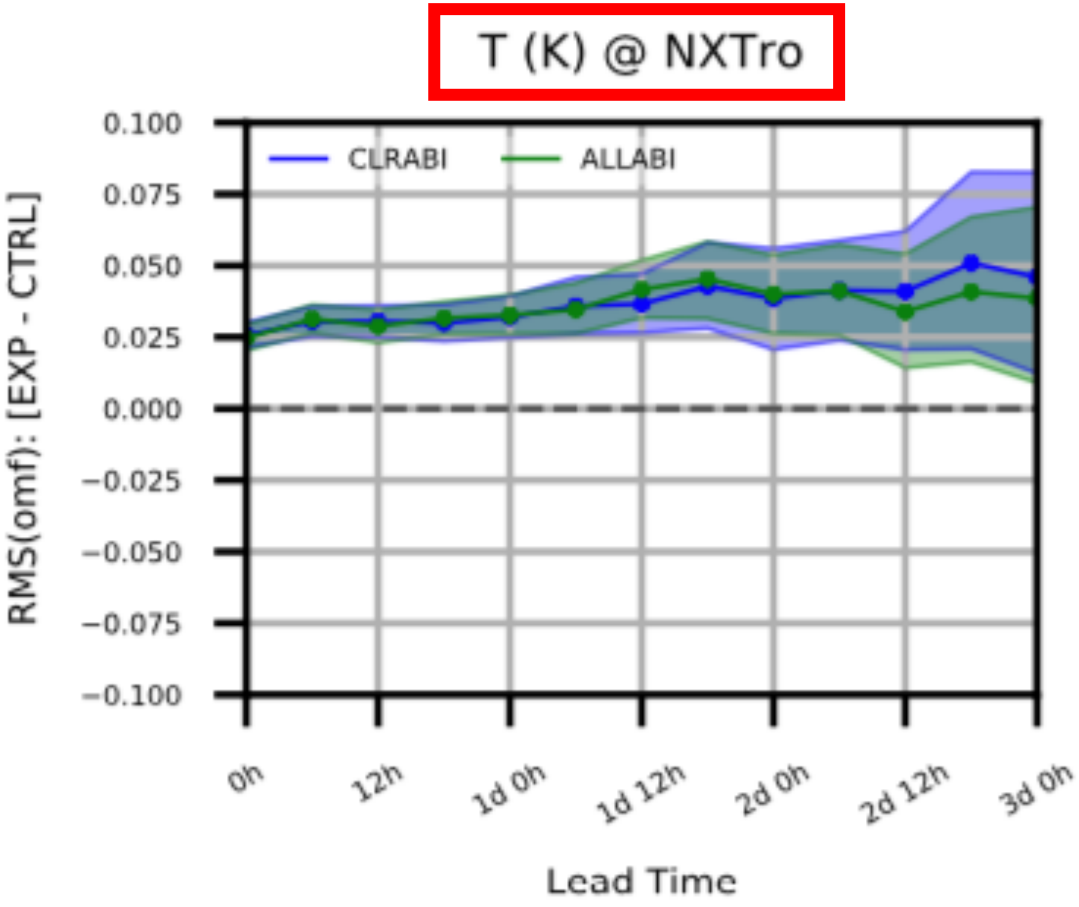
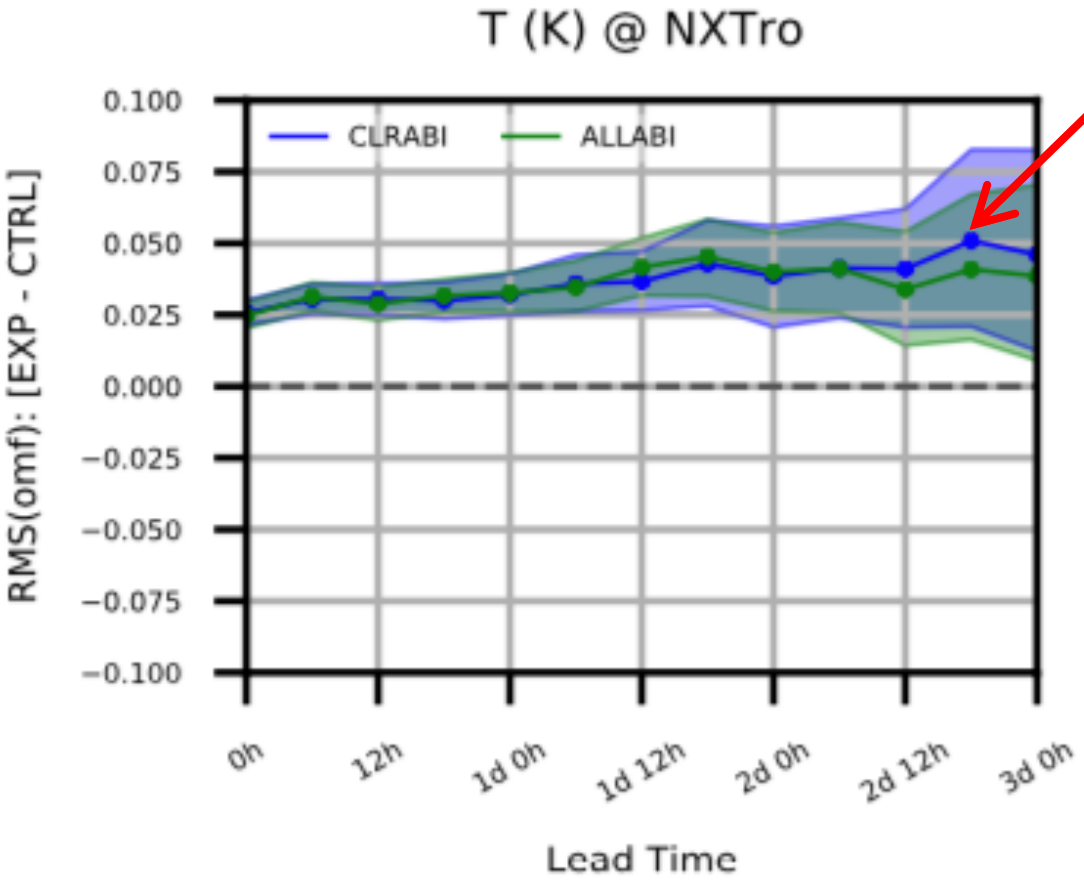
NCAR/MM (PANDA-C) Observation-space Diagnostics Tools

First, an example of the output: Aircraft OMF (0-3day FC) Verification



- y-axis: difference in RMSd between new experiment and prototype-III control experiment
- x-axis: forecast lead-time
- binVal: latitude band (category) + QC is good (category)
- shaded: 95% confidence interval from aggregated bootstrap across 2 x 27 cycles

How does data feed into each plot point?



