Overview of FT-NIRS data and neural network modeling: as developed using U.S. West Coast sablefish (Anoplomona fimbria) as a case study

John R. Wallace

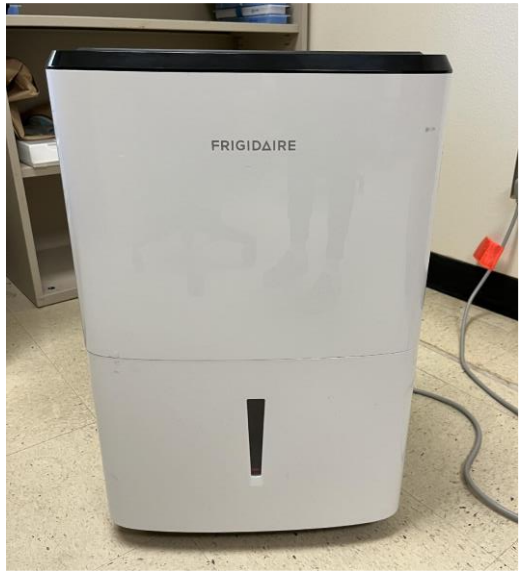
Emily Wallingford

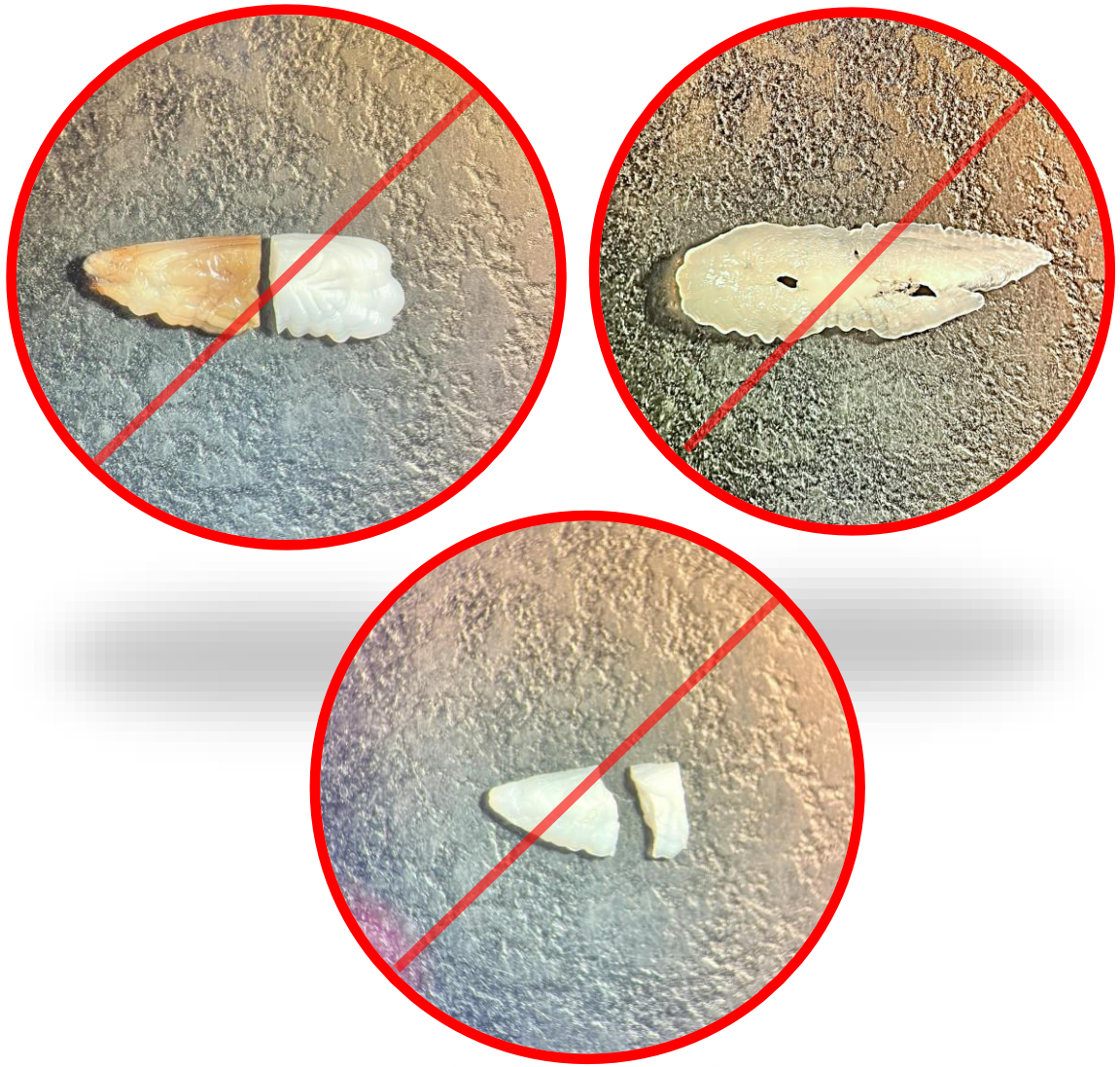
NWFSC - Seattle

Sablefish

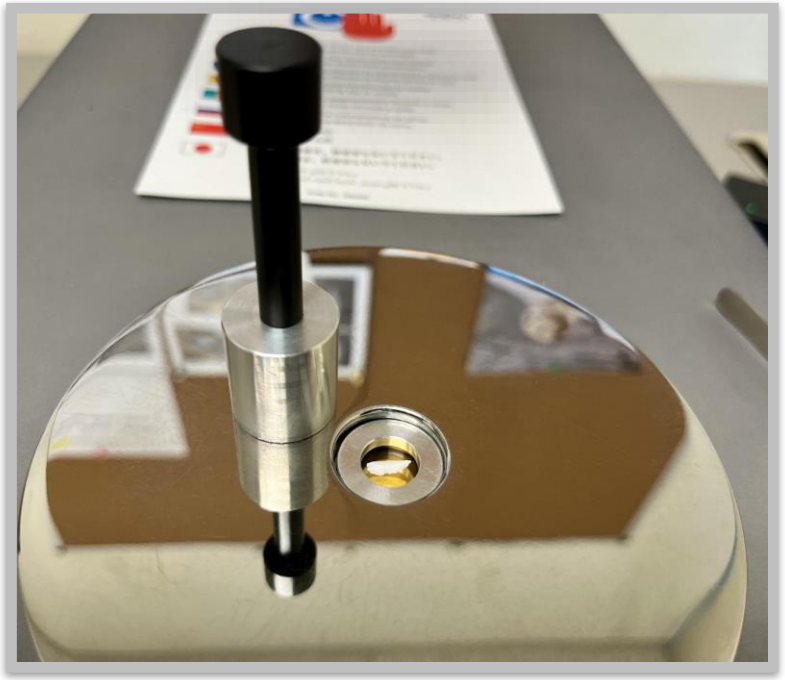
Credit: NOAA Fisheries

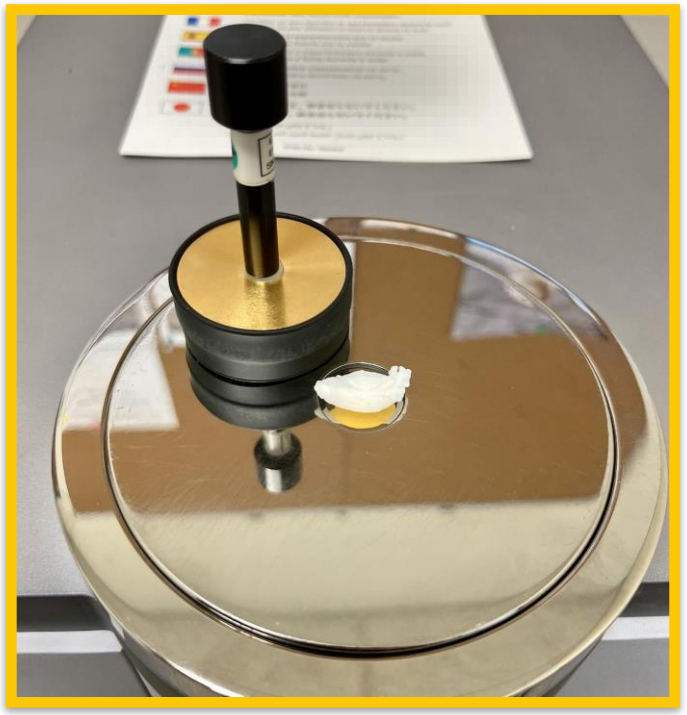
SCANNING METHODS



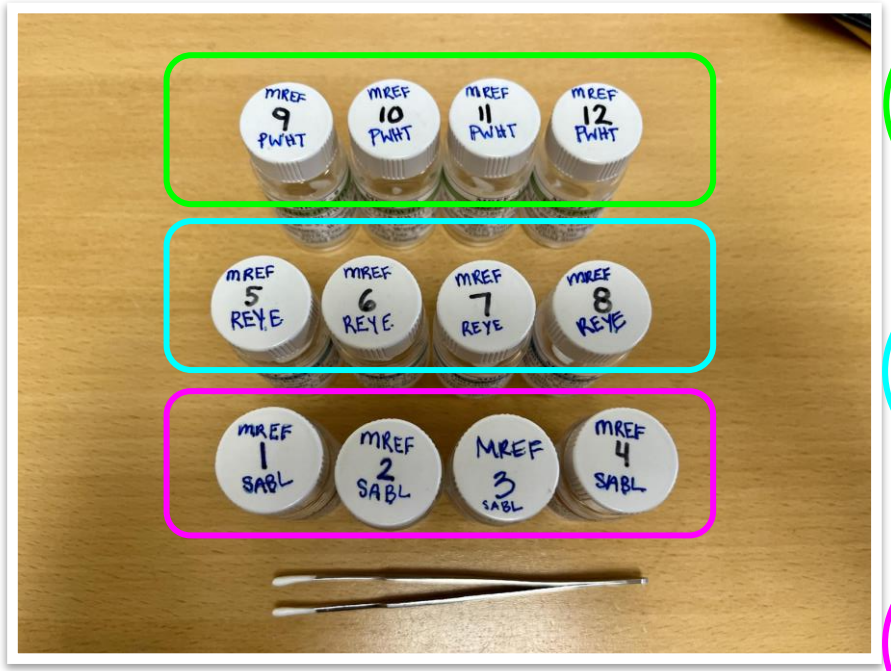


Missing pieces



