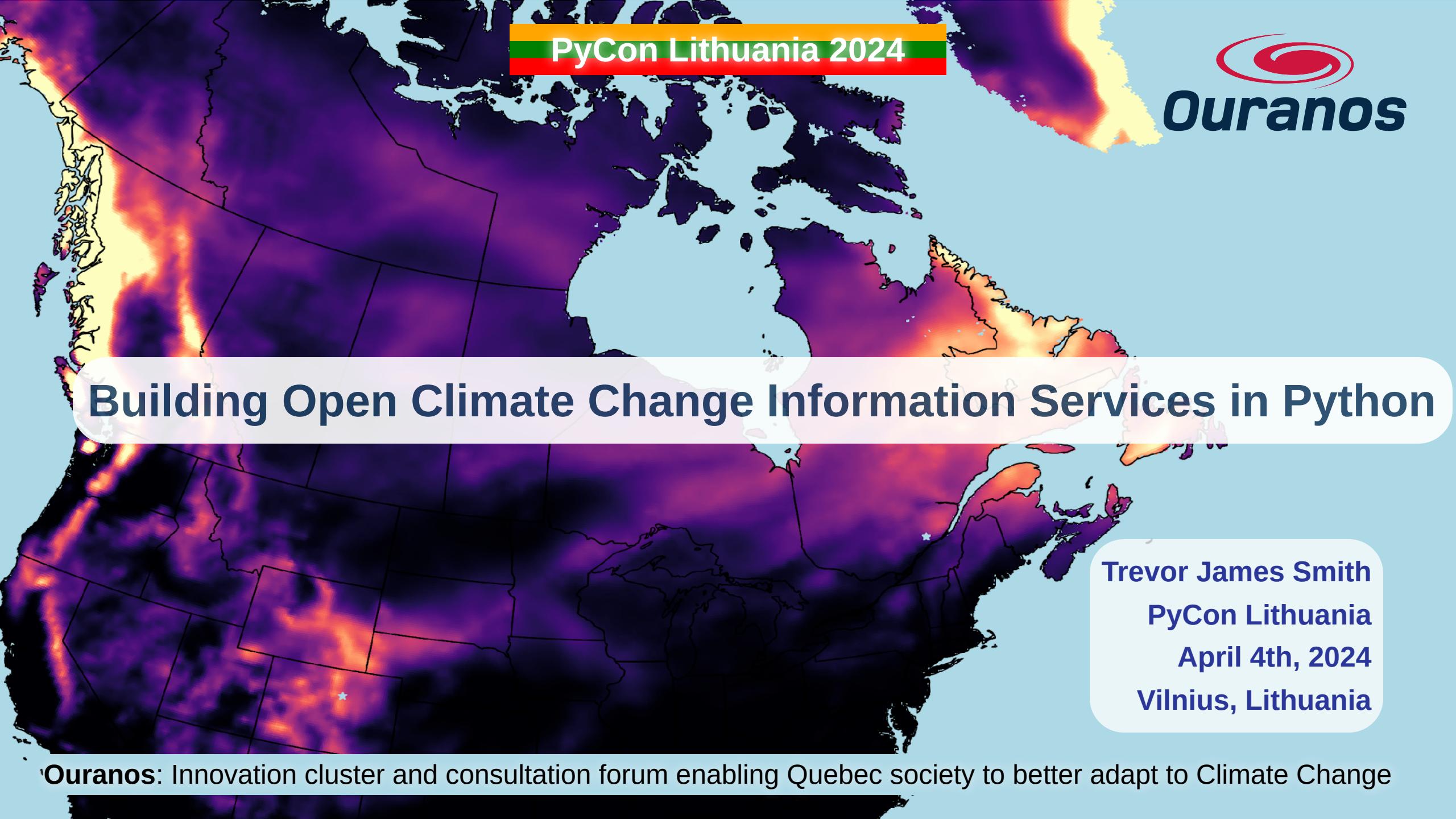


Building Open Climate Change Information Services in Python



Trevor James Smith
PyCon Lithuania
April 4th, 2024
Vilnius, Lithuania



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/packager/maintainer from Montréal, Québec, Canada 
- Studied climate change impacts on wine viticulture  in Southern Québec
- Making stuff with Python for ~6.5 years
- 僕は日本語を勉強しています! 

What is Ouranos?

- Non-profit research consortium established in 2003 in Montréal, Québec, Canada
 - Created in response to the January 1998 North American Ice Storm 
- Climate change adaptation and **climate information services**
- Climate Model Data Producer/Provider

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



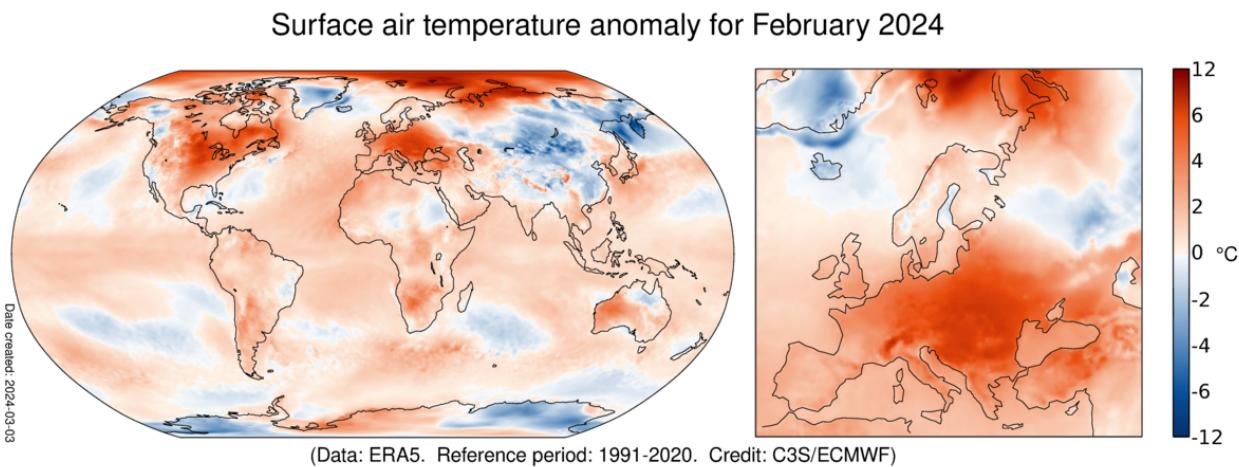
Regional Climatology and Adaptation to Climate Change

Ouranos is a collaborative innovation hub enabling Quebec society to better adapt to an evolving climate. The consortium brings together upwards of 450 researchers, experts, practitioners and decision-makers from an array of disciplines, collectively working on numerous applied research programs and projects.

SEARCH A PROJECT OR PUBLICATION

EXPLORE OUR PROGRAM





What's the climate situation?