- RMSd of OMF for CLRABI and CTRL experiments (**EXP**) at 2d18h forecast length (**FC**) from 00Z and 12Z cycles (**CY**) across 27 days
- Reading entire or partial obs, geoval, diag database across many combinations of **EXP/CY/FC** is costly even for moderate location counts (nlocs), and need not be repeated every time figures are generated
- Traditional statistical measures (Count, Mean, RMS, STD, MS, Min, Max) are easily aggregated across independent subpopulations
- Thus RMSd of OMF can be calculated independently for each combination of **EXP/CY/FC**, then aggregated as needed

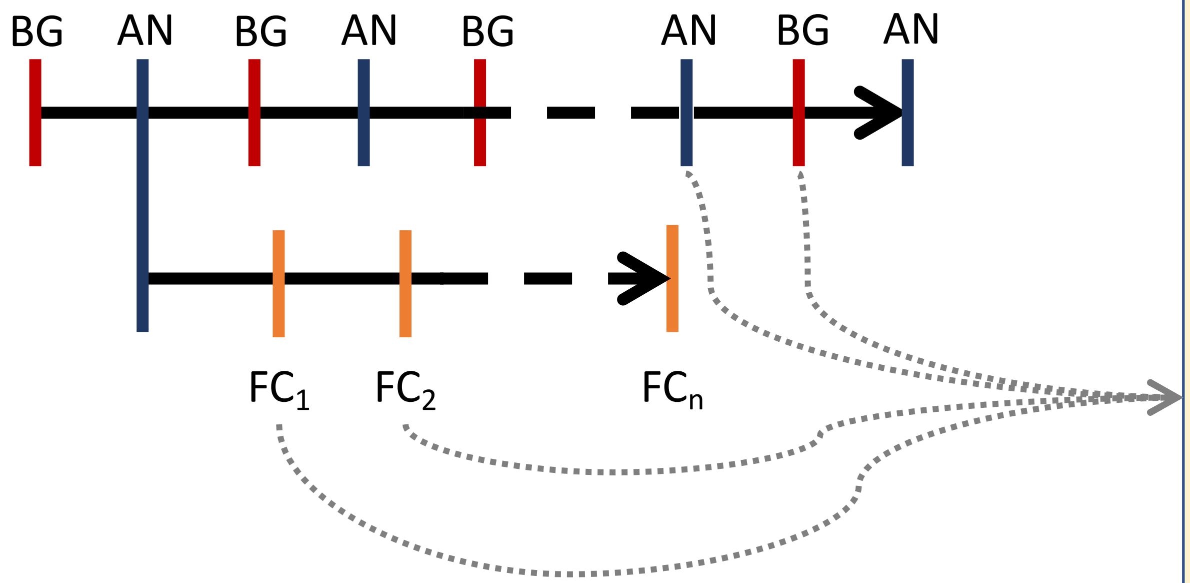
Two-part post-processing

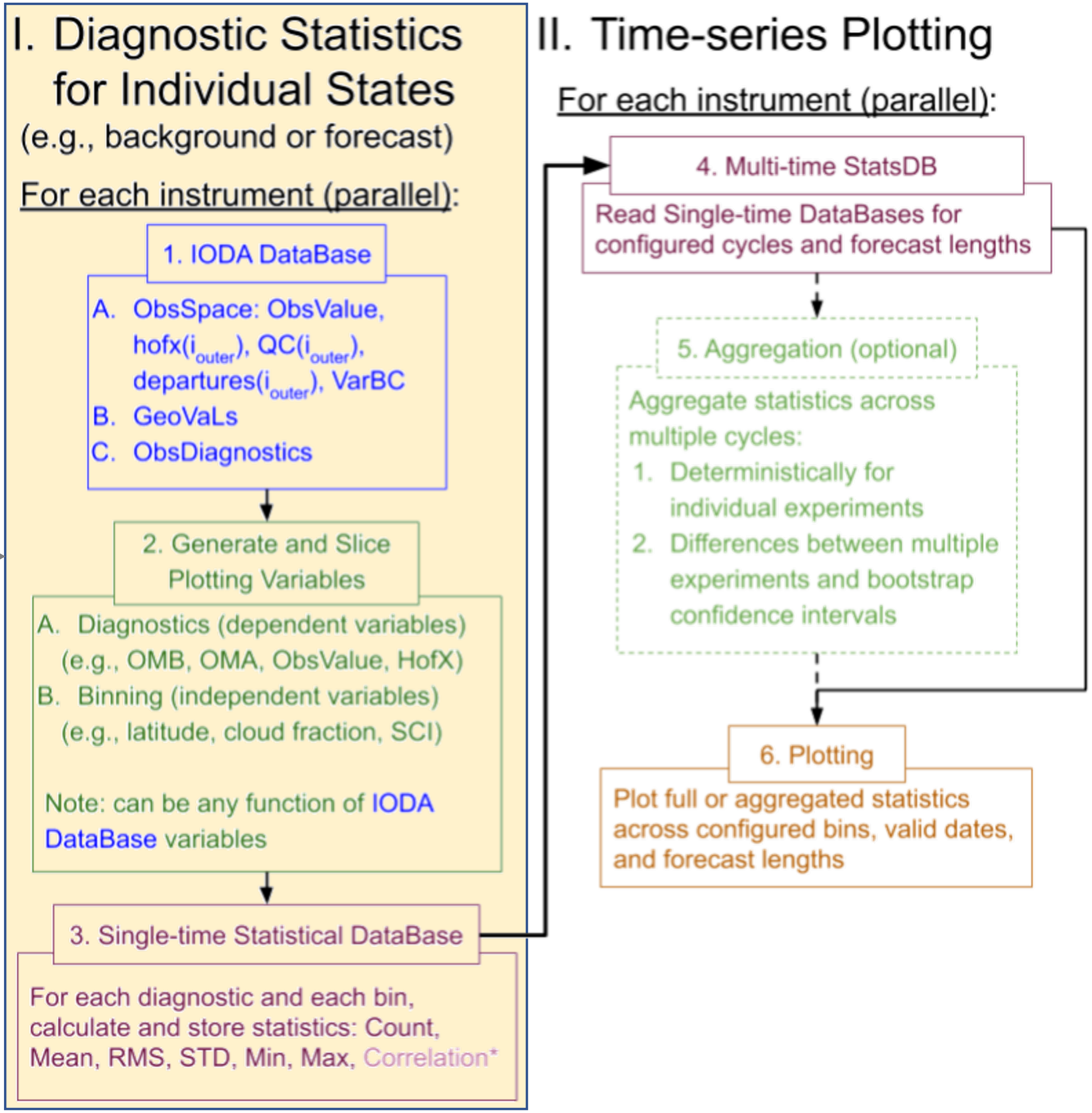
For each observation type (aircraft, sondes, gnssro, amsua_n19, abi_g16, etc…)

