



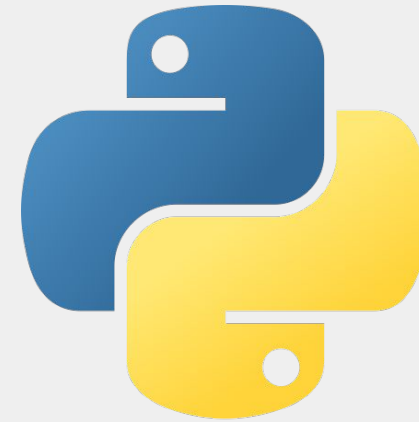
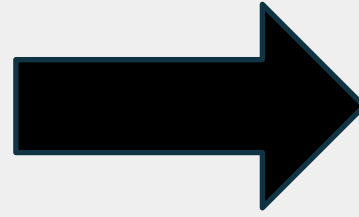
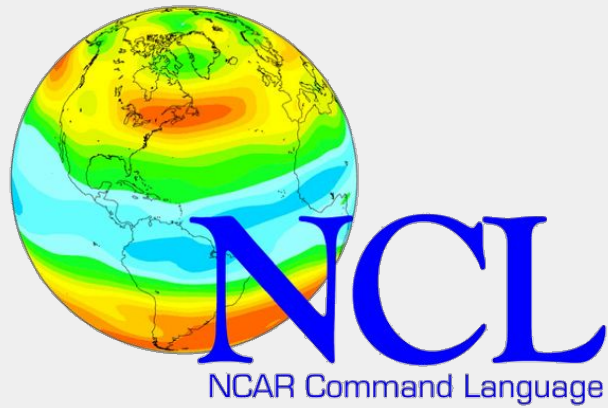
Expanding GeoCAT's Climatology Resources to Support the Transition from NCL to Python



Andy McKeen,
Vermont State University, NSF NCAR
Mentors: Anissa Zacharias & Katelyn FitzGerald
July 31, 2024



NCL to Python



- COMP
- EXAMPLES
- VIZ
- DATAFILES
- WRF-PYTHON
- APPLICATIONS

GeoCAT Applications

- Inspired by the NCL applications page
- Designed to be a quick reference demonstrating capabilities within the scientific Python ecosystem
- Comprised of two main sections

The screenshot shows the GeoCAT Applications website. On the left is a navigation sidebar with the following sections: Applications (Plot Types, Data Analysis, Date and Time), NCL to Python (NCL Index, NCL Applications), Contributing (Contributor Guide, Code of Conduct), and Getting Support (GitHub Issues, Feature Request Form). The main content area has a header with the GeoCAT logo and a search bar. Below the header, the title 'GeoCAT Applications' is followed by a paragraph: 'GeoCAT Applications is a community resource managed by the GeoCAT team. Inspired by the [NCL Applications](#) page, this is designed to be a quick reference demonstrating capabilities within the Scientific Python Ecosystem that may be relevant to your geoscience workflows.' The 'Python Examples' section contains three boxes: 'Dates and Times' with a link to 'Working with Date and Time', 'Data Analysis' with a link to 'Calculating Climatologies', and 'Regridding'. There is also a 'Plot Types' box. A green tip box states: 'Tip: If you're looking for NCL to Python examples, please visit [NCL Applications](#).' At the bottom right, there is a 'Next >' link to 'Plot Types'. The footer contains copyright information: 'By GeoCAT. © Copyright 2024, University Corporation for Atmospheric Research. The National Center for Atmospheric Research is sponsored by the National Science Foundation. Any opinions, findings and conclusions or recommendations expressed in this material do not necessarily reflect the views of the National Science Foundation.'

GeoCAT Applications

- Inspired by the NCL applications page
- Designed to be a quick reference demonstrating capabilities within the scientific Python ecosystem
- Comprised of two main sections

The screenshot shows the GeoCAT Applications website. The left sidebar contains a search bar and two red-bordered sections: 'Applications' with sub-items 'Plot Types', 'Data Analysis', and 'Date and Time'; and 'NCL to Python' with sub-items 'NCL Index' and 'NCL Applications'. Below these are sections for 'Contributing' and 'Getting Support'. The main content area features the title 'GeoCAT Applications', a brief description, and a 'Python Examples' section with three boxes: 'Dates and Times' (with a link to 'Working with Date and Time'), 'Data Analysis' (with a link to 'Calculating Climatologies'), and 'Regridding'. A 'Plot Types' box is also present. A green tip box contains the text: 'Tip: If you're looking for NCL to Python examples, please visit [NCL Applications](#).' A 'Next >' link points to 'Plot Types'. The footer includes copyright information for GeoCAT and the National Science Foundation.

