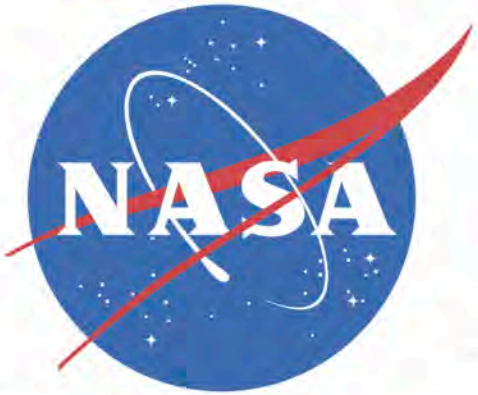
Leveraging Python & Jupyter Notebook to validate satellite Ocean Color retrievals

Joel P Scott

NASA GSFC
OB.DAAC / SeaBASS

International Ocean Colour Science Meeting

2019-04-09, Breakout Workshop



My Coding Background

- Formally trained in: FORTRAN and C
- During graduate school: Maple, MATLAB, Mathematica, GrADS, & IDL
- Early career: FORTRAN (processing/speed) and MATLAB (visualization & analysis)
- But...
  - MATLAB is costly after school
  - Toolboxes not always portable (collaboration, reproducibility, etc)

Summer 2014

- Two colleagues and I: all new code would be written in Python
- Still had MATLAB, if needed
  - But rarely did
- By the end of the summer, Python was my language of choice.

Why Python?

- Open-source license
- Multi-platform interoperability → Windows, Mac, & Linux
- Established language
    - Existed since the late 80s
    - Popular among web, app developers, & data science communities
    - Well-documented online, with a major StackOverflow presence
- Versatile modules for tons of application
    - Machine learning
    - Data processing, analysis, & visualization
    - Database queries
    - HTTPS requests
    - JSON parsing, etc

My Setup

- Package manager: Anaconda
- Frequently used modules
  - NumPy (vector arithmetic functions)
  - Xarray and/or pandas (array & dataset manipulation, netCDF support)
  - Matplotlib (MATLAB-esque plots & more)

Python can be executed on the command line, via a script, or in an IDE (Integrated Development Environment).

- Jupyter Notebook
  - Runs in a browser
  - Cells are individually executable
  - Supports Markdown
  - Help prompts (tab-suggestions & shift-tab doc-string)
  - Supports in-line plots
  - Exportable as .py, .pdf, .tex, .html, etc
- Runs Reveal.js via RISE

