UWILD – University of Washington Ice-Liquid Discriminator

single particle phase classifications and 1 Hz particle size distributions/heterogeneity estimate

Authors

Johannes Mohrmann[1] (lead, corresponding)	jkcm@uw.edu	0000-0002-1025-5192
Joseph Finlon[1]	jfinlon@uw.edu	0000-0003-4304-4698
Rachel Atlas[1]	ratlas@uw.edu	0000-0002-4112-8735
Jeremy Lu[1]	jlu43@uw.edu	0000-0001-5892-8133
Ian Hsiao[1]	ianjjhsiao@gmail.com	0000-0002-7025-2663
Robert Wood[1]	robwood2@uw.edu	0000-0002-1401-3828

[1]University of Washington, Department of Atmospheric Sciences

1.0 Data Set Description:

v1.0, preliminary data

This document describes the data set generated using the University of Washington Ice-Liquid Discriminator, a cloud particle phase classification data set based on observations from the Two-Dimensional Stereo (2D-S) probe. The raw data set consists of particle-by-particle classifications (ice, liquid, or not classified), for each particle imaged by the 2D-S. Also included is a 1-Hz aggregated product. This includes particle size distributions (separated by phase) and a measure of cloud sub-1 Hz heterogeneity.

The data set clouds sampled during the Southern Ocean Clouds, Radiation, Aerosol Transport Experimental Study (SOCRATES) field campaign (McFarquhar et al., 2020), where the 2D-S was flown aboard the NSF/National Center for Atmospheric Research (NCAR) Gulfstream V (G-V) aircraft. Two files (one particle-by-particle classification file, one 1-Hz aggregated classification file) are included for each SOCRATES flight, with the exception of RF15 for which no data is provided. Data are collocated with the GV observational platform, spanning from 30˚S-70˚S, 130˚E-180˚E, and 2018-01-15-2018-02-24.

This data set is supplemented by a companion paper (Atlas et al., 2021), describing the motivation, development, and scientific context of the data set. No restrictions are placed on this data, though any work intending to make use of it should cite both the data repository and the companion paper.

2.0 Instrument Description

This is a derived data set and therefore has no independent instrument. For documentation on the 2D-S probe, see https://www.eol.ucar.edu/instruments/two-dimensional-stereo-particle-imaging-probe.

3.0 Data Collection and Processing

A detailed description of the data processing, with explanation, can be found in the companion paper to this data set (Atlas et al., 2021). This paper also includes the validation and comparison performed on the classifications. The full processing code is available at https://github.com/jkcm/UW-particle-phase-classifier/releases/tag/v1.0. Below is a step-by-step summary:

For the particle-by-particle classifications:

1. Raw 2D-S image strips are processed using the University of Illinois/Oklahoma OAP Processing Software (UIOOPS) processing software (MacFarquhar, 2018) which extracts a feature vector for each particle.
2. Inter-arrival time is added using particle timestamps. Particles are filtered for classification based on the following criteria:
    a. Inter-arrival time must be positive (eliminated bad timestamp particles)
    b. Area ratio must be greater than 0.2
    c. Particle center must be inside the photodiode array
    d. UIOOPS quality flag must be either 48, 72, or 104
    e. Particle pixel count must be > 25 (area-equivalent diameter > 0.056 mm)
    f. Particle fine detail ratio must be greater than 0
3. A training, test, and validation set is selected and used to train a random forest model (see companion paper for details on this process). This trained model is available in the Github repository.
4. The resulting model is run on each processed particle-by-particle 2D-S file (one per flight) to determine the particle phase. Model classification confidence is also recorded.

For the 1-Hz aggregated product:

1. The 1-Hz flight data file (EOL, 2019) is used to get a full list of timesteps for each flight, as well as aircraft speed.
2. By matching the 1-Hz timesteps to the particle-by-particle timesteps, seconds which contain any classified particles are selected. For each second and for both the ice and liquid phase, all matching particles of that phase are selected and binned into 50 logarithmically spaced size bins, according to their area-equivalent diameter assuming spherical particles, creating the phase particle size distributions. These are the "count_darea_*_ml" variables.
3. A second size distribution, using the maximum dimension of a minimum enclosing circle to define particle size, is also generated; these are the "count_dmax_*_ml" variables. These distributions and those described in the previous step, are considered 'deterministic', i.e., they do not take into account model classification confidence.
4. For each 1-Hz sample, as the phase classification from the UWILD scheme is probabilistic, and the model confidence is a good estimator for the probability of correct classification, the model confidence is used to bootstrap 30 realizations of the sample; see companion paper for more details on the justification for this. Thus, for each 1-Hz sample and each size bin, there are 30 possible values for the ice and liquid counts, from which statistics can be drawn. The statistics saved from the bootstrapped samples are, for each size bin: mean count, median count, min count, max count, std deviation, 25^{th} percentile count, and 75^{th} percentile count. A simple usage

would be that the best estimate size distribution is the mean counts per bin from bootstrapping, with the standard deviation indicating the uncertainty in the particle counts.

