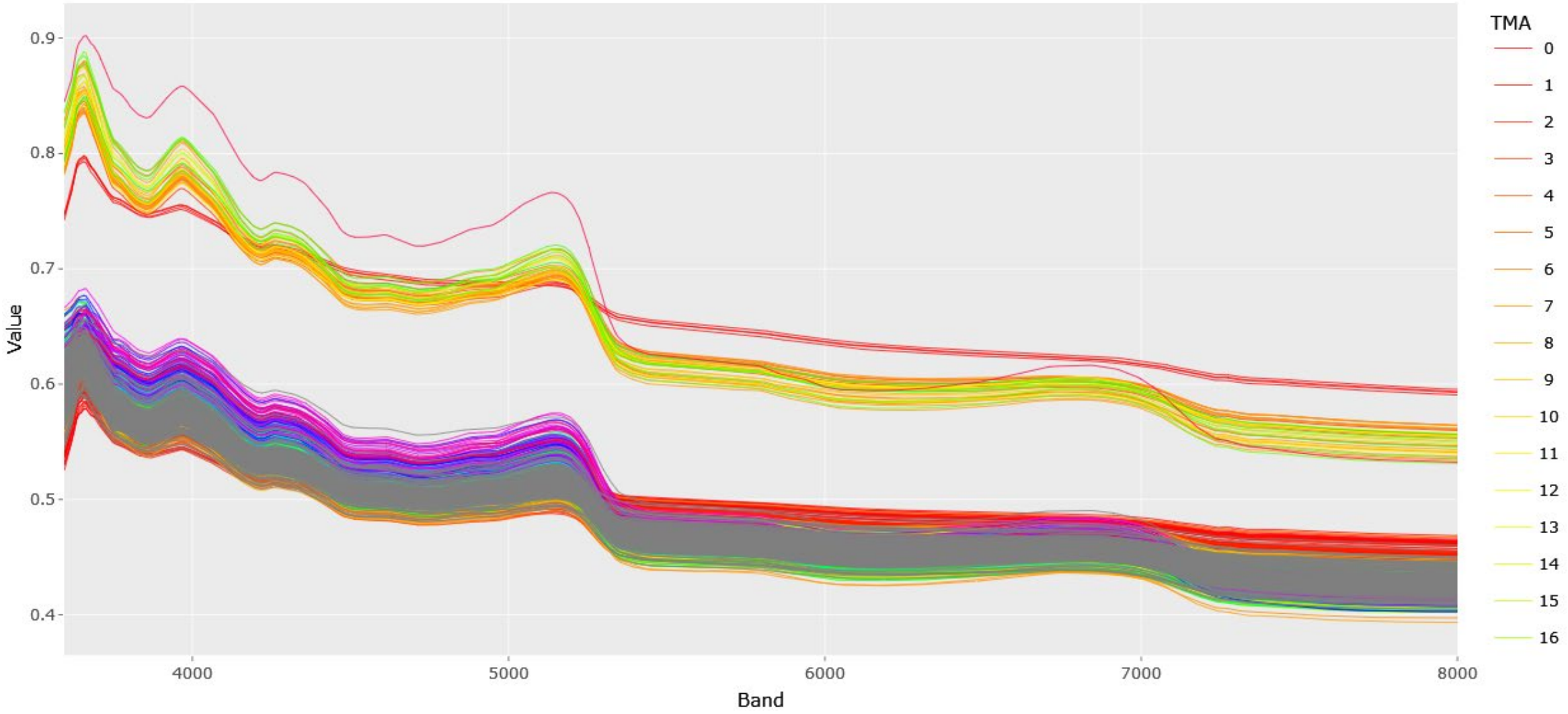
Exploring the use of FT-NIRS for ageing sablefish (Anoplomona fimbria) and Pacific hake (Merluccius Productus) off the U.S. West Coast