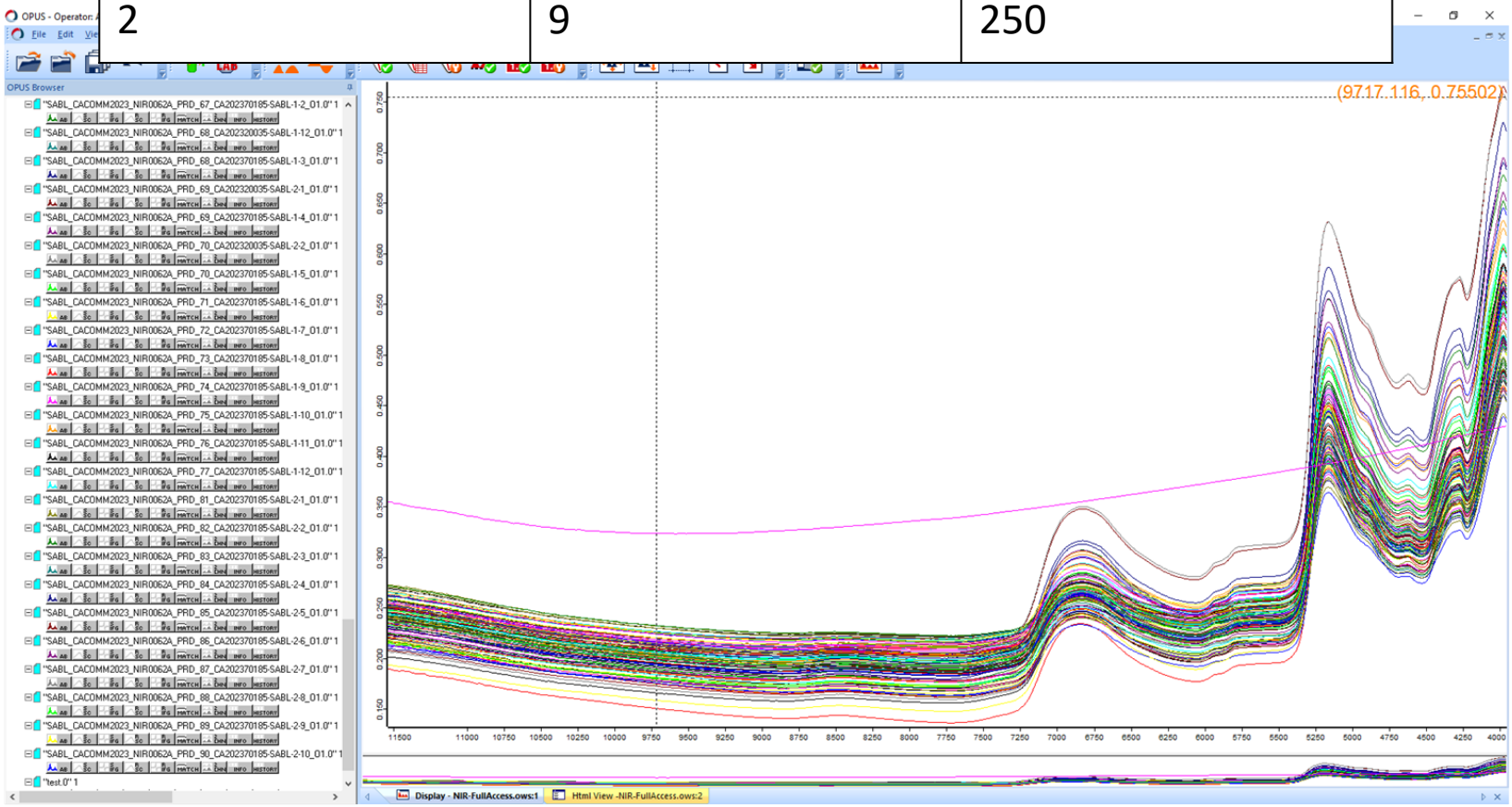


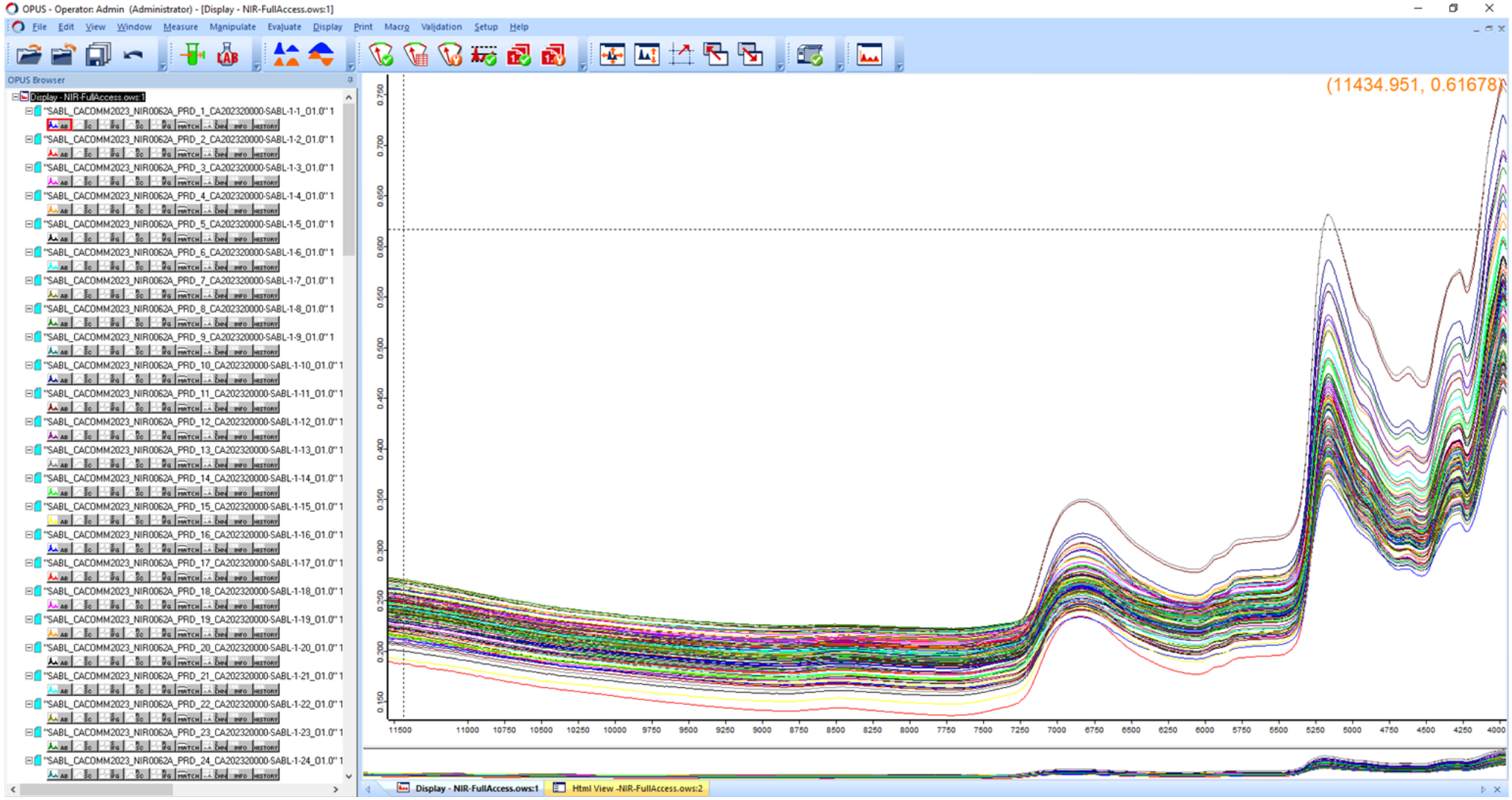
Reference Scans



Production Scans

Operators	Machine Operating Hours	Avg Production Scans
1	6	175
2	9	250

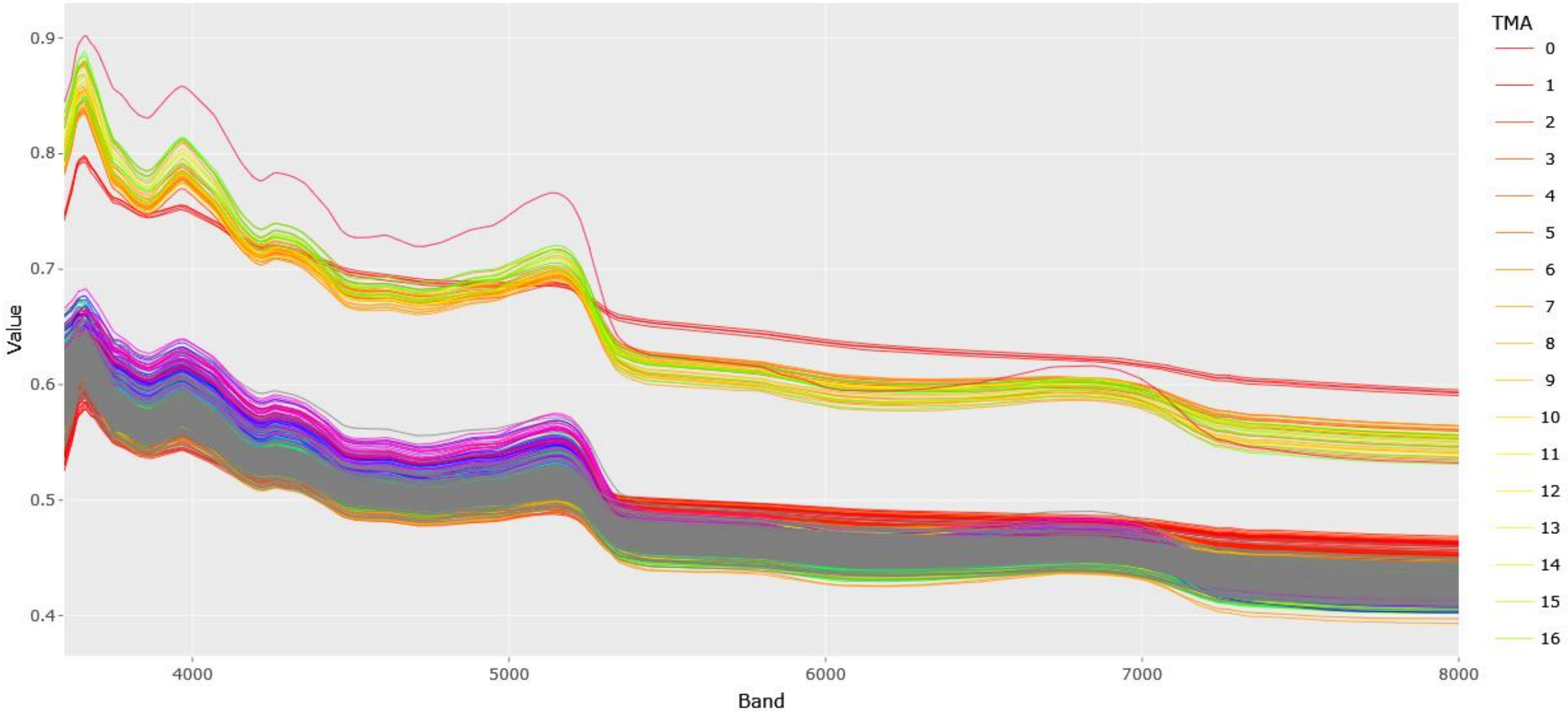



OPUS - Operator: Admin (Administrator) - [Display - NIR-FullAccess.ows:1]
File Edit View Window Measure Manipulate Evaluate Display Print Macro Validation Setup Help
OPUS Browser
Display - NIR-FullAccess.ows:1
(11434.951, 0.61678)
Display - NIR-FullAccess.ows:1