5. A liquid water content estimate is also calculated assuming spherical particles, both using the deterministic classification ("lwc_ml") as well as using the bootstrapping method ("lwc_ml_mean", "lwc_ml_stddev", etc.).
6. Using the method described in the companion paper, a heterogeneity measure, which corresponds to the likely number of occurrences of adjacent particles of differing phases within a given second, is also calculated and included in the 1-Hz flight file.

4.0 Data Format

Both data file types are available as NetCDF4 files, created using Python 3.7.6 and the xarray library 0.13.0. Encoding is done using the HDF5 backend engine: h5netcdf=0.7.4, hdf5=1.10.5, h5py=2.10.0

4.1 1-Hz Particle Size Distributions and Heterogeneity Parameters

The 1-Hz PSD data has two dimensions: a time dimension corresponding to the 1-Hz flight file time dimension, and a size dimension. File naming convention is as follows, with XX representing the SOCRATES flight number: UW_particle_classifications.1hz.rfXX.nc

Sample header:

```
netcdf UW_particle_classifications.1hz.rf01 {
dimensions:
        time = 26091 ;
        bin_edges = 50 ;
        size_bin = 49 ;
variables:
        int64 time(time) ;
                time:units = "seconds since 2018-01-15T22:50:00" ;
                time:calendar = "proleptic_gregorian" ;
        double bin_edges(bin_edges) ;
                bin_edges:_FillValue = NaN ;
                bin_edges:units = "mm" ;
        double bin_width(size_bin) ;
                bin_width:_FillValue = NaN ;
                bin_width:units = "cm" ;
        double count_dmax_all(time, size_bin) ;
                count_dmax_all:_FillValue = NaN ;
                count_dmax_all:long_name = "Count by max diameter, all particles" ;
                count_dmax_all:units = "count" ;
                count_dmax_all:coordinates = "bin_width" ;
        double count_dmax_liq_ml(time, size_bin) ;
                count_dmax_liq_ml:_FillValue = NaN ;
                count_dmax_liq_ml:long_name = "Count by max diameter, liquid particles, UWILD
classification" ;
                count_dmax_liq_ml:units = "count" ;
                count_dmax_liq_ml:coordinates = "bin_width" ;
        double count_dmax_liq_holroyd(time, size_bin) ;
                count_dmax_liq_holroyd:_FillValue = NaN ;
                count_dmax_liq_holroyd:long_name = "Count by max diameter, liquid particles,
Holroyd classification" ;
```

```
                count_dmax_liq_holroyd:units = "count" ;
                count_dmax_liq_holroyd:coordinates = "bin_width" ;
        double count_dmax_liq_ar(time, size_bin) ;
                count_dmax_liq_ar:_FillValue = NaN ;
                count_dmax_liq_ar:long_name = "Count by max diameter, liquid particles, Area
Ratio classification" ;
                count_dmax_liq_ar:units = "count" ;
                count_dmax_liq_ar:coordinates = "bin_width" ;
        double count_dmax_ice_ml(time, size_bin) ;
                count_dmax_ice_ml:_FillValue = NaN ;
                count_dmax_ice_ml:long_name = "Count by max diameter, ice particles, UWILD
classification" ;
                count_dmax_ice_ml:units = "count" ;
                count_dmax_ice_ml:coordinates = "bin_width" ;
        double count_dmax_ice_holroyd(time, size_bin) ;
                count_dmax_ice_holroyd:_FillValue = NaN ;
                count_dmax_ice_holroyd:long_name = "Count by max diameter, ice particles,
Holroyd classification" ;
                count_dmax_ice_holroyd:units = "count" ;
                count_dmax_ice_holroyd:coordinates = "bin_width" ;
        double count_dmax_ice_ar(time, size_bin) ;
                count_dmax_ice_ar:_FillValue = NaN ;
                count_dmax_ice_ar:long_name = "Count by max diameter, ice particles, Area
Ratio classification" ;
                count_dmax_ice_ar:units = "count" ;
                count_dmax_ice_ar:coordinates = "bin_width" ;
        double count_darea_all(time, size_bin) ;
                count_darea_all:_FillValue = NaN ;
                count_darea_all:long_name = "Count by area-equivalent diameter, all particles"
;
                count_darea_all:units = "count" ;
                count_darea_all:coordinates = "bin_width" ;
        double count_darea_liq_ml(time, size_bin) ;
                count_darea_liq_ml:_FillValue = NaN ;
                count_darea_liq_ml:long_name = "Count by area-equivalent diameter, liquid