1. For each **EXP/CY/FC** index: create statistics database file
   - For each observed variable and each *configured* combination of binning functions and bounds
     - **Bin locations and calculate statistics (computational work)**
   - Write all statistics and metadata to individual database file (netcdf)
2. Generate "analyses" (figures, gross statistical information, etc...)
   - Create "StatsDB" object (wrapper class for a pandas DataFrame object) that includes **EXP/CY/FC** indices specified in *configuration*
   - Create "analyses" based on *configuration* and data available in StatsDB object

Binning function: similar to UFO ObsFunction class; custom-defined function of variables in ObsSpace, GeoVaLs, and ObsDiagnostics, including identity function

Two-part post-processing





I. Diagnostic Statistics for Individual States

(e.g., background or forecast)

For each instrument (parallel):

1. IODA DataBase

A. ObsSpace: ObsValue, hofx(i_{outer}), QC(i_{outer}), departures(i_{outer}), VarBC

B. GeoVaLs

C. ObsDiagnostics

2. Generate and Slice Plotting Variables

A. Diagnostics (dependent variables) (e.g., OMB, OMA, ObsValue, HofX)

B. Binning (independent variables) (e.g., latitude, cloud fraction, SCI)

Note: can be any function of IODA DataBase variables

3. Single-time Statistical DataBase

For each diagnostic and each bin, calculate and store statistics: Count, Mean, RMS, STD, Min, Max, Correlation*

II. Time-series Plotting

For each instrument (parallel):

4. Multi-time StatsDB

Read Single-time DataBases for configured cycles and forecast lengths

5. Aggregation (optional)

Aggregate statistics across multiple cycles:

1. Deterministically for individual experiments
2. Differences between multiple experiments and bootstrap confidence intervals

6. Plotting

Plot full or aggregated statistics across configured bins, valid dates, and forecast lengths

StatsDB class

- Can be sliced across any of these pandas MultiIndex variables:

  expName, fcTDelta, cyDTime, varName,

  diagName, **binVar**, **binVal**, **binMethod**

- Binning values (**binVal**) can be *categorical* (e.g., cloudy, clear, land, sea, latitude band, QC flag) or *continuously varying* (cloud fraction, latitude, zenith angle, glint angle)

- The combination of **binMethod**, **binVar**, and **binVal** enables MANY unique binning strategies to be achieved, e.g., Northern Extratropics, good QC, and zenith angle between 0 and 10 degrees

Aircraft OMF (0-3day FC) Verification

y-axis: difference in RMSd between experiment and control

x-axis: forecast lead-time (**FC**) **binVal**: latitude band (category) + PreQC is good (category)

