fluorescenceOffset

August 20, 2018

1 Offset calibration of Spray's fluorescence data

The objective here is to find an offset for each mission.

Concept: It would be expected to observe very low, if not zero, cholorphyll in deep ocean. Therefore, deep profiles should be able to see a zero chrolophyll reading.

Steps applied here:

1. Load and clean data
2. Prepare data. Calculate some products:
    - days in mission (dt)
    - minimum fl per profile (fl_min)
    - Max depth of fl per profile (profile_max_depth)
    - Fl deep reference (fl_deep_ref)
3. Data overview and sanity check: How does the raw data looks like?
4. Quality Control. For now, let's focus on the easier cases and remove the missions that are too different and would require quite high offset.
5. Vertical structure. How does the fl(z) looks like?
    - Deep layers aren't zero fl
    - Deeper casts dominate
    - Small tendency on deep measurements
6. fl0, Minimum fl in a mission as zero reference
7. fl_texp

```
In [1]: %matplotlib inline

        from datetime import timedelta

        import numpy as np
        import pandas as pd
        import xarray as xr
        import pymc3 as pm

        import matplotlib.pyplot as plt
        from matplotlib.ticker import NullFormatter
```

```
In [ ]: #import warnings
        #warnings.filterwarnings('ignore')
```

2 Loading data

The file flMatched_4km.hdf was created by matchup.py and contains all spray missions from CUGN plus fluorescence from MODIS-Aqua, MODIS-Terra, and VIIRS that are inside a space/time range of each profile.

```
In [239]: inputFilename = '../data/flMatched_4km.hdf'
          profile = pd.read_hdf(inputFilename, key='profile')
          data = pd.read_hdf(inputFilename, key='data')
```

```
In [240]: spray = xr.merge([profile.to_xarray(), data.to_xarray()],
                           join='inner')

          aux_coords = ['ndive', 'datetime', 'lat', 'lon', 'mission',
                        'mission_id', 'experiment', 'experiment_id']
          spray.set_coords(aux_coords, inplace=True)

          print(spray)
```

```
<xarray.Dataset>
Dimensions:        (depth: 100, profile_id: 99752)
Coordinates:
  * profile_id     (profile_id) int64 20215 20216 20217 20218 20219 20220 ...
    ndive          (profile_id) int64 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 ...
    datetime       (profile_id) datetime64[ns] 2006-10-16T19:46:51 ...
    lat            (profile_id) float64 34.35 34.35 34.35 34.34 34.34 34.34 ...
    lon            (profile_id) float64 -119.8 -119.8 -119.8 -119.8 -119.8 ...
    mission_id     (profile_id) int64 106 106 106 106 106 106 106 106 106 ...
    mission        (profile_id) object '06A00501' '06A00501' '06A00501' ...
    experiment_id  (profile_id) int64 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 ...
    experiment     (profile_id) object 'CUGN_line_80' 'CUGN_line_80' ...
  * depth          (depth) float64 10.0 20.0 30.0 40.0 50.0 60.0 70.0 80.0 ...
Data variables:
    temp           (depth, profile_id) float64 16.57 16.38 16.24 16.16 16.39 ...
    sal            (depth, profile_id) float64 33.42 33.41 33.43 33.43 33.43 ...
    fl             (depth, profile_id) float64 1.382 1.23 1.429 2.101 2.857 ...
```

2.1 Keep only the usefull data

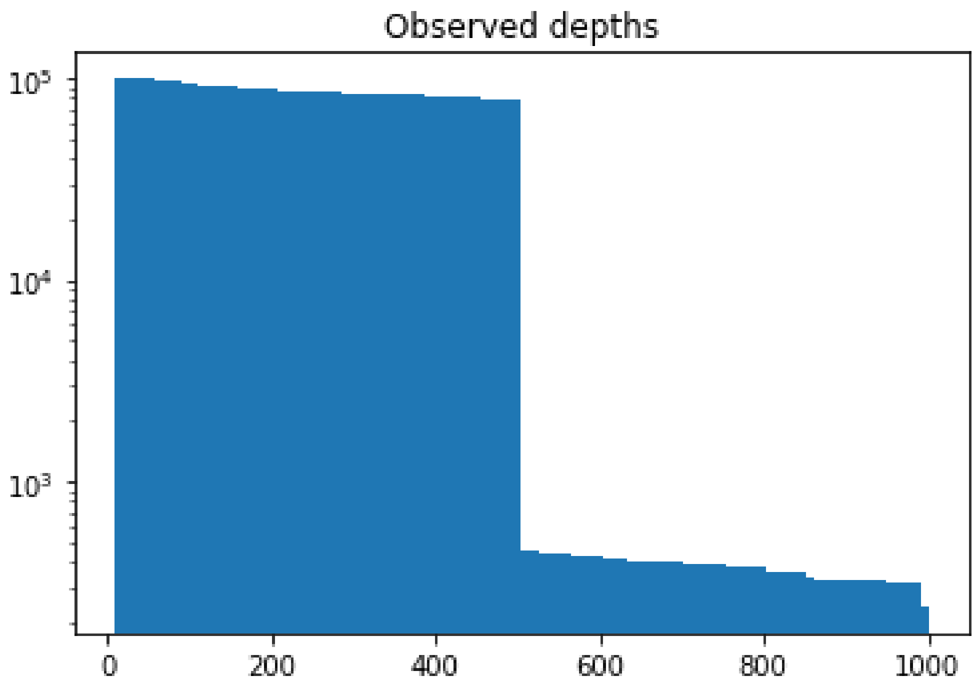
Removing depths without any data (temp, sal, and fl), therefore depths of NaN. The number of profiles should be preserved.

```
In [241]: spray = spray.dropna(dim='depth', how='all', subset=['fl', 'temp', 'sal'])
          print('After removing depths without data')
          print(spray.dims)
          print('Included depths')
          print(spray.depth)
```

```
After removing depths without data
Frozen(SortedKeysDict(OrderedDict([('profile_id', 99752), ('depth', 100)])))
Included depths
<xarray.DataArray 'depth' (depth: 100)>
array([  10.,   20.,   30.,   40.,   50.,   60.,   70.,   80.,   90.,  100.,
        110.,  120.,  130.,  140.,  150.,  160.,  170.,  180.,  190.,  200.,
        210.,  220.,  230.,  240.,  250.,  260.,  270.,  280.,  290.,  300.,
        310.,  320.,  330.,  340.,  350.,  360.,  370.,  380.,  390.,  400.,
        410.,  420.,  430.,  440.,  450.,  460.,  470.,  480.,  490.,  500.,
        510.,  520.,  530.,  540.,  550.,  560.,  570.,  580.,  590.,  600.,
        610.,  620.,  630.,  640.,  650.,  660.,  670.,  680.,  690.,  700.,
        710.,  720.,  730.,  740.,  750.,  760.,  770.,  780.,  790.,  800.,
        810.,  820.,  830.,  840.,  850.,  860.,  870.,  880.,  890.,  900.,
        910.,  920.,  930.,  940.,  950.,  960.,  970.,  980.,  990., 1000.])
Coordinates:
  * depth    (depth) float64 10.0 20.0 30.0 40.0 50.0 60.0 70.0 80.0 90.0 ...
```

```
In [242]: # Before I would truncate at 500 m.
          # spray = spray.isel(depth=spray.depth<=500)
          # print(spray.depth)
          # print(spray.dims)
```

```
In [243]: plt.hist(spray.fl.to_series().dropna().reset_index()['depth'], bins=100)
          plt.yscale('log', nonposy='clip')
          noprint = plt.title('Observed depths')
```



Only few profiles go below 500m. Maybe consider to restrict the data to the upper 500m? It might be usefull to check dependency of fluorometer versus temperature or pressure.

3 Preparing data

- Removing depths without any valid data
- Getting some derivate values, like relative time since start of the mission, etc.

3.0.1 Create dt as days since start of the mission

```
In [244]: def days_since_start(x):
              return (x - x.min()) / (np.timedelta64(1, 's') * 86400)

          spray['days'] = spray.datetime.groupby('mission').apply(days_since_start)
          spray.days.attrs['description'] = "Number of days since the start of the mission."

          spray.set_coords('days', inplace=True)
```

```
Out[244]: <xarray.Dataset>
          Dimensions:         (depth: 100, profile_id: 99752)
          Coordinates:
            * profile_id      (profile_id) int64 20215 20216 20217 20218 20219 20220 ...
              ndive           (profile_id) int64 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 ...
              datetime        (profile_id) datetime64[ns] 2006-10-16T19:46:51 ...
              lat             (profile_id) float64 34.35 34.35 34.35 34.34 34.34 34.34 ...
```

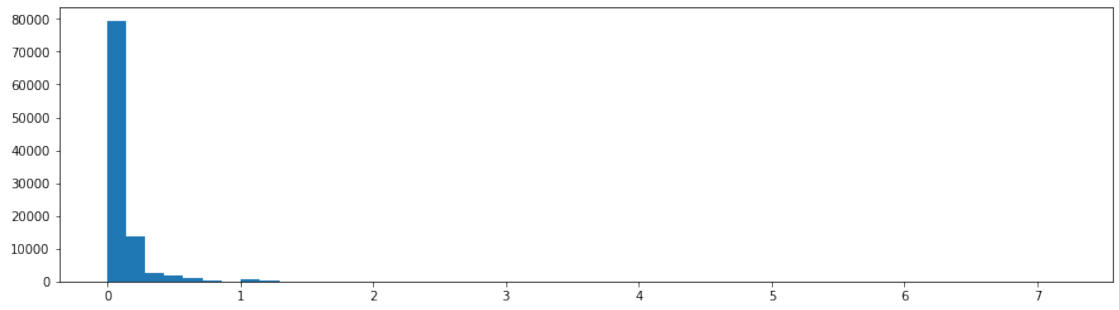
```
    lon             (profile_id) float64 -119.8 -119.8 -119.8 -119.8 -119.8 ...
    mission_id      (profile_id) int64 106 106 106 106 106 106 106 106 106 ...
    mission         (profile_id) object '06A00501' '06A00501' '06A00501' ...
    experiment_id   (profile_id) int64 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 ...
    experiment      (profile_id) object 'CUGN_line_80' 'CUGN_line_80' ...
  * depth           (depth) float64 10.0 20.0 30.0 40.0 50.0 60.0 70.0 80.0 ...
    days            (profile_id) float64 0.0 0.02537 0.04983 0.07683 0.1064 ...
Data variables:
    temp            (depth, profile_id) float64 16.57 16.38 16.24 16.16 16.39 ...
    sal             (depth, profile_id) float64 33.42 33.41 33.43 33.43 33.43 ...
    fl              (depth, profile_id) float64 1.382 1.23 1.429 2.101 2.857 ...
```

3.0.2 Obtain lowest fl per profile

Restricts to upper 500 m to avoid potential variability due to few profiles going up to 1000 m.

```
In [245]: # Minima fluorescence observed on each profile
          spray['fl_min'] = spray.isel(depth=spray.depth<=500).fl.min(dim='depth')
          spray.fl_min.attrs['description'] = "Minimum fl observed per profile"

          plt.figure(figsize=(15,4))
          plt.hist(spray.fl_min, bins=50)

          print(spray)
```

```
<xarray.Dataset>
Dimensions:         (depth: 100, profile_id: 99752)
Coordinates:
  * profile_id      (profile_id) int64 20215 20216 20217 20218 20219 20220 ...
    ndive           (profile_id) int64 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 ...
    datetime        (profile_id) datetime64[ns] 2006-10-16T19:46:51 ...
    lat             (profile_id) float64 34.35 34.35 34.35 34.34 34.34 34.34 ...
    lon             (profile_id) float64 -119.8 -119.8 -119.8 -119.8 -119.8 ...
    mission_id      (profile_id) int64 106 106 106 106 106 106 106 106 106 ...
    mission         (profile_id) object '06A00501' '06A00501' '06A00501' ...
    experiment_id   (profile_id) int64 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 ...
    experiment      (profile_id) object 'CUGN_line_80' 'CUGN_line_80' ...
  * depth           (depth) float64 10.0 20.0 30.0 40.0 50.0 60.0 70.0 80.0 ...
    days            (profile_id) float64 0.0 0.02537 0.04983 0.07683 0.1064 ...
Data variables:
    temp            (depth, profile_id) float64 16.57 16.38 16.24 16.16 16.39 ...
    sal             (depth, profile_id) float64 33.42 33.41 33.43 33.43 33.43 ...
    fl              (depth, profile_id) float64 1.382 1.23 1.429 2.101 2.857 ...
    fl_min          (profile_id) float64 0.2884 0.2508 0.2347 0.1904 0.144 ...
```

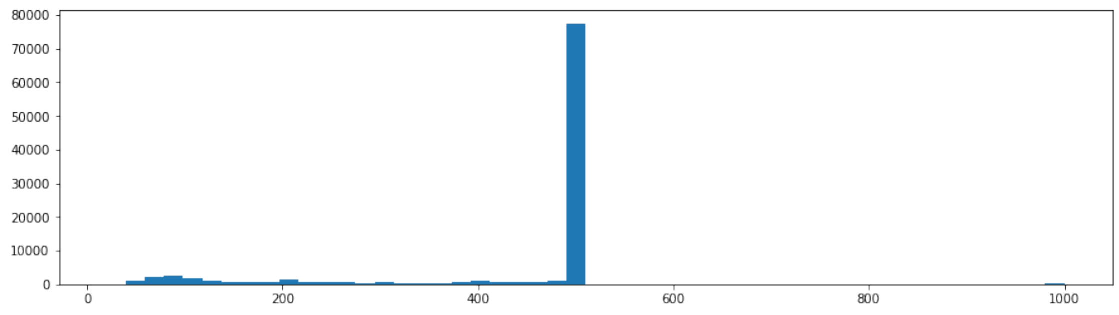


3.0.3 Obtain max depth per profile of observed fl

Most of the Chl activity happens near the surface. Profiles that don't go deep enough shall not have provide any reference of no Chl activity, thus could mislead the offset estimate.

```
In [246]: max_depth = spray.fl.to_series().dropna().reset_index('depth').groupby('profile_id')
          spray['profile_max_depth'] = max_depth.to_xarray()
          del(max_depth)
          spray.profile_max_depth.attrs['description'] = "Deepest fl measurement per profile."

          plt.figure(figsize=(15,4))
          plt.hist(spray.profile_max_depth, bins=50)
          spray
```

```
Out[246]: <xarray.Dataset>
          Dimensions:            (depth: 100, profile_id: 99752)
          Coordinates:
            * profile_id         (profile_id) int64 20215 20216 20217 20218 20219 ...
              ndive              (profile_id) int64 1 2 3 4 5 6 7 8 9 10 11 12 13 14 ...
              datetime           (profile_id) datetime64[ns] 2006-10-16T19:46:51 ...
              lat                (profile_id) float64 34.35 34.35 34.35 34.34 34.34 ...
              lon                (profile_id) float64 -119.8 -119.8 -119.8 -119.8 ...
              mission_id         (profile_id) int64 106 106 106 106 106 106 106 106 ...
              mission            (profile_id) object '06A00501' '06A00501' '06A00501' ...
              experiment_id      (profile_id) int64 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 ...
              experiment         (profile_id) object 'CUGN_line_80' 'CUGN_line_80' ...
            * depth              (depth) float64 10.0 20.0 30.0 40.0 50.0 60.0 70.0 ...
              days               (profile_id) float64 0.0 0.02537 0.04983 0.07683 ...
          Data variables:
              temp               (depth, profile_id) float64 16.57 16.38 16.24 16.16 ...
              sal                (depth, profile_id) float64 33.42 33.41 33.43 33.43 ...
              fl                 (depth, profile_id) float64 1.382 1.23 1.429 2.101 ...
              fl_min             (profile_id) float64 0.2884 0.2508 0.2347 0.1904 ...
              profile_max_depth  (profile_id) float64 60.0 80.0 80.0 90.0 110.0 150.0 ...
```



3.0.4 Fluorescence deep reference

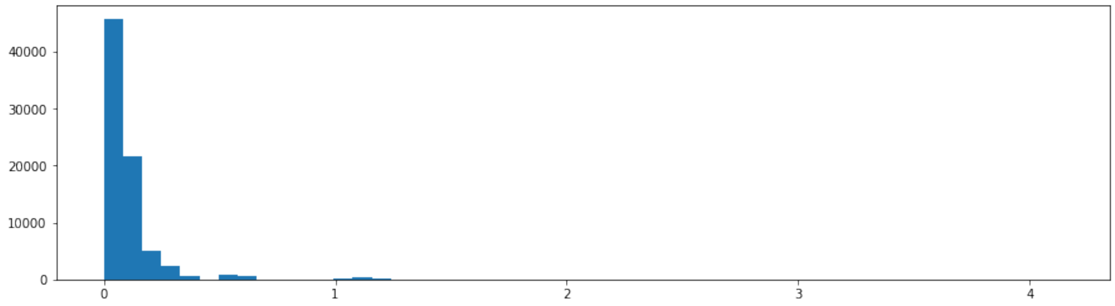
Originally I used the smallest value observed in the profile, which had some high frequency variability. As suggested by Dan, I'm now using the percentile of 5% of all fl observed between 300 and 500 m. This gives a little more 'stability', and is reasobable to expect that 5% of the measurements on this layer should be zero or near zero.

I'm also restricting to profiles that measures at least up to 400m, thus it has at least 100m (from 300 to 400m) of data.

```
In [247]: fl_deep_ref = spray.isel(
              profile_id=spray.profile_max_depth >= 400,
              depth=((spray.depth >= 300) & (spray.depth <=500))
              ).fl.dropna(dim='profile_id').quantile(0.01, dim='depth').reset_coords(drop=True
          spray['fl_deep_ref'] = fl_deep_ref
          del(fl_deep_ref)

          plt.figure(figsize=(15,4))
          plt.hist(spray.fl_deep_ref.dropna(dim='profile_id'), bins=50)

          spray
```

```
Out[247]: <xarray.Dataset>
          Dimensions:            (depth: 100, profile_id: 99752)
          Coordinates:
            * profile_id         (profile_id) int64 20215 20216 20217 20218 20219 ...
              ndive              (profile_id) int64 1 2 3 4 5 6 7 8 9 10 11 12 13 14 ...
              datetime           (profile_id) datetime64[ns] 2006-10-16T19:46:51 ...
              lat                (profile_id) float64 34.35 34.35 34.35 34.34 34.34 ...
              lon                (profile_id) float64 -119.8 -119.8 -119.8 -119.8 ...
              mission_id         (profile_id) int64 106 106 106 106 106 106 106 106 ...
              mission            (profile_id) object '06A00501' '06A00501' '06A00501' ...
              experiment_id      (profile_id) int64 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 ...
              experiment         (profile_id) object 'CUGN_line_80' 'CUGN_line_80' ...
            * depth              (depth) float64 10.0 20.0 30.0 40.0 50.0 60.0 70.0 ...
              days               (profile_id) float64 0.0 0.02537 0.04983 0.07683 ...
          Data variables:
```

```
    temp               (depth, profile_id) float64 16.57 16.38 16.24 16.16 ...
    sal                (depth, profile_id) float64 33.42 33.41 33.43 33.43 ...
    fl                 (depth, profile_id) float64 1.382 1.23 1.429 2.101 ...
    fl_min             (profile_id) float64 0.2884 0.2508 0.2347 0.1904 ...
    profile_max_depth  (profile_id) float64 60.0 80.0 80.0 90.0 110.0 150.0 ...
    fl_deep_ref        (profile_id) float64 nan nan nan nan nan nan nan nan ...
```



3.0.5 Local night time

From longitude estimate the local time as a dt from Grenweeich, so there are no jumps between the timezones but a a continuous time offset. From that I assume daylight between 6 to 18 hrs.

One potential improvement is to consider latitude and period of the year to the estimate the day extension. But for now this should be a good approximation.

```
In [248]: from pandas import Timedelta
```

```
In [249]: local_dt = spray.lon.to_series().apply(lambda x: Timedelta(x/360., 'D')).to_xarray()
          local_time = (spray.datetime - local_dt)
          spray['night_time'] = (local_time.dt.hour < 6) | (local_time.dt.hour > 18)
          del(local_dt)
          del(local_time)
          spray.set_coords('night_time', inplace=True)
          spray
```

```
Out[249]: <xarray.Dataset>
          Dimensions:            (depth: 100, profile_id: 99752)
          Coordinates:
            * profile_id         (profile_id) int64 20215 20216 20217 20218 20219 ...
              ndive              (profile_id) int64 1 2 3 4 5 6 7 8 9 10 11 12 13 14 ...
              datetime           (profile_id) datetime64[ns] 2006-10-16T19:46:51 ...
              lat                (profile_id) float64 34.35 34.35 34.35 34.34 34.34 ...
              lon                (profile_id) float64 -119.8 -119.8 -119.8 -119.8 ...
              mission_id         (profile_id) int64 106 106 106 106 106 106 106 106 ...
              mission            (profile_id) object '06A00501' '06A00501' '06A00501' ...
              experiment_id      (profile_id) int64 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 ...
              experiment         (profile_id) object 'CUGN_line_80' 'CUGN_line_80' ...
```

```
    * depth             (depth) float64 10.0 20.0 30.0 40.0 50.0 60.0 70.0 ...
      days              (profile_id) float64 0.0 0.02537 0.04983 0.07683 ...
      night_time        (profile_id) bool True True True True False False ...
  Data variables:
      temp              (depth, profile_id) float64 16.57 16.38 16.24 16.16 ...
      sal               (depth, profile_id) float64 33.42 33.41 33.43 33.43 ...
      fl                (depth, profile_id) float64 1.382 1.23 1.429 2.101 ...
      fl_min            (profile_id) float64 0.2884 0.2508 0.2347 0.1904 ...
      profile_max_depth (profile_id) float64 60.0 80.0 80.0 90.0 110.0 150.0 ...
      fl_deep_ref       (profile_id) float64 nan nan nan nan nan nan nan nan ...
```

4 Sanity check

Let's check if the loaded data makes sense. An earlier version of the matchup script had a bug that restricted to only one line, so I want to be sure that everything looks good before starting the analysis.

```
In [250]: print("The dataset considered here covers from {0} to {1}".format(
              spray.datetime.min().values,
              spray.datetime.max().values))
```

```
The dataset considered here covers from 2005-04-21T20:29:09.000000000 to 2018-07-02T17:43:21.0
```

It wasn't supposed to have any negative fluorescence.

```
In [251]: tmp = spray.fl.reset_coords(drop=True).to_series().dropna()
          print(tmp[tmp<0])

          spray.sel(profile_id=174796).reset_coords()[['mission', 'profile_id', 'datetime']]
```

```
depth  profile_id
30.0   174796        -0.005143
Name: fl, dtype: float64
```

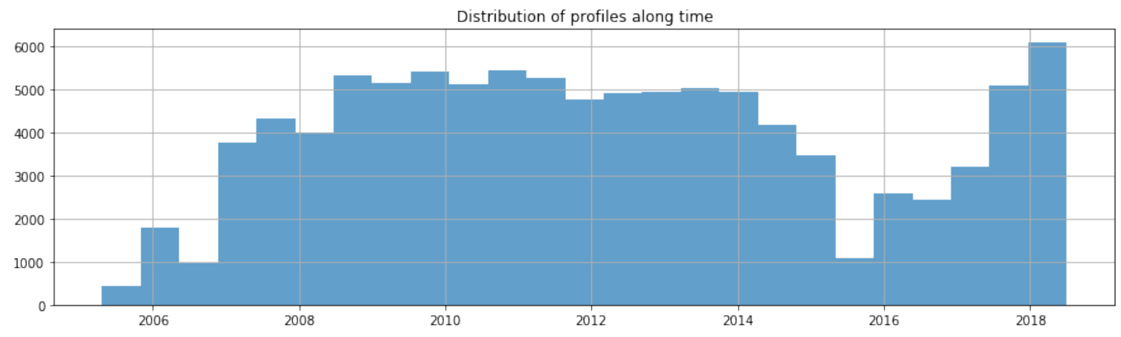
```
Out[251]: <xarray.Dataset>
          Dimensions:     ()
          Data variables:
              mission     <U8 '12602501'
              profile_id  int64 174796
              datetime    datetime64[ns] 2012-07-17T07:18:45
```

4.1 Profiles available along time

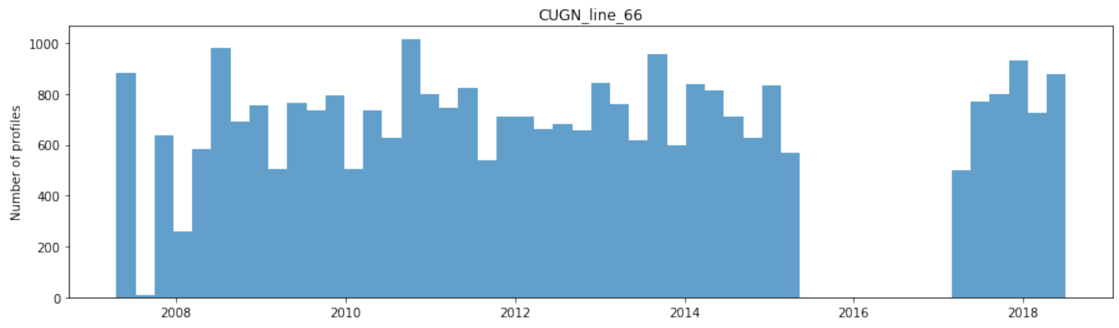
How is the data distribution along the time? Sampling patterns could mislead my conclusions.

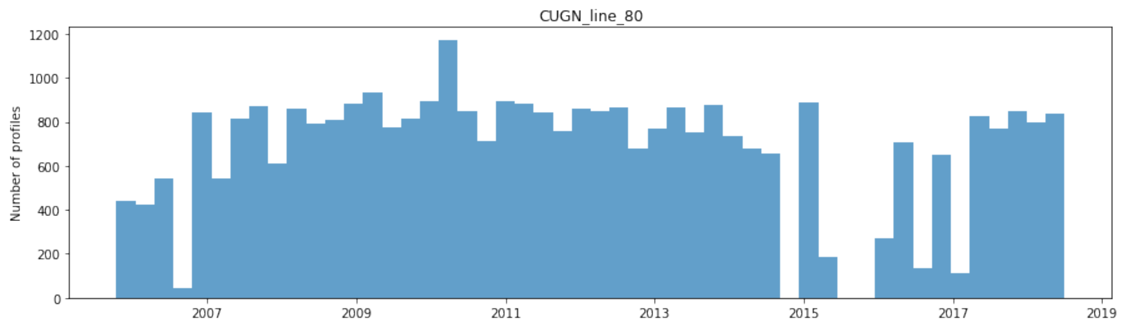
Is it real the lack of data on line 66 during late 2015 and 2016? Note that I only took the missions that had at least one single measurement of fl, therefore, a mission lacking fl sensor or bad sensor would not be shown here.

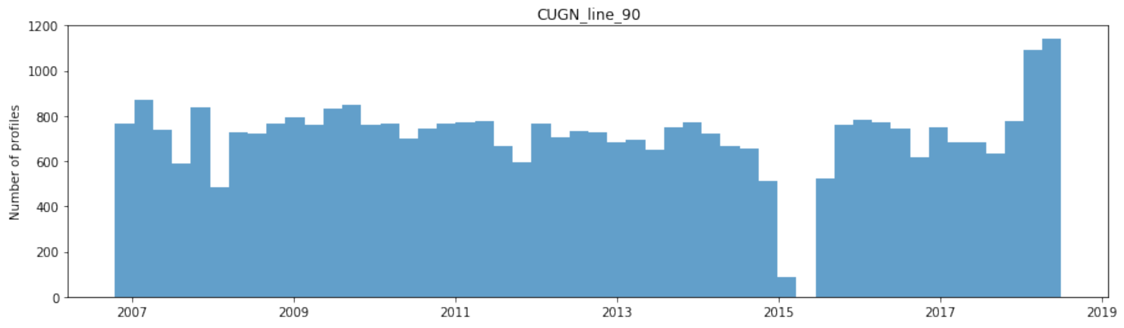
```
In [252]: fig = plt.figure(figsize=(15,4))
          spray.datetime.to_series().hist(bins=25, alpha=0.7)
          noprint = plt.title('Distribution of profiles along time')
```

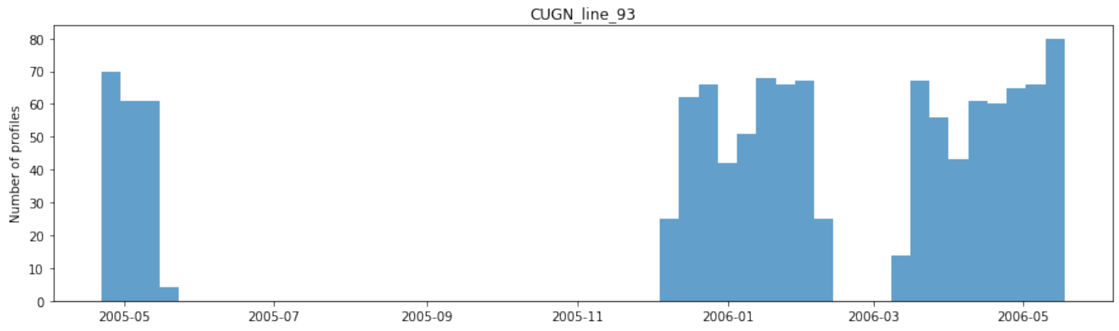


```
In [253]: for experiment_name, grp in spray.groupby('experiment'):
              plt.figure(figsize=(15,4))
              plt.hist(grp.datetime.to_series(), bins=50, alpha=0.7)
              plt.title(experiment_name)
              plt.ylabel('Number of profiles')
```





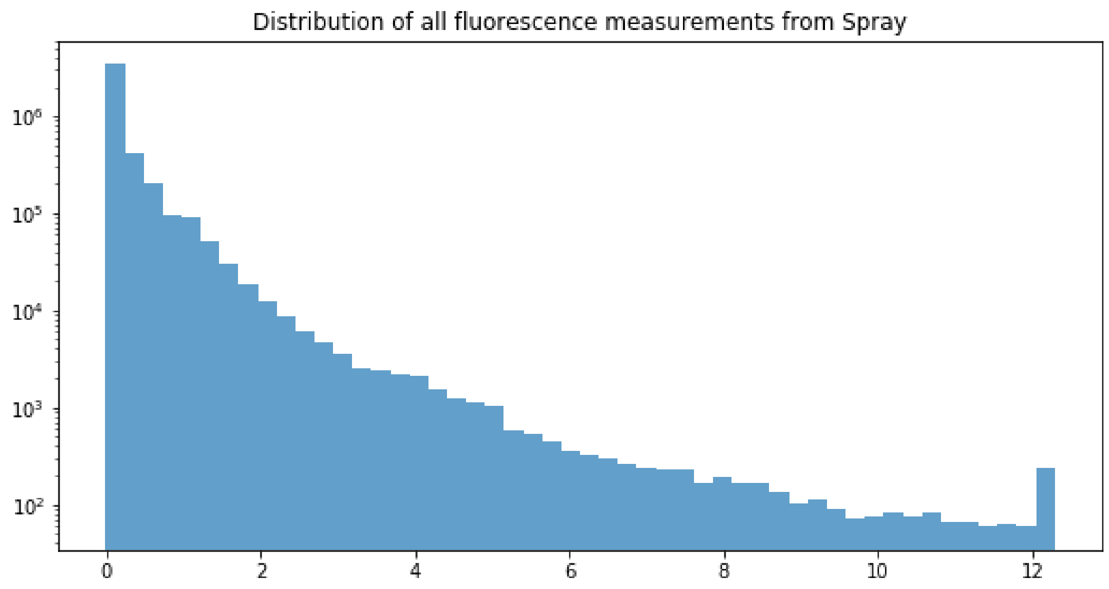




4.2 Distribution of observed fluorescence in all depths

I believe that 12 is the sensor upper limit.

