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

This material is based upon work supported by the National Center for Atmospheric Research, which is a major facility sponsored by the National Science Foundation under Cooperative Agreement No. 1852977.

NCL to Python

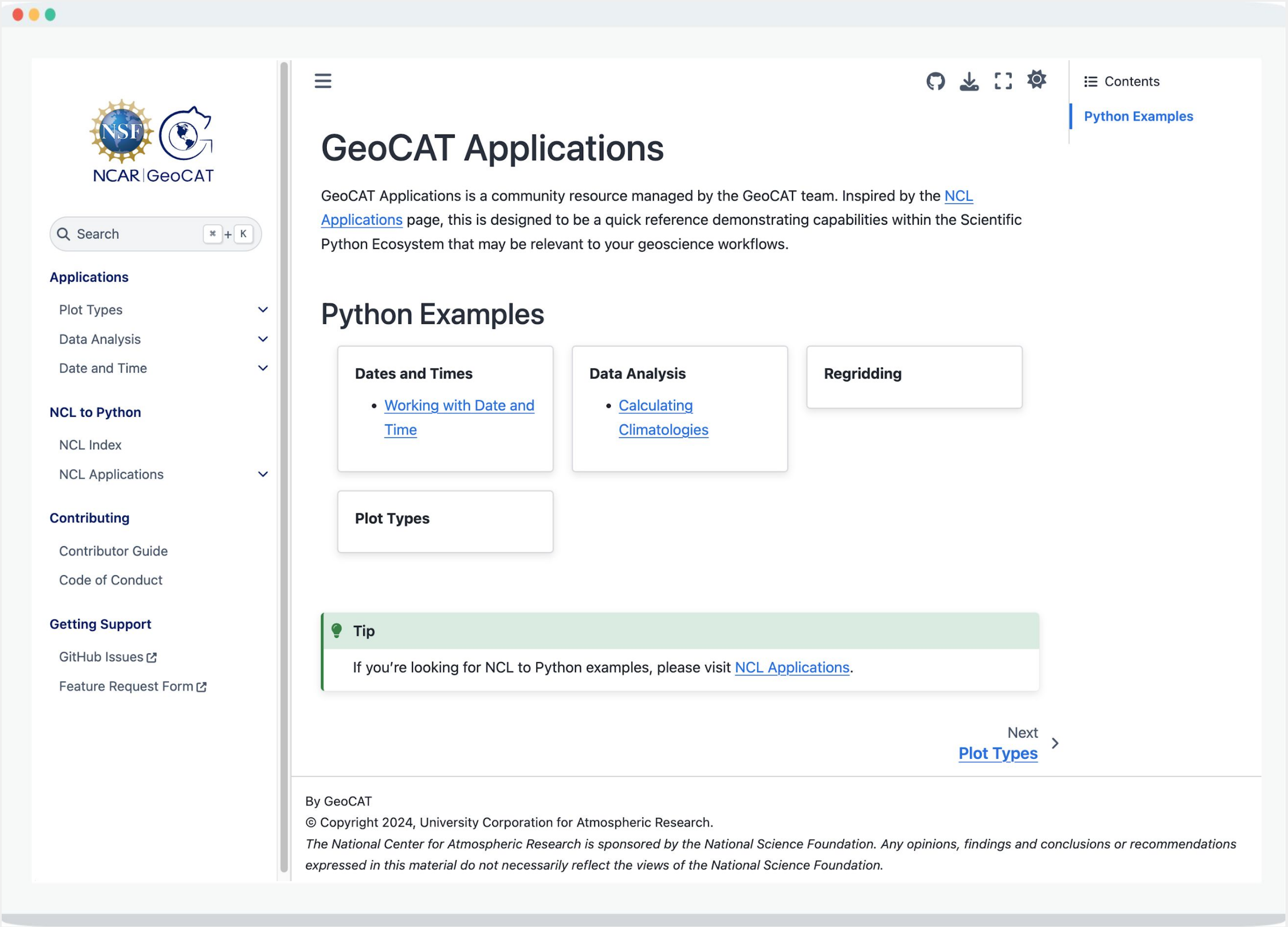
NCL
NCAR Command Language