shaded: 95% confidence interval from aggregated bootstrap across all cycles

lines: 2 experiments

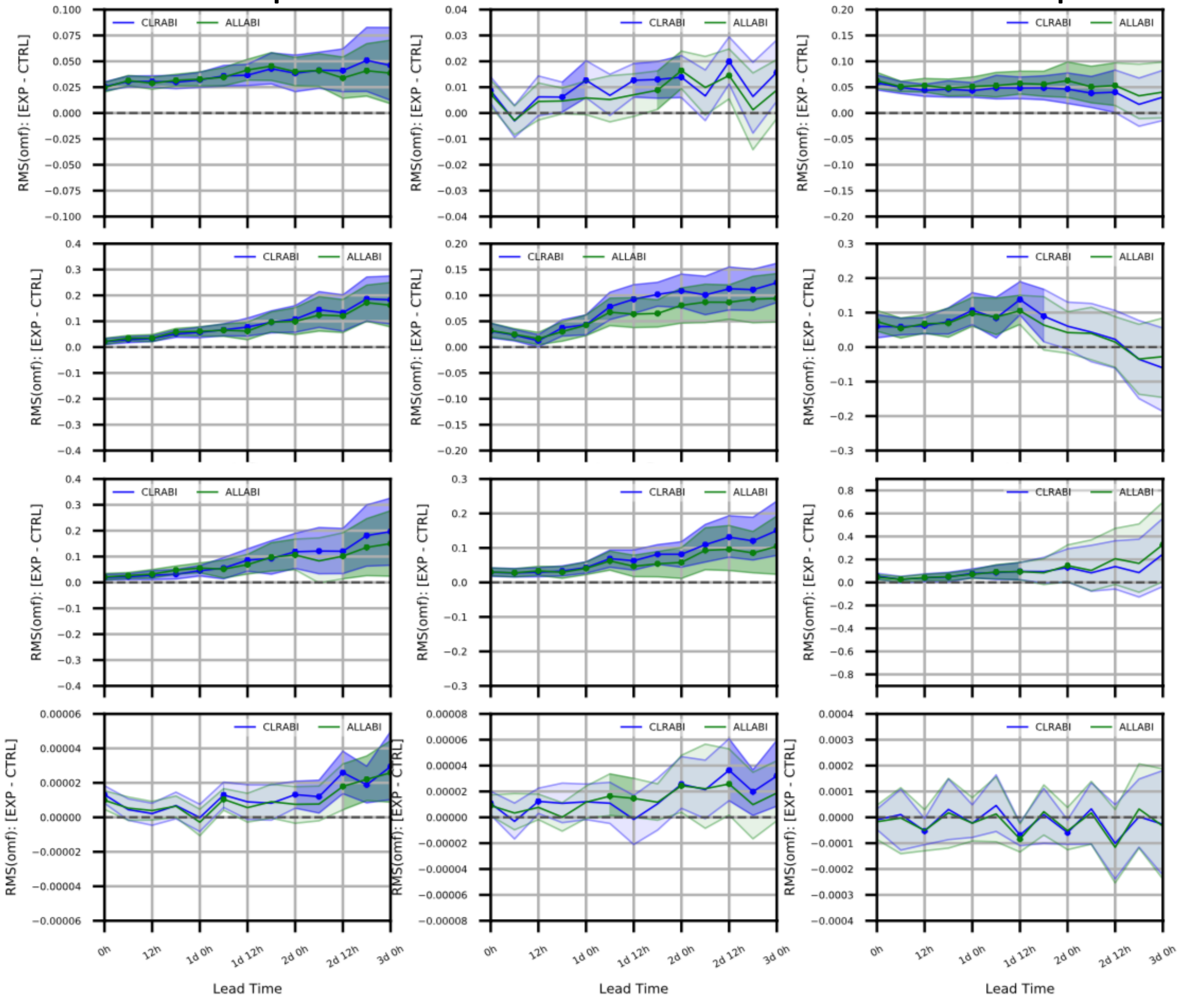
ABI CLRSKY

ABI ALLSKY

Northern Extratropics

Tropics

Southern Extratropics



T (K)

U (m/s)

V (m/s)

Qv (kg/kg)

AMSUA NOAA-19 for one month experiment

y-axis: RMS(OMF-6hr)

x-axis: cycle date (**CY**)

subplots: channels

binVal:

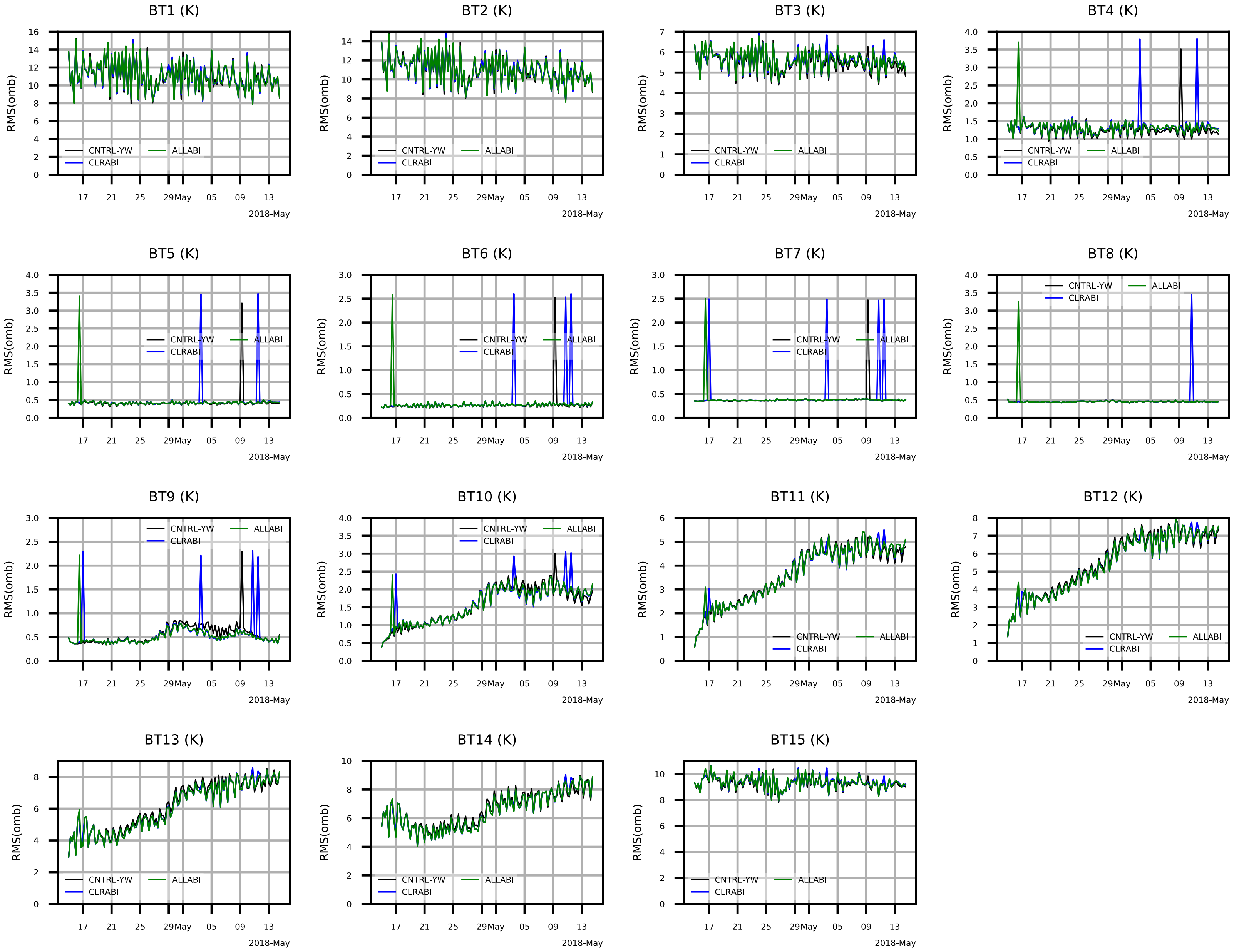
PreQC is GOOD (category)

lines: 3 experiments

CONTROL

ABI CLRSKY

ABI ALLSKY



AHI WV channel OMB (6-hr FC) Verification

x-axis: difference in RMSd between experiment and control

y-axis: latitude **binVals**: cloudiness (category), latitude (1D), QC is GOOD (category)