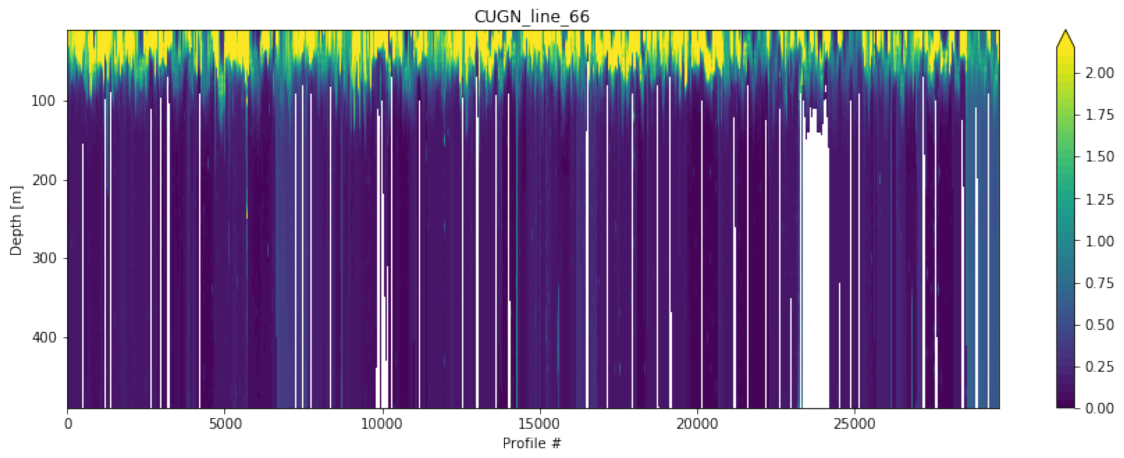
```
In [254]: x = spray.fl.to_masked_array().compressed()
          fig = plt.figure(figsize=(10,5))
          plt.hist(x, 50, alpha=0.7)
          plt.yscale('log', nonposy='clip')
          noprint = plt.title('Distribution of all fluorescence measurements from Spray')
```

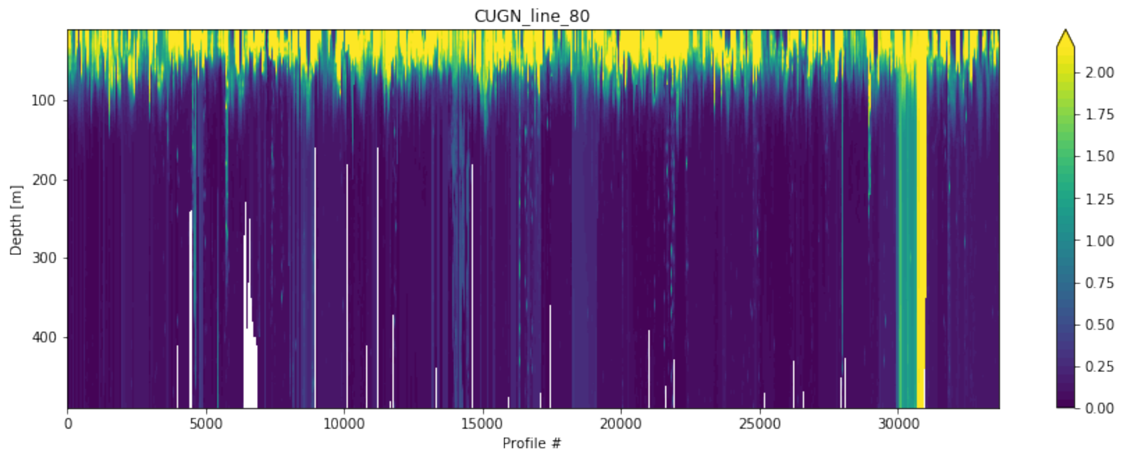


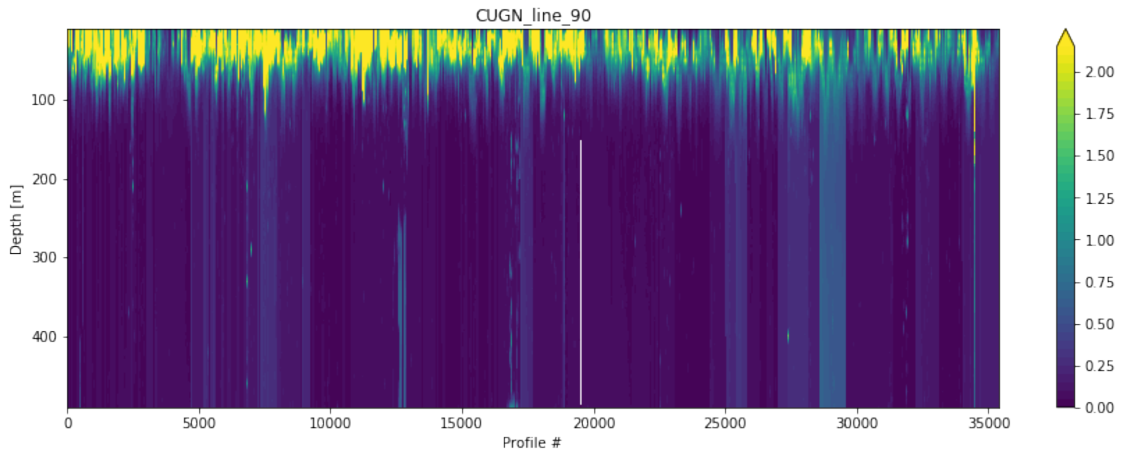
4.2.1 Each line along time

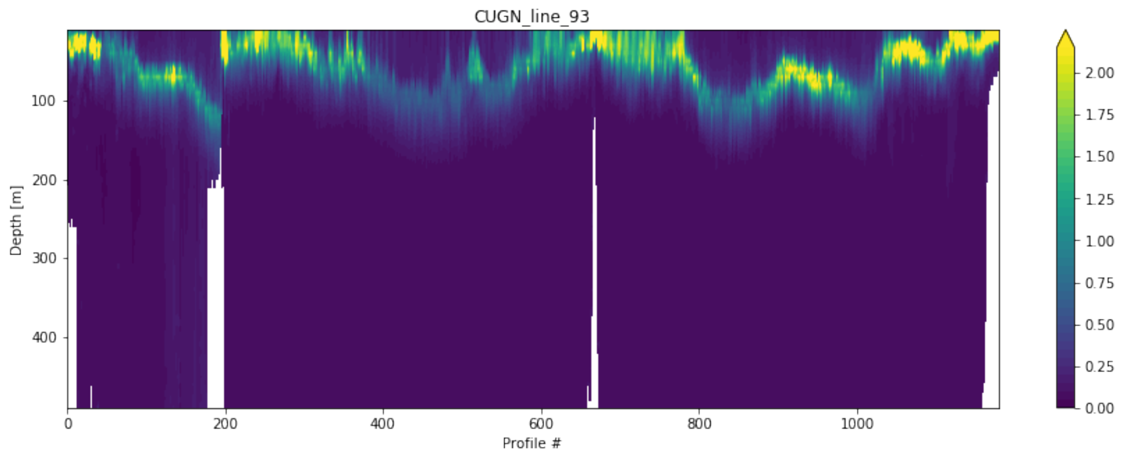
This is the figure that makes explicit the existence of different clusters along the time. Those changes are coincident with the end of missions, thus there is a signal that is not natural but due to the change on sensors. Line 90 is the most explicit, with bands.

```
In [255]: scale = np.arange(0, 2.2, 0.05)
          for experiment_name, grp in spray.isel(depth = spray.depth < 500).groupby('experimen
              plt.figure(figsize=(15,5))
              plt.contourf(range(grp.dims['profile_id']), grp.depth, grp.fl, scale, extend='ma
              #plt.contourf(grp.fl, scale)
              plt.colorbar()
              plt.gca().invert_yaxis()
              plt.title(experiment_name)
              plt.xlabel('Profile #')
              plt.ylabel('Depth [m]')
```


CUGN_line_66
Depth [m]
100
200
300
400
Profile #
0
5000
10000
15000
20000
25000
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00


CUGN_line_80
Depth [m]
100
200
300
400
Profile #
0
5000
10000
15000
20000
25000
30000
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00


CUGN_line_90
Depth [m]
100
200
300
400
Profile #
0
5000
10000
15000
20000
25000
30000
35000
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00

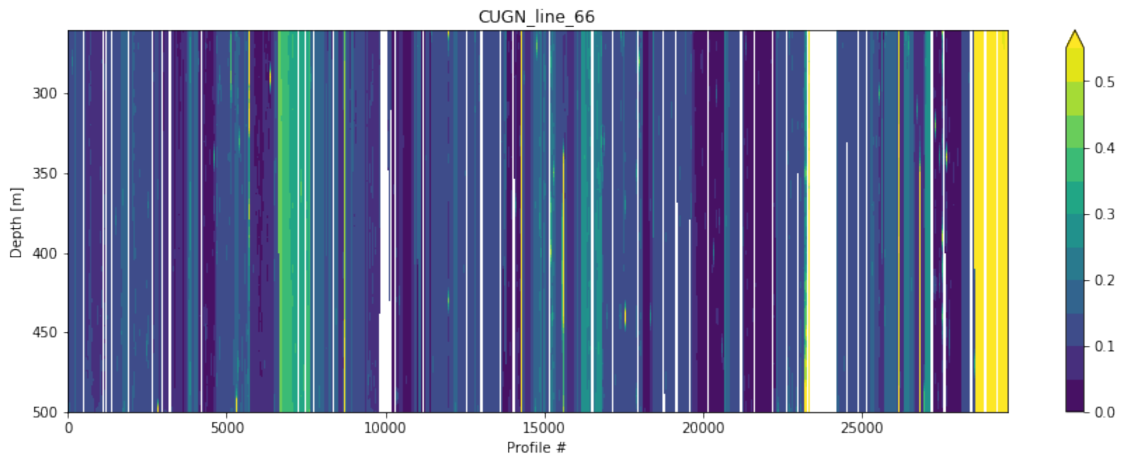


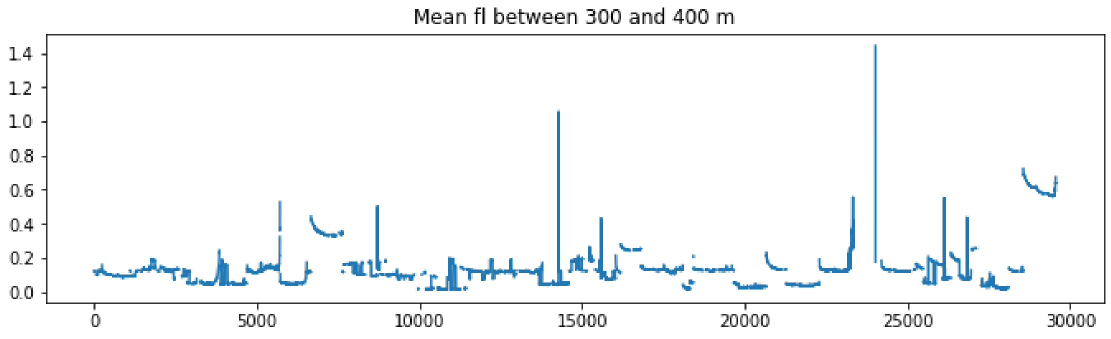
4.2.2 Closer look in the deeper layers

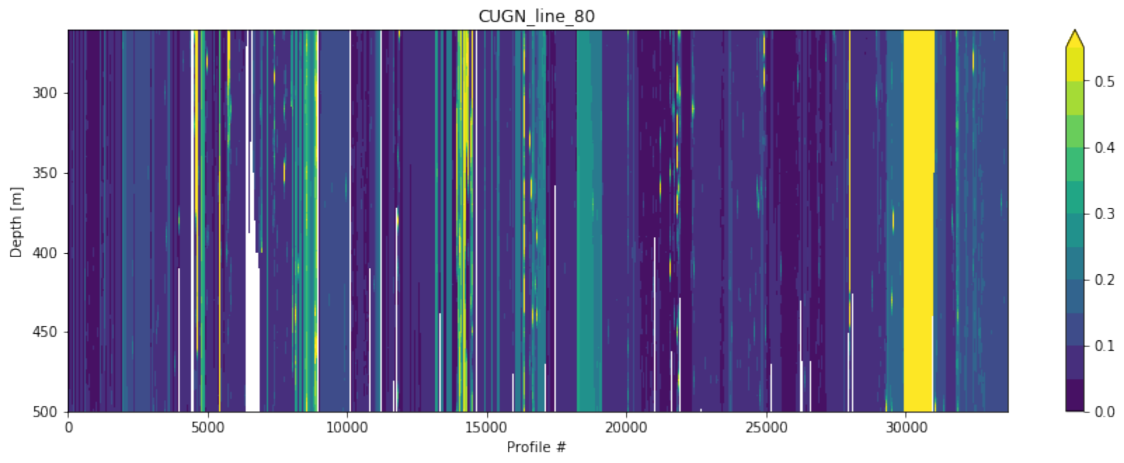
I would expect mostly low values, or at most a low frequency variability.

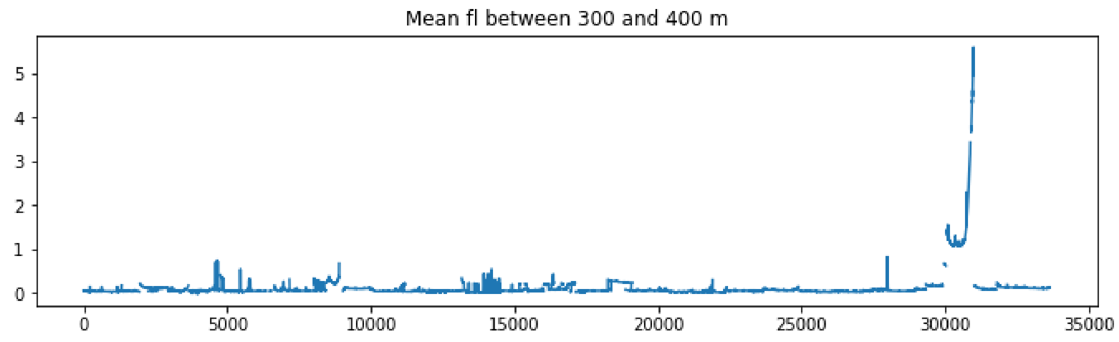
Assuming that deep layers would have the same constant background Chlorophyll concentration, nearly zero, the plots above should be nearly constant in deep layers. We should not be able to distinguish between missions.

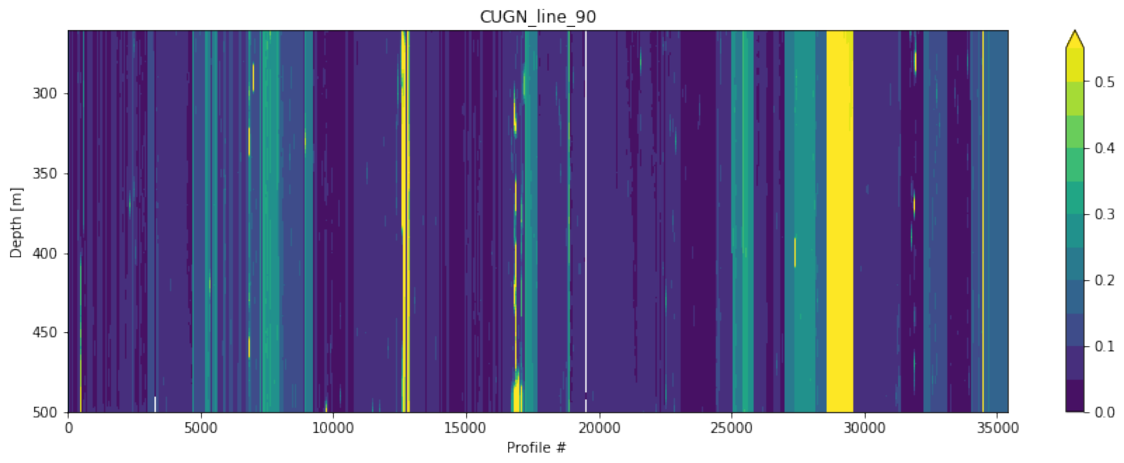
```
In [256]: scale = np.arange(0, 0.6, 0.05)
          for experiment_name, grp in spray.isel(depth = (spray.depth > 250) & (spray.depth <=
              plt.figure(figsize=(15,5))
              plt.contourf(range(grp.dims['profile_id']), grp.depth, grp.fl, scale, extend='ma
              #plt.contourf(grp.fl, scale)
              plt.colorbar()
              plt.gca().invert_yaxis()
              plt.title(experiment_name)
              plt.xlabel('Profile #')
              plt.ylabel('Depth [m]')

              plt.figure(figsize=(11.5,3))
              plt.plot(grp.isel(depth = (grp.depth >= 300) & (grp.depth <= 400))['fl'].mean(di
              plt.title('Mean fl between 300 and 400 m')
```

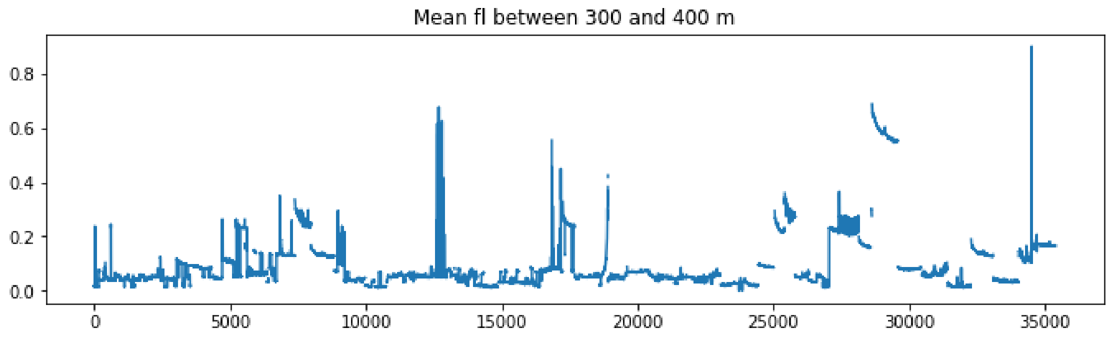

CUGN_line_66
Depth [m]
Profile #
300
350
400
450
500
0
5000
10000
15000
20000
25000
0.0
0.1
0.2
0.3
0.4
0.5

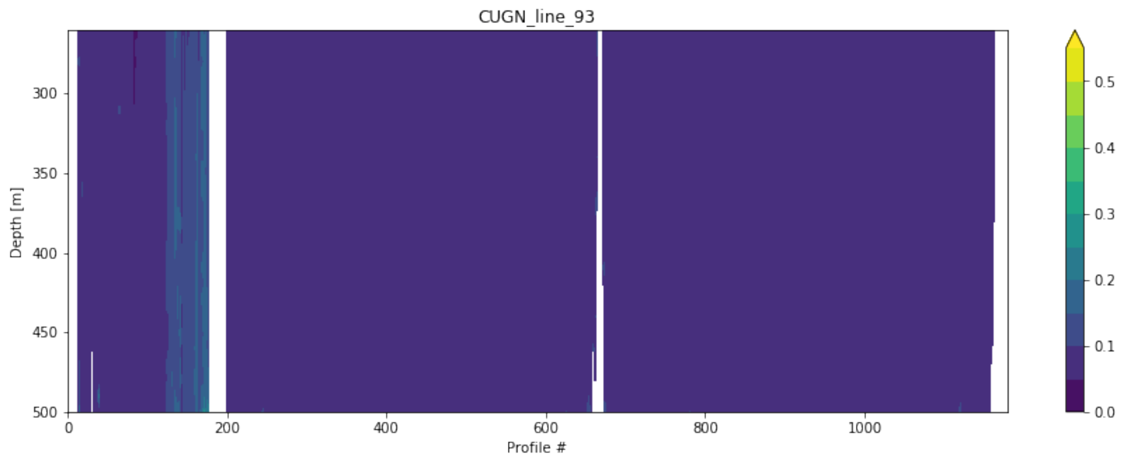

Mean fl between 300 and 400 m
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
0
5000
10000
15000
20000
25000
30000

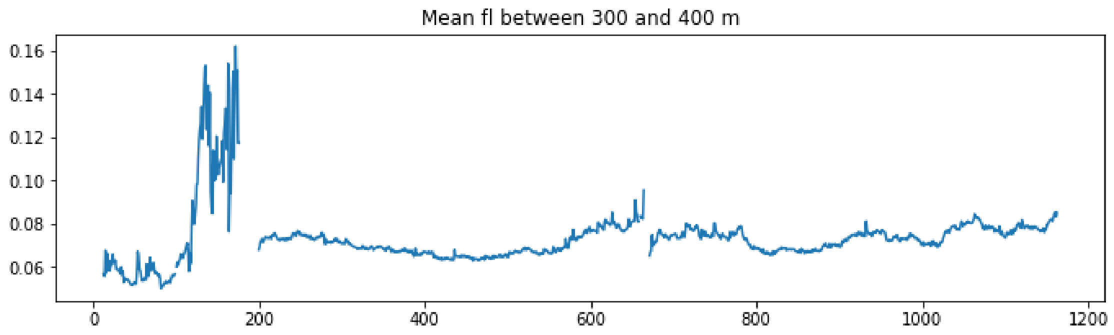

CUGN_line_80
Depth [m]
Profile #
300
350
400
450
500
0
5000
10000
15000
20000
25000
30000
0.0
0.1
0.2
0.3
0.4
0.5


Mean fl between 300 and 400 m
5
4
3
2
1
0
0
5000
10000
15000
20000
25000
30000
35000


CUGN_line_90
Depth [m]
300
350
400
450
500
0
5000
10000
15000
20000
25000
30000
35000
Profile #
0.5
0.4
0.3
0.2
0.1
0.0


Mean fl between 300 and 400 m
0.8
0.6
0.4
0.2
0.0
0
5000
10000
15000
20000
25000
30000
35000





5 Quality Control

Some missions have a distinguishing high chlorophyll signal in the deep layers. I don't know if that's a real signal or bad sensor. For this first analysis I'll discart these ones with a simple criteria.

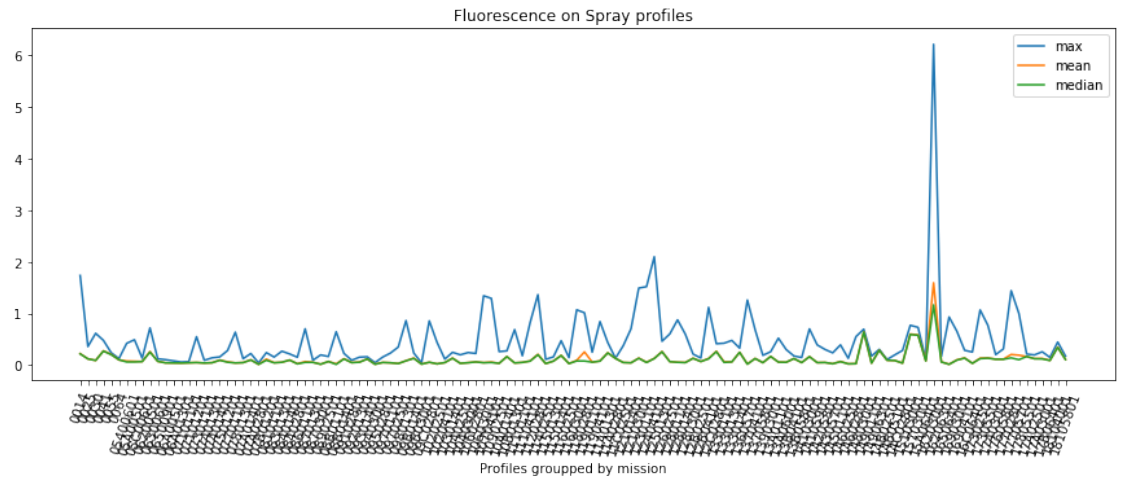
5.0.1 Are there missions with persistent strong bias?

Originally I used fl_min as criteria (minimum fluorescence per profile), but later I found fl_min might be questionable. Well, I dont' need fl_min, so better use fl aggregated in the deep layer . . . .

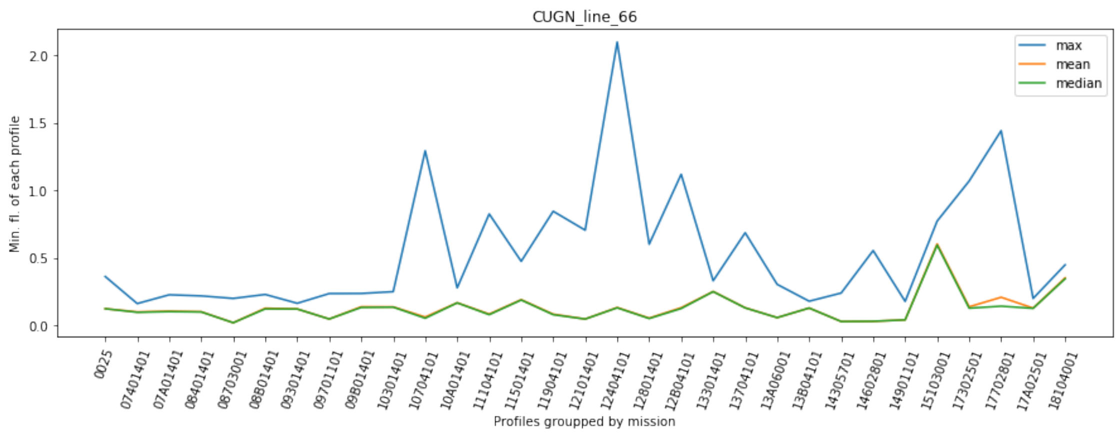
```
In [257]: # Groupping dataset by mission, so that apply procedures for each mission subset.

          idx = (spray.depth >= 250) & (spray.depth <= 500)
          # Confirming the depth levels considered.
          print(spray.isel(depth=idx).depth)

          spray_deep_layer = spray.isel(depth=idx)
```

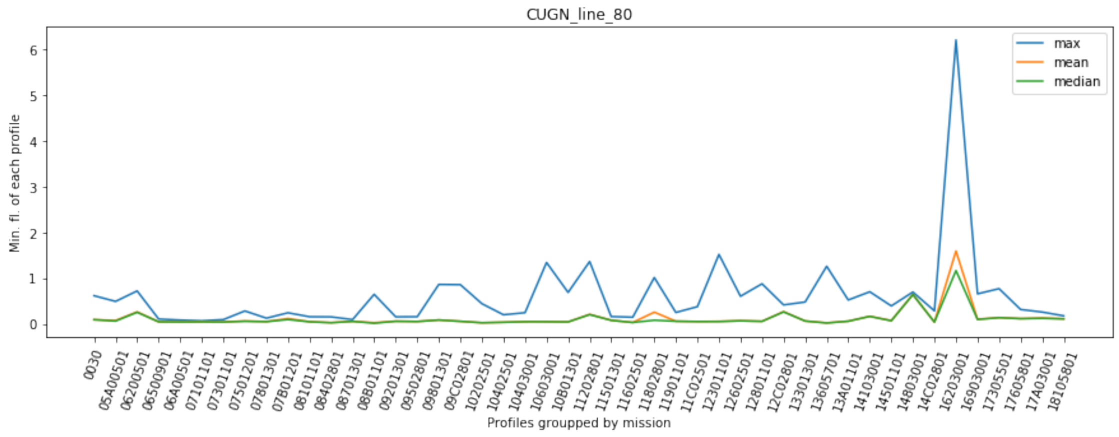
```
         grp = spray_deep_layer.groupby('mission')

         grp_mission = grp.first()['mission']

         plt.figure(figsize=(15, 5))
         plt.plot(grp_mission, grp.max()['fl'], label='max')
         plt.plot(grp_mission, grp.mean()['fl'], label='mean')
         plt.plot(grp_mission, grp.median()['fl'], label='median')

         plt.title('Fluorescence on Spray profiles')
         plt.xlabel('Profiles groupped by mission')
         plt.legend()
         ticks = plt.xticks(rotation=70)
```

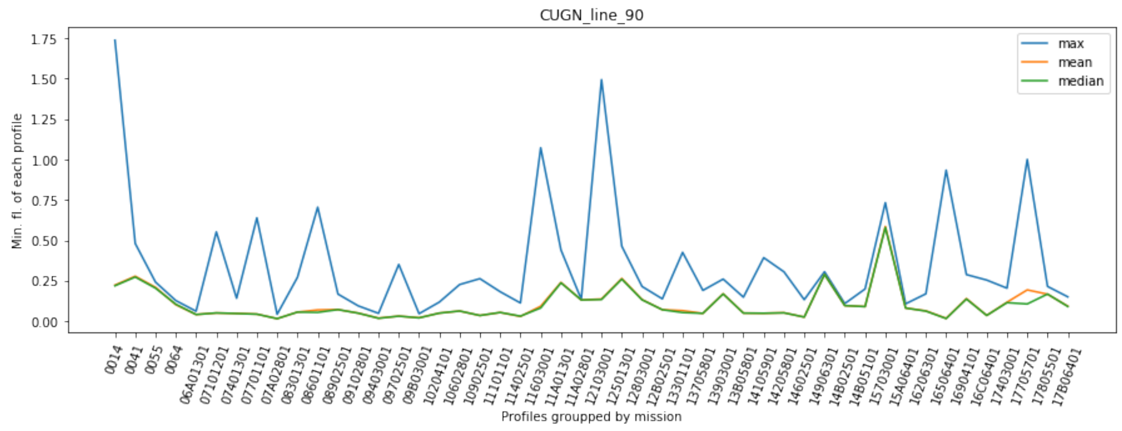
```
<xarray.DataArray 'depth' (depth: 26)>
array([250., 260., 270., 280., 290., 300., 310., 320., 330., 340., 350., 360.,
       370., 380., 390., 400., 410., 420., 430., 440., 450., 460., 470., 480.,
       490., 500.])
Coordinates:
  * depth    (depth) float64 250.0 260.0 270.0 280.0 290.0 300.0 310.0 320.0 ...
```

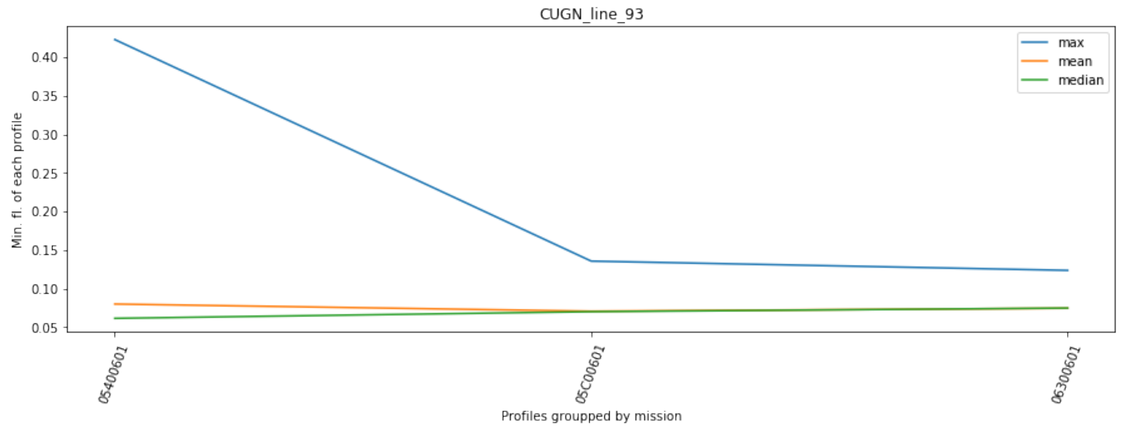


```
In [258]: # for experiment_name, experiment in spray.groupby('experiment'):
          for experiment_name, experiment in spray_deep_layer.groupby('experiment'):
              grp = experiment.groupby('mission')

              grp_mission = grp.first()['mission']

              plt.figure(figsize=(15, 4.5))
              plt.plot(grp_mission, grp.max()['fl'], label='max')
              plt.plot(grp_mission, grp.mean()['fl'], label='mean')
              plt.plot(grp_mission, grp.median()['fl'], label='median')

              plt.title(experiment_name)
              plt.xlabel('Profiles groupped by mission')
              plt.ylabel('Min. fl. of each profile')
              plt.legend()
              ticks = plt.xticks(rotation=70)
```









5.0.2 Removing missions with strong and consistent bias

Looks like most of the missions are fine. But a few missions have minimum persistently too high. I'll remove completely this mission for now.

ATENTION!!! I might want to go back here. Are these missions really bad measurements or an special event?

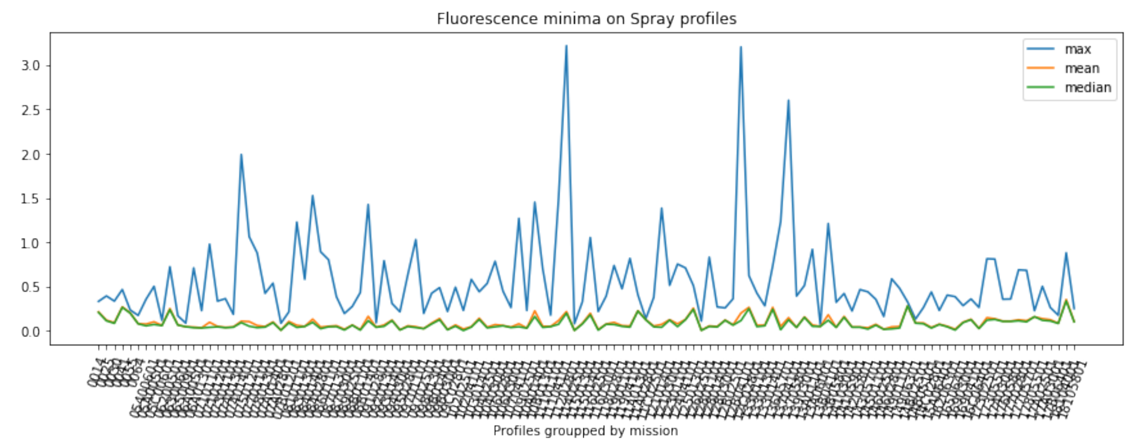
```
In [259]: grp = spray_deep_layer.groupby('mission')
          grp_mission = grp.first()['mission']
          compromised_missions = (grp_mission[grp.median()['fl'] > 0.5]).to_series()
          print("Removing missions: {0}".format(list(compromised_missions)))
          print("Original number of profiles: ", spray.dims['profile_id'])
          idx = [m in compromised_missions for m in spray.mission.to_series()]
          spray = spray.drop(spray.profile_id[idx], dim='profile_id')
          print("Profiles after cleanned: ", spray.dims['profile_id'])
          del(spray_deep_layer)
          spray
```

```
Removing missions: ['14803001', '15103001', '15703001', '16203001']
Original number of profiles:  99752
Profiles after cleanned:  96718
```

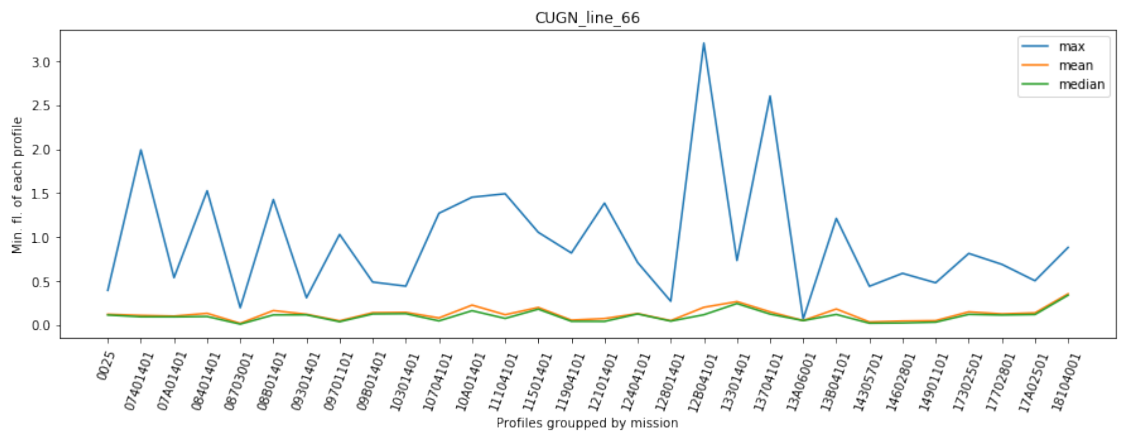
```
Out[259]: <xarray.Dataset>
          Dimensions:            (depth: 100, profile_id: 96718)
          Coordinates:
            * profile_id         (profile_id) int64 20215 20216 20217 20218 20219 ...
              ndive              (profile_id) int64 1 2 3 4 5 6 7 8 9 10 11 12 13 14 ...
              datetime           (profile_id) datetime64[ns] 2006-10-16T19:46:51 ...
              lat                (profile_id) float64 34.35 34.35 34.35 34.34 34.34 ...
              lon                (profile_id) float64 -119.8 -119.8 -119.8 -119.8 ...
              mission_id         (profile_id) int64 106 106 106 106 106 106 106 106 ...
              mission            (profile_id) object '06A00501' '06A00501' '06A00501' ...
              experiment_id      (profile_id) int64 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 ...
              experiment         (profile_id) object 'CUGN_line_80' 'CUGN_line_80' ...
            * depth              (depth) float64 10.0 20.0 30.0 40.0 50.0 60.0 70.0 ...
              days               (profile_id) float64 0.0 0.02537 0.04983 0.07683 ...
              night_time         (profile_id) bool True True True True False False ...
          Data variables:
              temp               (depth, profile_id) float64 16.57 16.38 16.24 16.16 ...
              sal                (depth, profile_id) float64 33.42 33.41 33.43 33.43 ...
              fl                 (depth, profile_id) float64 1.382 1.23 1.429 2.101 ...
              fl_min             (profile_id) float64 0.2884 0.2508 0.2347 0.1904 ...
              profile_max_depth  (profile_id) float64 60.0 80.0 80.0 90.0 110.0 150.0 ...
              fl_deep_ref        (profile_id) float64 nan nan nan nan nan nan nan nan ...
```

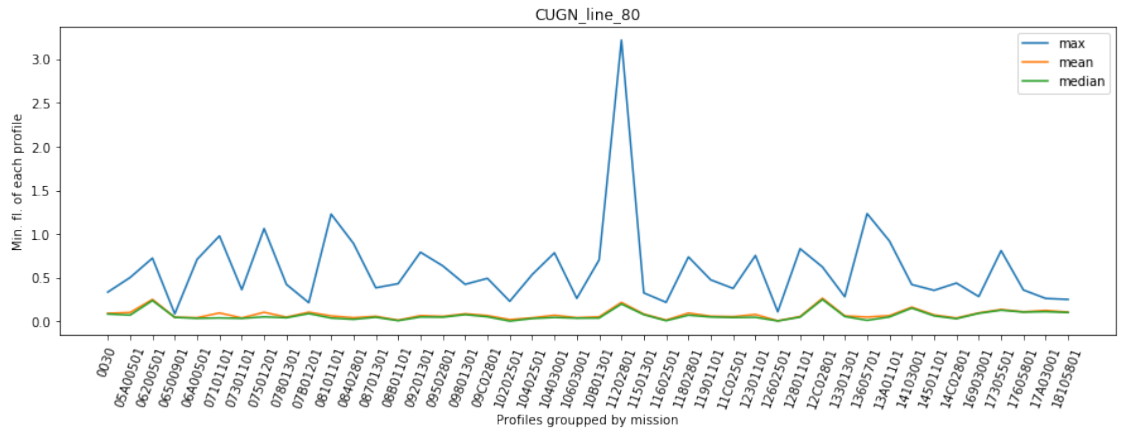
5.0.3 Reviewing plots without bad missions

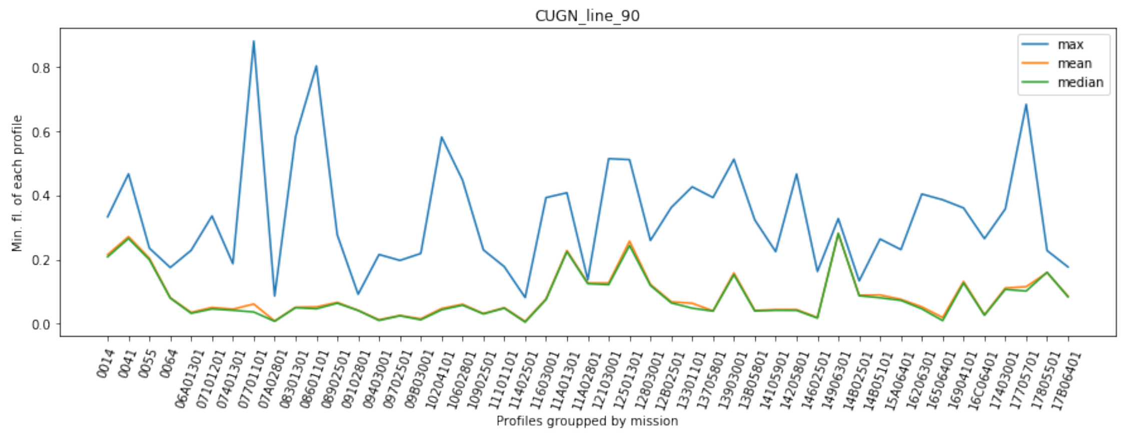
```
In [260]: # Update groupping to reflect removed missions
          idx = (spray.depth >= 250) & (spray.depth <= 500)
          spray_deep_layer = spray.isel(depth=idx)
          grp = spray_deep_layer.groupby('mission')

          grp_mission = grp.first()['mission']

          plt.figure(figsize=(15, 4.5))
          plt.plot(grp_mission, grp.max()['fl_min'], label='max')
          plt.plot(grp_mission, grp.mean()['fl_min'], label='mean')
          plt.plot(grp_mission, grp.median()['fl_min'], label='median')

          plt.title('Fluorescence minima on Spray profiles')
          plt.xlabel('Profiles groupped by mission')
          plt.legend()
          ticks = plt.xticks(rotation=70)
```

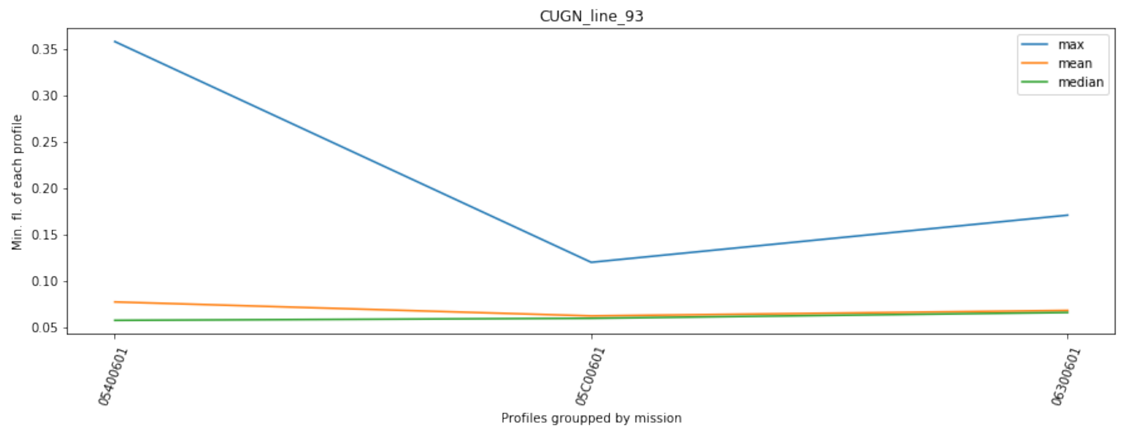


```
In [261]: for experiment_name, experiment in spray_deep_layer.groupby('experiment'):
              grp = experiment.groupby('mission')

              grp_mission = grp.first()['mission']

              plt.figure(figsize=(15, 4.5))
              plt.plot(grp_mission, grp.max()['fl_min'], label='max')
              plt.plot(grp_mission, grp.mean()['fl_min'], label='mean')
              plt.plot(grp_mission, grp.median()['fl_min'], label='median')

              plt.title(experiment_name)
              plt.xlabel('Profiles groupped by mission')
              plt.ylabel('Min. fl. of each profile')
              plt.legend()
              ticks = plt.xticks(rotation=70)
```




CUGN_line_80
max
mean
median
Min. fl. of each profile
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0030
05A00501
06200501
06500901
06A00501
07101101
07301101
07501201
07801301
07B01201
08101101
08402801
08701301
08B01101
09201301
09502801
09801301
09C02801
10202501
10402501
10403001
10603001
10B01301
11202801
11501301
11602501
11802801
11901101
11C02501
12301101
12602501
12801101
12C02801
13301301
13605701
13A01101
14103001
14501101
14C02801
16903001
17305501
17605801
17A03001
18105801
Profiles groupped by mission


CUGN_line_90
max
mean
median
Min. fl. of each profile
0.8
0.6
0.4
0.2
0.0
0014
0041
0055
0064
06A01301
07101201
07401301
07701101
07A02801
08301301
08601101
08902501
09102801
09403001
09702501
09B03001
10204101
10602801
10902501
11101101
11402501
11603001
11A01301
11A02801
12103001
12501301
12803001
12B02501
13301101
13705801
13903001
13B05801
14105901
14205801
14602501
14906301
14B02501
14B05101
15A06401
16206301
16506401
16904101
16C06401
17403001
17705701
17805501
17B06401
Profiles groupped by mission


CUGN_line_93
max
mean
median
Min. fl. of each profile
0.35
0.30
0.25
0.20
0.15
0.10
0.05
05400601
05C00601
06300601
Profiles groupped by mission

6 Vertical structure

Most of the fl signal happens in the upper 100m, as expected.

