

# for\_wintertime\_users

February 22, 2024

## 1 NetCDF Tutorial

### Maxime Benoît-Gagné

This tutorial was given at the meeting of the Numerical Ecology in Oceanography Laboratory (NEOLab), Université Laval, on April 21, 2023.

This Jupyter notebook is part of the project for\_wintertime\_users containing documentation and examples on how to use the outputs of the default simulation EXP-0 described in Benoît-Gagné et al. (submitted).

The documentation and examples are available on:

[https://github.com/maximebenoitgagne/for\\_wintertime\\_users/tree/main](https://github.com/maximebenoitgagne/for_wintertime_users/tree/main).

The original project of Benoît-Gagné et al. (submitted) can be found on:

<https://github.com/maximebenoitgagne/wintertime/tree/v1.5>.

## 2 Definitions

**NetCDF file for a region:** A NetCDF file for many latitudes and many longitudes, for example, a NetCDF file for the whole Baffin Bay.

**NetCDF file for a specific location:** A NetCDF file for only one latitude and one longitude, for example, a NetCDF file for the location of an ice camp.

**Results of Benoît-Gagné et al. (submitted):** NetCDF files containing the results of the default simulation of Benoît-Gagné et al. (submitted). The simulation was adapted to the location of the Qikiqtarjuaq sea ice camp in western Baffin Bay (67.4797°N, -63.7895°E) in 2016 during the Green Edge sea ice camp mission. The model was run with a spinup of 10 years to stabilise the system. Use the data of that tenth year. The dimensions of the data are the depth steps and the time steps.

## 3 What you will learn in this tutorial

- Find the results of Benoît-Gagné et al. (submitted).
- Understand the structure of the results of Benoît-Gagné et al. (submitted).
- Explore the structure of a NetCDF file for a specific location with Panoply.
- Plot a NetCDF file for a specific location with Panoply.

- Read a NetCDF file for a specific location with Python.\*
- Export a NetCDF file for a specific location into a comma-separated values (CSV) file with Python.\*
- Plot a variable from the results of Benoît-Gagné et al. (submitted) with Python.\*
- Find more examples of Python scripts reading the results of Benoît-Gagné et al. (submitted).\*

Note: The goals with an asterisk (\*) have the following prerequisites:

- Knowledge of Anaconda.
- Basic knowledge of Python.

## 4 What you will not learn in this tutorial

- Anaconda.
- Python.
- Use a NetCDF file for a specific location when the metadata doesn't contain sufficient information to retrieve the depth steps and the time steps of the file. Actually, the only solution would be to ask someone who knows the answer, for example, the human author of the files.
- Export a NetCDF file for a specific location into a CSV file with Panoply.
- Write a NetCDF file.
- Read a NetCDF file for a specific location in R.
- Use a NetCDF file for a region.

## 5 Installation steps

These installation steps were tested for MacOS 11 (Big Sur).

- Install Panoply (<https://www.giss.nasa.gov/tools/panoply/download/>).
- Install Anaconda for Python 3 as described on <https://datacarpentry.org/python-ecology-lesson/setup.html>:
  - Go to <https://www.anaconda.com/products/distribution>.
  - Click Download.
  - Double-click the graphical installer .pkg file.
  - Follow the instructions. Select the default settings.
  - Verify the installation by entering the following command in the Terminal (Launchpad icon in the Dock, type Terminal, click Terminal):

```
conda --version
```

If your version of conda is < 4.4:

```
conda update conda
```

- Enter the following commands in the Terminal. The creation of the conda environment can take approximately 15 minutes.

```
git clone https://github.com/maximebenoitgagne/for_wintertime_users.git
cd for_wintertime_users/
```

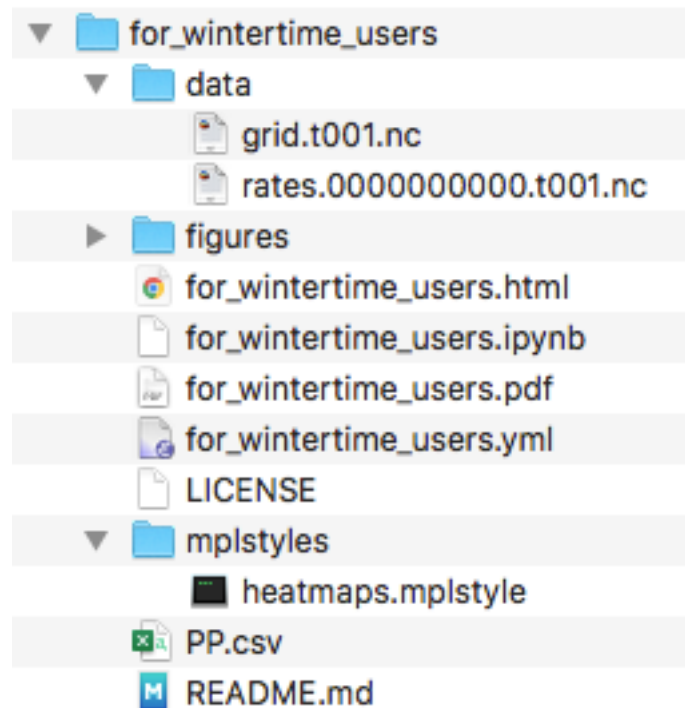
```
conda env create -f for_wintertime_users.yml
conda activate for_wintertime_users
jupyter notebook
```

Once the environment `for_wintertime_users` is created, the Jupyter notebook can be launched simply with:

```
cd for_wintertime_users/
conda activate for_wintertime_users
jupyter notebook
```

## 6 Find the data

The results of Benoît-Gagné et al. (submitted) are available on [https://github.com/maximebenoitgagne/wintertime/tree/v1.5/data/DataS8\\_output\\_mitgcm/exp0](https://github.com/maximebenoitgagne/wintertime/tree/v1.5/data/DataS8_output_mitgcm/exp0). Two relatively small files of these results are also in the directory `data` of this project.



## 7 Read a NetCDF file with Panoply

Panoply is a graphical user interface (GUI) tool to explore and plot NetCDF files.

Steps

- Click on Panoply.
- Select `grid.t001.nc`.

Panoply — Sources

Create Plot Combine Plot Open Dataset

Datasets Catalogs Bookmarks

Name	Long Name	Type
grid.t001.nc	grid.t001.nc	Local File
Depth	Depth	—
drC	drC	1D
drF	drF	1D
dxC	dxC	1D
dxF	dxF	—
dxG	dxG	1D
dxV	dxV	Geo2D
dyC	dyC	1D
dyF	dyF	—
dyG	dyG	1D
dyU	dyU	Geo2D
fCori	fCori	—
fCoriG	fCoriG	Geo2D
HFacC	HFacC	Geo2D
HFacS	HFacS	Geo2D
HFacW	HFacW	Geo2D
R_low	R_low	—
rA	rA	—
rAs	rAs	1D
rAw	rAw	1D
rAz	rAz	Geo2D
RC	RC	1D
RF	RF	1D
RL	RL	1D
Ro_surf	Ro_surf	—
RU	RU	1D
X	longitude of cell center	—
XC	XC	—
XG	XG	Geo2D
Xp1	longitude of cell corner	1D
Y	latitude of cell center	—
YC	YC	—
YG	YG	Geo2D
Yp1	latitude of cell corner	1D
Z	vertical coordinate of cell center	1D
Zl	vertical coordinate of upper cell interf...	1D
Zp1	vertical coordinate of cell interface	1D
Zu	vertical coordinate of lower cell interf...	1D

Show: All variables

**File "grid.t001.nc"**

File type: NetCDF-3/CDM

netcdf file: /Users/maximebenoit-gagne/My%20Drive/ABC/Cours

dimensions:

```

Z = 75;
Zp1 = 76;
Zu = 75;
Zl = 75;
X = 1;
Y = 1;
Xp1 = 2;
Yp1 = 2;

```

variables:

```

double RC(Z=75);
:description = "R coordinate of cell center";
:units = "m";

double RF(Zp1=76);
:description = "R coordinate of cell interface";
:units = "m";

double RU(Zu=75);
:description = "R coordinate of upper interface";
:units = "m";

double RL(Zl=75);
:description = "R coordinate of lower interface";
:units = "m";

double drC(Zp1=76);
:description = "r cell center separation";

double drF(Z=75);
:description = "r cell face separation";

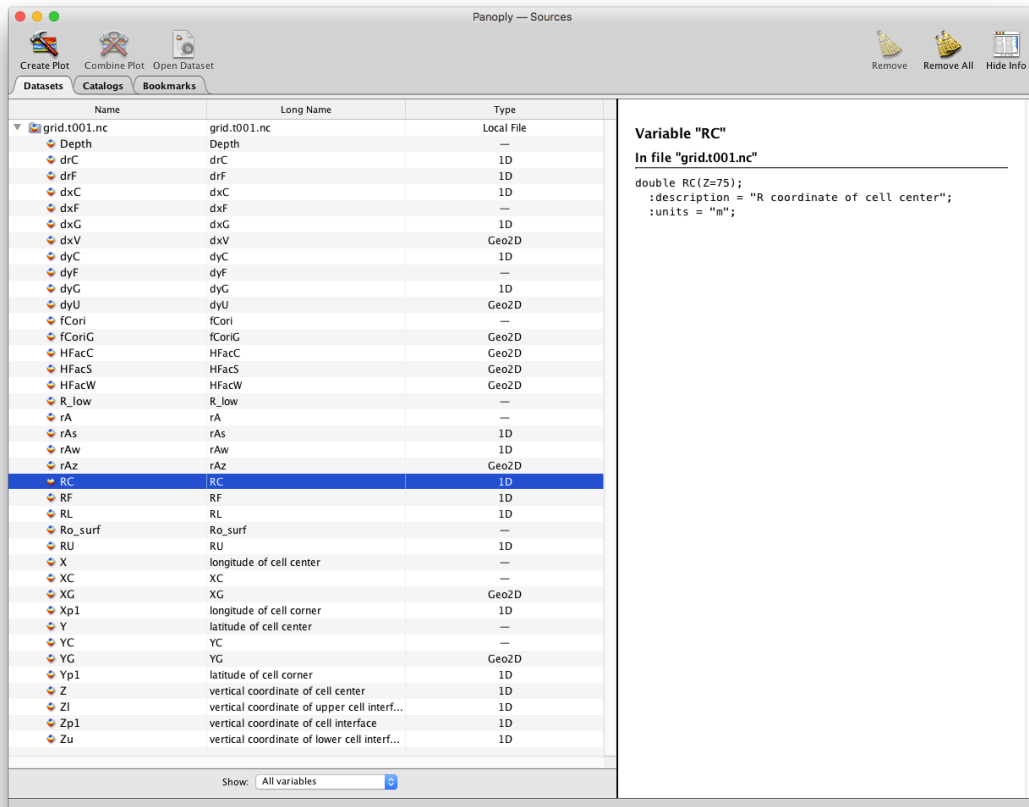
double XG(Yp1=2, Xp1=2);
:description = "X coordinate of cell corner (Vortici";
:units = "degree_east";

double YG(Yp1=2, Xp1=2);
:description = "Y coordinate of cell corner (Vortici";
:units = "degree_north";

double dxC(Y=1, Xp1=2);

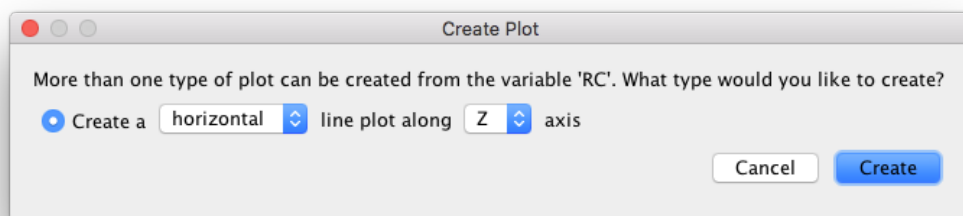
```

- Select the variable RC.