GeoCAT

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

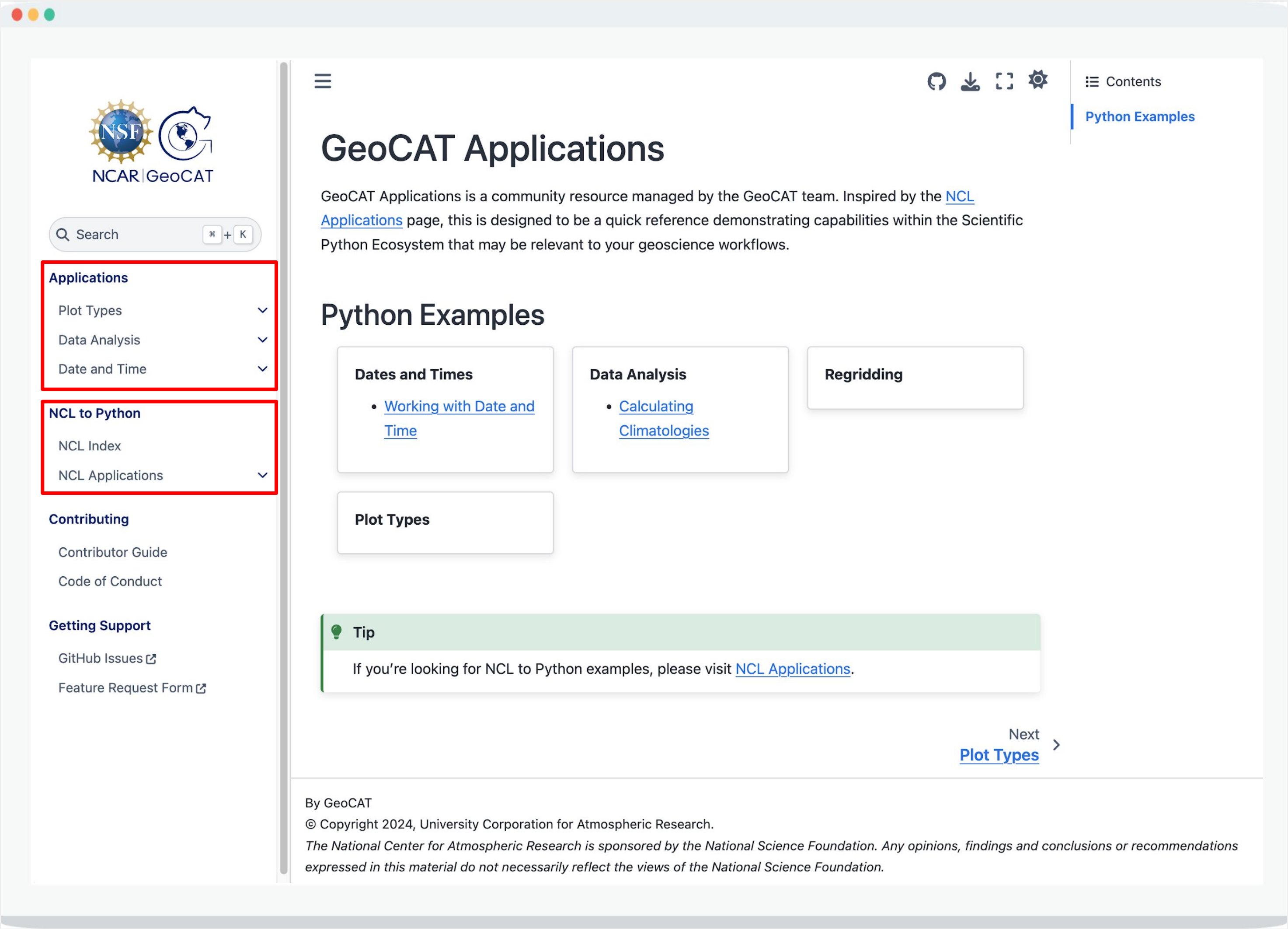
GeoCAT Applications

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



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



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

```python