Calculating the day of the week

July						
Sun	Mon	Tue	Wed	Thu	Fri	Sat
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

Calculating the day of the week

Grab and Go

```
import cftime

day = 4
month = 6
year = 2024

dow = cftime.datetime(
    year, month, day, calendar='proleptic_gregorian', has_year_zero=True
).strftime("%w")
print(dow)
```

2

Calculating the day of the week

Grab and Go

```
import cftime

day = 4
month = 6
year = 2024

dow = cftime.datetime(
    year, month, day, calendar='proleptic_gregorian', has_year_zero=True
).strftime("%w")
print(dow)
```

2

Calculating the day of the week

Grab and Go

```
import cftime

day = 4
month = 6
year = 2024

dow = cftime.datetime(
    year, month, day, calendar='proleptic_gregorian', has_year_zero=True
).strftime("%w")
print(dow)
```

2

Calculating the day of the week

Grab and Go

```
import cftime

day = 4
month = 6
year = 2024

dow = cftime.datetime(
    year, month, day, calendar='proleptic_gregorian', has_year_zero=True
).strftime("%w")
print(dow)
```

2

Calculating the day of the week

Grab and Go

```
import cftime

day = 4
month = 6
year = 2024

dow = cftime.datetime(
    year, month, day, calendar='proleptic_gregorian', has_year_zero=True
).strftime("%w")
print(dow)
```

2

Year 0

Python

January							February						
Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa
						1			1	2	3	4	5
2	3	4	5	6	7	8	6	7	8	9	10	11	12
9	10	11	12	13	14	15	13	14	15	16	17	18	19
16	17	18	19	20	21	22	20	21	22	23	24	25	26
23	24	25	26	27	28	29	27	28	29				
30	31												

NCL

January							February						
Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa
1	2	3	4	5	6	7				1	2	3	4
8	9	10	11	12	13	14	5	6	7	8	9	10	11
15	16	17	18	19	20	21	12	13	14	15	16	17	18
22	23	24	25	26	27	28	19	20	21	22	23	24	25
29	30	31					26	27	28	29			

Year 0

Python & NCL



March						
Su	Mo	Tu	We	Th	Fr	Sa
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	



Applications

The screenshot shows the 'Calculating Climatologies' page on the GeoCAT Applications website. The left sidebar contains a navigation menu with 'Applications' highlighted in red. Under 'Applications', 'Calculating Climatologies' is selected. The main content area includes an 'Overview' section with a list of bullet points, an 'Example Data' section with a code block and a terminal output, and a 'Contents' sidebar on the right.

```
from pythia_datasets import DATASETS
import xarray as xr

# Get data
filepath = DATASETS.fetch("CESM2_sst_data.nc")
ds = xr.open_dataset(filepath)

ds
```

```
/Users/amckeen/miniconda3/envs/geocat-applications/lib/python3.12/site-packages/xarray/conve
var = coder.decode(var, name=name)

xarray.Dataset
```

NCL to Python

The screenshot shows the 'Climatology' page on the GeoCAT Applications website. The left sidebar contains a navigation menu with 'NCL to Python' highlighted in red. Under 'NCL to Python', 'Climatology' is selected. The main content area includes an 'Overview' section with a list of bullet points, a 'calcDayAnomTLL' section with a code block, and a 'Grab and Go' section with a code block and a terminal output.

```
import xarray as xr

ds = xr.open_dataset(
    "../ncl_faw/b_e21.BHIST.f09_g17.CMIP6-historical.003.cice.h1.aice_d.18500101-20141231.nc"
)
aice = ds.aice_d[54749:58399, :, :]
DayTLL = aice.groupby(aice.time.dt.dayofyear)
cImDayTLL = DayTLL.mean(dim="time")
calcDayAnomTLL = DayTLL - cImDayTLL

calcDayAnomTLL[0, 365, 300]
```

```
xarray.DataArray 'aice_d'
```

Applications

The screenshot shows the GeoCAT Applications website. The main content area is titled "Calculating Climatologies" and includes an "Overview" section with bullet points about working with `xarray` and `groupby()`. Below this is an "Example Data" section with a code block for loading data from a repository. A red box highlights the "Data Analysis" menu item in the left sidebar, which also contains "Calculating Climatologies" and "Date and Time".

NCL to Python

The screenshot shows the GeoCAT Applications website. The main content area is titled "Climatology" and includes an "Overview" section explaining that climatology functions can be calculated using `xarray` and `geocat.comp`. Below this is a "Grab and Go" section with a code block for loading data and calculating anomalies. The left sidebar has the "Climatology" menu item highlighted.