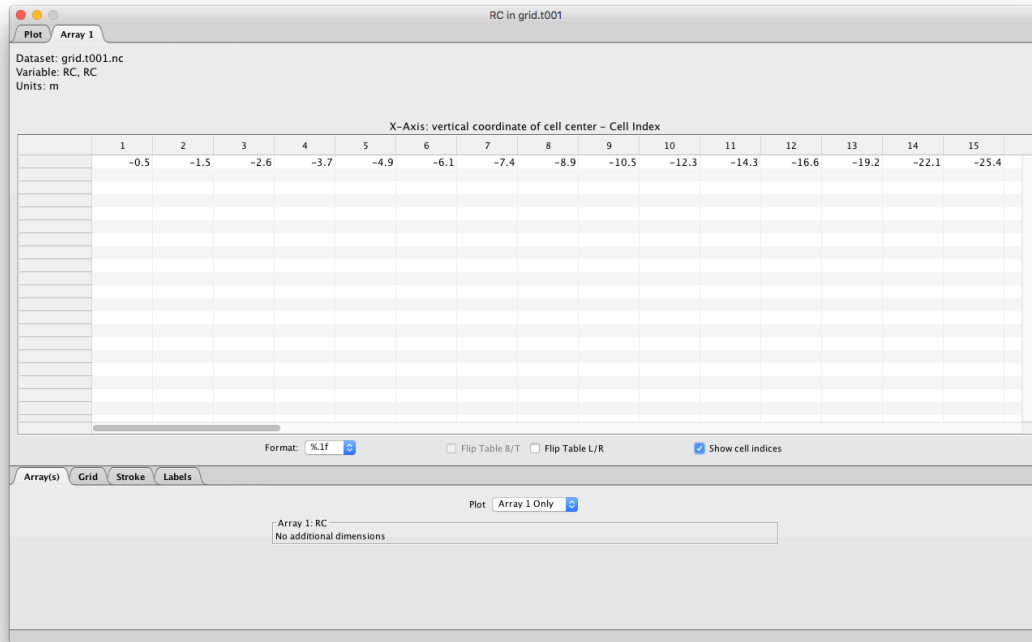
We see that the variable RC in grid.t001.nc contains the values of the 75 depth steps.

- Double-click on RC.

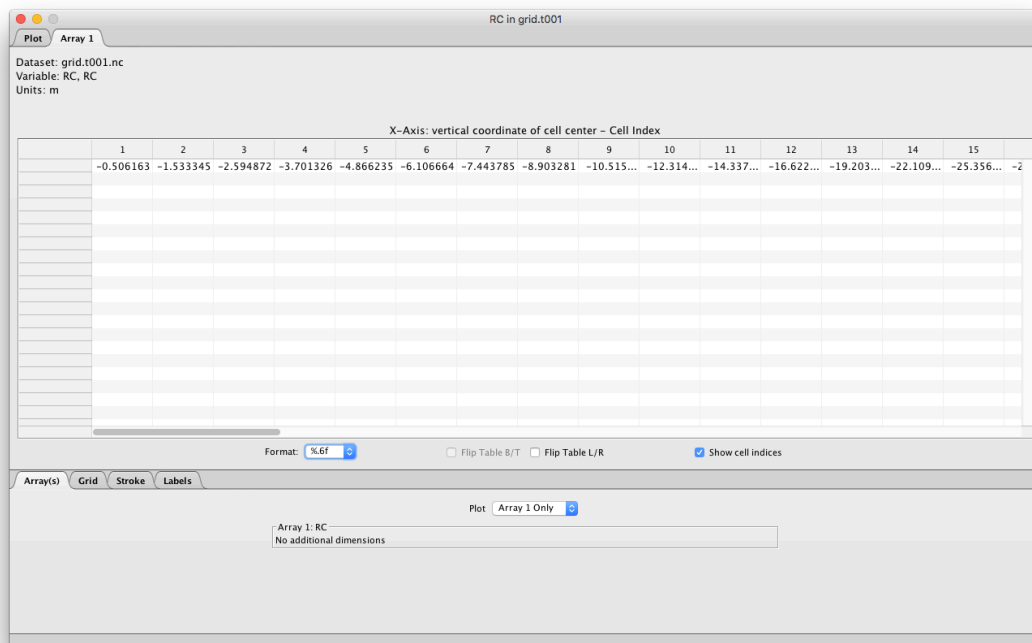


- Click Create.





- Select Format `%.6f`.



We now see explicitly the values of the 75 depth steps. Note that the vertical grid is irregular. We have to take that into account if we want to transform units by  $\text{m}^{-3}$  into units by  $\text{m}^{-2}$ .

We will now open another NetCDF file. We will open rates.0000000000.t001.nc.

- File > Open > rates.0000000000.t001.nc.

The screenshot shows the Panoply Sources window. On the left, a table lists various datasets. The dataset 'rates.0000000000.t001.nc' is selected and highlighted in blue. On the right, the NetCDF file metadata is displayed.

Name	Long Name	Type
dxG	dxG	1D
dxV	dxV	Geo2D
dyC	dyC	1D
dyF	dyF	—
dyG	dyG	1D
dyU	dyU	Geo2D
fCori	fCori	—
fCoriG	fCoriG	Geo2D
HFacC	HFacC	Geo2D
HFacS	HFacS	Geo2D
HFacW	HFacW	Geo2D
R_low	R_low	—
rA	rA	—
rAs	rAs	1D
rAw	rAw	1D
rAz	rAz	Geo2D
RC	RC	1D
RF	RF	1D
RL	RL	1D
Ro_surf	Ro_surf	—
RU	RU	1D
X	longitude of cell center	—
XC	XC	—
XG	XG	Geo2D
Xp1	longitude of cell corner	1D
Y	latitude of cell center	—
YC	YC	—
YG	YG	Geo2D
Yp1	latitude of cell corner	1D
Z	vertical coordinate of cell center	1D
Zl	vertical coordinate of upper cell interf...	1D
Zp1	vertical coordinate of cell interface	1D
Zu	vertical coordinate of lower cell interf...	1D
rates.0000000000.t001.nc	rates.0000000000.t001.nc	Local File
Denit	Denit	Geo2D
diag_levels	diag_levels	1D
iter	iteration count	1D
Nfix	Nfix	Geo2D
PP	PP	Geo2D
T	model time	1D
X	longitude of cell center	—
Y	latitude of cell center	—

File "rates.0000000000.t001.nc"

File type: NetCDF-3/CDM

```
netcdf file:/Users/maximebenoit-gagne/My%20Drive/ABC/Cours
dimensions:
  T = UNLIMITED; // (3650 currently)
  Zmd000075 = 75;
  X = 1;
  Y = 1;
variables:
  int iter(T=3650);
    :long_name = "iteration_count";

  double diag_levels(Zmd000075=75);
    :description = "Indices of vertical levels within th

  double PP(T=3650, Zmd000075=75, Y=1, X=1);
    :description = "Primary Production";
    :units = "mmol C/m^3/s";

  double Nfix(T=3650, Zmd000075=75, Y=1, X=1);
    :description = "N fixation";
    :units = "mmol N/m^3/s";

  double Denit(T=3650, Zmd000075=75, Y=1, X=1);
    :description = "Denitrification";
    :units = "mmol N/m^3/s";

  double T(T=3650);
    :long_name = "model_time";
    :units = "seconds since 2016-01-01 00:00:00";
    :CoordinateAxisType = "Time";

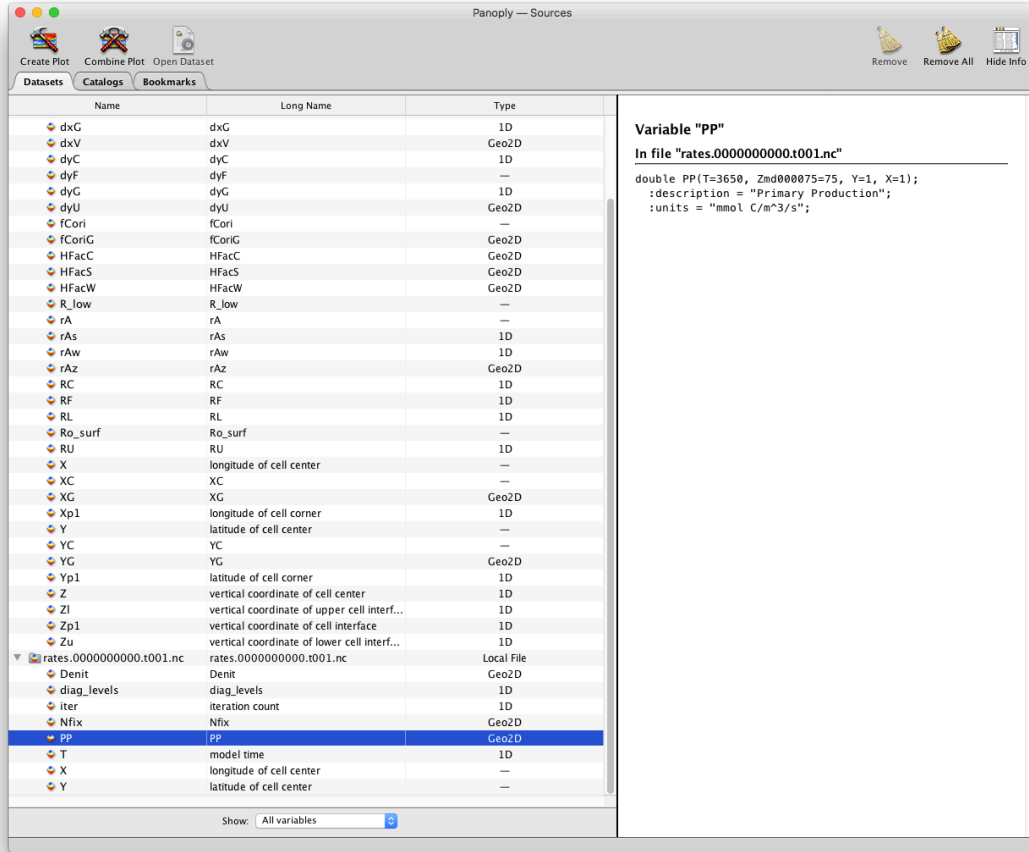
  double X(X=1);
    :long_name = "longitude of cell center";
    :units = "degrees_east";
    :CoordinateAxisType = "Lon";

  double Y(Y=1);
    :long_name = "latitude of cell center";
    :units = "degrees_north";
    :CoordinateAxisType = "Lat";

// global attributes:
:MITgcm_version = "checkpoint67d";
:build_user = "benoitga";
:build_host = "gra-login1";
:build_date = "Fri Mar 25 14:34:10 EDT 2022";
```

- Select the variable PP.