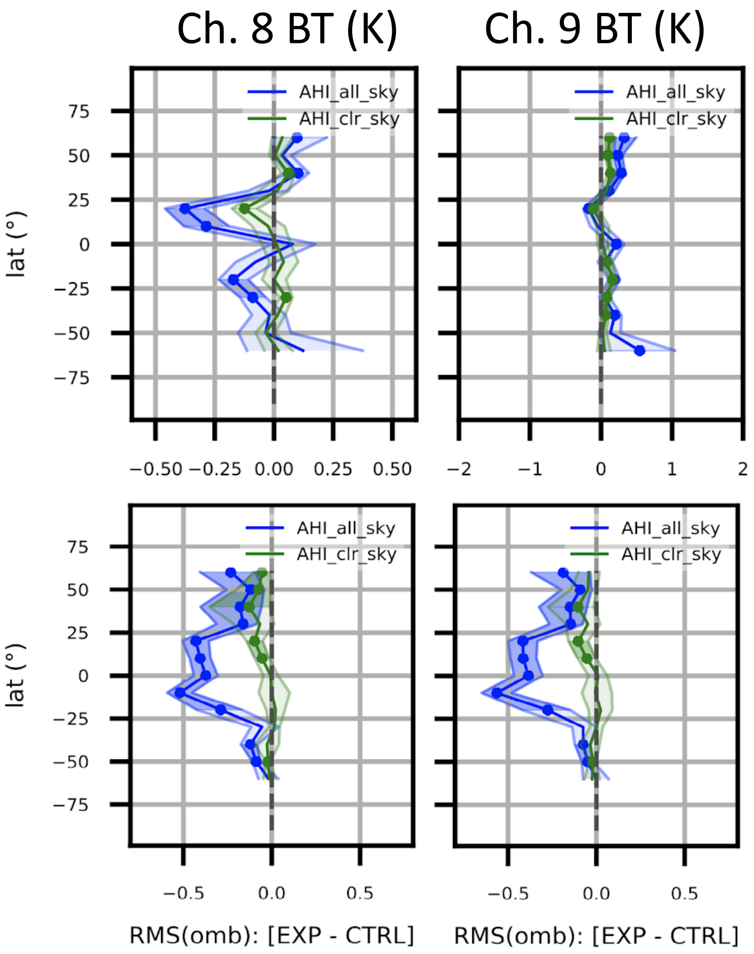
shaded: 95% confidence interval from aggregated bootstrap across all cycles

lines: 2 experiments

AHI CLRSKY

AHI ALLSKY

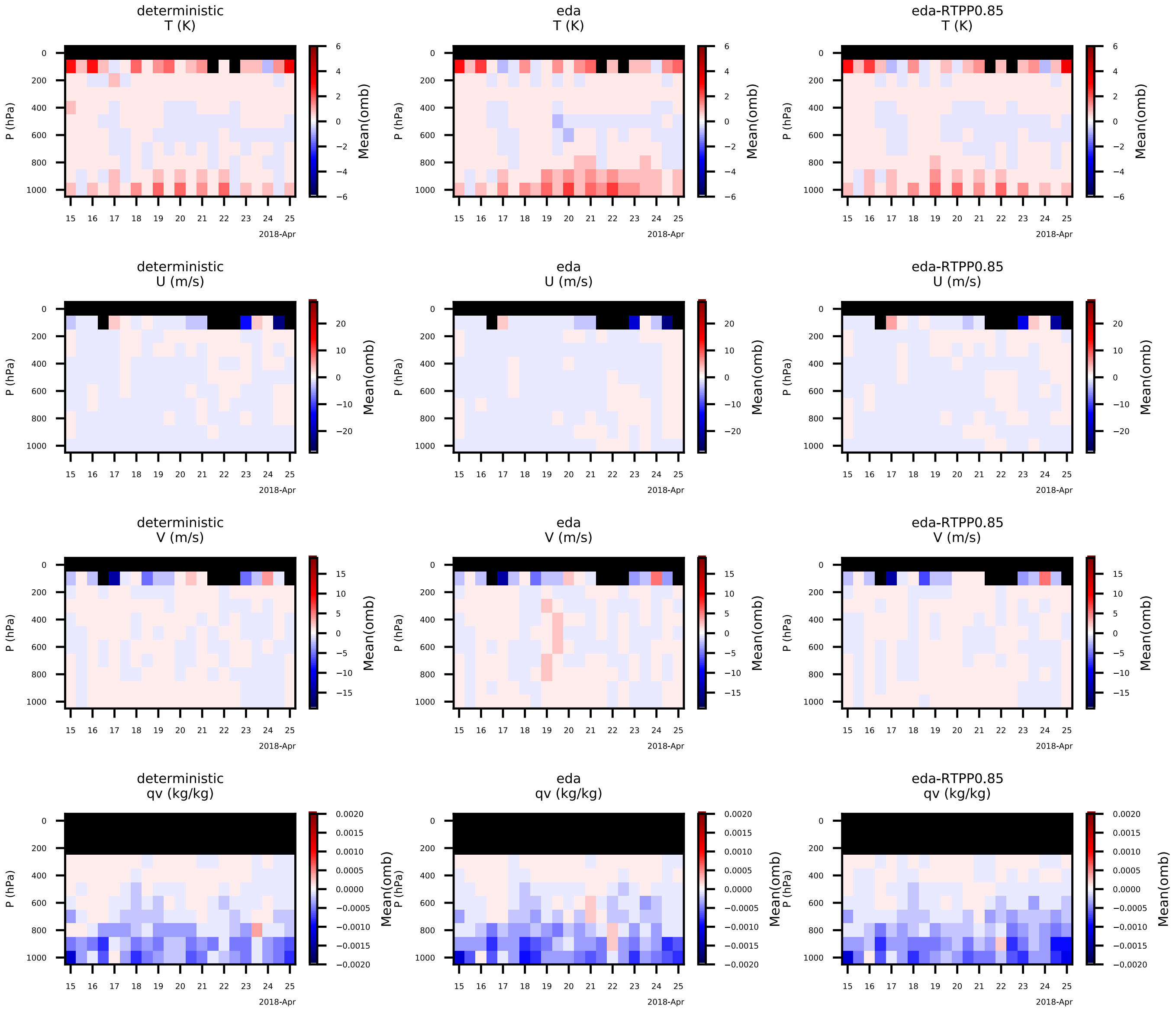
CF ≡ Retrieved Cloud Fraction



CF < 0.05 (clear)

CF > 0.95 (cloudy)

Aircraft OMB (6-hr FC) Verification (2D)



x-axis: cycle date (**CY**)

y-axis: pressure

color: Mean OMB (bias)

columns: 3 experiments

rows: observed variables (T, U, V, qv)

binVals: pressure (1D)

PreQC is GOOD (category)

Notes:

- Fixed pressure bins are selected; cells with no data show up as black
- Any 1D variable can go on y-axis: pressure, height, latitude, local hour, etc…