Html View -NIR-FullAccess.ows:2

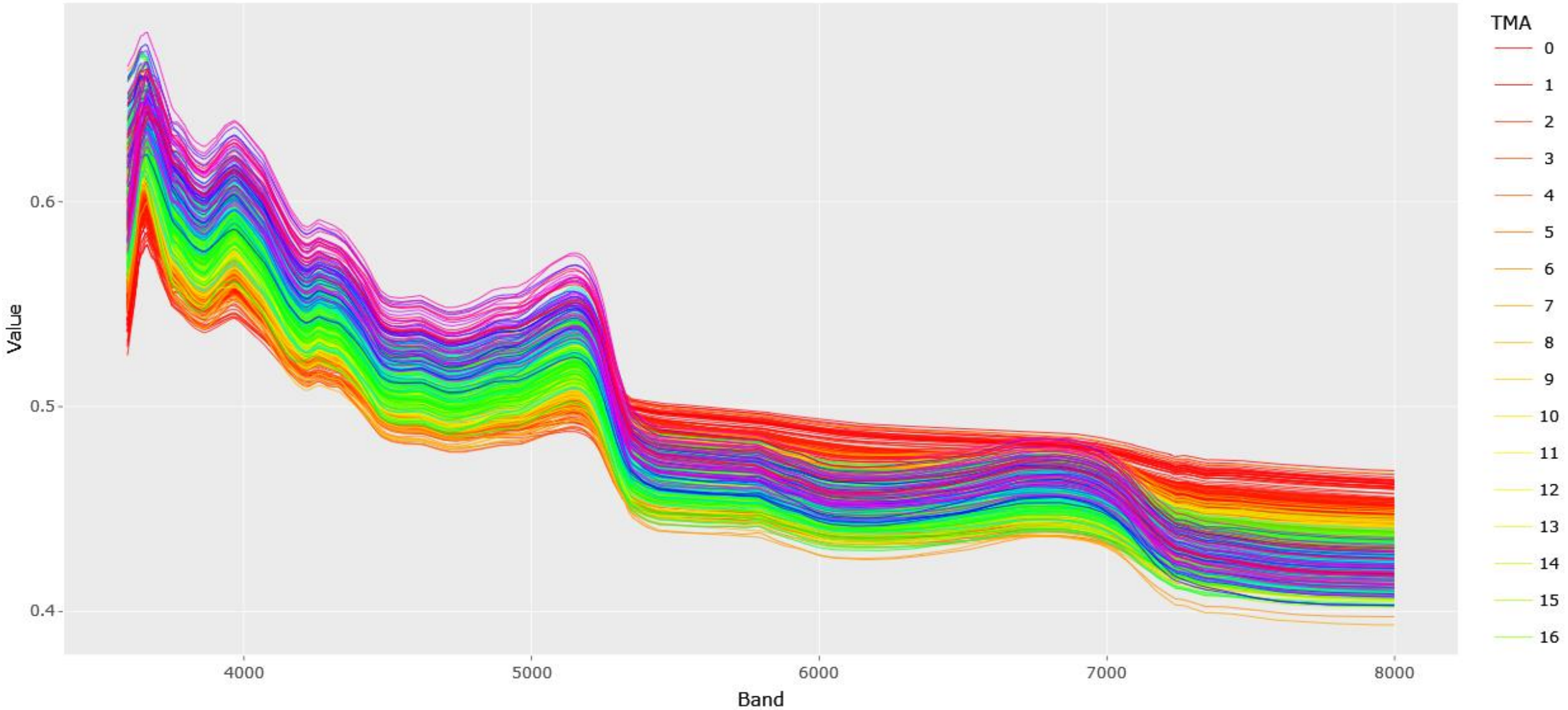
Definitions

- TMA = Traditional Method of Aging (Break and burn for this modeling.) Not the true age.
- Delta = the amount added to $\mathbb{R}$ estimated age before rounding to an integer.
  Delta is a negative number.
- SAD = Sum of the Absolute Deviations
  - In the agreement figures, 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. Hence, the unstandardized SAD is only consistent within a dataset but seems well suited to rounded age data.
- MRE = Mean Relative error = SAD/N, where N is the number of otoliths summed in SAD.
- APE = Average Percent Error:
  - Often used by traditional age readers to quantify how good the double reads are.
- RMSE = Square Root of the Mean Squared Error.
- FCNN is a Fully Connected Neural Net model, where every node in the neural net is connected to all the other nodes.
- Convoluted NN (CNN) is where not all nodes are connected.