John R. Wallace

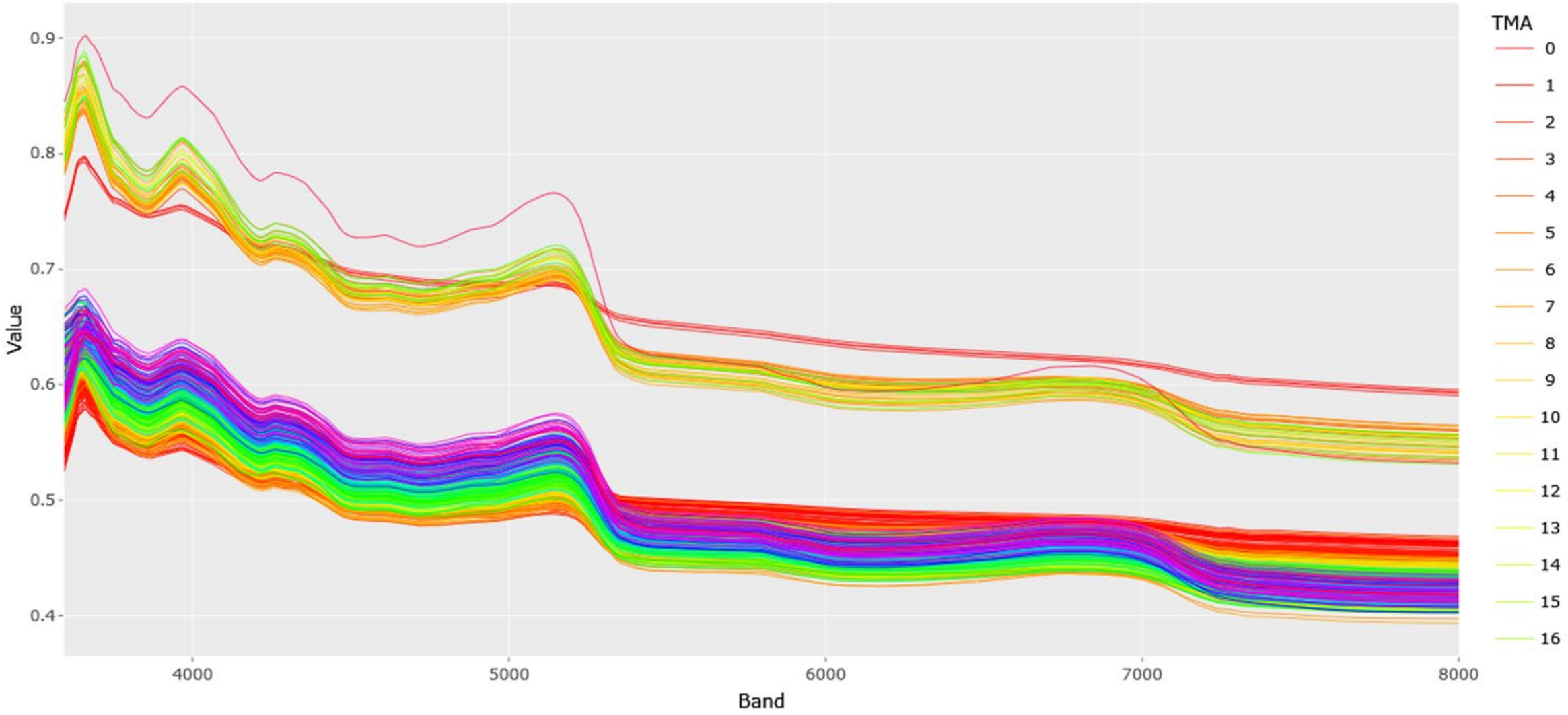
NWFSC - Seattle

Sablefish

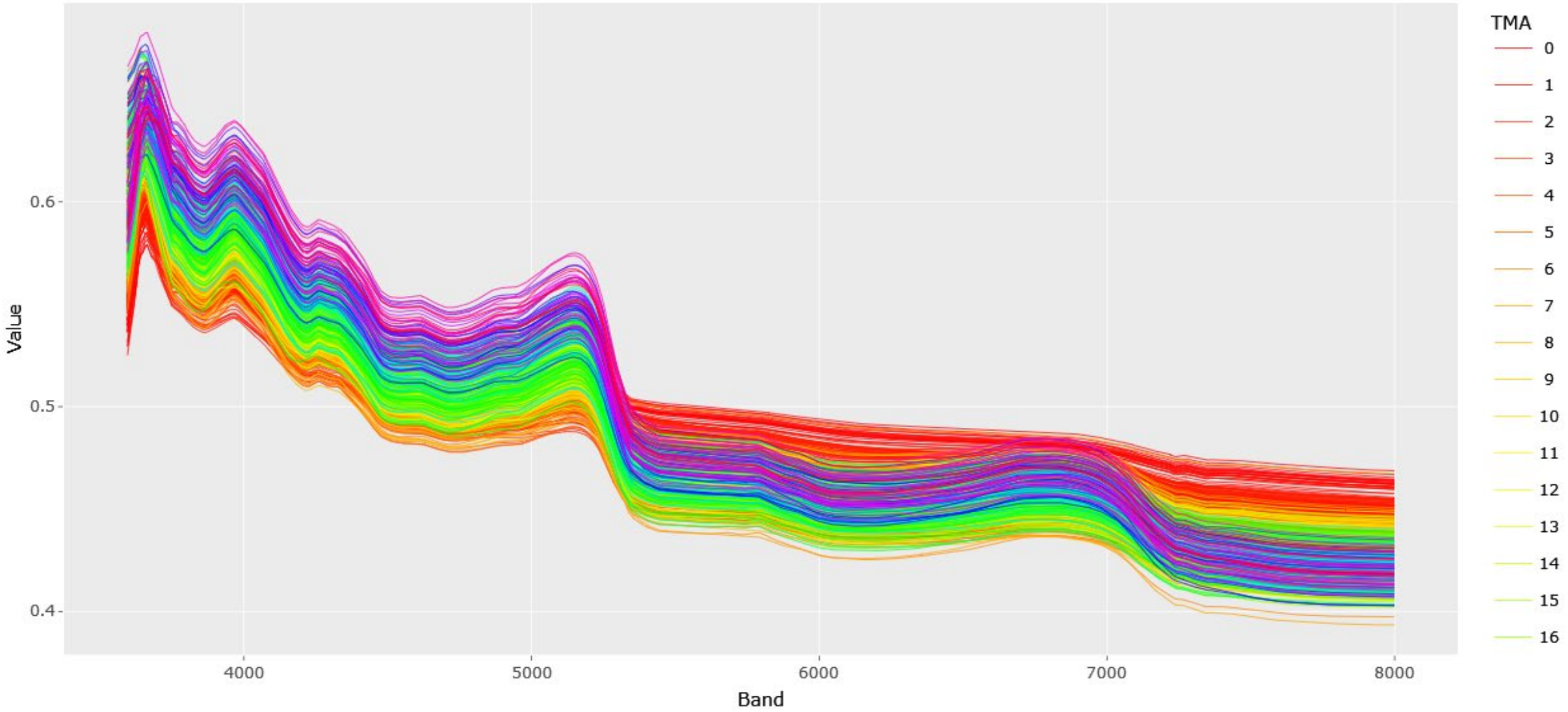
Raw Spectra with Missing TMA in Grey



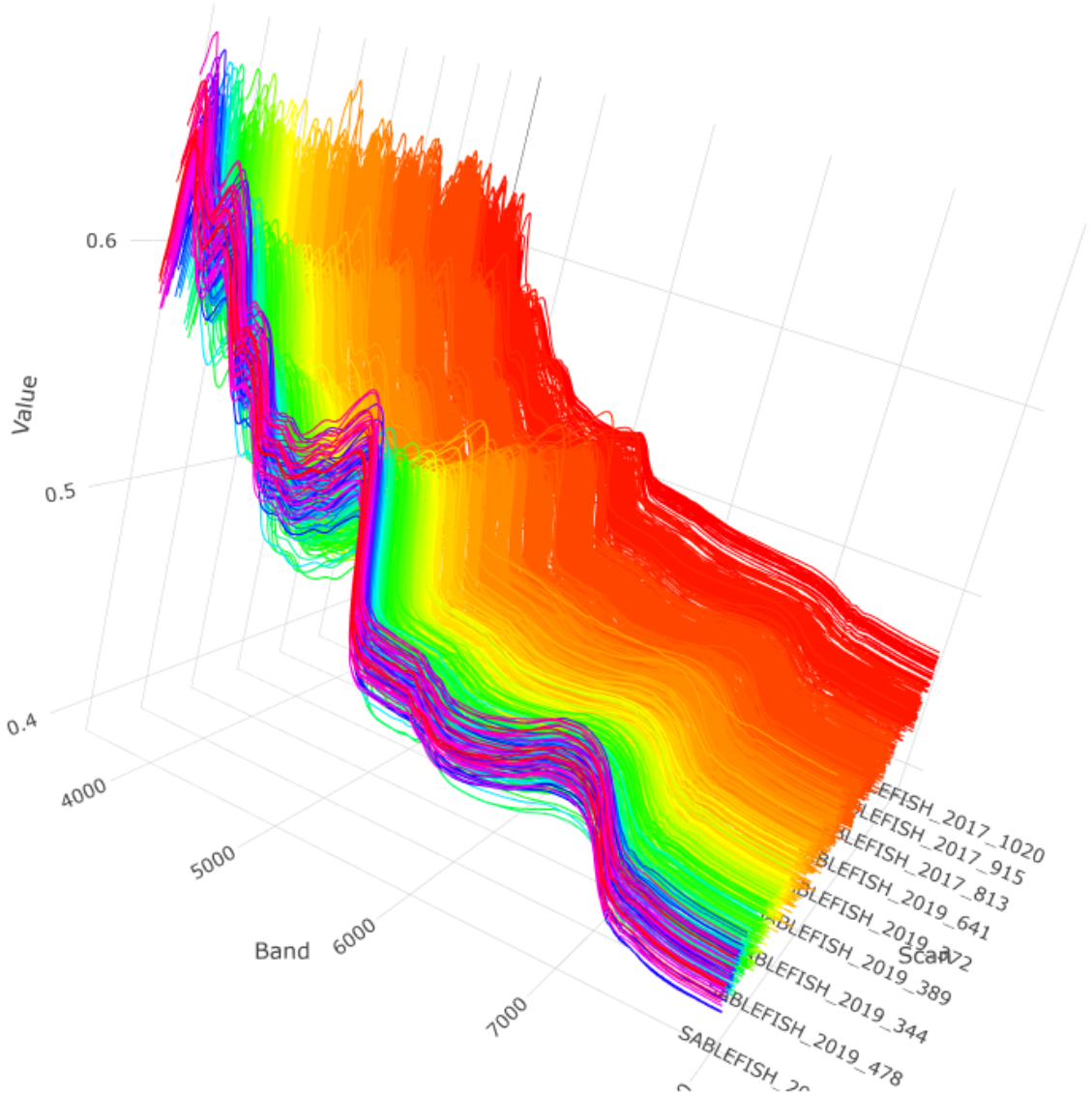
Raw Spectra with Missing TMA Removed



Extreme Spectra Removed



3D View


Value
0.6
0.5
0.4
4000
5000
6000
7000
Band
Scan
SABLEFISH_2017_1020
SABLEFISH_2017_915
SABLEFISH_2017_813
SABLEFISH_2019_641
SABLEFISH_2019_389
SABLEFISH_2019_344
SABLEFISH_2019_478

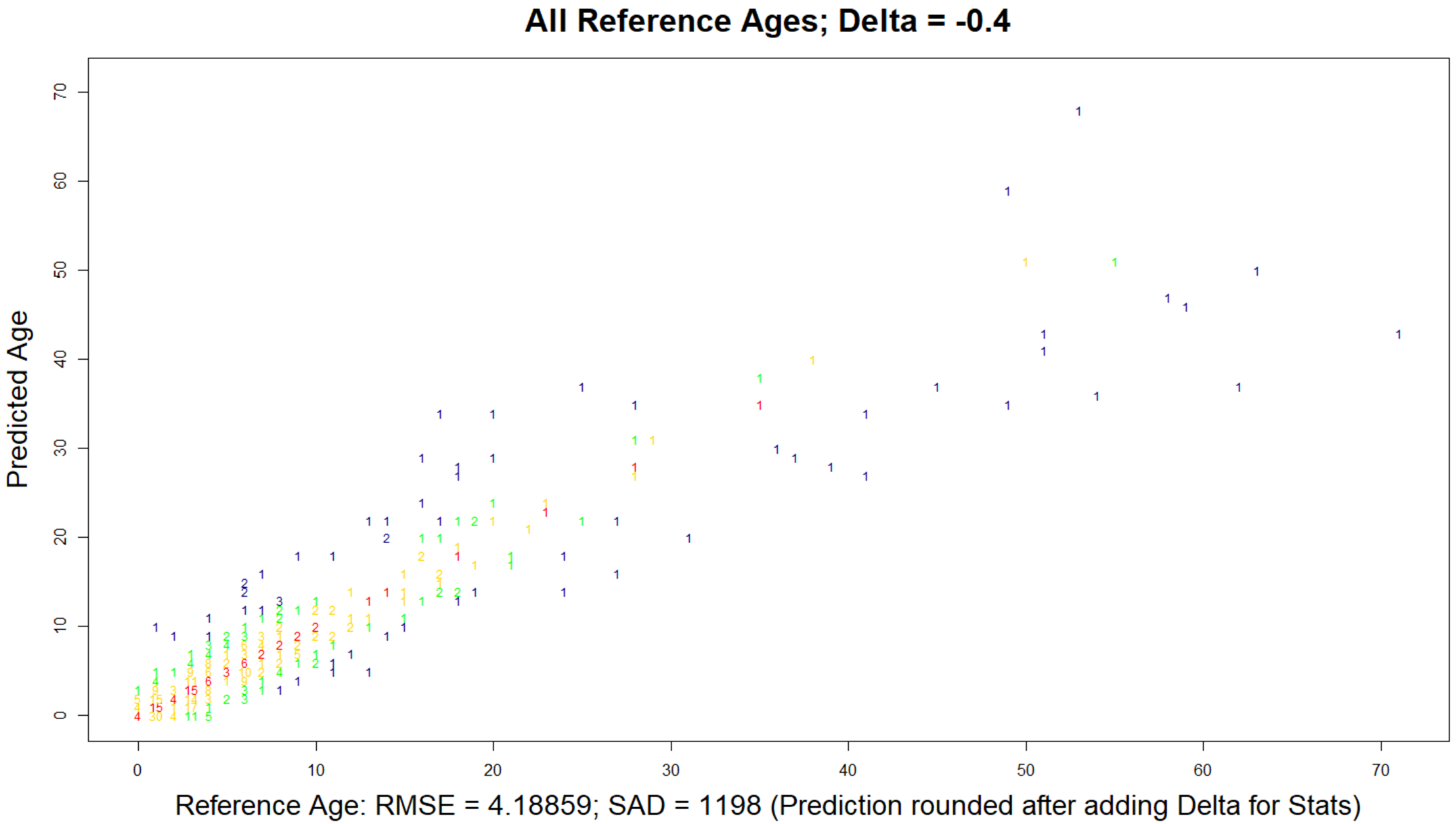
Prelude to the Roadmap

- Validation levels, from deep in the code to higher
  - In the NN model, on the epoch level, there is an optional, user selected validation level. (I used 20%.)
  - On the one pass level there is the main data split into the training set and the test set. (I used a 2/3, 1/3 split.)
  - On the k-fold level, a 10-fold model format was used. One tenth of the data was left untouched for testing the 2/3, 1/3 model from the remaining 90% of the data.

Roadmap

- Raw Data
- PLS model (2/3, 1/3 split)
- Neural Net (NN) Model (same 2/3, 1/3 split) compared to PLS
- NN models with the k-folding
- Medians taken over X number of complete k-folds, each with a different pseudo random number generation start point.

Sablefish, PLS Model Predicted onto Test Data (1/3 of the Total)



Moved to NN models

- It didn’t appear that the PLS models were that good, but I wasn’t sure if it was the PLS model limitations or the Sablefish otoliths were difficult to get good results from.

- Moved on to Neural Net models
  - Other packages on R were not very useful until I found the ‘keras’ R package that sits on top of Google’s TensorFlow software.