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						
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					

February						
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				

NCL

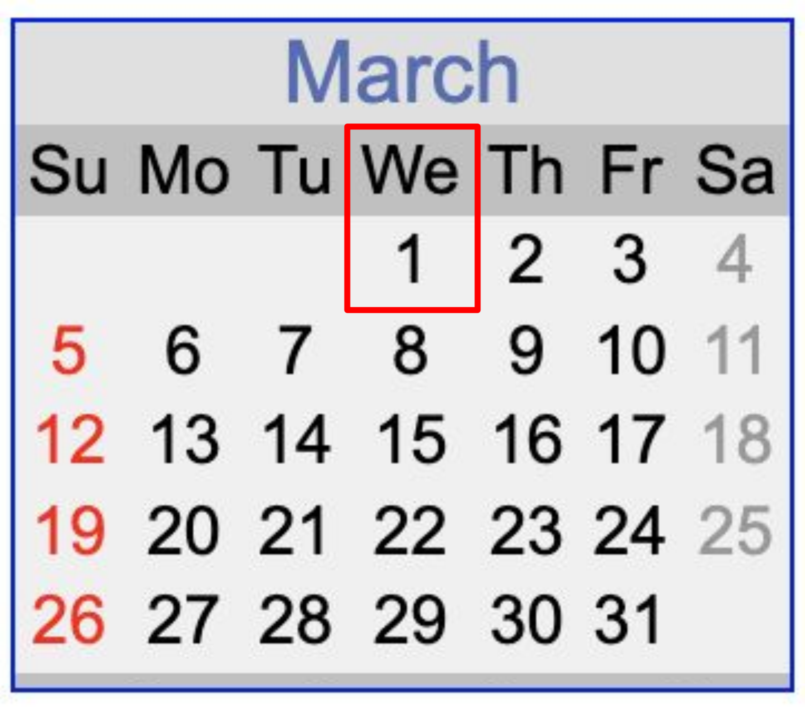
January						
Su	Mo	Tu	We	Th	Fr	Sa
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

February						