particles, UWILD classification" ;
                count_darea_liq_ml:units = "count" ;
                count_darea_liq_ml:coordinates = "bin_width" ;
        double count_darea_liq_holroyd(time, size_bin) ;
                count_darea_liq_holroyd:_FillValue = NaN ;
                count_darea_liq_holroyd:long_name = "Count by area-equivalent diameter, liquid
particles, Holroyd classification" ;
                count_darea_liq_holroyd:units = "count" ;
                count_darea_liq_holroyd:coordinates = "bin_width" ;
        double count_darea_liq_ar(time, size_bin) ;
                count_darea_liq_ar:_FillValue = NaN ;
                count_darea_liq_ar:long_name = "Count by area-equivalent diameter, liquid
particles, Area Ratio classification" ;
                count_darea_liq_ar:units = "count" ;
                count_darea_liq_ar:coordinates = "bin_width" ;
        double count_darea_ice_ml(time, size_bin) ;
                count_darea_ice_ml:_FillValue = NaN ;
                count_darea_ice_ml:long_name = "Count by area-equivalent diameter, ice
particles, UWILD classification" ;
                count_darea_ice_ml:units = "count" ;
                count_darea_ice_ml:coordinates = "bin_width" ;
        double count_darea_ice_holroyd(time, size_bin) ;
                count_darea_ice_holroyd:_FillValue = NaN ;
                count_darea_ice_holroyd:long_name = "Count by area-equivalent diameter, ice
particles, Holroyd classification" ;
                count_darea_ice_holroyd:units = "count" ;
```

```
                count_darea_ice_holroyd:coordinates = "bin_width" ;
        double count_darea_ice_ar(time, size_bin) ;
                count_darea_ice_ar:_FillValue = NaN ;
                count_darea_ice_ar:long_name = "Count by area-equivalent diameter, ice
particles, Area Ratio classification" ;
                count_darea_ice_ar:units = "count" ;
                count_darea_ice_ar:coordinates = "bin_width" ;
        double sample_volume(time, size_bin) ;
                sample_volume:_FillValue = NaN ;
                sample_volume:units = "cm**3" ;
                sample_volume:long_name = "Sample volume" ;
                sample_volume:coordinates = "bin_width" ;
        double lwc_ml(time) ;
                lwc_ml:_FillValue = NaN ;
                lwc_ml:units = "g m**-3" ;
                lwc_ml:long_name = "Liquid Water Content, UWILD classification" ;
        double lwc_holroyd(time) ;
                lwc_holroyd:_FillValue = NaN ;
                lwc_holroyd:units = "g m**-3" ;
                lwc_holroyd:long_name = "Liquid Water Content, Holroyd classification" ;
        double lwc_ar(time) ;
                lwc_ar:_FillValue = NaN ;
                lwc_ar:units = "g m**-3" ;
                lwc_ar:long_name = "Liquid Water Content, Area Ratio classification" ;
        int64 deadtime_flag(time) ;
                deadtime_flag:description = "0: < 80% deadtime for period; 1: Recommend
skipping period due to high dead time" ;
                deadtime_flag:long_name = "Probe deadtime flag" ;
                deadtime_flag:flag_values = 0L, 1L ;
                string deadtime_flag:flag_meanings = "probe_deadtime_less_than_80_percent",
"probe_deadtime_greater_than_80_percent" ;
        double count_dmax_liq_ml_mean(time, size_bin) ;
                count_dmax_liq_ml_mean:_FillValue = NaN ;
                count_dmax_liq_ml_mean:long_name = "Count by max diameter, liquid particles,
UWILD classification mean from bootstrapping" ;
                count_dmax_liq_ml_mean:units = "count" ;
                count_dmax_liq_ml_mean:coordinates = "bin_width" ;
        double count_dmax_ice_ml_mean(time, size_bin) ;
                count_dmax_ice_ml_mean:_FillValue = NaN ;
                count_dmax_ice_ml_mean:long_name = "Count by max diameter, ice particles,
UWILD classification mean from bootstrapping" ;
                count_dmax_ice_ml_mean:units = "count" ;
                count_dmax_ice_ml_mean:coordinates = "bin_width" ;
        double count_darea_liq_ml_mean(time, size_bin) ;
                count_darea_liq_ml_mean:_FillValue = NaN ;
                count_darea_liq_ml_mean:long_name = "Count by area-equivalent diameter, liquid
particles, UWILD classification mean from bootstrapping" ;
                count_darea_liq_ml_mean:units = "count" ;
                count_darea_liq_ml_mean:coordinates = "bin_width" ;
