nctoolkit
Cheat Sheet

Creating datasets

ds = nc.open_data(foo.nc)
Open a local file as a dataset.
ds = nc.open_url('https://foo.foo.nc')
Open/download a file as a dataset.
ds = nc.open_thredds('https://foo.foo.nc')
Use thredds/opendap file as a dataset.

Visualizing data

ds.plot()
Plot all data in a dataset.
ds.plot('var')
Plot a specific variable.

Subsetting data

**ds.subset(lon = [lon_min, lon_max],
lat = [lat_min, lat_max])**
Crop to a latlon box.
ds.subset(variables = [var1, var2])
Select a list of variables.
ds.subset(years = [2000, 2001])
Select a list of years.
ds.subset(months = [5, 6])
Select a list of years.
ds.drop(variables = ['var1', 'var2])
Remove a list of variables.

Rolling methods

Rolling methods require a window to average over.

ds.rolling_mean(20)
Calculate rolling mean using a window of 20.
ds.rolling_min(10)
Calculate rolling min using a window of 10.
ds.rolling_max(5)
Calculate rolling max using a window of 5.
ds.rolling_sum(20)
Calculate rolling sum using a window of 20.

Exporting datasets

ds.to_xarray()
Export as xarray dataset.
ds.to_dataframe()
Export as pandas dataframe.
ds.to_nc('foo/foo.nc')
Export as netCDF file.

Accessing attributes

ds.variables
List dataset variables.
ds.years
List dataset years.
ds.months
List dataset months.
ds.times
List dataset times.
ds.size
Display dataset size.
ds.current
Display dataset files.

Merging methods

ds.merge("variables")
Merge dataset of files with different variables.
ds.merge("time")
Merge dataset of files with different timesteps.

Copying dataset

ds_copy = ds.copy()
Copy a dataset.

Global settings

nc.options(lazy = False)
Set evaluation to eager/non-lazy.
nc.options(temp_dir ='/foo')
Set temporary directory to use in session.
nc.options(cores = 6)
Set number of cores to use when processing multi-file datsets
nc.options(parallel = True)
Tell nctoolkit multiple datasets will be processed in parallel

Temporal methods

Temporal averaging methods require a list, which spec ifies the time periods to average over, the elements of which must be 'year', 'month', 'day'. Defaults to 'time', i .e. an average over all time steps.

ds.tmean('year')
Calculate the annual mean.
ds.tmean(["year", "month"])
Calculate the mean for each month in each year.
ds.tmin()
Calculate the temporal minimum.
ds.tmax()
Calculate the temporal maximum.
ds.tmedian()
Calculate the temporal median.
ds.trange()
Calculate the temporal range.
ds.tpercentile(95)
Calculate the 95th percentile.
ds.tvariance()
Calculate the temporal variance.

ds.shift(hours = -1)
Shift time back 1 hour. Other valid arguments: 'days', 'months', 'years'.
ds.tcumsum()
Temporal cumulative sum.
ds.first_above(0)
Identify 1st time step variables are positive.
ds.first_below(0)
Identify 1st time step variables are negative.

Vertical methods

ds.vertical_mean(fixed = True)
Calculate vertical mean per grid-cell.
ds.vertical_min()
Calculate vertical minimum per grid-cell.
ds.vertical_max()
Calculate vertical maximum per grid-cell.
ds.top()
Extract the top-cell, e.g. the sea-surface.
ds.vertical_interp([10, 20,30], fixed = True)
Interpolate to a list of vertical depths.

Ensemble methods

Ensemble methods allow the comparison of files with t he same timesteps and grid. Calculations are done per -grid-cell.

ds.ensemble_mean()
Calculate mean across an ensemble.
ds.ensemble_max()
Calculate maximum across an ensemble.
ds.ensemble_min()
Calculate minimum across an ensemble.
ds.ensemble_range()
Calculate range across an ensemble.

Spatial methods

Spatial methods are calculated per time-step

ds.spatial_mean()
Calculate the spatial mean.
ds.spatial_min()
Calculate the spatial minimum.
ds.spatial_max()
Calculate the spatial maximum.
ds.spatial_sum()
Calculate the spatial sum.
ds.zonal_mean()
Calculate the zonal mean.
ds.meridonial_mean()
Calculate the meridonial mean.

Random hacks

ds.zip()
Zip dataset files.
ds.format('nc4')
Change netCDF format of dataset files.
ds.as_missing([0, 100])
Set values within a range to missing.
ds.rename({'old_foo':'new_foo')
Change the name of a variable.
ds.set_units({'var':'foo/s')
Set the units for a variable.
ds.set_longnames({'foo':'a long foo')
Set the long names for variables.

Creating variables

New variables can be created using the assign method. This requires a lambda function. Operations are carried out per-grid-cell and timestep.

ds.assign(new = lambda x: x.old + 10)
Calculate a new variable, which is just an old one plus 10.
ds.assign(new = lambda x: x.old > spatial_mean(x.old))
Create a variable which identifies if a grid cell is higher than the spatial mean.
For more examples see the nctoolkit website.

Multi-dataset methods

Multi-dataset methods let you add/subtract dataset fro m others so long as their grids and timesteps are com patible. Calculations carried out per-timestep and grid cell

ds + ds1
Add one dataset to another.
ds – ds1
Subtract one dataset from another.
ds * ds1
Multiply a dataset by another.
ds / ds1
Divide a dataset by another.
ds > ds1
Do a dataset's values exceed another's?
ds < ds1
Are a dataset's values less than another's?

Regridding

ds.regrid('foo.nc')
Regrid to a file's grid.
ds.regrid(ds2)
Regrid to another dataset's grid.
ds.to_latlon(lon = [lon_min, lon_max],
lat = [lat_min, lat_max],
res = [lon_res, lat_res])
Regrid to a regular latlon grid, with specified latlon ranges and resolutions.
ds.resample_grid(2)
Resample, selecting everything other lon/lat grid cell