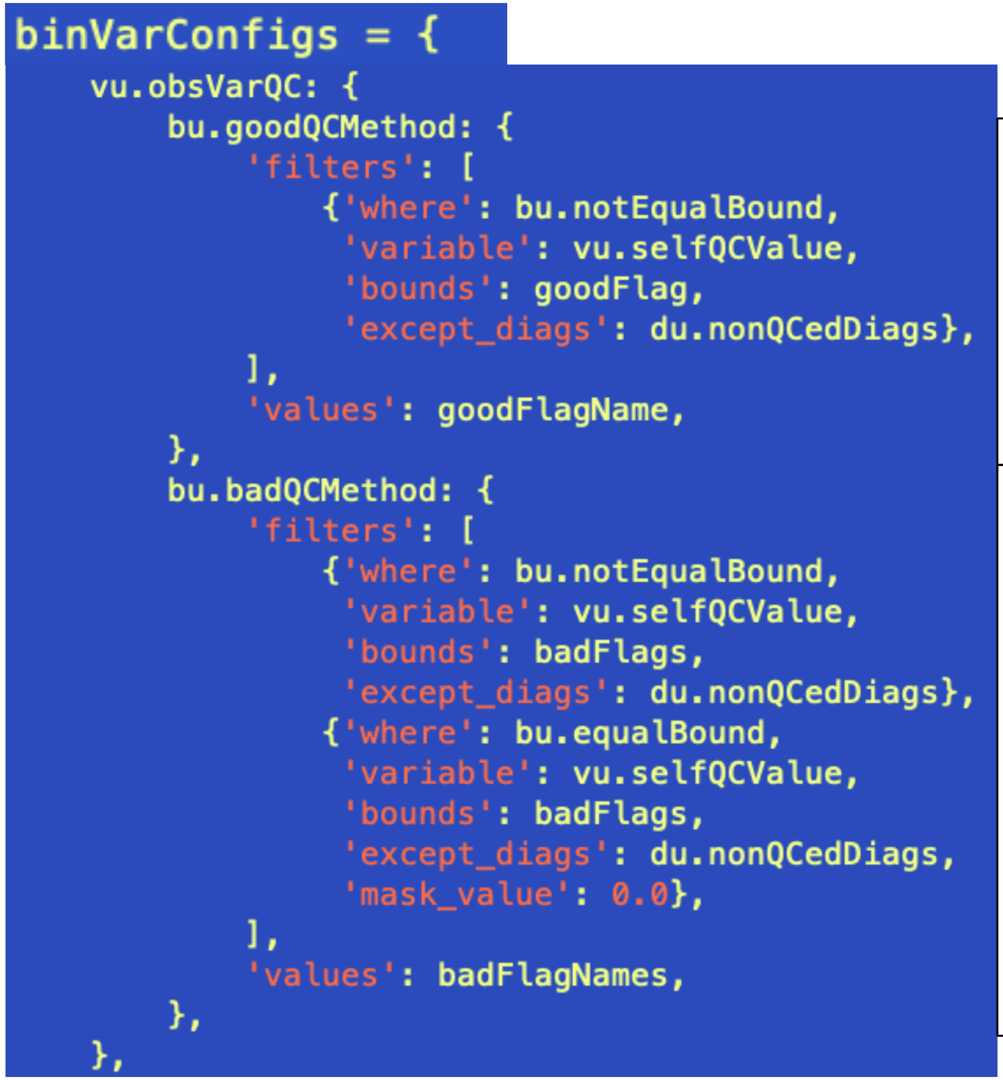
More info

1. Additional product: calculate and print gross statistics for any binning strategy (e.g., all GOOD data)
2. Extensibility:
    - We have all the data slicing needed to create obs-space score cards, with many options for contents
    - Same two-part strategy will work for model-space diagnostics too! The plotting classes/functions from obs-space analyses are reusable due to the generic nature of the StatsDB MultiIndex
    - New binVars and binMethods are added easily as python dictionary entries, but could be replaced with YAML
    - Additional diagnostics are easily added (not just Obs-Model). E.g., ObsValue, HofX, ObsError, or any function of any IODA database variables
    - May be able to add correlation as a statistic, but would require a bit of work

Thank you. Questions?

Extra

binning_configs excerpts



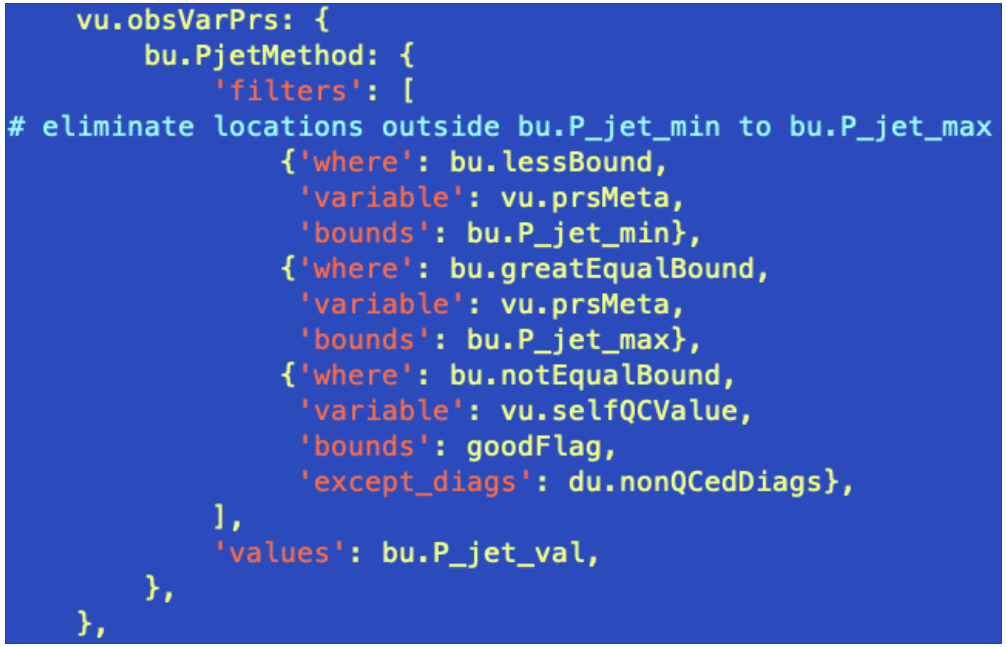
```
binVarConfigs = {
    vu.obsVarQC: {
        bu.goodQCMethod: {
            'filters': [
                {'where': bu.notEqualBound,
                 'variable': vu.selfQCValue,
                 'bounds': goodFlag,
                 'except_diags': du.nonQCedDiags},
            ],
            'values': goodFlagName,
        },
        bu.badQCMethod: {
            'filters': [
                {'where': bu.notEqualBound,
                 'variable': vu.selfQCValue,
                 'bounds': badFlags,
                 'except_diags': du.nonQCedDiags},
                {'where': bu.equalBound,
                 'variable': vu.selfQCValue,
                 'bounds': badFlags,
                 'except_diags': du.nonQCedDiags,
                 'mask_value': 0.0},
            ],
            'values': badFlagNames,
        },
    },
```