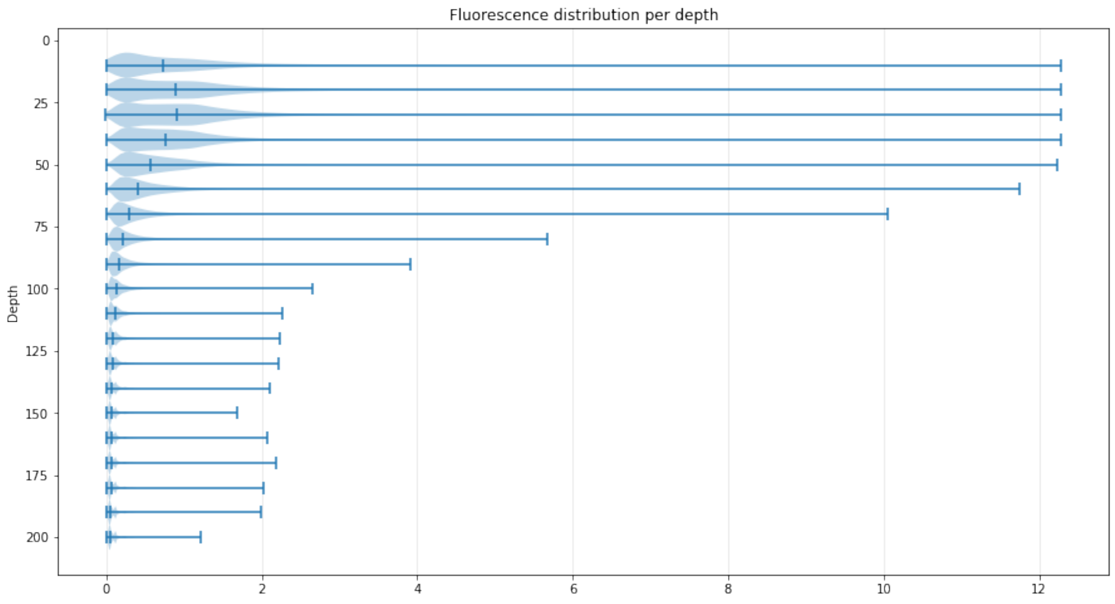
Assuming that deep layers would have the same constant background Chlorophyll concentration, nearly zero, the fluorescence should be nearly constant in deep layers. We should not be able to distinguish between missions.

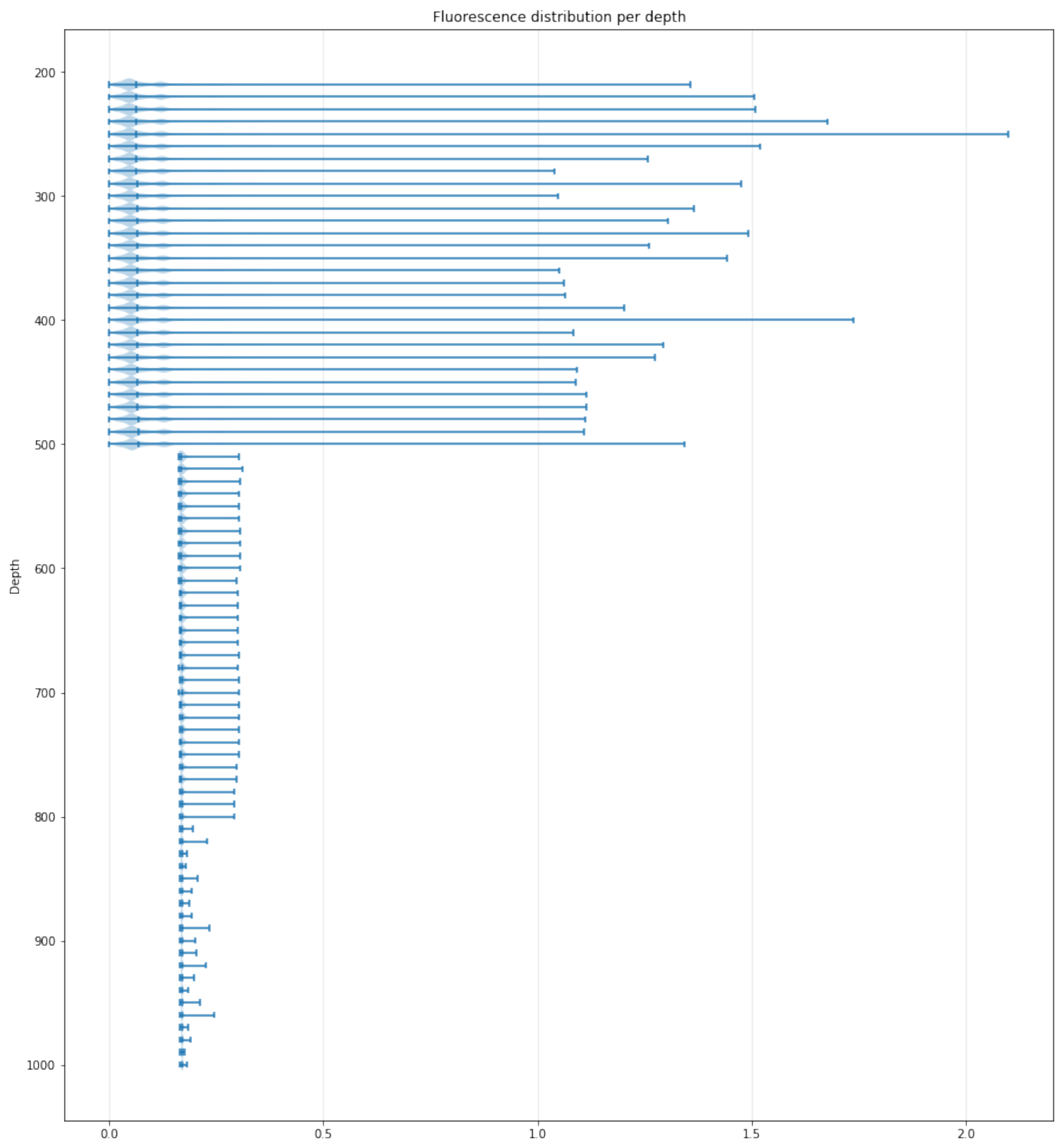
There is something different on the missions that made below 1000m.

Notes: - There is no level where zero fl dominates, even the deep measurements show some fluorescence.

- Any measurement shallower than 100 m will probably not have a neglibible fluorescence, thus shallow profiles can't be used to identify the offset. Below 200m might be a good threshold.
- Are the few high values below 200 m real?

```
In [25]: # depth = spray.depth.to_series().tolist()
         # data = [spray.sel(depth=d).fl.to_series().dropna().tolist() for d in depth]

         fig, ax = plt.subplots(nrows=1, ncols=1, figsize=(15, 8))

         ax.xaxis.grid(True, alpha=0.3)
         depth = spray.isel(depth=spray.depth <= 200).depth.to_series().tolist()
         data = [spray.sel(depth=d).fl.to_series().dropna().tolist() for d in depth]
         # plot violin plot
         ax.violinplot(data, depth,
                       widths=10,
                       points=200,
                       vert=False,
                       showmeans=False,
                       showextrema=True,
                       showmedians=True)

         ax.set_title('Fluorescence distribution per depth')
         ax.set_ylabel('Depth')
         ax.invert_yaxis()


         fig, ax = plt.subplots(nrows=1, ncols=1, figsize=(15, 17))

         ax.xaxis.grid(True, alpha=0.3)
         depth = spray.isel(depth=spray.depth > 200).depth.to_series().tolist()
         data = [spray.sel(depth=d).fl.to_series().dropna().tolist() for d in depth]
         # plot violin plot
         ax.violinplot(data, depth,
                       widths=10,
                       points=200,
                       vert=False,
                       showmeans=False,
                       showextrema=True,
                       showmedians=True)

         ax.set_title('Fluorescence distribution per depth')
         ax.set_ylabel('Depth')
         ax.invert_yaxis()


         del(depth)
         del(data)
```





6.0.1 Does shallow waters misguide the minimum fluorescence?

As suggested by the global violin plots, shallow profiles always detect some chlorophyll, thus those must not be used to determine the mission offset.

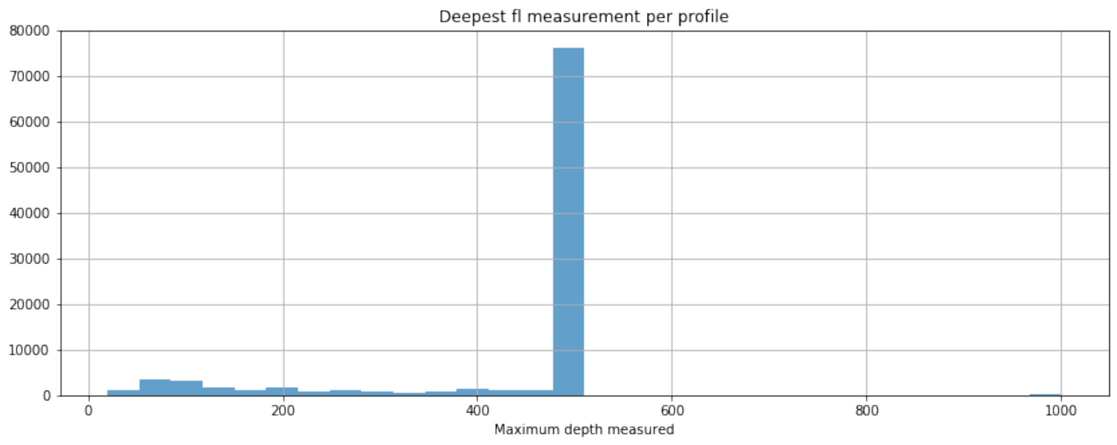
What is the distribution of maximum profile depth? Most of the measurements are from 500 m dives, so even removing shallow dives leaves sufficient data to estimate minimum florescence.

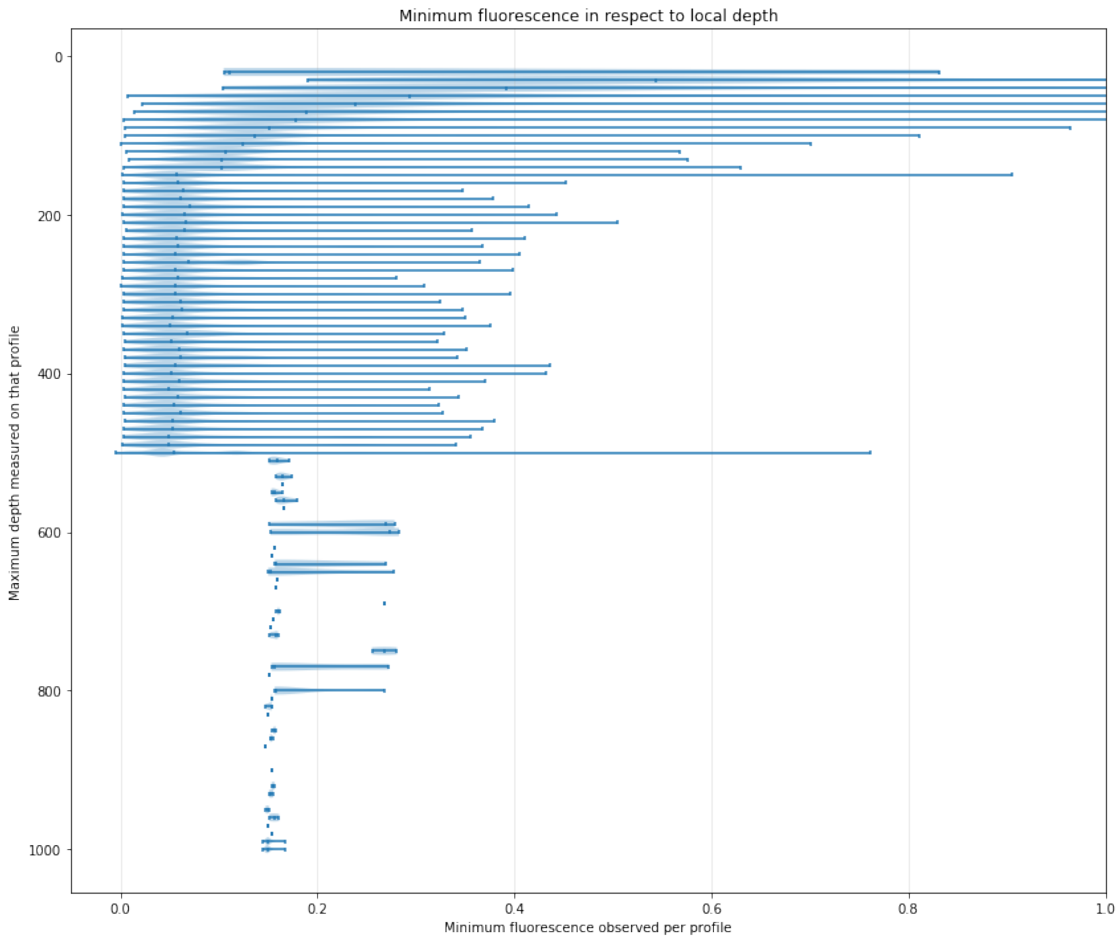
There is a large step on the median between 140 and 150 m (lower figure). A more conservative approach might be to use only profiles with at least 200 m measurements. The top figure confirms that shallow stations are minority, thus restrict to deep stations to find the offset will still leave plenty profiles for statistical robustness.

There is a dependency on the cast vertical extension (probably local depth), thus **minimum fluorescence from shallow dives, probably near the coast and more primary activity, should be removed and not considered to estimate fl_0.**

6.0.2 Conclusion: Do not use fl_min from shallow dives!!!

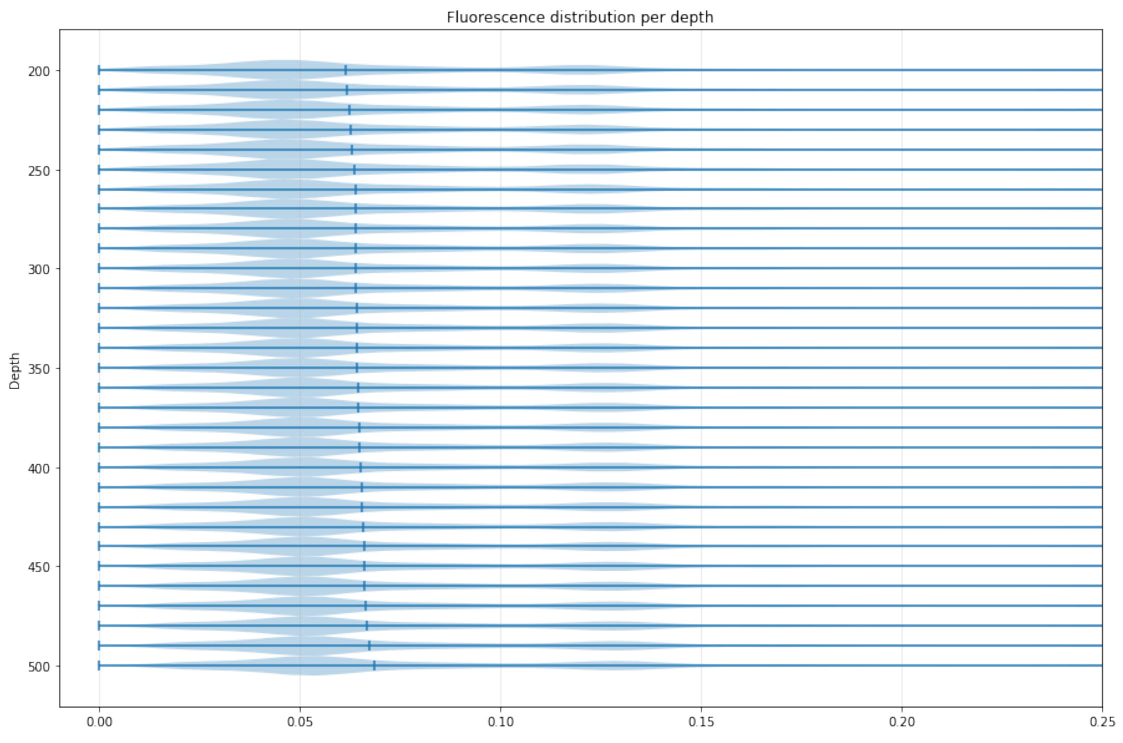
```
In [26]: # Only when fl measurement was valid
         data = spray.fl.to_series().dropna().reset_index()

         # For each profile get the depth of the deepest fl measurement, and the minimum fl
         depth_max = data.groupby('profile_id')['depth'].max()
         fl_min = data.groupby('profile_id')['fl'].min()


         data = pd.concat([depth_max, fl_min], axis='columns')

         # ==== Profile depth distribution
         plt.figure(figsize=(14,5))
         data.depth.hist(bins=30, alpha=0.7)
         plt.title('Deepest fl measurement per profile')
         plt.xlabel('Maximum depth measured')

         # ==== Distribution of minimum fl in respect to maximum depth of that profile
         depth = sorted(data.depth.unique())
         data = [data[data.depth == d]['fl'].tolist() for d in depth]


         fig, ax = plt.subplots(nrows=1, ncols=1, figsize=(14, 12))

         ax.xaxis.grid(True, alpha=0.3)

         # plot violin plot
         ax.violinplot(data, depth,
                       widths=10,
                       points=200,
                       vert=False,
                       showmeans=False,
                       showextrema=True,
                       showmedians=True)

         ax.set_title('Minimum fluorescence in respect to local depth')
         ax.set_xlabel('Minimum fluorescence observed per profile')
         ax.set_ylabel('Maximum depth measured on that profile')
         ax.invert_yaxis()
         ax.set_xlim(-0.05, 1)

         del(data)
         del(fl_min)
```





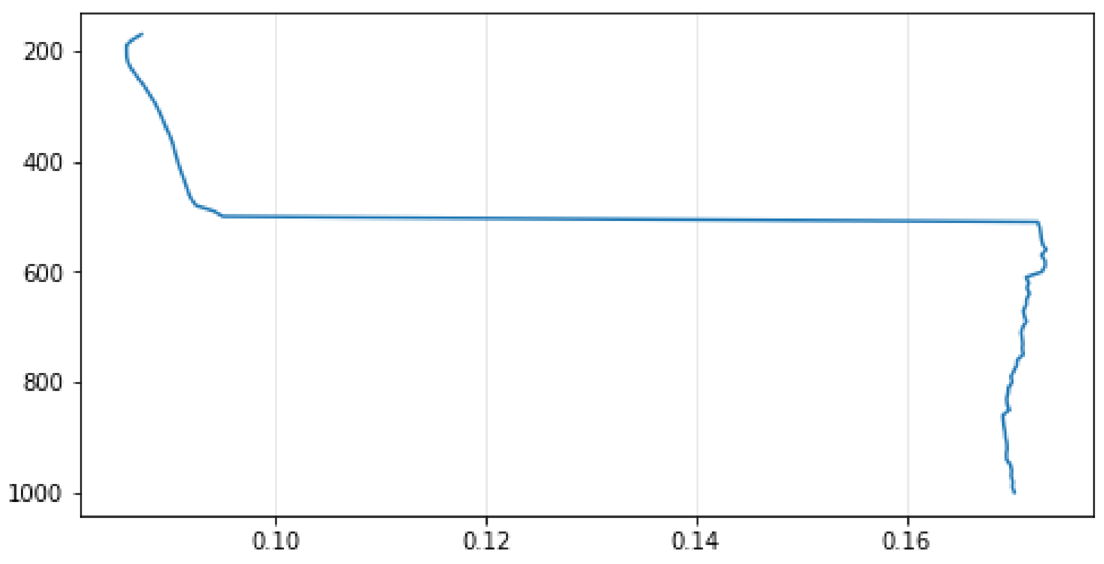
ATENTION: Some missions are clearly different, when depth show higher values. But it doesn't look like as a simple offset. Build a PDF of fl observed by each mission. Could I cluster the missions quality from the PDF?

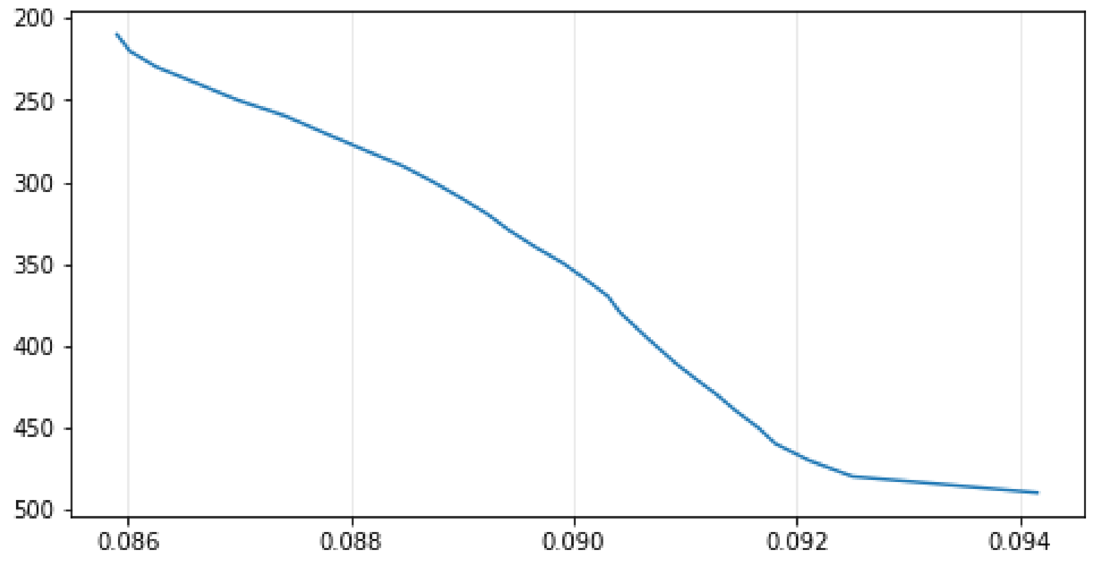
```
In [27]: depth = spray.isel(depth = (spray.depth >= 200) & (spray.depth <= 500)).depth.to_seri
         data = [spray.sel(depth=d).fl.to_series().dropna().tolist() for d in depth]

         fig, ax = plt.subplots(nrows=1, ncols=1, figsize=(15, 10))

         ax.xaxis.grid(True, alpha=0.3)

         # plot violin plot
         ax.violinplot(data, depth,
                       widths=10,
                       points=200,
                       vert=False,
                       showmeans=False,
                       showextrema=True,
                       showmedians=True)

         ax.set_title('Fluorescence distribution per depth')
         ax.set_ylabel('Depth')
         ax.invert_yaxis()

         noprint = plt.xlim(xmin=-0.01, xmax=0.25)
```



```
In [28]: fig, ax = plt.subplots(nrows=2, ncols=1, figsize=(8, 9))
```

```
layer = spray.isel(depth = (spray.depth > 160))
ax[0].xaxis.grid(True, alpha=0.3)
#depth = spray_deep.depth.to_series().tolist()
ax[0].plot(layer.fl.mean(dim='profile_id'), layer.depth)
#ax[0].plot([np.ma.array(d).mean() for d in data], depth)
ax[0].invert_yaxis()


layer = spray.isel(depth = (spray.depth > 200) & (spray.depth < 500))
ax[1].xaxis.grid(True, alpha=0.3)
ax[1].plot(layer.fl.mean(dim='profile_id'), layer.depth)
ax[1].invert_yaxis()
ax[1].xaxis

del(layer)
```





There is a tendency of increasing fl with depth in the deep measurements. That is not necessarily associated with pressue, it could be with temperature. Independent of what it is, it's on the order of 0.1 in 300 m, so I'll neglect it for now. First, first order error, later I try these improvements.

7 Working on the offset (bias)

The fluorescence profile from Spray can lack any zero measurement. That could be due to particulates other than Chlorophyll or a bias in the sensor reading. In any case it is necessary to find this offset.

ADD AN EXAMPLE PROFILE SHOWING THE SUBSURFACE MAXIMA AND A MINIMA THAT IS NOT NECESSARILY IN THE BOTTOM

7.1 General statistics on minima fluorescence of each profile

```
In [29]: # Work in a robust estimate of minimum fl. Probably the average of the three lowest m

         def robust_min(x):
             y = x.dropna(dim='depth')
             if len(y) == 0:
                 return x[0]
             else:
                 y = y.sortby(y)
                 return y[:3]
             return x.sortby(x, ascending=False)[0]
             return np.mean(pd.Series(x).dropna()[:3])
```

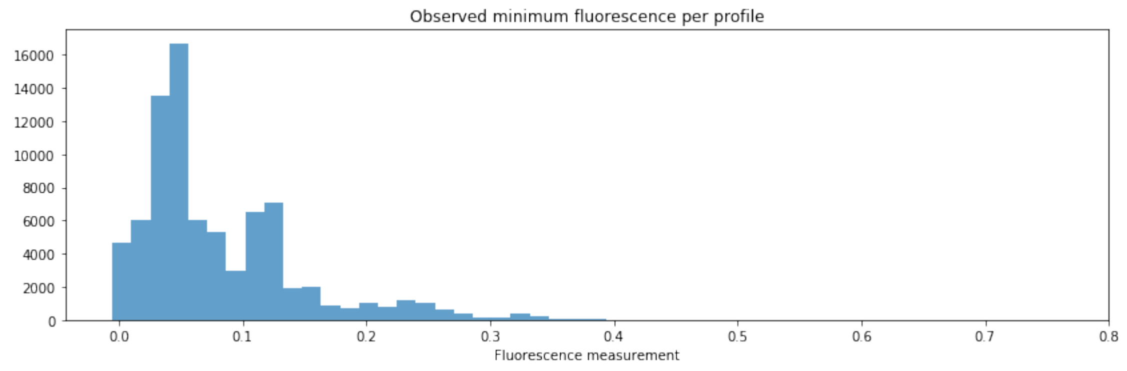
```
In [30]: #spray_deep['fl_min_rbst'] = np.nan * spray_deep.fl_min
         #for profile_id, grp in spray_deep.groupby('profile_id'):
         #    print(profile_id)
         #    np.mean(grp.fl.to_series().dropna()[:3])
```

```
In [31]: # spray_deep = spray.isel(depth=spray.depth >= 200)
         spray_offshore = spray.isel(profile_id=spray.profile_max_depth >= 350)

         print('Shallowest measurement from spray_deep: ', float(spray_offshore.depth.min()))
```

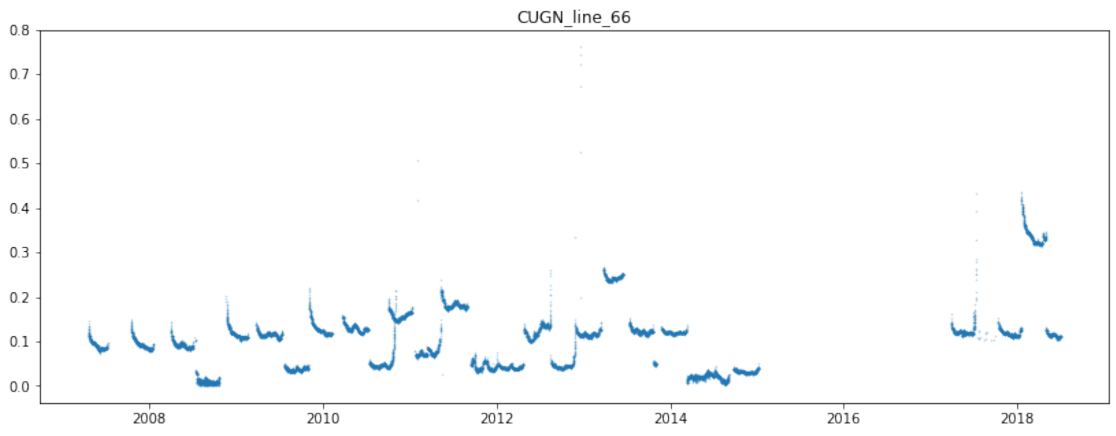
```
Shallowest measurement from spray_deep:  10.0
```

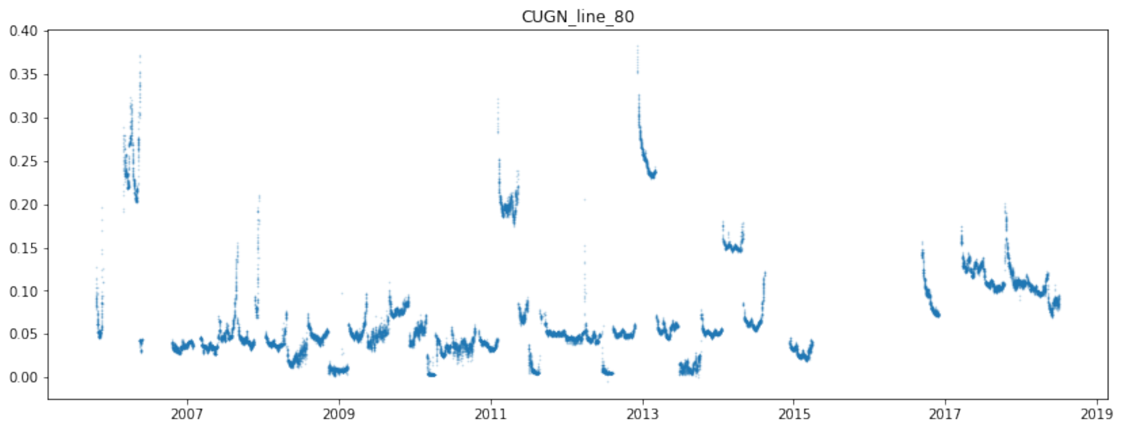
```
In [32]: plt.figure(figsize=(14,4))
         h = plt.hist(spray_offshore.fl_min.to_series().dropna(), bins=50, alpha=0.7)
         plt.title('Observed minimum fluorescence per profile')
         plt.xlabel('Fluorescence measurement')

         spray_offshore.fl_min.to_series().describe()
```

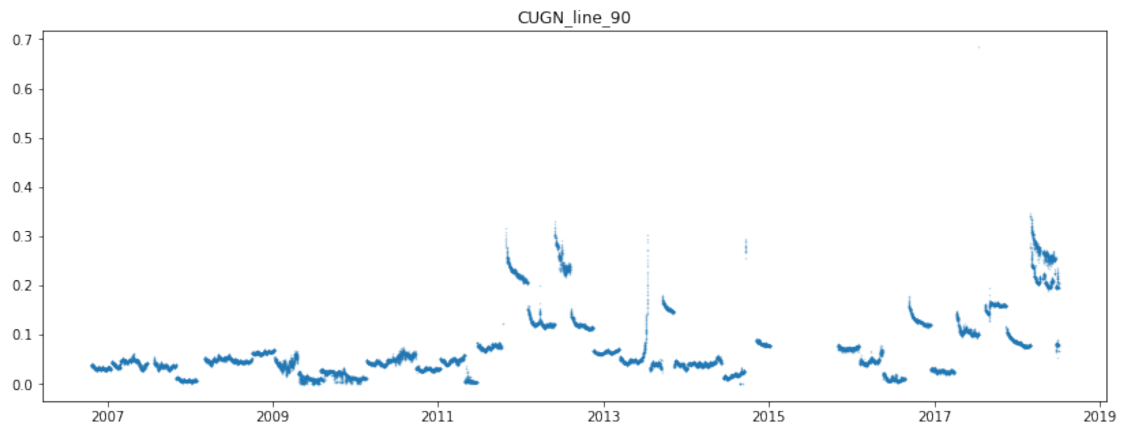
```
Out[32]: count    80766.000000
         mean         0.080905
         std          0.066075
         min         -0.005143
         25%          0.037500
         50%          0.054818
         75%          0.115333
         max          0.761333
         Name: fl_min, dtype: float64
```

