The Earth Column Model Collaboratory (EMC^2): A new open source community framework for comparing column outputs from large-scale models with ground-based remote-sensing measurements

ROBERT JACKSON, ISRAEL SILBER, ANN FRIDLIND, SCOTT COLLIS

This project is funded by the Atmospheric System Research Science Focus Area

Motivation

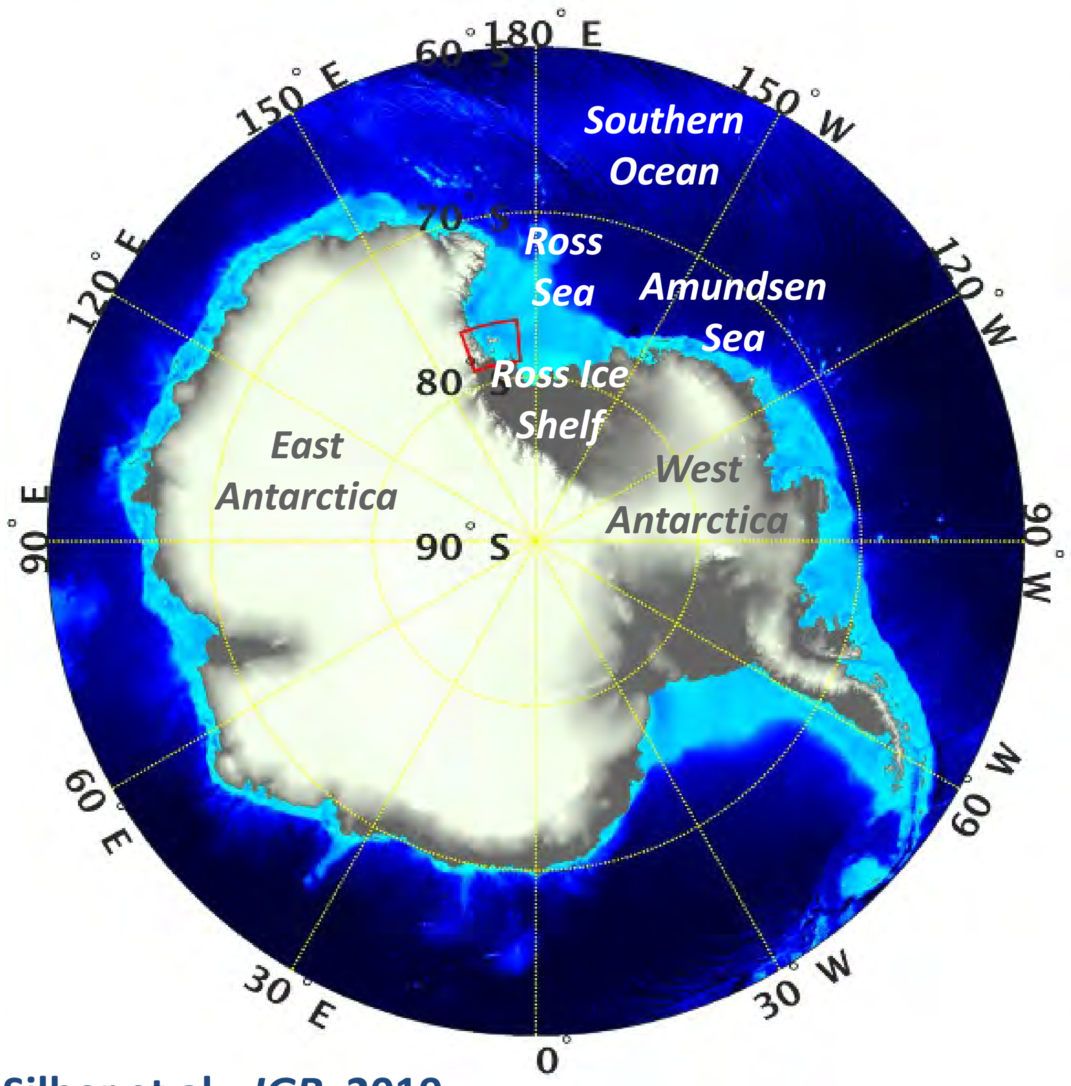
- Inadequate representation of cloud microphysical processes in large scale models
- Model evaluation requires a direct (“apples-to-apples”) comparison with observations
  - Remote sensing observations do not retrieve some microphysical quantities and suffer from detectability constraints
  - Resolution differences
- Various forward simulators exist for different purposes, e.g., COSP, CR-SIM
- Large-scale models continuously evolve – moving targets.
- **Flexible, easy to implement, fast evolving framework for ground-based instruments is needed.**