All good QC

All bad QC

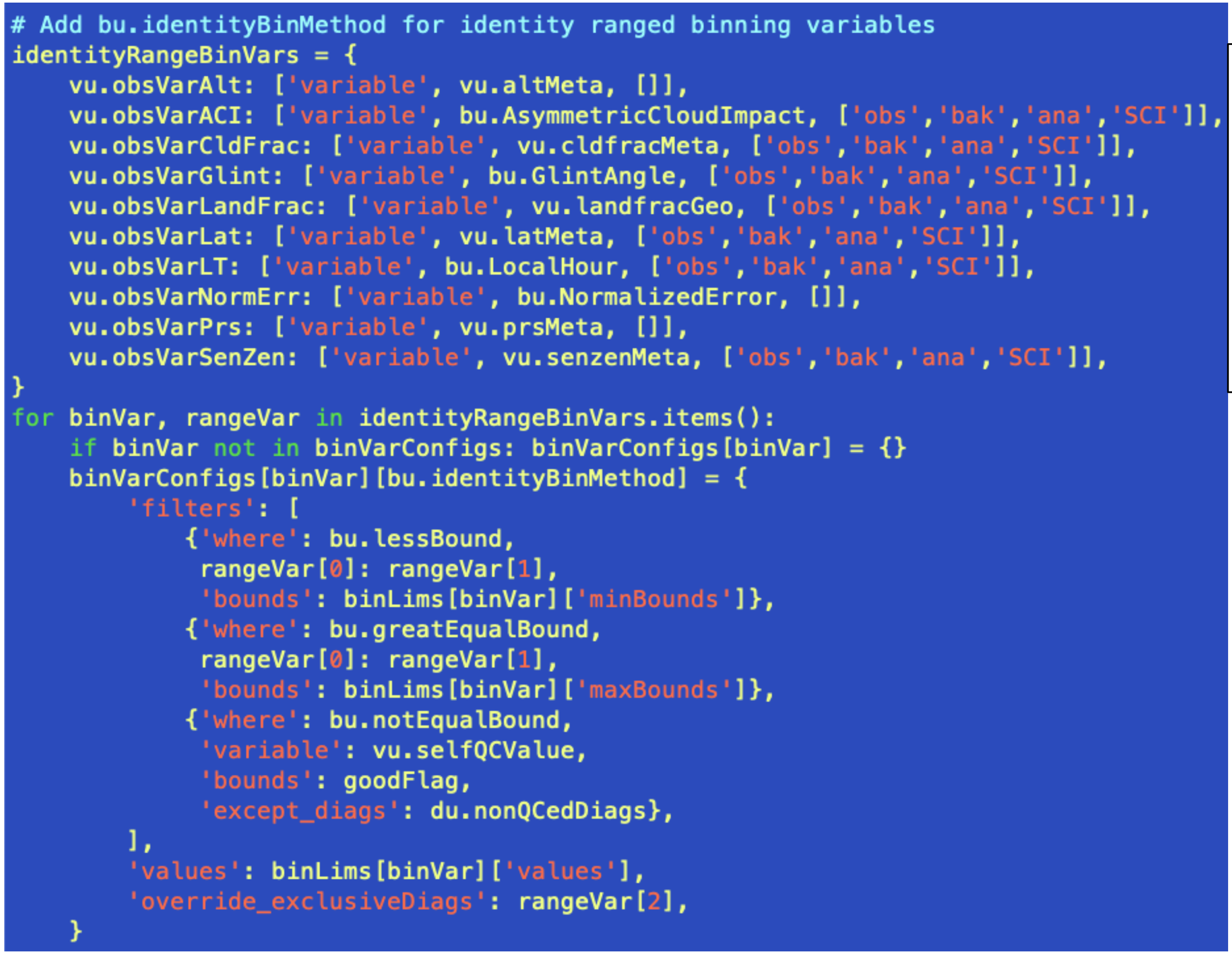
binVar = vu.obsVarQC (iteration dependent)

Jet stream pressure bounds (binVar = vu.obsVarPrs)



```
    vu.obsVarPrs: {
        bu.PjetMethod: {
            'filters': [
# eliminate locations outside bu.P_jet_min to bu.P_jet_max
                {'where': bu.lessBound,
                 'variable': vu.prsMeta,
                 'bounds': bu.P_jet_min},
                {'where': bu.greatEqualBound,
                 'variable': vu.prsMeta,
                 'bounds': bu.P_jet_max},
                {'where': bu.notEqualBound,
                 'variable': vu.selfQCValue,
                 'bounds': goodFlag,
                 'except_diags': du.nonQCedDiags},
            ],
            'values': bu.P_jet_val,
        },
    },
```