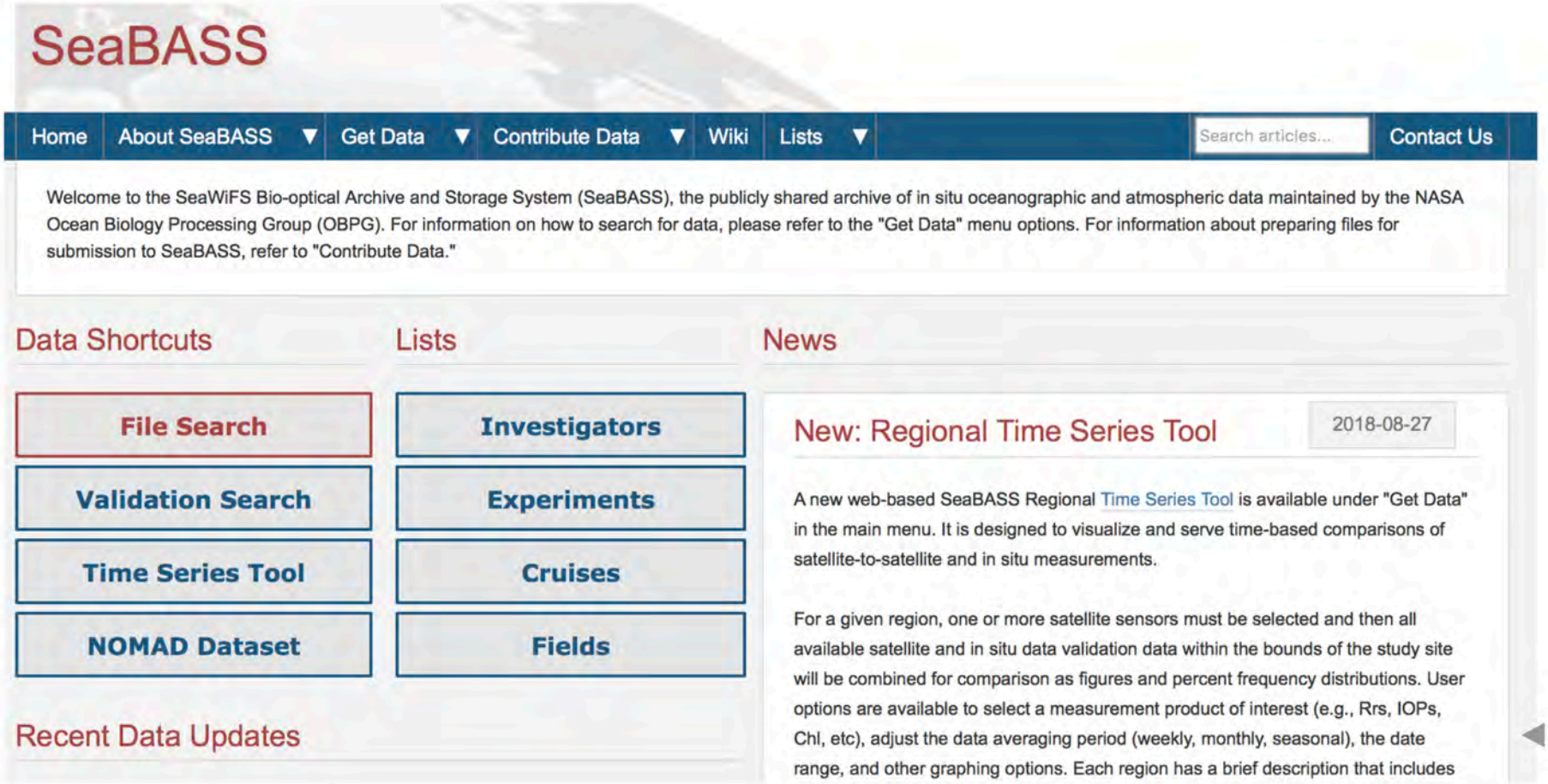
Why I'm a fan of Python

Example:

1. Start with an "in situ" data file, in the SeaBASS format, Gulf of Maine, Scotia-Prince Ferry (Balch, 2002)
2. Find coincident Ocean Color satellite granules
3. Extract satellite data and pair it to the in situ data (matchups!)
4. Derive "goodness-of-fit" metrics
5. Plot the matchup data (2-ways)

What is SeaBASS?

SeaBASS is a bio-optical data archive at NASA's OB.DAAC (Ocean Biology Distributed Active Archive Center).



What is a SeaBASS file?