Earth Column Model Collaboratory (EMC^2) Overview

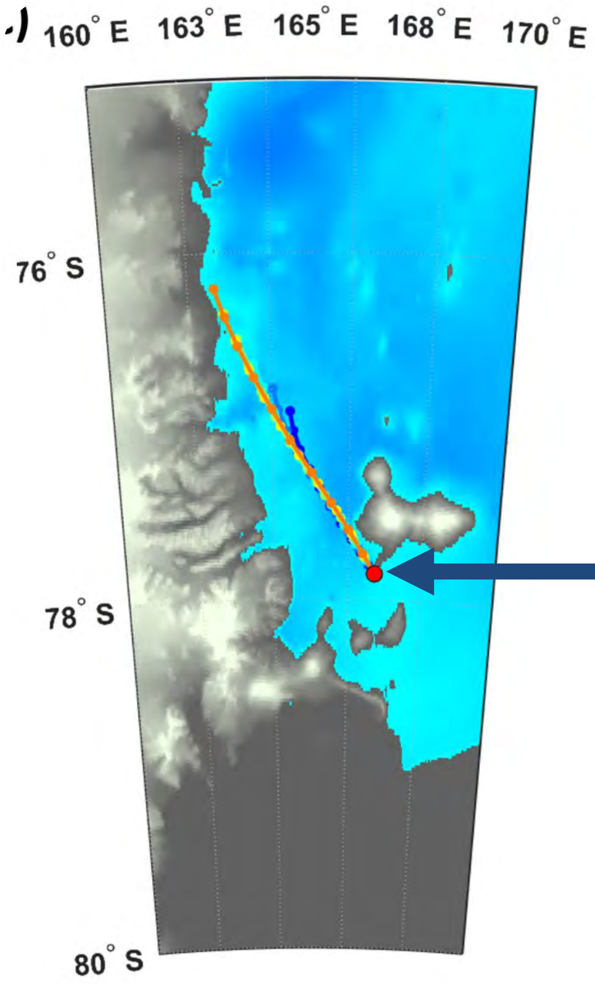
- Open Source offline framework for large scale model-observation comparison
- Fully written in python (Xarray-based data model for easy integration with netCDF datasets)
- Includes a subcolumn generator (expansion of Lamer et al., *GMD*, 2018) and a forward calculation components for ground-based instruments
- Faithful to the physics implemented in large-scale model microphysics schemes (e.g., fully compatible with the MG2 scheme)

https://columncolab.github.io/EMC2/

SCM Case study example from AWARE (August 16, 2016)



Silber et al., *JGR*, 2019



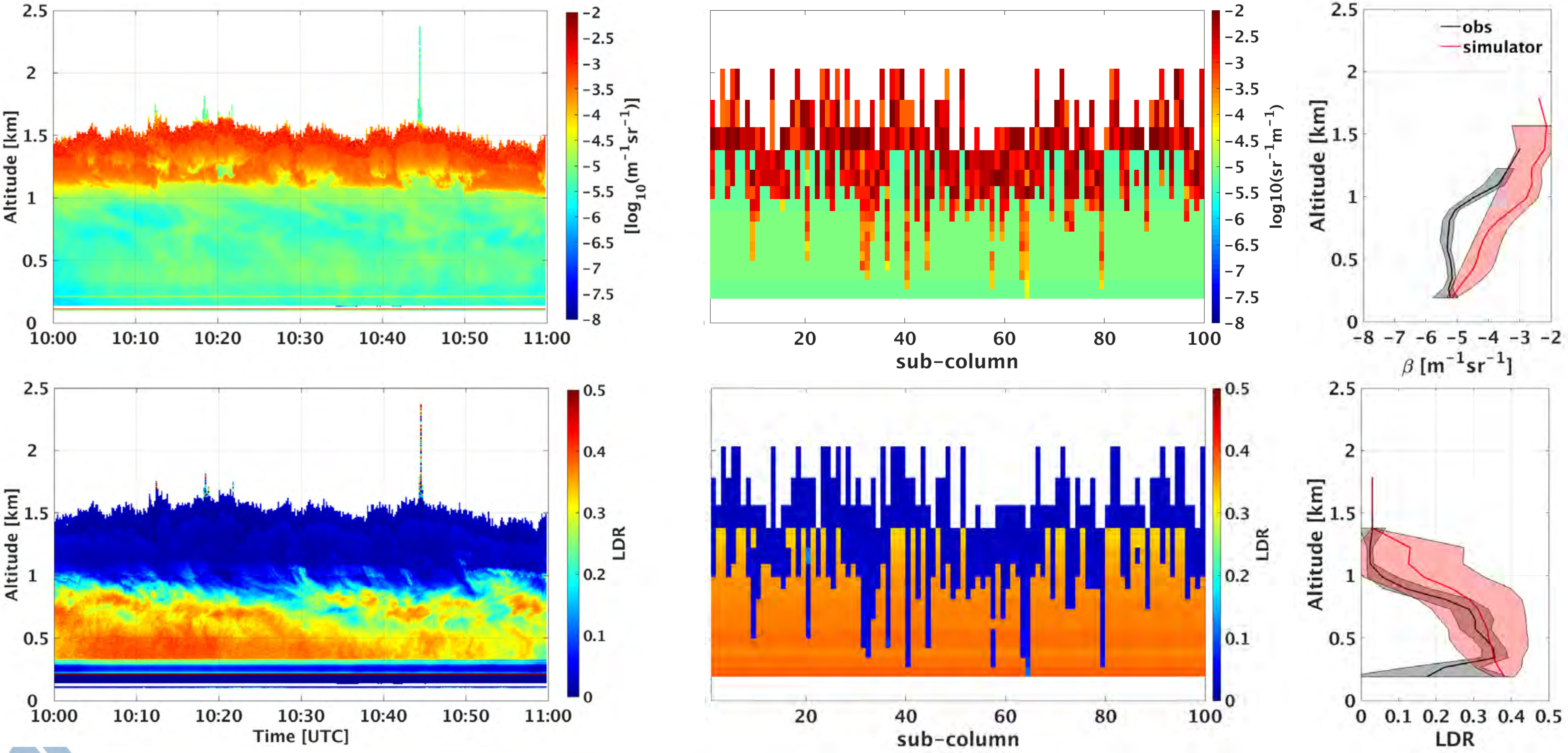
This case study describes the quasi-Lagrangian formation and evolution of a highly supercooled drizzling cloud ($T_{CB} \approx -25$ °C) in an initially stable atmosphere