Applications

The screenshot shows the 'Calculating Climatologies' notebook page on the GeoCAT Applications website. The page title is 'Calculating Climatologies' and it includes an 'Overview' section with the text: 'Calculating climatologies in Python is a common task in geoscience workflows. This notebook will cover:'. Below this, there are two bullet points: 'Working with `xarray` and its `groupby()` functionality' and 'A resource guide to point you to more detailed information depending on your use case'. The 'Example Data' section states: 'The dataset used in this notebook originated from the Community Earth System Model v2 (CESM2), and is retrieved from the [Pythia-datasets repository](#)'. It further explains: 'The dataset contains 15 years of monthly mean sea surface temperatures (TOS) from January 2000 to December 2014'. A code block shows the following Python code:

```
from pythia_datasets import DATASETS
import xarray as xr

# Get data
filepath = DATASETS.fetch("CESM2_sst_data.nc")
ds = xr.open_dataset(filepath)

ds
```

 Below the code block, the output is shown as `xarray.Dataset`. The left sidebar contains navigation links for 'Applications', 'Data Analysis', 'Date and Time', 'NCL to Python', 'Contributing', and 'Getting Support'. The right sidebar shows a 'Contents' table of contents.

NCL to Python

The screenshot shows the 'Climatology' notebook page on the GeoCAT Applications website. The page title is 'Climatology' and it includes an 'Overview' section with the text: 'The climatology functions listed below can be calculated using `xarray` and/or `geocat.comp`'. Below this, there are several bullet points listing functions: `calcDayAnomTLL`, `calcMonAnomTLL`, `clmDayTLL`, `clmMonTLL`, `month_to_season`, `rmMonAnnCycTLL`, and `stdMonTLL`. The 'calcDayAnomTLL' section states: '`calcDayAnomTLL` calculates daily anomalies from a daily data climatology'. The 'Grab and Go' section shows a code block with the following Python code:

```
import xarray as xr

ds = xr.open_dataset(
    "../ncl_faw/b_e21.BHIST.f09_g17.CMIP6-historical.003.cice.h1.aice_d.18500101-20141231.nc"
)
aice = ds.aice_d[54749:58399, :, :]
DayTLL = aice.groupby(aice.time.dt.dayofyear)
clmDayTLL = DayTLL.mean(dim="time")
calcDayAnomTLL = DayTLL - clmDayTLL

calcDayAnomTLL[0, 365, 300]
```

 Below the code block, the output is shown as `xarray.DataArray 'aice_d'`. The left sidebar contains navigation links for 'Applications', 'Data Analysis', 'Date and Time', 'NCL to Python', 'Contributing', and 'Getting Support'. The right sidebar shows a 'Contents' table of contents. A red box highlights the 'NCL to Python' section in the left sidebar.

Applications

The screenshot shows the NCAR GeoCAT website interface. The main heading is "Calculating Climatologies". Under the "Overview" section, it states: "Calculating climatologies in Python is a common task in geoscience workflows. This notebook will cover:" followed by two bullet points: "Working with `xarray` and its `groupby()` functionality" and "A resource guide to point you to more detailed information depending on your use case". Below this is the "Example Data" section, which explains that the dataset is from the Community Earth System Model v2 (CESM2) and contains 15 years of monthly mean sea surface temperatures (TOS) from January 2000 to December 2014. A code block shows the following Python code:

```
from pythia_datasets import DATASETS
import xarray as xr

# Get data
filepath = DATASETS.fetch("CESM2_sst_data.nc")
ds = xr.open_dataset(filepath)

ds
```

The output of the code is shown as `xarray.Dataset`. A red box highlights the "Applications" menu in the left sidebar, which includes "Data Analysis" and "Calculating Climatologies".

NCL to Python

The screenshot shows the NCAR GeoCAT website interface for the "Climatology" notebook. The main heading is "Climatology". Under the "Overview" section, it states: "The climatology functions listed below can be calculated using `xarray` and/or `geocat.comp`". Below this is a list of functions: `calcDayAnomTLL`, `calcMonAnomTLL`, `clmDayTLL`, `clmMonTLL`, `month_to_season`, `rmMonAnnCycTLL`, and `stdMonTLL`. The "Grab and Go" section contains a code block with the following Python code:

```
import xarray as xr

ds = xr.open_dataset(
    "../ncl_faw/b.e21.BHIST.f09_g17.CMIP6-historical.003.cice.h1.aice_d.18500101-20141231.nc"
)
aice = ds.aice_d[54749:58399, :, :]
DayTLL = aice.groupby(aice.time.dt.dayofyear)
clmDayTLL = DayTLL.mean(dim="time")
calcDayAnomTLL = DayTLL - clmDayTLL

calcDayAnomTLL[0, 365, 300]
```

The output of the code is shown as `xarray.DataArray 'aice_d'`. The left sidebar shows the "NCL to Python" menu, which includes "NCL Applications" and "Climatology".