- Climate Change is having major impacts on Earth's environmental systems
- IPCC: **Global average temperature has increased $> 1.1 \text{ }^{\circ}\text{C}$ since 1850s.**
 - $> 1.5 \text{ }^{\circ}\text{C}$ is considered to be beyond a safe limit

What's the climate data situation?

Climate data is growing exponentially in size and complexity

- New climate models being developed every year
- More climate simulations being produced every day
- Higher resolution input **and** output datasets (gridded data)
- Specialised analyses and more personalized user needs

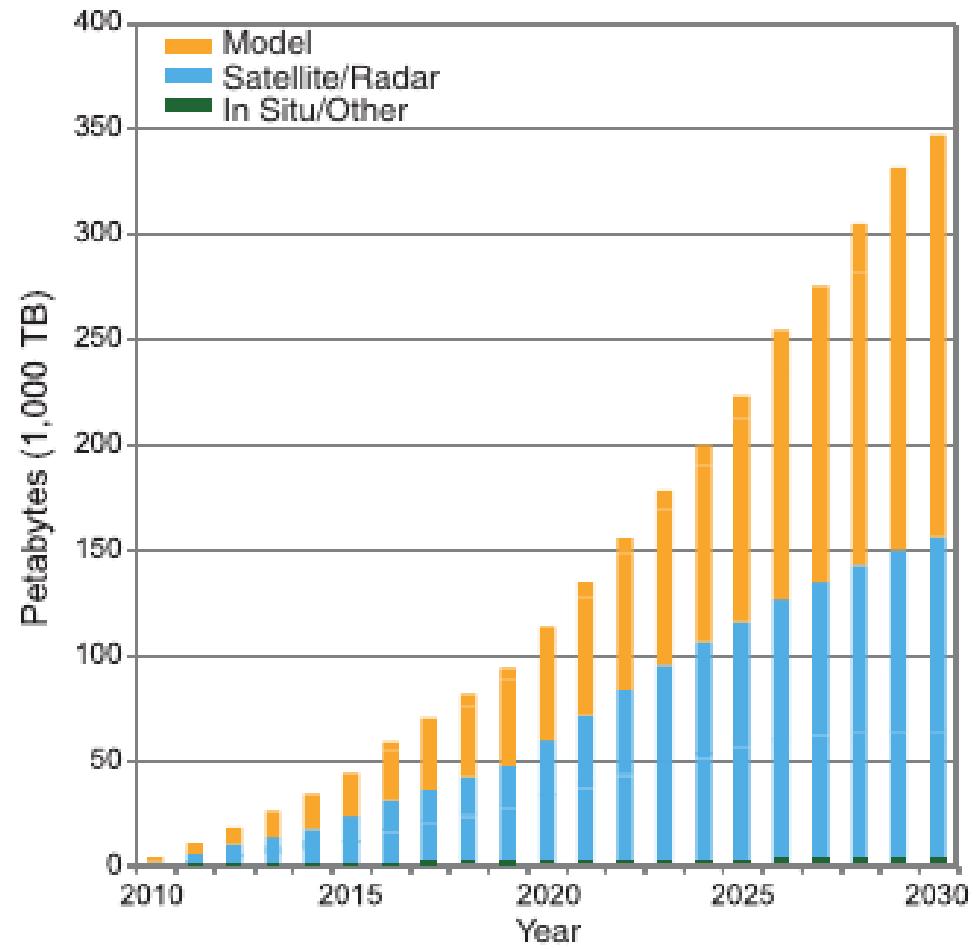


Fig. 2. The volume of worldwide climate data is expanding rapidly, creating challenges for both physical archiving and sharing, as well as for ease of access and finding what's needed, particularly if you are not a climate scientist. The figure shows the projected increase in global climate data holdings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

WHAT WE DO



DELIVER CLIMATE SERVICES
DRIVEN BY USER NEEDS



PROVIDE ACCESS TO
CLIMATE INFORMATION



BUILD LOCAL CAPACITY



OFFER TRAINING AND SUPPORT

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**

What information do Climate Services provide?

Climate Indicators, e.g.:

- **Hot Days** (Days with temperature \geq 22 deg Celsius);
- **Beginning / End / Length of the growing season;**
- **Average seasonal rainfall** (3-Month moving average precipitation);
- *Many more examples*

Planning Tools, e.g. :

- Maps 
- Point estimates at geographic locations 
- Gridded values 
- **Not really sure what they need? ?**
 - Guidance from experts!

Climate Services in the 2010s

- 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 output data uniformity between researchers !?
 - Lots of bugs 🐛 and human error 🤷
- Data analysis/requests served manually ⏳
- Software testing + data validation? Not really. 🙄

Building a Climate Services library?

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

Is there Python in this talk?

- Yes

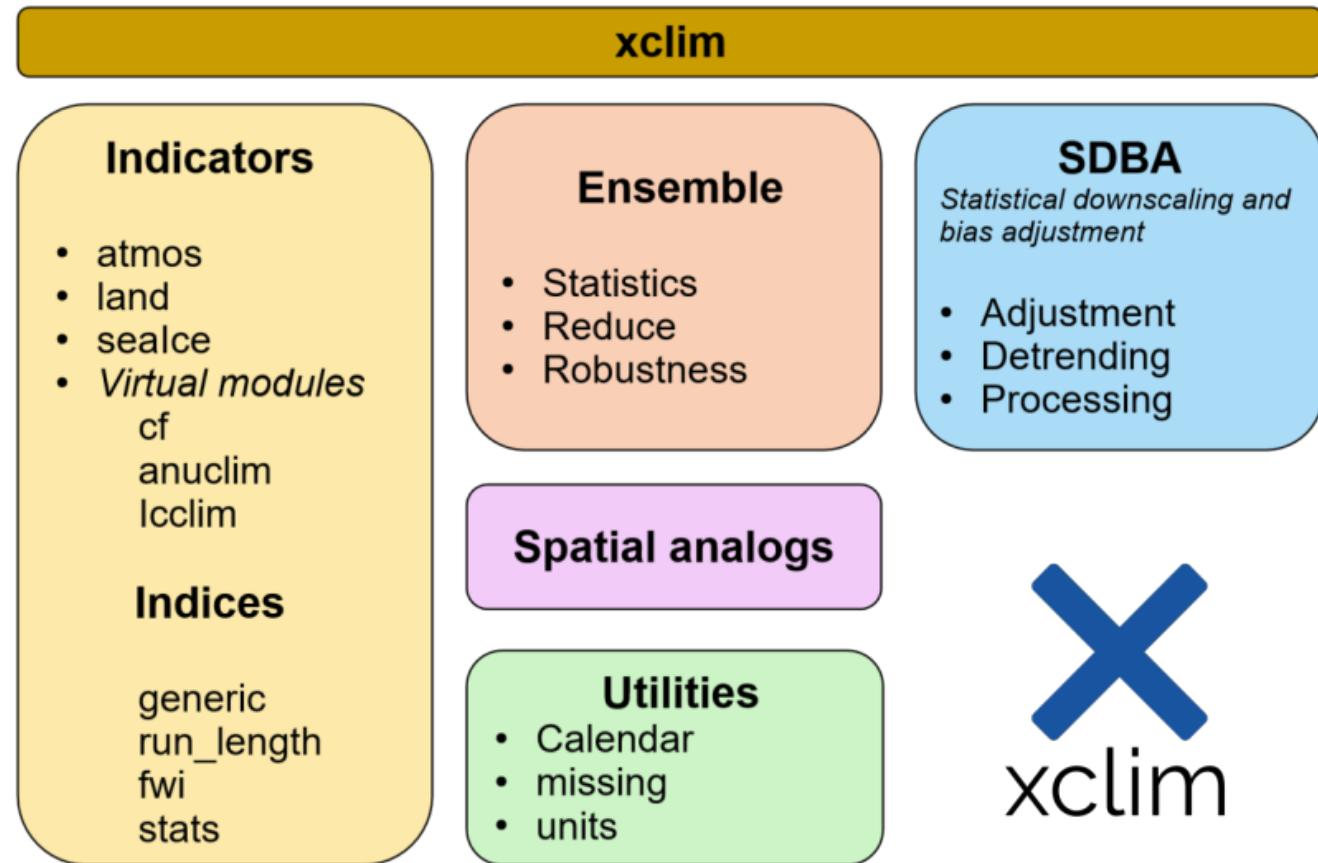
Why build a Climate Services library in Python?