        double count_darea_ice_ml_mean(time, size_bin) ;
                count_darea_ice_ml_mean:_FillValue = NaN ;
                count_darea_ice_ml_mean:long_name = "Count by area-equivalent diameter, ice
particles, UWILD classification mean from bootstrapping" ;
                count_darea_ice_ml_mean:units = "count" ;
                count_darea_ice_ml_mean:coordinates = "bin_width" ;
        double count_dmax_liq_ml_median(time, size_bin) ;
                count_dmax_liq_ml_median:_FillValue = NaN ;
                count_dmax_liq_ml_median:long_name = "Count by max diameter, liquid particles,
UWILD classification median from bootstrapping" ;
                count_dmax_liq_ml_median:units = "count" ;
                count_dmax_liq_ml_median:coordinates = "bin_width" ;
```

```
        double count_dmax_ice_ml_median(time, size_bin) ;
                count_dmax_ice_ml_median:_FillValue = NaN ;
                count_dmax_ice_ml_median:long_name = "Count by max diameter, ice particles,
UWILD classification median from bootstrapping" ;
                count_dmax_ice_ml_median:units = "count" ;
                count_dmax_ice_ml_median:coordinates = "bin_width" ;
        double count_darea_liq_ml_median(time, size_bin) ;
                count_darea_liq_ml_median:_FillValue = NaN ;
                count_darea_liq_ml_median:long_name = "Count by area-equivalent diameter,
liquid particles, UWILD classification median from bootstrapping" ;
                count_darea_liq_ml_median:units = "count" ;
                count_darea_liq_ml_median:coordinates = "bin_width" ;
        double count_darea_ice_ml_median(time, size_bin) ;
                count_darea_ice_ml_median:_FillValue = NaN ;
                count_darea_ice_ml_median:long_name = "Count by area-equivalent diameter, ice
particles, UWILD classification median from bootstrapping" ;
                count_darea_ice_ml_median:units = "count" ;
                count_darea_ice_ml_median:coordinates = "bin_width" ;
        double count_dmax_liq_ml_min(time, size_bin) ;
                count_dmax_liq_ml_min:_FillValue = NaN ;
                count_dmax_liq_ml_min:long_name = "Count by max diameter, liquid particles,
UWILD classification min from bootstrapping" ;
                count_dmax_liq_ml_min:units = "count" ;
                count_dmax_liq_ml_min:coordinates = "bin_width" ;
        double count_dmax_ice_ml_min(time, size_bin) ;
                count_dmax_ice_ml_min:_FillValue = NaN ;
                count_dmax_ice_ml_min:long_name = "Count by max diameter, ice particles, UWILD
classification min from bootstrapping" ;
                count_dmax_ice_ml_min:units = "count" ;
                count_dmax_ice_ml_min:coordinates = "bin_width" ;
        double count_darea_liq_ml_min(time, size_bin) ;
                count_darea_liq_ml_min:_FillValue = NaN ;
                count_darea_liq_ml_min:long_name = "Count by area-equivalent diameter, liquid
particles, UWILD classification min from bootstrapping" ;
                count_darea_liq_ml_min:units = "count" ;
                count_darea_liq_ml_min:coordinates = "bin_width" ;
        double count_darea_ice_ml_min(time, size_bin) ;
                count_darea_ice_ml_min:_FillValue = NaN ;
                count_darea_ice_ml_min:long_name = "Count by area-equivalent diameter, ice
particles, UWILD classification min from bootstrapping" ;
                count_darea_ice_ml_min:units = "count" ;
                count_darea_ice_ml_min:coordinates = "bin_width" ;
        double count_dmax_liq_ml_max(time, size_bin) ;
                count_dmax_liq_ml_max:_FillValue = NaN ;
                count_dmax_liq_ml_max:long_name = "Count by max diameter, liquid particles,
UWILD classification max from bootstrapping" ;
                count_dmax_liq_ml_max:units = "count" ;
                count_dmax_liq_ml_max:coordinates = "bin_width" ;
        double count_dmax_ice_ml_max(time, size_bin) ;
                count_dmax_ice_ml_max:_FillValue = NaN ;
                count_dmax_ice_ml_max:long_name = "Count by max diameter, ice particles, UWILD
classification max from bootstrapping" ;
                count_dmax_ice_ml_max:units = "count" ;
                count_dmax_ice_ml_max:coordinates = "bin_width" ;
        double count_darea_liq_ml_max(time, size_bin) ;
                count_darea_liq_ml_max:_FillValue = NaN ;