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			

Day of the Week

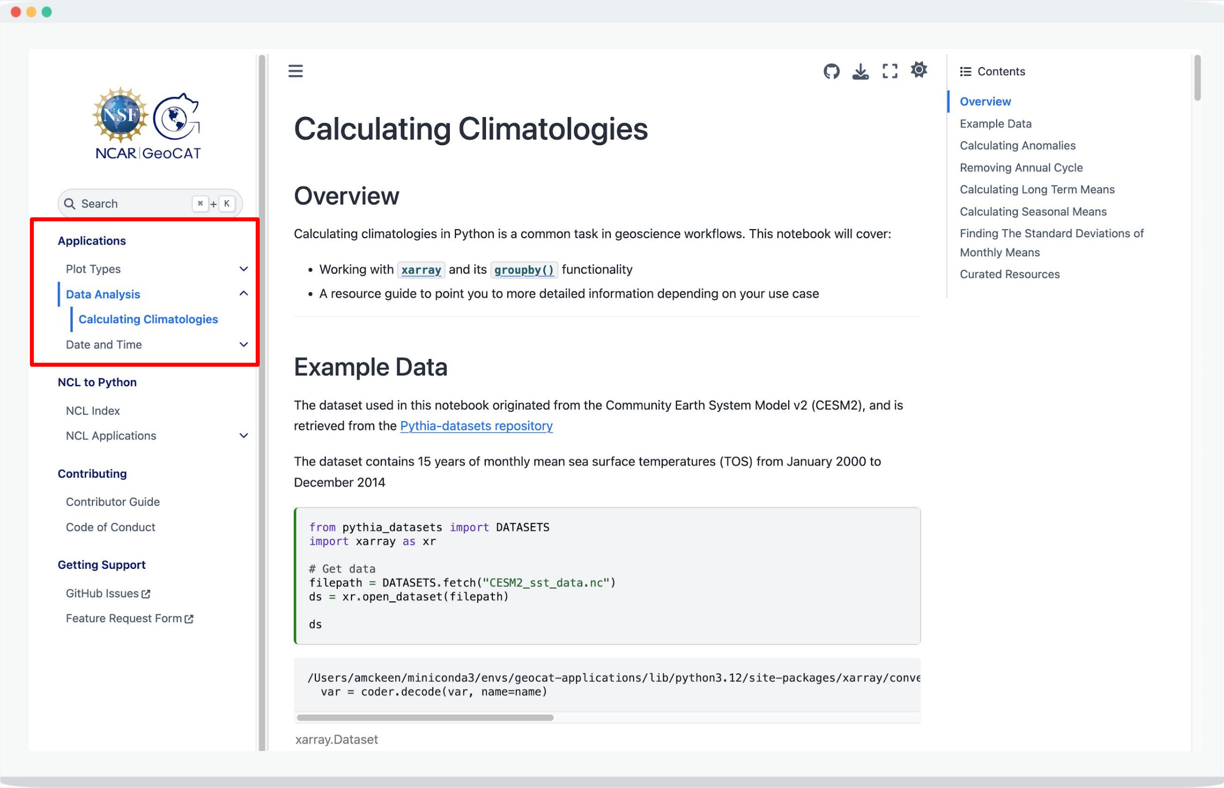
Year 0

Python & NCL

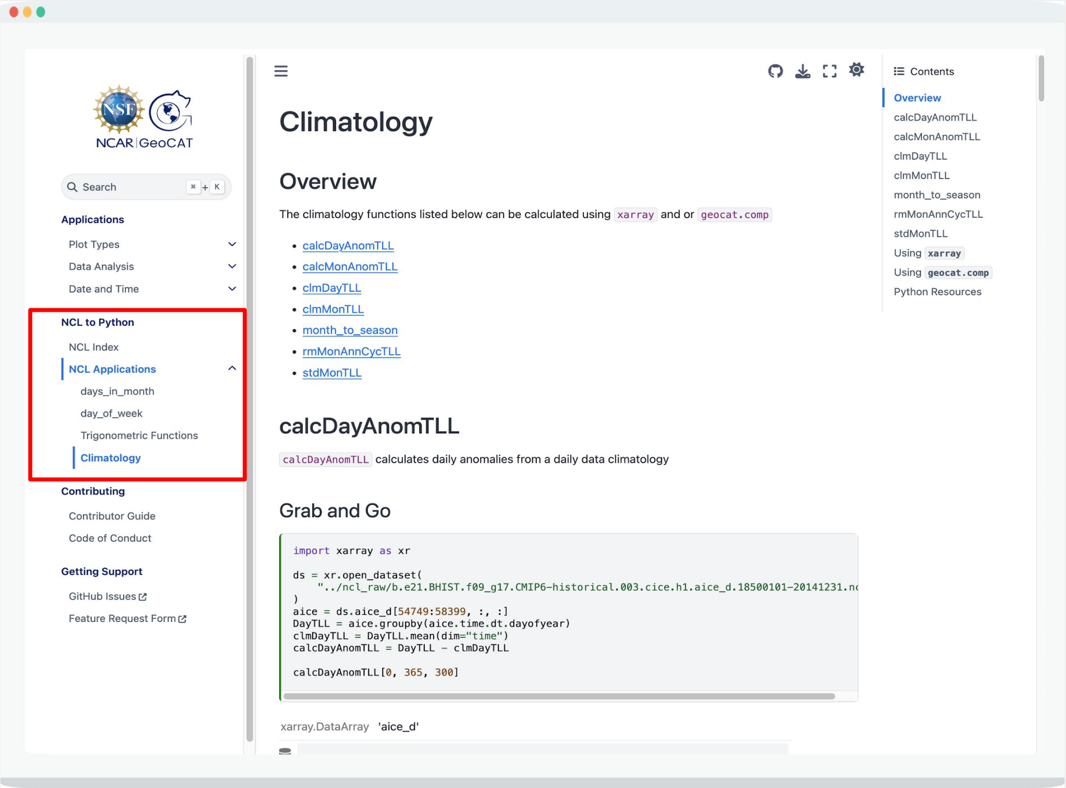


GeoCAT Applications

Applications

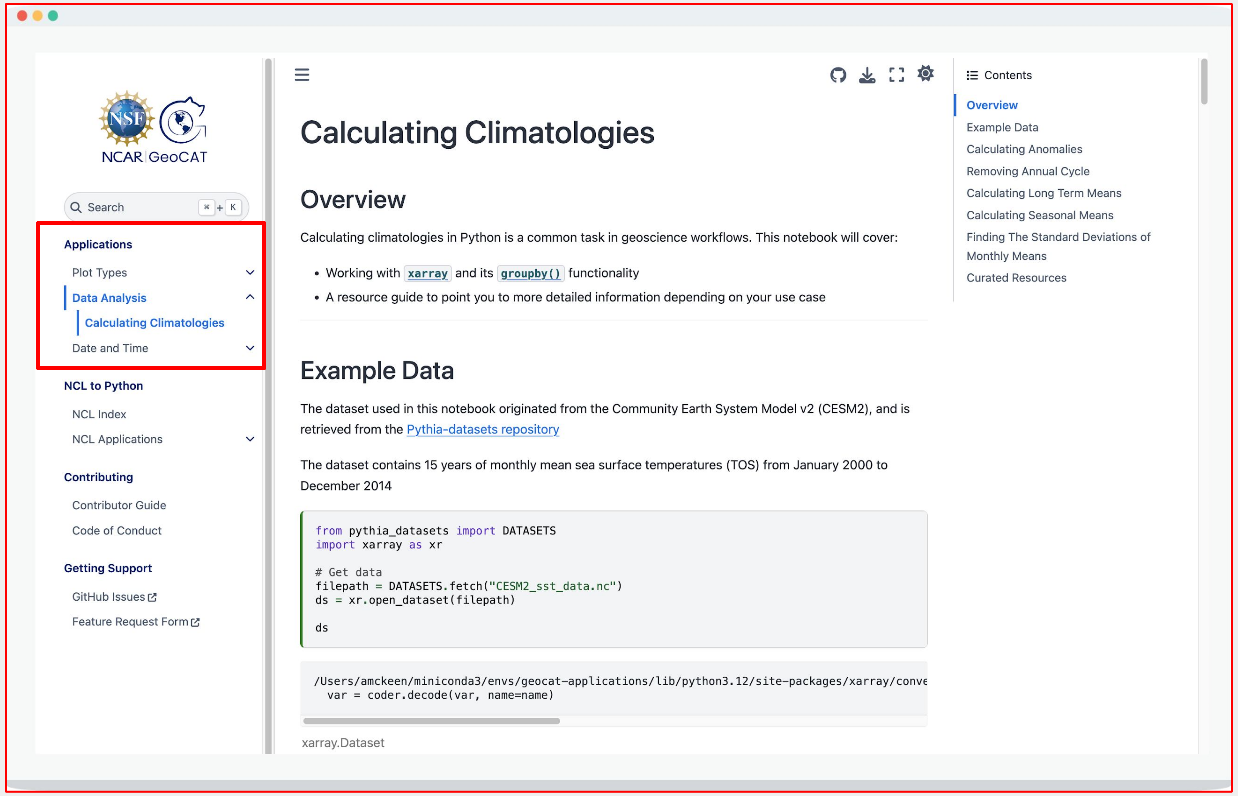


NCL to Python

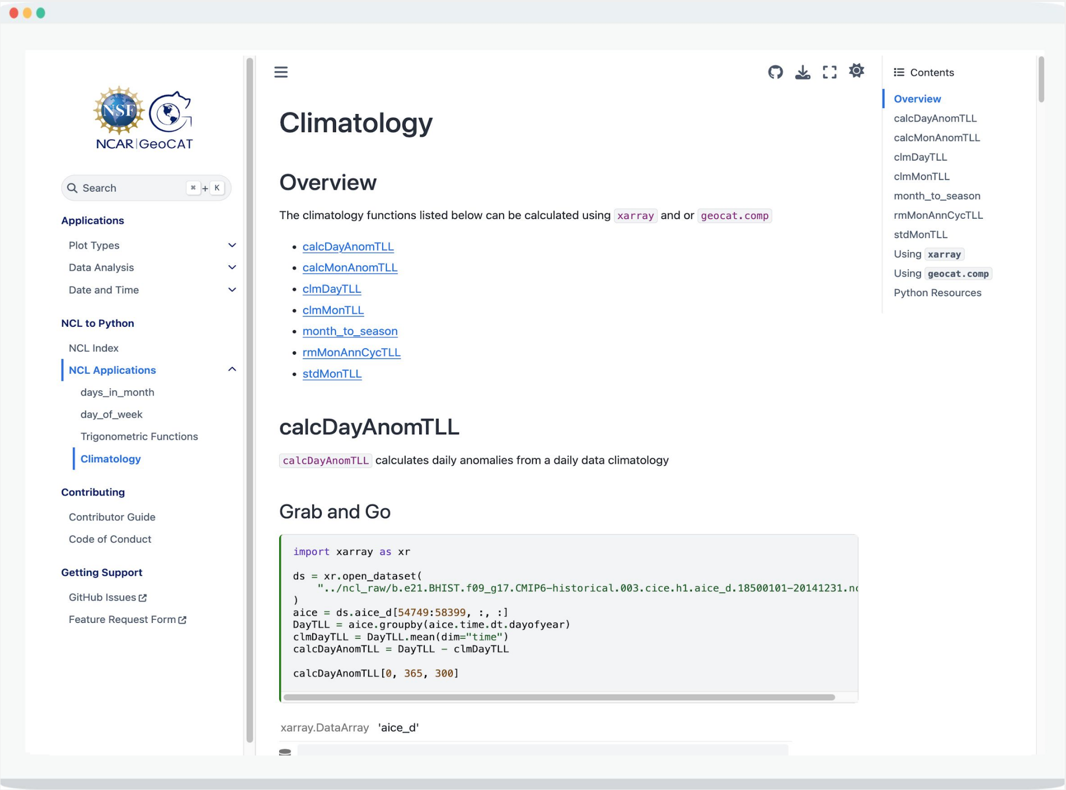


GeoCAT Applications

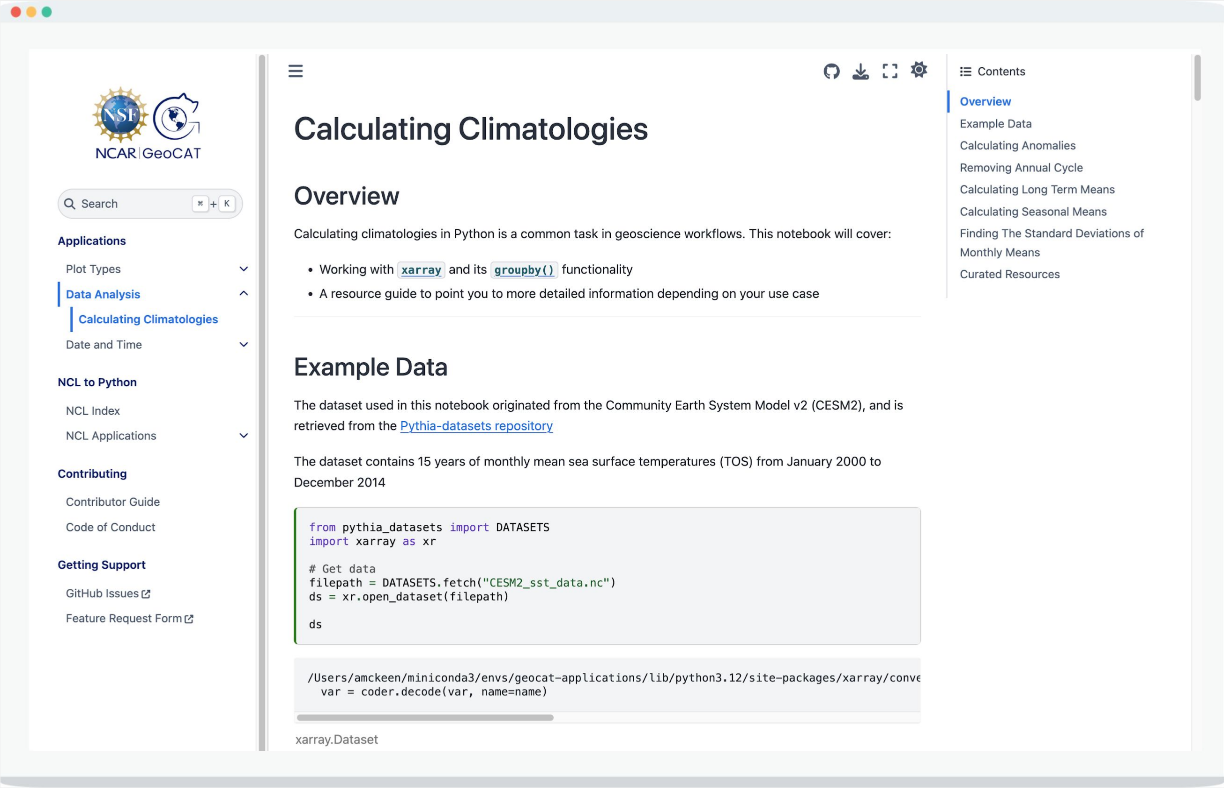