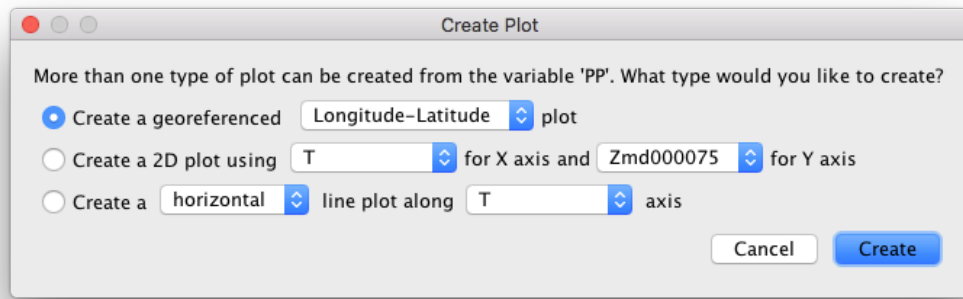




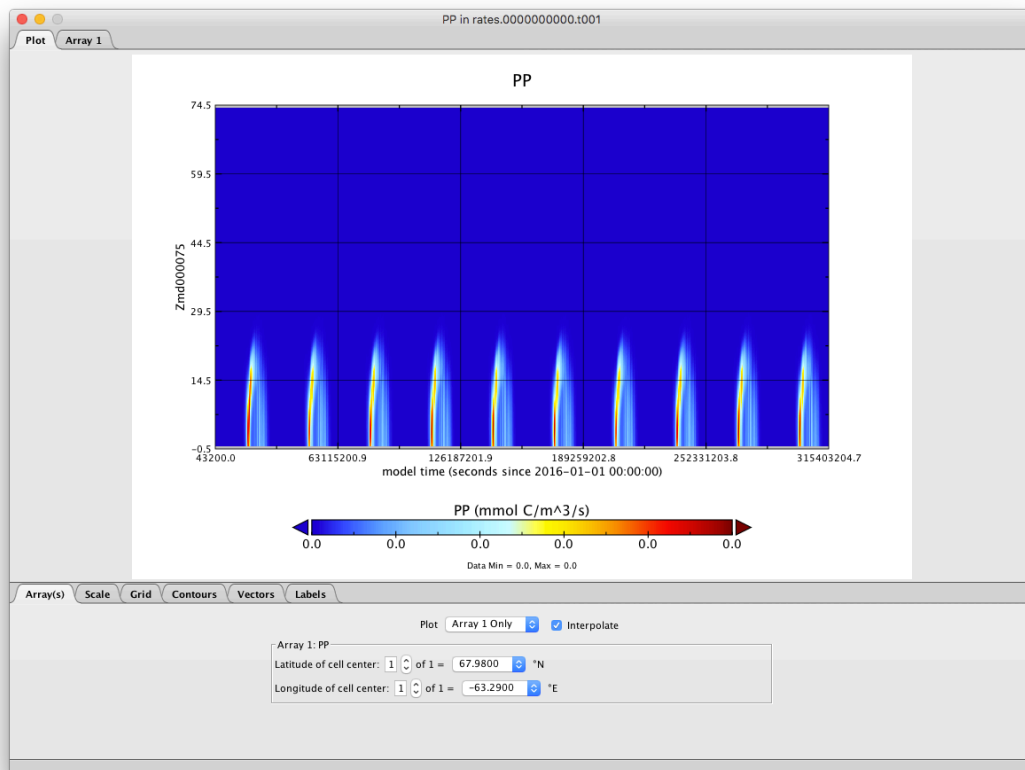
We see that the variable PP is the primary production in  $\text{mmol C m}^{-3} \text{ s}^{-1}$ . The dimension T is the daily time steps over 10 years ( $10 \times 365 = 3650$ ). The dimension Zmd000075 is the 75 depth steps.

## 8 Plot a NetCDF file with Panoply

- Double-click on PP.



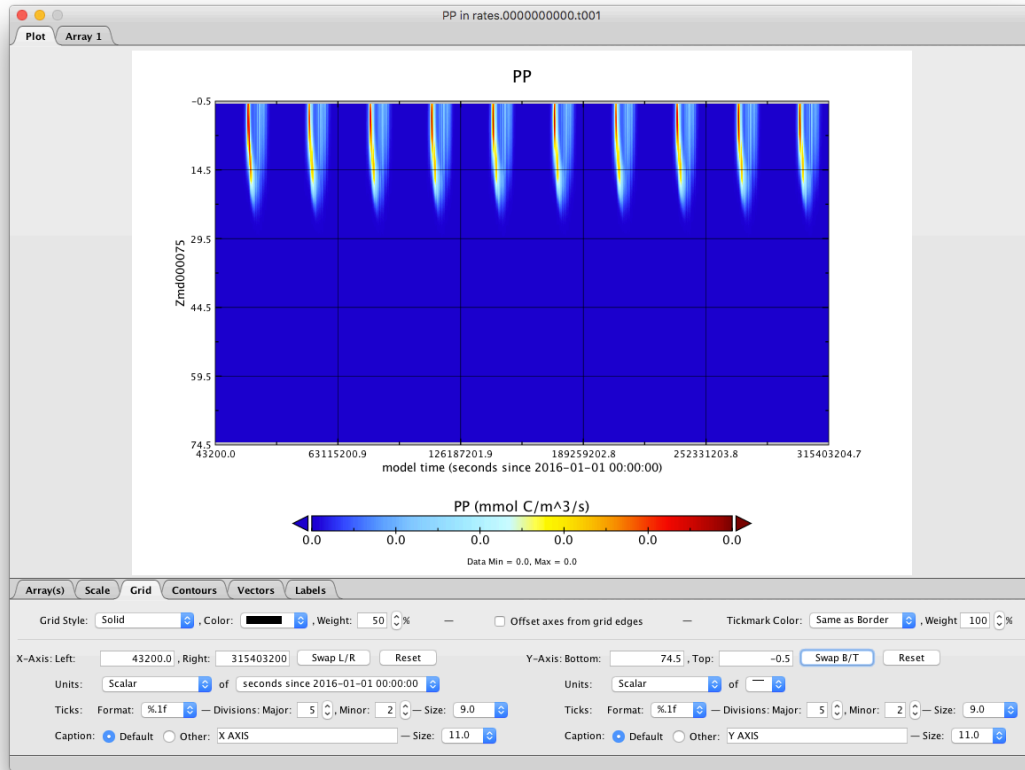
- Check Create a 2D plot using T for X axis and Zmd000075 for Y axis.
- Click Create.



Note that the Y axis contains the indices of the depths and not the values of the depths. The values of the depths are not in the file rates.0000000000.nc. They are only in the variable RC of grid.t001.nc.

Note also that the depths on the Y axis are in the reverse order. This is because the first depth index is at the water surface and is shown at the bottom of the plot. We would rather show it at the top of the plot.

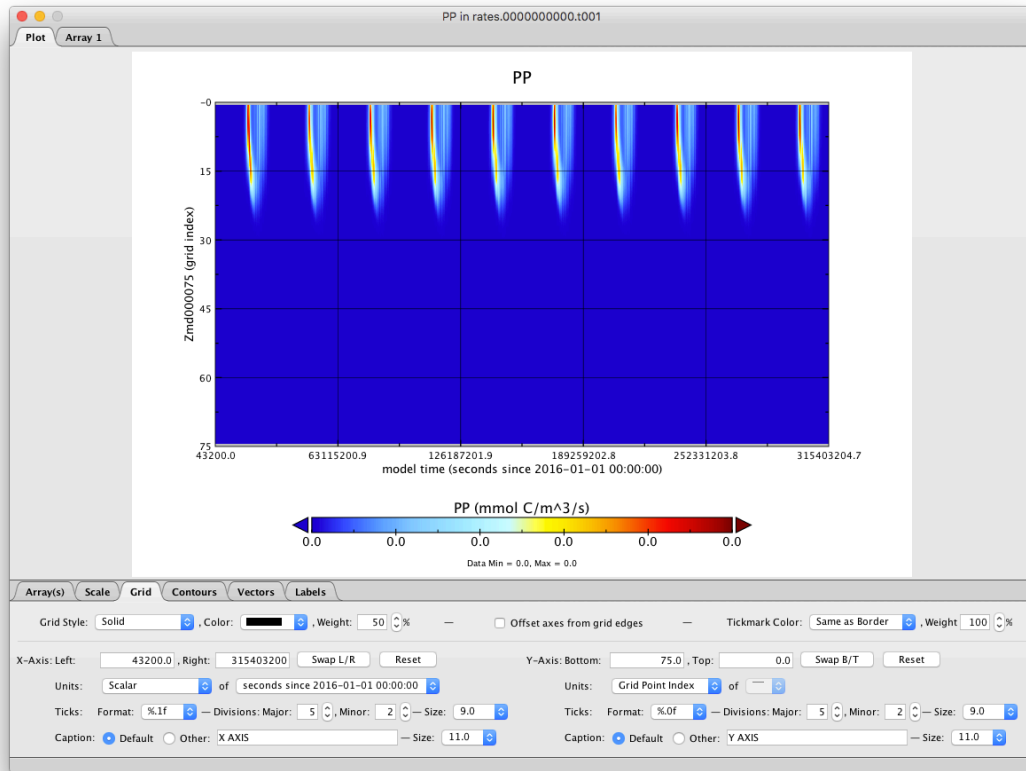
- Grid > Swap B/T.



We now see the primary productivity over 10 years. We can see the 10 yearly spring blooms. It worths noting again that the Y axis contains the indices of the depths from 0 to 74 and not the values of the depths.