- Robust, trusted, and fast scientific Python libraries
- Python's Readability / Reviewability
- 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**

Xclim : Climate Services for Python

 [Ouranosinc/xclim](https://github.com/Ouranosinc/xclim)

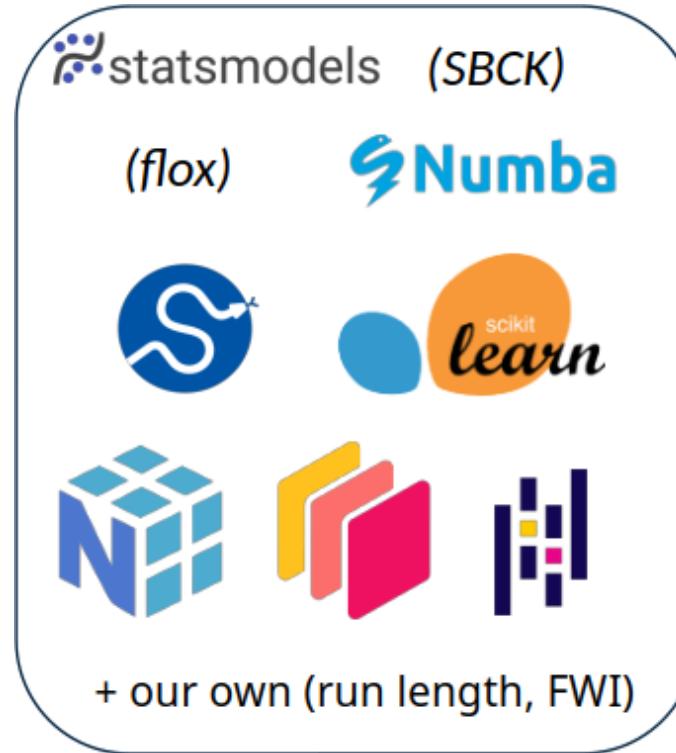
- **Asynchronous IO** and **fast**
- **Open Source** design
- **Standards-compliant** metadata
- **Extensible / modular**
- **Operational**



How did we build Xclim?



- Data Structure



- Algorithms



- Data and Metadata Conventions

Climate Indicator Example - Average Snow Depth

```
@declare_units(snd="[length]")
```

```
def snow_depth(
```

```
    snd: xarray.DataArray,
```

```
    freq: str = "YS",
```

```
) -> xarray.DataArray:
```

```
    """Mean of daily average snow depth.
```

Resample the original daily mean snow depth series by taking the mean over each period.

Parameters

snd : xarray.DataArray

Mean daily snow depth.

freq : str

Resampling frequency.

Returns

xarray.DataArray, [same units as snd]

The mean daily snow depth at the given time frequency

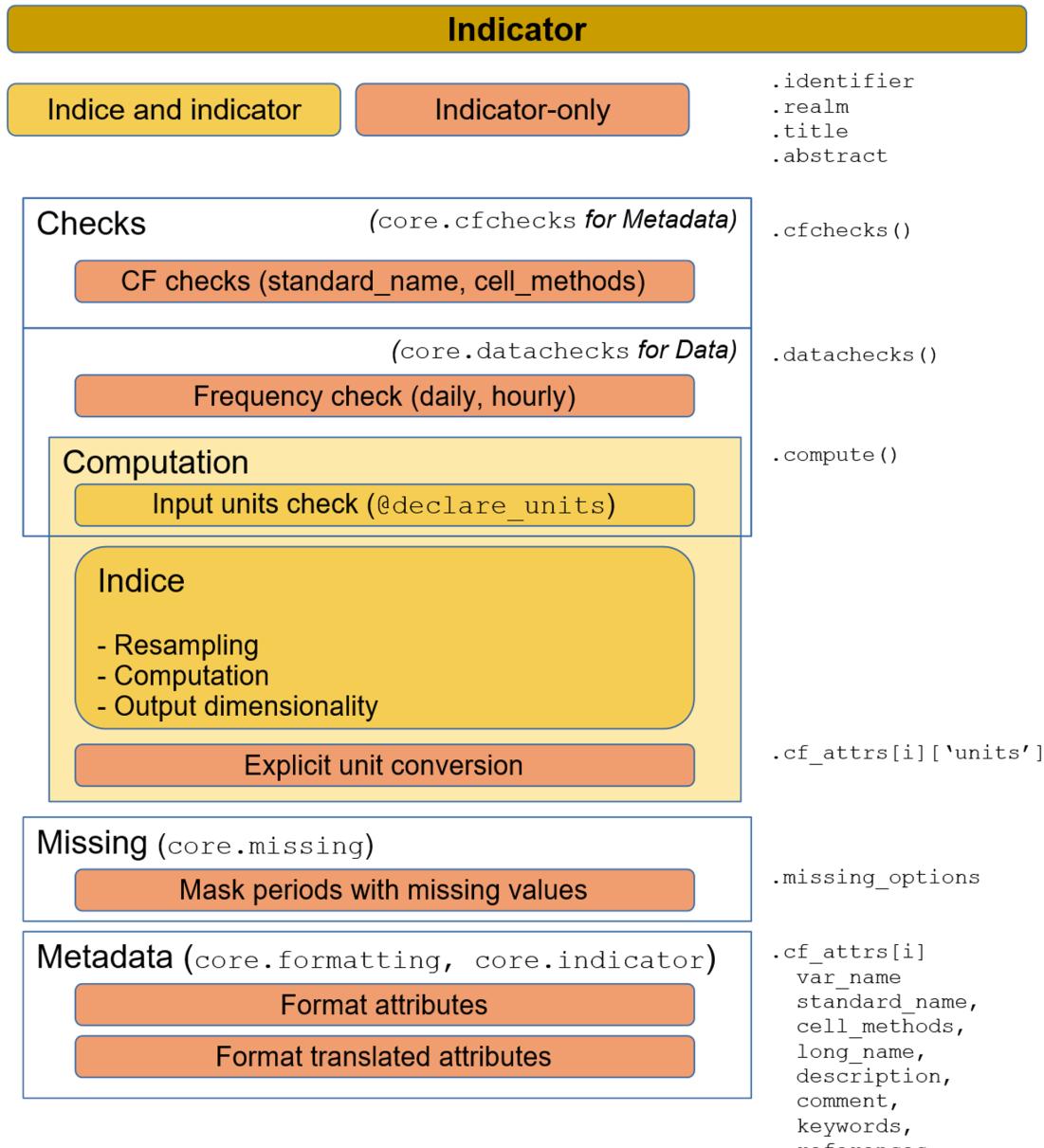
"""

```
return snd.resample(time=freq).mean(dim="time").assign_attrs(units=snd.units)
```