Calculating Long Term Means

```
from pythia_datasets import DATASETS
import xarray as xr
import matplotlib.pyplot as plt

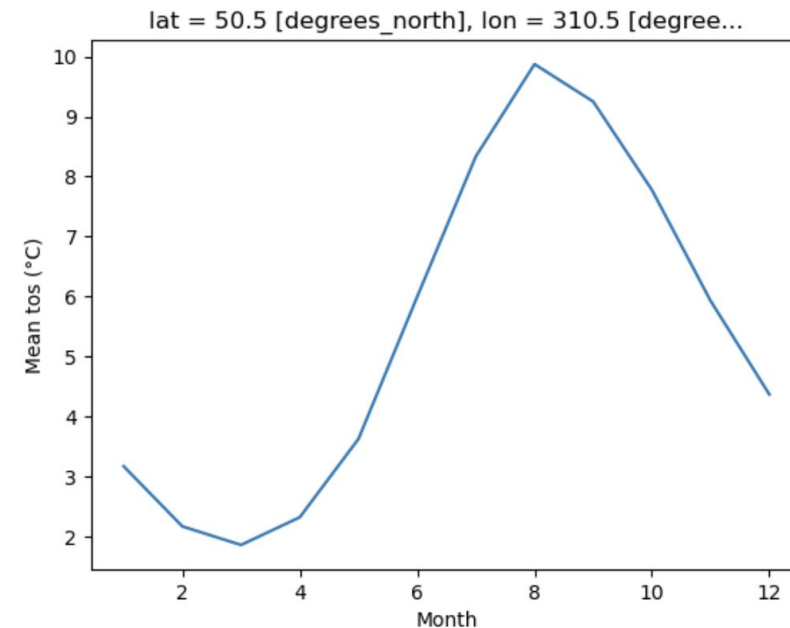
# Get data
filepath = DATASETS.fetch("CESM2_sst_data.nc")
ds = xr.open_dataset(filepath)

# Calculate long term mean
tos_monthly = ds.tos.groupby(ds.time.dt.month)
tos_clim = tos_monthly.mean(dim="time")

tos_clim
```

Visualization

```
# Plot an example location of the calculated long term means
tos_clim.sel(lon=310, lat=50, method="nearest").plot()
plt.ylabel("Mean tos (°C)")
plt.xlabel("Month")
```



Calculating Long Term Means

```
from pythia_datasets import DATASETS
import xarray as xr
import matplotlib.pyplot as plt

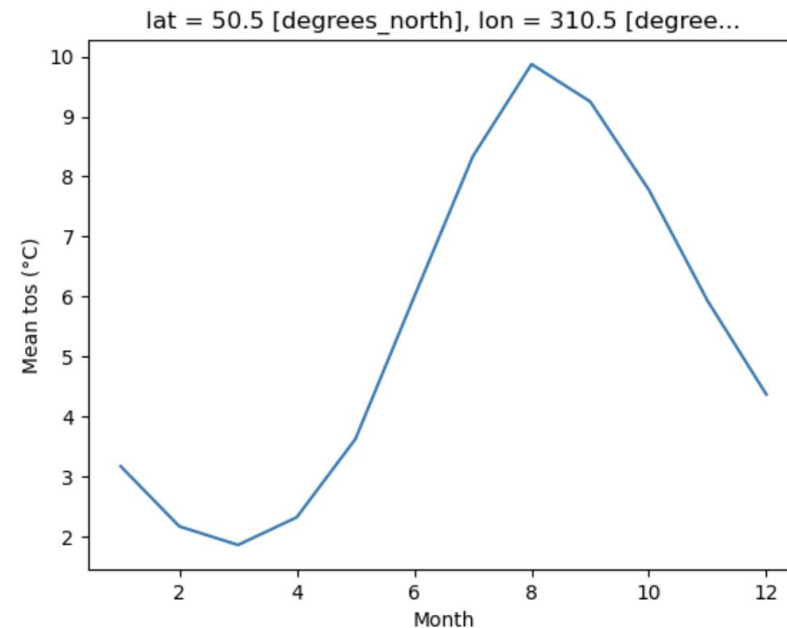
# Get data
filepath = DATASETS.fetch("CESM2_sst_data.nc")
ds = xr.open_dataset(filepath)

# Calculate long term mean
tos_monthly = ds.tos.groupby(ds.time.dt.month)
tos_clim = tos_monthly.mean(dim="time")

tos_clim
```

Visualization

```
# Plot an example location of the calculated long term means
tos_clim.sel(lon=310, lat=50, method="nearest").plot()
plt.ylabel("Mean tos (°C)")
plt.xlabel("Month")
```