Software Issues

- Google's TensorFlow (Python sitting on top of C++)
- Keras NN modeling software
- Full Keras package in R (few advanced examples found on the net)
- Currently there are various version conflicts (was better a few years ago)
  - Hard to get to work in WSL (Window's Subsystem for Linux)
  - Very hard to get working in Native Windows (but it is faster, then WSL).
- All sits on top of the Nvidia's CUDA (Compute Unified Device Architecture) software which makes a graphics card into a GPU (Graphics Processing Unit)

Hardware issues

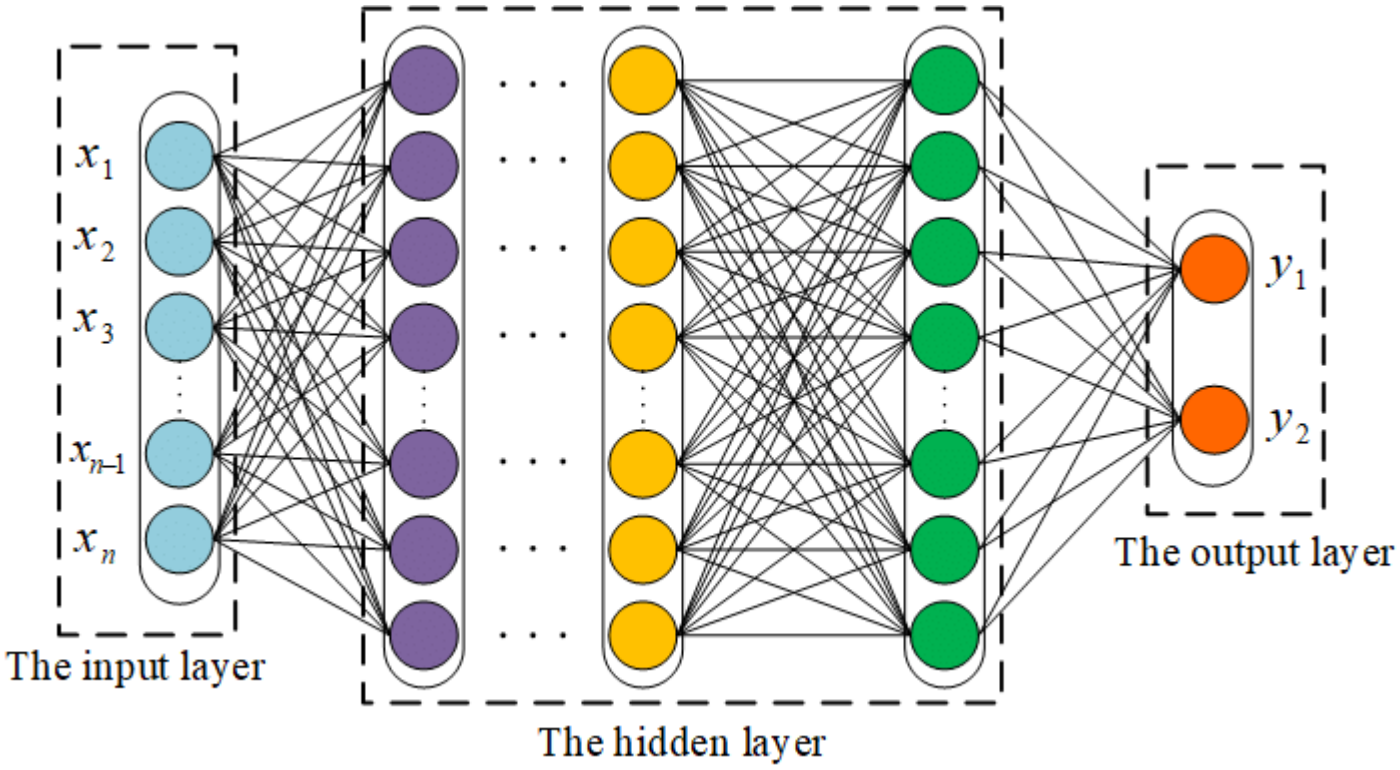
Once the software works, you need a NN Model

- FCNN is a fully connected model, where every node in the neural net is connected to all the other nodes.

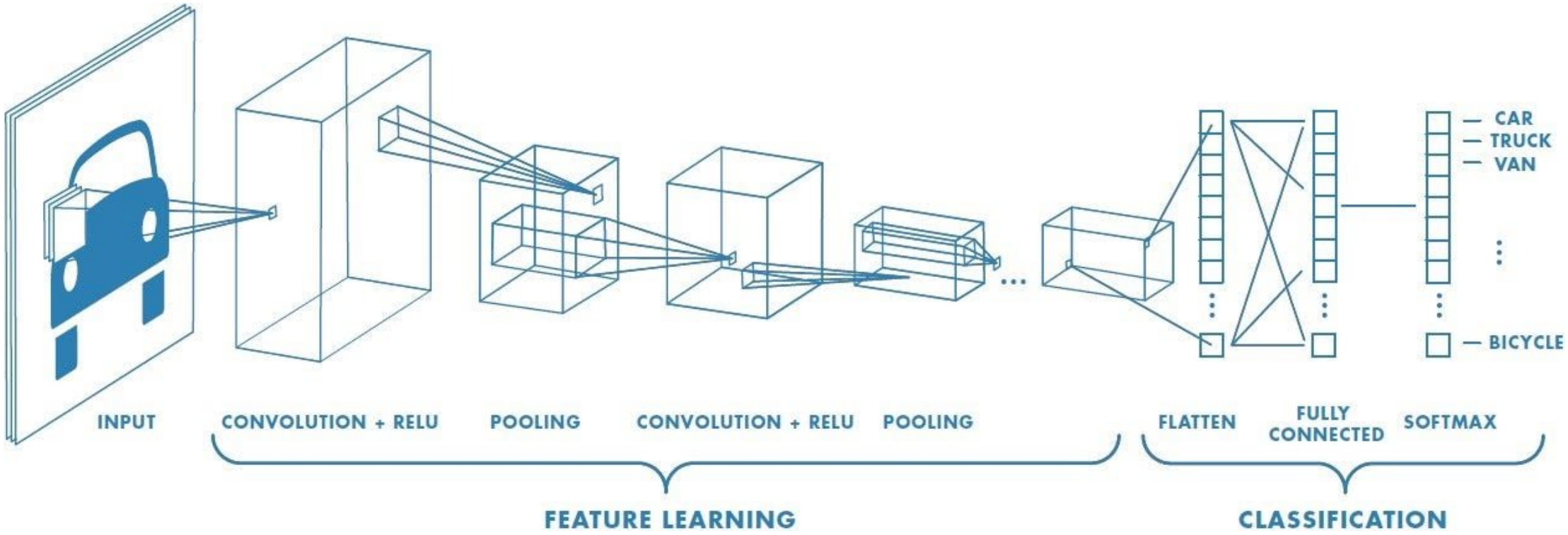
- Other more complex, models did not perform as well.
  - Many of the more complex models are fully or partially convoluted with many hidden layers
  - Convoluted NN is where not all nodes are connected (more work to be done).

- *Use of Artificial Neural Networks and NIR Spectroscopy for Non-Destructive Grape Texture Prediction. Basile et al. Foods 2022, 11, 281.*
  - “We found that increasing the number of hidden layers resulted in a worsening of the prediction of our parameters.”

Fully Connected Neural Net (FCNN) Model



Convolutional Neural Network (2D example)



- In the training of a neural network, a common practice is to normalize the input
- data (mean close to 0). Normalized data generally increase the learning rate and lead to
- faster convergence. A min-max normalization was applied to scale the input variables in
- the interval [0,1].

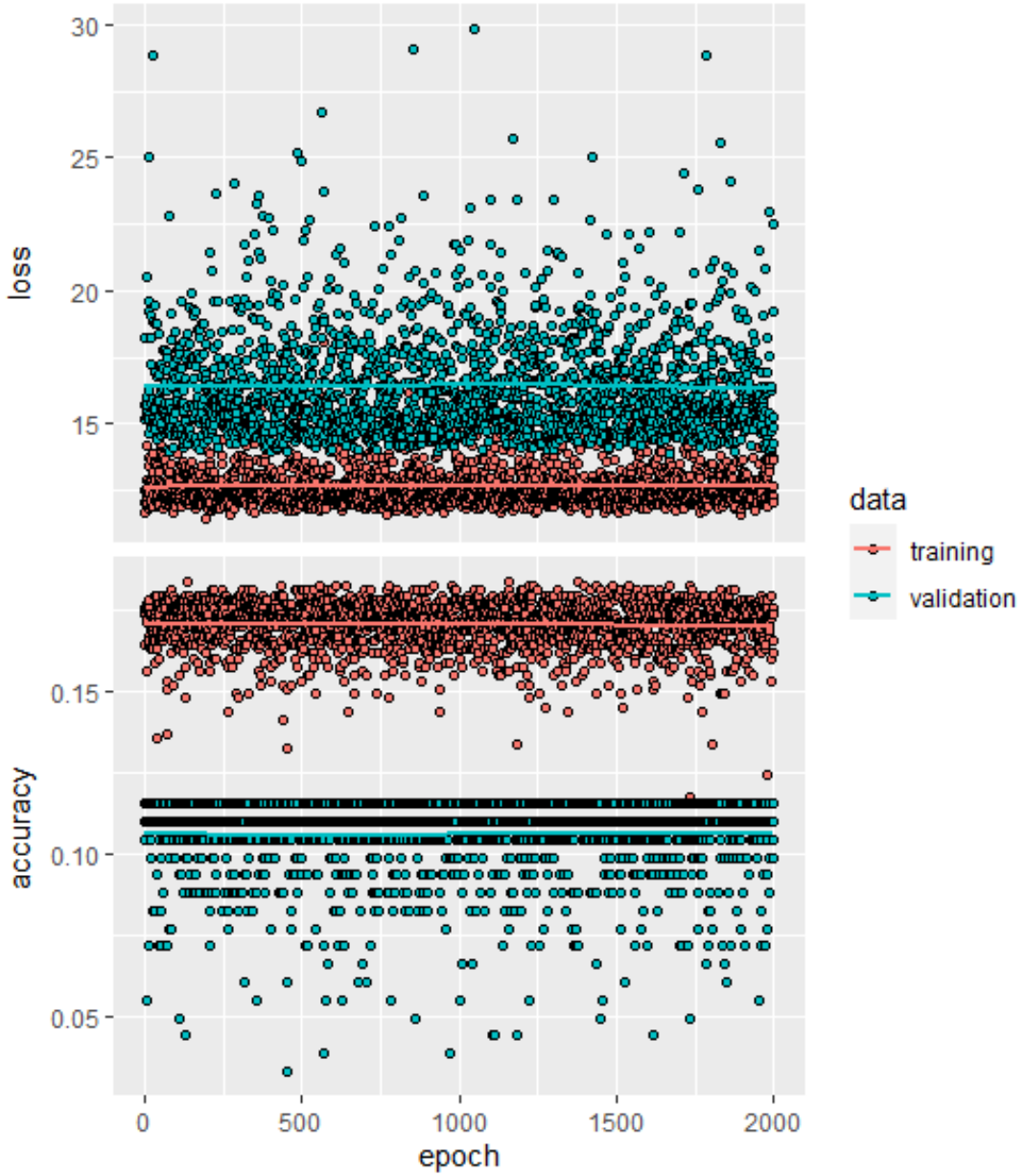
Neural Net Modeling Basics

- Normalize the input. Getting the mean close to 0, and not doing min-max normalization, worked best for the iPLS input.
- That data needs to be split into training and testing sets.
- To match the PLS, initially the same 2/3 of the data was used for training and 1/3 for testing.
- A neural net ‘epoch’ means training the neural network with all the training data for one cycle.
- In an epoch, all of the data is used exactly once. A forward pass and a backward pass together are counted as one pass.