- Self-describing ASCII data file
- Metadata headers
- Delimited (comma, space, or tab) data matrix

```
/begin_header
/identifier_product_doi=10.5067/SeaBASS/SCOTIA_PRINCE_FERRY/DATA001
/received=20021209
/affiliations=Bigelow_Laboratory_for_Ocean_Sciences
/investigators=William_Balch
/contact=bbalch@bigelow.org
/experiment=Scotia_Prince_ferry
/cruise=s020710w
/data_type=flow_thru
/west_longitude=-69.7643[deg]
/east_longitude=-66.2084[deg]
/north_latitude=43.7783[deg]
/south_latitude=43.6262[deg]
/start_date=20020710
/end_date=20020710
/start_time=12:49:48[gmt]
/end_time=21:29:14[gmt]
/delimiter=space
/fields=year,month,day,hour,minute,second,lat,lon,Wt,sal,bb470,bb514,bb676,CHL,ag
/units=yyyy,mo,dd,hh,mn,ss,degrees,degrees,degreesc,psu,1/m,1/m,1/m,mg/m^3,1/m,1/
! original_file_name=s020710w.txt
/data_file_name=main-s020710w.txt
/missing=-99
/calibration_files=ac90194.020426,hydroscat.020423,sas3332f.020218
/documents=readme.txt
/data_status=preliminary
/measurement_depth=3
/water_depth=NA
/wind_speed=NA
/wave_height=NA
/station=NA
/secchi_depth=NA
/cloud_percent=NA
/end_header@
2002 7 10 12 49 48 43.7783 -66.2084 12.18 31.79 9.72E-03 -9.90E+01 6.71E-03 2.57
2002 7 10 12 51 53 43.7762 -66.2445 11.97 31.73 9.56E-03 9.10E-03 6.68E-03 1.69E
2002 7 10 12 53 59 43.7746 -66.2674 11.82 31.81 9.25E-03 9.30E-03 6.53E-03 1.49E
2002 7 10 12 56 10 43.7736 -66.2822 11.7 31.84 8.94E-03 7.23E-03 6.40E-03 1.46E+0
2002 7 10 12 58 25 43.7726 -66.2971 11.59 31.87 8.68E-03 6.39E-03 6.32E-03 1.52E
2002 7 10 13 0 41 43.7714 -66.3127 11.6 31.82 8.42E-03 6.91E-03 6.19E-03 1.57E+00
2002 7 10 13 2 53 43.7703 -66.3284 11.67 31.81 8.30E-03 6.11E-03 6.17E-03 1.79E+0
2002 7 10 13 5 4 43.7695 -66.3437 11.72 31.78 8.16E-03 6.15E-03 6.26E-03 1.76E+00
2002 7 10 13 7 15 43.7694 -66.3593 11.65 31.82 7.96E-03 6.13E-03 6.12E-03 1.51E+0
```

Making a matchup

- Leverage SeaBASS's stand-alone satellite validation matchup scripts
  - Written in Python 3
  - Avaliable via NASA SeaDAS software

Satellite Validation Match-up Tools

The following tools provide a work flow to create satellite validation match-ups to data contained in SeaBASS formatted-files.

Script Name	Description	Language	Location
fd_matchup.py	find relevant OB.DAAC satellite granules from in situ points or a SeaBASS data file	Python	part of SeaDAS software package
mk_matchup.py	generate coincident satellite validation match-ups for a SeaBASS data file	Python	part of SeaDAS software package

1. Find matching satellite granules: fd_matchup.py

- Inputs
  - SeaBASS file: Gulf of Maine Scotia-Prince Ferry data (Balch, 2002)
  - Satellite instrument: MODIS-Aqua
  - Data type: Ocean Color satellite product suite → *Chl* retrievals via OCI algorithm

- Actions
  - Locates coincident Level-2 satellite granules
  - via NASA's Earthdata Search's CMR API (nominally within +/- 3 hrs)

- Outputs
  - List of granules & their URLs
  - Download matching netCDF granules

In [2]:
```
cmd_find = 'fd_matchup.py --sat modisa --data_type oc --seabass_file ' + \
           str(file_sb) + ' --get_data ' + str(dir_tmp)

%time pid = subprocess.run(cmd_find, stdout=subprocess.PIPE, stderr=subprocess.PIPE, \
                    encoding='ascii', shell=True)

print('\nSTDOUT:\n',pid.stdout)
```

```
CPU times: user 12.8 ms, sys: 15.2 ms, total: 28 ms
Wall time: 6min 39s

STDOUT:
 Matching AQUA/MODIS granule (A2002191170500.L2_LAC_OC.nc) found for: data/main-s020710w.txt
Downloading A2002191170500.L2_LAC_OC.nc to: tmp
SUCCESS!

Matching AQUA/MODIS granule (A2002191184000.L2_LAC_OC.nc) found for: data/main-s020710w.txt
Downloading A2002191184000.L2_LAC_OC.nc to: tmp
SUCCESS!

Number of granules found: 2
```

2. Pair satellite data to in situ targets: mk_matchup.py

- Inputs
  - SeaBASS file
  - One (or more) L2 satellite granules
- Actions
  - Extract & pair satellite data to in situ data
  - Applies exclusion criteria from Bailey & Werdell (2006)
    - 5x5 pixel extract
    - Greater than 50% valid pixels
    - Solar zenith angle less than 70° & satellite zenith angle less than 60°
    - Max time difference +/- 3 hours
    - Max coefficient of variation less than 0.15
- Output: appends satellite matchups to SeaBASS file as additional columns