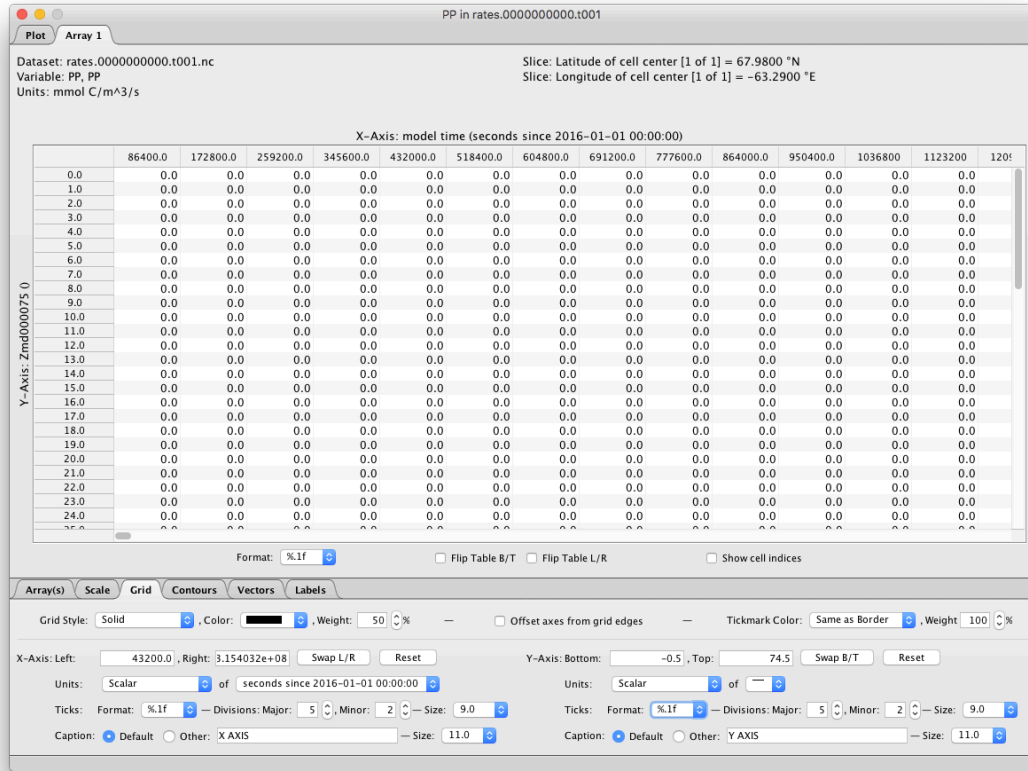
To make it clearer, we will show the indices of the depths from 1 to 75 instead of 0 to 74. In order to do that:

- Y-Axis: Units: Grid Point Index.
- Y-Axis: Ticks: Format `%0f`.
- Swap B/T.

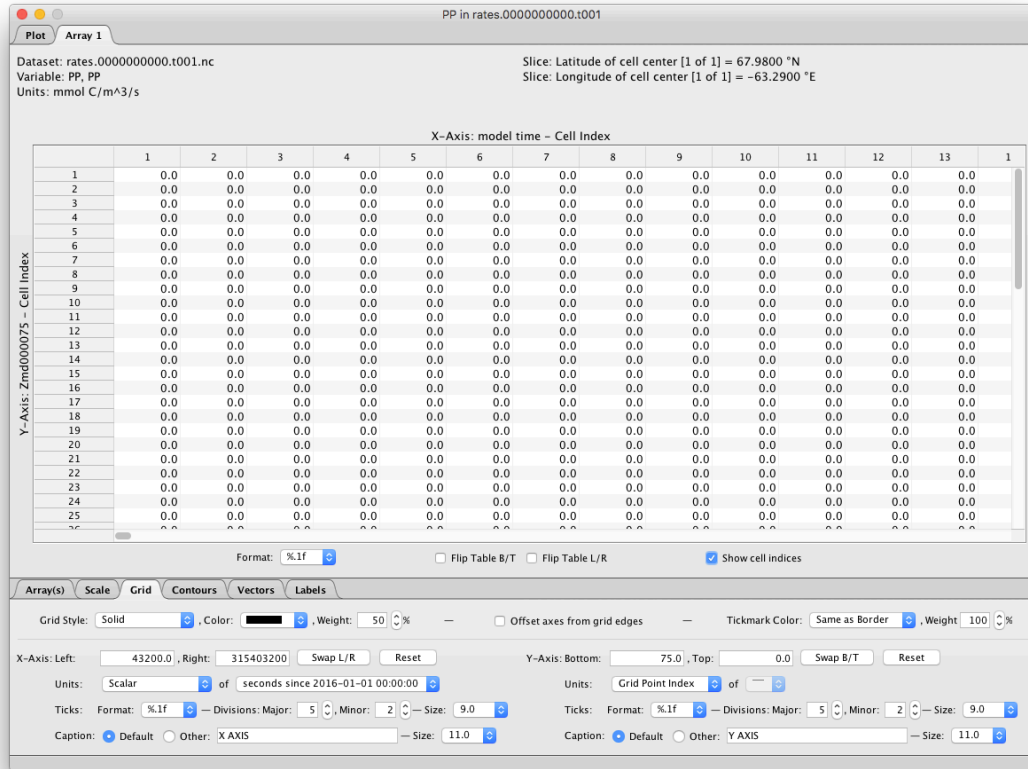


Verify.

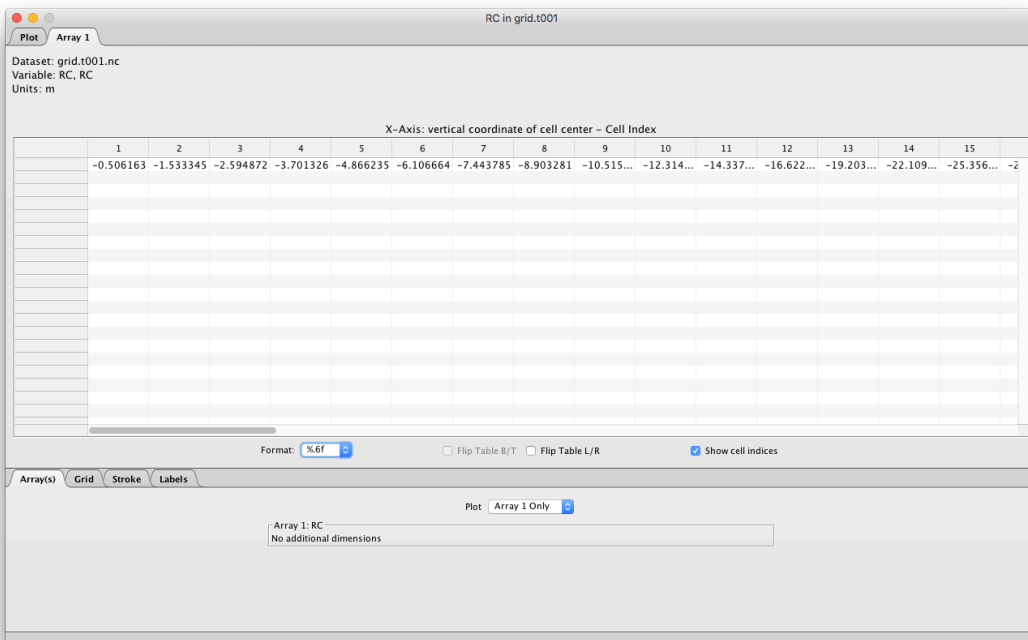
- Select tab Array 1.



- Show cell indices.

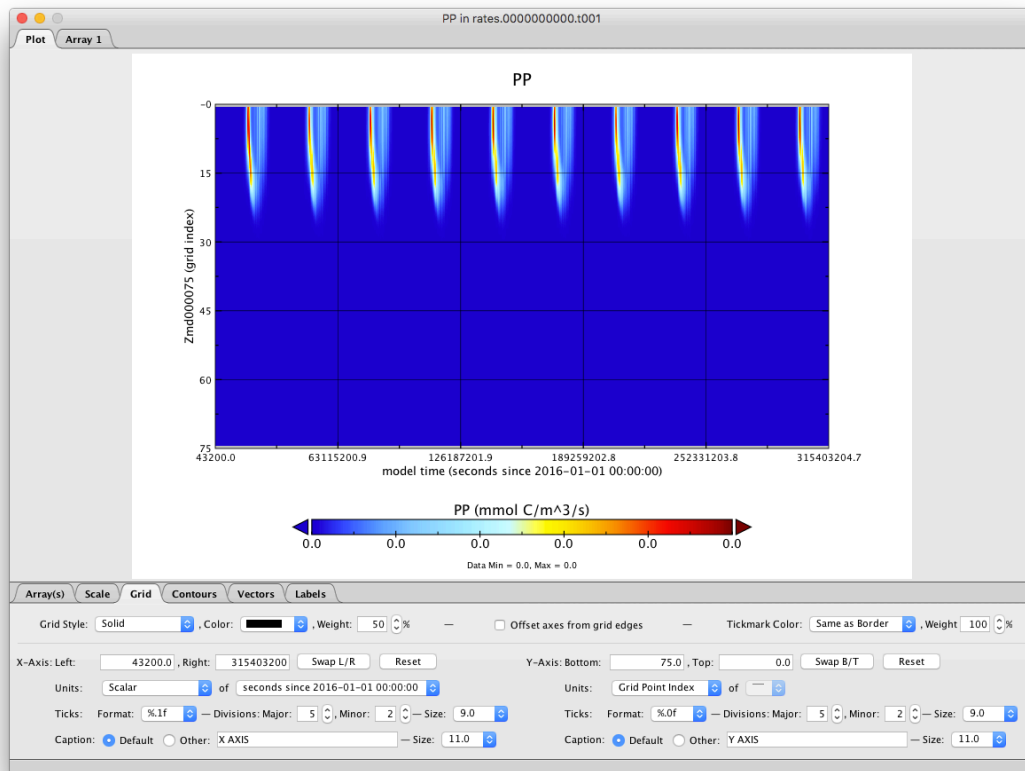


We will retrieve the value of the depth at the depth index 15 using the variable RC in grid.t001.nc.