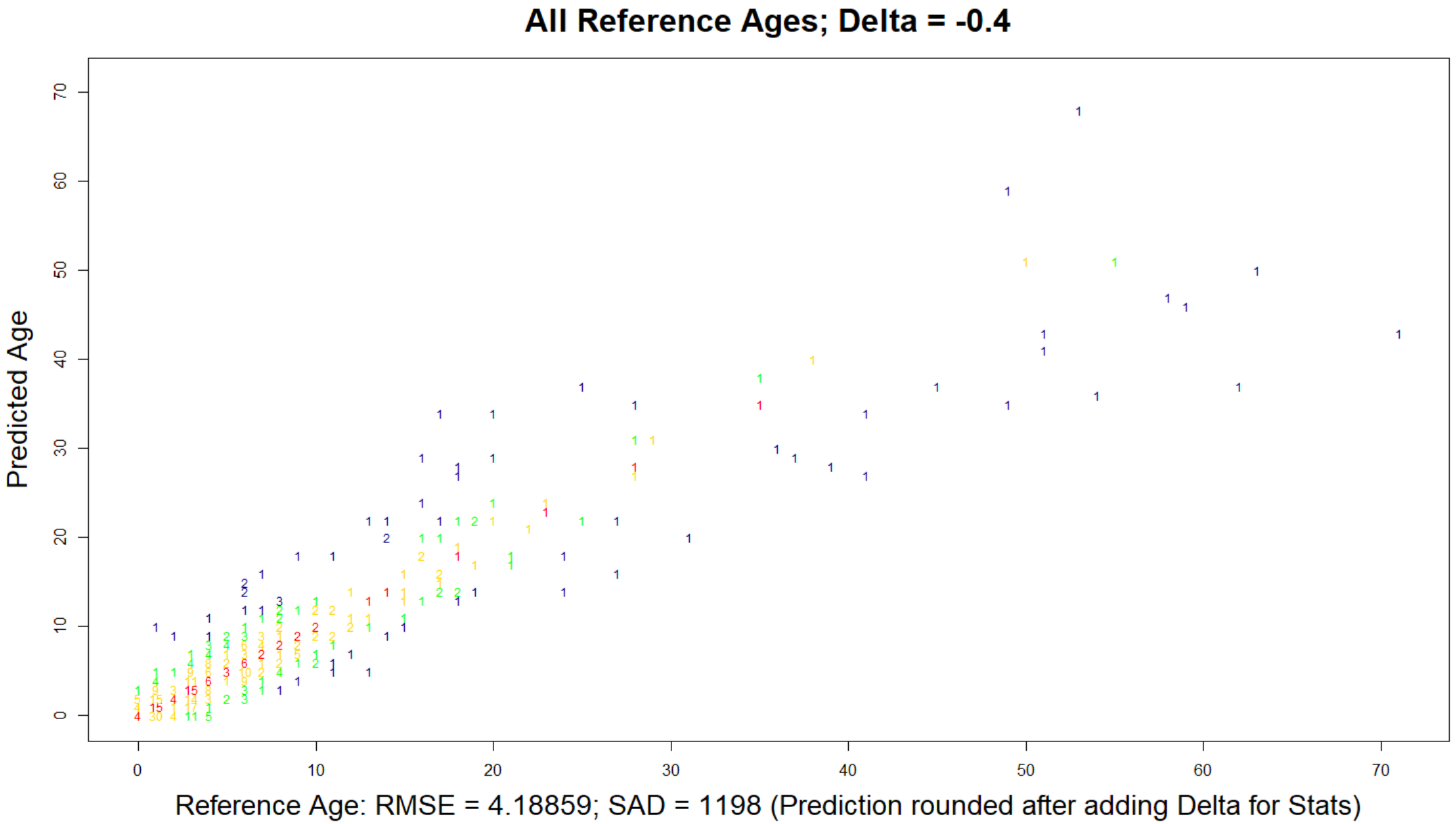
Neural Net Modeling Basics (cont.)

- An epoch is made up of one or more batches. Batch size is the number of samples to work through before updating the internal model parameters.
- At the end of the batch, the predictions are compared to the expected output variables and an error is calculated. From this error, the update algorithm is used to improve the model, e.g. move down along the error gradient.
- The training was structured into 8 iterations 500 epochs each, with testing against the 1/3 test data done at the end on each iteration to view progress.
- Batch sizes of 32 and a validation split of 0.2 (80% of the data was used to train and 20% to test the model) was used.

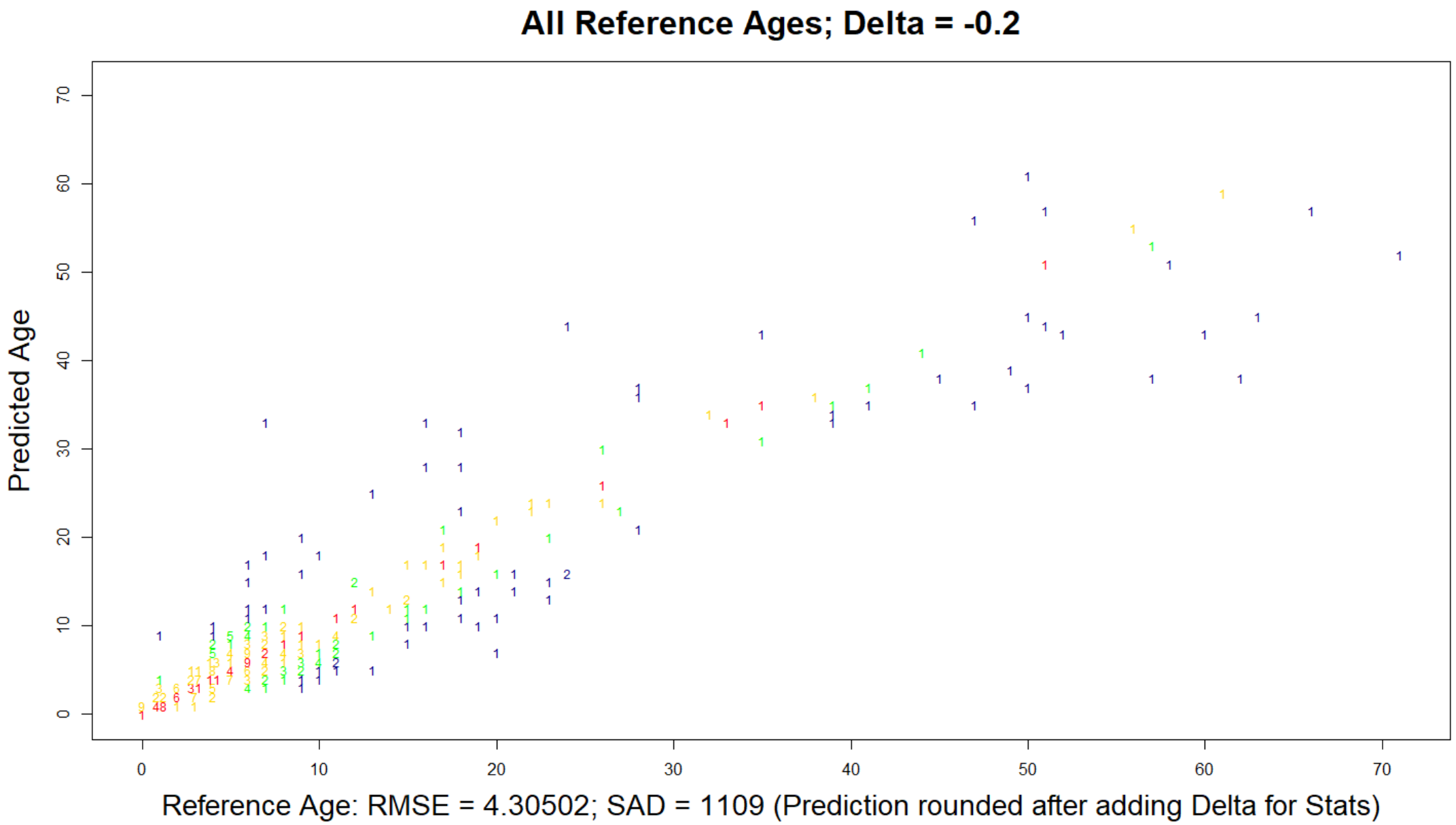
Loss and Accuracy for Training and Validation over the Epochs



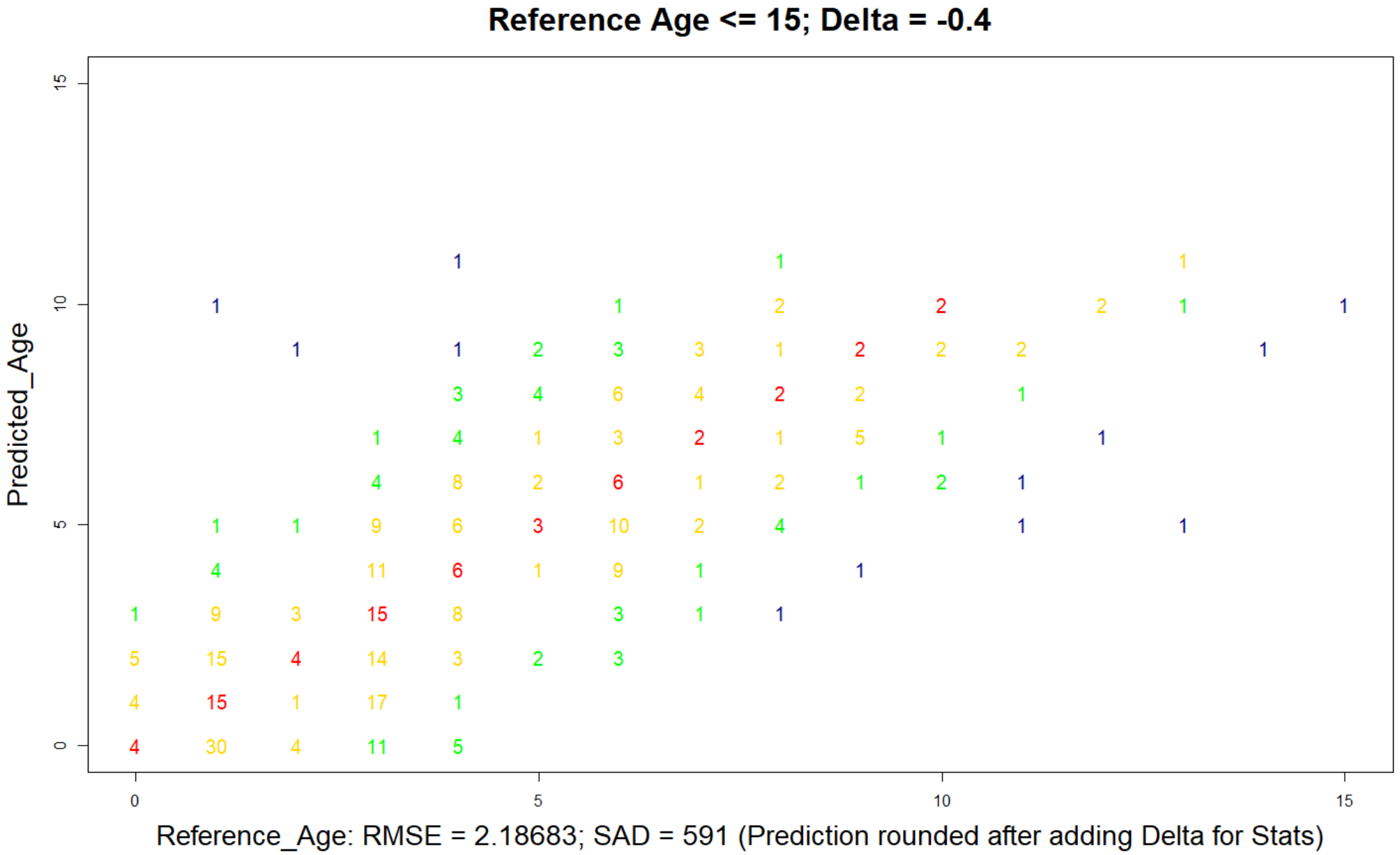
Sablefish, PLS Model Predicted onto Test Data (1/3 of the Total)