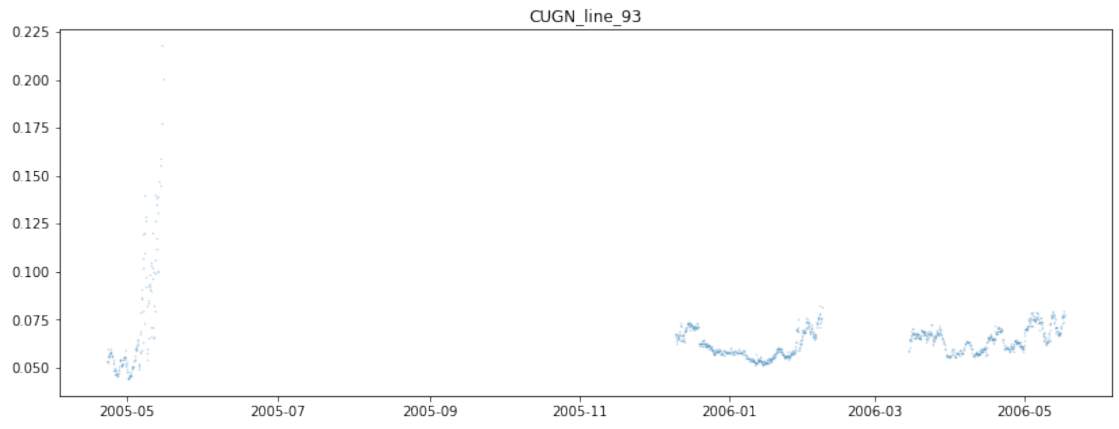


```
In [33]: for experiment_name, grp in spray_offshore.groupby('experiment'):
             fig = plt.figure(figsize=(14,5))
             plt.plot(grp.datetime, grp.fl_min, '.', markersize=1, alpha=0.3)
             noprint = plt.title(experiment_name)
```







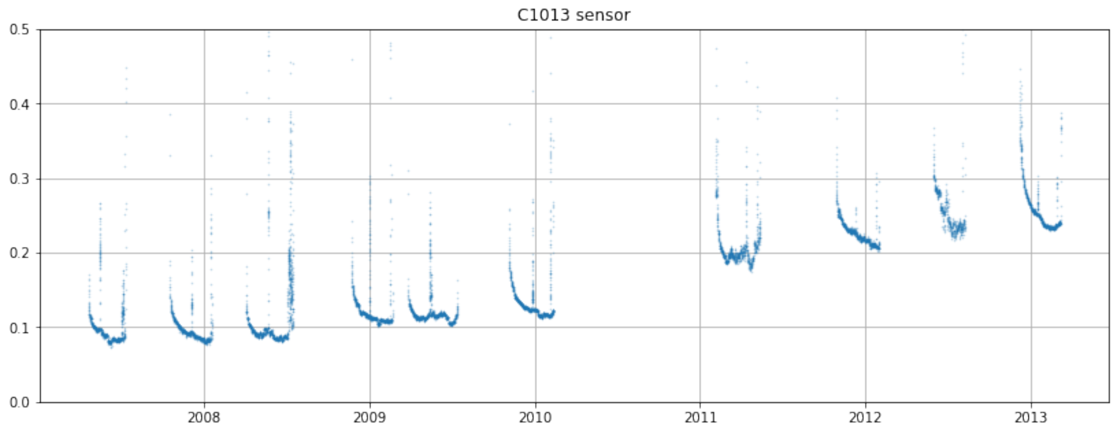


Looks like most of the cases are close to zero, which is a good thing. The few

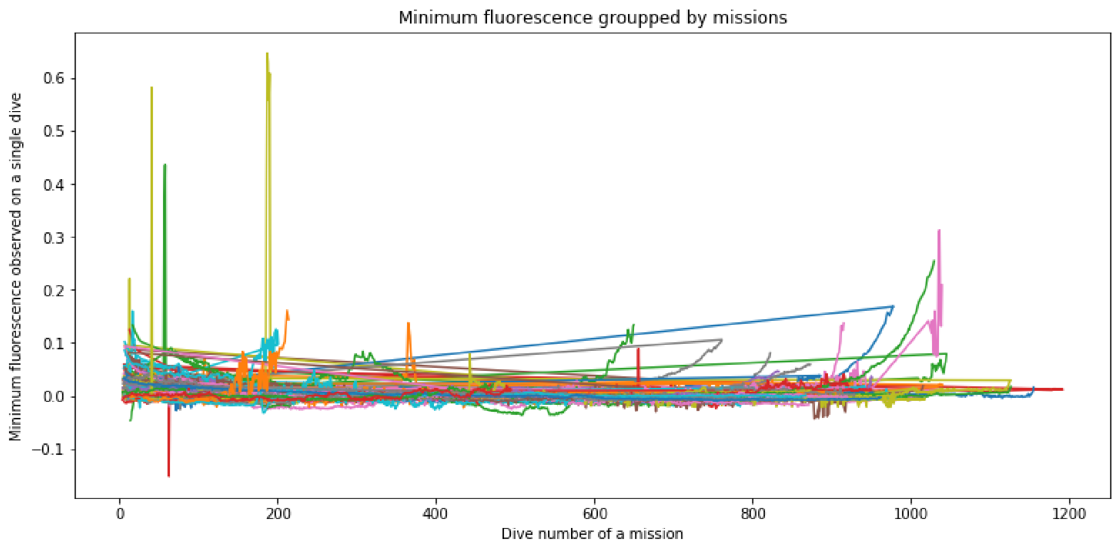
7.1.1 There is a dependency on the life of a sensor

Sensor C1013

```
In [34]: m66 = ['07401401', '07A01401', '08401401', '08B01401', '09301401', '09B01401', '112028
         ]
         subset = spray.isel(profile_id=[m in m66 for m in spray.mission.to_series()])

         fig = plt.figure(figsize=(14,5))
         plt.plot(subset.datetime, subset.fl_min, '.', markersize=1, alpha=0.3)
         plt.title('C1013 sensor')
         plt.ylim(0, 0.5)
         plt.grid()
```



```
In [35]: fig = plt.figure(figsize=(13,6))

         for experiment_name, experiment in spray_offshore.groupby('experiment'):
             grp = experiment.groupby('mission')

             for label, m in grp:
                 plt.plot(m.ndive, m.fl_min-m.fl_min.median())

         trash = plt.title('Minimum fluorescence groupped by missions')
         trash = plt.xlabel('Dive number of a mission')
         trash = plt.ylabel('Minimum fluorescence observed on a single dive')
```



It is interesting how many events seems to be consistent with neighbor dives. Could that be a real nature event? What are the other particulates that can optically respond like Chl-a?

There is a pattern here. Something happens around dive 350, latter again around dive 700. Note that the biggest event is coherent with that pattern, happenning around 1050.

ATENTION!!! Create a plot of offset versus distance to coast and another versus local depth. Could those events coincide with near coast profiles, thus more particulates including Chl-a?

7.2 Defining bias

Let's define one consistent bias per mission, and adjust each full mission by that single bias.

7.2.1 Minimum Fluorescence per mission

Before I was taking the actual minimum fl value per mission, but now I get the 0.1%, i.e. 0.001, percentile to give some robustness. This cause a small ammount (0.1%) of values to be negative, which I force to be equal to zero.

```
In [36]: def fl0_mission_min(x):
             return x - x.min()

         def fl0_mission_rmin(x):
             "q in range [0,1]"
             y = x - x.quantile(0.001)
             return y.where(y>0, 0)

         # spray['fl_unbias'] = spray.fl.groupby('mission').apply(fl0_mission_min)
         spray['fl_unbias'] = spray.fl.groupby('mission').apply(fl0_mission_rmin)

         # spray_offshore['fl_unbias'] = spray_offshore.fl.groupby('mission').apply(fl0_missio
         spray_offshore['fl_unbias'] = spray_offshore.fl.groupby('mission').apply(fl0_mission_r
```

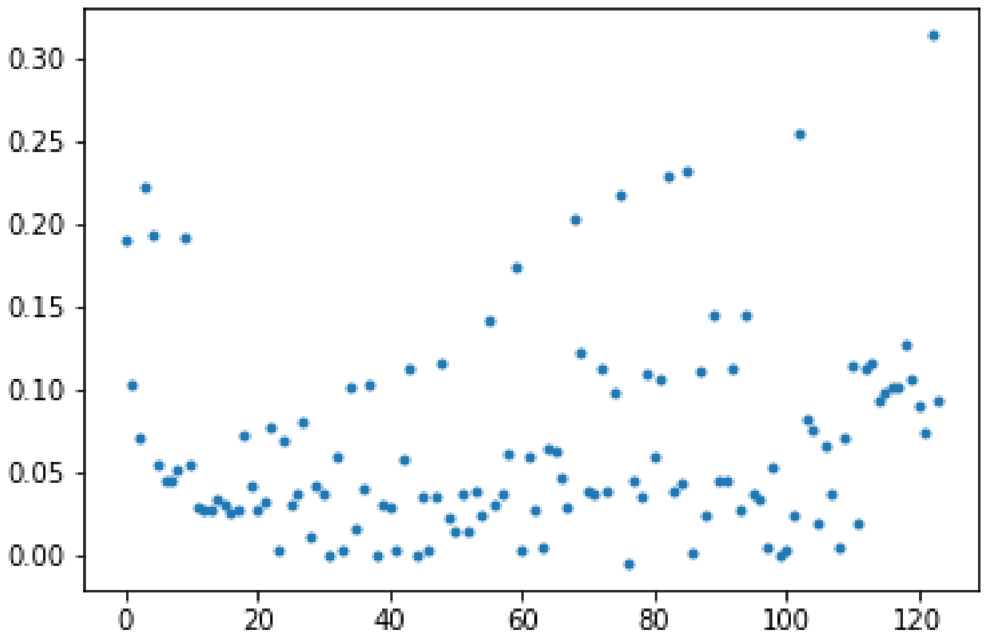
```
In [37]: spray
```

```
Out[37]: <xarray.Dataset>
         Dimensions:           (depth: 100, profile_id: 96718)
         Coordinates:
           * profile_id        (profile_id) int64 20215 20216 20217 20218 20219 ...
             ndive             (profile_id) int64 1 2 3 4 5 6 7 8 9 10 11 12 13 14 ...
             datetime          (profile_id) datetime64[ns] 2006-10-16T19:46:51 ...
             lat               (profile_id) float64 34.35 34.35 34.35 34.34 34.34 ...
             lon               (profile_id) float64 -119.8 -119.8 -119.8 -119.8 ...
             mission_id        (profile_id) int64 106 106 106 106 106 106 106 106 ...
             mission           (profile_id) object '06A00501' '06A00501' '06A00501' ...
             experiment_id     (profile_id) int64 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 ...
             experiment        (profile_id) object 'CUGN_line_80' 'CUGN_line_80' ...
           * depth             (depth) float64 10.0 20.0 30.0 40.0 50.0 60.0 70.0 ...
             days              (profile_id) float64 0.0 0.02537 0.04983 0.07683 ...
             night_time        (profile_id) bool True True True True False False ...
             quantile          float64 0.001
         Data variables:
             temp              (depth, profile_id) float64 16.57 16.38 16.24 16.16 ...
```

```
    sal                  (depth, profile_id) float64 33.42 33.41 33.43 33.43 ...
    fl                   (depth, profile_id) float64 1.382 1.23 1.429 2.101 ...
    fl_min               (profile_id) float64 0.2884 0.2508 0.2347 0.1904 ...
    profile_max_depth    (profile_id) float64 60.0 80.0 80.0 90.0 110.0 150.0 ...
    fl_deep_ref          (profile_id) float64 nan nan nan nan nan nan nan nan ...
    fl_unbias            (depth, profile_id) float64 1.354 1.201 1.401 2.072 ...
```

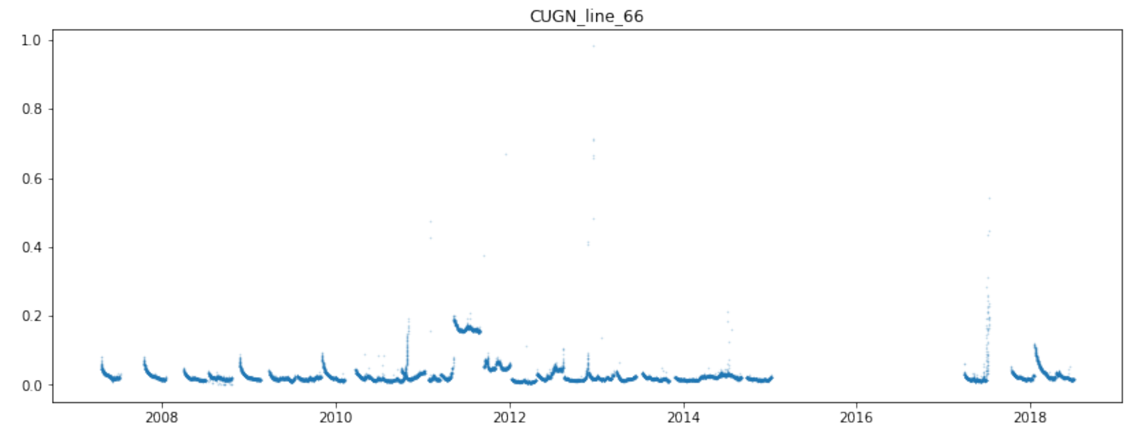
```
In [38]: fl_mission_bias = spray_offshore.groupby('mission').min()['fl']
         plt.plot(fl_mission_bias, '.')
```

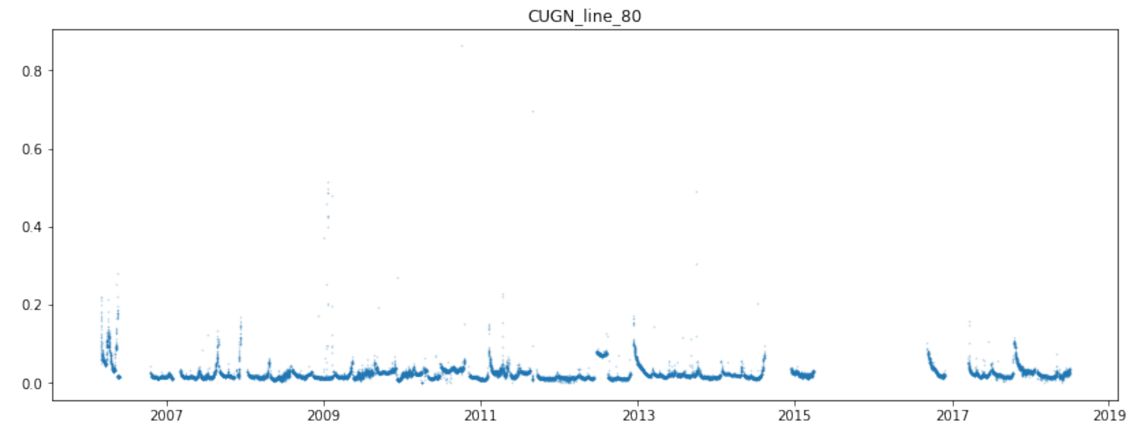
```
Out[38]: [<matplotlib.lines.Line2D at 0x1563cc320>]
```

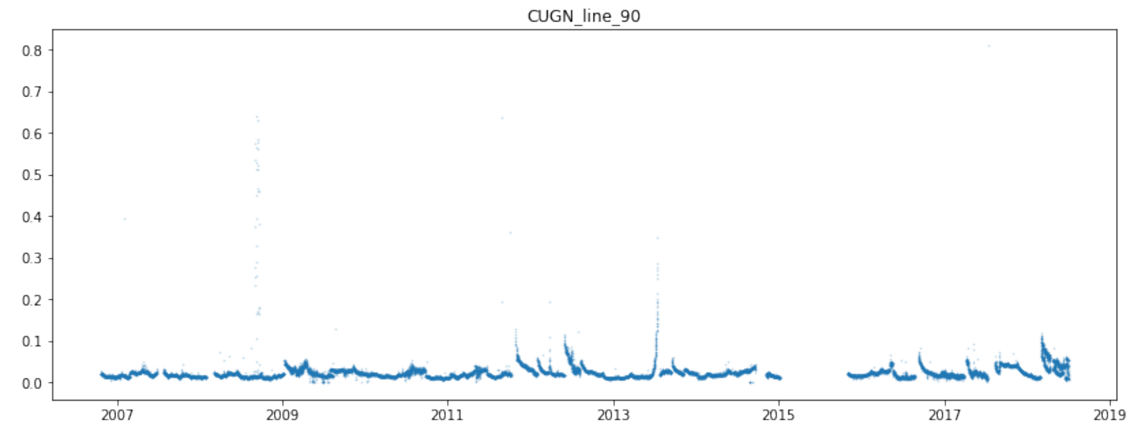


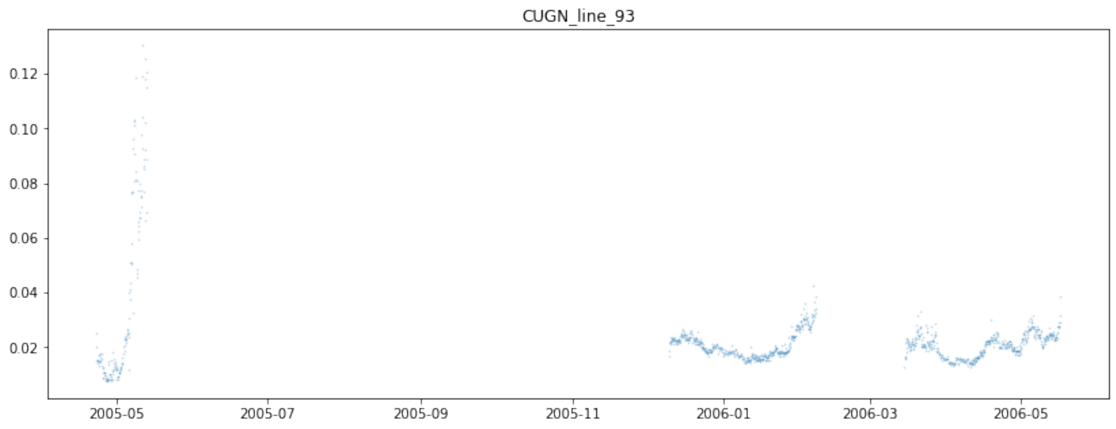
7.2.2 Reviewing plots for unbiased fluorescence

```
In [39]: for experiment_name, grp in spray_offshore.groupby('experiment'):
             fig = plt.figure(figsize=(14,5))
             plt.plot(grp.datetime, grp.fl_unbias.sel(depth=450), '.', markersize=1, alpha=0.3
             noprint = plt.title(experiment_name)
```

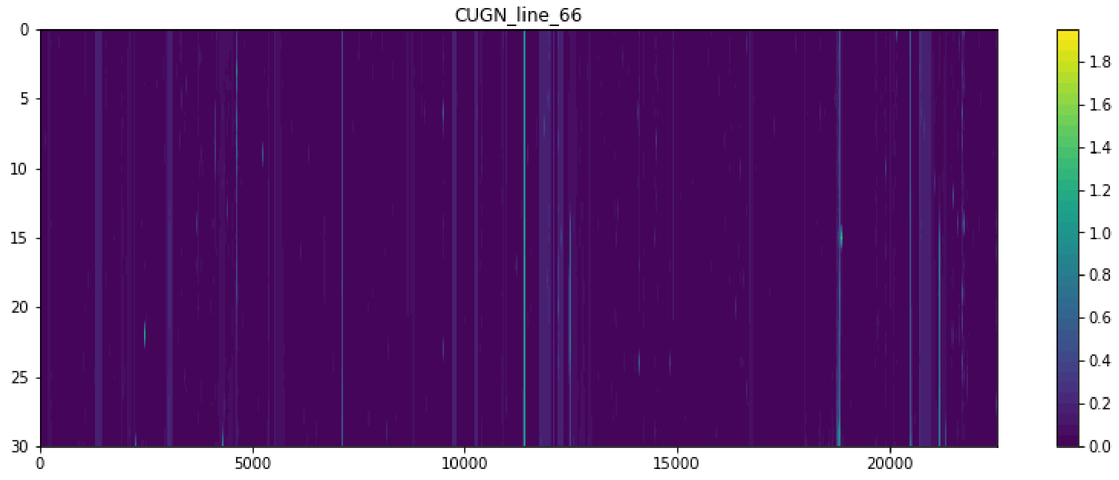

CUGN_line_66
1.0
0.8
0.6
0.4
0.2
0.0
2008
2010
2012
2014
2016
2018

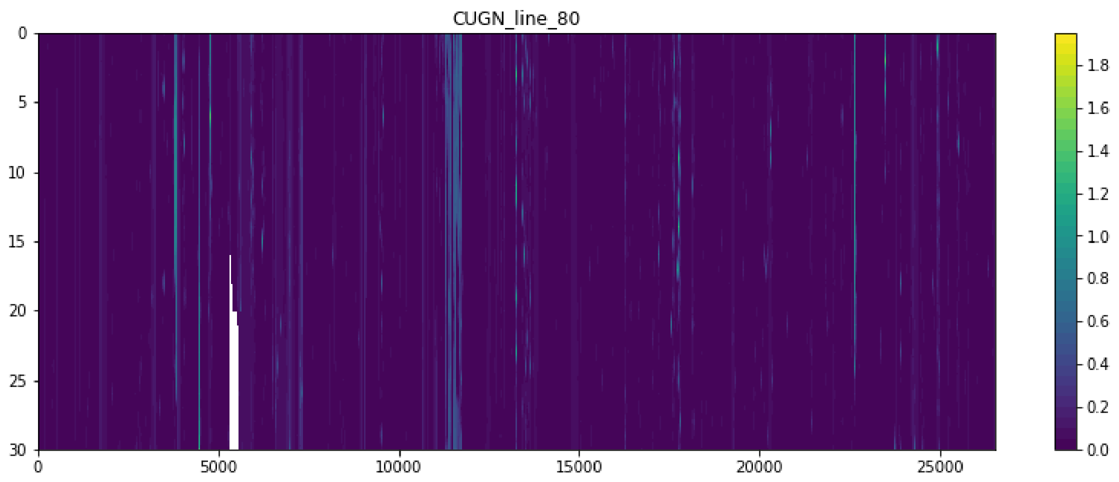

CUGN_line_80
0.8
0.6
0.4
0.2
0.0
2007
2009
2011
2013
2015
2017
2019

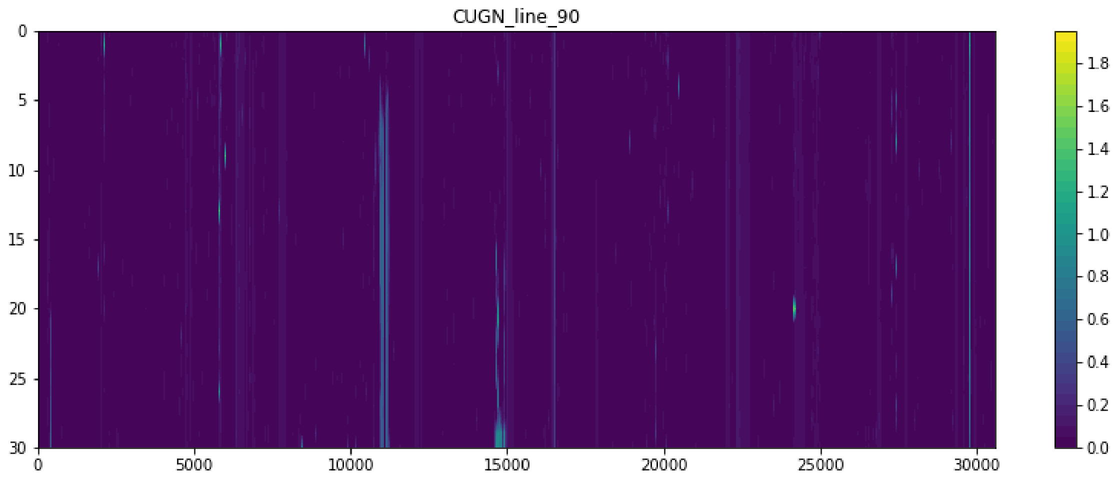

CUGN_line_90
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
2007
2009
2011
2013
2015
2017
2019

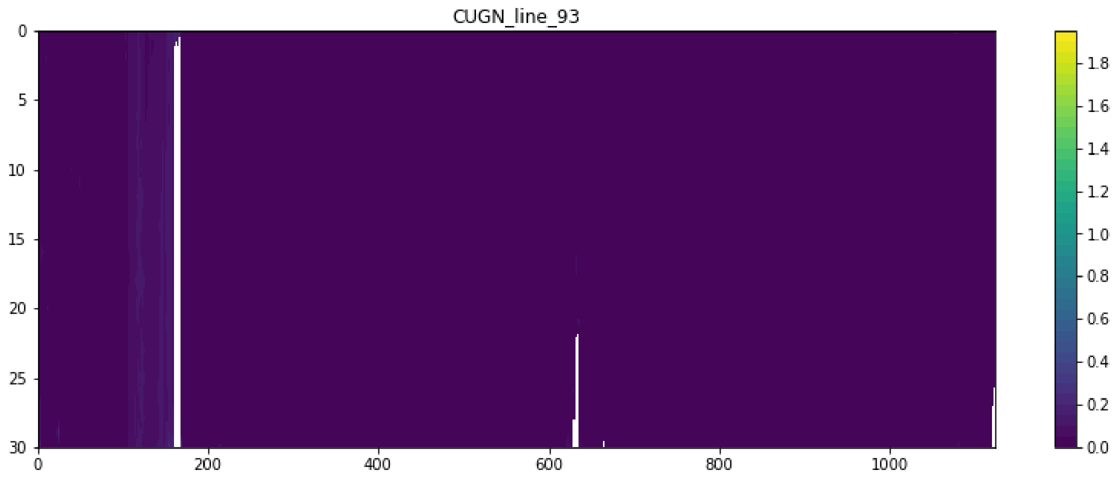


```
In [40]: # Limiting to upper 500m for better view

         scale = np.arange(0, 2, 0.05)
         idx = (spray_offshore.depth >= 200) & (spray_offshore.depth <= 500)
         for experiment_name, grp in spray_offshore.isel(depth=idx).groupby('experiment'):
             plt.figure(figsize=(14,5))
             #plt.contourf(grp.datetime, grp.depth, grp.fluorescence, scale)
             plt.contourf(grp.fl_unbias, scale)
             plt.colorbar()
             plt.gca().invert_yaxis()
             noprint = plt.title(experiment_name)
```

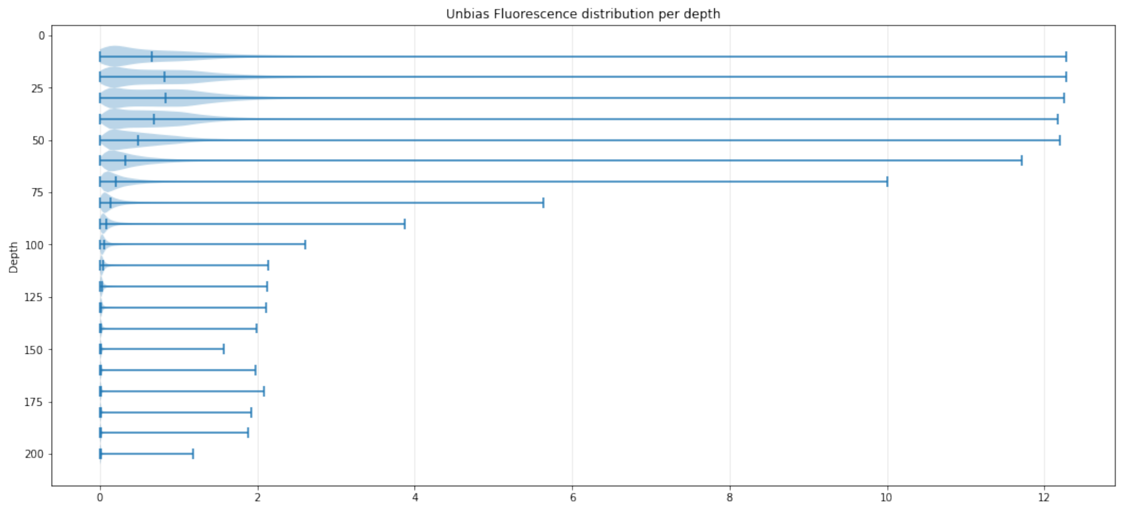


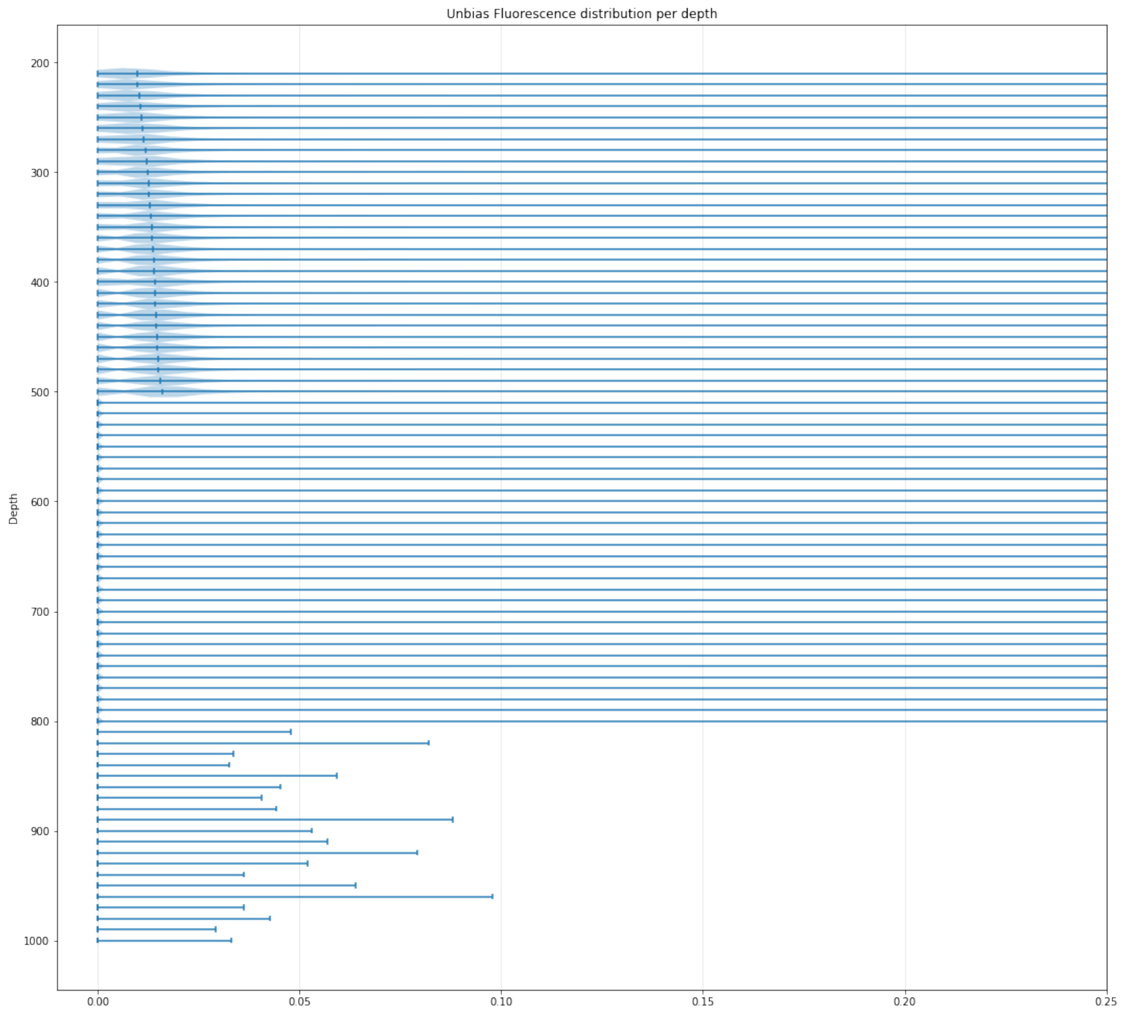

CUGN_line_80
0
5
10
15
20
25
30
0
5000
10000
15000
20000
25000
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
1.8


CUGN_line_90
0
5
10
15
20
25
30
0
5000
10000
15000
20000
25000
30000
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
1.8


CUGN_line_93
0
5
10
15
20
25
30
0
200
400
600
800
1000
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
1.8

```
In [41]: fig, ax = plt.subplots(nrows=1, ncols=1, figsize=(18, 8))

         ax.xaxis.grid(True, alpha=0.3)
         depth = spray.isel(depth=spray.depth <= 200).depth.to_series().tolist()
         data = [spray.sel(depth=d).fl_unbias.to_series().dropna().tolist() for d in depth]
         # plot violin plot
         ax.violinplot(data, depth,
                       widths=10,
                       points=200,
                       vert=False,
                       showmeans=False,
                       showextrema=True,
                       showmedians=True)

         ax.set_title('Unbias Fluorescence distribution per depth')
         ax.set_ylabel('Depth')
         ax.invert_yaxis()


         fig, ax = plt.subplots(nrows=1, ncols=1, figsize=(18, 17))

         ax.xaxis.grid(True, alpha=0.3)
         depth = spray.isel(depth=spray.depth > 200).depth.to_series().tolist()
         data = [spray.sel(depth=d).fl_unbias.to_series().dropna().tolist() for d in depth]
         # plot violin plot
         ax.violinplot(data, depth,
                       widths=10,
                       points=200,
                       vert=False,
                       showmeans=False,
                       showextrema=True,
                       showmedians=True)

         ax.set_title('Unbias Fluorescence distribution per depth')
         ax.set_ylabel('Depth')
         ax.invert_yaxis()
         plt.xlim(xmin=-0.01, xmax=0.25)


         del(depth)
         del(data)
```


Unbias Fluorescence distribution per depth
Depth
0
25
50
75
100
125
150
175
200
0
2
4
6
8
10
12


Unbias Fluorescence distribution per depth
Depth
200
300
400
500
600
700
800
900
1000
0.00
0.05
0.10
0.15
0.20
0.25

7.2.3 Zoom in the lower values

- As noted before, there is a trend of higher minimum fluorescence values as one goes deep.
- Also, note that the supposedly unbias minimum fl is not zero.

7.3 Assuming fl_0 as an exponential decay

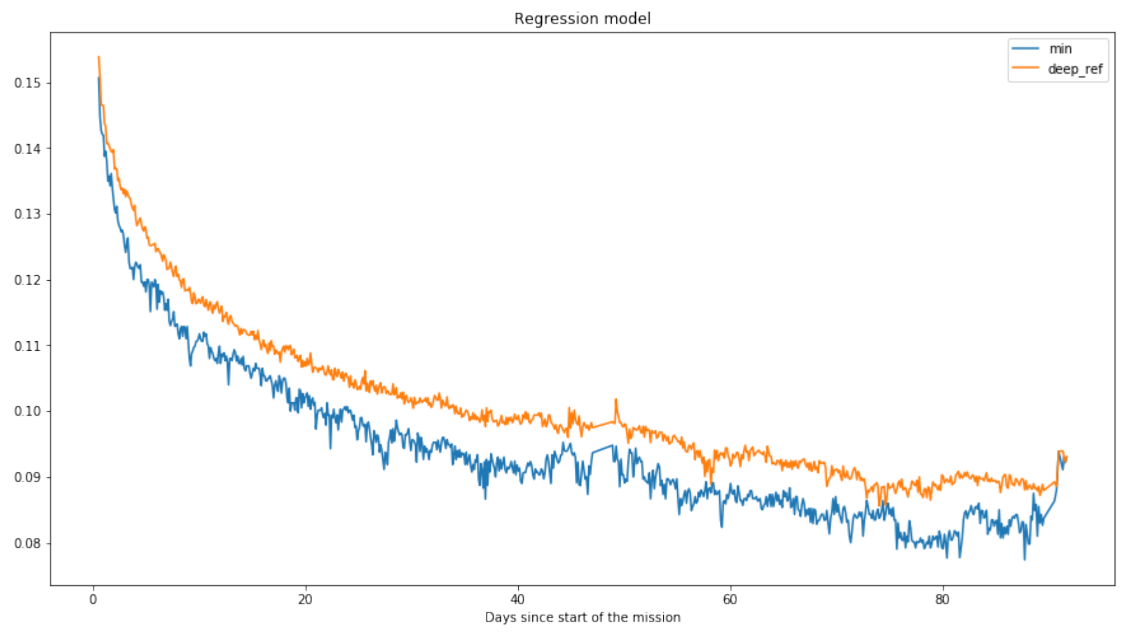
Looks like early in the mission the minimum fluorescence observed is quite high and along the mission it decays until stabilizes.