                count_darea_liq_ml_max:long_name = "Count by area-equivalent diameter, liquid
particles, UWILD classification max from bootstrapping" ;
                count_darea_liq_ml_max:units = "count" ;
                count_darea_liq_ml_max:coordinates = "bin_width" ;
        double count_darea_ice_ml_max(time, size_bin) ;
```

```
                count_darea_ice_ml_max:_FillValue = NaN ;
                count_darea_ice_ml_max:long_name = "Count by area-equivalent diameter, ice
particles, UWILD classification max from bootstrapping" ;
                count_darea_ice_ml_max:units = "count" ;
                count_darea_ice_ml_max:coordinates = "bin_width" ;
        double count_dmax_liq_ml_stddev(time, size_bin) ;
                count_dmax_liq_ml_stddev:_FillValue = NaN ;
                count_dmax_liq_ml_stddev:long_name = "Count by max diameter, liquid particles,
UWILD classification standard deviation from bootstrapping" ;
                count_dmax_liq_ml_stddev:units = "count" ;
                count_dmax_liq_ml_stddev:coordinates = "bin_width" ;
        double count_dmax_ice_ml_stddev(time, size_bin) ;
                count_dmax_ice_ml_stddev:_FillValue = NaN ;
                count_dmax_ice_ml_stddev:long_name = "Count by max diameter, ice particles,
UWILD classification standard deviation from bootstrapping" ;
                count_dmax_ice_ml_stddev:units = "count" ;
                count_dmax_ice_ml_stddev:coordinates = "bin_width" ;
        double count_darea_liq_ml_stddev(time, size_bin) ;
                count_darea_liq_ml_stddev:_FillValue = NaN ;
                count_darea_liq_ml_stddev:long_name = "Count by area-equivalent diameter,
liquid particles, UWILD classification standard deviation from bootstrapping" ;
                count_darea_liq_ml_stddev:units = "count" ;
                count_darea_liq_ml_stddev:coordinates = "bin_width" ;
        double count_darea_ice_ml_stddev(time, size_bin) ;
                count_darea_ice_ml_stddev:_FillValue = NaN ;
                count_darea_ice_ml_stddev:long_name = "Count by area-equivalent diameter, ice
particles, UWILD classification standard deviation from bootstrapping" ;
                count_darea_ice_ml_stddev:units = "count" ;
                count_darea_ice_ml_stddev:coordinates = "bin_width" ;
        double count_dmax_liq_ml_25pct(time, size_bin) ;
                count_dmax_liq_ml_25pct:_FillValue = NaN ;
                count_dmax_liq_ml_25pct:long_name = "Count by max diameter, liquid particles,
UWILD classification 25th percentile from bootstrapping" ;
                count_dmax_liq_ml_25pct:units = "count" ;
                count_dmax_liq_ml_25pct:coordinates = "bin_width" ;
        double count_dmax_ice_ml_25pct(time, size_bin) ;
                count_dmax_ice_ml_25pct:_FillValue = NaN ;
                count_dmax_ice_ml_25pct:long_name = "Count by max diameter, ice particles,
UWILD classification 25th percentile from bootstrapping" ;
                count_dmax_ice_ml_25pct:units = "count" ;
                count_dmax_ice_ml_25pct:coordinates = "bin_width" ;
        double count_darea_liq_ml_25pct(time, size_bin) ;
                count_darea_liq_ml_25pct:_FillValue = NaN ;
                count_darea_liq_ml_25pct:long_name = "Count by area-equivalent diameter,
liquid particles, UWILD classification 25th percentile from bootstrapping" ;
                count_darea_liq_ml_25pct:units = "count" ;
                count_darea_liq_ml_25pct:coordinates = "bin_width" ;
        double count_darea_ice_ml_25pct(time, size_bin) ;
                count_darea_ice_ml_25pct:_FillValue = NaN ;
                count_darea_ice_ml_25pct:long_name = "Count by area-equivalent diameter, ice
particles, UWILD classification 25th percentile from bootstrapping" ;
                count_darea_ice_ml_25pct:units = "count" ;
                count_darea_ice_ml_25pct:coordinates = "bin_width" ;
        double count_dmax_liq_ml_75pct(time, size_bin) ;
                count_dmax_liq_ml_75pct:_FillValue = NaN ;
                count_dmax_liq_ml_75pct:long_name = "Count by max diameter, liquid particles,
UWILD classification 75th percentile from bootstrapping" ;
                count_dmax_liq_ml_75pct:units = "count" ;
                count_dmax_liq_ml_75pct:coordinates = "bin_width" ;
        double count_dmax_ice_ml_75pct(time, size_bin) ;
                count_dmax_ice_ml_75pct:_FillValue = NaN ;
```

```