In [4]:
```
lis_datasat = sorted(list(dir_tmp.glob('*.nc')))

cmd_make = 'mk_matchup.py' + \
           ' --seabass_file ' + str(file_sb) + \
           ' --sat_file ' + ' '.join([str(fname_L2) for fname_L2 in lis_datasat])

%time pid = subprocess.run(cmd_make, stdout=subprocess.PIPE, stderr=subprocess.PIPE, \
                    encoding='ascii', shell=True)

print('\nSTDOUT:\n',pid.stdout)
```

```
CPU times: user 7.36 ms, sys: 9.53 ms, total: 16.9 ms
Wall time: 4min

STDOUT:
 Looking for satellite/in situ match-ups for: data/main-s020710w.txt
Checking: tmp/A2002191170500.L2_LAC_OC.nc
Satellite/in situ match-up(s) found.
Checking: tmp/A2002191184000.L2_LAC_OC.nc
No valid satellite match-ups found.
```

3. Load and plot matchup data from SeaBASS file

- Via the SeaBASS Python module (SB_support.py)
  - readSB class

```
In [7]: from SB_support import readSB

        ds = readSB(filename=file_sb, mask_missing=True, no_warn=True)
```

- Loads entire SeaBASS file using native Python structures
  - metadata headers
  - auxillary comments
  - fields/units
  - data matrix

In [8]:
```
for key in ds.variables:
    print(ds.variables[key])
```

```
('modis_aqua_angstrom', 'none')
('modis_aqua_angstrom_sd', 'none')
('modis_aqua_rrs_412', 'sr^-1')
('modis_aqua_rrs_412_sd', 'sr^-1')
('modis_aqua_rrs_443', 'sr^-1')
('modis_aqua_rrs_443_sd', 'sr^-1')
('modis_aqua_rrs_469', 'sr^-1')
('modis_aqua_rrs_469_sd', 'sr^-1')
('modis_aqua_rrs_488', 'sr^-1')
('modis_aqua_rrs_488_sd', 'sr^-1')
('modis_aqua_rrs_531', 'sr^-1')
('modis_aqua_rrs_531_sd', 'sr^-1')
('modis_aqua_rrs_547', 'sr^-1')
('modis_aqua_rrs_547_sd', 'sr^-1')
('modis_aqua_rrs_555', 'sr^-1')
('modis_aqua_rrs_555_sd', 'sr^-1')
('modis_aqua_rrs_645', 'sr^-1')
('modis_aqua_rrs_645_sd', 'sr^-1')
('modis_aqua_rrs_667', 'sr^-1')
('modis_aqua_rrs_667_sd', 'sr^-1')
('modis_aqua_rrs_678', 'sr^-1')
('modis_aqua_rrs_678_sd', 'sr^-1')
```

In [9]:
```
print('\n'.join(ds.comments[:35]))
```

```
original_file_name=s020710w.txt

bb514 values have been corrected for instrument biofouling.
ac9 values have been corrected for temperature, salinity and biofouling of the instrument.


Pre cruise bb514 for HPLC water was 0.000682
Post cruise bb514 for HPLC water was 0.0008068

ac9 water calibrations
Channel     Precruise        Postcruise
a412        -0.000815        -0.000815
a440        -0.013657        -0.013657
a488        -0.030516        -0.030516
a510        -0.031062        -0.031062
a555        -0.034711        -0.034711
a630        -0.037208        -0.037208
a650        -0.034883        -0.034883
a676        -0.048446        -0.048446
a715        -0.020361        -0.020361
c412        -0.001934        -0.001934
c440        -0.005771        -0.005771
```