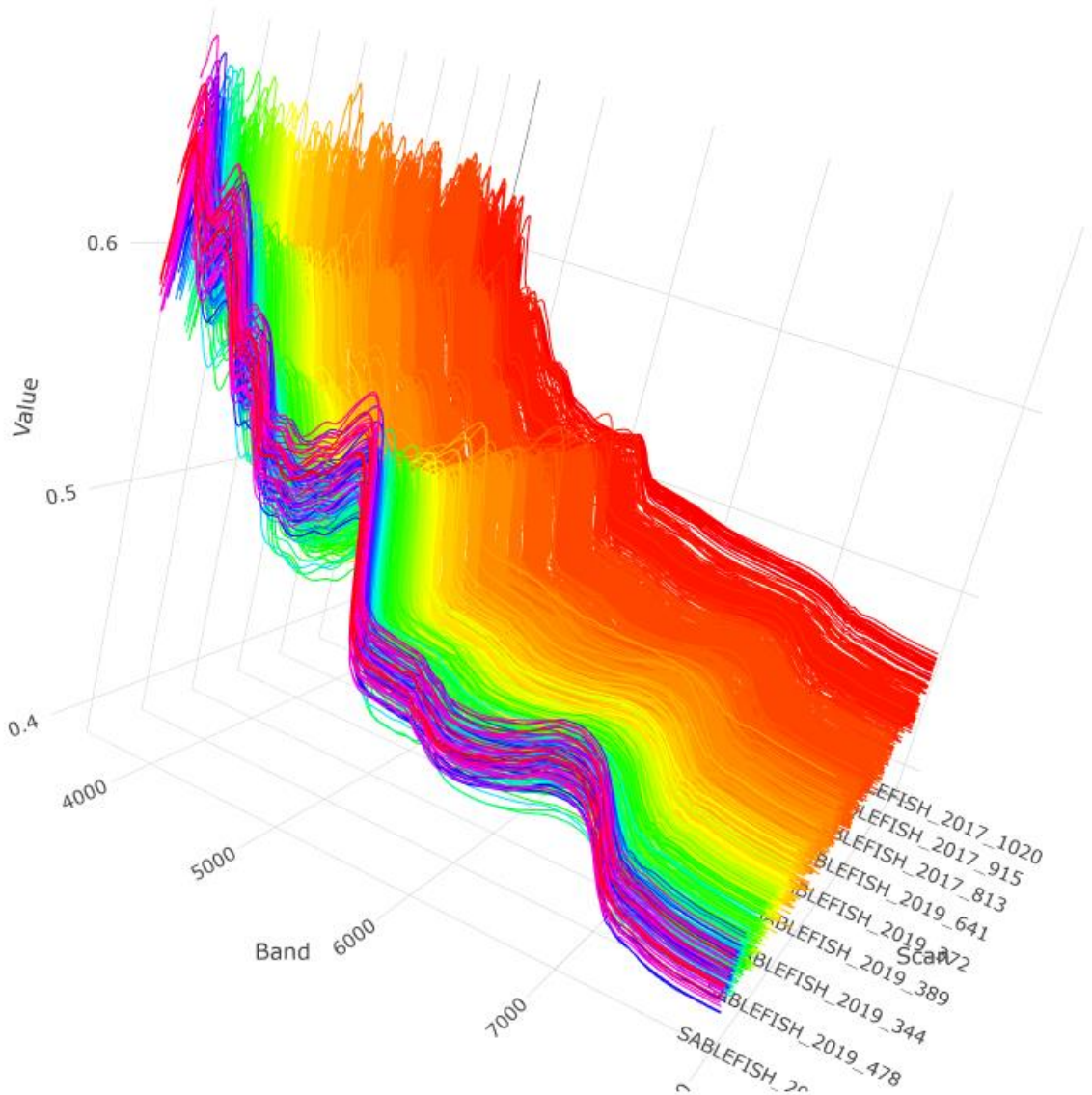
Spectra with Missing Traditional Method of Aging (TMA) in Grey



Extreme Spectra and Missing TMA 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_572
SABLEFISH_2019_389
SABLEFISH_2019_344
SABLEFISH_2019_478

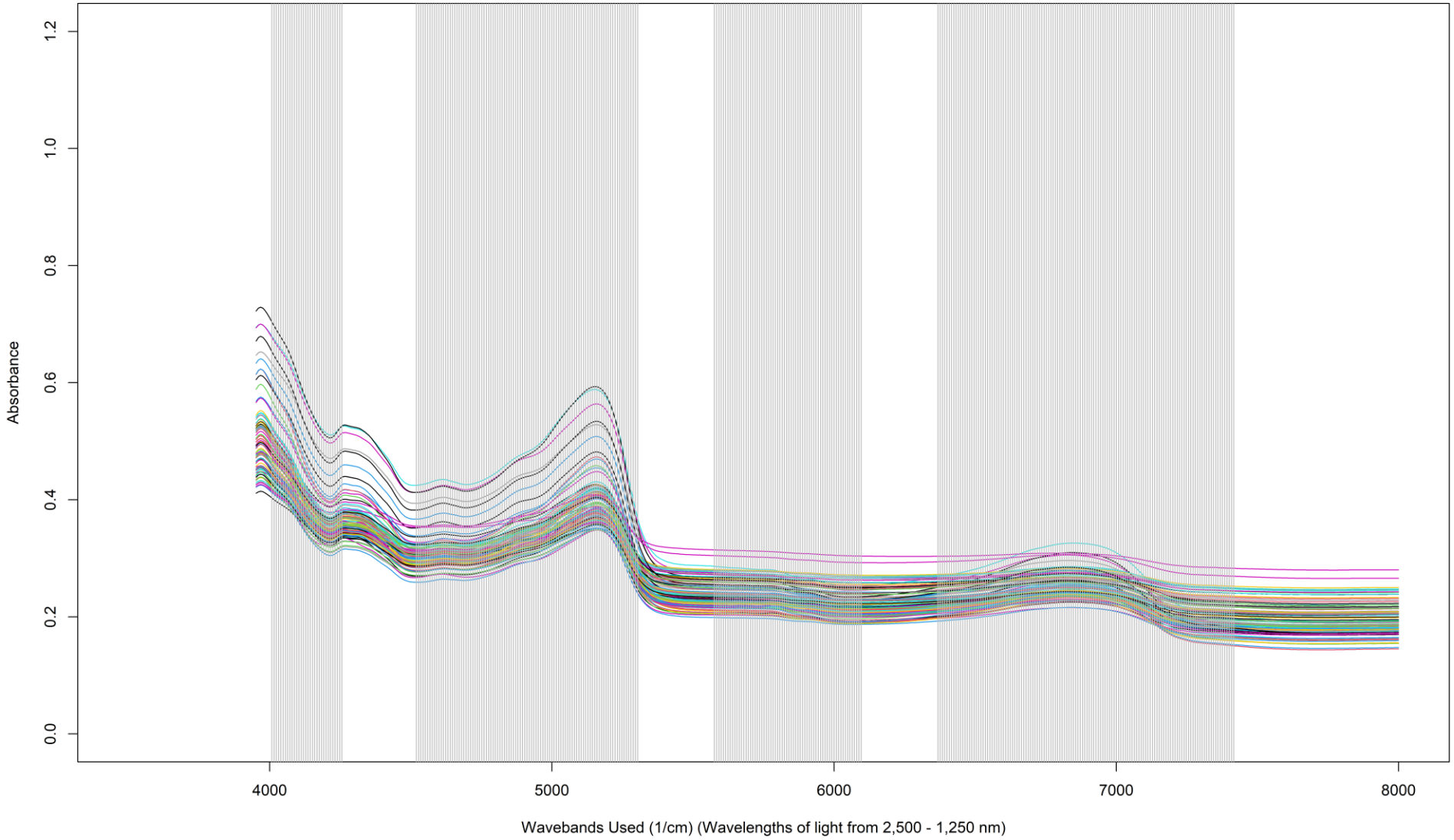
Prelude to the Roadmap

- Validation levels, from deep in the code to higher levels
  - In the NN model, on the epoch level, there is an optional, user selected validation level. I used the standard 20%. (80% of the data was used to train and 20% to test the model.)
  - On the mid-level there is the main data split into the training set and the test set. I used a common 2/3, 1/3 split.
  - On the k-fold level, a 10-fold model was the standard used. One tenth of the data was left untouched for testing the 2/3, 1/3 model from the remaining 90% of the data. If a 10-fold model was not feasible, a 5-fold model was used.