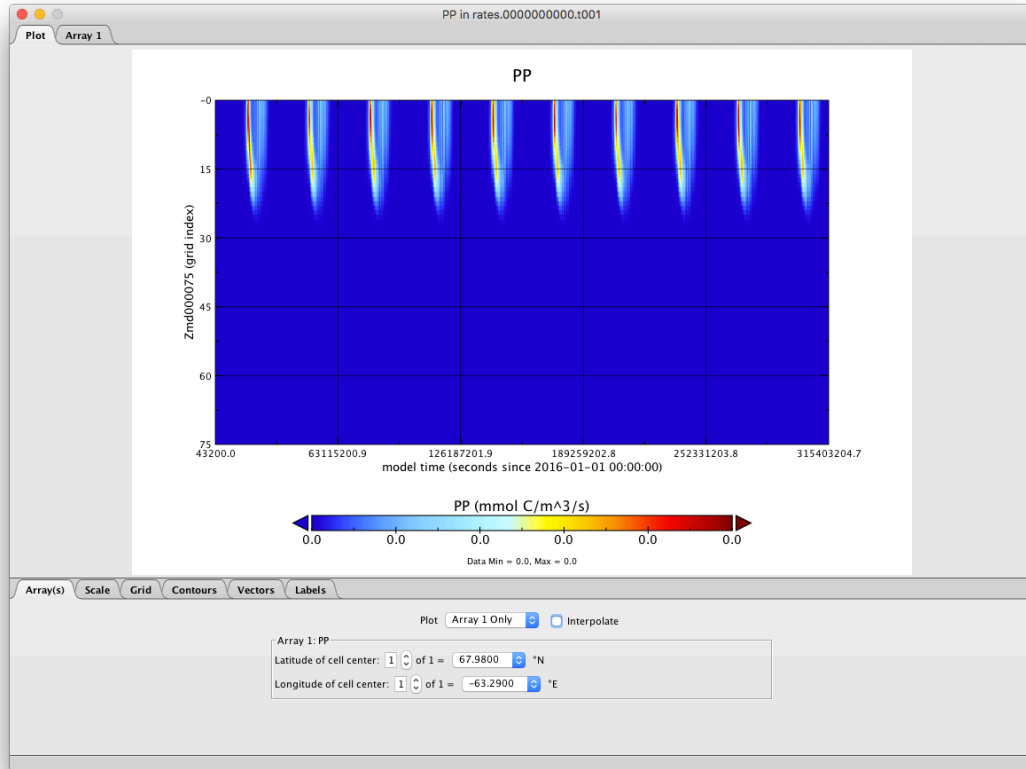
We see that the value of the depth at the depth index 15 is 25.356 m.

- Select the tab Plot.



To make the rows corresponding to each depth clearer:

- Select the tab Array(s).
- Uncheck Interpolate.

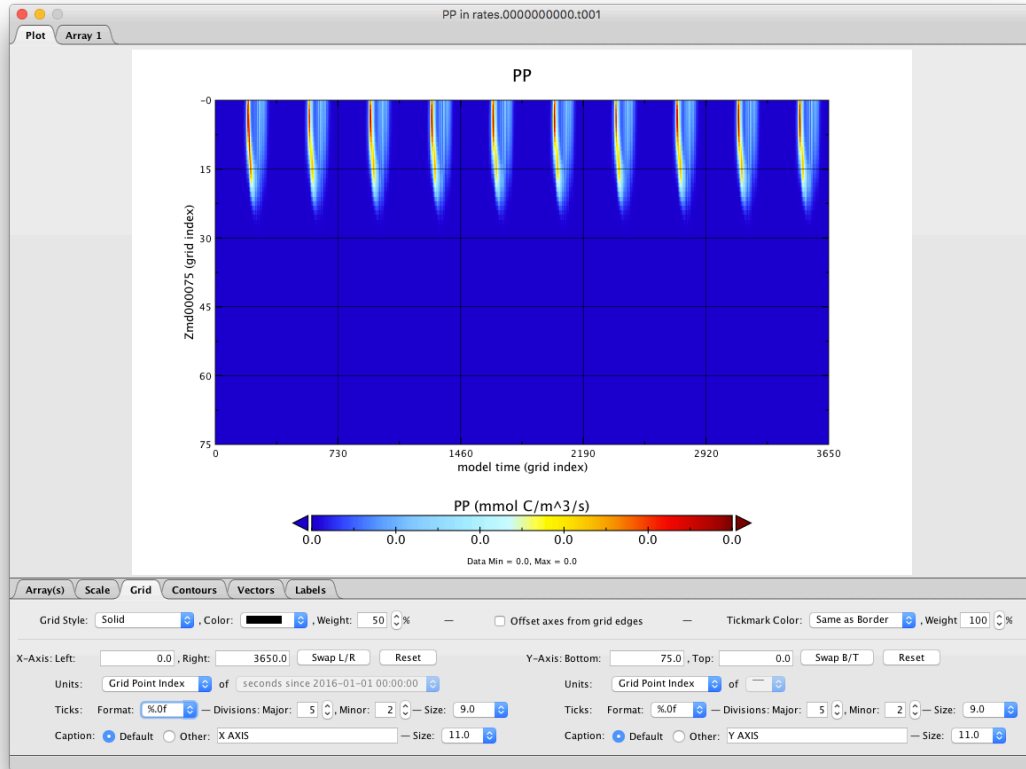


The fifteenth row, just above the tick 15 on the Y-Axis, corresponds to a depth of 25.356 m.

We will now make the X-Axis clearer by showing the indices of the time steps.

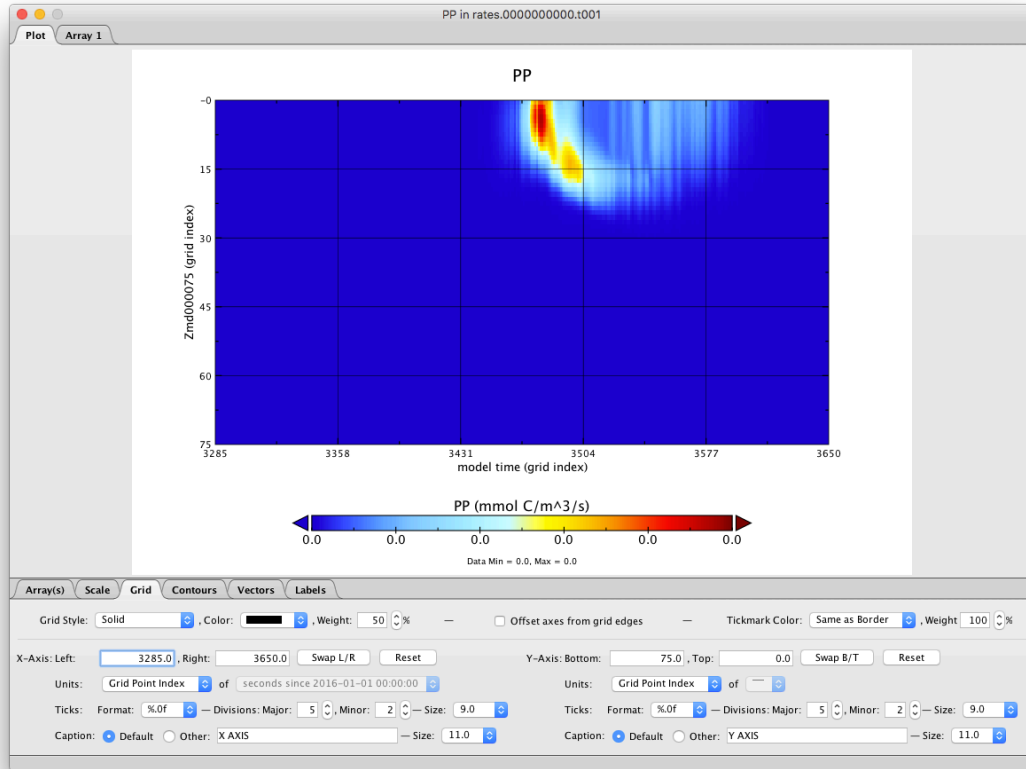
- Select the tab Grid.
- X-Axis: Units: Grid Point Index.
- X-Axis: Ticks: Format: %.0f.





The X axis is now the index of the day over the 10 years of the spinup. We want to select only the tenth year. The indices of the X axis will be  $365 \times 9 = 3285$  (inclusive) to  $365 \times 10 = 3650$  (exclusive).

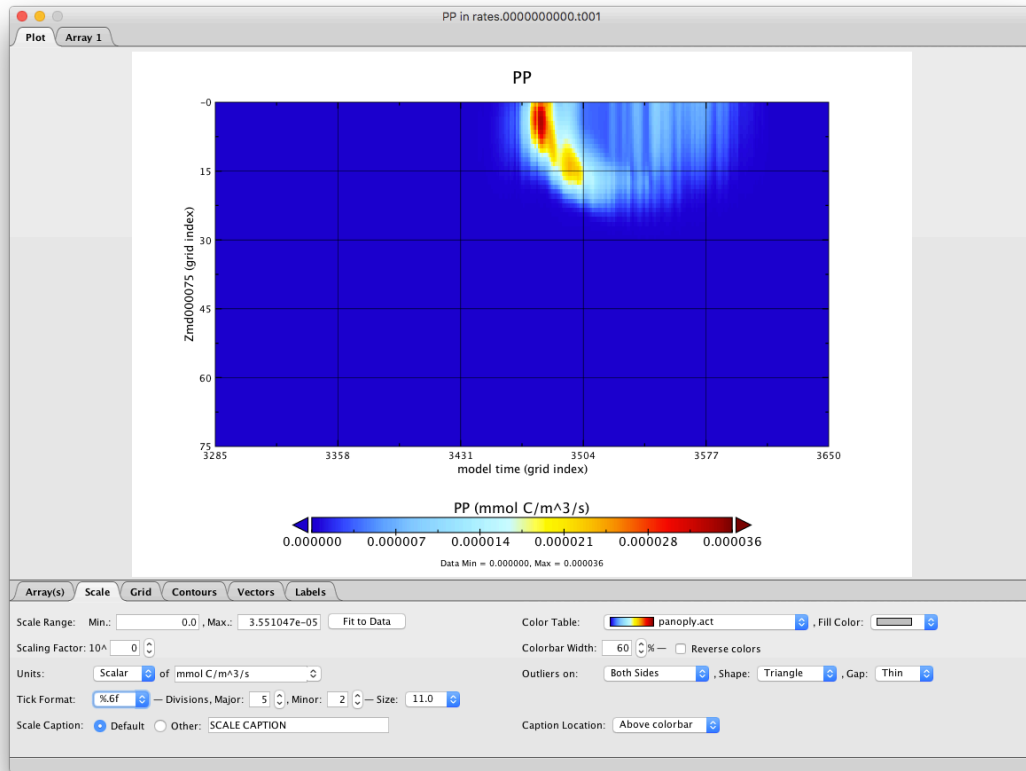
- X-Axis: Left: 3285.



It worths noting that the selection of the tenth year in the Grid tab affects only the plot in the tab Plot. It doesn't affect the array in the tab Array 1. The array in the tab Array 1 will continue to contain the values for the 10 years.