I'll use a Bayesian approach to find the expontential decay parameters. First let's illustrate it with a single mission, 07A01401 from CUGN line 66.

```
In [42]: np.unique(spray.isel(profile_id=spray.experiment=='CUGN_line_66')['mission'].to_series

Out[42]: array(['0025', '07401401', '07A01401', '08401401', '08703001', '08B01401',
                '09301401', '09701101', '09B01401', '10301401', '10704101',
                '10A01401', '11104101', '11501401', '11904101', '12101401',
                '12404101', '12801401', '12B04101', '13301401', '13704101',
                '13A06001', '13B04101', '14305701', '14602801', '14901101',
                '17302501', '17702801', '17A02501', '18104001'], dtype=object)
```

```
In [43]: # subset = spray_offshore.isel(profile_id=spray_offshore.mission=='08B01401')
         # idx = (subset.depth >= 300) & (subset.depth <=500)
         # subset['fl_deep_mean'] = subset.isel(depth=idx).fl.quantile(.25, dim='depth')
```

```
In [44]: subset = spray_offshore.isel(profile_id=spray_offshore.mission=='07A01401')

         subset = subset.reset_coords()[['ndive', 'days', 'fl_min', 'fl_deep_ref']].to_datafra
         subset.dropna(inplace=True)
         subset.sort_values(by='ndive', inplace=True)

         fig = plt.figure(figsize=(15,8))
         plt.plot(subset.days, subset.fl_min, label='min')
         plt.plot(subset.days, subset.fl_deep_ref, label='deep_ref')
         plt.legend()
         plt.title("Regression model")
         plt.xlabel("Days since start of the mission")

Out[44]: Text(0.5,0,'Days since start of the mission')
```



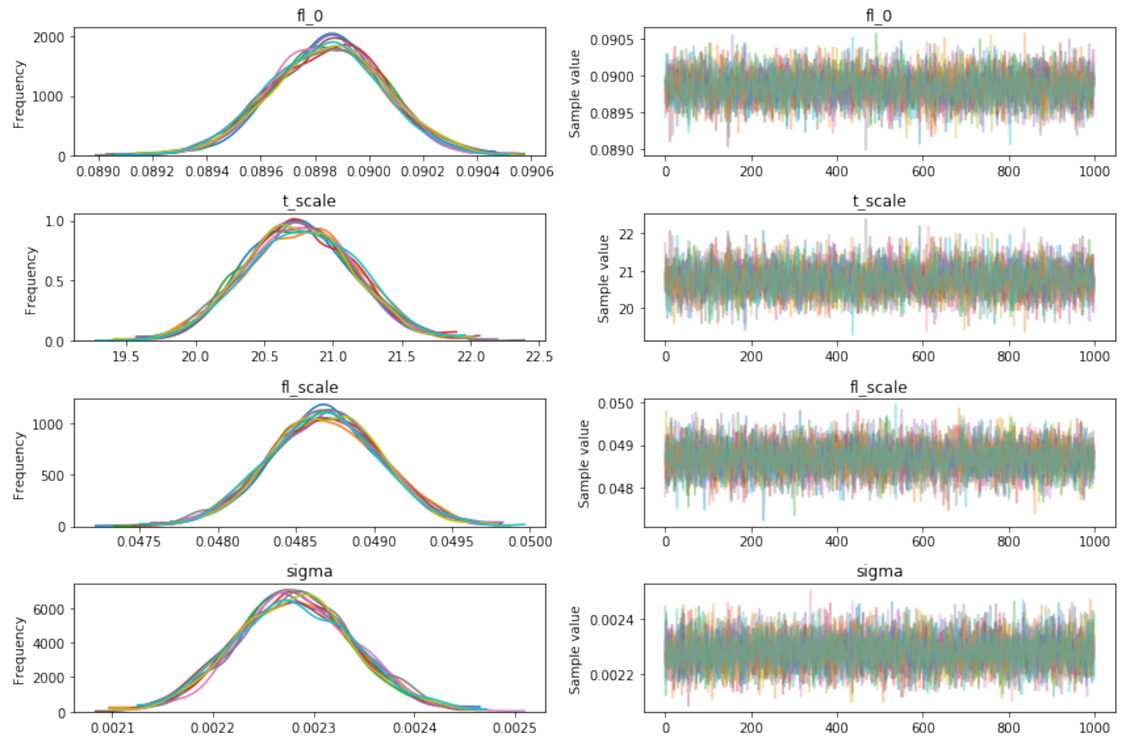
7.3.1 Demo, for one mission

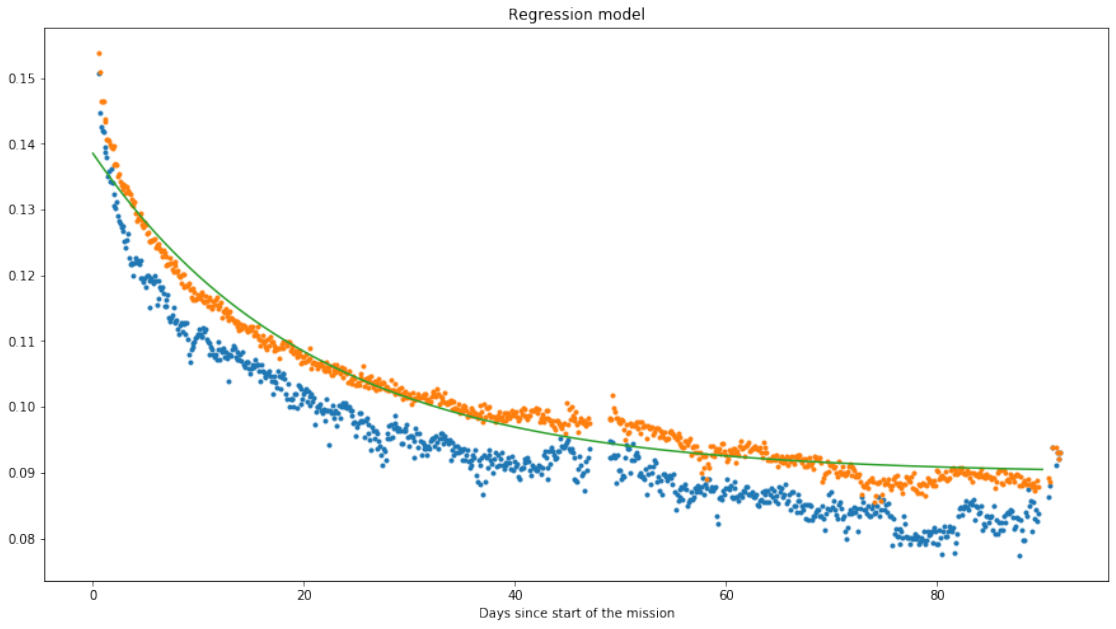
```
In [45]: def fit_fl_exponential_decay(X, Y):

             basic_model = pm.Model()

             with basic_model:

                 # Priors for unknown model parameters
                 fl_0 = pm.Normal('fl_0', mu=0, sd=.3)
                 # beta = pm.Normal('beta', mu=0, sd=10, shape=2)
                 t_scale = pm.HalfNormal('t_scale', sd=100)
                 fl_scale = pm.HalfNormal('fl_scale', sd=1)
                 sigma = pm.HalfNormal('sigma', sd=1)

                 # Expected value of outcome
                 mu = fl_0 + fl_scale * np.exp(- X / t_scale)

                 # Likelihood (sampling distribution) of observations
                 Y_obs = pm.Normal('Y_obs', mu=mu, sd=sigma, observed=Y)

             with basic_model:
                 # draw 5000 posterior samples
                 trace = pm.sample(draws=1000, chains=10, tune=1000, progressbar=False)

             return trace
```

```
In [46]: idx = (spray.profile_max_depth >= 400) & (spray.mission=='07A01401')
         subset = spray.isel(profile_id=idx)
         # subset['time'] = (subset.datetime - subset.datetime.min()) / (np.timedelta64(1, 's'
         subset = subset.reset_coords()[['ndive', 'days', 'fl_min', 'fl_deep_ref']].to_datafra
         subset.dropna(inplace=True)
         subset.sort_values(by='ndive', inplace=True)

         X = np.array(subset.days)
         Y = np.array(subset.fl_deep_ref)
         trace = fit_fl_exponential_decay(X=X, Y=Y)
         pm.traceplot(trace)

         summary = pm.summary(trace)
         x = np.arange(int(max(X)))
         y = summary['mean']['fl_0'] + summary['mean']['fl_scale'] * np.exp(-x / summary['mean

         fig = plt.figure(figsize=(15,8))
         plt.plot(X, subset.fl_min, '.')
         plt.plot(subset.days, subset.fl_deep_ref, '.')
         plt.plot(x, y)
         plt.title("Regression model")
         plt.xlabel("Days since start of the mission")
```

```
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
```

```
Out[46]: Text(0.5,0,'Days since start of the mission')
```



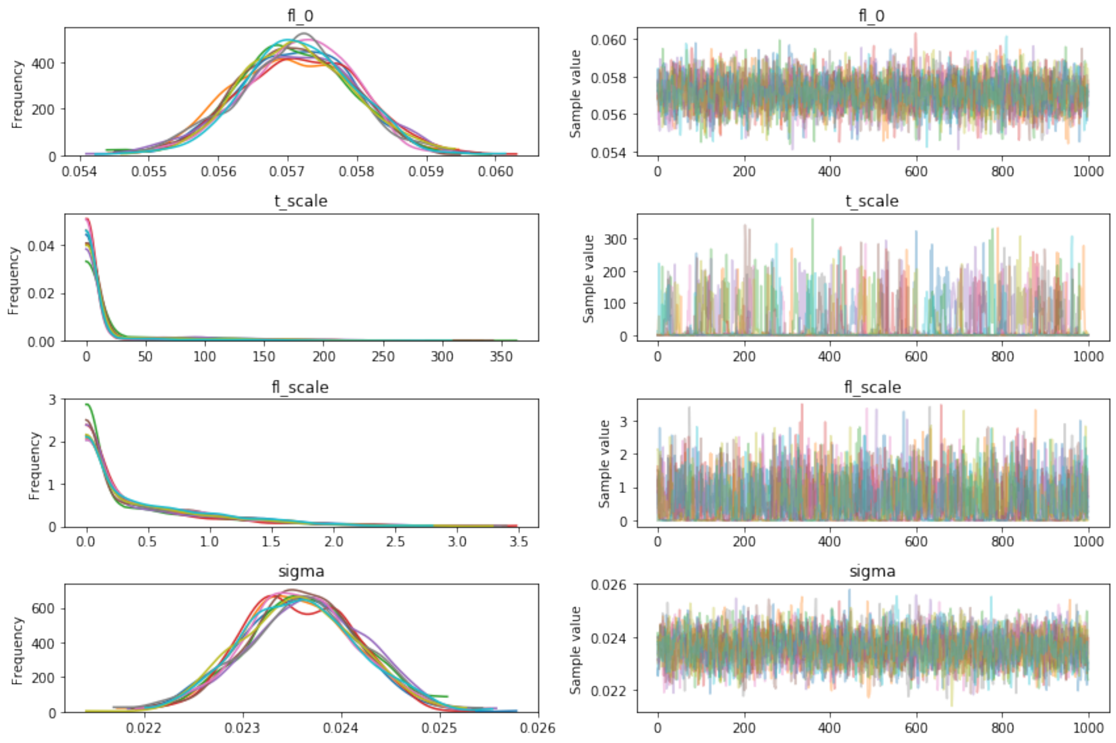


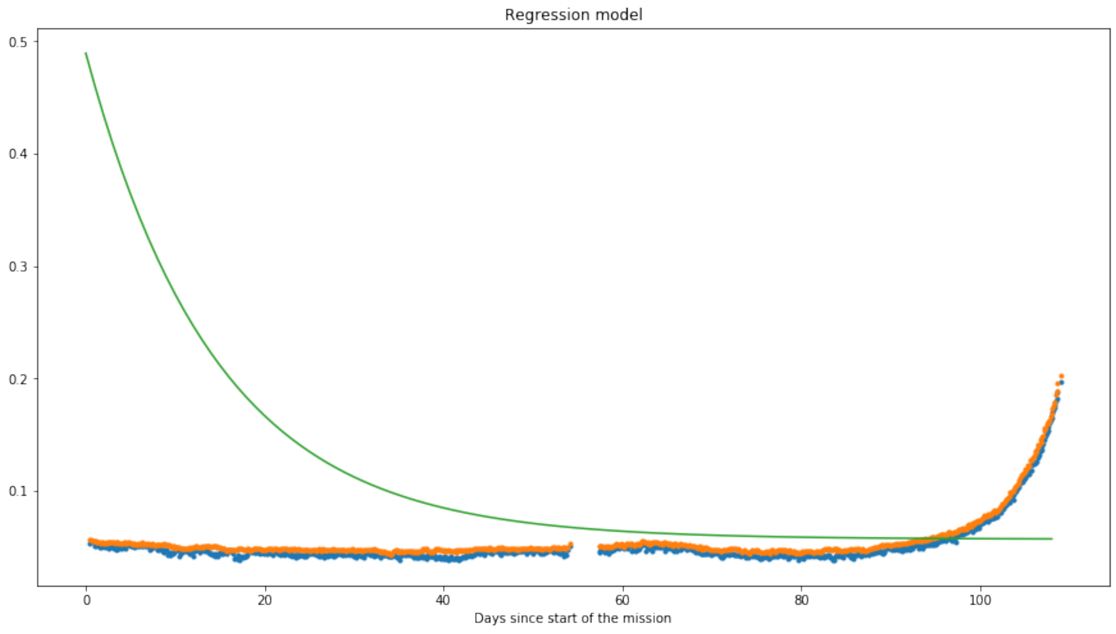
```
In [47]: idx = (spray.profile_max_depth >= 400) & (spray.mission=='10704101')
         subset = spray.isel(profile_id=idx)
```

```
        # subset['time'] = (subset.datetime - subset.datetime.min()) / (np.timedelta64(1, 's'
        subset = subset.reset_coords()[['ndive', 'days', 'fl_min', 'fl_deep_ref']].to_datafra
        subset.dropna(inplace=True)
        subset.sort_values(by='ndive', inplace=True)

        X = np.array(subset.days)
        Y = np.array(subset.fl_deep_ref)
        trace = fit_fl_exponential_decay(X=X, Y=Y)
        pm.traceplot(trace)

        summary = pm.summary(trace)
        x = np.arange(int(max(X)))
        y = summary['mean']['fl_0'] + summary['mean']['fl_scale'] * np.exp(-x / summary['mean

        fig = plt.figure(figsize=(15,8))
        plt.plot(X, subset.fl_min, '.')
        plt.plot(subset.days, subset.fl_deep_ref, '.')
        plt.plot(x, y)
        plt.title("Regression model")
        plt.xlabel("Days since start of the mission")
```

```
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
There were 6 divergences after tuning. Increase `target_accept` or reparameterize.
There were 23 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.946576409319251, but should be c
There were 6 divergences after tuning. Increase `target_accept` or reparameterize.
There were 5 divergences after tuning. Increase `target_accept` or reparameterize.
There were 2 divergences after tuning. Increase `target_accept` or reparameterize.
There were 43 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.715372510829087, but should be c
There were 8 divergences after tuning. Increase `target_accept` or reparameterize.
There were 2 divergences after tuning. Increase `target_accept` or reparameterize.
There were 23 divergences after tuning. Increase `target_accept` or reparameterize.
The number of effective samples is smaller than 25% for some parameters.
```

```
Out[47]: Text(0.5,0,'Days since start of the mission')
```





```
In [48]: fl_exp_fit = {
             '0025': {},
'07401401': {},
'07A01401': {},
'08401401': {},
'08703001': {},
'08B01401': {},
'09301401': {},
'09B01401': {},
'10301401': {},
'10A01401': {},
'11501401': {},
'12101401': {},
'13301401': {},
'13704101': {},
'13B04101': {},
'17A02501': {},
'18104001': {},
'07801301': {},
'08701301': {},
'09801301': {},
'10402501': {},
'10B01301': {},
'11202801': {},
'11602501': {},
'11901101': {},
'11C02501': {},
'12301101': {},
'12602501': {},
'12C02801': {},
'13301301': {},
'13A01101': {},
'14103001': {},
'14C02801': {},
'16903001': {},
'17305501': {},
'17605801': {},
'17A03001': {},
'18105801': {},
'0014': {},
'0041': {},
'06A01301': {},
'07701101': {},
'09403001': {},
'09B03001': {},
'11A02801': {},
'12103001': {},
'12501301': {},
'12803001': {},
'13903001': {},
            '13B05801': {},
            '14B05101': {},
            '16506401': {},
            '16904101': {},
            '16C06401': {},
            '17403001': {},
            '17B06401': {}

        }
```

```
In [216]: nColumns = 4
          fl_to_fit = 'fl_min'

          missions_list = []
          for experiment_name, experiment in spray_offshore.groupby('experiment'):
              print(experiment_name)
              nFigs = len(np.unique(experiment.mission))
              nRows = int(np.ceil(float(nFigs)/nColumns))
              fig, axes = plt.subplots(nrows=nRows, ncols=nColumns, figsize=(18, int(nRows * 2
              i = -1
              for mission_name, mission in experiment.groupby('mission'):
                  missions_list.append(mission_name)
                  mission = mission.reset_coords()[['ndive', 'days', 'fl_min', 'fl_deep_ref']]
                  mission.dropna(inplace=True)
                  mission.sort_values(by='ndive', inplace=True)

                  i += 1
                  if len(mission) > 0:
                      trace = fit_fl_exponential_decay(X=np.array(mission.days), Y=np.array(mi
                      # pm.traceplot(trace)

                      summary = pm.summary(trace)['mean']
                      x = np.arange(int(max(mission.days)))
                      y = summary['fl_0'] + summary['fl_scale'] * np.exp(-x / summary['t_scale

                      if nRows > 1:
                          nr = int(i/nColumns)
                          nc = i%nColumns
                          ax = axes[nr, nc]
                      else:
                          ax = axes[i]
                      ax.plot(mission.days, mission.fl_deep_ref, alpha=0.3)
                      ax.plot(mission.days, mission.fl_min, alpha=0.3)
                      if mission_name in fl_exp_fit:
                          ax.plot(x, y)
                          fl_exp_fit[mission_name] = summary
                      else:
                          ax.plot(x, y, 'r')
```

```
                    ax.set_xlim(0, 120)
                    ax.set_title(mission_name)
                    ax.xaxis.set_visible(False)
```

```
CUGN_line_66
```

```
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
The acceptance probability does not match the target. It is 0.8865818735809298, but should be
The acceptance probability does not match the target. It is 0.8830355409980185, but should be
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
The acceptance probability does not match the target. It is 0.8864578275318145, but should be
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
The acceptance probability does not match the target. It is 0.8908019067606378, but should be
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
```

```
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
The acceptance probability does not match the target. It is 0.8786956416261997, but should be 
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
There were 7 divergences after tuning. Increase `target_accept` or reparameterize.
There were 49 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.6905979806115989, but should be 
There were 19 divergences after tuning. Increase `target_accept` or reparameterize.
There were 7 divergences after tuning. Increase `target_accept` or reparameterize.
There were 7 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.9687752355226756, but should be 
There were 22 divergences after tuning. Increase `target_accept` or reparameterize.
There were 46 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.6920643661610621, but should be 
There were 16 divergences after tuning. Increase `target_accept` or reparameterize.
There were 11 divergences after tuning. Increase `target_accept` or reparameterize.
The number of effective samples is smaller than 25% for some parameters.
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
There were 81 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.6229576174710753, but should be 
There were 32 divergences after tuning. Increase `target_accept` or reparameterize.
There were 73 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.6407891681521155, but should be 
There were 73 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.6271086709798114, but should be 
There were 28 divergences after tuning. Increase `target_accept` or reparameterize.
There were 14 divergences after tuning. Increase `target_accept` or reparameterize.
There were 5 divergences after tuning. Increase `target_accept` or reparameterize.
There were 18 divergences after tuning. Increase `target_accept` or reparameterize.
There were 17 divergences after tuning. Increase `target_accept` or reparameterize.
There were 18 divergences after tuning. Increase `target_accept` or reparameterize.
The number of effective samples is smaller than 10% for some parameters.
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
```

```
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
The acceptance probability does not match the target. It is 0.888951080455917, but should be cl
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
There were 23 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.6826329100091754, but should be c
There were 1 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.8915602436596958, but should be c
There were 9 divergences after tuning. Increase `target_accept` or reparameterize.
There were 5 divergences after tuning. Increase `target_accept` or reparameterize.
There were 11 divergences after tuning. Increase `target_accept` or reparameterize.
There were 1 divergences after tuning. Increase `target_accept` or reparameterize.
There were 3 divergences after tuning. Increase `target_accept` or reparameterize.
There were 90 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.5682455242675956, but should be c
There were 17 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.6983567288196739, but should be c
The number of effective samples is smaller than 10% for some parameters.
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
There were 8 divergences after tuning. Increase `target_accept` or reparameterize.
There were 3 divergences after tuning. Increase `target_accept` or reparameterize.
There were 7 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.6375332257239721, but should be c
There were 4 divergences after tuning. Increase `target_accept` or reparameterize.
There were 2 divergences after tuning. Increase `target_accept` or reparameterize.
There were 4 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.7159865961476897, but should be c
There were 27 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.6700735245029703, but should be c
There were 2 divergences after tuning. Increase `target_accept` or reparameterize.
There were 26 divergences after tuning. Increase `target_accept` or reparameterize.
The number of effective samples is smaller than 10% for some parameters.
```

```
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
There were 40 divergences after tuning. Increase `target_accept` or reparameterize.
There were 813 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.009601013636475659, but should be
There were 53 divergences after tuning. Increase `target_accept` or reparameterize.
There were 263 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.3245461892033305, but should be
There were 417 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.1703514152847811, but should be
There were 46 divergences after tuning. Increase `target_accept` or reparameterize.
There were 399 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.1560152315252243, but should be
There were 46 divergences after tuning. Increase `target_accept` or reparameterize.
There were 15 divergences after tuning. Increase `target_accept` or reparameterize.
There were 43 divergences after tuning. Increase `target_accept` or reparameterize.
The gelman-rubin statistic is larger than 1.4 for some parameters. The sampler did not converg
The estimated number of effective samples is smaller than 200 for some parameters.
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
The acceptance probability does not match the target. It is 0.9657329415822169, but should be
There were 18 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.9548803374290773, but should be
There were 6 divergences after tuning. Increase `target_accept` or reparameterize.
There were 14 divergences after tuning. Increase `target_accept` or reparameterize.
There were 9 divergences after tuning. Increase `target_accept` or reparameterize.
There were 1 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.918988899859502, but should be c
There were 24 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.6862231772674567, but should be
There were 385 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.3510066835204064, but should be
```

```
There were 2 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.8812269040687545, but should be c
The estimated number of effective samples is smaller than 200 for some parameters.
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
The acceptance probability does not match the target. It is 0.8845745484591485, but should be c
There were 1 divergences after tuning. Increase `target_accept` or reparameterize.
There were 2 divergences after tuning. Increase `target_accept` or reparameterize.
There were 9 divergences after tuning. Increase `target_accept` or reparameterize.
There were 3 divergences after tuning. Increase `target_accept` or reparameterize.
There were 1 divergences after tuning. Increase `target_accept` or reparameterize.
The number of effective samples is smaller than 25% for some parameters.
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
The acceptance probability does not match the target. It is 0.8823901590630783, but should be c
The number of effective samples is smaller than 25% for some parameters.
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
The acceptance probability does not match the target. It is 0.8845813972994029, but should be c
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
The acceptance probability does not match the target. It is 0.8843840632459304, but should be c
```

```
CUGN_line_80
```

```
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
There were 4 divergences after tuning. Increase `target_accept` or reparameterize.
There were 5 divergences after tuning. Increase `target_accept` or reparameterize.
There were 236 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.30101838739808723, but should be
There were 5 divergences after tuning. Increase `target_accept` or reparameterize.
There were 2 divergences after tuning. Increase `target_accept` or reparameterize.
The number of effective samples is smaller than 10% for some parameters.
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
The acceptance probability does not match the target. It is 0.9513335446627247, but should be c
There were 549 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.09691627273795182, but should be
There were 16 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.61126699496256, but should be clo
There were 3 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.9241910555045014, but should be c
There were 1 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.9132371159608268, but should be c
There were 1 divergences after tuning. Increase `target_accept` or reparameterize.
There were 6 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.97857352389625, but should be clo
The gelman-rubin statistic is larger than 1.4 for some parameters. The sampler did not converge
The estimated number of effective samples is smaller than 200 for some parameters.
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
There were 1 divergences after tuning. Increase `target_accept` or reparameterize.
There were 135 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.42030240797428053, but should be
There were 1 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.912731169233722, but should be cl
There were 21 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.5124253267588411, but should be c
There were 60 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.5060793597979761, but should be c
There were 89 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.529414299039387, but should be cl
There were 1 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.8897290073371742, but should be c
There were 28 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.5177580270672062, but should be c
The estimated number of effective samples is smaller than 200 for some parameters.
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
The acceptance probability does not match the target. It is 0.8806026637357661, but should be c
The acceptance probability does not match the target. It is 0.907075974410836, but should be cl
There were 2 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.9135564925571543, but should be c
```

```
There were 2 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.8907061095125739, but should be c
There were 1 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.7177155641834101, but should be c
The number of effective samples is smaller than 25% for some parameters.
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
There were 42 divergences after tuning. Increase `target_accept` or reparameterize.
There were 56 divergences after tuning. Increase `target_accept` or reparameterize.
There were 36 divergences after tuning. Increase `target_accept` or reparameterize.
There were 71 divergences after tuning. Increase `target_accept` or reparameterize.
There were 207 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.3842058134320957, but should be c
There were 82 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.7151340475321355, but should be c
There were 45 divergences after tuning. Increase `target_accept` or reparameterize.
There were 49 divergences after tuning. Increase `target_accept` or reparameterize.
There were 52 divergences after tuning. Increase `target_accept` or reparameterize.
There were 42 divergences after tuning. Increase `target_accept` or reparameterize.
The number of effective samples is smaller than 10% for some parameters.
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
The acceptance probability does not match the target. It is 0.8824805971529773, but should be c
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
There were 2 divergences after tuning. Increase `target_accept` or reparameterize.
There were 11 divergences after tuning. Increase `target_accept` or reparameterize.
There were 4 divergences after tuning. Increase `target_accept` or reparameterize.
There were 1 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.8832627101768922, but should be c
There were 10 divergences after tuning. Increase `target_accept` or reparameterize.
There were 5 divergences after tuning. Increase `target_accept` or reparameterize.
There were 6 divergences after tuning. Increase `target_accept` or reparameterize.
There were 64 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.5409105554998822, but should be c
The number of effective samples is smaller than 10% for some parameters.
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
```

```
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
There were 14 divergences after tuning. Increase `target_accept` or reparameterize.
There were 14 divergences after tuning. Increase `target_accept` or reparameterize.
There were 11 divergences after tuning. Increase `target_accept` or reparameterize.
There were 54 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.5674277040358351, but should be c
There were 3 divergences after tuning. Increase `target_accept` or reparameterize.
There were 3 divergences after tuning. Increase `target_accept` or reparameterize.
There were 55 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.5609784626371971, but should be c
There were 1 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.8944382070421533, but should be c
There were 8 divergences after tuning. Increase `target_accept` or reparameterize.
There were 66 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.535237096887517, but should be cl
The estimated number of effective samples is smaller than 200 for some parameters.
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
There were 1 divergences after tuning. Increase `target_accept` or reparameterize.
There were 1 divergences after tuning. Increase `target_accept` or reparameterize.
There were 1 divergences after tuning. Increase `target_accept` or reparameterize.
There were 109 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.5455759617846038, but should be c
There were 4 divergences after tuning. Increase `target_accept` or reparameterize.
There were 4 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.7158418699505228, but should be c
The acceptance probability does not match the target. It is 0.8998626790000198, but should be c
There were 19 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.7043785885034828, but should be c
There were 5 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.6998583529596393, but should be c
The number of effective samples is smaller than 25% for some parameters.
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
There were 7 divergences after tuning. Increase `target_accept` or reparameterize.
```

```
The acceptance probability does not match the target. It is 0.43092815023737013, but should be
The acceptance probability does not match the target. It is 0.8826529511726126, but should be c
The acceptance probability does not match the target. It is 0.8949796265199785, but should be c
The acceptance probability does not match the target. It is 0.6946320582653329, but should be c
The acceptance probability does not match the target. It is 0.6792158302774304, but should be c
The number of effective samples is smaller than 25% for some parameters.
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
There were 2 divergences after tuning. Increase `target_accept` or reparameterize.
There were 46 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.3184314362825426, but should be c
The acceptance probability does not match the target. It is 0.8960052685502047, but should be c
The acceptance probability does not match the target. It is 0.6967639982472624, but should be c
The acceptance probability does not match the target. It is 0.9715315244978487, but should be c
There were 3 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.7123780870283392, but should be c
There were 2 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.6157140655215434, but should be c
The acceptance probability does not match the target. It is 0.9285743523522338, but should be c
The number of effective samples is smaller than 10% for some parameters.
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
The acceptance probability does not match the target. It is 0.9434363738718516, but should be c
There were 30 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.6979241719105228, but should be c
There were 176 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.518940375544113, but should be cl
There were 15 divergences after tuning. Increase `target_accept` or reparameterize.
There were 9 divergences after tuning. Increase `target_accept` or reparameterize.
There were 66 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.6445904418613245, but should be c
There were 5 divergences after tuning. Increase `target_accept` or reparameterize.
There were 1 divergences after tuning. Increase `target_accept` or reparameterize.
There were 22 divergences after tuning. Increase `target_accept` or reparameterize.
The number of effective samples is smaller than 10% for some parameters.
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
The acceptance probability does not match the target. It is 0.9130808737606037, but should be c
```

```
There were 6 divergences after tuning. Increase `target_accept` or reparameterize.
There were 5 divergences after tuning. Increase `target_accept` or reparameterize.
There were 290 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.4480135113859286, but should be c
There were 1 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.8924876595333366, but should be c
There were 12 divergences after tuning. Increase `target_accept` or reparameterize.
There were 18 divergences after tuning. Increase `target_accept` or reparameterize.
There were 34 divergences after tuning. Increase `target_accept` or reparameterize.
There were 6 divergences after tuning. Increase `target_accept` or reparameterize.
There were 7 divergences after tuning. Increase `target_accept` or reparameterize.
The number of effective samples is smaller than 10% for some parameters.
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
The acceptance probability does not match the target. It is 0.9287207862945637, but should be c
The acceptance probability does not match the target. It is 0.8990769102901068, but should be c
There were 4 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.7046758474947773, but should be c
There were 1 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.649294518878277, but should be cl
There were 3 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.6890862430544186, but should be c
There were 3 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.878982001581372, but should be cl
There were 4 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.601713685426198, but should be cl
The number of effective samples is smaller than 10% for some parameters.
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
There were 6 divergences after tuning. Increase `target_accept` or reparameterize.
There were 9 divergences after tuning. Increase `target_accept` or reparameterize.
There were 12 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.7116079648192183, but should be c
There were 33 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.4171859800360058, but should be c
There were 26 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.5977206599143661, but should be c
There were 27 divergences after tuning. Increase `target_accept` or reparameterize.
There were 31 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.674560528505864, but should be cl
```

```
There were 11 divergences after tuning. Increase `target_accept` or reparameterize.
There were 94 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.6068262155307456, but should be c
There were 22 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.6374944850677312, but should be c
The estimated number of effective samples is smaller than 200 for some parameters.
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
The acceptance probability does not match the target. It is 0.881058482783298, but should be cl
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
The acceptance probability does not match the target. It is 0.8867407183193573, but should be c
The acceptance probability does not match the target. It is 0.8836907368681203, but should be c
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
There were 12 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.8804018629463969, but should be c
There were 63 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.6957550429528694, but should be c
There were 55 divergences after tuning. Increase `target_accept` or reparameterize.
There were 15 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.8798169754021224, but should be c
There were 12 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.9454766908305013, but should be c
There were 5 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.8799792787277646, but should be c
There were 106 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.594889315809684, but should be cl
There were 12 divergences after tuning. Increase `target_accept` or reparameterize.
There were 59 divergences after tuning. Increase `target_accept` or reparameterize.
There were 26 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.6669156912697815, but should be c
The gelman-rubin statistic is larger than 1.2 for some parameters.
The estimated number of effective samples is smaller than 200 for some parameters.
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
The acceptance probability does not match the target. It is 0.8826651831152501, but should be c
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
The acceptance probability does not match the target. It is 0.8851779014292962, but should be c
The acceptance probability does not match the target. It is 0.8877945440452093, but should be c
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
The acceptance probability does not match the target. It is 0.8971263945304027, but should be c
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
There were 2 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.923149371821954, but should be cl
There were 4 divergences after tuning. Increase `target_accept` or reparameterize.
There were 418 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.10398170921528317, but should be
There were 1 divergences after tuning. Increase `target_accept` or reparameterize.
There were 2 divergences after tuning. Increase `target_accept` or reparameterize.
There were 46 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.6862938025217027, but should be c
There were 6 divergences after tuning. Increase `target_accept` or reparameterize.
There were 4 divergences after tuning. Increase `target_accept` or reparameterize.
There were 22 divergences after tuning. Increase `target_accept` or reparameterize.
There were 7 divergences after tuning. Increase `target_accept` or reparameterize.
The estimated number of effective samples is smaller than 200 for some parameters.
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
```

```
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
The acceptance probability does not match the target. It is 0.8810469471636627, but should be c
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
There were 1 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.8996934840395548, but should be c
There were 17 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.9788486707724315, but should be c
The acceptance probability does not match the target. It is 0.9712823205789441, but should be c
There were 93 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.6044791105293504, but should be c
There were 42 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.6740814867806187, but should be c
There were 2 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.9040711956298849, but should be c
There were 369 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.3445151498507916, but should be c
There were 11 divergences after tuning. Increase `target_accept` or reparameterize.
There were 37 divergences after tuning. Increase `target_accept` or reparameterize.
The number of effective samples is smaller than 10% for some parameters.
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
The acceptance probability does not match the target. It is 0.8844577212955268, but should be c
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
The acceptance probability does not match the target. It is 0.92159787343477, but should be clo
There were 74 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.2249841937472559, but should be c
The acceptance probability does not match the target. It is 0.966135385179883, but should be cl
The acceptance probability does not match the target. It is 0.9351856705244184, but should be c
The number of effective samples is smaller than 10% for some parameters.
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
```

```
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
The acceptance probability does not match the target. It is 0.8826991626448849, but should be c
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
The acceptance probability does not match the target. It is 0.8851376131517349, but should be c
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
```

```
CUGN_line_90
```

```
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
There were 59 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.67276038404284, but should be clo
There were 131 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.4641954569205348, but should be c
There were 78 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.5558148938374913, but should be c
There were 84 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.5626204259137443, but should be c
There were 16 divergences after tuning. Increase `target_accept` or reparameterize.
There were 28 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.6986717293194289, but should be c
```

```
There were 69 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.5931961350933779, but should be c
There were 5 divergences after tuning. Increase `target_accept` or reparameterize.
There were 59 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.6600262960886033, but should be c
There were 79 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.5911689095153839, but should be c
The gelman-rubin statistic is larger than 1.05 for some parameters. This indicates slight probl
The estimated number of effective samples is smaller than 200 for some parameters.
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
There were 45 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.6063508520283298, but should be c
There were 83 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.56312361988428, but should be clo
There were 21 divergences after tuning. Increase `target_accept` or reparameterize.
There were 73 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.6309634320271562, but should be c
There were 220 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.5574050866394308, but should be c
There were 472 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.09794831002003535, but should be
There were 29 divergences after tuning. Increase `target_accept` or reparameterize.
There were 52 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.5831128008443103, but should be c
There were 26 divergences after tuning. Increase `target_accept` or reparameterize.
There were 24 divergences after tuning. Increase `target_accept` or reparameterize.
The gelman-rubin statistic is larger than 1.4 for some parameters. The sampler did not converge
The estimated number of effective samples is smaller than 200 for some parameters.
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
There were 126 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.6030699604321265, but should be c
There were 39 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.6985603819369499, but should be c
There were 34 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.7194148857964627, but should be c
There were 1 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.8865124310216426, but should be c
There were 6 divergences after tuning. Increase `target_accept` or reparameterize.
```

```
The acceptance probability does not match the target. It is 0.9393367024639486, but should be c
There were 68 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.6845331384395921, but should be c
There were 1 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.8893398604942997, but should be c
There were 28 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.6972013585268871, but should be c
The number of effective samples is smaller than 10% for some parameters.
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
The acceptance probability does not match the target. It is 0.8860152620444998, but should be c
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
There were 16 divergences after tuning. Increase `target_accept` or reparameterize.
There were 1 divergences after tuning. Increase `target_accept` or reparameterize.
There were 4 divergences after tuning. Increase `target_accept` or reparameterize.
There were 59 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.5596109960856946, but should be c
There were 2 divergences after tuning. Increase `target_accept` or reparameterize.
There were 7 divergences after tuning. Increase `target_accept` or reparameterize.
There were 6 divergences after tuning. Increase `target_accept` or reparameterize.
The number of effective samples is smaller than 25% for some parameters.
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
There were 4 divergences after tuning. Increase `target_accept` or reparameterize.
There were 45 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.703600156218959, but should be cl
There were 16 divergences after tuning. Increase `target_accept` or reparameterize.
There were 21 divergences after tuning. Increase `target_accept` or reparameterize.
There were 15 divergences after tuning. Increase `target_accept` or reparameterize.
There were 8 divergences after tuning. Increase `target_accept` or reparameterize.
There were 27 divergences after tuning. Increase `target_accept` or reparameterize.
There were 26 divergences after tuning. Increase `target_accept` or reparameterize.
There were 10 divergences after tuning. Increase `target_accept` or reparameterize.
There were 531 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.359983764035452, but should be cl
The number of effective samples is smaller than 10% for some parameters.
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
There were 2 divergences after tuning. Increase `target_accept` or reparameterize.
There were 21 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.673292541879322, but should be cl
There were 4 divergences after tuning. Increase `target_accept` or reparameterize.
There were 6 divergences after tuning. Increase `target_accept` or reparameterize.
There were 3 divergences after tuning. Increase `target_accept` or reparameterize.
There were 9 divergences after tuning. Increase `target_accept` or reparameterize.
The number of effective samples is smaller than 25% for some parameters.
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
There were 109 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.5678372479526247, but should be c
There were 30 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.6473283021441499, but should be c
There were 49 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.6634038034590831, but should be c
There were 218 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.5708119349529548, but should be c
There were 65 divergences after tuning. Increase `target_accept` or reparameterize.
```

```
The acceptance probability does not match the target. It is 0.6469142826509982, but should be c
There were 40 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.611919219298118, but should be cl
There were 319 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.4885571647490394, but should be c
There were 178 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.5342210179723196, but should be c
The number of effective samples is smaller than 10% for some parameters.
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
There were 10 divergences after tuning. Increase `target_accept` or reparameterize.
There were 9 divergences after tuning. Increase `target_accept` or reparameterize.
There were 4 divergences after tuning. Increase `target_accept` or reparameterize.
There were 14 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.6913085046732326, but should be c
There were 8 divergences after tuning. Increase `target_accept` or reparameterize.
There were 8 divergences after tuning. Increase `target_accept` or reparameterize.
There were 27 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.6252608364231493, but should be c
There were 24 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.6825749947719251, but should be c
The acceptance probability does not match the target. It is 0.9232156691409951, but should be c
There were 5 divergences after tuning. Increase `target_accept` or reparameterize.
The number of effective samples is smaller than 10% for some parameters.
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
There were 28 divergences after tuning. Increase `target_accept` or reparameterize.
There were 17 divergences after tuning. Increase `target_accept` or reparameterize.
There were 13 divergences after tuning. Increase `target_accept` or reparameterize.
There were 12 divergences after tuning. Increase `target_accept` or reparameterize.
There were 5 divergences after tuning. Increase `target_accept` or reparameterize.
There were 43 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.6613069439005034, but should be c
There were 28 divergences after tuning. Increase `target_accept` or reparameterize.
There were 3 divergences after tuning. Increase `target_accept` or reparameterize.
There were 23 divergences after tuning. Increase `target_accept` or reparameterize.
There were 3 divergences after tuning. Increase `target_accept` or reparameterize.
The number of effective samples is smaller than 10% for some parameters.
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
```

```
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
The acceptance probability does not match the target. It is 0.9061483149218349, but should be c
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
The acceptance probability does not match the target. It is 0.5550406531609459, but should be c
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
There were 30 divergences after tuning. Increase `target_accept` or reparameterize.
There were 29 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.8826386881957327, but should be c
There were 17 divergences after tuning. Increase `target_accept` or reparameterize.
There were 15 divergences after tuning. Increase `target_accept` or reparameterize.
There were 27 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.7135756196131436, but should be c
There were 19 divergences after tuning. Increase `target_accept` or reparameterize.
There were 9 divergences after tuning. Increase `target_accept` or reparameterize.
There were 9 divergences after tuning. Increase `target_accept` or reparameterize.
There were 12 divergences after tuning. Increase `target_accept` or reparameterize.
The number of effective samples is smaller than 25% for some parameters.
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
There were 1 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.902520719186215, but should be cl
The acceptance probability does not match the target. It is 0.9116130279812508, but should be c
There were 1 divergences after tuning. Increase `target_accept` or reparameterize.
```

```
The acceptance probability does not match the target. It is 0.9610313543036489, but should be c
There were 2 divergences after tuning. Increase `target_accept` or reparameterize.
There were 9 divergences after tuning. Increase `target_accept` or reparameterize.
There were 15 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.8922551514305348, but should be c
The acceptance probability does not match the target. It is 0.9683796630302389, but should be c
The gelman-rubin statistic is larger than 1.05 for some parameters. This indicates slight probl
The estimated number of effective samples is smaller than 200 for some parameters.
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
The acceptance probability does not match the target. It is 0.8907204879500459, but should be c
The acceptance probability does not match the target. It is 0.8795933043736462, but should be c
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
There were 3 divergences after tuning. Increase `target_accept` or reparameterize.
There were 7 divergences after tuning. Increase `target_accept` or reparameterize.
There were 4 divergences after tuning. Increase `target_accept` or reparameterize.
There were 7 divergences after tuning. Increase `target_accept` or reparameterize.
There were 44 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.6434519821568399, but should be c
There were 1 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.8887518302893558, but should be c
There were 19 divergences after tuning. Increase `target_accept` or reparameterize.
The number of effective samples is smaller than 25% for some parameters.
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
There were 16 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.9458303373817271, but should be c
There were 406 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.41053189638235366, but should be
There were 20 divergences after tuning. Increase `target_accept` or reparameterize.
There were 38 divergences after tuning. Increase `target_accept` or reparameterize.
There were 57 divergences after tuning. Increase `target_accept` or reparameterize.
There were 404 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.4533160180338617, but should be c
There were 7 divergences after tuning. Increase `target_accept` or reparameterize.
There were 22 divergences after tuning. Increase `target_accept` or reparameterize.
The gelman-rubin statistic is larger than 1.05 for some parameters. This indicates slight probl
The estimated number of effective samples is smaller than 200 for some parameters.
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
```