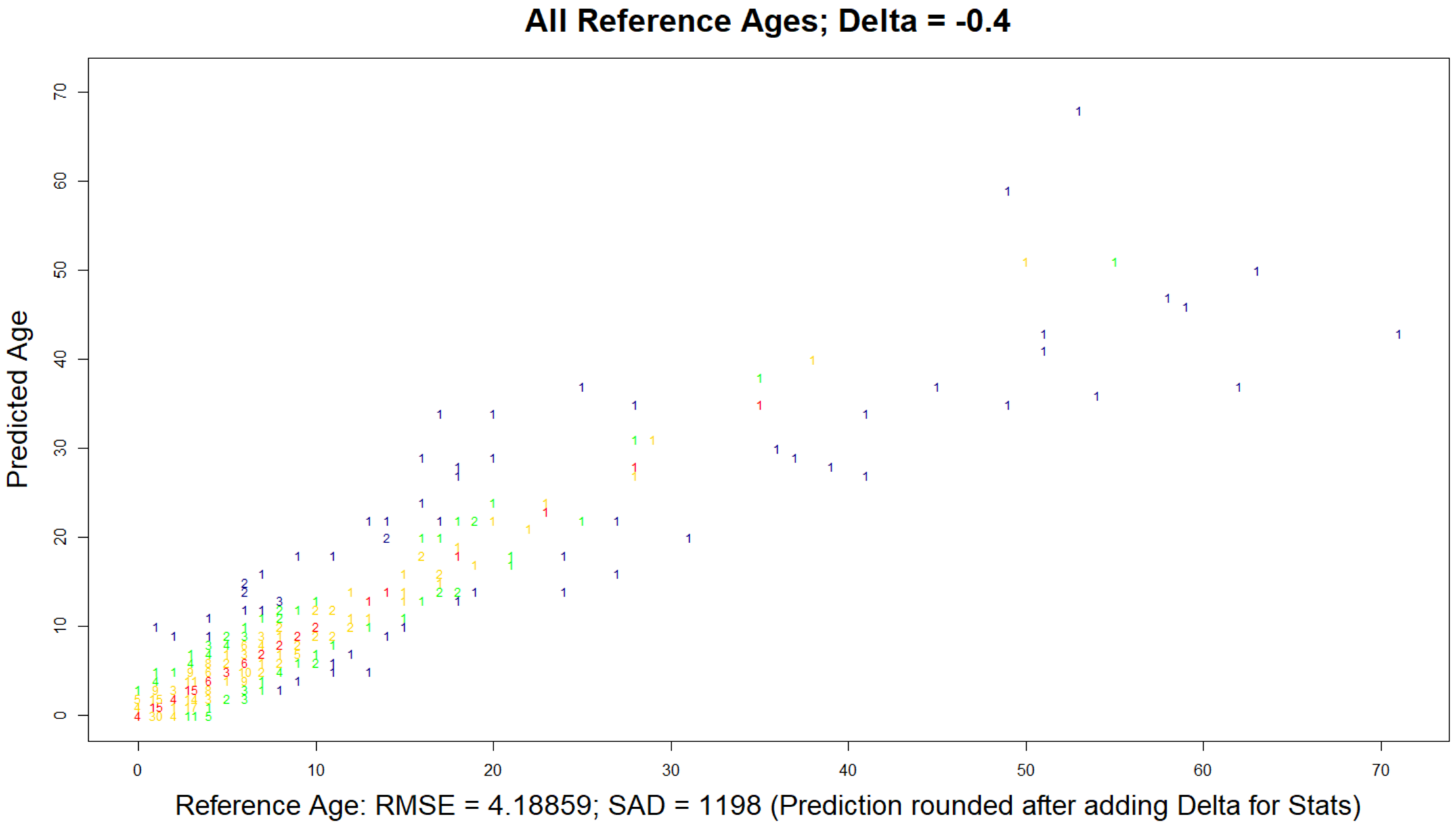
Roadmap

1. Raw Data
2. iPLS (iterative Partial Least Squares) model used to identify waveband ranges that are most informative. (Spectral analysis initially followed Jordan Healey's code - thanks!)
3. PLS model (2/3, 1/3 split)
4. Neural Net (NN) model, using the same data to compare to the PLS model
5. NN models with k-folding, starting with the same iPLS result.
6. Medians taken over 20 complete k-folds, each with a different pseudo random number generation starting point.

Example of wavebands identified via iPLS (vertical lines)



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

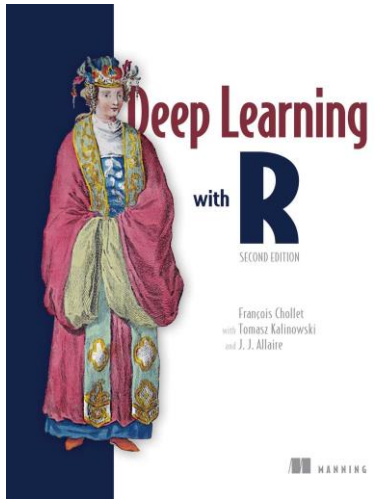


Moved to NN models

- It appeared that the PLS models were not great, 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' (https://keras.io) R package (https://tensorflow.rstudio.com; RStudio is not needed) that sits on top of Google's TensorFlow software (https://www.tensorflow.org).