Xclim algorithm design

Two ways of calculating indicators

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



What does Xclim do? □ Units Management

```
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)

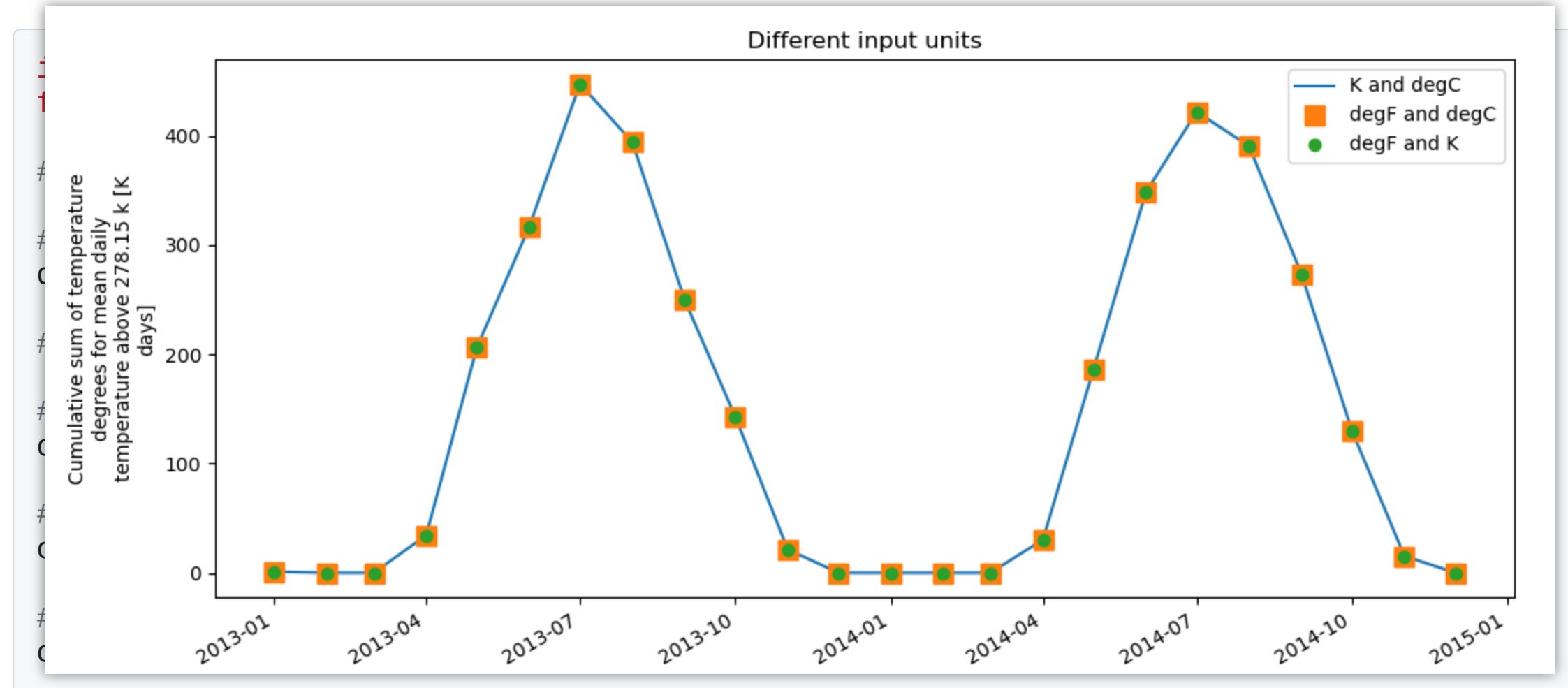
# Calculate indicators with different units

# 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? □ Missing Data and Metadata Locales

