III. Model documentation and write-up

Information included in this section may be shared publicly with challenge results. You can respond to these questions in an e-mail or as an attached file. Please number your responses.

1. Who are you (mini-bio) and what do you do professionally? If you are on a team, please complete this block for each member of the team.

Dr. Patrick Broxton is an Assistant Research Professor in the School of Natural Resources and the Environment at the University of Arizona

Dr. Xubin Zeng is the Agnese N. Haury Chair in Environment, Professor of Atmospheric Sciences, and Director of the Climate Dynamics and Hydrometeorolgy Collaborative at the University of Arizona.

Dr. Niu is an associate professor of hydrometeorology, land surface modeler, and associate editor of Water Resources Research.

Dr. Ali Behrangi is a University of Arizona Distinguished Scholar and Professor of Hydrology and Atmospheric Sciences.

2. What motivated you to compete in this challenge?

We have been researching snowpack for many years and are highly motivated to improve snowpack monitoring in the western US and elsewhere. Furthermore, we have, and continue to be interested in estimating Snow Water Equivalent (SWE) over large areas from sparse station data.

3. High level summary of your approach: what did you do and why?

Our codes use Multilinear Regression (MLR) models to predict SWE for grid cells across the western US based on the provided snow station data (Figure 1). Models are trained to predict either ground measurements of SWE (if there are enough measurements for a particular grid cell), or SWE data taken from a gridded SWE dataset called the University of Arizona (UA) SWE dataset (if they are not). In addition to the MLR models, there is also code to fill missing snow station predictor data and to perform some bias correction of the models if necessary. Ultimately, model predictions are made by averaging an ensemble of MLR models.

One of the most important aspects of our solution is how it fills missing data (by incorporating the UA SWE data during model training, as well as filling missing predictor data during both model training and model inference). During model training, we chose to incorporate UA SWE data because UA SWE does a good job estimating SWE across larger regions (i.e. making it useful to estimate how SWE at unknown locations should relate to SWE at the predictor stations). In addition, we developed an innovative approach to fill the missing snow station data based on data from other nearby snow stations. We tried a few different machine learning methods but ended up using MLR because it was stable and provided adequate results

over a wide range of conditions. We trained our algorithm to maximize its performance over the whole western U.S., rather than for individual regions.

4. Do you have any useful charts, graphs, or visualizations from the process?

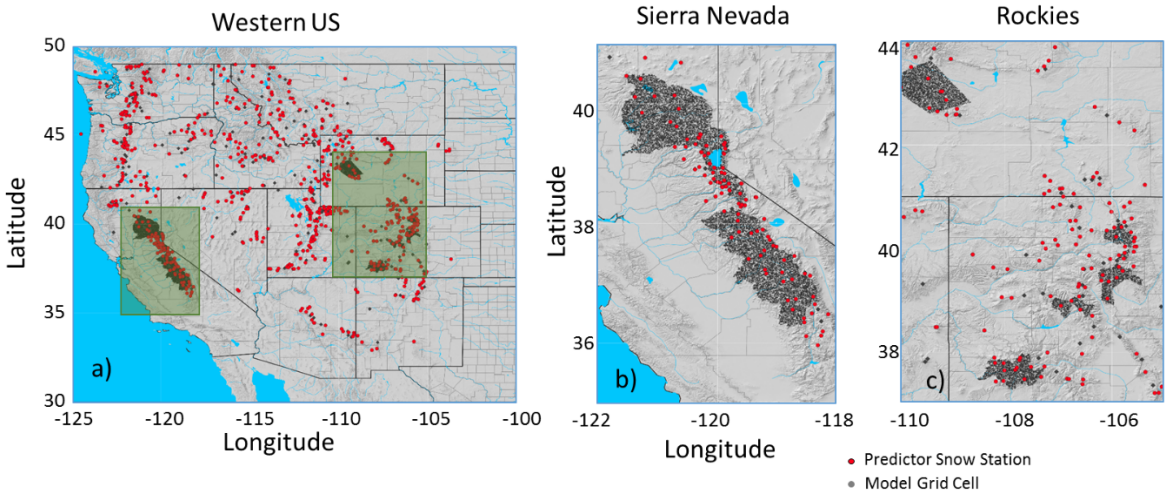

Figure 1: Locations of ground stations used as predictor variables (red dots), and grid cells where SWE is predicted (grey dots) for the a) Western US, b) Sierra Nevada, and c) Rockies.