Model column "approaches" McMurdo Station at 10 UTC (9 h simulation time)

Trade-off between observational temporal resolution and model spatial resolution

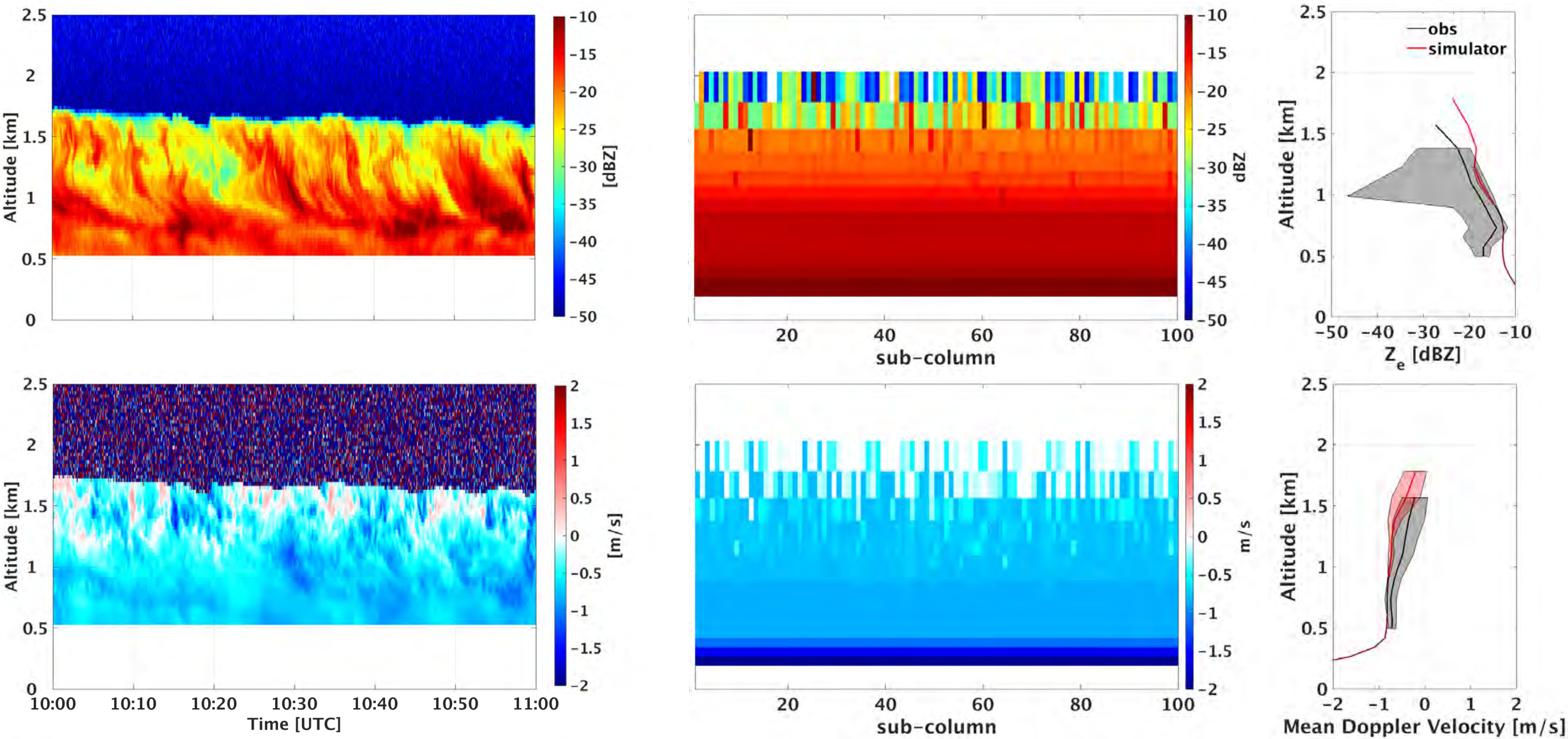
HSRL
left – obs, right – simulator (100 subcolumns)