                count_dmax_ice_ml_75pct:long_name = "Count by max diameter, ice particles,
UWILD classification 75th percentile from bootstrapping" ;
                count_dmax_ice_ml_75pct:units = "count" ;
                count_dmax_ice_ml_75pct:coordinates = "bin_width" ;
        double count_darea_liq_ml_75pct(time, size_bin) ;
                count_darea_liq_ml_75pct:_FillValue = NaN ;
                count_darea_liq_ml_75pct:long_name = "Count by area-equivalent diameter,
liquid particles, UWILD classification 75th percentile from bootstrapping" ;
                count_darea_liq_ml_75pct:units = "count" ;
                count_darea_liq_ml_75pct:coordinates = "bin_width" ;
        double count_darea_ice_ml_75pct(time, size_bin) ;
                count_darea_ice_ml_75pct:_FillValue = NaN ;
                count_darea_ice_ml_75pct:long_name = "Count by area-equivalent diameter, ice
particles, UWILD classification 75th percentile from bootstrapping" ;
                count_darea_ice_ml_75pct:units = "count" ;
                count_darea_ice_ml_75pct:coordinates = "bin_width" ;
        double lwc_ml_mean(time) ;
                lwc_ml_mean:_FillValue = NaN ;
                lwc_ml_mean:long_name = "Liquid Water Content, UWILD classification mean from
bootstrapping" ;
                lwc_ml_mean:units = "g m**-3" ;
        double lwc_ml_median(time) ;
                lwc_ml_median:_FillValue = NaN ;
                lwc_ml_median:long_name = "Liquid Water Content, UWILD classification median
from bootstrapping" ;
                lwc_ml_median:units = "g m**-3" ;
        double lwc_ml_min(time) ;
                lwc_ml_min:_FillValue = NaN ;
                lwc_ml_min:long_name = "Liquid Water Content, UWILD classification min from
bootstrapping" ;
                lwc_ml_min:units = "g m**-3" ;
        double lwc_ml_max(time) ;
                lwc_ml_max:_FillValue = NaN ;
                lwc_ml_max:long_name = "Liquid Water Content, UWILD classification max from
bootstrapping" ;
                lwc_ml_max:units = "g m**-3" ;
        double lwc_ml_stddev(time) ;
                lwc_ml_stddev:_FillValue = NaN ;
                lwc_ml_stddev:long_name = "Liquid Water Content, UWILD classification standard
deviation from bootstrapping" ;
                lwc_ml_stddev:units = "g m**-3" ;
        double lwc_ml_25pct(time) ;
                lwc_ml_25pct:_FillValue = NaN ;
                lwc_ml_25pct:long_name = "Liquid Water Content, UWILD classification 25th
percentile from bootstrapping" ;
                lwc_ml_25pct:units = "g m**-3" ;
        double lwc_ml_75pct(time) ;
                lwc_ml_75pct:_FillValue = NaN ;
                lwc_ml_75pct:long_name = "Liquid Water Content, UWILD classification 75th
percentile from bootstrapping" ;
                lwc_ml_75pct:units = "g m**-3" ;
        float particle_counts(time) ;
                particle_counts:_FillValue = NaNf ;
                particle_counts:long_name = "number of classified particles per timestep" ;
                particle_counts:units = "count" ;
        float phase_flip_counts(time) ;
                phase_flip_counts:_FillValue = NaNf ;
                phase_flip_counts:long_name = "probabilistic estimate of number of flips from
ice to liquid or liquid to ice of adjacent particles" ;
                phase_flip_counts:units = "count" ;
        float all_particle_counts(time) ;
```

```
                all_particle_counts:_FillValue = NaNf ;
                all_particle_counts:long_name = "number of classified+unclassified particles
per timestep" ;
                all_particle_counts:units = "count" ;

// global attributes:
                :version = "1.0" ;
                :Contact = "Johannes Mohrmann (jkcm@uw.edu)" ;
                :Institution = "University of Washington, Dept. of Atmospheric Sciences" ;
                :Creation\ Time = "2021-06-30 13:32:40.295865" ;
                :Project = "UW random forest particle phase identification, 1 Hz particle size
distributions" ;
                :Website = "https://github.com/jkcm/UW-particle-phase-classifier" ;
                :References = "Random forest relies on output from the UIOPS 2DS processing
suite, DOI:10.5281/zenodo.3667054" ;
                :authors = "Rachel Atlas, Joe Finlon, Ian Hsiao, Jeremy Lu, Johannes Mohrmann"
;
                :_NCProperties = "version=2,netcdf=4.6.2,hdf5=1.10.5" ;}
```