Applications



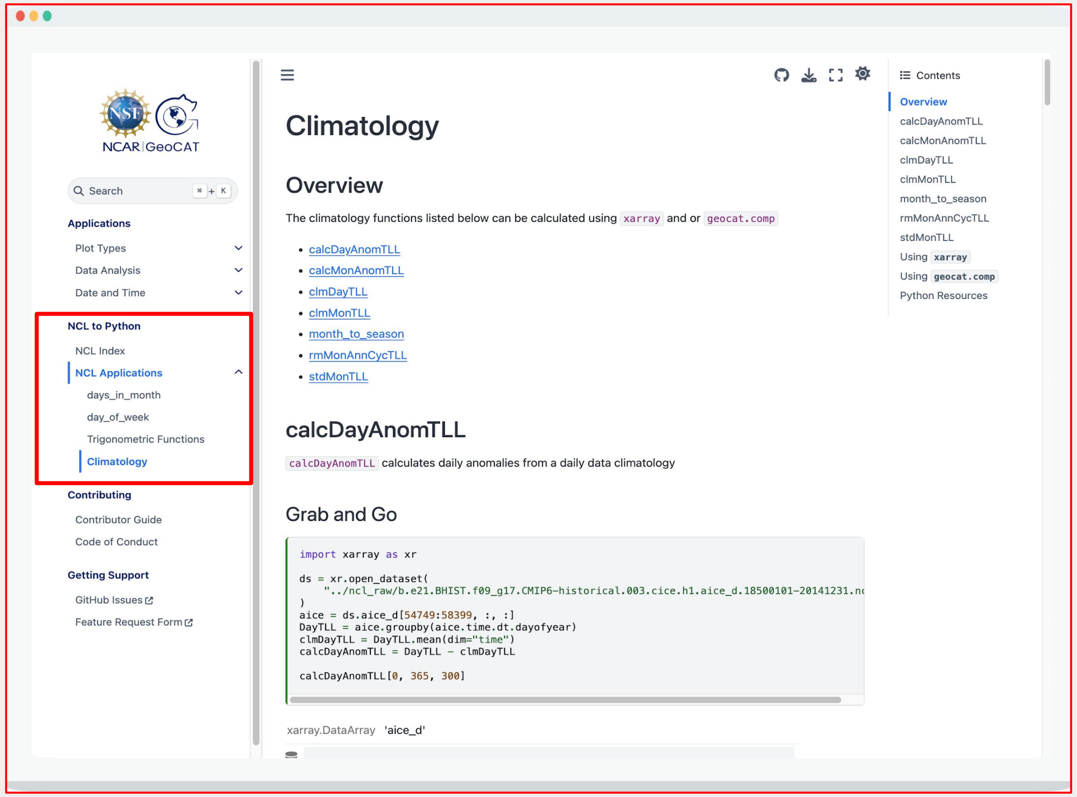
NCL to Python



Applications

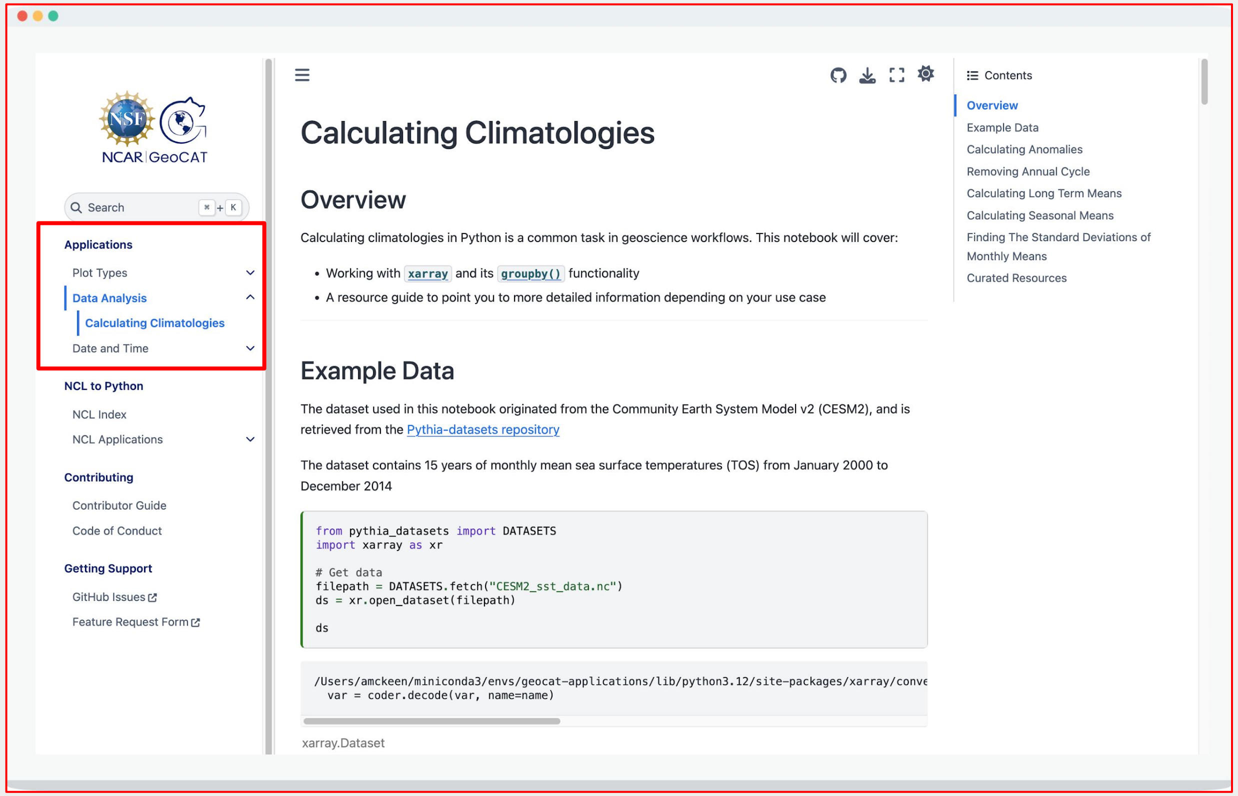


NCL to Python

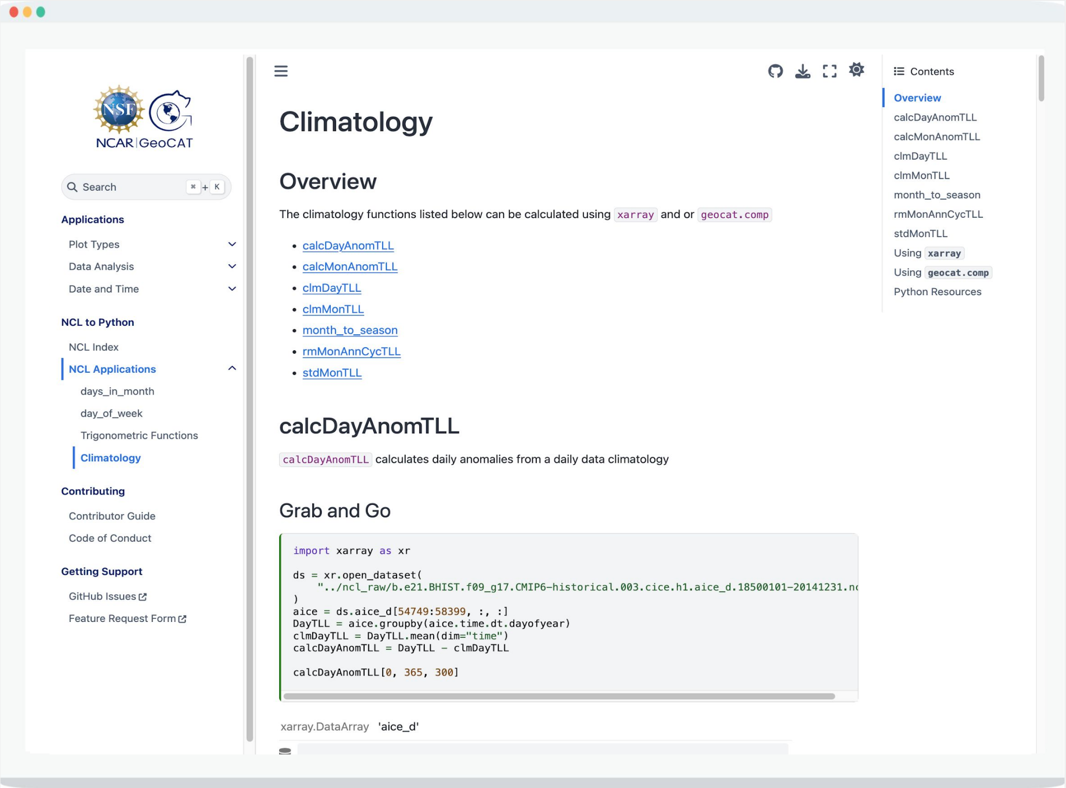


Calculating Climatologies

Applications



NCL to Python

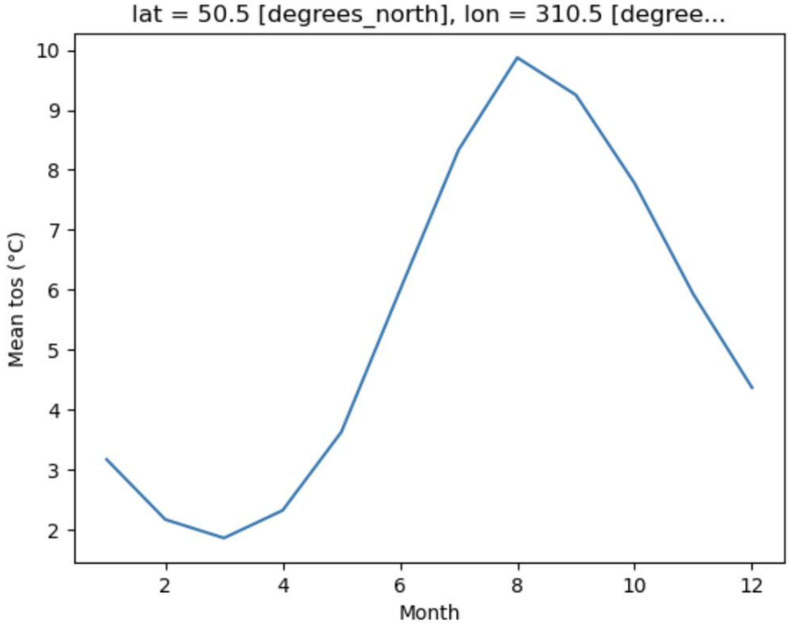


Calculating Long Term Means