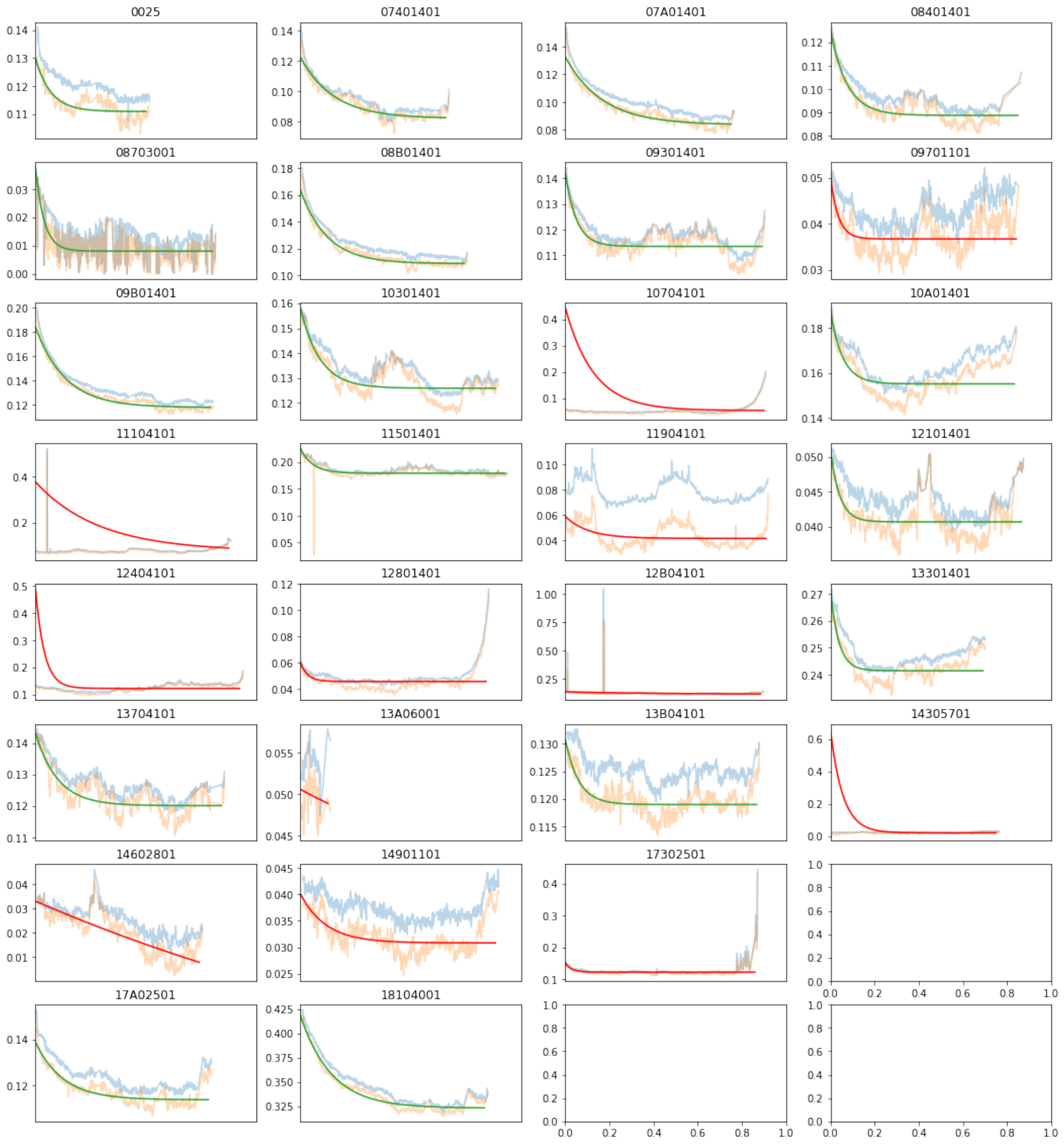
```
NUTS: [sigma, fl_scale, t_scale, fl_0]
There were 35 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.9501515553067541, but should be c
There were 6 divergences after tuning. Increase `target_accept` or reparameterize.
There were 3 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.8884467713381023, but should be c
The acceptance probability does not match the target. It is 0.94335925210297, but should be clo
There were 30 divergences after tuning. Increase `target_accept` or reparameterize.
There were 2 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.893091630353101, but should be cl
There were 1 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.8827745636912605, but should be c
There were 4 divergences after tuning. Increase `target_accept` or reparameterize.
The number of effective samples is smaller than 10% for some parameters.
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
There were 257 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.46181170043214975, but should be
There were 3 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.9565784144306558, but should be c
There were 17 divergences after tuning. Increase `target_accept` or reparameterize.
There were 7 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.9545069398881245, but should be c
The acceptance probability does not match the target. It is 0.9341254859100142, but should be c
There were 72 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.6253652857265904, but should be c
There were 7 divergences after tuning. Increase `target_accept` or reparameterize.
There were 51 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.7014300033607639, but should be c
The number of effective samples is smaller than 10% for some parameters.
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
There were 70 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.7115754385754934, but should be c
There were 167 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.49546114017519177, but should be
There were 176 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.41797857381837505, but should be
There were 72 divergences after tuning. Increase `target_accept` or reparameterize.
There were 429 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.11041441403948894, but should be
There were 174 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.23574024238385097, but should be
There were 104 divergences after tuning. Increase `target_accept` or reparameterize.
There were 75 divergences after tuning. Increase `target_accept` or reparameterize.
There were 169 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.3766354949759195, but should be c
There were 74 divergences after tuning. Increase `target_accept` or reparameterize.
The gelman-rubin statistic is larger than 1.2 for some parameters.
The estimated number of effective samples is smaller than 200 for some parameters.
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
There were 71 divergences after tuning. Increase `target_accept` or reparameterize.
The chain reached the maximum tree depth. Increase max_treedepth, increase target_accept or rep
There were 28 divergences after tuning. Increase `target_accept` or reparameterize.
The chain reached the maximum tree depth. Increase max_treedepth, increase target_accept or rep
There were 68 divergences after tuning. Increase `target_accept` or reparameterize.
The chain reached the maximum tree depth. Increase max_treedepth, increase target_accept or rep
There were 115 divergences after tuning. Increase `target_accept` or reparameterize.
The chain reached the maximum tree depth. Increase max_treedepth, increase target_accept or rep
There were 34 divergences after tuning. Increase `target_accept` or reparameterize.
The chain reached the maximum tree depth. Increase max_treedepth, increase target_accept or rep
There were 26 divergences after tuning. Increase `target_accept` or reparameterize.
The chain reached the maximum tree depth. Increase max_treedepth, increase target_accept or rep
There were 28 divergences after tuning. Increase `target_accept` or reparameterize.
The chain reached the maximum tree depth. Increase max_treedepth, increase target_accept or rep
There were 49 divergences after tuning. Increase `target_accept` or reparameterize.
The chain reached the maximum tree depth. Increase max_treedepth, increase target_accept or rep
There were 82 divergences after tuning. Increase `target_accept` or reparameterize.
The chain reached the maximum tree depth. Increase max_treedepth, increase target_accept or rep
There were 192 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.3954021441743536, but should be c
The chain reached the maximum tree depth. Increase max_treedepth, increase target_accept or rep
The number of effective samples is smaller than 10% for some parameters.
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
The acceptance probability does not match the target. It is 0.8824981092316895, but should be c
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
There were 321 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.47212282966108615, but should be
There were 3 divergences after tuning. Increase `target_accept` or reparameterize.
```

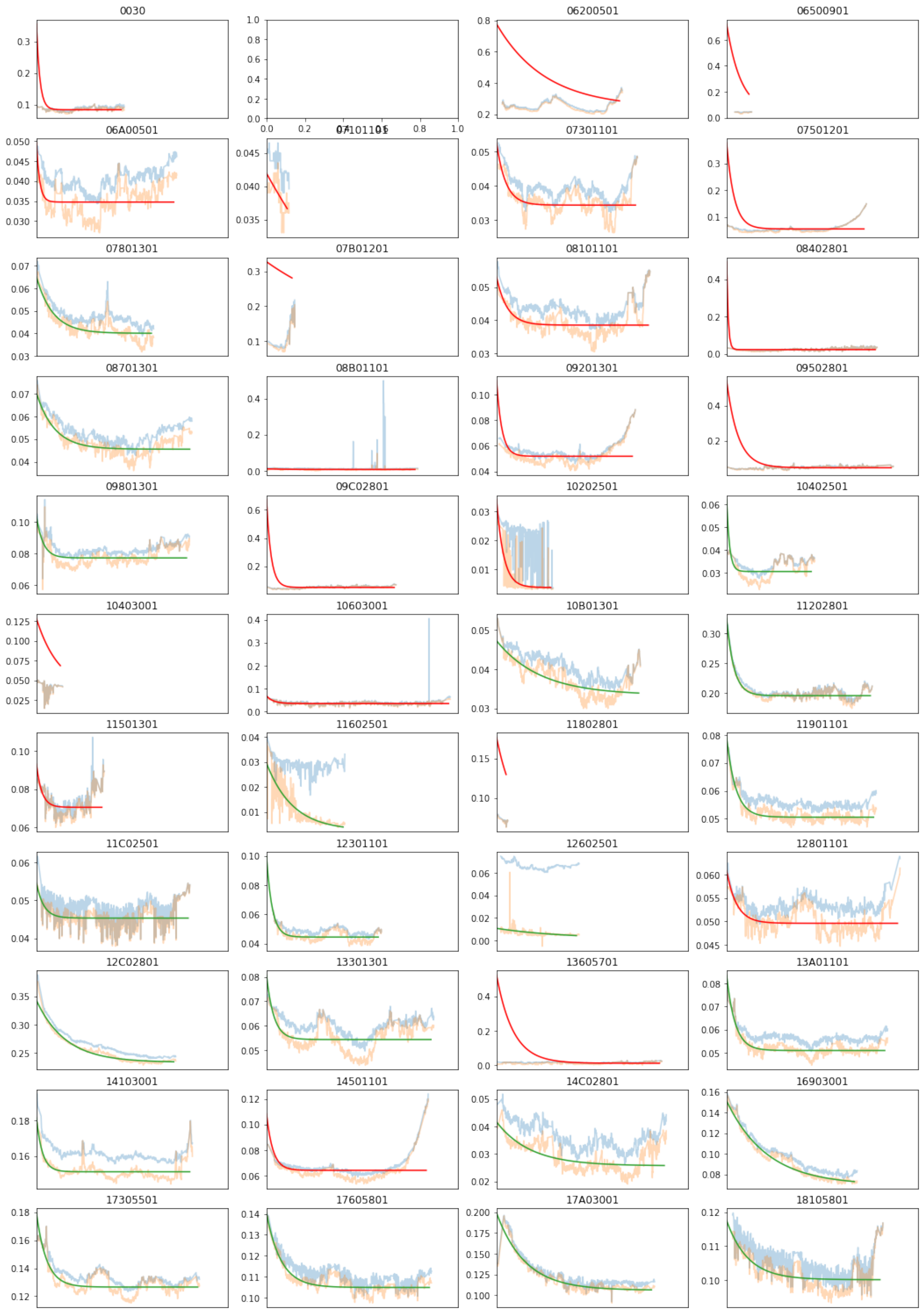
```
The acceptance probability does not match the target. It is 0.885085212828823, but should be cl
There were 11 divergences after tuning. Increase `target_accept` or reparameterize.
There were 1 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.9077213107163103, but should be c
The acceptance probability does not match the target. It is 0.8790465050247391, but should be c
There were 18 divergences after tuning. Increase `target_accept` or reparameterize.
There were 7 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.9248516391407208, but should be c
There were 137 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.5883179790243291, but should be c
There were 2 divergences after tuning. Increase `target_accept` or reparameterize.
The number of effective samples is smaller than 10% for some parameters.
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
There were 11 divergences after tuning. Increase `target_accept` or reparameterize.
There were 4 divergences after tuning. Increase `target_accept` or reparameterize.
There were 3 divergences after tuning. Increase `target_accept` or reparameterize.
There were 6 divergences after tuning. Increase `target_accept` or reparameterize.
There were 3 divergences after tuning. Increase `target_accept` or reparameterize.
There were 6 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.7208447665946961, but should be c
There were 2 divergences after tuning. Increase `target_accept` or reparameterize.
There were 15 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.7153122656222832, but should be c
There were 5 divergences after tuning. Increase `target_accept` or reparameterize.
There were 20 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.7003599021583563, but should be c
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
```

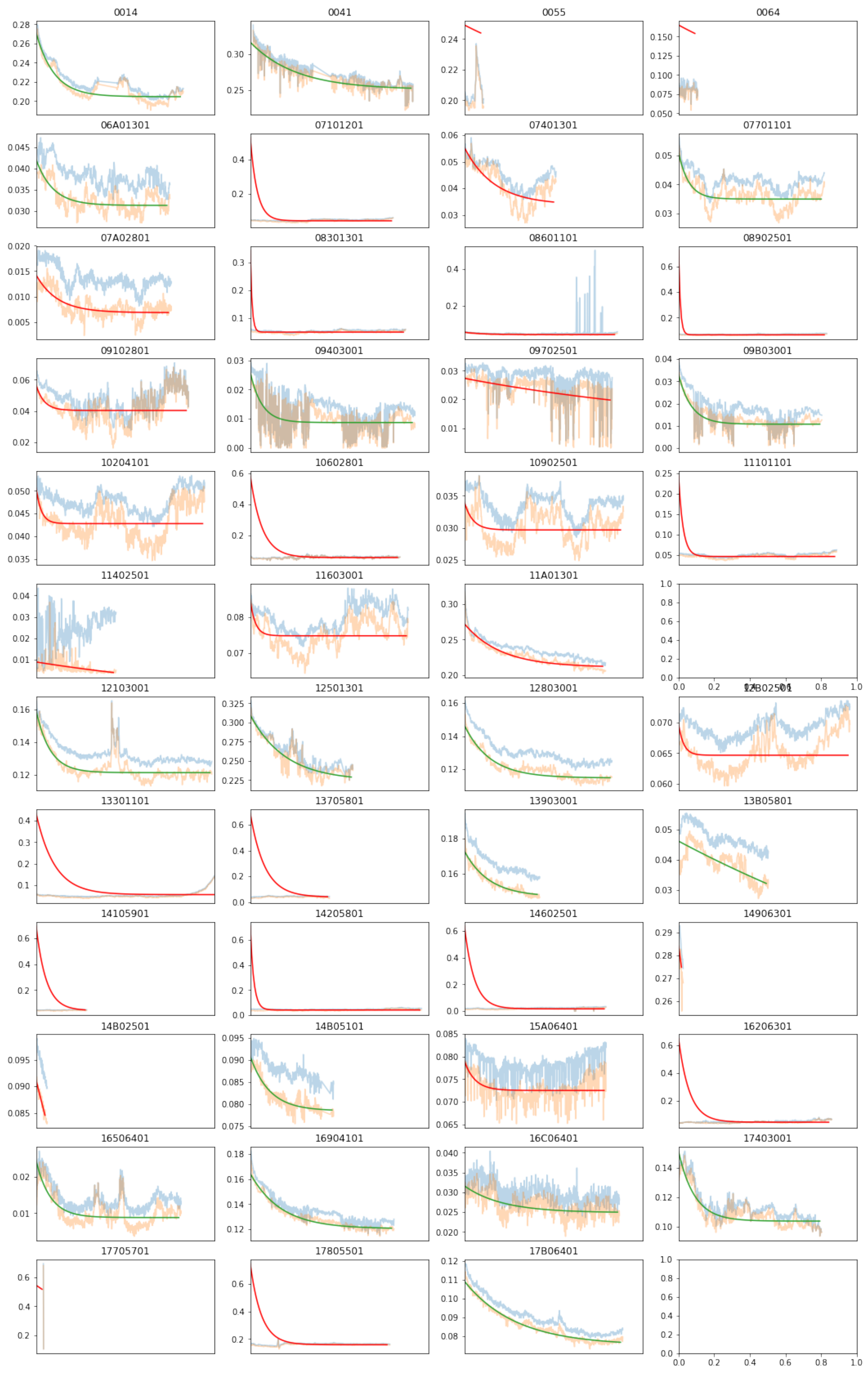
```
NUTS: [sigma, fl_scale, t_scale, fl_0]
There were 192 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.5139046831717842, but should be c
There were 5 divergences after tuning. Increase `target_accept` or reparameterize.
There were 1 divergences after tuning. Increase `target_accept` or reparameterize.
There were 4 divergences after tuning. Increase `target_accept` or reparameterize.
There were 18 divergences after tuning. Increase `target_accept` or reparameterize.
There were 118 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.6237455161661792, but should be c
There were 30 divergences after tuning. Increase `target_accept` or reparameterize.
There were 231 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.5137041000179288, but should be c
There were 1 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.8973331449161646, but should be c
There were 7 divergences after tuning. Increase `target_accept` or reparameterize.
The estimated number of effective samples is smaller than 200 for some parameters.
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
The acceptance probability does not match the target. It is 0.8851017170431659, but should be c
The acceptance probability does not match the target. It is 0.8799702597907219, but should be c
```

```
CUGN_line_93
```

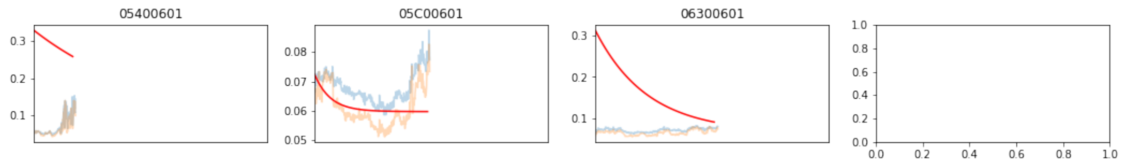
```
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
There were 39 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.6164415144971146, but should be c
The acceptance probability does not match the target. It is 0.9624044474720236, but should be c
There were 5 divergences after tuning. Increase `target_accept` or reparameterize.
There were 25 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.7073399770657034, but should be c
There were 146 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.495562940653721, but should be cl
There were 82 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.5090175564264328, but should be c
There were 27 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.7016249118496476, but should be c
There were 14 divergences after tuning. Increase `target_accept` or reparameterize.
There were 77 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.5610282921647456, but should be c
There were 13 divergences after tuning. Increase `target_accept` or reparameterize.
The gelman-rubin statistic is larger than 1.05 for some parameters. This indicates slight probl
The estimated number of effective samples is smaller than 200 for some parameters.
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, fl_scale, t_scale, fl_0]
There were 49 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.6305367045699944, but should be
There were 6 divergences after tuning. Increase `target_accept` or reparameterize.
There were 50 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.5911197197238363, but should be
There were 4 divergences after tuning. Increase `target_accept` or reparameterize.
There were 4 divergences after tuning. Increase `target_accept` or reparameterize.
There were 63 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.6062945954494582, but should be
There were 42 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.6489921306434533, but should be
The acceptance probability does not match the target. It is 0.9814931153917915, but should be
There were 18 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.7100227072452227, but should be
There were 35 divergences after tuning. Increase `target_accept` or reparameterize.
The acceptance probability does not match the target. It is 0.6984297467513427, but should be
The number of effective samples is smaller than 10% for some parameters.
```


0025
07401401
07A01401
08401401
08703001
08B01401
09301401
09701101
09B01401
10301401
10704101
10A01401
11104101
11501401
11904101
12101401
12404101
12801401
12B04101
13301401
13704101
13A06001
13B04101
14305701
14602801
14901101
17302501
17A02501
18104001


0030
07101101
06200501
06500901
06A00501
07301101
07501201
07801301
07B01201
08101101
08402801
08701301
08B01101
09201301
09502801
09801301
09C02801
10202501
10402501
10403001
10603001
10B01301
11202801
11501301
11602501
11802801
11901101
11C02501
12301101
12602501
12801101
12C02801
13301301
13605701
13A01101
14103001
14501101
14C02801
16903001
17305501
17605801
17A03001
18105801



0014
0041
0055
0064
06A01301
07101201
07401301
07701101
07A02801
08301301
08601101
08902501
09102801
09403001
09702501
09B03001
10204101
10602801
10902501
11101101
11402501
11603001
11A01301
12103001
12501301
12803001
12B02501
13301101
13705801
13903001
13B05801
14105901
14205801
14602501
14906301
14B02501
14B05101
15A06401
16206301
16506401
16904101
16C06401
17403001
17705701
17805501
17B06401



```
In [50]: fl_unbias_exp = []
         for m in np.unique(spray.mission):
             subset = spray.isel(profile_id=spray.mission==m)
             try:
                 coef = fl_exp_fit[m]
                 offset = coef['fl_0'] + coef['fl_scale'] * np.exp(-subset.days / coef['t_scal
                 tmp = subset.fl - offset
             except:
                 tmp = subset.fl_unbias
             tmp = tmp.where(((tmp>0) | tmp.isnull()), 0)
             fl_unbias_exp.append(tmp)

         spray['fl_unbias_exp'] = xr.concat(fl_unbias_exp, dim='profile_id')

         del(fl_unbias_exp)
         del(subset)
         del(coef)
         del(offset)
         del(tmp)

In [51]: del(spray['spray_unbias_exp'])
```

```
        ---------------------------------------------------------------------------

        KeyError                                  Traceback (most recent call last)

        <ipython-input-51-a121b3164f4d> in <module>()
    ----> 1 del(spray['spray_unbias_exp'])


        ~/.virtualenvs/fluorescence/lib/python3.6/site-packages/xarray/core/dataset.py in __de
        895         """Remove a variable from this dataset.
        896         """
    --> 897         del self._variables[key]
        898         self._coord_names.discard(key)
        899
```

```
KeyError: 'spray_unbias_exp'
```

```
In [ ]: spray
```

```
In [ ]: mission_name = '07A01401'
        scale = np.arange(-5, 2.00, 0.25)

        subset = spray.isel(profile_id=spray.mission==mission_name, depth=(spray.depth<500)).sc

        plt.figure(figsize=(18,18))
        plt.contour(subset.fl)

        plt.subplot(4,1,1)
        plt.contourf(np.log(subset.fl), scale, extend='both')
        plt.colorbar()
        plt.gca().invert_yaxis()
        plt.title(mission_name + ': fl')

        plt.subplot(4,1,2)
        plt.contourf(np.log(subset.fl_unbias), scale, extend='both')
        plt.colorbar()
        plt.gca().invert_yaxis()
        plt.title(mission_name + ': fl')

        plt.subplot(4,1,3)
        plt.contourf(np.log(subset.fl_unbias_exp), scale, extend='both')
        plt.colorbar()
        plt.gca().invert_yaxis()
        plt.title(mission_name + ': fl')

        plt.subplot(4,1,4)
        plt.plot(subset.fl.max(dim='depth'), alpha=0.3)
        plt.plot(subset.fl_unbias.max(dim='depth'), alpha=0.3)
        plt.plot(subset.fl_unbias_exp.max(dim='depth'), alpha=0.3)
        plt.title('Maximum fl observed on each profile')
        plt.xlabel('n profile')
        plt.ylabel('maximum fl observed')
        plt.legend()

        plt.savefig('unbiasing-' + mission_name + '.png')
```

```
In [ ]: subset.sortby('datetime')
```

```
In [ ]: scale = np.arange(0, 0.8, 0.05)
        # scale = np.arange(-9, 1.5, 0.5)
        idx = (spray.depth >= 300) & (spray.depth <= 500)
        for experiment_name, grp in spray.isel(depth=idx, profile_id=spray.profile_max_depth >
            plt.figure(figsize=(18,15))
            # fig, axes = plt.subplots(nrows=3, ncols=1, figsize=(18, 15))

            #plt.figure(figsize=(14,5))
            #ax = axes[0]
            plt.subplot(3,1,1)
            #plt.contourf(grp.datetime, grp.depth, grp.fluorescence, scale)
            plt.contourf(grp.fl, scale, extend='max')
            plt.colorbar()
            plt.gca().invert_yaxis()
            plt.title(experiment_name + ': fl')

            # plt.figure(figsize=(14,5))
            plt.subplot(3,1,2)
            #plt.contourf(grp.datetime, grp.depth, grp.fluorescence, scale)
            plt.contourf(grp.fl_unbias, scale, extend='max')
            plt.colorbar()
            plt.gca().invert_yaxis()
            plt.title(experiment_name + ': fl-unbias')

            #plt.figure(figsize=(14,5))
            plt.subplot(3,1,3)
            #plt.contourf(grp.datetime, grp.depth, grp.fluorescence, scale)
            plt.contourf(grp.fl_unbias_exp, scale, extend='max')
            plt.colorbar()
            plt.gca().invert_yaxis()
            plt.title(experiment_name + ': fl-unbias exp')

            plt.savefig('unbias_' + experiment_name + '.png')
```

```
In [ ]: grp
```

```
In [ ]: spray_unbias_exp = xr.concat(output, dim='profile_id')
        spray_unbias_exp
```

```
In [ ]: xr.concat?
```

```
In [ ]: subset.fl_unbias
```

```
In [ ]: tmp = subset.fl - offset
        # Transforming negative values to 0
        tmp = tmp.where(((tmp>0) | tmp.isnull()), 1e-4)
        tmp
```

```
In [ ]: mission_name
```

```
In [ ]: test = spray_deep.isel(profile_id=spray_deep.mission == mission_name)
```

```
In [ ]: test.isel(profile_id=test.profile_id==243037).fl
In [ ]: mission_name

In [ ]: mission

In [ ]: len(mission)

In [ ]: axes

In [ ]: Nfigs = len(np.unique(experiment.mission))
        int(np.ceil(Nfigs/4.))

In [ ]: plt.plot(mission.dt, mission.fl_min)

In [ ]: mission

In [ ]: subset = spray.isel(profile_id=spray.experiment=='CUGN_line_66')
        subset

In [ ]: subset = spray_deep.isel(profile_id=spray_deep.experiment=='CUGN_line_66')

        subset = spray_deep.isel(profile_id=spray_deep.mission=='07A01401')
        subset['time'] = (subset.datetime - subset.datetime.min()) / (np.timedelta64(1, 's') *
        subset = subset.reset_coords()[['ndive', 'dt', 'fl_min']].to_dataframe()
        subset.dropna(inplace=True)
        subset.sort_values(by='ndive', inplace=True)

        X = np.array(subset.dt)
        Y = np.array(subset.fl_min)


        import pymc3 as pm

        basic_model = pm.Model()

        with basic_model:

            # Priors for unknown model parameters
            fl_0 = pm.Normal('fl_0', mu=0, sd=.3)
            # beta = pm.Normal('beta', mu=0, sd=10, shape=2)
            t_scale = pm.HalfNormal('t_scale', sd=100)
            fl_scale = pm.HalfNormal('fl_scale', sd=1)
            sigma = pm.HalfNormal('sigma', sd=1)

            # Expected value of outcome
            mu = fl_0 + fl_scale * np.exp(- X / t_scale)

            # Likelihood (sampling distribution) of observations
            Y_obs = pm.Normal('Y_obs', mu=mu, sd=sigma, observed=Y)
        with basic_model:
            # draw 5000 posterior samples
            trace = pm.sample(1000)

        pm.traceplot(trace)

        summary = pm.summary(trace)
        x = np.arange(int(max(X)))
        y = summary['mean']['fl_0'] + summary['mean']['fl_scale'] * np.exp(-x / summary['mean']

        fig = plt.figure(figsize=(15,8))
        plt.plot(X, Y)
        plt.plot(x, y)
        plt.title("Regression model")
        plt.xlabel("Days since start of the mission")

In [ ]: subset['time'] = (subset.datetime - subset.datetime.min()) / (np.timedelta64(1, 's') *

In [ ]: spray.datetime.shape

In [ ]: spray['dt'] = np.nan * spray.datetime

In [ ]: spray['dt'] = (spray.datetime - spray.datetime.min()) / (np.timedelta64(1, 's') * 8640

        for mission_name, grp in spray.groupby('mission'):
            dt = grp.dt - grp.dt.min()
            dt.reset_coords(drop=True, inplace=True)

            mission, grp

In [ ]: spray.groupby('mission').min().dt

In [ ]: spray.sel(profile_id=grp.profile_id[0]) += 10

In [ ]: spray

In [ ]: def days_since_start(x):
            return (x - x.min()) / (np.timedelta64(1, 's') * 86400)

        spray['dt'] = spray.datetime.groupby('mission').apply(days_since_start)

In [ ]: spray.reset_coords('datetime')['datetime'].groupby('mission').apply(relative_magnitude

In [ ]: spray

In [ ]: spray['dt'][:]

In [ ]: plt.plot(spray.isel(profile_id=spray.mission=='07A01401').dt)
In [ ]: idx = np.array(spray.mission == mission_name)
        # spray['fl_unbias'][:, idx] = spray.fl.isel(profile_id=idx) - bias
        spray['dt'][:, idx]

In [ ]: idx

In [ ]: spray

In [ ]: spray['dt'] = dt

In [ ]: spray

In [ ]: [i for i in grp.profile_id]

In [ ]: idx = spray.mission == m_name
            spray['fl_unbias'][:, idx] = spray.fl.isel(profile_id=idx) - bias

In [ ]: spray

In [ ]: grp

In [ ]: spray.sel(profile_id=grp.profile_id[0])

In [ ]: spray.sel(profile_id=dt.profile_id)

        dt.sel[misison=spray.mission]

In [ ]: t = spray.reset_coords('datetime')['datetime']

        t.groupby('mission').min()

In [ ]: for m in fl_mission_bias:
            bias = float(m)
            m_name = str(m.mission.values)
            idx = spray.mission == m_name
            spray['fl_unbias'][:, idx] = spray.fl.isel(profile_id=idx) - bias

In [ ]: t0 = spray.reset_coords('datetime').groupby('mission').min()['datetime']

In [ ]: spray.datetime - t0

In [ ]: ds

In [ ]: # Change this to a sequence of PDFs per level, or maybe violin plots

        fig = plt.figure(figsize=(8,7))
        #for label, p in ds.groupby('profileid'):
        #    plt.plot(p.fluorescence, ds.depth, '.', color='g', alpha=0.1)

        plt.plot(ds.fluorescence, ds.depth, '.', color='g', alpha=0.1)
        plt.gca().invert_yaxis()
```

The simplest approach to correct for the offset is to assume that every profile should measure at least once a zero concentration, and a second assumption is that the sensor cannot output less than an equivalent to zero.