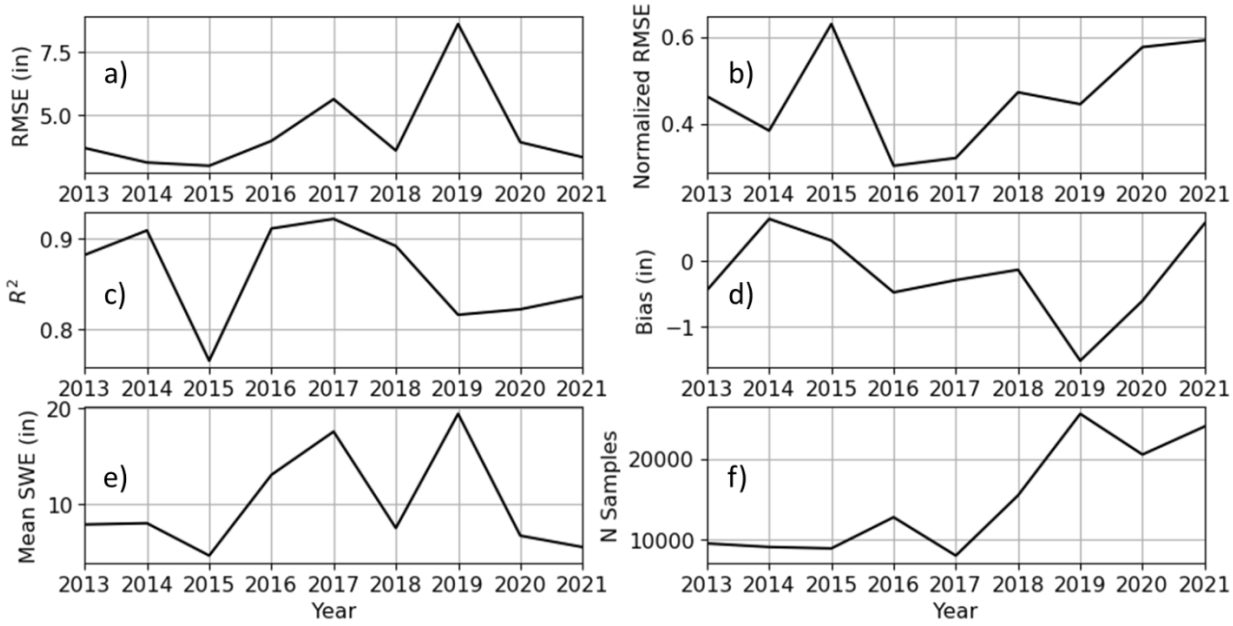

Figure 2: Yearly, cross-validated performance of the MLR models to predict SWE for validation grid cells. Panels a – d show performance in terms of RMSE, normalized RMSE, R^2, and bias (Modeled – Observed). Panel e shows the average observed SWE value for each grouping for each year, and panel f shows the number of contributing observations.

5. Copy and paste the 3 most impactful parts of your code and explain what each does and how it helped your model.

Code Snippet 1: Fill the missing predictor snow station data based on nearby snow stations (in TrainModels.py Lines 249-258 though there is also analogous code in RunModels.py Lines 119-126)

```
# Fill the missing ground station data in the training set
filled = np.empty(NDates_train)
for d in range(NDates_train):
    if np.nansum(weights * ~np.isnan(GroundSWE_train[locs, d])).sum() > 0:
        filled[d] = np.nansum((a * GroundSWE_train[locs, d]) * weights) / np.nansum(
            weights * ~np.isnan(GroundSWE_train[locs, d]))
    else:       # Unlikely case where we have 0 in the denominator
        if d > 0:
            filled[d] = filled[d-1]
        else:
            filled[d] = 0
```

The above code snippet creates a continuous estimate of predictor station data, based on relationships between SWE at a given station and SWE at surrounding stations, which is used in case the station has missing data. In the above code, the filled estimate is a weighted average of 'predictions' of SWE based on the relationship between SWE at the station and each of its neighbors. Each prediction is assigned a weight based on the correlation between SWE for each station pair, and only stations that are good predictors are considered. The weights automatically adjust in case any of the surrounding stations have missing data (and hence are not used: note the ~np.isnan(GroundSWE_train[locs, d])) part).