binning_configs excerpts

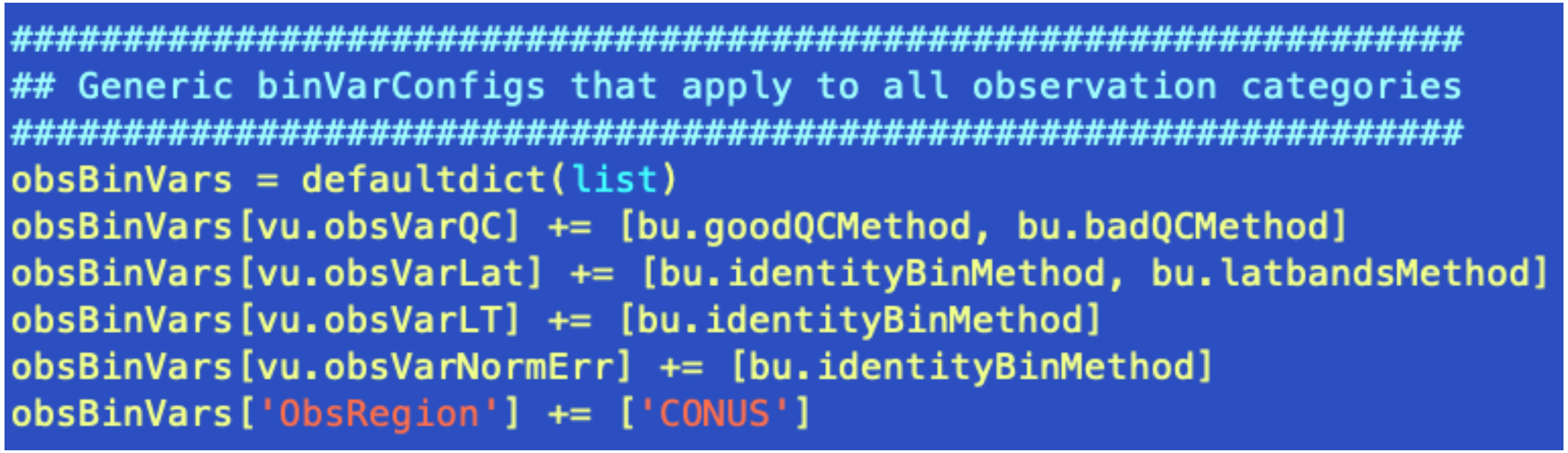


```
# Add bu.identityBinMethod for identity ranged binning variables
identityRangeBinVars = {
    vu.obsVarAlt: ['variable', vu.altMeta, []],
    vu.obsVarACI: ['variable', bu.AsymmetricCloudImpact, ['obs','bak','ana','SCI']],
    vu.obsVarCldFrac: ['variable', vu.cldfracMeta, ['obs','bak','ana','SCI']],
    vu.obsVarGlint: ['variable', bu.GlintAngle, ['obs','bak','ana','SCI']],
    vu.obsVarLandFrac: ['variable', vu.landfracGeo, ['obs','bak','ana','SCI']],
    vu.obsVarLat: ['variable', vu.latMeta, ['obs','bak','ana','SCI']],
    vu.obsVarLT: ['variable', bu.LocalHour, ['obs','bak','ana','SCI']],
    vu.obsVarNormErr: ['variable', bu.NormalizedError, []],
    vu.obsVarPrs: ['variable', vu.prsMeta, []],
    vu.obsVarSenZen: ['variable', vu.senzenMeta, ['obs','bak','ana','SCI']],
}
for binVar, rangeVar in identityRangeBinVars.items():
    if binVar not in binVarConfigs: binVarConfigs[binVar] = {}
    binVarConfigs[binVar][bu.identityBinMethod] = {
        'filters': [
            {'where': bu.lessBound,
             rangeVar[0]: rangeVar[1],
             'bounds': binLims[binVar]['minBounds']},
            {'where': bu.greatEqualBound,
             rangeVar[0]: rangeVar[1],
             'bounds': binLims[binVar]['maxBounds']},
            {'where': bu.notEqualBound,
             'variable': vu.selfQCValue,
             'bounds': goodFlag,
             'except_diags': du.nonQCedDiags},
        ],
        'values': binLims[binVar]['values'],
        'override_exclusiveDiags': rangeVar[2],
    }
```

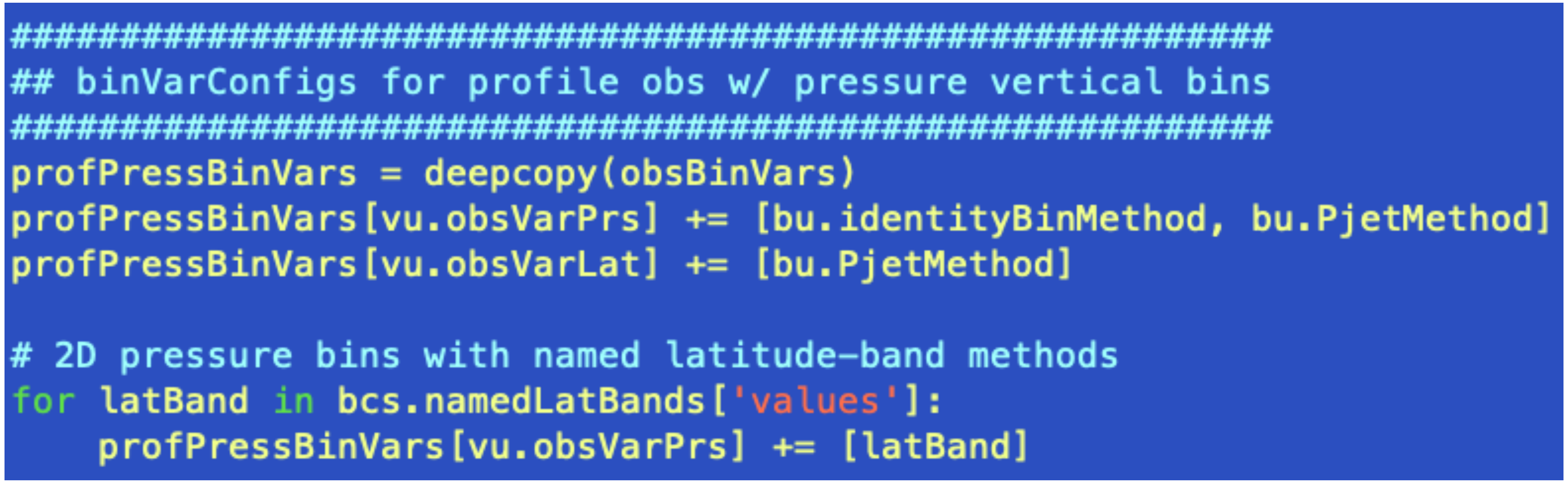
Multiple binVars

All 1D identity binMethods

binMethod selection



```
#################################################################
## Generic binVarConfigs that apply to all observation categories
#################################################################
obsBinVars = defaultdict(list)
obsBinVars[vu.obsVarQC] += [bu.goodQCMethod, bu.badQCMethod]
obsBinVars[vu.obsVarLat] += [bu.identityBinMethod, bu.latbandsMethod]
obsBinVars[vu.obsVarLT] += [bu.identityBinMethod]
obsBinVars[vu.obsVarNormErr] += [bu.identityBinMethod]
obsBinVars['ObsRegion'] += ['CONUS']
```



```
##########################################################
## binVarConfigs for profile obs w/ pressure vertical bins
##########################################################
profPressBinVars = deepcopy(obsBinVars)
profPressBinVars[vu.obsVarPrs] += [bu.identityBinMethod, bu.PjetMethod]
profPressBinVars[vu.obsVarLat] += [bu.PjetMethod]

# 2D pressure bins with named latitude-band methods
for latBand in bcs.namedLatBands['values']:
    profPressBinVars[vu.obsVarPrs] += [latBand]
```