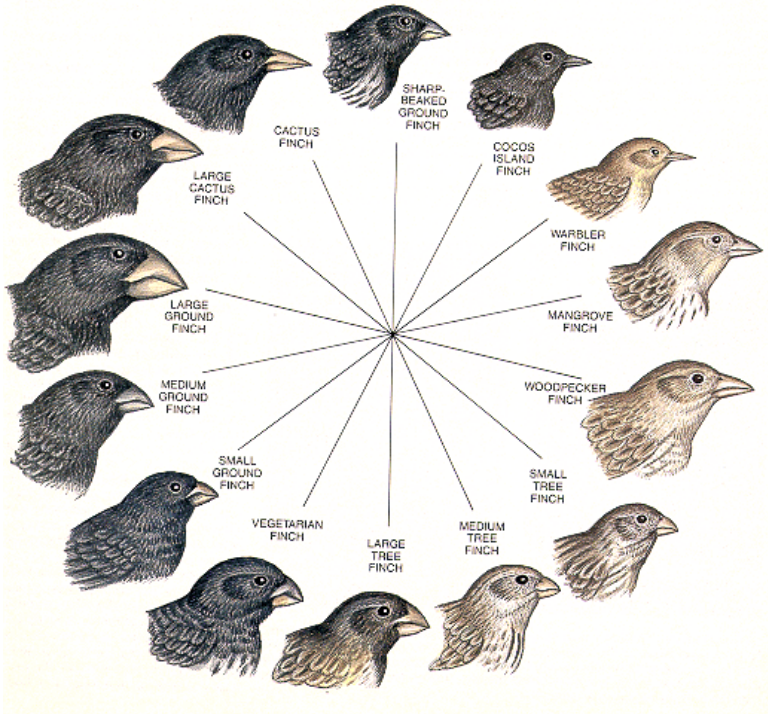
Microsoft Planetary Computer

- Computing Climate Indicators with xclim

Finch: Xclim as a Web Service

github.com/Bird-house/Finch

- **Web Processsing Services** (WPS)
  - Built with Python (**PyWPS**)
- Remote scientific analysis platforms
- *Bird-house likes to name their projects after birds*



Using the Finch Web Service from Python (`owslib`)