Calculating Long Term Means

```
from pythia_datasets import DATASETS
import xarray as xr
import matplotlib.pyplot as plt

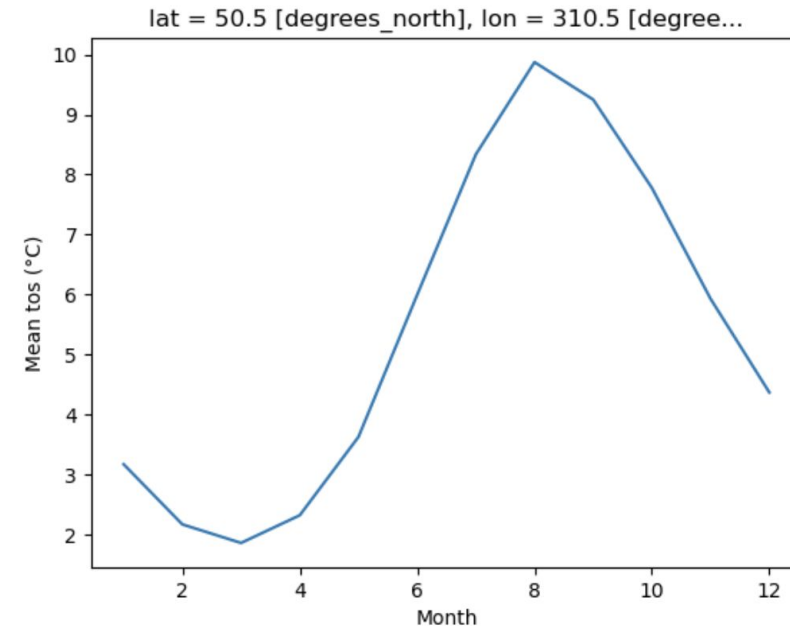
# Get data
filepath = DATASETS.fetch("CESM2_sst_data.nc")
ds = xr.open_dataset(filepath)

# Calculate long term mean
tos_monthly = ds.tos.groupby(ds.time.dt.month)
tos_clim = tos_monthly.mean(dim="time")

tos_clim
```

Visualization

```
# Plot an example location of the calculated long term means
tos_clim.sel(lon=310, lat=50, method="nearest").plot()
plt.ylabel("Mean tos (°C)")
plt.xlabel("Month")
```



Calculating Long Term Means

```
from pythia_datasets import DATASETS
import xarray as xr
import matplotlib.pyplot as plt

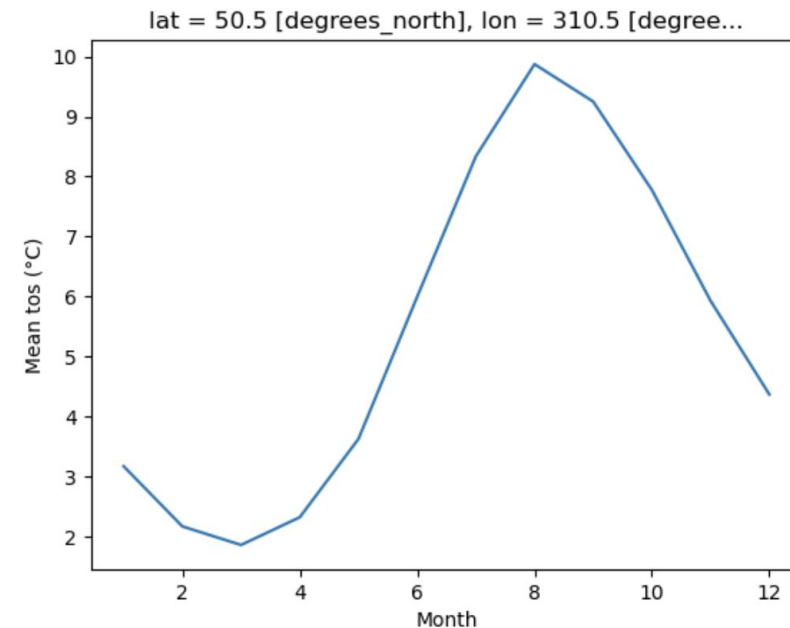
# Get data
filepath = DATASETS.fetch("CESM2_sst_data.nc")
ds = xr.open_dataset(filepath)

# Calculate long term mean
tos_monthly = ds.tos.groupby(ds.time.dt.month)
tos_clim = tos_monthly.mean(dim="time")

tos_clim
```

Visualization

```
# Plot an example location of the calculated long term means
tos_clim.sel(lon=310, lat=50, method="nearest").plot()
plt.ylabel("Mean tos (°C)")
plt.xlabel("Month")
```



Calculating Long Term Means

```
from pythia_datasets import DATASETS
import xarray as xr
import matplotlib.pyplot as plt

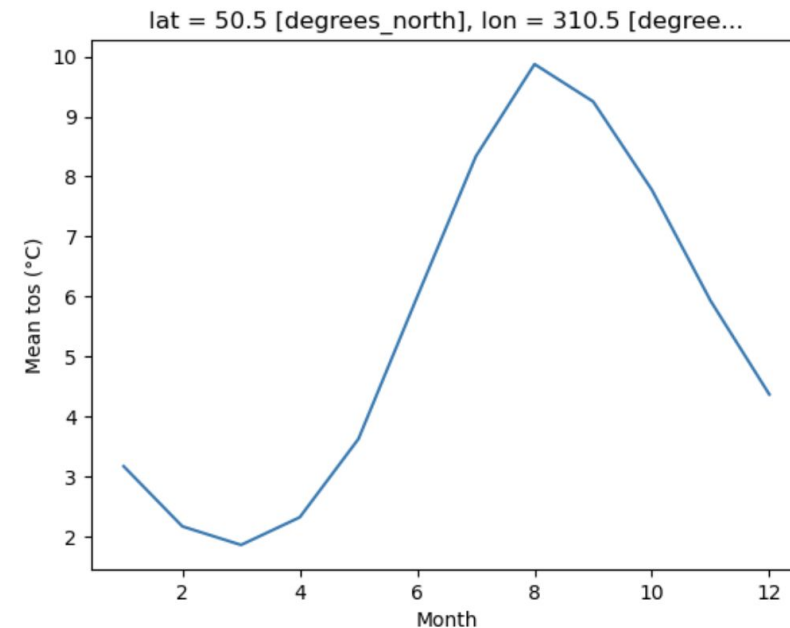
# Get data
filepath = DATASETS.fetch("CESM2_sst_data.nc")
ds = xr.open_dataset(filepath)

# Calculate long term mean
tos_monthly = ds.tos.groupby(ds.time.dt.month)
tos_clim = tos_monthly.mean(dim="time")

tos_clim
```

