NCToolkit Cheat Sheet

A quick two-page overview of NCToolkit's most-used methods.

v1.3.11

PML Plymouth Marine Laboratory

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 a thredds/OPeNDAP file as a dataset.

SUBSETTING DATA

```
ds.subset(lon=[lon_min, lon_max],
          lat=[lat_min, lat_max])
```
Crop to a lon/lat 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 months.

```
ds.drop(variables=["var1", "var2"])
```
Remove a list of variables.

VISUALIZING DATA

```
ds.plot()
```
Plot all data in a dataset.

```
ds.pub_plot()
```
Create a publication-quality static map.

MERGING METHODS

```
ds.merge("variable")
```
Merge a dataset of files with different variables.

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

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 an xarray dataset.

```
ds.to_dataframe()
```
Export as a pandas dataframe.

```
ds.to_nc("foo/foo.nc")
```
Export as a 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.

TEMPORAL METHODS

Temporal averaging methods take a list specifying the time periods to average over — elements must be “year”, “month” or “day”. Defaults to 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. Also takes days, months, years.

```
ds.tcumsum()
```
Temporal cumulative sum.

```
ds.first_above(0)
```
Identify the 1st time step where values are positive.

```
ds.first_below(0)
```
Identify the 1st time step where values are negative.

COPYING A DATASET

```
ds_copy = ds.copy()
```
Copy a dataset.

NCToolkit Cheat Sheet

Continued — vertical, spatial, ensemble and multi-dataset methods.

v1.3.11

PML Plymouth Marine Laboratory

VERTICAL METHODS

`ds.vertical_mean()`
Calculate the vertical mean per grid cell.

`ds.vertical_min()`
Calculate the vertical minimum per grid cell.

`ds.vertical_max()`
Calculate the vertical maximum per grid cell.

`ds.top()`
Extract the top cell, e.g. the sea surface.

`ds.bottom()`
Extract the bottom cell.

`ds.vertical_interp([10, 20, 30])`
Interpolate to a list of vertical depths.

ENSEMBLE METHODS

Ensemble methods compare files with the same timesteps and grid. Calculations are done per grid cell.

`ds.ensemble_mean()`
Calculate the mean across an ensemble.

`ds.ensemble_max()`
Calculate the maximum across an ensemble.

`ds.ensemble_min()`
Calculate the minimum across an ensemble.

`ds.ensemble_range()`
Calculate the range across an ensemble.

GLOBAL SETTINGS

`nc.options(lazy=False)`
Set evaluation to eager/non-lazy.

`nc.options(temp_dir="/foo")`
Set the temporary directory to use in this session.

`nc.options(cores=6)`
Set the number of cores to use when processing multi-file datasets.

`nc.options(parallel=True)`
Tell NCToolkit multiple datasets will be processed in parallel.

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.

```
ds.to_transect(start=[lon1, lat1],
               end=[lon2, lat2], nsteps=100)
```
Extract a straight-line transect between two points.

MULTI-DATASET METHODS

Add, subtract or compare one dataset with another, so long as their grids and timesteps are compatible. Calculations are 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?

RANDOM HACKS

`ds.zip()`
Zip dataset files.

`ds.format("nc4")`
Change the 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 name for a variable.

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 identifying whether a grid cell exceeds the spatial mean.

For more examples, see the NCToolkit website.

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 lon/lat grid with a specified extent and resolution.

`ds.resample_grid(2)`
Resample, selecting every other lon/lat grid cell.