- YouTube video on neural networks:
  - https://m.youtube.com/watch?v=aircAruvnKk

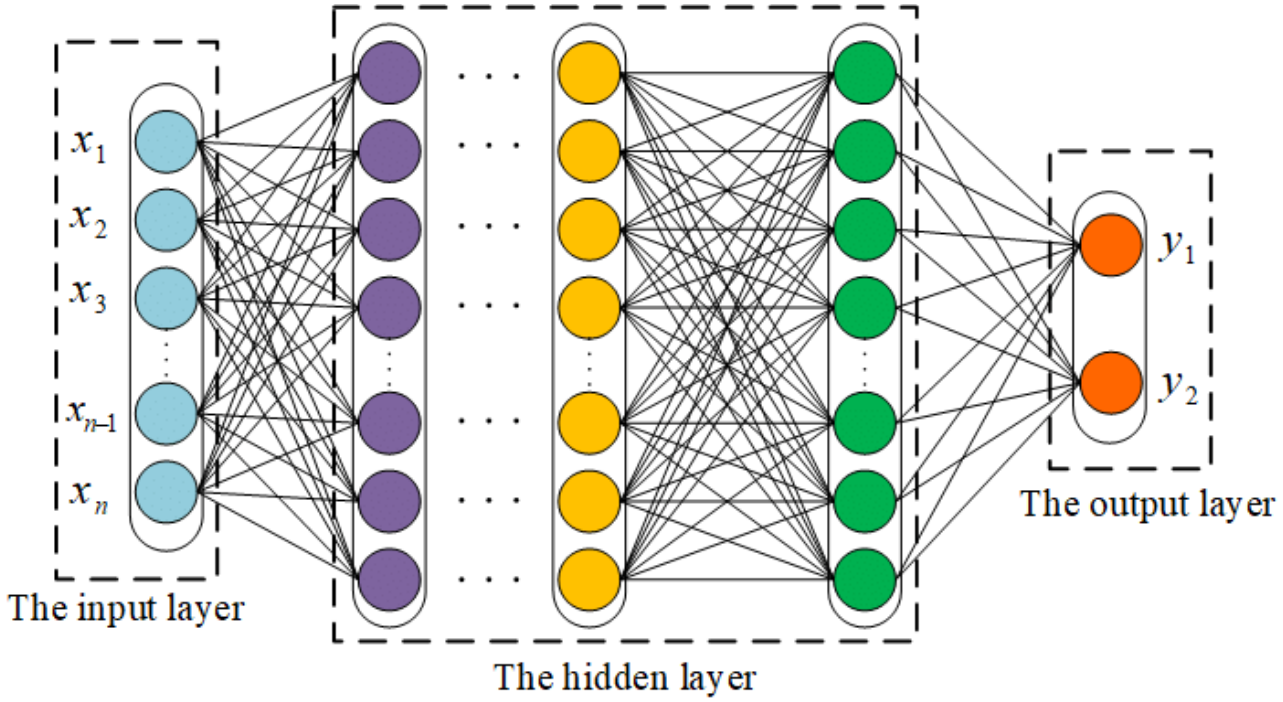


Software used and Issues

- Google's TensorFlow (created with Python sitting on top of C++)
- Keras NN modeling software (often done in Python)
- Complete Keras package in R (Only a few advanced examples found on the Web.)
- Currently there are various version conflicts (was better a few years ago)
  - Hard to get to work in WSL (Window's Subsystem for Linux)
  - Harder to get working in Native Windows (but it is faster, than WSL).
  - Even harder in Windows Server 2019 (work abandoned).
- All this sits on top of the Nvidia's CUDA (Compute Unified Device Architecture) software which makes a high-end graphics card into a GPU (Graphics Processing Unit)

Once the software works, you need a NN Model

- I followed the reference below and also found simple models worked best for NIR Spectroscopy on otoliths. 1D spectroscopy is not 2D or 3D (greyscale) image recognition for which more complex models work better.
- *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."



Example of a More Complex 1D Model

- Other, more complex, 1D models did not perform as well for the amount of data I had.
  - Many of the more complex models are fully or partially convoluted with many hidden layers
- *Using a One-Dimensional Convolutional Neural Network on Visible and Near-Infrared Spectroscopy to Improve Soil Phosphorus Prediction in Madagascar*. Kawamura et al. Remote Sens. 2021, 13, 1519. (They have a lot of data to work with.)