Visualization

```
# Plot an example location of the calculated long term means
tos_clim.sel(lon=310, lat=50, method="nearest").plot()
plt.ylabel("Mean tos (°C)")
plt.xlabel("Month")
```



Curated Resources

To learn more about calculating climatologies in Python, we suggest:

- This [ClimateMatch Academy notebook](#) on xarray Data Analysis and Climatology
- This [Project Pythia Foundations tutorial](#) on Computations and Masks with xarray
- The [xarray user guide](#) on working with time series data

< Previous
[Data Analysis](#)

Next >
[Dates and Times](#)

Applications

The screenshot shows the NCAR GeoCAT website interface. The main content area is titled "Calculating Climatologies" and includes an "Overview" section with the text: "Calculating climatologies in Python is a common task in geoscience workflows. This notebook will cover:" followed by a bulleted list: "Working with `xarray` and its `groupby()` functionality" and "A resource guide to point you to more detailed information depending on your use case". Below this is an "Example Data" section with a code block:

```
from pythia_datasets import DATASETS
import xarray as xr

# Get data
filepath = DATASETS.fetch("CESM2_sst_data.nc")
ds = xr.open_dataset(filepath)

ds
```

The code block is followed by a terminal output showing the file path and the resulting `xarray.Dataset` object.

NCL to Python

The screenshot shows the NCAR GeoCAT website interface. The main content area is titled "Climatology" and includes an "Overview" section with the text: "The climatology functions listed below can be calculated using `xarray` and/or `geocat.comp`". Below this is a list of functions: `calcDayAnomTLL`, `calcMonAnomTLL`, `clmDayTLL`, `clmMonTLL`, `month_to_season`, `rmMonAnnCycTLL`, and `stdMonTLL`. Below the list is a section titled "calcDayAnomTLL" with the text: "`calcDayAnomTLL` calculates daily anomalies from a daily data climatology". Below this is a section titled "Grab and Go" with a code block:

```
import xarray as xr

ds = xr.open_dataset(
    "../ncl_raw/b_e21.BHIST.f09_g17.CMIP6-historical.003.cice.h1.aice_d.18500101-20141231.nc"
)
aice = ds.aice_d[54749:58399, :, :]
DayTLL = aice.groupby(aice.time.dt.dayofyear)
clmDayTLL = DayTLL.mean(dim="time")
calcDayAnomTLL = DayTLL - clmDayTLL

calcDayAnomTLL[0, 365, 300]
```

The code block is followed by a terminal output showing the resulting `xarray.DataArray` object: `xarray.DataArray 'aice_d'`.

Climatology

calcDayAnomTLL	Calculates daily anomalies from a daily data climatology.
calcMonAnomTLL	Calculates monthly anomalies by subtracting the long term mean from each point (time,lat,lon version)
clmDayTLL	Calculates long term daily means (daily climatology) from daily data.
clmMonTLL	Calculates long term monthly means (monthly climatology) from monthly data: (time,lat,lon) version
month_to_season	Computes a user-specified three-month seasonal mean (DJF, JFM, FMA, MAM, AMJ, MJJ, JJA, JAS, ASO, SON, OND, NDJ).
rmMonAnnCycTLL	Removes the annual cycle from "monthly" data.
stdMonTLL	Calculates standard deviations of monthly means.

NCL Index

NCL Function	Description
calcDayAnomTLL	Calculates daily anomalies from a daily data climatology
calcMonAnomTLL	Calculates monthly anomalies by subtracting the long term mean from each point
clmDayTLL	Calculates long term daily means (daily climatology) from daily data
clmMonTLL	Calculates long term monthly means (monthly climatology) from monthly data
month_to_season	Computes a user-specified three-month seasonal mean
rmMonAnnCycTLL	Removes the annual cycle from monthly data
stdMonTLL	Calculates standard deviations of monthly means