In [10]:
```
print(ds.length)
print(ds.data['sal'])
```

```
266
[31.79, 31.73, 31.81, 31.84, 31.87, 31.82, 31.81, 31.78, 31.82, 31.83, 31.81, 31.85, 31.99, 31.96, 32.09, 32.1
7, 31.95, 31.92, 32.2, 32.3, 32.23, 32.28, 32.44, 32.45, 32.47, 32.5, 32.54, 32.5, 32.52, 32.52, 32.45, 32.39,
32.4, 32.28, 32.25, 32.24, 32.24, 32.25, 32.2, 32.21, 32.24, 32.18, 32.17, 32.15, 32.14, 32.14, 32.16, 32.17,
32.16, 32.16, 32.15, 32.14, 32.14, 32.12, 32.12, 32.13, 32.12, 32.11, 32.1, 32.1, 32.11, 32.11, 32.1, 32.1, 32
.1, 32.08, 32.06, 32.05, 32.04, 32.03, 32.03, 32.03, 32.03, 32.03, 32.03, 32.03, 32.04, 32.03, 32.03, 32.02, 3
2.02, 32.02, 32.02, 32.02, 32.02, 32.01, 32.01, 32.01, 32.02, 32.02, 32.02, 32.03, 32.04, 32.05, 32.04, 32.04,
32.04, 32.04, 32.04, 32.05, 32.04, 32.03, 32.03, 32.03, 32.02, 32.02, 32.02, 32.03, 32.04, 32.03, 32.04, 32.04
, 32.04, 32.05, 32.05, 32.04, 32.04, 32.04, 32.04, 32.04, 32.03, 32.03, 32.02, 32.03, 32.04, 32.04, 32.04, 32.
04, 32.04, 32.04, 32.05, 32.06, 32.07, 32.07, 32.06, 32.07, 32.06, 32.06, 32.05, 32.05, 32.05, 32.07, 32.07, 3
2.06, 32.09, 32.11, 32.12, 32.11, 32.09, 32.1, 32.09, 32.08, 32.06, 32.04, 32, 31.98, 31.96, 31.96, 31.99, 32.
02, 32.07, 32.08, 32.09, 32.08, 32.02, 31.97, 31.96, 31.94, 31.92, 31.91, 31.9, 31.89, 31.91, 31.92, 31.93, 31
.92, 31.91, 31.89, 31.89, 31.88, 31.87, 31.85, 31.83, 31.81, 31.8, 31.79, 31.79, 31.79, 31.79, 31.79, 31.78, 3
1.76, 31.76, 31.76, 31.76, 31.86, 31.85, 31.83, 31.83, 31.85, 31.84, 31.84, 31.83, 31.81, 31.82, 31.82, 31.81,
31.8, 31.8, 31.8, 31.79, 31.75, 31.7, 31.66, 31.63, 31.62, 31.61, 31.6, 31.57, 31.49, 31.49, 31.54, 31.52, 31.
47, 31.42, 31.46, 31.46, 31.49, 31.47, 31.38, 31.32, 31.33, 31.35, 31.54, 31.41, 31.44, 31.42, 31.38, 31.35, 3
1.35, 31.36, 31.35, 31.34, 31.35, 31.35, 31.36, 31.37, 31.43, 31.45, 31.48, 31.5, 31.51, 31.49, 31.47, 31.45,
31.46, 31.45, 31.45, 31.43, 31.43, 31.43, 31.42, 31.38, 31.35, 31.3, 31.22]
```

Matchup: where both in situ & satellite data are valid on the same row

In [11]:
```
chl_sat    = [] #list to contain satellite matchup values
chl_insitu = [] #list to contain in situ matchup values

for chl_sat_i,chl_insitu_i in zip(ds.data['modis_aqua_chlor_a'], ds.data['chl']):

    if not isnan(chl_sat_i) and not isnan(chl_insitu_i):
        chl_insitu.append(chl_insitu_i)
        chl_sat.append(chl_sat_i)

[chl_insitu, chl_sat] = avg_duplicate_matchups(chl_insitu, chl_sat) #average together in situ targets within same
```

4. Compute goodness-of-fit statistics

- Chlorophyll is log-normally distributed → Multiplicative metrics (Seegers et al, 2018)

- Mean bias

$$Mean\ bias = 10^{\left(\frac{\sum_{i=1}^{n} \log_{10}(M_i) - \log_{10}(O_i)}{n}\right)}$$

In [12]:
```
bias = 10**np.mean([a - b for a,b in zip(np.log10(chl_sat), np.log10(chl_insitu))])
```

- Mean absolute error (MAE)

$$MAE = 10^{\left(\frac{\sum_{i=1}^{n} |\log_{10}(M_i) - \log_{10}(O_i)|}{n}\right)}$$