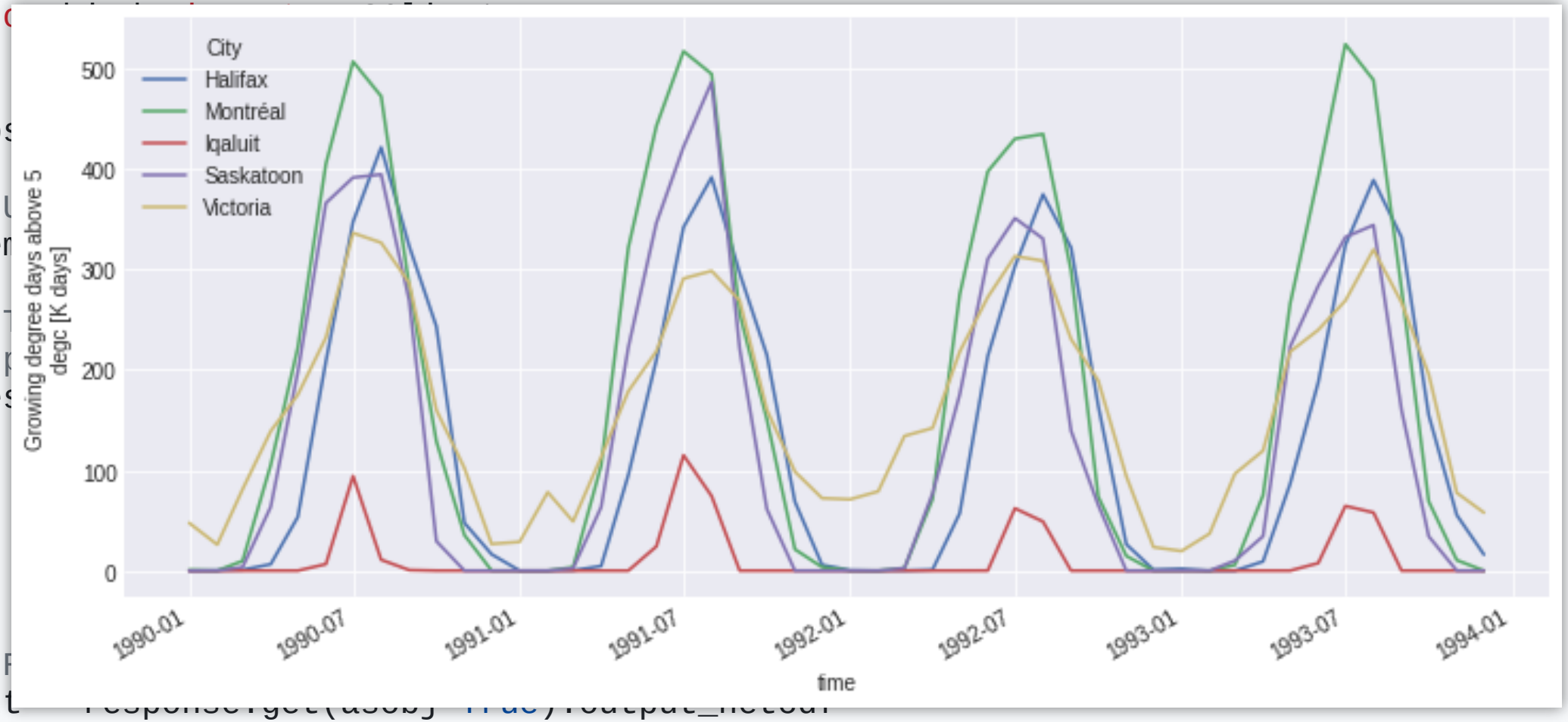
```python
from owslib.wps import WebProcessingService

# URL running our service
finch_url = "https://pavics.ouranos.ca/twitcher/ows/proxy/finch/wps"

# Connect to the Finch WPS service
finch = WebProcessingService(pavics_url)

# Get a listing of all processes
finch.processes

print(len(finch.processes))  # --> 430 supported indicators and analyses!
```