clmDayTLL

`clmDayTLL` calculates long term daily means (daily climatology) from daily data

Grab and Go

```
import xarray as xr

ds = xr.open_dataset(
    "../nc1_raw/b.e21.BHIST.f09_g17.CMIP6-historical.003.cice.h1.aice_d.18500101-20141231.nc"
)
aice = ds.aice_d[54749:58399, :, :]
DayTLL = aice.groupby(aice.time.dt.dayofyear)
clmDayTLL = DayTLL.mean(dim="time")

clmDayTLL[0, 365, 300]
```

xarray.DataArray 'aice_d'

array(0.40314254, dtype=float32)

```
; clmDayTLL
; Adapted from https://www.ncl.ucar.edu/Document/Functions/Contributed/clmDayTLL.shtml
f = addfile("b.e21.BHIST.f09_g17.CMIP6-historical.003.cice.h1.aice_d.18500101-20141231.nc", "r")
time = f->time(54749:58398)
TIME = cd_calendar(time,0)
year = toint(TIME(:,0))
month = toint(TIME(:,1))
day = toint(TIME(:,2))
ddd = day_of_year(year, month, day)
yyyyddd = year*1000+ddd
aice = f->aice_d(54749:58398, :, :)
aiceClmDay = clmDayTLL(aice, yyyyddd) ; aiceClmDay = 0.4031426
print(aiceClmDay(0,365,300))
```


clmDayTLL

`clmDayTLL` calculates long term daily means (daily climatology) from daily data

Grab and Go

```
import xarray as xr

ds = xr.open_dataset(
    "../ncl_raw/b.e21.BHIST.f09_g17.CMIP6-historical.003.cice.h1.aice_d.18500101-20141231.nc"
)
aice = ds.aice_d[54749:58399, :, :]
DayTLL = aice.groupby(aice.time.dt.dayofyear)
clmDayTLL = DayTLL.mean(dim="time")

clmDayTLL[0, 365, 300]
```

xarray.DataArray 'aice_d'

 array(0.40314254, dtype=float32)

```
; clmDayTLL
; Adapted from https://www.ncl.ucar.edu/Document/Functions/Contributed/clmDayTLL.shtml
f = addfile("b.e21.BHIST.f09_g17.CMIP6-historical.003.cice.h1.aice_d.18500101-20141231.nc", "r")
time = f->time(54749:58398)
TIME = cd_calendar(time,0)
year = toint(TIME(:,0))
month = toint(TIME(:,1))
day = toint(TIME(:,2))
ddd = day_of_year(year, month, day)
yyyyddd = year*1000+ddd
aice = f->aice_d(54749:58398, :, :)
aiceClmDay = clmDayTLL(aice, yyyyddd) ; aiceClmDay = 0.4031426
print(aiceClmDay(0,365,300))
```

month_to_season

`month_to_season` computes a user-specified three-month seasonal mean (DJF, JFM, FMA, MAM, AMJ, MJJ, JJA, JAS, ASO, SON, OND, NDJ)

`xarray` may not be sufficient to calculate custom seasonal means. If you need to work with custom seasons, we suggest using `geocat.comp.climatologies.month_to_season`


Grab and Go

```
import xarray as xr
import geocat.comp as gc

ds = xr.open_dataset(
    "../ncl_raw/b.e21.BHIST.f09_g17.CMIP6-historical.003.cice.h.aice.185001-201412.nc"
)
aice = ds.aice[1799:1919, :, :]
mon_to_season = gc.climatologies.month_to_season(aice, "ASO")

mon_to_season[0, 365, 300]
```

xarray.DataArray 'aice'

 array(4.9227783e-06, dtype=float32)

```
; month_to_season
; Adapted from https://www.ncl.ucar.edu/Document/Functions/Contributed/calcDayAnomTLL.shtml
f = addfile("b.e21.BHIST.f09_g17.CMIP6-historical.003.cice.h.aice.185001-201412.nc", "r")
aice = f->aice(1799:1918, :, :)
aiceSeason = month_to_season(aice, "ASO") ; aiceSeason = 4.922778e-06
print(aiceSeason(0, 365, 300))
```

Climatology Receipt

Comparison

rmMonAnnCycTLL:	
python:	0.09780758619308472
ncl:	0.09780759
calcMonAnomTLL:	
python:	0.09780758619308472
ncl:	0.09780759
clmDayTLL:	
python:	0.4031425416469574
ncl:	0.4031426
clmMonTLL:	
python:	0.126130610704422
ncl:	0.1261306
stdMonTLL:	
python:	0.10731684416532516
ncl:	0.1073168
month_to_season:	
python:	4.9227783165406436e-06
ncl:	4.922778e-06
calcDayAnomTLL:	
python:	0.21562078595161438
ncl:	0.2156208

Differences

```
for c in ncl_results.keys() & results.keys():  
    print(f"{c}:")  
    print(f"\t{results[c] - ncl_results[c]}")
```

```
rmMonAnnCycTLL:  
    -3.806915283011136e-09  
calcMonAnomTLL:  
    -3.806915283011136e-09  
clmDayTLL:  
    -5.835304262014063e-08  
clmMonTLL:  
    1.0704421987695056e-08  
stdMonTLL:  
    4.416532516093863e-08  
month_to_season:  
    3.165406432182791e-13  
calcDayAnomTLL:  
    -1.404838562146793e-08
```