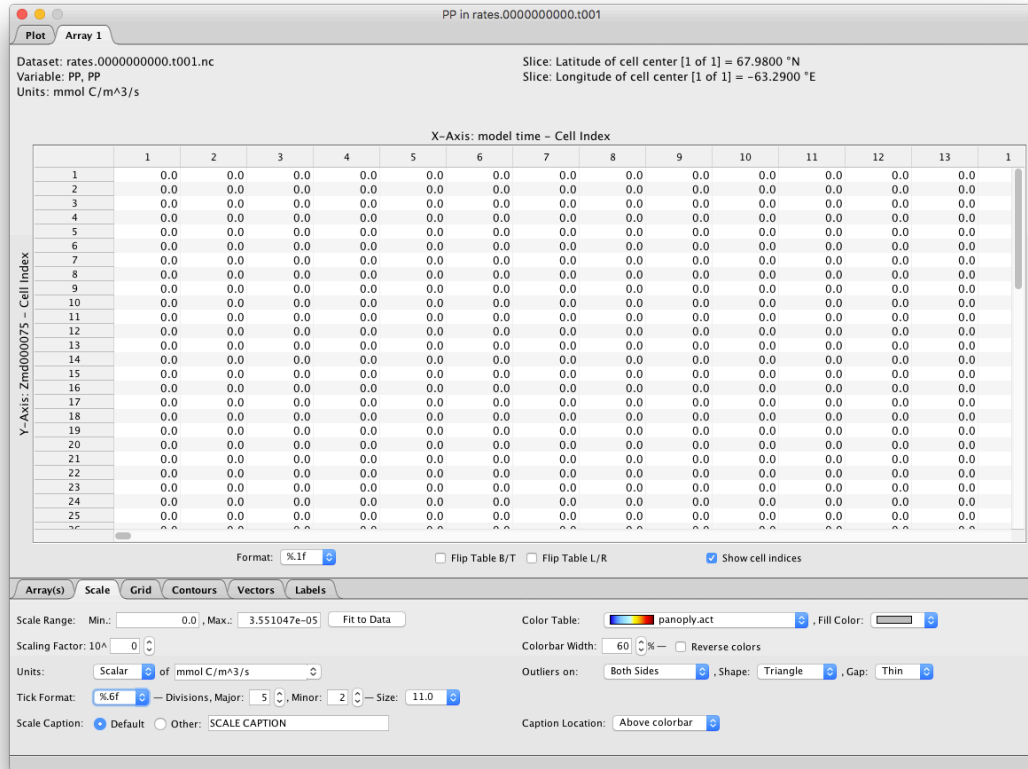
We can improve the precision of the colour bar:

- Select tab Scale.
- Tick Format:  $\%.6f$ .



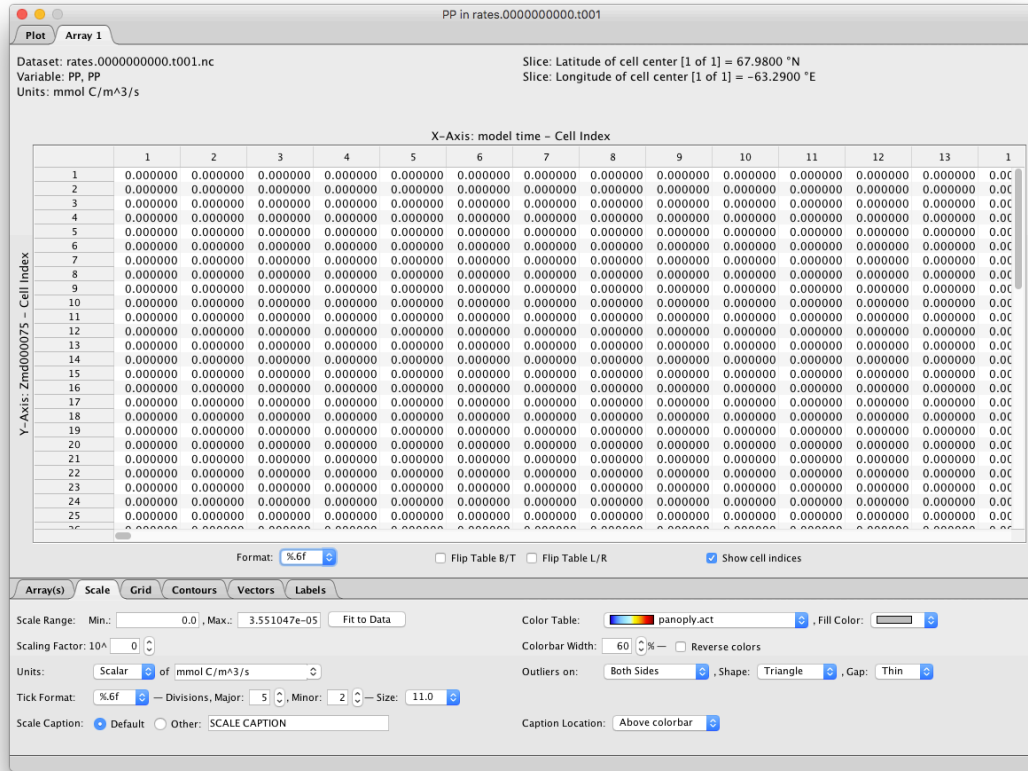
We can find a specific numeric value in the tab Array 1.

- Select tab Array 1.



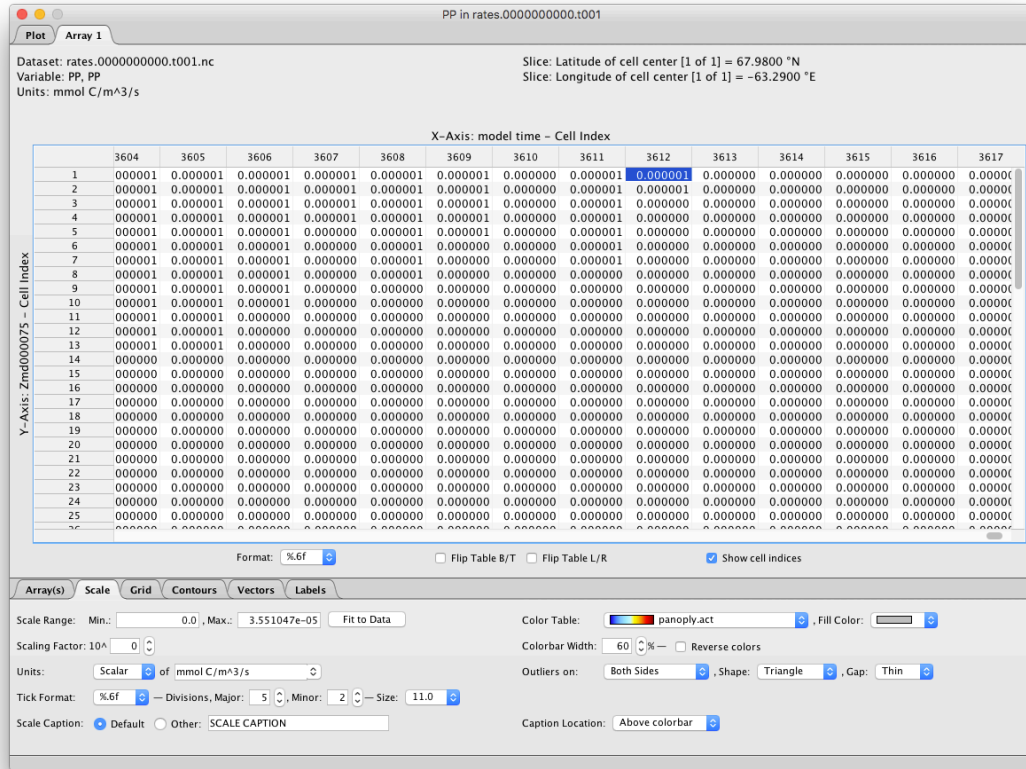
We can improve the precision.

- Format: %.6f.



We now see explicitly the values of the 75\*3650 primary productivities.

Move the horizontal scroll bar to the right to see some values above 0.



## 9 Read a NetCDF file with Python

### 9.1 First, load libraries

```
[1]: import netCDF4
import numpy as np
```

### 9.2 Select year

```
[2]: nbyears=10
last_year=np.arange(365*(nbyears-1),365*(nbyears))
last_year.shape
```

```
[2]: (365,)
```

```
[3]: for i,v in enumerate(last_year):
    if(i<5 or i>=360):
        print(i,v)
    elif(i==5):
```

```
print('...')
```

```
0 3285
1 3286
2 3287
3 3288
4 3289
...
360 3645
361 3646
362 3647
363 3648
364 3649
```

### 9.3 Grid

See the documentation on the grid of the output of the MITgcm model: [https://darwin3.readthedocs.io/en/latest/getting\\_started/getting\\_started.html#grid](https://darwin3.readthedocs.io/en/latest/getting_started/getting_started.html#grid).

RC is the r coordinate of cell center (in m).

```
[4]: gridfile='data/grid.t001.nc'
```

```
[5]: ncfile=gridfile
variable='RC'
try:
    # open the netCDF file for reading
    fh=netCDF4.Dataset(ncfile,'r')
except:
    raise IOError("File not found: {}".format(ncfile))
try:
    # read the data in variable named v
    RC=fh.variables[variable][:]
except:
    raise IOError("Variable {} not found in {}".format(variable, ncfile))
fh.close()
```

We switch sign to have positive depths.

```
[6]: # switch sign
RC=-RC
RC.shape
```

```
[6]: (75,)
```

```
[7]: for i,v in enumerate(RC):
    print(i,v)
```

0 0.5061625  
1 1.5333449999999997  
2 2.5948719999999996  
3 3.7013264999999995  
4 4.866235  
5 6.106664499999999  
6 7.443784999999999  
7 8.903281  
8 10.5154235  
9 12.314566000000001  
10 14.337813500000001  
11 16.6226945  
12 19.203892000000003  
13 22.109410500000003  
14 25.356929  
15 28.9512475  
16 32.883559500000004  
17 37.1327755  
18 41.6684945  
19 46.454799  
20 51.453981  
21 56.6295605  
22 61.9483055  
23 67.381287  
24 72.9041905  
25 78.4971245  
26 84.1441395  
27 89.832647  
28 95.5528315  
29 101.2971055  
30 107.0596445  
31 112.836004  
32 118.622804  
33 124.417478  
34 130.2180915  
35 136.02317200000002  
36 141.8316135  
37 147.6425875  
38 153.4554685  
39 159.269784  
40 165.08516799999998  
41 170.9013685  
42 176.718179  
43 182.5354395  
44 188.35304349999998  
45 194.17090699999997  
46 199.98896899999997  
47 205.80717599999997



```

48 211.62548949999996
49 217.44388699999996
50 223.26235349999996
51 229.08086549999996
52 234.89940799999997
53 240.71797349999997
54 246.53655399999997
55 252.35515749999996
56 258.17376899999994
57 263.99238799999995
58 269.81102199999999
59 275.62965599999999
60 281.44828999999999
61 287.26692399999985
62 293.08557349999984
63 298.90422299999983
64 304.72285699999998
65 310.54150649999998
66 316.36015599999998
67 322.17878999999976
68 327.99743949999976
69 333.81608899999975
70 339.63473849999974
71 345.45338799999973
72 351.2720374999997
73 357.0906869999997
74 362.9093209999997

```

## 9.4 Primary productivity

```
[8]: ppfile='data/rates.0000000000.t001.nc'
```

PP is the primary production (in  $\text{mmol C m}^{-3} \text{ s}^{-1}$ ).

```
[9]: ncfile=ppfile
variable='PP'
try:
    # open the netCDF file for reading
    fh=netCDF4.Dataset(ncfile,'r')
except:
    raise IOError("File not found: {}".format(ncfile))
try:
    # read the data in variable named v
    array2d_idepth_iT_ppfull=fh.variables[variable][:]
except:
    raise IOError("Variable {} not found in {}".format(variable, ncfile))
fh.close()
```

```
array2d_iddepth_iT_ppfull.shape
```

```
[9]: (3650, 75, 1, 1)
```

```
[10]: array2d_iddepth_iT_ppfull=array2d_iddepth_iT_ppfull.squeeze()  
array2d_iddepth_iT_ppfull.shape
```

```
[10]: (3650, 75)
```

```
[11]: array2d_iddepth_iT_ppfull=array2d_iddepth_iT_ppfull.transpose()  
array2d_iddepth_iT_ppfull.shape
```

```
[11]: (75, 3650)
```

```
[12]: array2d_iddepth_iT_pp=array2d_iddepth_iT_ppfull[:,last_year]  
array2d_iddepth_iT_pp.shape
```

```
[12]: (75, 365)
```

The row is the index of the depth (0-based). The column is the index of the day of year of the tenth year (0-based).