AWARE (August 16, 2016)
ModelE3 SCM vs obs



KAZR
left - obs, right - simulator (100 subcolumns)

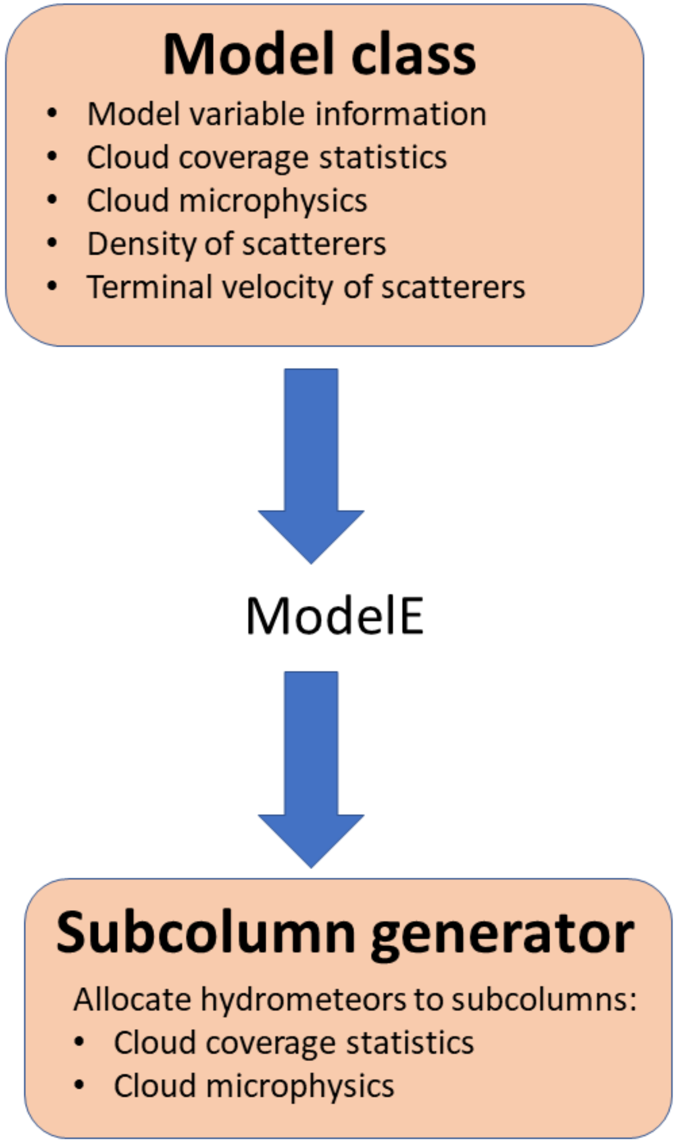
AWARE (August 16, 2016)
ModelE3 SCM vs obs