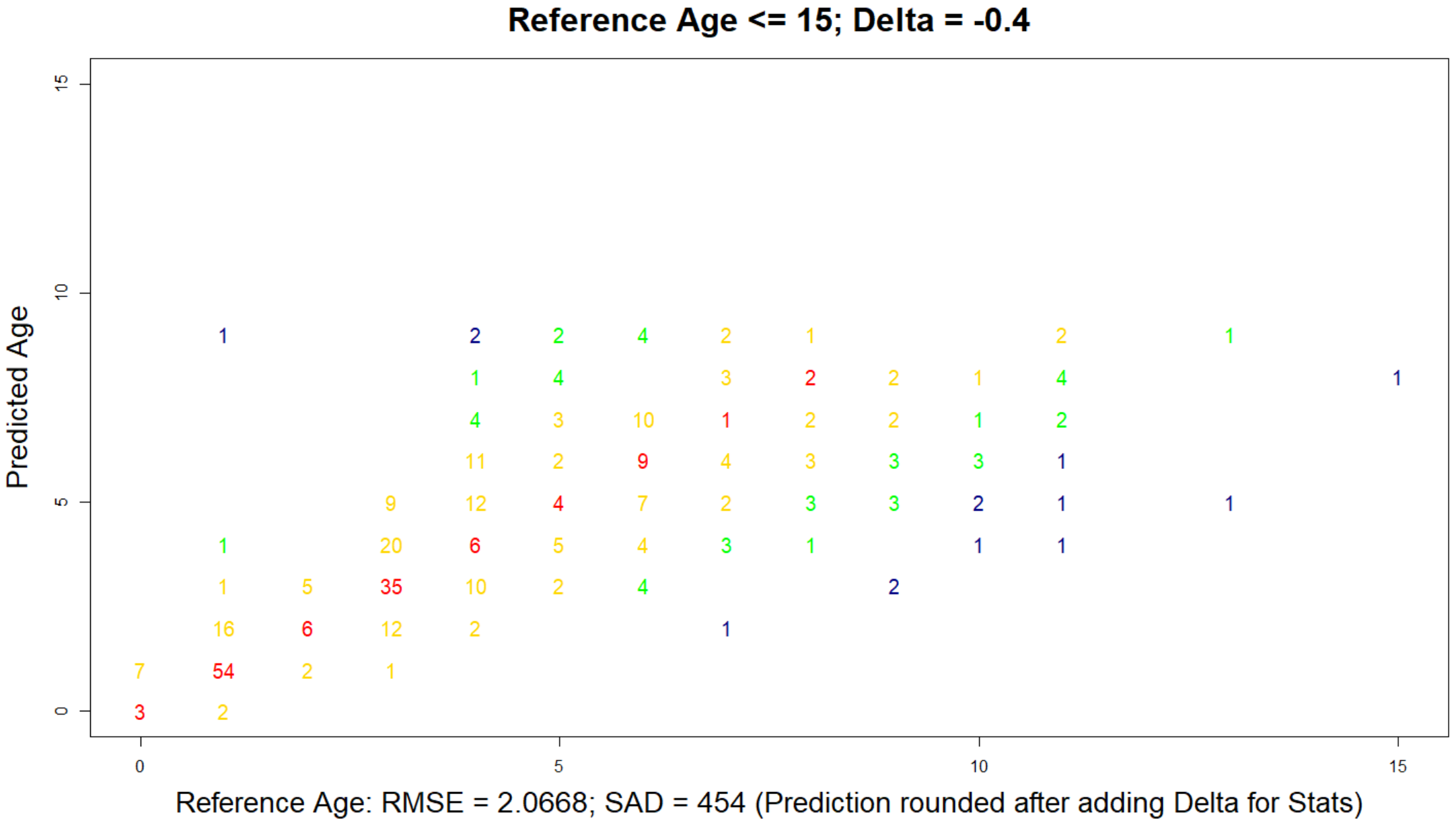
Sablefish, FCNN Model Predicted onto Test Data (1/3 of the Total)



Sablefish, PLS Predicted onto the Test Data, Ref Age <=15

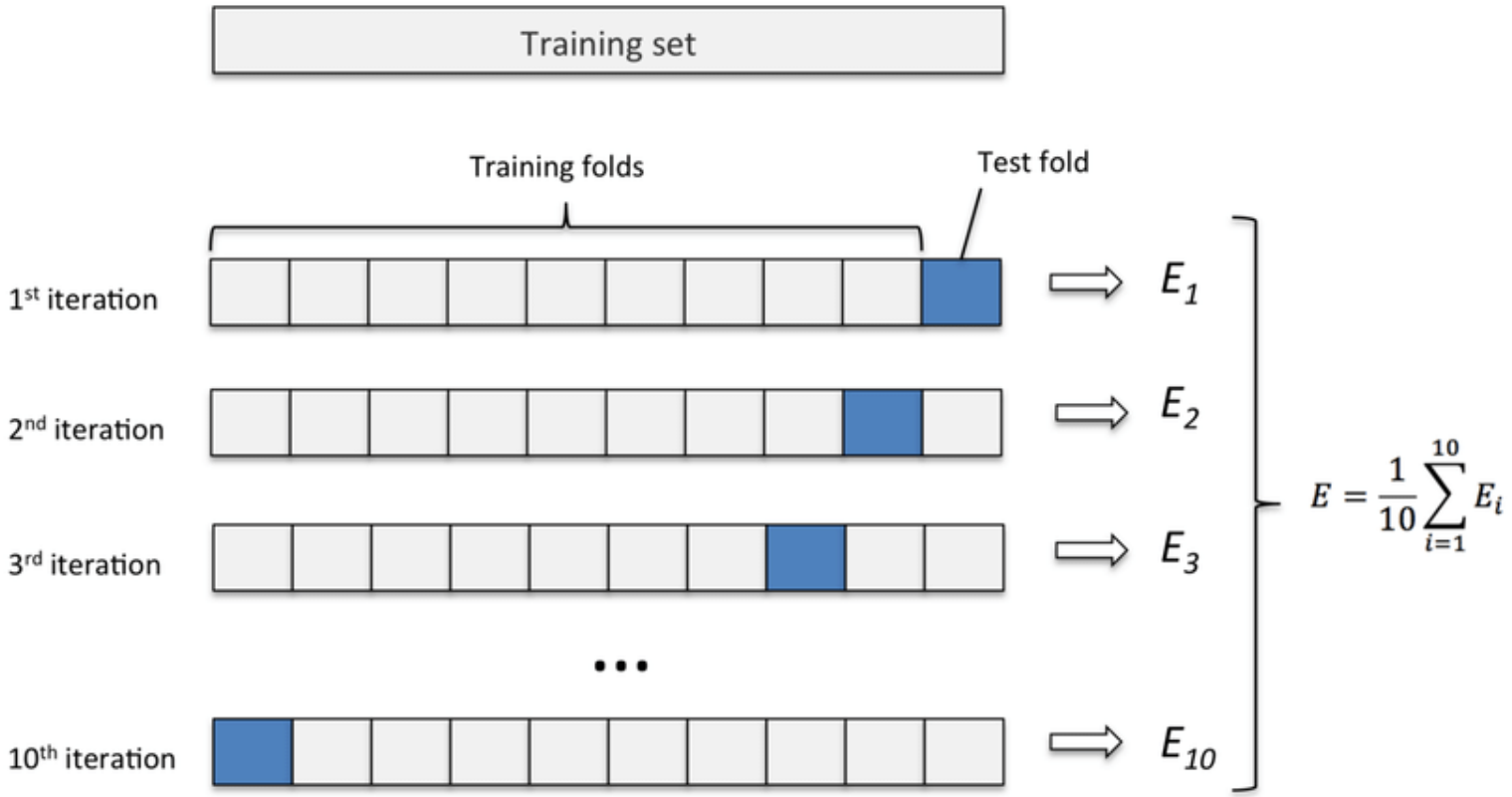


Sablefish, FCNN Predicted onto the Test Data, Ref Age <=15



Next Step: K-fold Modeling

- A 10 k-fold model format was used.
- One tenth of the data was left untouched for testing a model from the remaining 90% of the data.