```
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")
```

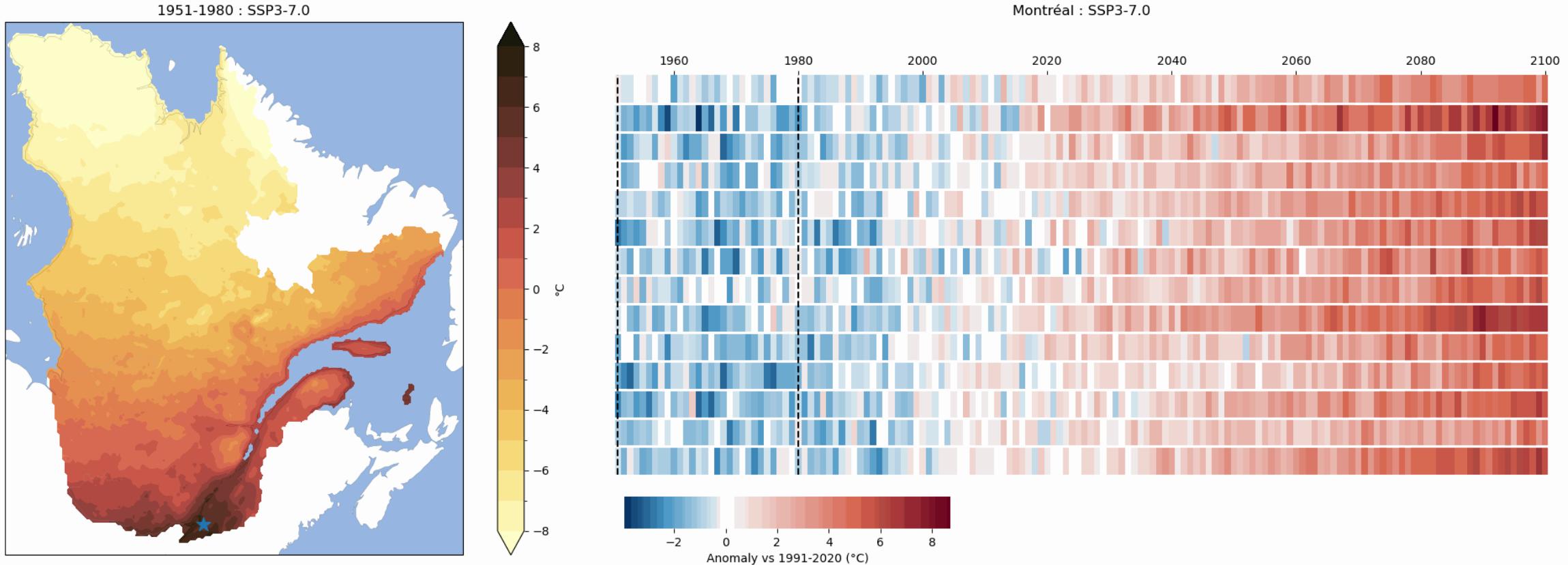
What does xclim do? □ Missing Data and Metadata Locales

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

import import
ds = xr
with xc
# set
) :
    met
) :
    # C
FD

  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 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.
```

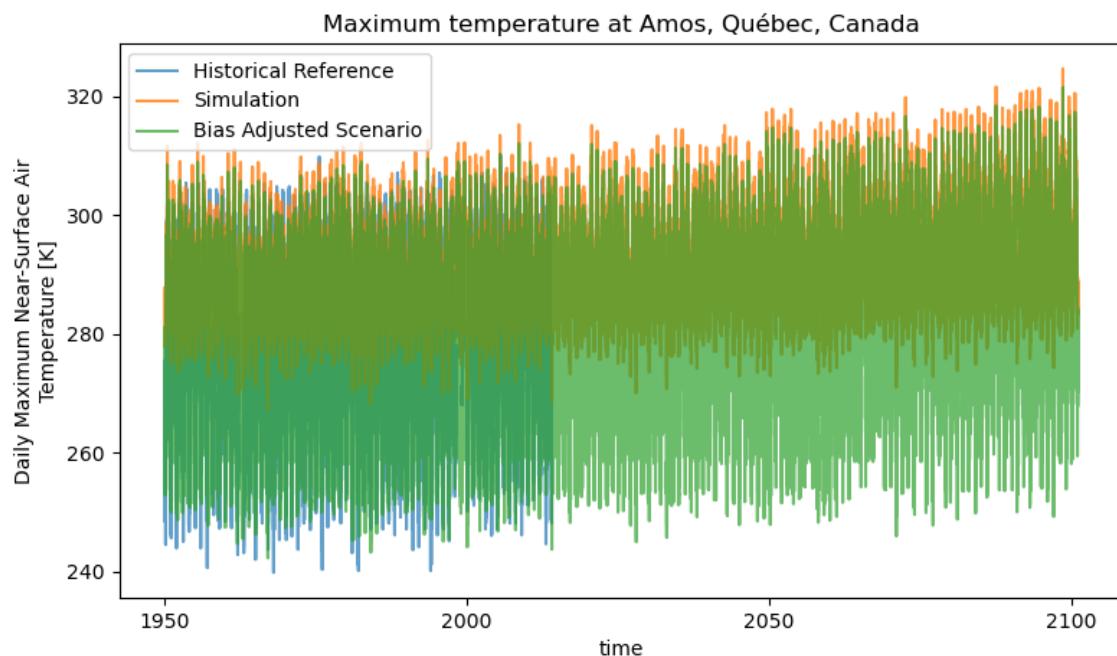
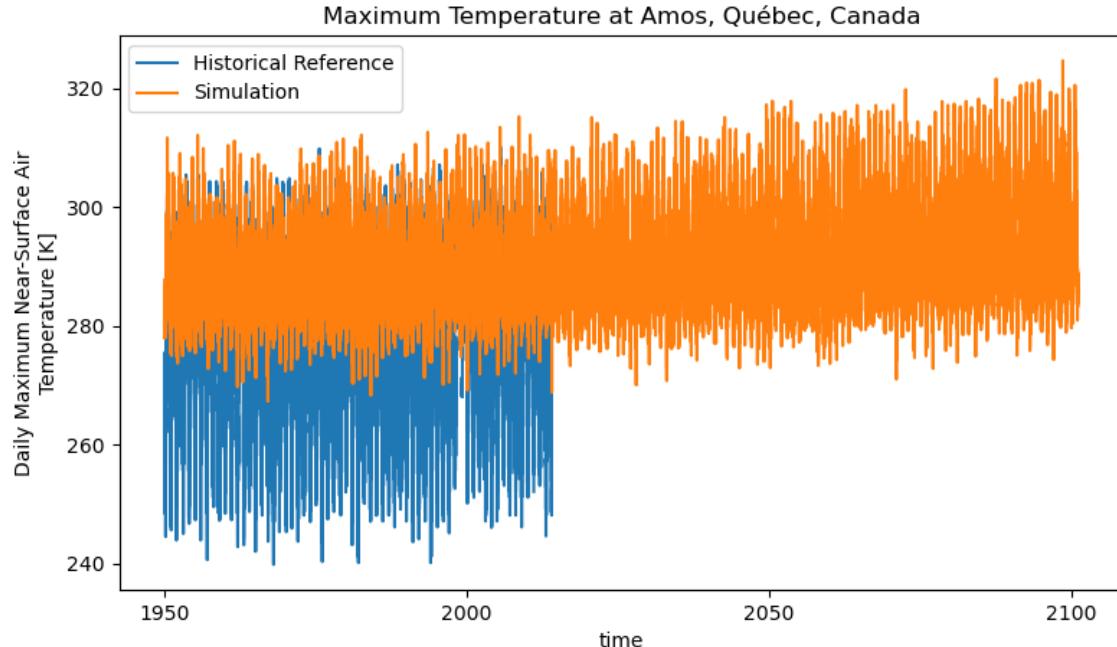
What does Xclim do □ Climate Ensemble Mean Analysis



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 statistics package **SBCK**
 - `dOTC` , `CDFt` , others



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 tools

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 not powerful enough for the analyses
- The user knows programming but not Python :
 - A biologist who uses R or a different program for their work
 - An engineer who just needs a range of estimates for future rainfall
- The user just wants to see some custom maps :
 - Agronomist who is curious about average growing conditions in 10 years?