Let's subtract each profile by its minimum value observed.

```
In [ ]: #ds['fl0'] = ds.fluorescence - ds.fl_min
```

```
In [ ]: fig = plt.figure(figsize=(8,7))
        #for label, p in ds.groupby('profileid'):
        #    plt.plot(p.fl_unbias, ds.depth, '.', color='g', alpha=0.1)

        plt.plot(ds.fl_unbias, ds.depth, '.', color='g', alpha=0.1)
        plt.gca().invert_yaxis()
```

8 Gain: How to scale?

The idea here is that the Spray measurements should be coherent with the satellite reading. Let's compare with MODIS-Aqua.

In this dataset are all matchups that I found between Spray profiles and MODIS-Aqua L2. The L2 level means all calibrations and corrections were applied, but the data was not interpolated, therefore, these satellite values are the instantaneous readings from Aqua. The next alternative are daily or 8-day regularly binned, which changes the time scale of the satellite data, truncating all higher frequency. Working with L2 avoid the requirement on assumptions on timescales included in the samples.

A matchup is the nearest (in distance) pixel from the closest (in time) swath from Aqua. The limits allowed were 24hrs and 30 km difference. These are probably wider than desired, but the matchup process is a little slow so it is better to allow more matchups and filter them here before fit the corrections.

```
In [ ]: ds['fl_max'] = ds.fl_unbias.max(dim='depth')
        h = plt.hist(ds.fl_max, bins=50)
        ds.fl_max.to_series().describe()
```

```
In [ ]: valid_only = ds[['fl_max', 'chlor_a']].dropna(dim='profileid', how='any')


        # the random data
        x = valid_only.fl_max
        y = valid_only.chlor_a

        nullfmt = NullFormatter()         # no labels

        # definitions for the axes
        left, width = 0.1, 0.65
        bottom, height = 0.1, 0.65
        bottom_h = left_h = left + width + 0.02

        rect_scatter = [left, bottom, width, height]
        rect_histx = [left, bottom_h, width, 0.2]
        rect_histy = [left_h, bottom, 0.2, height]

        # start with a rectangular Figure
        plt.figure(1, figsize=(8, 8))

        axScatter = plt.axes(rect_scatter)
        axHistx = plt.axes(rect_histx)
        axHisty = plt.axes(rect_histy)

        # no labels
        axHistx.xaxis.set_major_formatter(nullfmt)
        axHisty.yaxis.set_major_formatter(nullfmt)

        # the scatter plot:
        axScatter.scatter(x, y, alpha=.2)

        # now determine nice limits by hand:
        binwidth = 0.25
        xymax = 10
        lim = (int(xymax/binwidth) + 1) * binwidth

        axScatter.set_xlim((-binwidth, lim))
        axScatter.set_ylim((-binwidth, lim))

        bins = np.arange(-binwidth, lim + binwidth, binwidth)
        axHistx.hist(x, bins=bins)
        axHisty.hist(y, bins=bins, orientation='horizontal')

        axHistx.set_xlim(axScatter.get_xlim())
        axHisty.set_ylim(axScatter.get_ylim())
```

```
In [ ]: ds['fl_sum'] = ds.fl_unbias.isel(depth=slice(8)).sum(axis=0)
```

```
In [ ]: spray
```

```
In [ ]: valid_only = ds[['fl_sum', 'chlor_a']].dropna(dim='profileid', how='any')


        # the random data
        x = valid_only.fl_sum
        y = valid_only.chlor_a

        nullfmt = NullFormatter()         # no labels

        # definitions for the axes
        left, width = 0.1, 0.65
        bottom, height = 0.1, 0.65
        bottom_h = left_h = left + width + 0.02

        rect_scatter = [left, bottom, width, height]
        rect_histx = [left, bottom_h, width, 0.2]
        rect_histy = [left_h, bottom, 0.2, height]

        # start with a rectangular Figure
        plt.figure(1, figsize=(8, 8))

        axScatter = plt.axes(rect_scatter)
        axHistx = plt.axes(rect_histx)
        axHisty = plt.axes(rect_histy)

        # no labels
        axHistx.xaxis.set_major_formatter(nullfmt)
        axHisty.yaxis.set_major_formatter(nullfmt)

        # the scatter plot:
        axScatter.scatter(x, y, alpha=.2)

        # now determine nice limits by hand:
        binwidth = 0.25
        xymax = 10
        lim = (int(xymax/binwidth) + 1) * binwidth

        axScatter.set_xlim((-binwidth, lim))
        axScatter.set_ylim((-binwidth, lim))

        bins = np.arange(-binwidth, lim + binwidth, binwidth)
        axHistx.hist(x, bins=bins)
        axHisty.hist(y, bins=bins, orientation='horizontal')

        axHistx.set_xlim(axScatter.get_xlim())
        axHisty.set_ylim(axScatter.get_ylim())
```

```
In [ ]: np.max([np.max(np.fabs(x)), np.max(np.fabs(y))])
        np.fabs(x)
```

Let's find exact localtime to define if measurement was done on daylight or night time. This will be important for the NPQ correction.

```
In [ ]: assert ds.longitude.max() <= 180, "I'm assuming longitudes between -180 to 180"

        tmp = ds[['datetime', 'longitude']].to_dataframe()[['datetime', 'longitude']]
        ds['localtime'] = tmp.apply(lambda row: row['datetime'] + timedelta(hours=row['longitu
```

```
In [ ]: subset = spray_deep.isel(profile_id=spray_deep.experiment=='CUGN_line_66')
```

```
In [ ]: sorted(subset.mission.to_series().unique())
In [ ]: subset = spray_deep.isel(profile_id=spray_deep.mission=='07A01401')
        subset['time'] = (subset.datetime - subset.datetime.min()) / (np.timedelta64(1, 's') *
        subset = subset.reset_coords()[['ndive', 'time', 'fl_min']].to_dataframe()
        subset.dropna(inplace=True)
        subset.sort_values(by='ndive', inplace=True)

        fig = plt.figure(figsize=(15,8))
        plt.plot(subset.time, subset.fl_min)
```

```
In [ ]: X = np.array(subset.time)
        Y = np.array(subset.fl_min)
```

```
In [ ]: fig = plt.figure(figsize=(15,8))
        #plt.plot(X, Y)

        alpha = 0.12
        beta = 14.02

        fl_min_post = 0.112 + 0.069 * np.exp(-X / 10.757)
        plt.plot(X, fl_min_post)
```

```
In [ ]: import pymc3 as pm

        basic_model = pm.Model()

        with basic_model:

            # Priors for unknown model parameters
            alpha = pm.Normal('alpha', mu=0, sd=.3)
            # beta = pm.Normal('beta', mu=0, sd=10, shape=2)
            beta = pm.HalfNormal('beta', sd=100)
            scale = pm.HalfNormal('scale', sd=1)
            sigma = pm.HalfNormal('sigma', sd=1)

            # Expected value of outcome
            mu = alpha + scale * np.exp(- X / beta)

            # Likelihood (sampling distribution) of observations
            Y_obs = pm.Normal('Y_obs', mu=mu, sd=sigma, observed=Y)

            # draw 500 posterior samples
            # trace = pm.sample(500)
```

```
In [ ]: map_estimate = pm.find_MAP(model=basic_model)

        map_estimate
```

```
In [ ]: with basic_model:
            # draw 500 posterior samples
            trace = pm.sample(5000)
```

```
In [ ]: pm.traceplot(trace)

In [ ]: pm.summary(trace).round(3)

In [ ]: summary = pm.summary(trace)

In [ ]: np.percentile(Y, 10)

In [ ]: summary = pm.summary(trace)
        x = np.arange(int(max(X)))
        y = summary['mean']['alpha'] + summary['mean']['scale'] * np.exp(-x / summary['mean'][

        plt.plot(X, Y)
        plt.plot(x, y)

In [ ]: Y.std()

In [ ]: plt.contourf(subset.fl)
        plt.colorbar()

In [ ]: import numpy as np
        import matplotlib.pyplot as plt
        plt.style.use('seaborn-darkgrid')

        # Initialize random number generator
        np.random.seed(123)

        # True parameter values
        alpha, sigma = 1, 1
        beta = [1, 2.5]

        # Size of dataset
        size = 100

        # Predictor variable
        X1 = np.random.randn(size)
        X2 = np.random.randn(size) * 0.2

        # Simulate outcome variable
        Y = alpha + beta[0]*X1 + beta[1]*X2 + np.random.randn(size)*sigma

In [ ]: fig, axes = plt.subplots(1, 2, sharex=True, figsize=(10,4))
        axes[0].scatter(X1, Y)
        axes[1].scatter(X2, Y)
        axes[0].set_ylabel('Y'); axes[0].set_xlabel('X1'); axes[1].set_xlabel('X2');

In [ ]: import pymc3 as pm
        print('Running on PyMC3 v{}'.format(pm.__version__))
```

```
In [ ]: basic_model = pm.Model()

        with basic_model:

            # Priors for unknown model parameters
            alpha = pm.Normal('alpha', mu=0, sd=10)
            beta = pm.Normal('beta', mu=0, sd=10, shape=2)
            sigma = pm.HalfNormal('sigma', sd=1)

            # Expected value of outcome
            mu = alpha + beta[0]*X1 + beta[1]*X2

            # Likelihood (sampling distribution) of observations
            Y_obs = pm.Normal('Y_obs', mu=mu, sd=sigma, observed=Y)

In [ ]: map_estimate = pm.find_MAP(model=basic_model)

        map_estimate

In [ ]: map_estimate = pm.find_MAP(model=basic_model, method='powell')

        map_estimate

In [ ]: with basic_model:
            # draw 500 posterior samples
            trace = pm.sample(500)

In [ ]: trace['alpha'][-5:]

In [ ]: with basic_model:

            # instantiate sampler
            step = pm.Slice()

            # draw 5000 posterior samples
            trace = pm.sample(5000, step=step)

In [ ]: pm.traceplot(trace);

In [ ]: pm.summary(trace).round(2)

In [ ]: from pandas_datareader import data

In [ ]: pm.Exponential?

In [ ]: import matplotlib.pyplot as plt
        import numpy as np
        import scipy.stats as st
        plt.style.use('seaborn-darkgrid')
        x = np.linspace(0, 3, 100)
        for lam in [0.5, 1., 2.]:
            pdf = st.expon.pdf(x, scale=1.0/lam)
            plt.plot(x, pdf, label=r'$\lambda$ = {}'.format(lam))
        plt.xlabel('x', fontsize=12)
        plt.ylabel('f(x)', fontsize=12)
        plt.legend(loc=1)
        plt.show()
```

```
In [ ]: IQR/1.349
```

```
In [299]: dL_agg = 10e3
          # satellites = ['seawifs', 'aqua', 'terra', 'viirs']
          satellites = ['aqua', 'terra', 'viirs']


          matchup = []
          for satname in satellites:
              inrange = pd.read_hdf(inputFilename, key=satname)


              grp = inrange[inrange.dL<=dL_agg].groupby('profile_id')
              tmp = pd.DataFrame({
                  'sat': satname,
                  'chl_count': grp.chlor_a.count(),
                  'chl_mean': grp.chlor_a.mean(),
                  'chl_median': grp.chlor_a.median(),
                  'chl_psdstd': (grp.chlor_a.quantile(.75) - grp.chlor_a.quantile(.25))/1.349,
                  'chl_std': grp.chlor_a.std(),
                  'chl_sem': grp.chlor_a.sem()})
              matchup.append(tmp.set_index('sat', append=True).to_xarray())

          matchup = xr.merge(matchup, join='outer')
          print(matchup)

          del(inrange)
          del(grp)
          del(tmp)
```

```
<xarray.Dataset>
Dimensions:     (profile_id: 67004, sat: 3)
Coordinates:
  * profile_id  (profile_id) int64 793 797 798 800 803 805 809 816 820 823 ...
  * sat         (sat) object 'aqua' 'terra' 'viirs'
Data variables:
    chl_count   (profile_id, sat) float64 12.0 11.0 13.0 12.0 11.0 13.0 13.0 ...
    chl_mean    (profile_id, sat) float32 0.7363032 0.80052286 0.42922863 ...
    chl_median  (profile_id, sat) float32 0.6272783 0.53259164 0.4206165 ...
    chl_psdstd  (profile_id, sat) float64 0.1824 0.4808 0.05015 0.1824 ...
    chl_std     (profile_id, sat) float32 0.30510858 0.40528625 0.056684013 ...
    chl_sem     (profile_id, sat) float64 0.08808 0.1222 0.01572 0.08808 ...
```

```
In [300]: spray
```

```
Out[300]: <xarray.Dataset>
          Dimensions:             (depth: 100, profile_id: 96718, sat: 3)
          Coordinates:
            * profile_id          (profile_id) int64 20215 20216 20217 20218 20219 ...
              ndive               (profile_id) int64 1 2 3 4 5 6 7 8 9 10 11 12 13 14 ...
              datetime            (profile_id) datetime64[ns] 2006-10-16T19:46:51 ...
              lat                 (profile_id) float64 34.35 34.35 34.35 34.34 34.34 ...
              lon                 (profile_id) float64 -119.8 -119.8 -119.8 -119.8 ...
              mission_id          (profile_id) int64 106 106 106 106 106 106 106 106 ...
              mission             (profile_id) object '06A00501' '06A00501' '06A00501' ...
              experiment_id       (profile_id) int64 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 ...
              experiment          (profile_id) object 'CUGN_line_80' 'CUGN_line_80' ...
            * depth               (depth) float64 10.0 20.0 30.0 40.0 50.0 60.0 70.0 ...
              days                (profile_id) float64 0.0 0.02537 0.04983 0.07683 ...
              night_time          (profile_id) bool True True True True False False ...
            * sat                 (sat) object 'aqua' 'terra' 'viirs'
          Data variables:
              temp                (depth, profile_id) float64 16.57 16.38 16.24 16.16 ...
              sal                 (depth, profile_id) float64 33.42 33.41 33.43 33.43 ...
              fl                  (depth, profile_id) float64 1.382 1.23 1.429 2.101 ...
              fl_min              (profile_id) float64 0.2884 0.2508 0.2347 0.1904 ...
              profile_max_depth   (profile_id) float64 60.0 80.0 80.0 90.0 110.0 150.0 ...
              fl_deep_ref         (profile_id) float64 nan nan nan nan nan nan nan nan ...
              chl_count           (profile_id, sat) float64 22.0 37.0 nan 22.0 37.0 nan ...
              chl_mean            (profile_id, sat) float32 1.9567965 0.9745653 nan ...
              chl_median          (profile_id, sat) float32 1.5632914 0.7632589 nan ...
              chl_psdstd          (profile_id, sat) float64 0.6793 0.353 nan 0.6793 ...
              chl_std             (profile_id, sat) float32 1.0249827 0.32767114 nan ...
              chl_sem             (profile_id, sat) float64 0.2185 0.05387 nan 0.2185 ...
              fl_sum              (profile_id) float64 39.29 41.99 45.07 53.51 63.31 ...
```

```
In [301]: spray = spray.merge(matchup, join='left')
```

```
        ---------------------------------------------------------------------------

        MergeError                                Traceback (most recent call last)

        <ipython-input-301-100bd7d90eda> in <module>()
    ----> 1 spray = spray.merge(matchup, join='left')
        ~/.virtualenvs/fluorescence/lib/python3.6/site-packages/xarray/core/dataset.py in merge
       2233         variables, coord_names, dims = dataset_merge_method(
       2234             self, other, overwrite_vars=overwrite_vars, compat=compat,
    -> 2235             join=join)
       2236 
       2237         return self._replace_vars_and_dims(variables, coord_names, dims,


        ~/.virtualenvs/fluorescence/lib/python3.6/site-packages/xarray/core/merge.py in datase
        544         priority_arg = 2
        545 
    --> 546     return merge_core(objs, compat, join, priority_arg=priority_arg)
        547 
        548 


        ~/.virtualenvs/fluorescence/lib/python3.6/site-packages/xarray/core/merge.py in merge_
        428 
        429     priority_vars = _get_priority_vars(aligned, priority_arg, compat=compat)
    --> 430     variables = merge_variables(expanded, priority_vars, compat=compat)
        431     assert_unique_multiindex_level_names(variables)
        432 


        ~/.virtualenvs/fluorescence/lib/python3.6/site-packages/xarray/core/merge.py in merge_
        162             else:
        163                 try:
    --> 164                     merged[name] = unique_variable(name, variables, compat)
        165                 except MergeError:
        166                     if compat != 'minimal':


        ~/.virtualenvs/fluorescence/lib/python3.6/site-packages/xarray/core/merge.py in unique_
         81                                  'objects to be combined:\n'
         82                                  'first value: %r\nsecond value: %r'
    ---> 83                                  % (name, out, var))
         84             if combine_method:
         85                 # TODO: add preservation of attrs into fillna


        MergeError: conflicting values for variable 'chl_count' on objects to be combined:
    first value: <xarray.Variable (profile_id: 96718, sat: 3)>
    array([[22., 37., nan],
           [22., 37., nan],
           [22., 37., nan],
           ...,
           [34., 49., 44.],
           [63., 21., 71.],
           [41., 19., 47.]])
    second value: <xarray.Variable (profile_id: 96718, sat: 3)>
    array([[14., 25., nan],
           [15., 27., nan],
           [16., 27., nan],
           ...,
           [23., 34., 29.],
           [44., 14., 48.],
           [29., 12., 32.]])
```

```
In [73]: # spray = xr.merge([profile.to_xarray(), matchup, data.to_xarray()], join='left')
```

```
        ---------------------------------------------------------------------------

        NameError                                 Traceback (most recent call last)

        <ipython-input-73-59975c1ad960> in <module>()
    ----> 1 spray = xr.merge([profile.to_xarray(), matchup, data.to_xarray()], join='left')


        NameError: name 'data' is not defined
```

```
In [265]: spray
```

```
Out[265]: <xarray.Dataset>
          Dimensions:            (depth: 100, profile_id: 96718, sat: 3)
          Coordinates:
            * profile_id         (profile_id) int64 20215 20216 20217 20218 20219 ...
              ndive              (profile_id) int64 1 2 3 4 5 6 7 8 9 10 11 12 13 14 ...
              datetime           (profile_id) datetime64[ns] 2006-10-16T19:46:51 ...
              lat                (profile_id) float64 34.35 34.35 34.35 34.34 34.34 ...
              lon                (profile_id) float64 -119.8 -119.8 -119.8 -119.8 ...
              mission_id         (profile_id) int64 106 106 106 106 106 106 106 106 ...
              mission            (profile_id) object '06A00501' '06A00501' '06A00501' ...
              experiment_id      (profile_id) int64 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 ...
              experiment         (profile_id) object 'CUGN_line_80' 'CUGN_line_80' ...
            * depth              (depth) float64 10.0 20.0 30.0 40.0 50.0 60.0 70.0 ...
              days               (profile_id) float64 0.0 0.02537 0.04983 0.07683 ...
              night_time         (profile_id) bool True True True True False False ...
            * sat                (sat) object 'aqua' 'terra' 'viirs'
          Data variables:
              temp               (depth, profile_id) float64 16.57 16.38 16.24 16.16 ...
              sal                (depth, profile_id) float64 33.42 33.41 33.43 33.43 ...
              fl                 (depth, profile_id) float64 1.382 1.23 1.429 2.101 ...
              fl_min             (profile_id) float64 0.2884 0.2508 0.2347 0.1904 ...
              profile_max_depth  (profile_id) float64 60.0 80.0 80.0 90.0 110.0 150.0 ...
              fl_deep_ref        (profile_id) float64 nan nan nan nan nan nan nan nan ...
              chl_count          (profile_id, sat) float64 22.0 37.0 nan 22.0 37.0 nan ...
              chl_mean           (profile_id, sat) float32 1.9567965 0.9745653 nan ...
              chl_median         (profile_id, sat) float32 1.5632914 0.7632589 nan ...
              chl_psdstd         (profile_id, sat) float64 0.6793 0.353 nan 0.6793 ...
              chl_std            (profile_id, sat) float32 1.0249827 0.32767114 nan ...
              chl_sem            (profile_id, sat) float64 0.2185 0.05387 nan 0.2185 ...
```

```
In [266]: np.diff(np.concatenate(([5], spray.depth)))
```

```
Out[266]: array([ 5., 10., 10., 10., 10., 10., 10., 10., 10., 10., 10., 10., 10.,
                 10., 10., 10., 10., 10., 10., 10., 10., 10., 10., 10., 10., 10.,
                 10., 10., 10., 10., 10., 10., 10., 10., 10., 10., 10., 10., 10.,
                 10., 10., 10., 10., 10., 10., 10., 10., 10., 10., 10., 10., 10.,
                 10., 10., 10., 10., 10., 10., 10., 10., 10., 10., 10., 10., 10.,
                 10., 10., 10., 10., 10., 10., 10., 10., 10., 10., 10., 10., 10.,
                 10., 10., 10., 10., 10., 10., 10., 10., 10., 10., 10., 10., 10.,
                 10., 10., 10., 10., 10., 10., 10., 10., 10.])
```

```
In [267]: # spray['fl_sum'] = 5 * spray.fl_unbias_exp.sel(depth=10).reset_coords(drop=True) \
          #                    + 10 * spray.fl_unbias_exp.isel(depth=spray.depth<=30).sum(dim='

          # plt.figure(figsize=(15,4))
          # plt.plot(spray.fl_sum)
          #spray['fl_sum'] = 5 * spray.fl_unbias.sel(depth=10).reset_coords(drop=True) \
          #                   + 10 * spray.fl_unbias.isel(depth=spray.depth<=20).sum(dim='depth
          spray['fl_sum'] = 5 * spray.fl.sel(depth=10).reset_coords(drop=True) \
                              + 10 * spray.fl.isel(depth=spray.depth<=20).sum(dim='depth')
          # plt.plot(spray.fl_sum)

          spray
```

```
Out[267]: <xarray.Dataset>
          Dimensions:        (depth: 100, profile_id: 96718, sat: 3)
          Coordinates:
            * profile_id     (profile_id) int64 20215 20216 20217 20218 20219 ...
              ndive          (profile_id) int64 1 2 3 4 5 6 7 8 9 10 11 12 13 14 ...
              datetime       (profile_id) datetime64[ns] 2006-10-16T19:46:51 ...
              lat            (profile_id) float64 34.35 34.35 34.35 34.34 34.34 ...
              lon            (profile_id) float64 -119.8 -119.8 -119.8 -119.8 ...
              mission_id     (profile_id) int64 106 106 106 106 106 106 106 106 ...
              mission        (profile_id) object '06A00501' '06A00501' '06A00501' ...
              experiment_id  (profile_id) int64 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 ...
              experiment     (profile_id) object 'CUGN_line_80' 'CUGN_line_80' ...
            * depth          (depth) float64 10.0 20.0 30.0 40.0 50.0 60.0 70.0 ...
              days           (profile_id) float64 0.0 0.02537 0.04983 0.07683 ...
              night_time     (profile_id) bool True True True True False False ...
            * sat            (sat) object 'aqua' 'terra' 'viirs'
          Data variables:
```

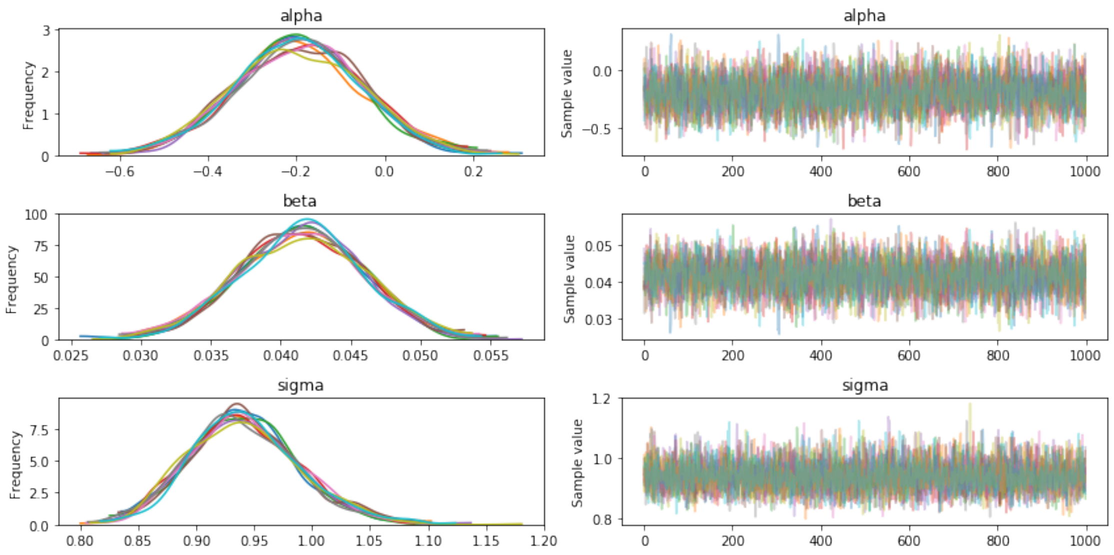
```
    temp               (depth, profile_id) float64 16.57 16.38 16.24 16.16 ...
    sal                (depth, profile_id) float64 33.42 33.41 33.43 33.43 ...
    fl                 (depth, profile_id) float64 1.382 1.23 1.429 2.101 ...
    fl_min             (profile_id) float64 0.2884 0.2508 0.2347 0.1904 ...
    profile_max_depth  (profile_id) float64 60.0 80.0 80.0 90.0 110.0 150.0 ...
    fl_deep_ref        (profile_id) float64 nan nan nan nan nan nan nan nan ...
    chl_count          (profile_id, sat) float64 22.0 37.0 nan 22.0 37.0 nan ...
    chl_mean           (profile_id, sat) float32 1.9567965 0.9745653 nan ...
    chl_median         (profile_id, sat) float32 1.5632914 0.7632589 nan ...
    chl_psdstd         (profile_id, sat) float64 0.6793 0.353 nan 0.6793 ...
    chl_std            (profile_id, sat) float32 1.0249827 0.32767114 nan ...
    chl_sem            (profile_id, sat) float64 0.2185 0.05387 nan 0.2185 ...
    fl_sum             (profile_id) float64 39.29 41.99 45.07 53.51 63.31 ...
```

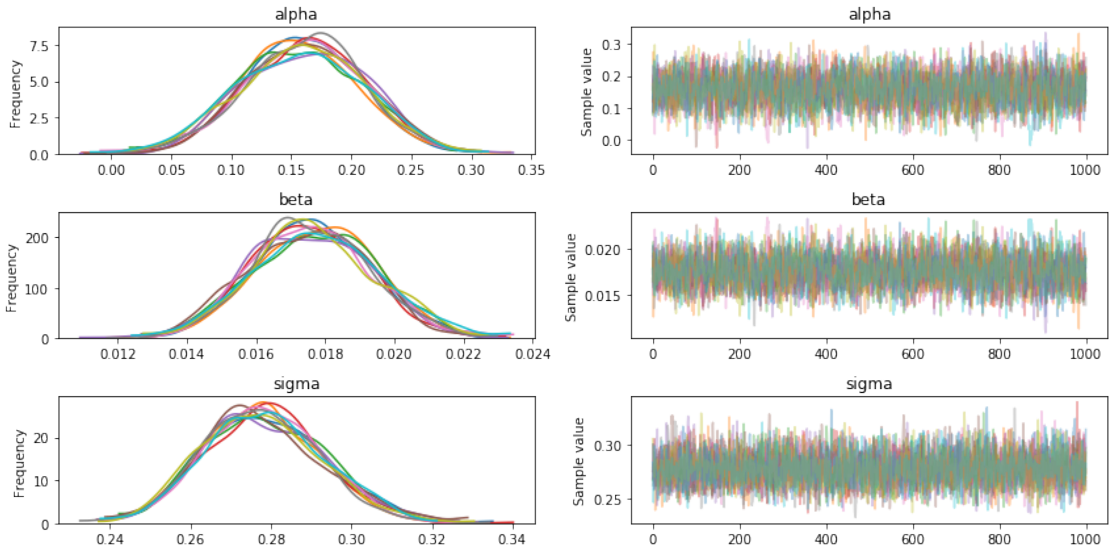
```
In [166]: import pymc3 as pm

          def fit_fl(fl, chl):

              basic_model = pm.Model()

              with basic_model:

                  # Priors for unknown model parameters
                  alpha = pm.Normal('alpha', mu=0, sd=10)
                  beta = pm.Normal('beta', mu=1, sd=2)
                  sigma = pm.HalfNormal('sigma', sd=1)

                  # Expected value of outcome
                  mu = alpha + beta * fl

                  # Likelihood (sampling distribution) of observations
                  Y_obs = pm.Normal('Y_obs', mu=mu, sd=sigma, observed=chl)

              with basic_model:
                  trace = pm.sample(draws=1000, chains=10, tune=1000, progressbar=False)


              return trace
```

```
In [167]: subset = spray.sel(sat='aqua').isel(profile_id=(spray.night_time & (spray.mission=='
          #subset = spray.sel(sat='aqua').isel(profile_id=spray.night_time)[['fl_sum', 'chl_me

          X = np.array(subset.fl_sum)
          Y = np.array(subset.chl_mean)

          trace = fit_fl(fl=X, chl=Y)
          pm.traceplot(trace)
          summary = pm.summary(trace)
```

```
        print(summary)

        idx = subset['chl_std'] <= 1
        trace = fit_fl(fl=X[idx], chl=Y[idx])
        pm.traceplot(trace)
        summary = pm.summary(trace)
        print(summary)
```

```
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, beta, alpha]


           mean        sd  mc_error   hpd_2.5  hpd_97.5        n_eff      Rhat
alpha -0.195433  0.144366  0.002178 -0.487110  0.084290  4592.565076  0.999829
beta   0.041439  0.004486  0.000067  0.032963  0.050481  4559.957239  0.999678
sigma  0.941224  0.047132  0.000648  0.851075  1.035451  6249.445372  1.000303


Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, beta, alpha]
The acceptance probability does not match the target. It is 0.8867672329026418, but should be


           mean        sd  mc_error   hpd_2.5  hpd_97.5        n_eff      Rhat
alpha  0.158631  0.050097  0.000856  0.065581  0.258887  4056.363813  1.001221
beta   0.017620  0.001738  0.000030  0.014175  0.020895  4093.622166  1.001412
sigma  0.277965  0.014723  0.000170  0.250835  0.307381  6620.182915  1.000418
```





```
In [178]: import pymc3 as pm

          def fit_fl(fl, chl, chl_std=0):

              basic_model = pm.Model()

              with basic_model:

                  # Priors for unknown model parameters
                  alpha = pm.Normal('alpha', mu=0, sd=10)
                  beta = pm.Normal('beta', mu=1, sd=1)
                  sigma = pm.HalfNormal('sigma', sd=2)

                  # Expected value of outcome
                  mu = alpha + beta * fl

                  sd = sigma + chl_std

                  # Likelihood (sampling distribution) of observations
                  Y_obs = pm.Normal('Y_obs', mu=mu, sd=sd, observed=chl)

              with basic_model:
                  trace = pm.sample(draws=1000, chains=10, tune=1000, progressbar=False)

              return trace
```

```
In [170]: idx = (subset.chl_mean + subset.chl_sem) < 2

          subset.fl_sum[idx]

Out[170]: <xarray.DataArray 'fl_sum' (profile_id: 148)>
          array([ 13.602083,  12.418125,  10.650536,  15.23619 ,  12.349048,   5.79    ,
                   6.125625,  18.025625,  15.643   ,   6.6825  ,   3.218333,   3.55875 ,
                  20.97    ,  16.972333,   8.147381,   3.822708,  29.503125,  27.607708,
                  25.423125,  40.810714,  33.507857,  17.388125,  16.98125 ,  46.351875,
                  45.332917,  28.843333,  30.218571,  43.47375 ,  29.435   ,  21.886167,
                  35.250208,  52.945238,  81.06675 ,  47.89875 , 131.995   ,  83.071667,
                  26.679107,  31.538571,  34.0545  ,  28.843929,  51.703   ,  33.093333,
                  26.418   ,  24.76    ,  42.49875 ,  24.256667,  16.360417,  37.154792,
                  36.787857,  23.61375 ,  20.107292,  20.7075  ,  76.458333,  40.175833,
                  42.395417,  50.548333,  53.52   ,  43.742   ,  22.62625 ,  39.54675 ,
                  24.714   ,  30.321   ,  44.997333,  24.569   ,  15.0725  ,  42.30075 ,
                  16.378333,   7.277   ,   4.421   ,   9.124875,   4.297879,   3.48    ,
                  21.05489 ,  14.054464,   3.6    ,   3.617473,  23.774375,   4.855385,
                   5.6475  ,  17.5415  ,  26.841333,  13.575833,  12.770114,  29.194545,
                  14.524602,   9.340455,   8.21   ,  22.554167,   9.57    ,   6.479   ,
                  29.432   ,  14.06    ,   9.560182,  19.427045,   7.311818,   5.7725  ,
                  13.356   ,  58.108333,  61.376625,  58.84    ,  54.908333,  49.441875,
                  57.380455,  70.016   ,  62.424955,  54.750952,  56.277   ,  60.001875,
                  55.028333,  46.842273,  64.998333,  49.907333,  46.721667,  52.1775  ,
                  39.094286,  34.095   ,  30.867857,  31.2075  ,  22.479375,  18.763125,
                  11.907321,  23.975833,  23.247857,  16.411458,  11.331429,  21.72375 ,
                  18.470357,  33.275   ,  25.566429,  13.666042,  11.974286,  27.085714,
                  12.703125,   9.809167,  15.179167,   9.50625 ,   9.112083,   7.145625,
                  17.385833,  15.451667,   8.422917,   5.803125,  15.333333,  15.879167,
                   8.1325  ,   6.851667,  16.      ,  15.13    ])
          Coordinates:
              mission        (profile_id) object '08401401' '08401401' '08401401' ...
              experiment_id  (profile_id) int64 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 ...
              quantile       float64 0.001
              mission_id     (profile_id) int64 67 67 67 67 67 67 67 67 67 67 67 67 67 ...
              ndive          (profile_id) int64 198 199 200 206 207 208 209 215 216 ...
              days           (profile_id) float64 20.9 21.01 21.11 21.78 21.89 22.0 ...
            * profile_id     (profile_id) int64 117066 117073 117080 117121 117128 ...
              lat            (profile_id) float64 34.98 34.97 34.97 34.9 34.88 34.87 ...
              sat            <U4 'aqua'
              datetime       (profile_id) datetime64[ns] 2008-04-22T14:52:50 ...
              experiment     (profile_id) object 'CUGN_line_66' 'CUGN_line_66' ...
              lon            (profile_id) float64 -125.9 -125.9 -126.0 -126.1 -126.1 ...
              night_time     (profile_id) bool True True True True True True True True ...

In [277]: subset

Out[277]: <xarray.Dataset>
          Dimensions:        (profile_id: 210)
          Coordinates:
              mission        (profile_id) object '07401401' '07401401' '07401401' ...
              experiment_id  (profile_id) int64 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 ...
              days           (profile_id) float64 1.92 2.028 2.136 2.779 2.888 2.999 ...
              mission_id     (profile_id) int64 65 65 65 65 65 65 65 65 65 65 65 65 65 ...
              ndive          (profile_id) int64 22 23 24 30 31 32 33 34 39 40 41 42 43 ...
            * profile_id     (profile_id) int64 28094 28095 28096 28102 28103 28104 ...
              lat            (profile_id) float64 36.7 36.7 36.69 36.63 36.61 36.6 ...
              sat            <U4 'aqua'
              datetime       (profile_id) datetime64[ns] 2007-04-21T14:30:48 ...
              experiment     (profile_id) object 'CUGN_line_66' 'CUGN_line_66' ...
              lon            (profile_id) float64 -122.4 -122.4 -122.4 -122.6 -122.6 ...
              night_time     (profile_id) bool True True True True True True True True ...
          Data variables:
              fl_sum         (profile_id) float64 26.5 16.44 23.97 22.09 16.32 18.68 ...
              chl_mean       (profile_id) float32 0.5286218 0.5305221 0.5216202 ...
              chl_std        (profile_id) float32 0.08720115 0.0790803 0.07133923 ...
              chl_count      (profile_id) float64 24.0 21.0 20.0 10.0 37.0 37.0 41.0 ...
              chl_sem        (profile_id) float64 0.0178 0.01726 0.01595 0.02478 ...

In [278]: spray

Out[278]: <xarray.Dataset>
          Dimensions:            (depth: 100, profile_id: 96718, sat: 3)
          Coordinates:
            * profile_id         (profile_id) int64 20215 20216 20217 20218 20219 ...
              ndive              (profile_id) int64 1 2 3 4 5 6 7 8 9 10 11 12 13 14 ...
              datetime           (profile_id) datetime64[ns] 2006-10-16T19:46:51 ...
              lat                (profile_id) float64 34.35 34.35 34.35 34.34 34.34 ...
              lon                (profile_id) float64 -119.8 -119.8 -119.8 -119.8 ...
              mission_id         (profile_id) int64 106 106 106 106 106 106 106 106 ...
              mission            (profile_id) object '06A00501' '06A00501' '06A00501' ...
              experiment_id      (profile_id) int64 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 ...
              experiment         (profile_id) object 'CUGN_line_80' 'CUGN_line_80' ...
            * depth              (depth) float64 10.0 20.0 30.0 40.0 50.0 60.0 70.0 ...
              days               (profile_id) float64 0.0 0.02537 0.04983 0.07683 ...
              night_time         (profile_id) bool True True True True False False ...
            * sat                (sat) object 'aqua' 'terra' 'viirs'
          Data variables:
              temp               (depth, profile_id) float64 16.57 16.38 16.24 16.16 ...
              sal                (depth, profile_id) float64 33.42 33.41 33.43 33.43 ...
              fl                 (depth, profile_id) float64 1.382 1.23 1.429 2.101 ...
              fl_min             (profile_id) float64 0.2884 0.2508 0.2347 0.1904 ...
              profile_max_depth  (profile_id) float64 60.0 80.0 80.0 90.0 110.0 150.0 ...
              fl_deep_ref        (profile_id) float64 nan nan nan nan nan nan nan nan ...
              chl_count          (profile_id, sat) float64 22.0 37.0 nan 22.0 37.0 nan ...
              chl_mean           (profile_id, sat) float32 1.9567965 0.9745653 nan ...
              chl_median         (profile_id, sat) float32 1.5632914 0.7632589 nan ...
```

```
        chl_psdstd         (profile_id, sat) float64 0.6793 0.353 nan 0.6793 ...
        chl_std            (profile_id, sat) float32 1.0249827 0.32767114 nan ...
        chl_sem            (profile_id, sat) float64 0.2185 0.05387 nan 0.2185 ...
        fl_sum             (profile_id) float64 39.29 41.99 45.07 53.51 63.31 ...
```

```
In [283]: [v for v in spray.variables if 'depth' not in spray.variables[v].dims]
```