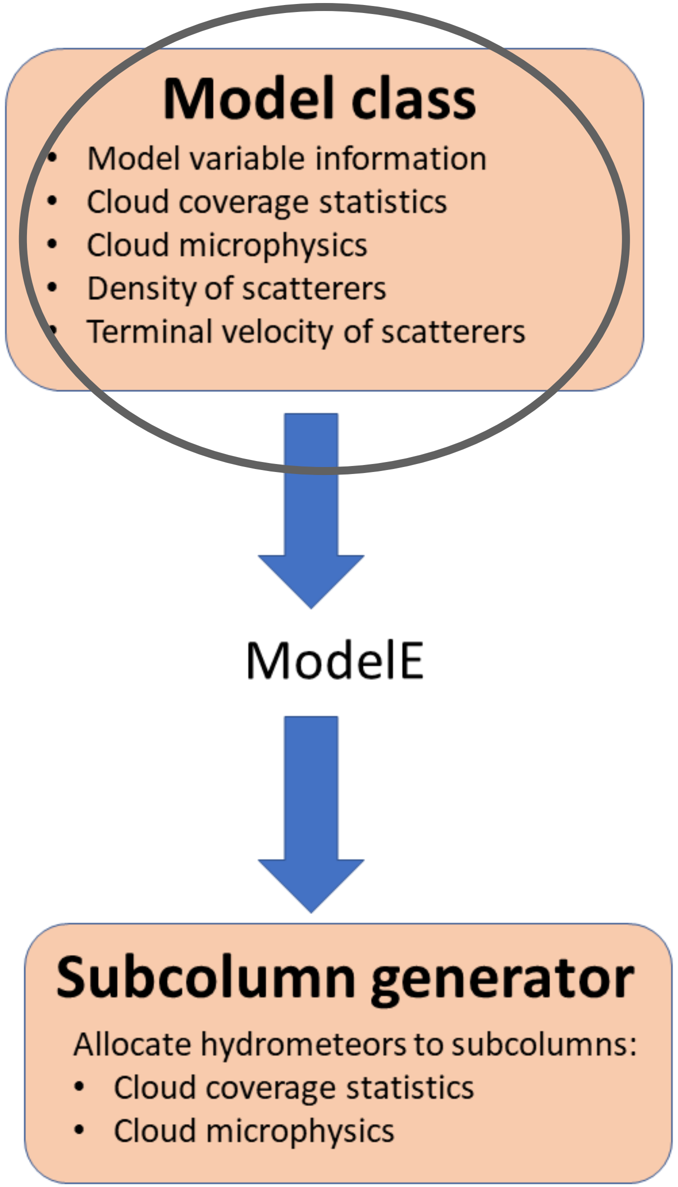


Earth Column Model Collaboratory (EMC^2) Software

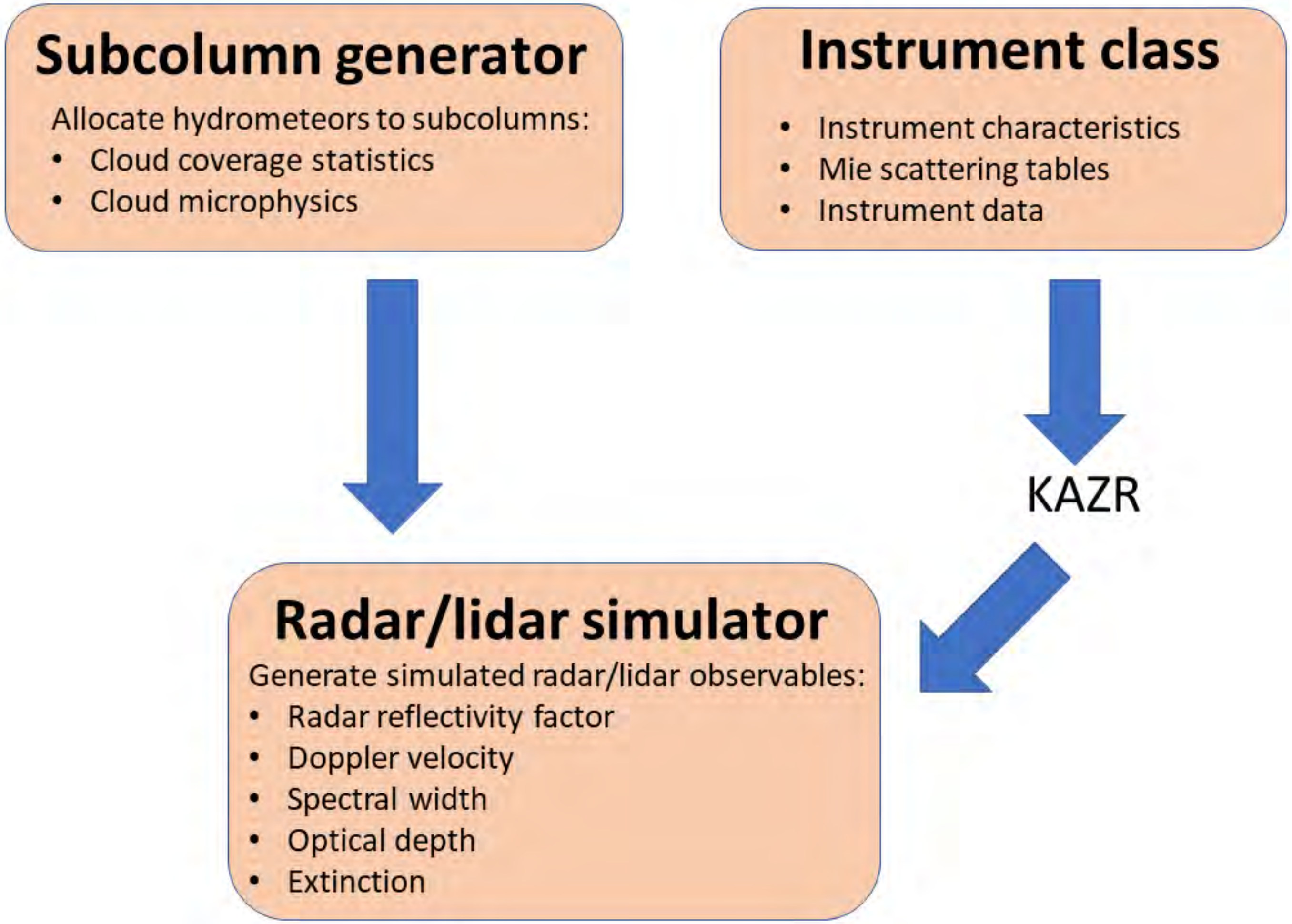
- **Easy documentation:** Automated documentation builder for easy documentation of procedures
- **Unit testing:** Ensures code integrity on a variety of platforms
- **Expandable structure:** Adaptability for a variety of models and sensors
  - Can operate on the GISS ModelE3 output and can be easily amended for E3SM, CAM6, etc.
  - Currently includes essential Mie scattering lookup tables and instrument characteristics for the ARM HSRL, Ceilometer, Raman lidar, MPL, KAZR, and WACR.