4.2 Particle-by-particle classifications

The particle-by-particle data has one dimension ('time'), with time monotonically increasing along this dimension. This merely counts up from 0 to the total particle count; the datetime associated with each particle is stored in a separate datetime variable. File naming convention is as follows, with XX representing the SOCRATES flight number: UW_particle_classifications.pbp.rfXX.nc

The "UW_flag" flag is used to encode data quality in a bitwise fashion:

- 0th bit: inter-arrival time was negative
- 1st bit: area ratio was below 0.2
- 2nd bit: particle center was outside of image array
- 3rd bit: UIOOPS auto-reject flag was one not of 48, 72, 104 (see UIOOPS documentation)
- 4th bit: Particle size was below 25 pixels
- 5th bit: There was missing data in the UIOOPS feature array
- 6th bit: Fine detail ratio was below 0

All these bits are set when a negative condition is met, and therefore **a value of 0 for the variable indicates good data**. Example Python code for filtering data:

```
data = xarray.open_dataset(data_filename)
uw_flag = data.UW_flag

#get indices of good data, no flag bits set
good_data_idx = uw_flag==0

#get indices where ONLY the particle-size-too-small bit (#4) is set; rightmost bit is bit#0
only_particle_too_small_idx = uw_flag==int(0b10000)

#get indices where both the particles-size-too-small (#4) and the negative IAT bit (#0) is set
too_small_and_neg_iat_idx = uw_flag==int(0b10001)

#get indices where particle-too-small bit (#4) is set, ignoring other bits, with bit shifting
particle_too_small_idx = (uw_flag.values>>4) & 1

data_subset = data.isel(time=good_data_idx)
```

Sample header:

```
netcdf UW_particle_classifications.1hz.rf03 {
dimensions:
        time = 15797629 ;
variables:
        double AR_threshold_phase(time) ;
                AR_threshold_phase:_FillValue = NaN ;
                string AR_threshold_phase:Units = "0:ice_1:liquid" ;
                string AR_threshold_phase:Description = "area ratio threshold classification,
AR>0.5=liquid" ;
        double Holroyd_phase(time) ;
                Holroyd_phase:_FillValue = NaN ;
                string Holroyd_phase:Units = "0:ice_1:liquid" ;
                string Holroyd_phase:Description = "Holroyd habit phase classification.
Habit=round=liquid, or small with poisson spot=liquid" ;
        double UW_certainty(time) ;
                UW_certainty:_FillValue = NaN ;
                string UW_certainty:Units = "0-1" ;
                string UW_certainty:Description = "UW phase prediction certainty" ;
        ubyte UW_flag(time) ;
                string UW_flag:Units = "--" ;
                string UW_flag:Description = "UW data quality flag: 0 is good values, 0th bit
is interarrival time=negative, 1st bit is area ratio index< 0.2, 2nd bit is image center not
in particle, 3rd bit is UIOPS auto_reject index is not in (48, 72, 104), 4th bit is size is
above 25 pixels, 5th bit is NaN in input data, 6th bit is negative fine detail ratios" ;
        double UW_phase(time) ;
                UW_phase:_FillValue = NaN ;
                string UW_phase:Units = "0:ice_1:liquid" ;
                string UW_phase:Description = "UW random forest phase classification,
1=liquid, 0=ice" ;
        double datetime(time) ;
                datetime:_FillValue = NaN ;
                string datetime:units = "seconds since 2018-01-22 20:48:52.406000" ;
                string datetime:calendar = "proleptic_gregorian" ;

// global attributes:
                string :Contact = "Johannes Mohrmann (jkcm@uw.edu)" ;
                string :Institution = "University of Washington, Dept. of Atmospheric
Sciences" ;
                string :Creation\ Time = "2021-03-05 09:07:06.808952" ;
                string :Project = "UW random forest particle phase identification" ;
                string :website = "https://github.com/jkcm/UW-particle-phase-classifier" ;
                string :References = "random forest relies on output from the UIOPS 2DS
processing suite, DOI:10.5281/zenodo.3667054" ;
                string :authors = "Rachel Atlas, Joe Finlon, Ian Hsiao, Jeremy Lu, Johannes
Mohrmann" ;
                string :_NCProperties = "version=2,h5netcdf=0.7.4,hdf5=1.10.5,h5py=2.10.0" ;
}
```

5.0 Data Remarks

A more complete assessment of the data set can be found in the companion paper.