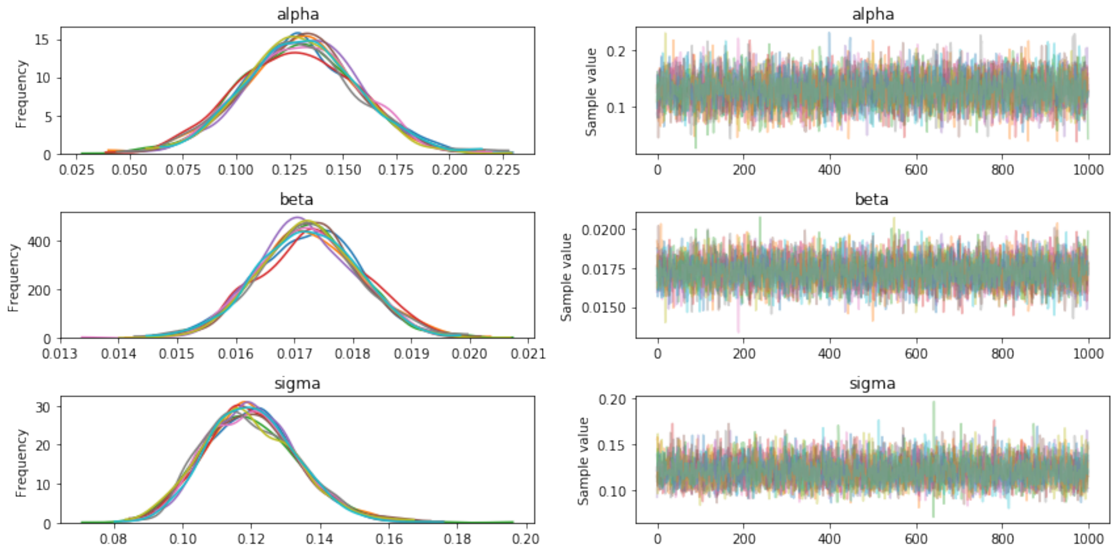
```
Out[283]: ['profile_id',
           'ndive',
           'datetime',
           'lat',
           'lon',
           'mission_id',
           'mission',
           'experiment_id',
           'experiment',
           'days',
           'fl_min',
           'profile_max_depth',
           'fl_deep_ref',
           'night_time',
           'sat',
           'chl_count',
           'chl_mean',
           'chl_median',
           'chl_psdstd',
           'chl_std',
           'chl_sem',
           'fl_sum']
```

```
In [333]:
```

```
Out[333]: <xarray.DataArray 'chl_count' ()>
          array(25.)
          Coordinates:
              sat      <U4 'aqua'
```

```
In [334]: varnames = ['fl_sum', 'chl_mean', 'chl_std', 'chl_count', 'chl_sem', 'chl_median', '
          # subset = spray.sel(sat='aqua').isel(profile_id=spray.night_time)[['fl_sum', 'chl_m
          # subset = spray.sel(sat='aqua').isel(profile_id=(spray.mission=='08401401'))[['fl_s
          subset = spray.sel(sat='aqua').isel(profile_id=(spray.night_time & (spray.mission=='

          # idx = (subset.chl_mean + subset.chl_sem) < 2
          idx = (subset.chl_count >= 8)& (subset.chl_std < 5)
          # idx = (subset.chl_count > 1)

          X = np.array(subset.fl_sum[idx])
          Y = np.array(subset.chl_median[idx])
          S = np.array(subset.chl_std.where(((subset.chl_count > 5) | (subset.chl_std >= 5.0))
          trace = fit_fl(fl=X, chl=Y, chl_std=S)

          pm.traceplot(trace)
          summary = pm.summary(trace)
          print(summary)
```

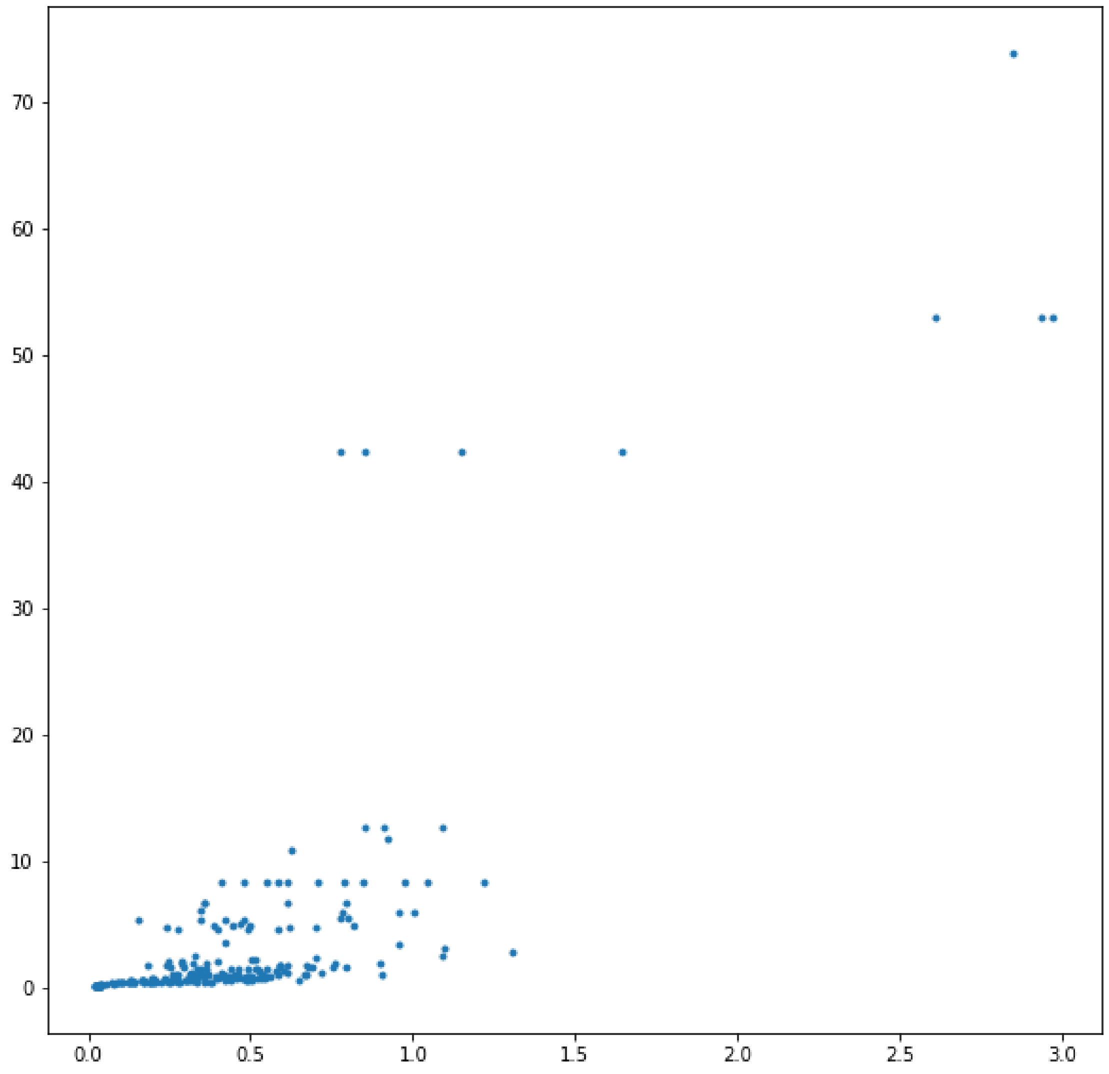
```
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (10 chains in 2 jobs)
NUTS: [sigma, beta, alpha]


           mean        sd  mc_error   hpd_2.5  hpd_97.5        n_eff      Rhat
alpha  0.129711  0.026652  0.000351  0.074383  0.178652  5757.174805  1.000497
beta   0.017237  0.000863  0.000011  0.015595  0.018952  5860.386487  1.000173
sigma  0.119999  0.013361  0.000157  0.095278  0.146434  6441.484611  0.999880
```

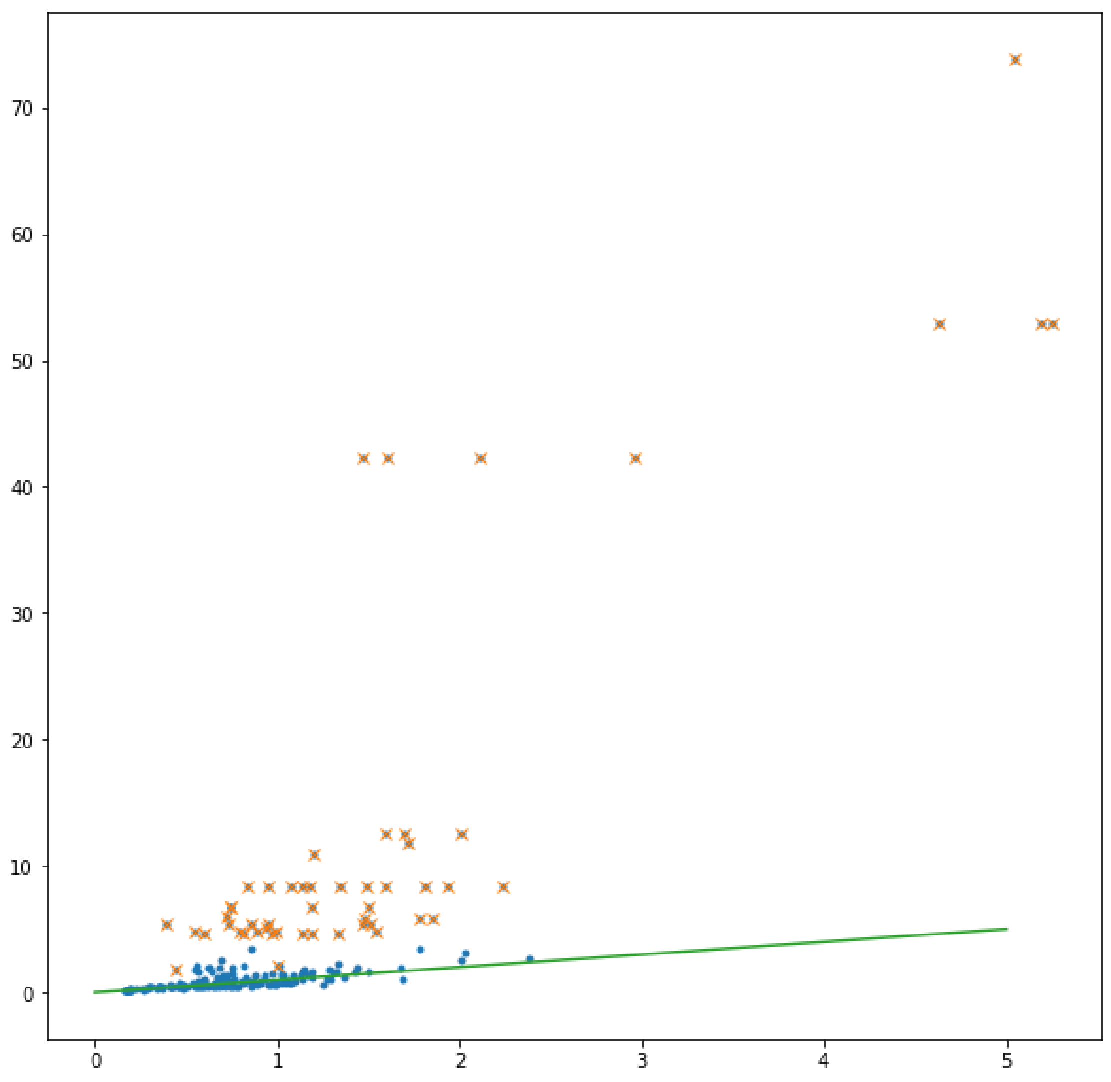


```
In [336]: fig = plt.figure(figsize=(10, 10))
          plt.plot(subset.fl_sum*.01, subset.chl_median, '.')

          fig = plt.figure(figsize=(10, 10))
          plt.plot(summary['mean']['alpha'] + summary['mean']['beta']*subset.fl_sum, subset.chl
          # plt.plot(X*summary['mean']['beta'], np.array(subset.chl_std), '.')


          idx = subset['chl_std'] > 5
```

```
plt.plot(summary['mean']['alpha'] + summary['mean']['beta']*subset.fl_sum[idx], subse
#plt.plot(X[idx]*summary['mean']['beta'], Y[idx], 'rx')
# plt.plot(X[idx]*summary['mean']['beta'], np.array(subset.chl_std)[idx], 'rx')

plt.plot([0,5], [0,5])


#plt.xlim(0, 5)
#plt.ylim(0, 30)
#plt.plot(x, y)
#plt.title("Regression model")
#plt.xlabel("Days since start of the mission")
```

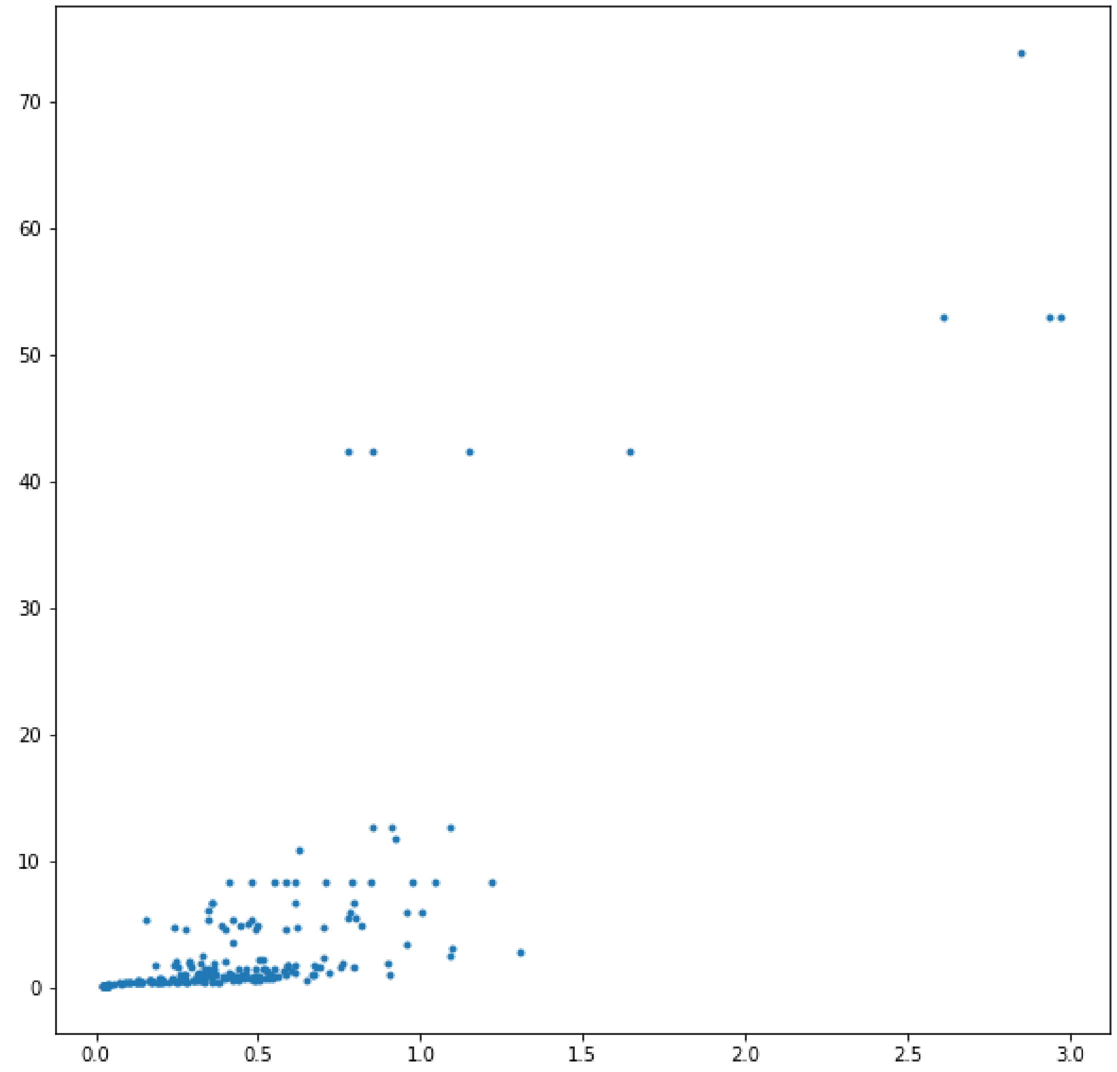
```
Out[336]: [<matplotlib.lines.Line2D at 0x1b9d9b0f0>]
```

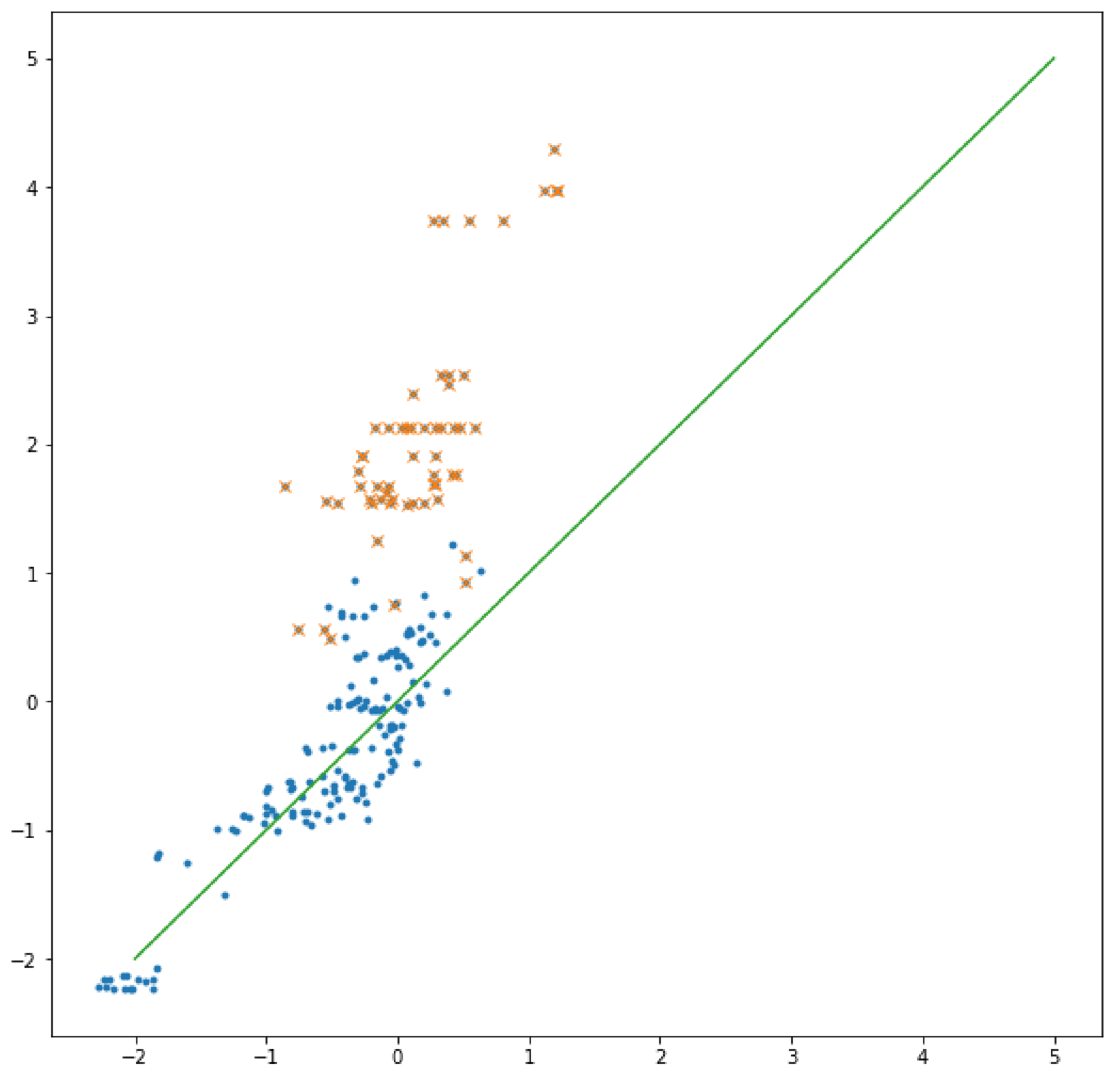




```
In [337]:

Out[337]: 0.01723661095795221

In [325]: fig = plt.figure(figsize=(10, 10))
          plt.plot(subset.fl_sum*.01, subset.chl_median, '.')

          fig = plt.figure(figsize=(10, 10))
          plt.plot(summary['mean']['alpha'] + summary['mean']['beta']*np.log(subset.fl_sum), np
          # plt.plot(X*summary['mean']['beta'], np.array(subset.chl_std), '.')


          idx = subset['chl_std'] > 2
          plt.plot(summary['mean']['alpha'] + summary['mean']['beta']*np.log(subset.fl_sum[idx]
          #plt.plot(X[idx]*summary['mean']['beta'], Y[idx], 'rx')
          # plt.plot(X[idx]*summary['mean']['beta'], np.array(subset.chl_std)[idx], 'rx')

          plt.plot([-2,5], [-2,5])


          #plt.xlim(0, 5)
          #plt.ylim(0, 30)
          #plt.plot(x, y)
          #plt.title("Regression model")
          #plt.xlabel("Days since start of the mission")
```

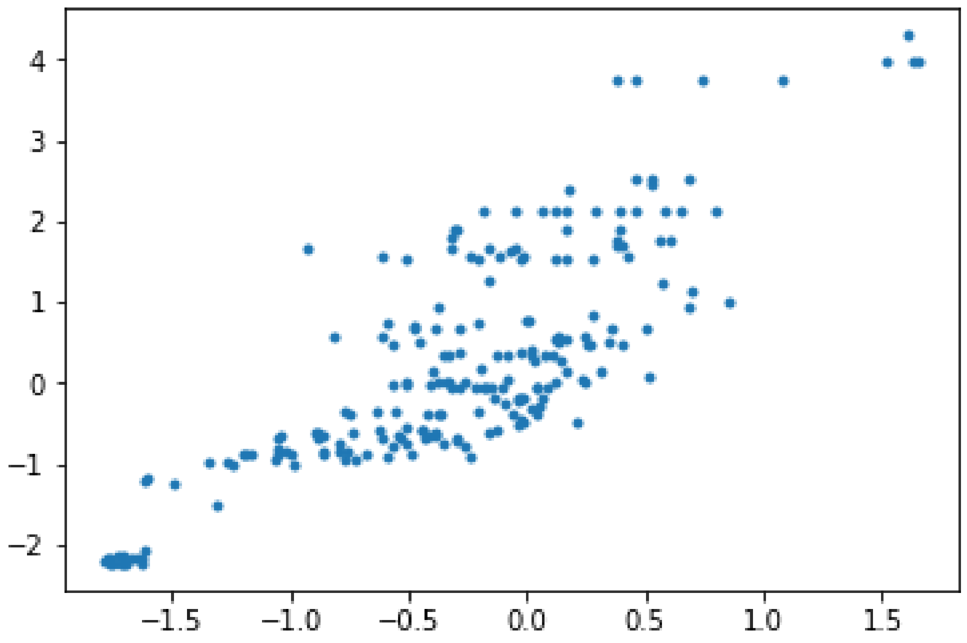
```
Out[325]: [<matplotlib.lines.Line2D at 0x1aefaed30>]
```





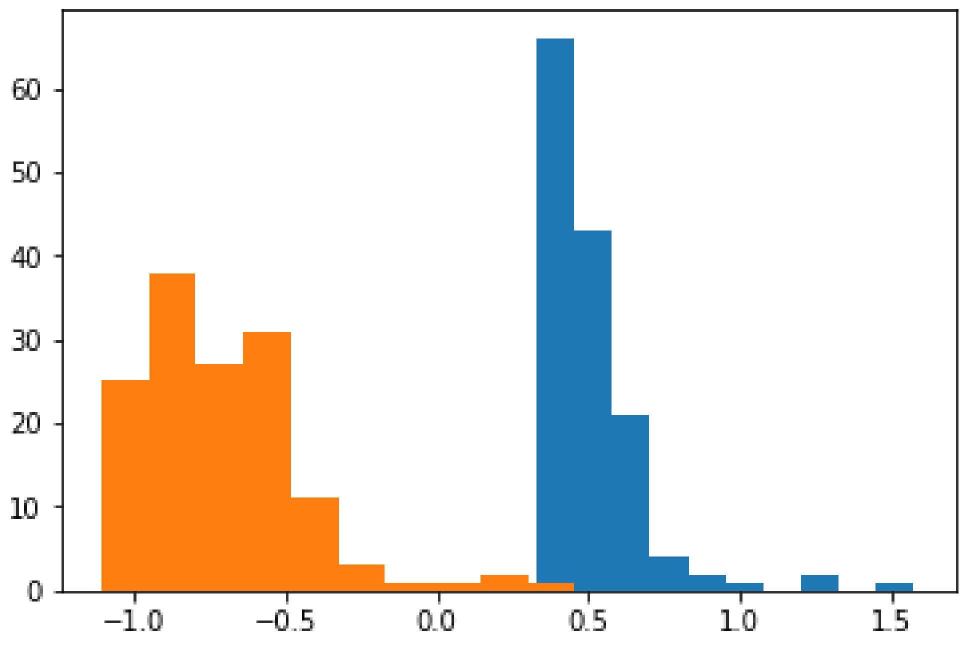
```
In [298]: plt.plot(np.log(summary['mean']['alpha'] + summary['mean']['beta']*subset.fl_sum), np
```

```
Out[298]: [<matplotlib.lines.Line2D at 0x144f4c4a8>]
```



```
In [305]: subset.isel(profile_id=(subset.chl_mean > 3) & (subset.chl_mean < 3.5))
          print(subset.profile_id)
          print(subset.sel(profile_id=28094))

          inrange = pd.read_hdf(inputFilename, key='aqua')
          inrange = inrange[inrange.profile_id==28094]

          plt.hist(inrange.chlor_a)
          plt.hist(np.log(inrange.chlor_a))

          print(inrange.chlor_a.mean())
          print(inrange.chlor_a.std())
```

```
<xarray.DataArray 'profile_id' (profile_id: 210)>
array([28094, 28095, 28096, ..., 28959, 28960, 28961])
Coordinates:
    mission        (profile_id) object '07401401' '07401401' '07401401' ...
    experiment_id  (profile_id) int64 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 ...
    days           (profile_id) float64 1.92 2.028 2.136 2.779 2.888 2.999 ...
    mission_id     (profile_id) int64 65 65 65 65 65 65 65 65 65 65 65 65 65 ...
    ndive          (profile_id) int64 22 23 24 30 31 32 33 34 39 40 41 42 43 ...
  * profile_id     (profile_id) int64 28094 28095 28096 28102 28103 28104 ...
    lat            (profile_id) float64 36.7 36.7 36.69 36.63 36.61 36.6 ...
    sat            <U4 'aqua'
    datetime       (profile_id) datetime64[ns] 2007-04-21T14:30:48 ...
    experiment     (profile_id) object 'CUGN_line_66' 'CUGN_line_66' ...
    lon             (profile_id) float64 -122.4 -122.4 -122.4 -122.6 -122.6 ...
    night_time      (profile_id) bool True True True True True True True True ...
<xarray.Dataset>
Dimensions:         ()
Coordinates:
    mission         <U8 '07401401'
    experiment_id   int64 1
    days            float64 1.92
    mission_id      int64 65
    ndive           int64 22
    profile_id      int64 28094
    lat             float64 36.7
    sat             <U4 'aqua'
    datetime        datetime64[ns] 2007-04-21T14:30:48
    experiment      <U12 'CUGN_line_66'
    lon             float64 -122.4
    night_time      bool True
Data variables:
    fl_sum          float64 26.5
    chl_mean        float32 0.5286218
    chl_std         float32 0.08720115
    chl_count       float64 24.0
    chl_sem         float64 0.0178
    chl_median      float32 0.49782938
    chl_psdstd      float64 0.1097
0.5099719
0.17724158
```



```
In [ ]: subset = spray.sel(sat='aqua')
        plt.hist(subset.chl_std.dropna(dim='profile_id'), bins=50)
        print(subset.chl_std.dropna(dim='profile_id').median())
        idx = subset.chl_count<2
        subset = subset.isel(profile_id=idx)
        subset

In [ ]: plt.hist(subset['chl_std'])

In [ ]: idx = subset['chl_std'] > 1

In [ ]: fig, ax = plt.subplots(nrows=1, ncols=1, figsize=(15, 8))

        ax.xaxis.grid(True, alpha=0.3)
        depth = spray.isel(depth=spray.depth <= 200).depth.to_series().tolist()
        data = [spray.sel(depth=d).fl.to_series().dropna().tolist() for d in depth]
        # plot violin plot
        ax.violinplot(data, depth,
                      widths=10,
                      points=200,
                      vert=False,
                      showmeans=False,
                      showextrema=True,
                      showmedians=True)

        ax.set_title('Fluorescence distribution per depth')
        ax.set_ylabel('Depth')
        ax.invert_yaxis()

In [ ]: chl_levels = np.arange(0, 10, 2.5)

In [ ]: depth = spray.isel(depth=spray.depth <= 120).depth.to_series().tolist()
        data = [subset.sel(depth=d).fl.to_series().dropna().tolist() for d in depth]

In [ ]: depth = spray.isel(depth=spray.depth <= 120).depth.to_series().tolist()

        subset = spray.sel(sat='aqua').dropna(dim='profile_id', subset=['chl_mean'])
        data = [subset.sel(depth=d).fl.to_series().dropna().tolist() for d in depth]

        print(spray.dims)
        print(subset)

        plt.hist(subset.chl_mean.isel(profile_id=subset.chl_mean<10), bins=50)

In [ ]: onesat = spray.sel(sat='aqua').dropna(dim='profile_id', subset=['chl_mean'])
        onesat.isel(profile_id=onesat.chl_mean<1)
```

```
In [54]: onesat = spray.isel(profile_id=spray.night_time==True).sel(sat='aqua').dropna(dim='pro
         onesat.isel(profile_id=onesat.chl_std<1)
         depth = onesat.isel(depth=onesat.depth <= 50).depth.to_series().tolist()

         N = 10
         C = int(N/2)
         max_chl = 3.8
         fig, ax = plt.subplots(nrows=C, ncols=2, figsize=(15, 14))
         bounds = np.linspace(0, max_chl, N+1)
         for i in range(N):
             c = int(i/C)
             r = int(i%C)
             ax[r,c].xaxis.grid(True, alpha=0.3)
             subset = onesat.isel(profile_id=(onesat.chl_mean>bounds[i]) & (onesat.chl_mean<bou

             data = [subset.isel(depth=[0,1]).fl_unbias.sum(dim='depth').to_series().dropna().t
             data += [subset.sel(depth=d).fl_unbias.to_series().dropna().tolist() for d in dept
             # plot violin plot
             ax[r,c].violinplot(data, [-10] + depth,
                       widths=10,
                       points=200,
                       vert=False,
                       showmeans=False,
                       showextrema=True,
                       showmedians=True)

             ax[r,c].vlines((bounds[i]+bounds[i+1])/2 - subset.chl_sem.mean(), 50, 0)
             ax[r,c].vlines((bounds[i]+bounds[i+1])/2 + subset.chl_sem.mean(), 50, 0)


             ax[r,c].set_title('Fluorescence distribution per depth')
             ax[r,c].set_ylabel('Depth')
             ax[r,c].invert_yaxis()
             ax[r,c].set_xlim(-0.1, max_chl + 0.1)


         plt.savefig('fl_dist_per_satchl.png')


        ---------------------------------------------------------------------------

        ValueError                                Traceback (most recent call last)

        <ipython-input-54-453bd021936e> in <module>()
    ----> 1 onesat = spray.isel(profile_id=spray.night_time==True).sel(sat='aqua').dropna(dim=
          2 onesat.isel(profile_id=onesat.chl_std<1)
          3 depth = onesat.isel(depth=onesat.depth <= 50).depth.to_series().tolist()
          4
```

```
      5 N = 10

    ~/.virtualenvs/fluorescence/lib/python3.6/site-packages/xarray/core/dataset.py in sel(s
   1464         """
   1465         pos_indexers, new_indexes = remap_label_indexers(self, method,
-> 1466                                                          tolerance, **indexers)
   1467         result = self.isel(drop=drop, **pos_indexers)
   1468         return result._replace_indexes(new_indexes)

    ~/.virtualenvs/fluorescence/lib/python3.6/site-packages/xarray/core/coordinates.py in r
    344
    345     pos_indexers, new_indexes = indexing.remap_label_indexers(
--> 346         obj, v_indexers, method=method, tolerance=tolerance
    347     )
    348     # attach indexer's coordinate to pos_indexers

    ~/.virtualenvs/fluorescence/lib/python3.6/site-packages/xarray/core/indexing.py in rema
    221     new_indexes = {}
    222
--> 223     dim_indexers = get_dim_indexers(data_obj, indexers)
    224     for dim, label in iteritems(dim_indexers):
    225         try:

    ~/.virtualenvs/fluorescence/lib/python3.6/site-packages/xarray/core/indexing.py in get_
    189     if invalid:
    190         raise ValueError("dimensions or multi-index levels %r do not exist"
--> 191                          % invalid)
    192
    193     level_indexers = defaultdict(dict)

    ValueError: dimensions or multi-index levels ['sat'] do not exist
```

```
In [ ]: depth
```

```
In [ ]: subset.chl_sem.mean()
```

```
In [ ]: plt.hist(onesat.chl_std.dropna(dim='profile_id'), bins=50)
```