## 10 Export a NetCDF file into a CSV file with Python

Suppose we want the comma-separated values (CSV) file PP.csv in a wide form (not data matrix form, a.k.a. tidy form) with 366 columns. The first column is the depth in m. The 365 following columns are the days of year. The first row is the headers. The 75 following rows are the depths.

```
depth_m,doy_001,doy_002,...,doy_365  
0.5061625,...  
...  
362.90932099999997,...
```

First, load libraries.

```
[13]: import pandas as pd
```

```
[14]: df_depth=pd.DataFrame({'depth_m':RC})  
df_depth
```

```
[14]:      depth_m  
0      0.506162  
1      1.533345  
2      2.594872  
3      3.701326  
4      4.866235  
..      ...
```

```

70 339.634738
71 345.453388
72 351.272037
73 357.090687
74 362.909321

```

```
[75 rows x 1 columns]
```

```
[15]: df_PP_names=['doy_{0:03}'.format(i) for i in range(1,366)]
n=len(df_PP_names)
for i in (0,1,2,3,4):
    print(df_PP_names[i])
print('...')
for i in (n-5,n-4,n-3,n-2,n-1):
    print(df_PP_names[i])
```

```

doy_001
doy_002
doy_003
doy_004
doy_005
...
doy_361
doy_362
doy_363
doy_364
doy_365

```

```
[16]: df_PP=pd.DataFrame(array2d_idepth_iT_pp)
df_PP.columns=df_PP_names
df_PP
```

```
[16]:
```

	doy_001	doy_002	doy_003	doy_004	doy_005	doy_006	doy_007	\
0	0.0	0.0	0.0	0.0	0.0	3.003517e-12	9.562626e-12	
1	0.0	0.0	0.0	0.0	0.0	2.746950e-12	9.208227e-12	
2	0.0	0.0	0.0	0.0	0.0	2.635478e-12	8.835832e-12	
3	0.0	0.0	0.0	0.0	0.0	2.376848e-12	8.455584e-12	
4	0.0	0.0	0.0	0.0	0.0	2.267716e-12	8.068476e-12	
..	...	...	...	...	...	...	...	
70	0.0	0.0	0.0	0.0	0.0	0.000000e+00	0.000000e+00	
71	0.0	0.0	0.0	0.0	0.0	0.000000e+00	0.000000e+00	
72	0.0	0.0	0.0	0.0	0.0	0.000000e+00	0.000000e+00	
73	0.0	0.0	0.0	0.0	0.0	0.000000e+00	0.000000e+00	
74	0.0	0.0	0.0	0.0	0.0	0.000000e+00	0.000000e+00	
	doy_008	doy_009	doy_010	...	doy_356	doy_357	doy_358	\
0	3.982557e-11	6.567130e-11	6.554848e-11	...	0.0	0.0	0.0	

1	3.840060e-11	6.336488e-11	6.324807e-11	...	0.0	0.0	0.0
2	3.689539e-11	6.092277e-11	6.081262e-11	...	0.0	0.0	0.0
3	3.535327e-11	5.841642e-11	5.831308e-11	...	0.0	0.0	0.0
4	3.377870e-11	5.585338e-11	5.575687e-11	...	0.0	0.0	0.0
..	...	...	...	...	...	...	...
70	0.000000e+00	0.000000e+00	0.000000e+00	...	0.0	0.0	0.0
71	0.000000e+00	0.000000e+00	0.000000e+00	...	0.0	0.0	0.0
72	0.000000e+00	0.000000e+00	0.000000e+00	...	0.0	0.0	0.0
73	0.000000e+00	0.000000e+00	0.000000e+00	...	0.0	0.0	0.0
74	0.000000e+00	0.000000e+00	0.000000e+00	...	0.0	0.0	0.0

	doy_359	doy_360	doy_361	doy_362	doy_363	doy_364	doy_365
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0
..	...	...	...	...	...	...	...
70	0.0	0.0	0.0	0.0	0.0	0.0	0.0
71	0.0	0.0	0.0	0.0	0.0	0.0	0.0
72	0.0	0.0	0.0	0.0	0.0	0.0	0.0
73	0.0	0.0	0.0	0.0	0.0	0.0	0.0
74	0.0	0.0	0.0	0.0	0.0	0.0	0.0

[75 rows x 365 columns]

```
[17]: df=pd.concat([df_depth,df_PP],axis='columns')
df
```

```
[17]:
```

	depth_m	doy_001	doy_002	doy_003	doy_004	doy_005	doy_006	\
0	0.506162	0.0	0.0	0.0	0.0	0.0	3.003517e-12	
1	1.533345	0.0	0.0	0.0	0.0	0.0	2.746950e-12	
2	2.594872	0.0	0.0	0.0	0.0	0.0	2.635478e-12	
3	3.701326	0.0	0.0	0.0	0.0	0.0	2.376848e-12	
4	4.866235	0.0	0.0	0.0	0.0	0.0	2.267716e-12	
..	...	...	...	...	...	...	...	
70	339.634738	0.0	0.0	0.0	0.0	0.0	0.000000e+00	
71	345.453388	0.0	0.0	0.0	0.0	0.0	0.000000e+00	
72	351.272037	0.0	0.0	0.0	0.0	0.0	0.000000e+00	
73	357.090687	0.0	0.0	0.0	0.0	0.0	0.000000e+00	
74	362.909321	0.0	0.0	0.0	0.0	0.0	0.000000e+00	

	doy_007	doy_008	doy_009	...	doy_356	doy_357	doy_358	\
0	9.562626e-12	3.982557e-11	6.567130e-11	...	0.0	0.0	0.0	
1	9.208227e-12	3.840060e-11	6.336488e-11	...	0.0	0.0	0.0	
2	8.835832e-12	3.689539e-11	6.092277e-11	...	0.0	0.0	0.0	
3	8.455584e-12	3.535327e-11	5.841642e-11	...	0.0	0.0	0.0	

4	8.068476e-12	3.377870e-11	5.585338e-11	...	0.0	0.0	0.0
..	...	...	...	...	...	...	...
70	0.000000e+00	0.000000e+00	0.000000e+00	...	0.0	0.0	0.0
71	0.000000e+00	0.000000e+00	0.000000e+00	...	0.0	0.0	0.0
72	0.000000e+00	0.000000e+00	0.000000e+00	...	0.0	0.0	0.0
73	0.000000e+00	0.000000e+00	0.000000e+00	...	0.0	0.0	0.0
74	0.000000e+00	0.000000e+00	0.000000e+00	...	0.0	0.0	0.0

	doy_359	doy_360	doy_361	doy_362	doy_363	doy_364	doy_365
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0
..	...	...	...	...	...	...	...
70	0.0	0.0	0.0	0.0	0.0	0.0	0.0
71	0.0	0.0	0.0	0.0	0.0	0.0	0.0
72	0.0	0.0	0.0	0.0	0.0	0.0	0.0
73	0.0	0.0	0.0	0.0	0.0	0.0	0.0
74	0.0	0.0	0.0	0.0	0.0	0.0	0.0

[75 rows x 366 columns]

```
[18]: outfile='PP.csv'
df.to_csv(outfile,index=False)
```

The screenshot shows an Excel spreadsheet with a data table. The columns are labeled 'depth\_m' (column 1) and 'doy\_001' through 'doy\_020' (columns 2-21). The rows represent different depths, with labels like 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, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 273, 274, 275, 276, 277, 278, 279, 280, 281, 282, 283, 284, 285, 286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298, 299, 300, 301, 302, 303, 304, 305, 306, 307, 308, 309, 310, 311, 312, 313, 314, 315, 316, 317, 318, 319, 320, 321, 322, 323, 324, 325, 326, 327, 328, 329, 330, 331, 332, 333, 334, 335, 336, 337, 338, 339, 340, 341, 342, 343, 344, 345, 346, 347, 348, 349, 350, 351, 352, 353, 354, 355, 356, 357, 358, 359, 360, 361, 362, 363, 364, 365, 366, 367, 368, 369, 370, 371, 372, 373, 374, 375, 376, 377, 378, 379, 380, 381, 382, 383, 384, 385, 386, 387, 388, 389, 390, 391, 392, 393, 394, 395, 396, 397, 398, 399, 400, 401, 402, 403, 404, 405, 406, 407, 408, 409, 410, 411, 412, 413, 414, 415, 416, 417, 418, 419, 420, 421, 422, 423, 424, 425, 426, 427, 428, 429, 430, 431, 432, 433, 434, 435, 436, 437, 438, 439, 440, 441, 442, 443, 444, 445, 446, 447, 448, 449, 450, 451, 452, 453, 454, 455, 456, 457, 458, 459, 460, 461, 462, 463, 464, 465, 466, 467, 468, 469, 470, 471, 472, 473, 474, 475, 476, 477, 478, 479, 480, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 505, 506, 507, 508, 509, 510, 511, 512, 513, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 525, 526, 527, 528, 529, 530, 531, 532, 533, 534, 535, 536, 537, 538, 539, 540, 541, 542, 543, 544, 545, 546, 547, 548, 549, 550, 551, 552, 553, 554, 555, 556, 557, 558, 559, 560, 561, 562, 563, 564, 565, 566, 567, 568, 569, 570, 571, 572, 573, 574, 575, 576, 577, 578, 579, 580, 581, 582, 583, 584, 585, 586, 587, 588, 589, 590, 591, 592, 593, 594, 595, 596, 597, 598, 599, 600, 601, 602, 603, 604, 605, 606, 607, 608, 609, 610, 611, 612, 613, 614, 615, 616, 617, 618, 619, 620, 621, 622, 623, 624, 625, 626, 627, 628, 629, 630, 631, 632, 633, 634, 635, 636, 637, 638, 639, 640, 641, 642, 643, 644, 645, 646, 647, 648, 649, 650, 651, 652, 653, 654, 655, 656, 657, 658, 659, 660, 661, 662, 663, 664, 665, 666, 667, 668, 669, 670, 671, 672, 673, 674, 675, 676, 677, 678, 679, 680, 681, 682, 683, 684, 685, 686, 687, 688, 689, 690, 691, 692, 693, 694, 695, 696, 697, 698, 699, 700, 701, 702, 703, 704, 705, 706, 707, 708, 709, 710, 711, 712, 713, 714, 715, 716, 717, 718, 719, 720, 721, 722, 723, 724, 725, 726, 727, 728, 729, 730, 731, 732, 733, 734, 735, 736, 737, 738, 739, 740, 741, 742, 743, 744, 745, 746, 747, 748, 749, 750, 751, 752, 753, 754, 755, 756, 757, 758, 759, 760, 761, 762, 763, 764, 765, 766, 767, 768, 769, 770, 771, 772, 773, 774, 775, 776, 777, 778, 779, 780, 781, 782, 783, 784, 785, 786, 787, 788, 789, 790, 791, 792, 793, 794, 795, 796, 797, 798, 799, 800, 801, 802, 803, 804, 805, 806, 807, 808, 809, 810, 811, 812, 813, 814, 815, 816, 817, 818, 819, 820, 821, 822, 823, 824, 825, 826, 827, 828, 829, 830, 831, 832, 833, 834, 835, 836, 837, 838, 839, 840, 841, 842, 843, 844, 845, 846, 847, 848, 849, 850, 851, 852, 853, 854, 855, 856, 857, 858, 859, 860, 861, 862, 863, 864, 865, 866, 867, 868, 869, 870, 871, 872, 873, 874, 875, 876, 877, 878, 879, 880, 881, 882, 883, 884, 885, 886, 887, 888, 889, 890, 891, 892, 893, 894, 895, 896, 897, 898, 899, 900, 901, 902, 903, 904, 905, 906, 907, 908, 909, 910, 911, 912, 913, 914, 915, 916, 917, 918, 919, 920, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000.