Climatology Receipt

Comparison

rmMonAnnCycTLL:	
python:	0.09780758619308472
ncl:	0.09780759
calcMonAnomTLL:	
python:	0.09780758619308472
ncl:	0.09780759
clmDayTLL:	
python:	0.4031425416469574
ncl:	0.4031426
clmMonTLL:	
python:	0.126130610704422
ncl:	0.1261306
stdMonTLL:	
python:	0.10731684416532516
ncl:	0.1073168
month_to_season:	
python:	4.9227783165406436e-06
ncl:	4.922778e-06
calcDayAnomTLL:	
python:	0.21562078595161438
ncl:	0.2156208

Differences

```
for c in ncl_results.keys() & results.keys():  
    print(f"{c}:")  
    print(f"\t{results[c] - ncl_results[c]}")
```

```
rmMonAnnCycTLL:  
    -3.806915283011136e-09  
calcMonAnomTLL:  
    -3.806915283011136e-09  
clmDayTLL:  
    -5.835304262014063e-08  
clmMonTLL:  
    1.0704421987695056e-08  
stdMonTLL:  
    4.416532516093863e-08  
month_to_season:  
    3.165406432182791e-13  
calcDayAnomTLL:  
    -1.404838562146793e-08
```


Comparison

rmMonAnnCycTLL:	
python:	0.09780758619308472
ncl:	0.09780759
calcMonAnomTLL:	
python:	0.09780758619308472
ncl:	0.09780759
clmDayTLL:	
python:	0.4031425416469574
ncl:	0.4031426
clmMonTLL:	
python:	0.126130610704422
ncl:	0.1261306
stdMonTLL:	
python:	0.10731684416532516
ncl:	0.1073168
month_to_season:	
python:	4.9227783165406436e-06
ncl:	4.922778e-06
calcDayAnomTLL:	
python:	0.21562078595161438
ncl:	0.2156208

Differences

```
for c in ncl_results.keys() & results.keys():  
    print(f"{c}:")  
    print(f"\t{results[c] - ncl_results[c]}")
```

rmMonAnnCycTLL:	-3.806915283011136e-09
calcMonAnomTLL:	-3.806915283011136e-09
clmDayTLL:	-5.835304262014063e-08
clmMonTLL:	1.0704421987695056e-08
stdMonTLL:	4.416532516093863e-08
month_to_season:	3.165406432182791e-13
calcDayAnomTLL:	-1.404838562146793e-08

Comparison

rmMonAnnCycTLL:	
python:	0.09780758619308472
ncl:	0.09780759
calcMonAnomTLL:	
python:	0.09780758619308472
ncl:	0.09780759
clmDayTLL:	
python:	0.4031425416469574
ncl:	0.4031426
clmMonTLL:	
python:	0.126130610704422
ncl:	0.1261306
stdMonTLL:	
python:	0.10731684416532516
ncl:	0.1073168
month_to_season:	
python:	4.9227783165406436e-06
ncl:	4.922778e-06
calcDayAnomTLL:	
python:	0.21562078595161438
ncl:	0.2156208



Differences

```
for c in ncl_results.keys() & results.keys():  
    print(f"{c}:")  
    print(f"\t{results[c] - ncl_results[c]}")
```

```
rmMonAnnCycTLL:  
    -3.806915283011136e-09  
calcMonAnomTLL:  
    -3.806915283011136e-09  
clmDayTLL:  
    -5.835304262014063e-08  
clmMonTLL:  
    1.0704421987695056e-08  
stdMonTLL:  
    4.416532516093863e-08  
month_to_season:  
    3.165406432182791e-13  
calcDayAnomTLL:  
    -1.404838562146793e-08
```

Conclusion

Thank you to my mentors:
Anissa Zacharias & Katelyn FitzGerald

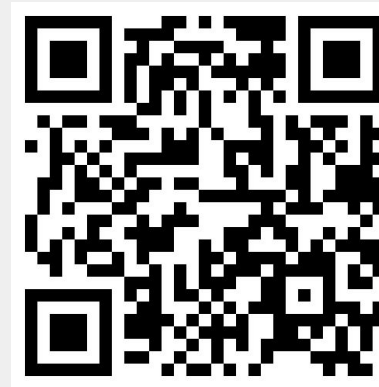
Thank you the GeoCAT team & the SIParCS program!



Questions?

Email me anytime:
andymckeen648@gmail.com

GitHub:
[andy-theia](#)



GeoCAT
Applications





Expanding GeoCAT's Climatology Resources to Support the Transition from NCL to Python



Andy McKeen,
Vermont State University, NSF NCAR
Mentors: Anissa Zacharias & Katelyn FitzGerald
July 31, 2024