```
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")
```



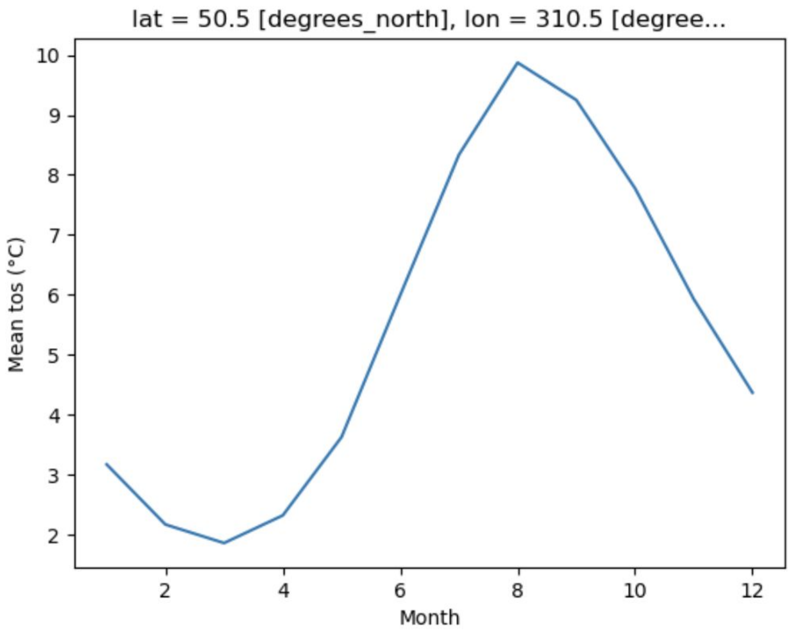
Calculating Climatologies

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")
```