Using the Finch Web Service from Python (`birdy`)

```
from birdy import WPSClient


wps = WPSClient(finch_url)

# Using the OPeNDAP protocol
remote_dataset = "www.exampledata.lt/climate.ncml"

# The indicator call looks a lot like the one from `xclim` but
# passing a url instead of an `xarray` object.
response = wps.growing_degree_days(
    remote_dataset,
    thresh='10 degC',
    freq='MS',
    variable='tas'
)

# Returned as a streaming `xarray` data object
out = response.get(asobj=True).output_netcdf

out.growing_degree_days.plot(hue='location')
```

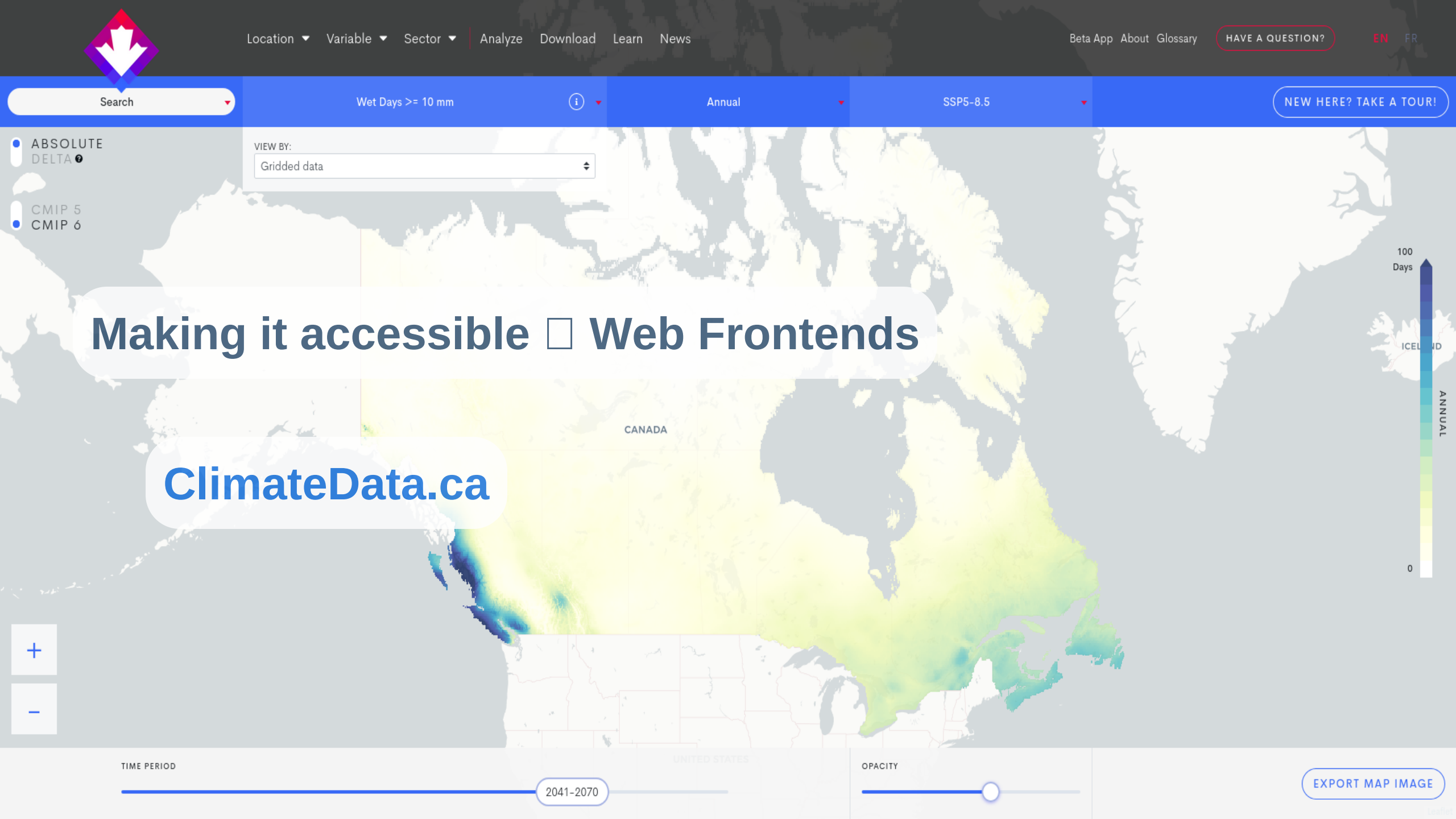
Bird-house/birdy -> PyWPS Helper Library

Using the Finch Web Service from Python (`birdy`)




```
fro

wps

# U
ren

# T
# p
res

)

# P
out    response.get(asobj=True).output_netcdf

out.growing_degree_days.plot(hue='location')
```

Bird-house/birdy -> PyWPS Helper Library


Location
Variable
Sector
Analyze
Download
Learn
News
Beta App
About
Glossary
HAVE A QUESTION?
EN
FR
Search
Wet Days >= 10 mm
Annual
SSP5-8.5
NEW HERE? TAKE A TOUR!
ABSOLUTE
DELTA
VIEW BY:
Gridded data
CMIP 5
CMIP 6
100
Days
0
ANNUAL
CANADA
Making it accessible Web Frontends
ClimateData.ca
TIME PERIOD
2041-2070
OPACITY
EXPORT MAP IMAGE

1 CHOOSE A DATASET

2 SELECT LOCATIONS

3 CUSTOMIZE VARIABLES

4 CHOOSE A TIMEFRAME

5 ADVANCED

Models

Full ensemble

Emissions Scenarios

SSP1-2.6 SSP2-4.5 SSP5-8.5

Percentiles (Unselect all to receive output from individual models)