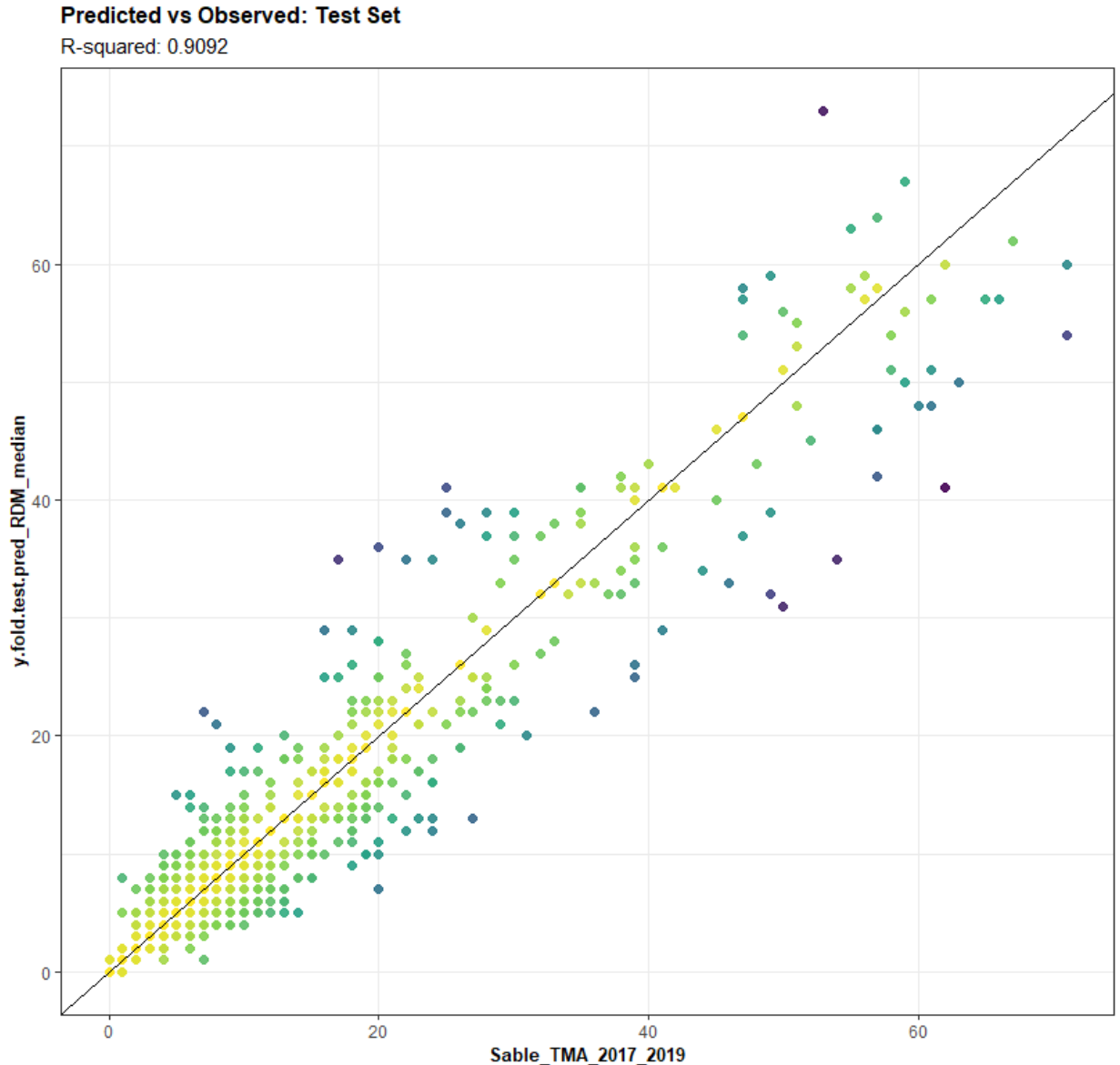
K-fold Modeling (cont.)

- Of that 90%, 2/3 was used for training and 1/3 for testing that particular model (as before).
- To train the sub-model, 500 neural net epochs were run on the training set and then tested against the test set.
- Eight such iterations, of 500 epochs each, were performed with (hopefully) model improvements at each step.
- As before, a validation split of 20% and a batch size of 32 was used.
- However, eventually the model performs worse (almost always by the 8th iteration or 4,000 epochs for this FCNN model on the Sablefish and Hake data), and hence the best fitting iteration is used for the current fold:

K-fold Modeling (cont.)

- After all 10% folds are set aside in turn, a complete fold set is finished, and each predicted point was never inside a model that predicted it.

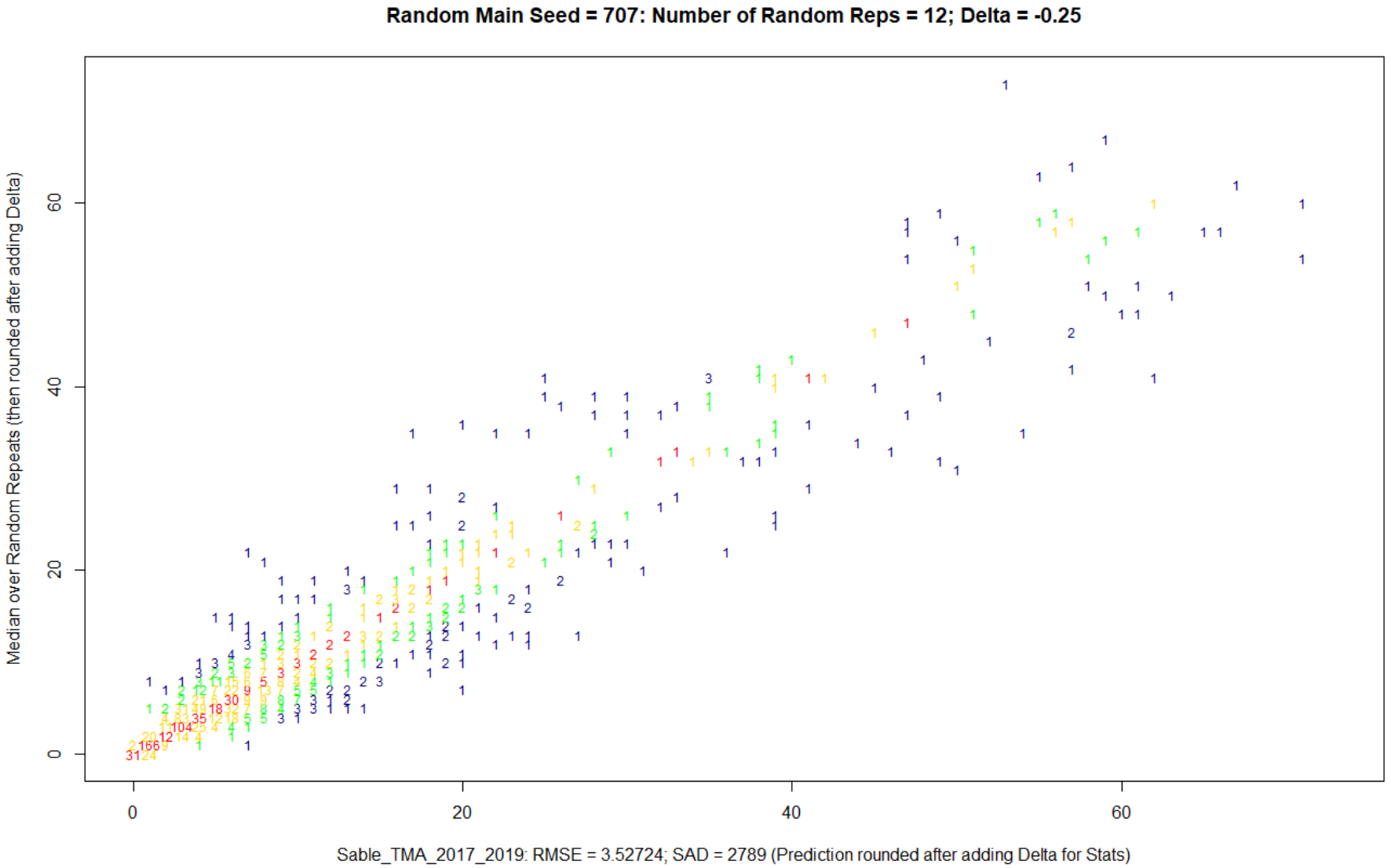
- Twenty complete k-fold models were run, each with a different pseudo random number seed, controlled by a main seed.


Predicted vs Observed: Test Set
R-squared: 0.9092
y.fold.test.pred_RDM_median
Sable_TMA_2017_2019
0
20
40
60

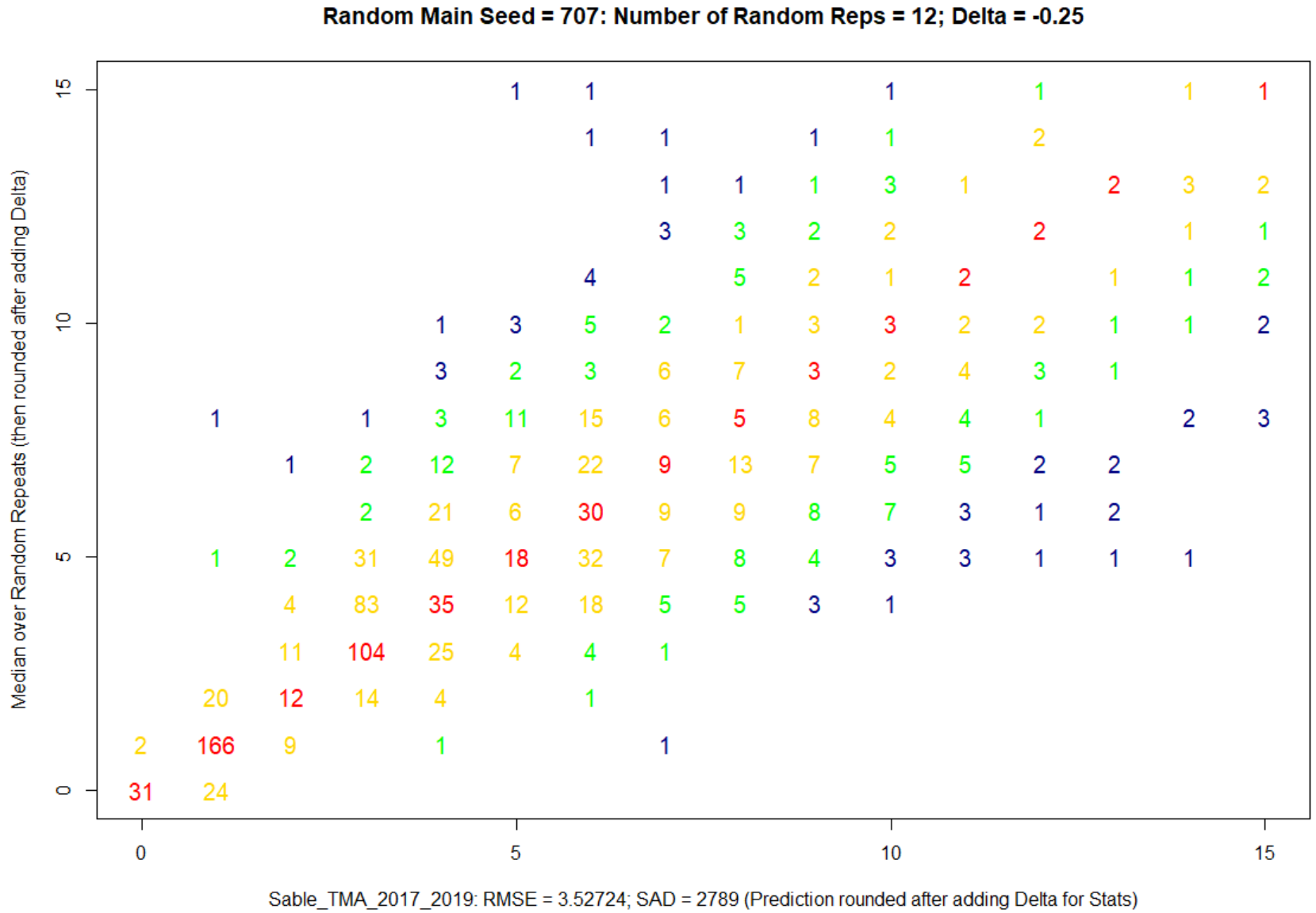
In the next Customized Agreement Figures