In [13]:
```
from sklearn.metrics import mean_absolute_error

mae = 10**mean_absolute_error(np.log10(chl_insitu), np.log10(chl_sat))
```

- Linear regression slope
  - Ordinary Least Squares (OLS) Reduced Major Axis (RMA)

In [14]:
```
linreg_rma = regress2(np.asarray(chl_insitu), np.asarray(chl_sat), \
                      _method_type_1 = 'OLS', \
                      _method_type_2 = 'reduced major axis', \
                      _weight_x = [], _weight_y = [], _need_intercept = True)
```

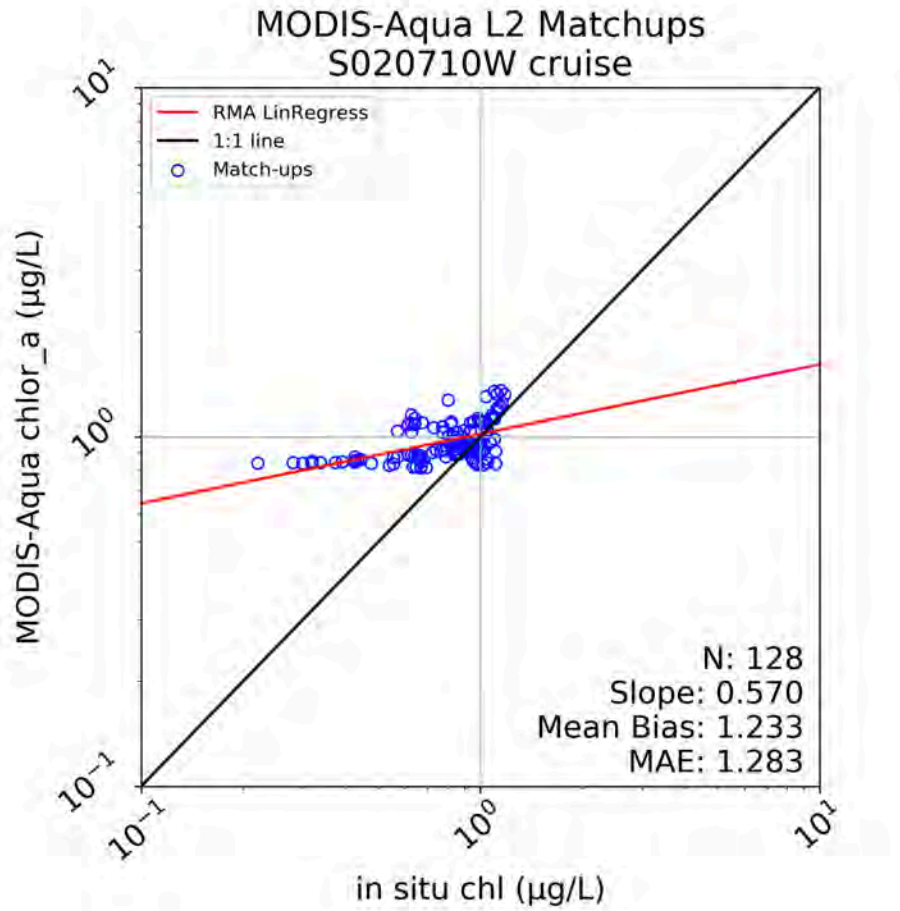
Matchup statistics:

In [15]:
```
print('Mean bias:\t\t{:.3f}'.format(bias))
print('Mean absolute error:\t{:.3f}'.format(mae))
print('RMA LinReg slope:\t{:.3f}'.format(linreg_rma['slope']))
```

```
Mean bias:              1.233
Mean absolute error:    1.283
RMA LinReg slope:       0.570
```

Plot 1: matchup data as a scatter plot

In [18]:
```
ax.scatter(chl_insitu, chl_sat, facecolor='None', edgecolor='b', label='Match ups')
ax.plot(xline, myExpFunc(xline, *popt), 'r-', label="RMA LinRegress")
ax.plot(xline, xline, color='black', label='1:1 line');
```