5 10 25 50 75 90 95

Temporal Frequency

Annual Monthly

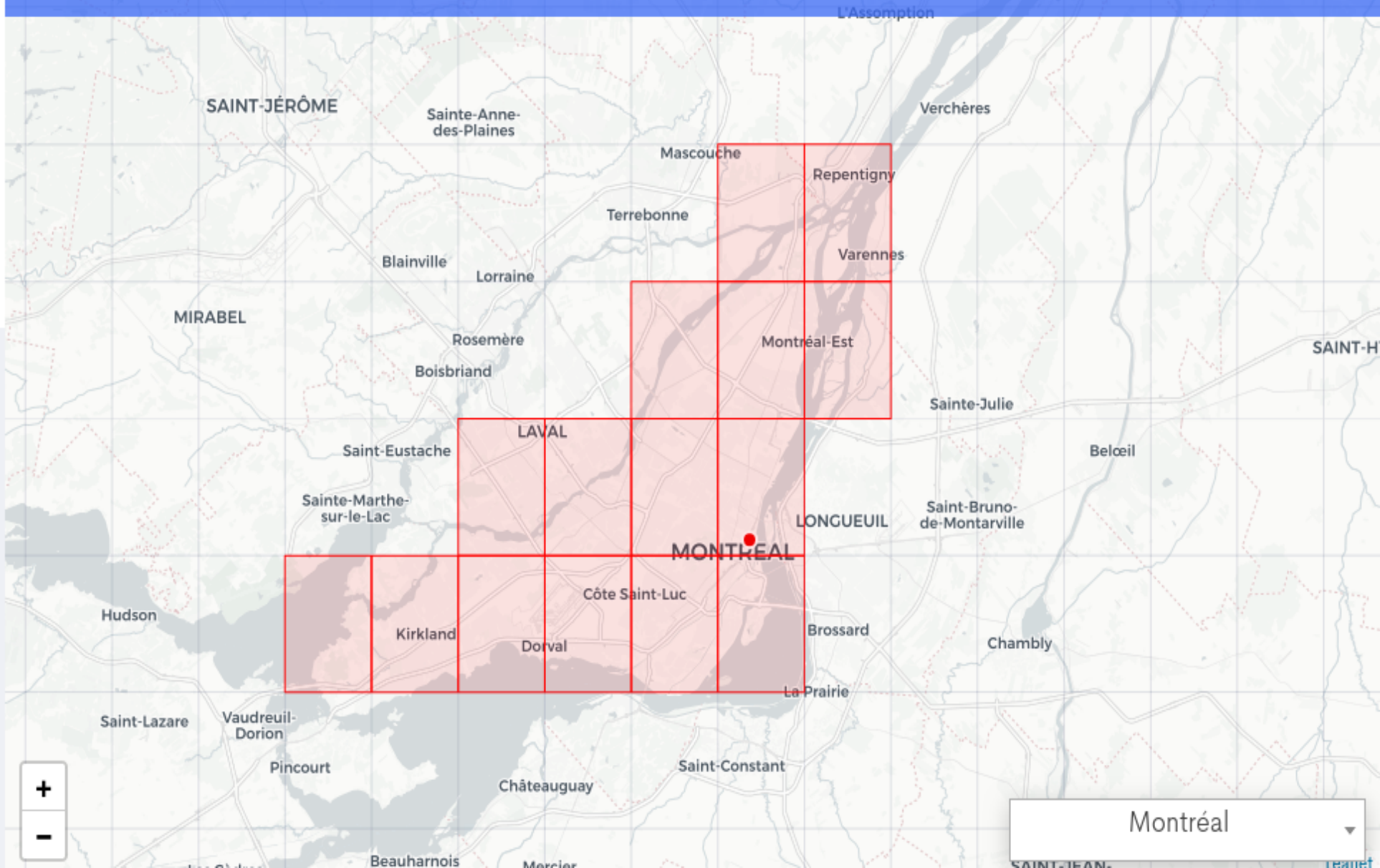
DATASET: CMIP6 (CanDCS-U6)

LOCATION: 15 Grids

VARIABLE(S): Degree Days Below a Threshold

TIMEFRAME: 2030 – 2060

OPTIONS: 1 Emissions Scenario | 3 Percentiles



Our Experience Adopting Python for Climate Science/Services

Before (circa 2016)

- `MATLAB`-based in-house libraries (**proprietary**)
  - No external libraries all in-house
- Issues with data storage/access/processing
  - Small team unable to meet demand
- Lack of uniformity between researchers
- Lots of bugs and human error
- Data analysis/requests served manually
- Software validation/testing???

After

- **Open Source** `Python` libraries (`numpy`, `sklearn`, `xarray`, etc.)
- Multithreading and streaming data formats (e.g. ZARR)
- Common tools built in-house and shared widely (`xclim`)
- Web service-based infrastructure
- Testing (`pytest`), Software CI/CD, and data validation
- Peer-Reviewed Software (**JOSS**)

Thanks!

Colleagues and collaborators

- Pascal Bourgault
- David Huard
- Trevor J. Smith
- Travis Logan
- Abel Aoun
- Juliette Lavoie
- Éric Dupuis
- Gabriel Rondeau-Genesse
- Carsten Ehbrecht
- Sarah Gammon
- Long Vu
- David Caron

and many more!

Ačiū!

Have a great rest of PyCon Lithuania!

Ouranosinc/xclim

JOSS 10.21105/joss.05415

DOI 10.5281/zenodo.10710942

Bird-house/finch

DOI 10.5281/zenodo.10870939