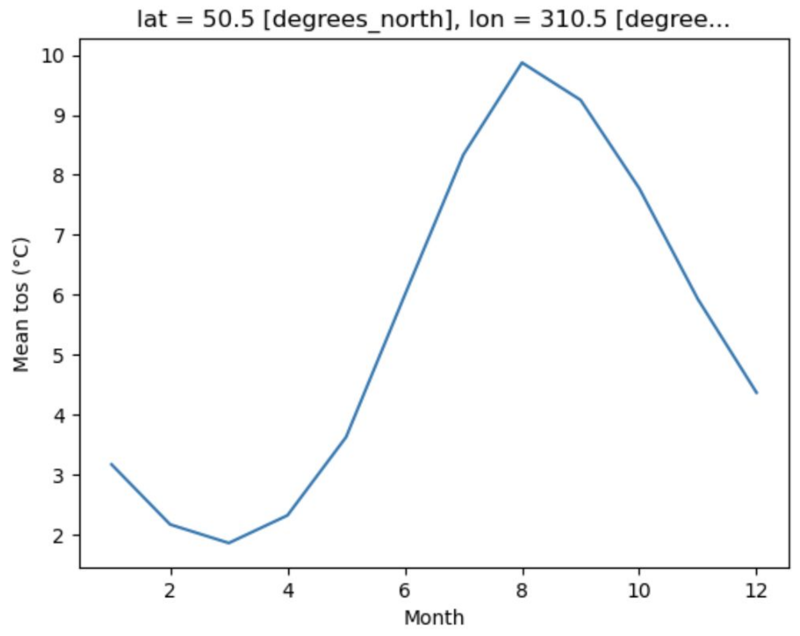
Calculating Climatologies

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")
```



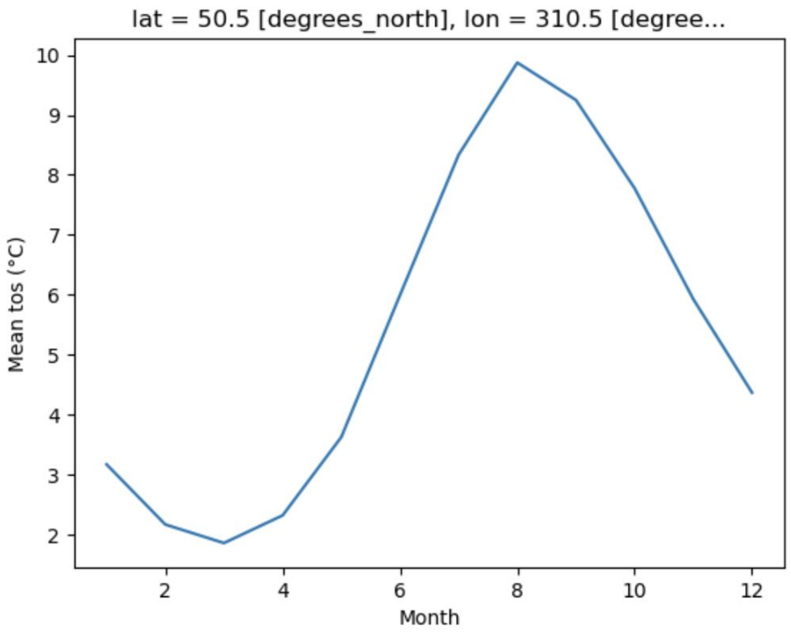
Calculating Climatologies

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")
```