Code snippet 2: Train the MLR models (in TrainModels.py, Lines 326-334)

```
sorted_rsquares = np.sort(rsquares)[::-1]
idx = np.argsort(rsquares)[::-1]
locs = locs[idx[:max(MinStations, np.count_nonzero(sorted_rsquares >= MinRSquare))]]
x = np.transpose(GroundSWE_train_filled[locs, :])
x = x + np.random.rand(x.shape[0], x.shape[1]) * 1E-5

# Train the MLR Model and apply to the training data
nanlocs = np.isnan(t)
mdl = LinearRegression().fit(x[~nanlocs, :], t[~nanlocs])
```

This is the code to train the multilinear regression models for a given grid cell based on SWE data from predictor stations. However, for each model, data is used only from predictor stations that demonstrate strong predictive skill (i.e. high R^2) between the station and the target grid cell (though each model must include more than a minimum number of stations). The first part of the code selects the predictor stations that have $R^2 >$ MinRSquare, or a minimum of MinStations training locations that have the highest R^2 with the target grid cell, and the second part trains the MLR model. Parameter values (MinRSquare and MinStations) are adjusted to maximize the overall model performance.

Code snippet 3: Run the MLR models (in RunModels.py, Lines 147-152)

```
y = Models[i]['mdl'].predict(x)
bias = Models[i]['bias']
y = np.transpose(y) + bias['c']
y[y < 0] = 0
AllSWE_ML[i, :] = bias['a'] * y ** bias['b']
```

This is the code to run the MLR models, and to apply any required bias correction factor for a given grid cell. The first line uses the MLR to make a prediction, and the subsequent lines apply the bias correction factor (which is of the form a * (x + c) ^ b), where a, b, and c are determined by analyzing the relationship between the MLR SWE predictions and SWE observations in the training dataset.

6. Please provide the machine specs and time you used to run your model.
   - CPU (model): **Intel Core i7-8550U @1.80GHz, 4 Cores**
   - GPU (model or N/A): **N/A**
   - Memory (GB): **8 GB**
   - OS: **Windows 10 / Windows 11**
   - Train duration: **~10 minutes per ensemble, ~100 minutes for the whole ensemble**
   - Inference duration: **~30-90 seconds, depending on the # of elements that are calculated**

7. Anything we should watch out for or be aware of in using your model (e.g. code quirks, memory requirements, numerical stability issues, etc.)?

   Overall, our model is very efficient and has low data requirements, using only the provided snow station data to make predictions. This makes it simple and robust to run. However, its reliance on snow station data alone can cause it to produce unexpected results if the station data itself is spurious. Indeed, there were a few instances noted both historically and during the competition when station data changed unexpectedly, which caused SWE estimates for some grid cells to also change unexpectedly. Future work could improve the ability of the model to identify and discard suspect observations on the fly.

8. Did you use any tools for data preparation or exploratory data analysis that aren't listed in your code submission?

   We did some quality checks of the ground station data to identify obvious spurious data during past years. In addition, codes were developed to extract the UA SWE data, as well as a few other remotely sensed SWE products, to the model grid cells for the training dataset. Prior to extraction, we also downscaled the UA SWE data (which is provided at 4 km resolution from https://nsidc.org/data/nsidc-0719) to 1 km to better match the 1km resolution used for this competition (see https://github.com/broxtopd/UASWE for a code to downscale the data).

9. How did you evaluate performance of the model other than the provided metric, if at all?

   Prior to 2022, we evaluated cross validated model performance for different years (see Figure 2) and different months. In addition to RMSE, we also evaluated model performance using R^2

(coefficient of determination), Normalized RMSE (where RMSE is normalized by the mean value), and overall model Bias.

10. What are some other things you tried that didn't necessarily make it into the final workflow (quick overview)?

    In addition to the approach that we used, we tried to use dynamic modelling as well as other snow datasets to help fill in the training dataset. In addition, we tried some other machine learning approaches (random forest models, and multilayer perceptron neural networks) to predict SWE.

11. If you were to continue working on this problem for the next year, what methods or techniques might you try in order to build on your work so far? Are there other fields or features you felt would have been very helpful to have?

    In the future, we could incorporate more variables to improve our models. For example, we could incorporate satellite data, which is especially useful for detecting if an area is snow covered or not, or in some cases, to directly estimate snow depth. Furthermore, it is important to better understand under what conditions our technique works well (e.g. in the mountains, close to regular snow monitoring stations) and where other methodologies might work better (e.g. in lowland areas, far from these stations).