## 11 Plot a NetCDF file with Python

First, load libraries.

```
[19]: import matplotlib as mpl
import matplotlib.pyplot as plt
```

```
[20]: plt.close("all")
```

one\_year\_for\_heatmaps is the coordinates on the X-Axis of the corners of quadrilaterals of the pcolormesh (see [https://matplotlib.org/stable/api/\\_as\\_gen/matplotlib.pyplot.pcolormesh.html](https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.pcolormesh.html)).

```
[21]: one_year_for_heatmaps=np.arange(0,366)
```

RF\_above100 is the coordinates on the Y-Axis of the corners of quadrilaterals of the pcolormesh. We decide to show only the top 100 m.

See the documentation on the grid of the output of the MITgcm model: [https://darwin3.readthedocs.io/en/latest/getting\\_started/getting\\_started.html#grid](https://darwin3.readthedocs.io/en/latest/getting_started/getting_started.html#grid).

RF is the r coordinate of cell interface (in m).

```
[22]: gridfile='data/grid.t001.nc'
```

```
[23]: ncfile=gridfile
variable='RF'
try:
    # open the netCDF file for reading
    fh=netCDF4.Dataset(ncfile,'r')
except:
    raise IOError("File not found: {}".format(ncfile))
try:
    # read the data in variable named v
    RF=fh.variables[variable][:]
except:
    raise IOError("Variable {} not found in {}".format(variable, ncfile))
fh.close()
```

We switch sign to have positive depths.

```
[24]: # swith sign
RF=-RF
RF.shape
```

```
[24]: (76,)
```

```
[25]: for i,v in enumerate(RF):
    print(i,v)
```

0 -0.0  
1 1.012325  
2 2.0543649999999998  
3 3.1353789999999995  
4 4.267274  
5 5.465196  
6 6.748132999999999  
7 8.139437  
8 9.667124999999999  
9 11.363722  
10 13.26541  
11 15.410217  
12 17.835172  
13 20.572612  
14 23.646209  
15 27.067649  
16 30.834846  
17 34.932272999999995  
18 39.33327799999999  
19 44.003710999999996  
20 48.90588699999999  
21 54.00207499999999  
22 59.25704599999999  
23 64.63956499999999  
24 70.123009  
25 75.685372  
26 81.308877  
27 86.979402  
28 92.685892  
29 98.419771  
30 104.17444  
31 109.944849  
32 115.727159  
33 121.518449  
34 127.316507  
35 133.119676  
36 138.926668  
37 144.736559  
38 150.548616  
39 156.362321  
40 162.177247000000002  
41 167.993089000000003  
42 173.809648000000004  
43 179.626710000000003  
44 185.444169000000004  
45 191.261918000000004  
46 197.079896000000005  
47 202.898042000000006

```

48 208.71631000000005
49 214.53466900000004
50 220.35310500000003
51 226.17160200000004
52 231.99012900000002
53 237.80868700000002
54 243.62726
55 249.445848
56 255.26446700000002
57 261.083071
58 266.901705
59 272.72033899999997
60 278.53897299999994
61 284.35760699999999
62 290.17624099999999
63 295.99490599999999
64 301.81353999999999
65 307.63217399999985
66 313.45083899999986
67 319.26947299999983
68 325.08810699999998
69 330.90677199999998
70 336.72540599999998
71 342.54407099999998
72 348.36270499999998
73 354.18136999999998
74 360.00000399999976
75 365.81863799999974

```

```
[26]: RF_above100=np.array(RF[RF<100])
```

```
[27]: def make_plots(ax):
    locs=np.array([0, 31, 59, 90, 120, 151,
                  181, 212, 243, 273, 304, 334])
    labels=('Jan-1','Feb-1','Mar-1','Apr-1','May-1','Jun-1',
            'Jul-1','Aug-1','Sep-1','Oct-1','Nov-1','Dec-1')
    h=ax.pcolormesh(one_year_for_heatmaps,
                    RF_above100,
                    array2d_iddepth_iT_pp[0:(RF_above100.size)-1,:],
                    cmap='viridis',
                    vmin=0,
                    vmax=0.00004)
    ax.set_xticklabels([])
    ax.set_xticks(locs)
    ax.set_xticklabels(labels)
    ax.set_ylabel('Depth (m)')
    ax.set_ylim(98.5,0)
```



```

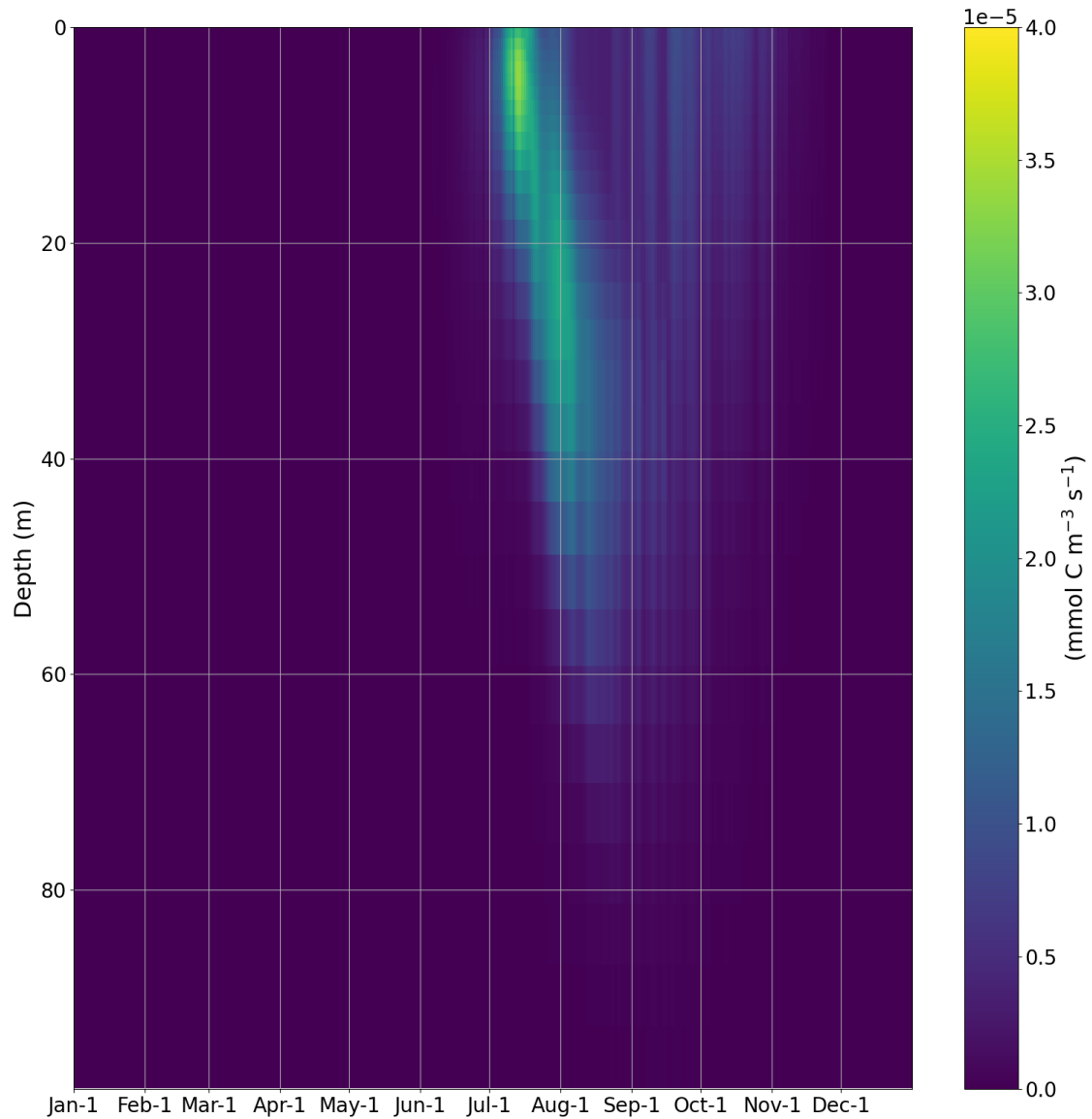
ax.grid()
cbar=plt.colorbar(h)
cbar.set_label('($\mathrm{ mmol\ C\ m^{-3}\ s^{-1} }$)')

plt.tight_layout()

with plt.style.context('mplstyles/heatmaps.mplstyle'):
    # Plot
    fig=plt.figure(figsize=(16, 16))
    ax=fig.add_subplot(111)
    make_plots(ax)

    # --- SAVE
    plt.savefig('figures/PP.png')
    plt.show()

```



## 12 More examples in Python

Other examples of Python scripts reading the results of Benoît-Gagné et al. (submitted) are available on <https://github.com/maximebenoitgagne/wintertime/blob/v1.5/wintertime.ipynb>.