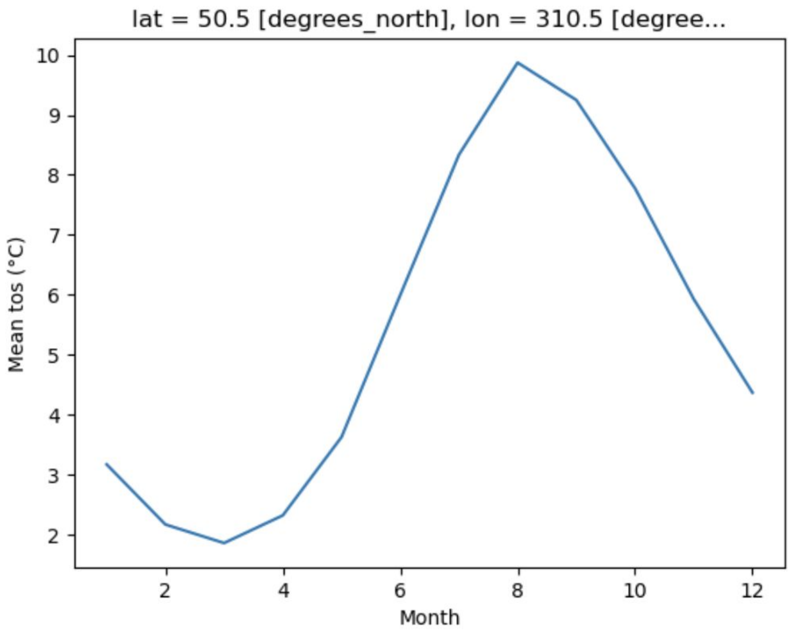
Quick Reference Materials

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

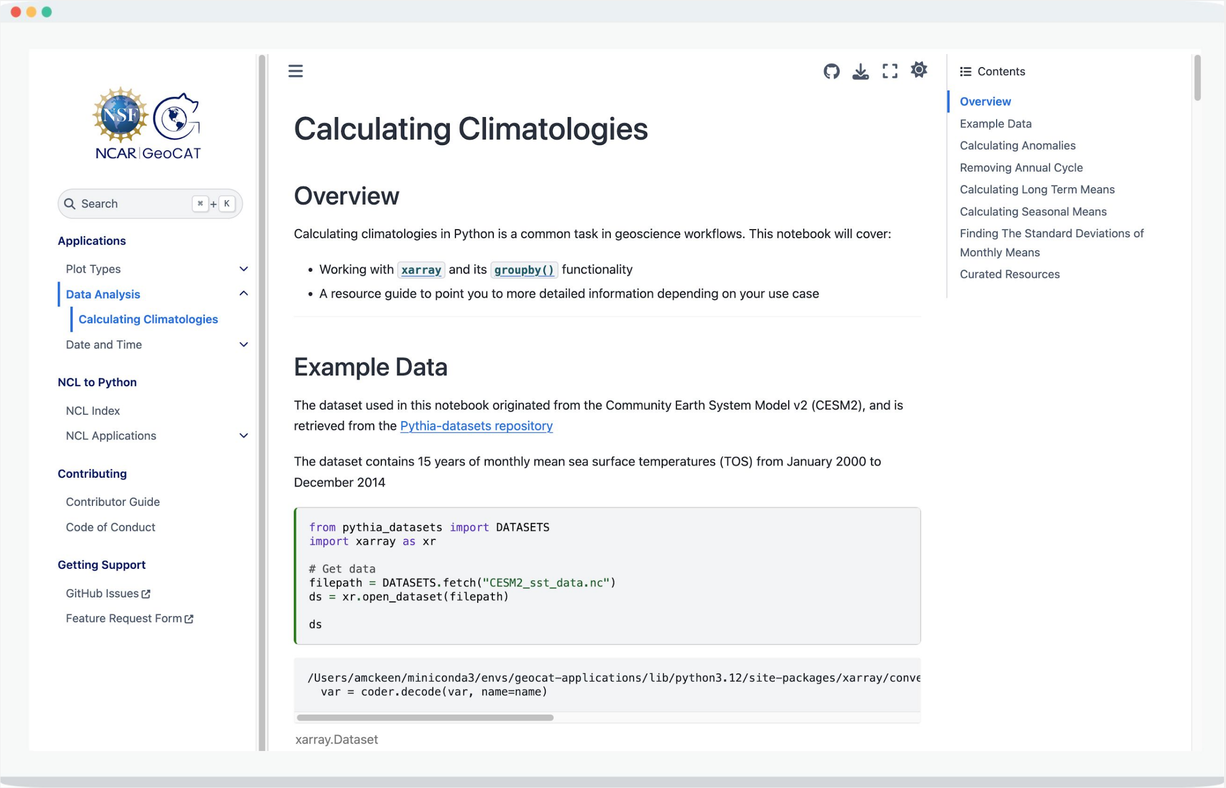
To learn more about calculating climatologies in Python, we suggest:

- 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

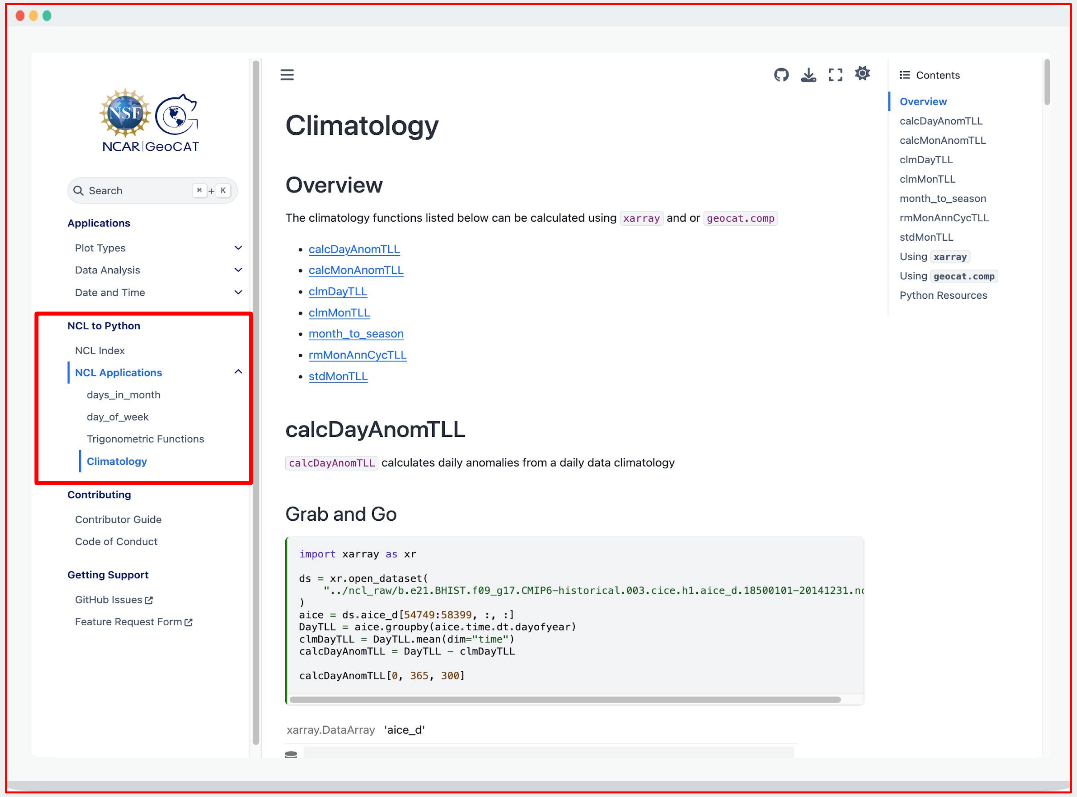
Previous
Data Analysis

Next
Dates and Times

Applications



NCL to Python



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(
    "../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))
```

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))
```

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:
        4.41653251609386 3e-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

```python
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
```

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

This material is based upon work supported by the National Center for Atmospheric Research, which is a major facility sponsored by the National Science Foundation under Cooperative Agreement No. 1852977.