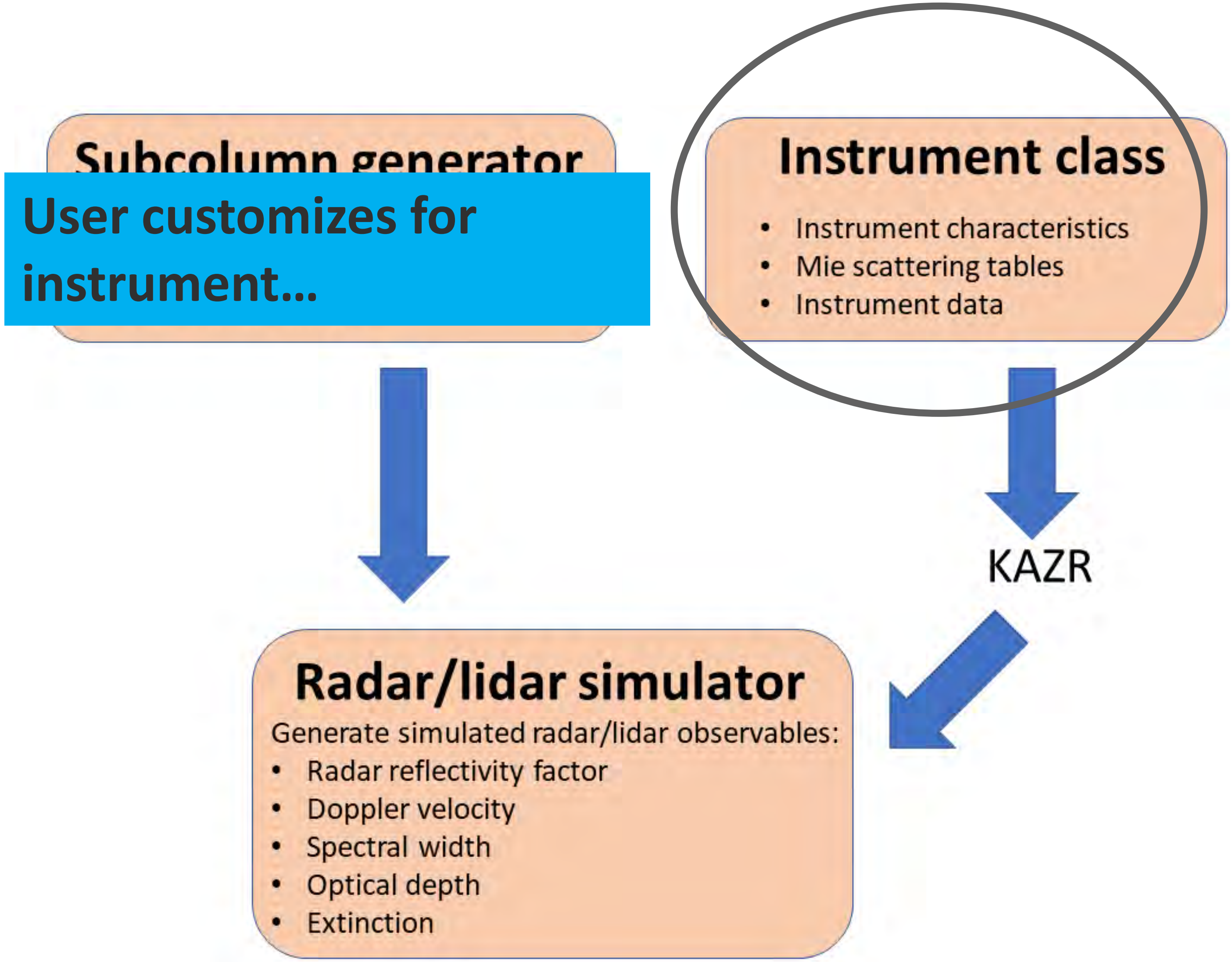
https://columncolab.github.io/EMC2/


Model class
Model variable information
Cloud coverage statistics
Cloud microphysics
Density of scatterers
Terminal velocity of scatterers
ModelE
Subcolumn generator
Allocate hydrometeors to subcolumns:
Cloud coverage statistics
Cloud microphysics



User customizes for model…


Subcolumn generator
Allocate hydrometeors to subcolumns:
• Cloud coverage statistics
• Cloud microphysics
Instrument class
• Instrument characteristics
• Mie scattering tables
• Instrument data
KAZR
Radar/lidar simulator
Generate simulated radar/lidar observables:
• Radar reflectivity factor
• Doppler velocity
• Spectral width
• Optical depth
• Extinction


Subcolumn generator
User customizes for instrument...
Instrument class
Instrument characteristics
Mie scattering tables
Instrument data
KAZR
Radar/lidar simulator
Generate simulated radar/lidar observables:
Radar reflectivity factor
Doppler velocity
Spectral width
Optical depth
Extinction

Radar/lidar simulator

Generate simulated radar/lidar observables:

- Radar reflectivity factor
- Doppler velocity
- Spectral width
- Optical depth
- Extinction

Comparison

- Atmospheric Community Toolkit
- Xarray matplotlib wrappers
- Visualization

EMC^2 demo notebook

In this notebook we show an example of how to run EMC2 using ModelE and HSRL data

In [1]:

```
import emc2
import act
from datetime import datetime
%pylab inline

from distributed import LocalCluster, Client
```

```
Populating the interactive namespace from numpy and matplotlib
```

```
C:\Users\rjackson\AppData\Local\Continuum\anaconda3\envs\emc2_env\lib\site-packages\IPython\core\magics\pylab.py:160: UserWarning: pylab import has clobbered these variables: ['datetime']
`%matplotlib` prevents importing * from pylab and numpy
  "\n`%matplotlib` prevents importing * from pylab and numpy"
```

First, we load the model data using the ModelE object

In [2]:

```
model_path = 'allvars.SCM_AWR_linft_BT0_unNa_noaer.nc'
my_model = emc2.core.model.ModelE(model_path)
```

After that, we load in the HSRL data using the HSRL object.

In [3]:

```
HSRL = emc2.core.instruments.HSRL()
HSRL.read_arm_netcdf_file('awrhsrlM1.a1.20160816.000000.cdf')
```

In [5]:

```python
Cluster = LocalCluster(n_workers=2)
client = Client(Cluster)
client
```

Out[5]:

Client	**Cluster**
Scheduler: tcp://127.0.0.1:64017	**Workers:** 2
Dashboard: http://127.0.0.1:8787/status	**Cores:** 8
	Memory: 17.07 GB

In [ ]:

```python
%time my_model = emc2.simulator.main.make_simulated_data(my_model, HSRL, 4, parallel=True)
```