Data from SOCRATES RF15 is not included, as the 2D-S raw data had quality issues that affected the UIOOPS processing and therefore the classification features.

The Github repository may be helpful for notebook examples on how to manipulate the data files.

Regarding the use of the trained model on either 2D-S data from a different campaign (e.g., in cirrus cloud), or on a different cloud particle imager, this is a complex question, and we encourage any researchers to contact the authors for a discussion on potential issues. The trained model is however openly available; used in conjunction with the UIOOPS feature extraction software, any user who wishes to apply this method to a similar data set is free to do so.

Regarding the differences between the bootstrapped and deterministic PSDs (specifically the variables `count_[dmax/darea]_[ice/liq]_ml_mean` and `count_[dmax/darea]_[ice/liq]_ml`), these are both attempting to capture the same variable, though only the bootstrapped PSD takes model confidence into account. In general these agree very well. The largest differences are when a data subsample contains highly unbalanced samples from each phase (i.e. much more of one phase than the other). Under these conditions, the absolute counts of the less commonly-occurring phase will be higher in the bootstrapped PSD than in the deterministic PSD, since necessarily some of the classifications in the more commonly-occurring phase with low model confidence will be randomly resampled to the less-commonly-occurring phase. This is evident on log-scale plots such as that found in Atlas et al. (2021; their figure 10). Our recommendation is that that the bootstrapped PSDs are the best estimate of the "true" PSDs, and should be used over the deterministic PSDs.

6.0 References

Atlas, R., Mohrmann, J., Finlon, J., Lu, J., Hsiao, I., Wood, R., and Diao, M.: The University of Washington Ice-Liquid Discriminator (UWILD) improves single particle phase classifications of hydrometeors within Southern Ocean clouds using machine learning, Atmos. Meas. Tech. Discuss. [preprint], https://doi.org/10.5194/amt-2021-123, in review, 2021.

Mohrmann, J., Finlon, J. A., Lu, J., Hsiao, I., and Atlas, R.: UW Ice Liquid Discriminator (UWILD) cloud particle classification software. Version 1.0, https://doi.org/10.5281/zenodo.4732532, https://doi.org/10.5281/zenodo.4732532, 2021.

McFarquhar, G. M., Finlon, J. A., Stechman, D. M., Wu, W., Jackson, R. C., and Freer, M.: University of Illinois/Oklahoma Optical Array Probe (OAP) Processing Software, https://doi.org/10.5281/zenodo.1285969, https://doi.org/10.5281/zenodo.1285969, 2018.

McFarquhar, G. M., Bretherton, C., Marchand, R., Protat, A., DeMott, P. J., Alexander,S. P., Roberts, G. C., Twohy, C. H., Toohey, D., Siems, S., Huang, Y.,Wood, R., Rauber, R. M., Lasher-Trapp, S., Jensen, J., Stith, J., Mace, J., Um, J., Järvinen, E., Schnaiter, M., Gettelman, A., Sanchez, K. J., McCluskey, C. S., Russell, L. M., McCoy, I. L., Atlas, R., Bardeen, C. G., Moore, K. A., Hill, T. C. J., Humphries, R. S., Keywood, M. D., Ristovski, Z., Cravigan, L., Schofield, R., Fairall, C., Mallet, M. D., Kreidenweis, S. M., Rainwater, B., D'Alessandro, J.,Wang, Y.,Wu,W., Saliba, G., Levin, E. J. T., Ding, S., Lang, F., Truong, S. C.,Wolff, C., Haggerty, J., Harvey, M. J., Klekociuk, A., and McDonald, A.: Observations of clouds, aerosols, precipitation, and surface radiation over the Southern Ocean: An overview of CAPRICORN, MARCUS, MICRE and SOCRATES, Bulletin of the American Meteorological Society, pp. 1–92, https://doi.org/10.1175/BAMS-D-20-0132.1, https://journals.ametsoc.org/view/journals/bams/aop/BAMS-D-20-0132.1/BAMS-D-20-0132.1.xml, 2021.

EOL: Low Rate (LRT - 1 sps) Navigation, State Parameter, and Microphysics Flight-Level Data. Version 1.3, https://doi.org/10.5065/D6M32TM9, https://data.eol.ucar.edu/dataset/552.002, 2019.