Tools



xclim



XSCEN

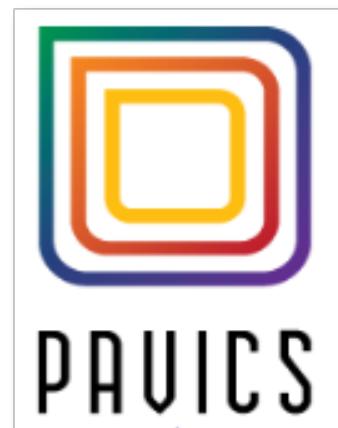


MIRANDA

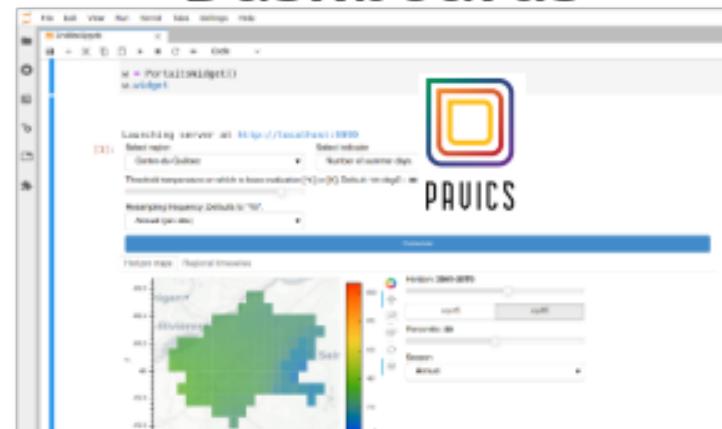


RavenPy

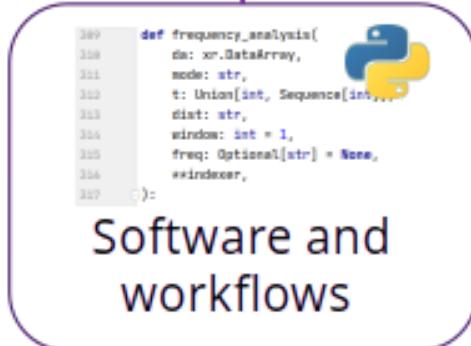
Platforms



Dashboards

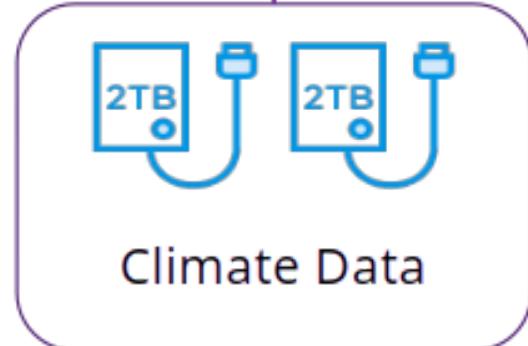


Climate Portraits

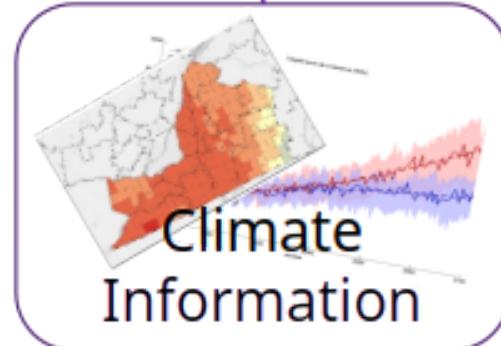


```
389 def frequency_analysis(  
390     da: xr.DataArray,  
391     mode: str,  
392     t: Union[int, Sequence[int],  
393             str],  
394     diast: str,  
395     window: int = 1,  
396     freq: Optional[str] = None,  
397     #indexer,  
398 ):
```