MODIS-Aqua L2 Matchups
S020710W cruise
RMA LinRegress
1:1 line
Match-ups
N: 128
Slope: 0.570
Mean Bias: 1.233
MAE: 1.283
MODIS-Aqua chlor_a (µg/L)
in situ chl (µg/L)
10^{1}
10^{0}
10^{-1}

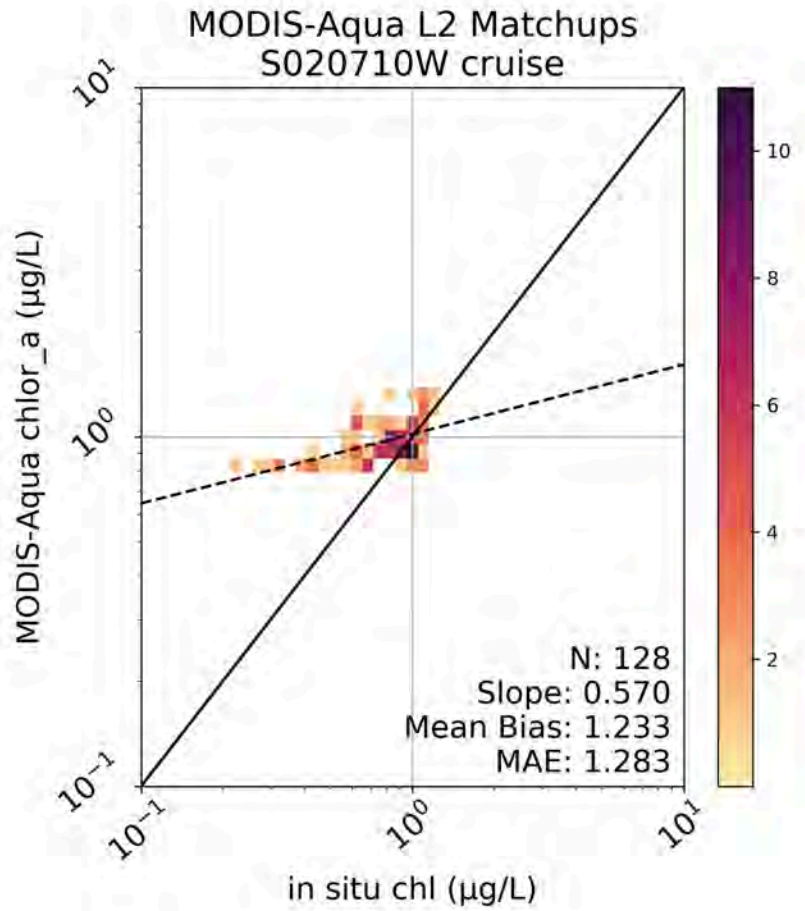
Plot 2: matchup data as a 2-dim histogram (joint-PDF)

- Create and fill a 50x50 grid-space with matchup density

In [23]:
```
xbins = np.logspace(np.log10(xlims[0]), np.log10(xlims[1]), num=50)
ybins = np.logspace(np.log10(ylims[0]), np.log10(ylims[1]), num=50)

h,xedges,yedges = np.histogram2d(chl_insitu, chl_sat, bins=[xbins, ybins])
X,Y = np.meshgrid(xedges, yedges)

p1 = ax.pcolormesh(X, Y, h.T, cmap=cm, vmin=1e-5)
ax.plot(xbins, myExpFunc(xbins, *popt), 'k--', label="RMA LinRegress")
ax.plot(xbins, xbins, color='black', label="1:1 line");
```


MODIS-Aqua L2 Matchups
S020710W cruise
MODIS-Aqua chlor_a (µg/L)
in situ chl (µg/L)
10^{1}
10^{0}
10^{-1}
10^{-1}
10^{0}
10^{1}
10
8
6
4
2
N: 128
Slope: 0.570
Mean Bias: 1.233
MAE: 1.283

Summary

- Why I love Python
  - Language is open source & well-supported
  - Code is shareable & reproducible
  - Scripts are efficient & versatile
- Many applications to Ocean Color
  - Find & download data
  - Create matchups
  - Read/write or analyze/plot data

Thank you!

Questions?

joel.scott@nasa.gov