In [18]:

```python
my_model.ds
```

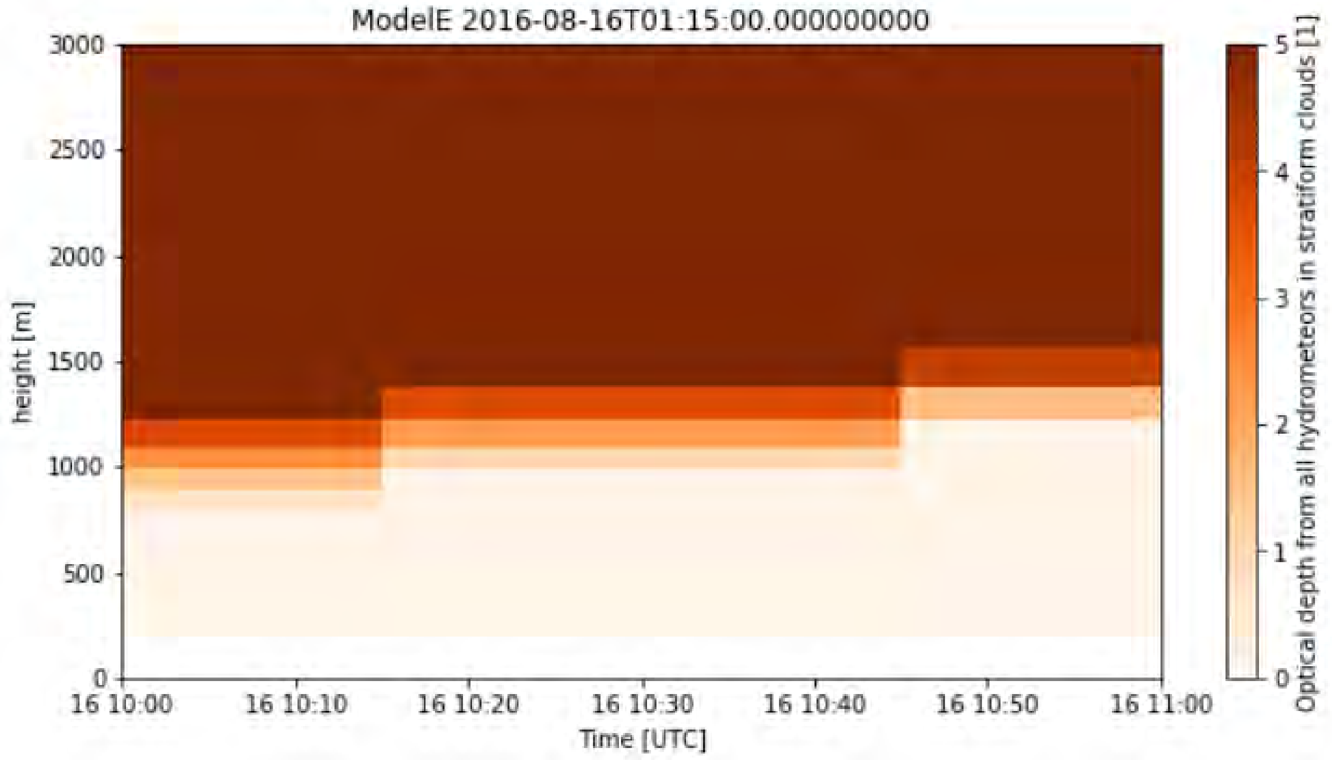
Out[18]:

```
<xarray.Dataset>
Dimensions:        (plm: 63, subcolumn: 16, time: 240)
Coordinates:
  * plm            (plm) float64 995.0 985.0 975.0 ... 0.247 0.139
  * subcolumn      (subcolumn) int64 0 1 2 3 4 5 ... 11 12 13 14 15
  * time           (time) datetime64[ns] 2016-08-16T01:15:00 ... 2016-08-21T00:45:00
    lat            float32 -77.85
    lon            float32 166.72
Data variables:
    QCImc          (time, plm) float32 dask.array<chunksize=(240, 63), meta=np.ndarray>
    QCLmc          (time, plm) float32 dask.array<chunksize=(240, 63), meta=np.ndarray>
    QPImc          (time, plm) float32 dask.array<chunksize=(240, 63), meta=np.ndarray>
    QPLmc          (time, plm) float32 dask.array<chunksize=(240, 63), meta=np.ndarray>
    axyp           float32 dask.array<chunksize=(), meta=np.ndarray>
    cIWPss         (time) float32 dask.array<chunksize=(240,), meta=np.ndarray>
    cLWPss         (time) float32 dask.array<chunksize=(240,), meta=np.ndarray>
```

Visualizing a single subcolumn

In [17]:

```python
model_display = emc2.plotting.SubcolumnDisplay(my_model, subplot_shape=(1,), figsize=(10, 5))
model_display.plot_subcolumn_timeseries('sub_col_OD_tot_strat', 1, subplot_index=(0,), cmap='Oranges', vmin=0, vmax=5,
                                        pressure_coords=False)
model_display.axes[0].set_xlim([datetime.datetime(2016, 8, 16, 10, 0), datetime.datetime(2016, 8, 16, 11, 0)])
model_display.axes[0].set_ylim([0, 3000])
model_display.fig.savefig('Column1.png', dpi=150)
```



Summary

- Framework required for easy "apples-to-apples" comparison of large scale models with observations
- Using radar/lidar simulator code applied to a ModelE and KAZR/HSRL example from AWARE campaign, develop EMC^2
- EMC^2 provides an easy to use, well tested, well documented framework for "apples to apples" model-obs comparisons.
- EMC^2 provides easy to expand framework to include wide array of models/observations
- Future plans include: Expanding EMC^2 to support other climate models/scanning radars

https://columncolab.github.io/EMC2/