Software and workflows

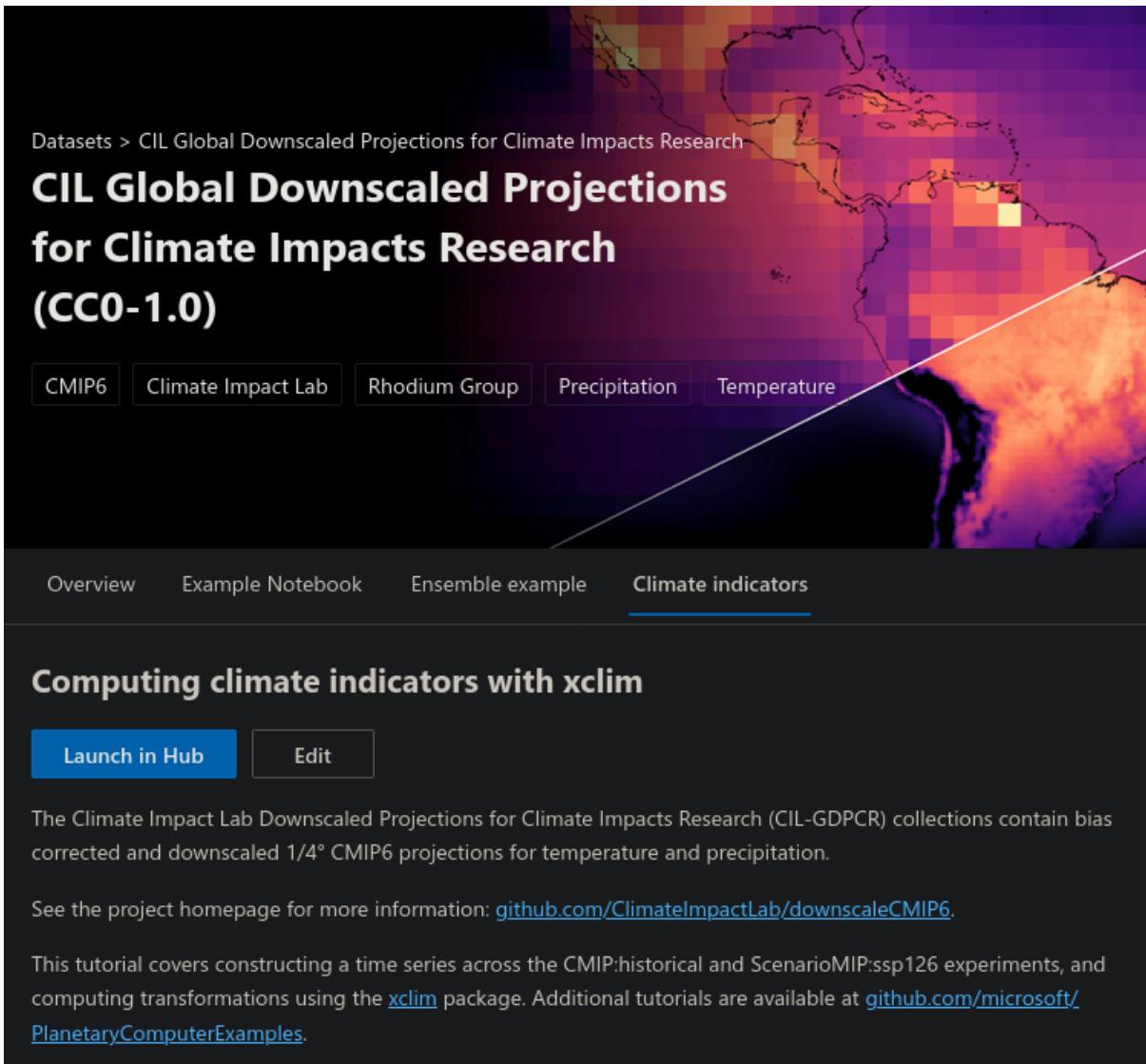


Climate Data



Climate Information





Xclim on Computation Platforms

Microsoft Planetary Computer

- Computing Climate Indicators with xclim

Enhancing Accessibility : Web Services

- WMS
 - Google Maps
- WFS
- WCS
- **WPS : *Web Processing Service***
 - Running geospatial analyses over the internet

Finch

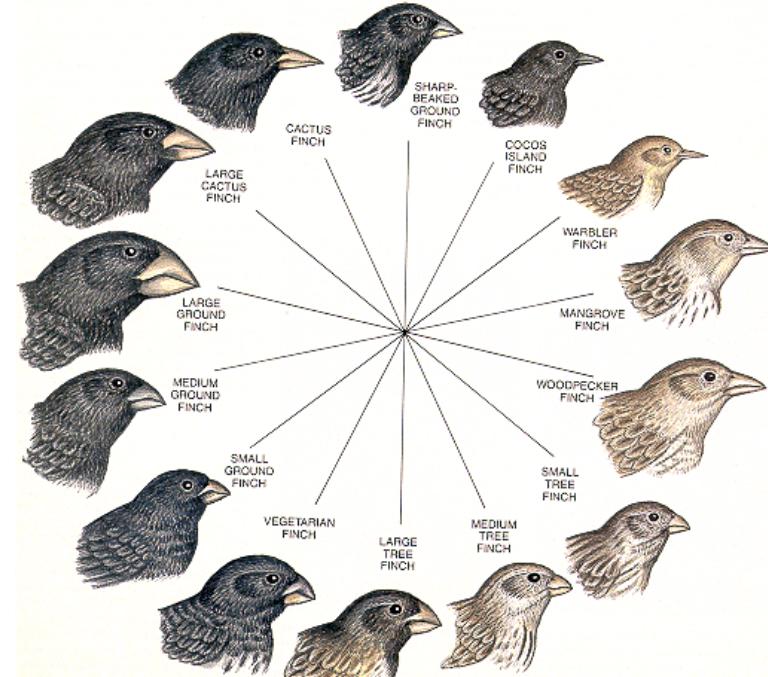
Xclim as a Web Service

 github.com/Bird-house/Finch

- Remote analysis of climate indicators
- Implements **Web Processing Service (WPS)** standard
 - Built in Python (**PyWPS**)
- *Bird-house likes to name their projects after birds*



The screenshot shows the GitHub repository page for 'bird-house/Finch'. At the top right, there's a logo consisting of four feathers in blue, yellow, green, and orange arranged in a circle. Below the logo, the repository name 'bird-house' is displayed in large bold letters, followed by the tagline 'Making Web Processing Services easy'. To the left of the tagline is a user icon and the text '5 followers'. To the right is a link 'http://bird-house.github.io/'. Below this header, there are navigation links: 'Overview' (selected), 'Repositories' (with 63 items), and 'Projects' (with 4 items). The main content area below these links is currently empty.



Using the Finch Web Service from 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)) # Hundreds of dynamically generated indicators and analyses!
```

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

```
from birdy import WPSClient

wps = WPSClient(finch_url)

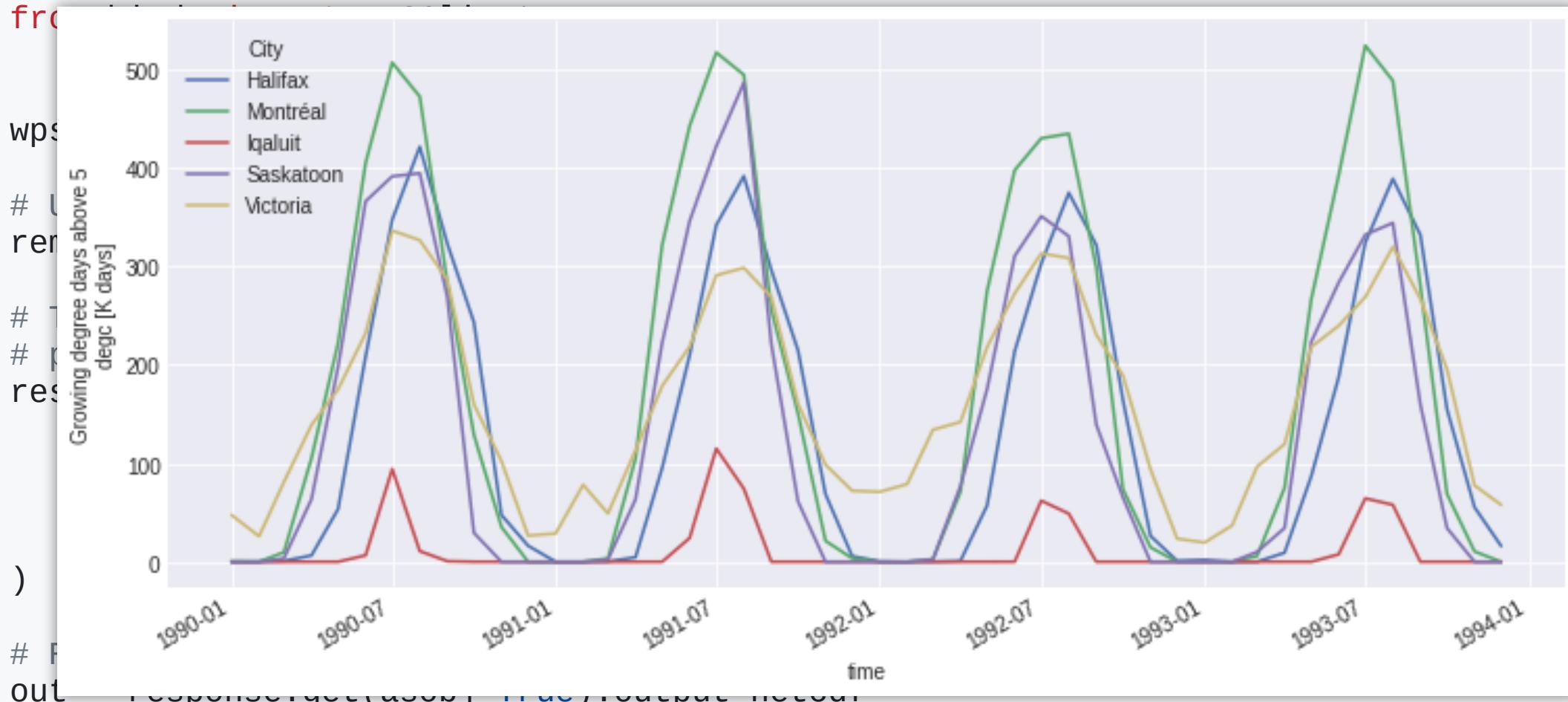
# Using the OPeNDAP protocol
remote_dataset = "www.exampledata.lt/climate.ncml"