- SAD = Sum of the Absolute Deviations
  - In the figure, this equals the sum of the correctly matched otoliths (zeros, in red); plus 1 for each estimated age off by one year from the TMA, plus 2 for each estimated age off by two years from the TMA, etc.
- RMSE = Square Root of the Mean Squared Error
- Delta = the amount added to $\mathbb{R}$ estimated age before rounding to an integer. (Delta is a negative number.)
- TMA = Traditional Method of Aging (Break and burn here)

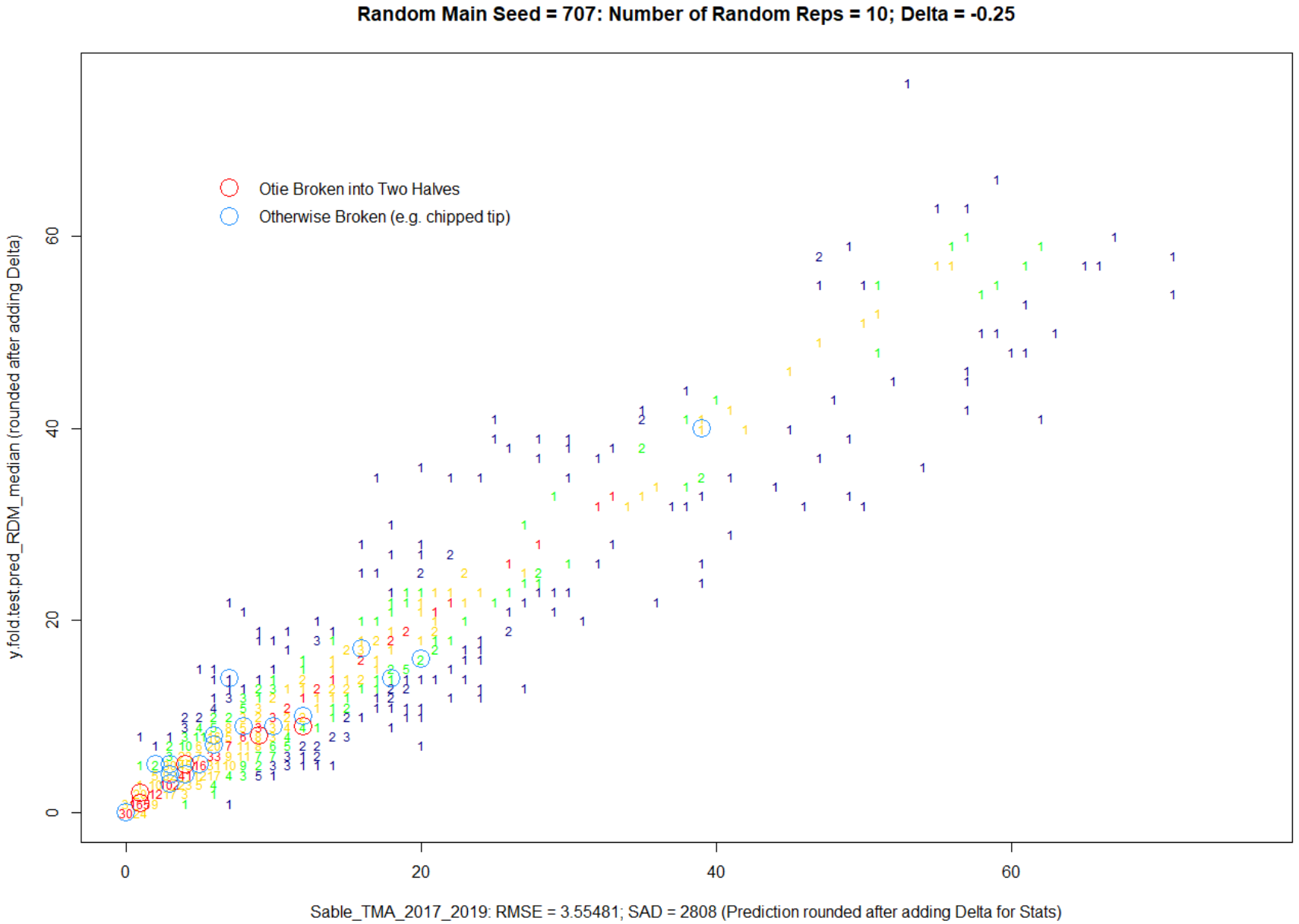
Median over 12 k-fold models vs TMA



Median over 12 k-fold models vs TMA (zoomed)



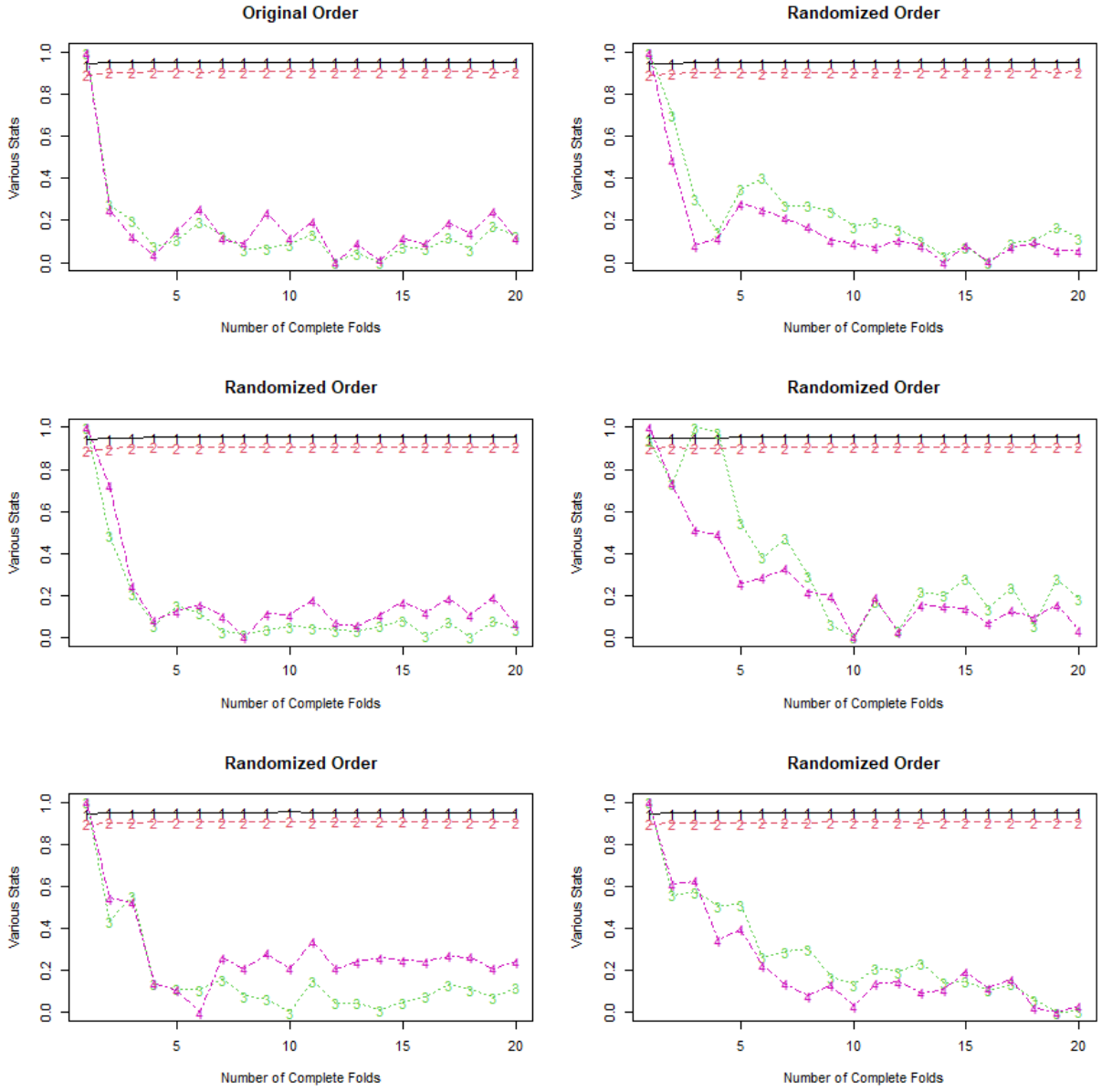
Broken Oties Intermixed with Unbroken Ones



Lastly

- Medians over predicted ages where taken for each additional k-fold model added at each step.