# 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')
```

Using the Finch Web Service from Python (birdy)



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



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
- CMIP 5
- CMIP 6

VIEW BY:
Gridded data

100
Days

ANNUAL

0

Making it accessible □ Web Frontends

ClimateData.ca

TIME PERIOD

2041-2070

OPACITY

EXPORT MAP IMAGE

1 CHOOSE A DATASET

DATASET	LOCATION	VARIABLE(S)	TIMEFRAME	OPTIONS
Saint-Sauveur CMIP6 (CanDCS- U6) Prévost	15 Grids	Degree Days Below a Threshold	2030 – 2060	1 Emissions Scenario 3 Percentiles

2 SELECT LOCATIONS

3 CUSTOMIZE VARIABLES

4 CHOOSE A TIMEFRAME

5 ADVANCED

Models

Full ensemble

Emissions Scenarios

SSP1-
[6](#) SSP2-
[5](#) SSP5-8.5

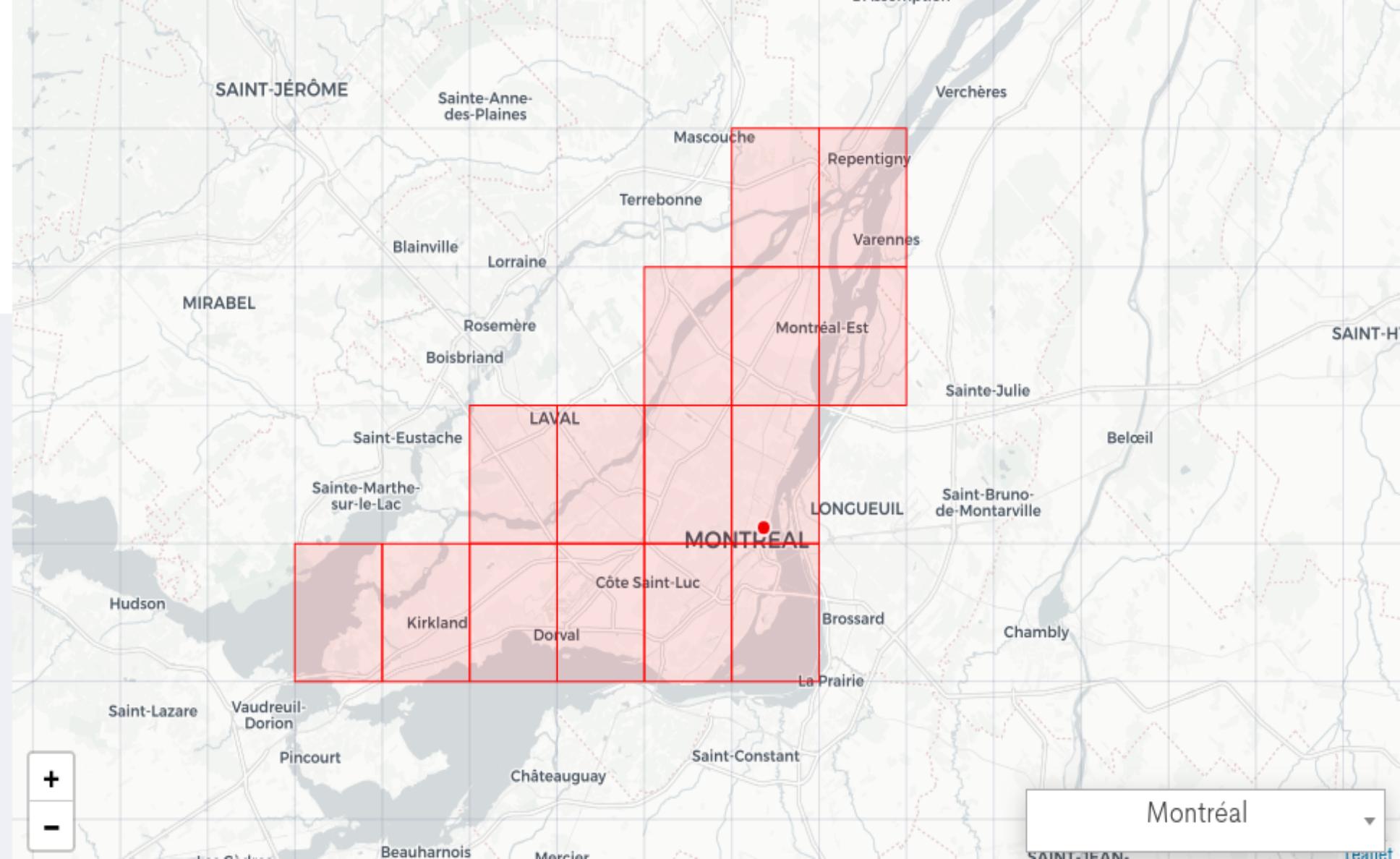
Percentiles (Unselect all to receive
output from individual
models)

5 10 25 50

75 90 95

Temporal Frequency

Annual Monthly



Modern-day Climate Science/Services

- 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` , `finch`)
- Docker-deployed Web-Service-based infrastructure
- Testing (`pytest`), CI/CD, and data validation workflows
- Peer-Reviewed Software (JOSS)

Thanks!

Ačiū!

Colleagues and Collaborators

- Pascal Bourgault
 - David Huard
 - Travis Logan
 - Abel Aoun
 - Juliette Lavoie
 - Éric Dupuis
 - Gabriel Rondeau-Genesse
 - Carsten Ehbrecht
 - Long Vu
 - Sarah Gammon
 - David Caron
- and many more contributors!**

Have a great rest of PyCon Lithuania!

Ouranosinc/xclim

JOSS [10.21105/joss.05415](https://doi.org/10.21105/joss.05415)

DOI [10.5281/zenodo.10710942](https://doi.org/10.5281/zenodo.10710942)

Bird-house/finch

DOI [10.5281/zenodo.10870939](https://doi.org/10.5281/zenodo.10870939)