Randomized Additions of a Full k-Fold

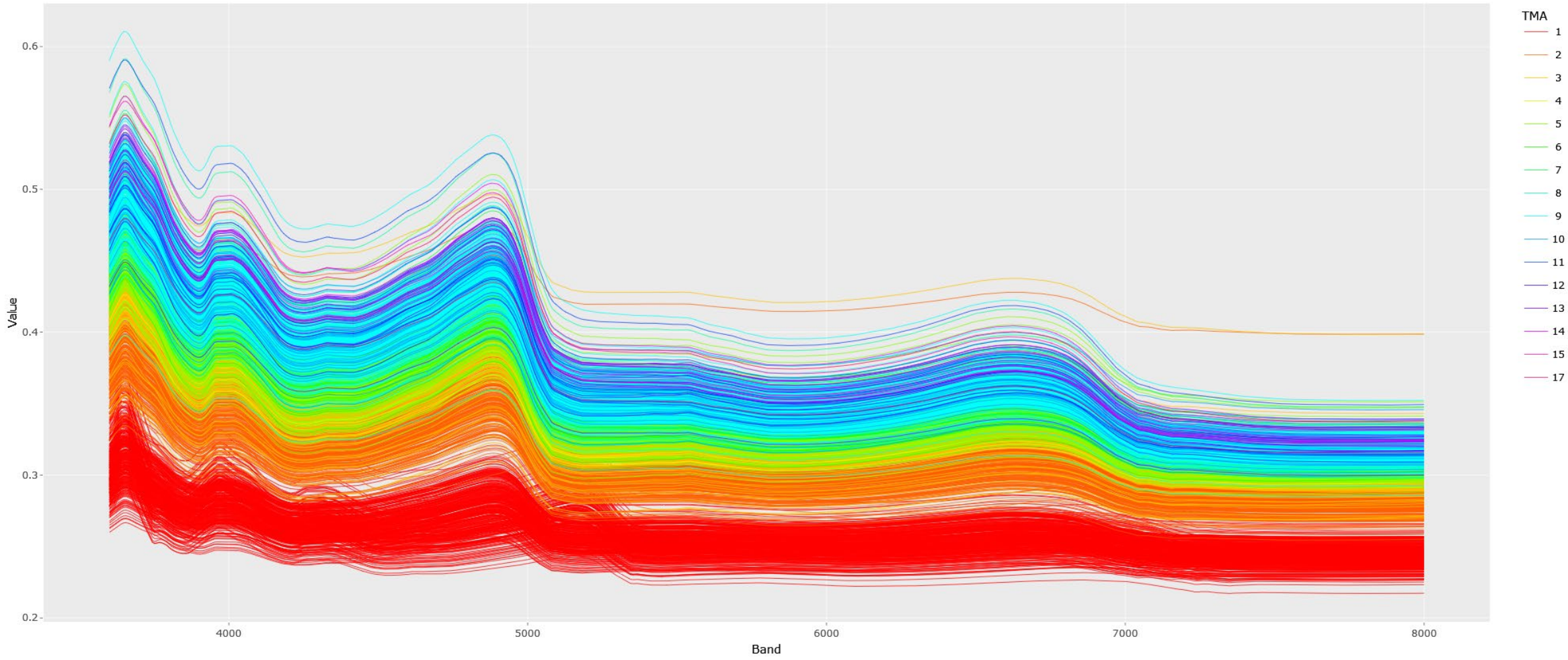


1: Correlation: Black
2: R squared: Red
3: RMSE: Green
4: SAD: Purple

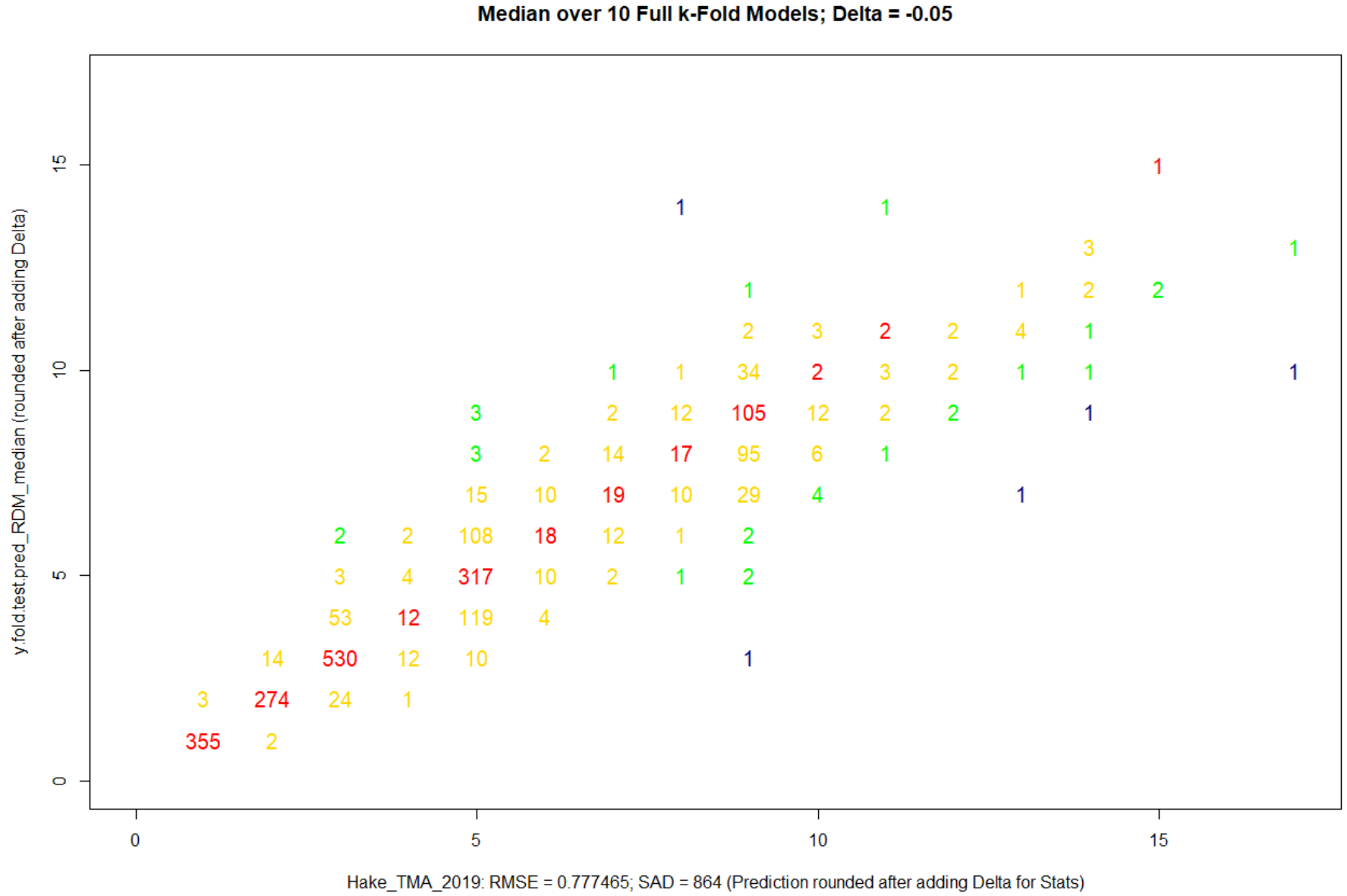
(In the original run order, the 12th model addition had the best stats.)

Pacific Hake

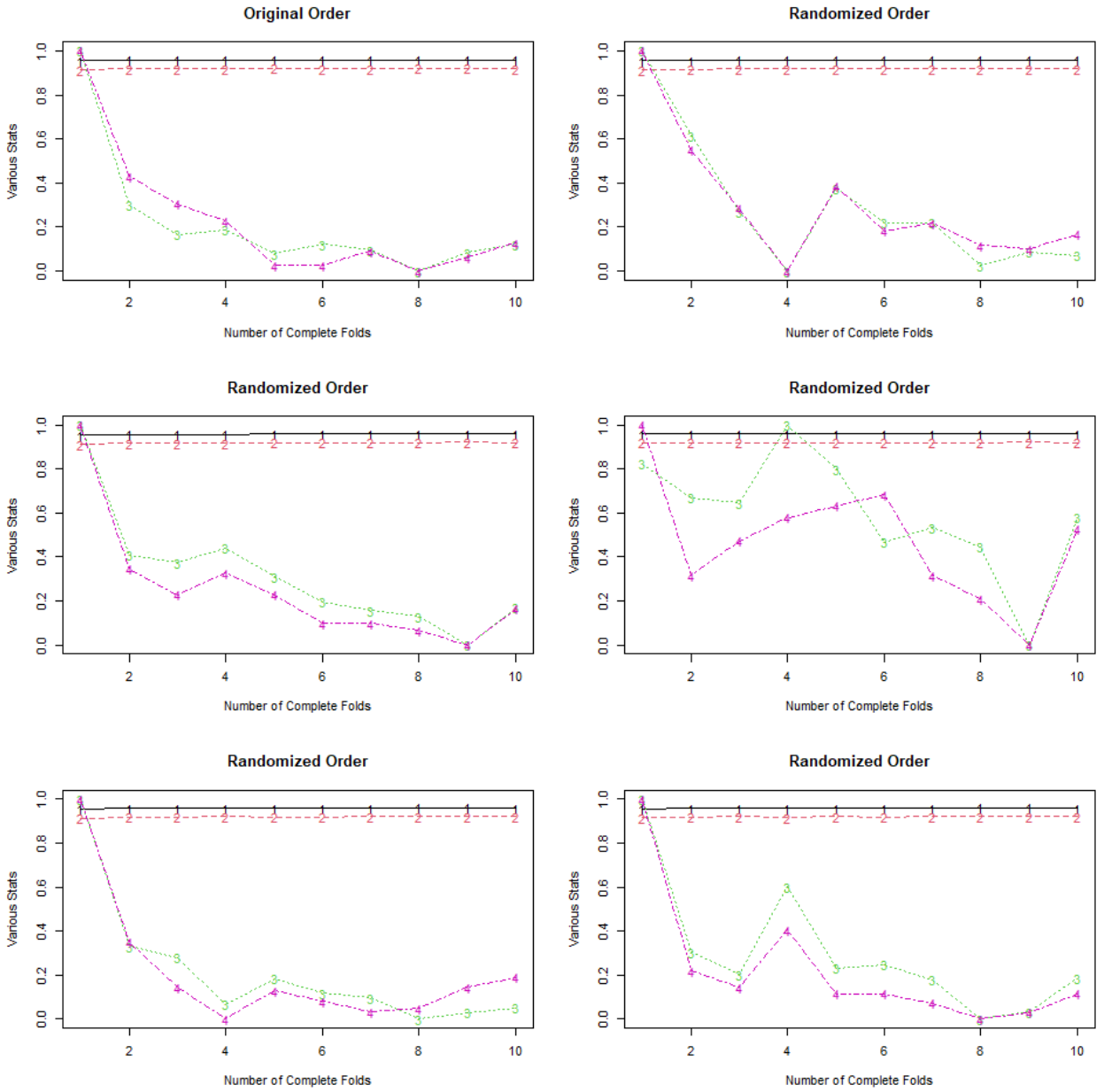
Hake, Raw Spectra



Median over 10 Full k-Fold Models



Hake, Randomized Additions of a Full k-Fold



1: Correlation: Black
2: R squared: Red
3: RMSE: Green
4: SAD: Purple

(In the original run order, the 9th model addition had the best stats.)

Future Direction

- Ensemble models
  - PLS, NN, and other models are used in an ensemble approach

Interactive URL's for plotly Figures

- https://soundbirds.github.io/Hake_Spectra_plotly/
- https://soundbirds.github.io/Hake_Spectra_plotly_3D/

An aside:

- Nvidia initially had no name and the co-founders named all their files NV, as in "next version". The need to incorporate the company prompted the co-founders to review all words with those two letters, leading them to "invidia", the Latin word for "envy".

- More NN references?