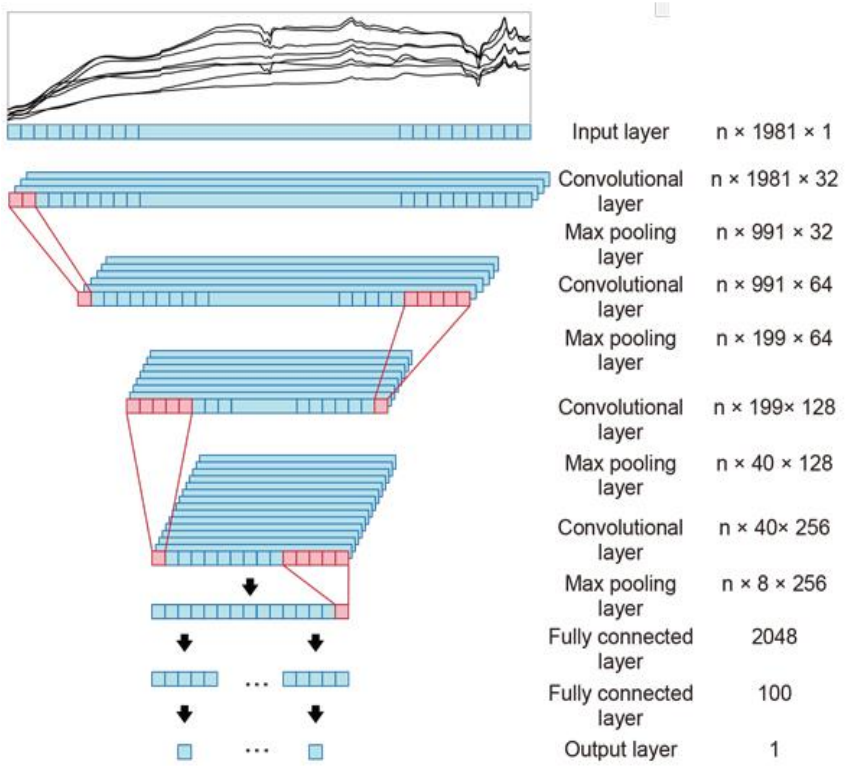


Figure from the reference

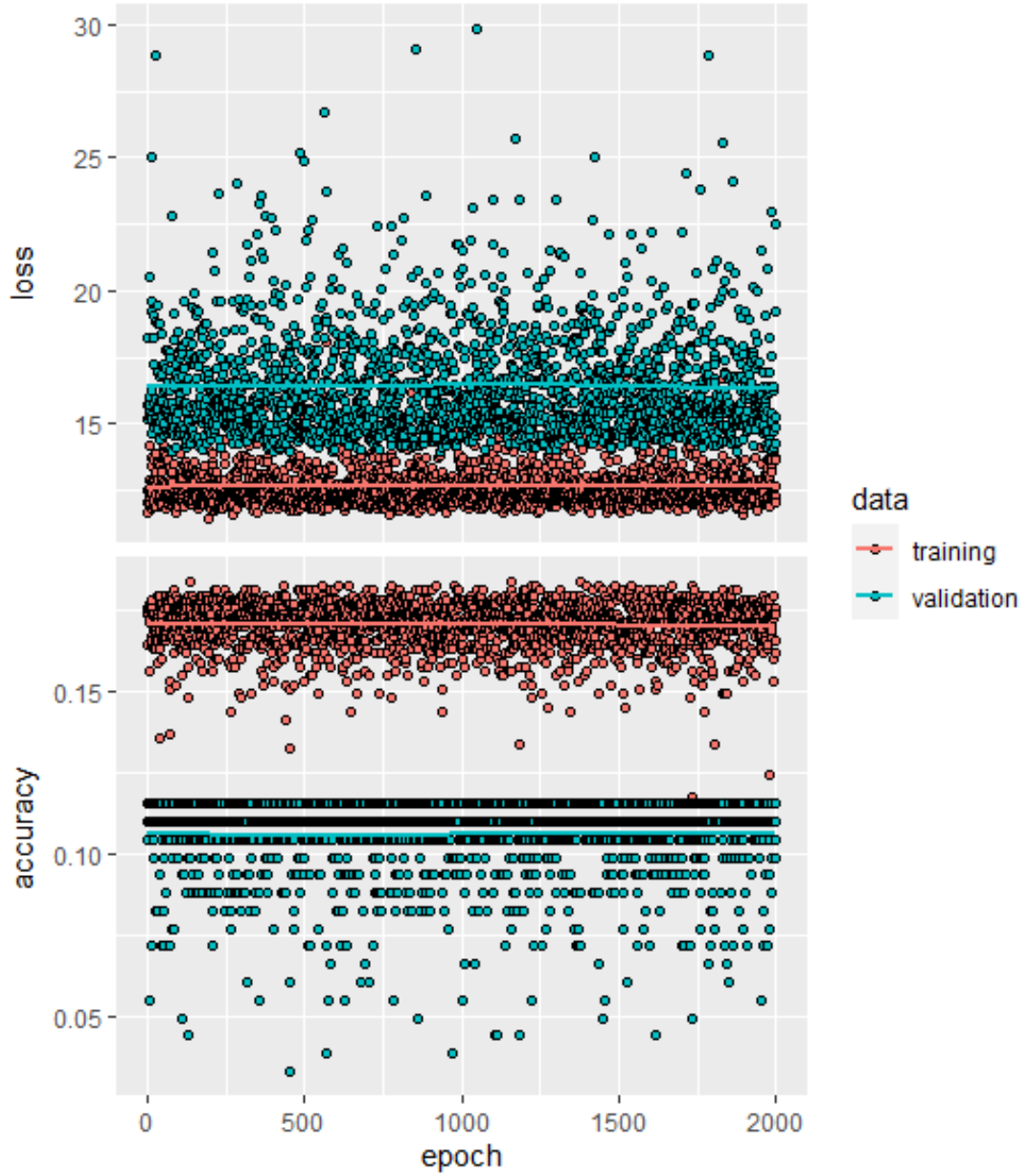
Neural Net Modeling Basics

- Normalize the input.
  - Getting the mean close to 0, and not doing min-max normalization when possible, worked best for the iPLS input that went into the NN models.
- That data needs to be split into training and testing sets.
- 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.

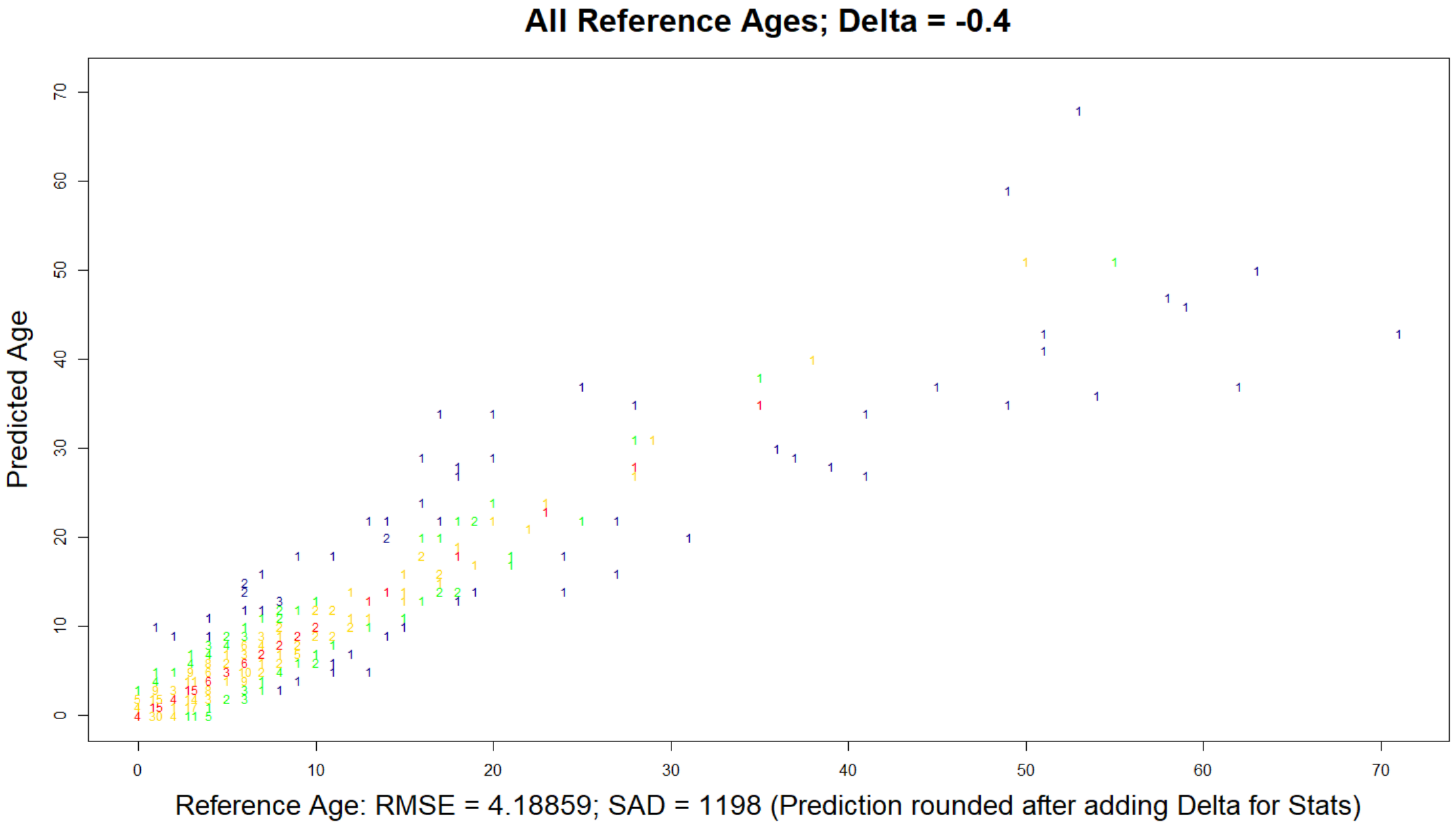
Neural Net Modeling Basics (cont.)

- 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 of 500 epochs each, with testing against the 1/3 test data done at the end on each iteration to view and record progress.
- Batch sizes of 32 and a validation split of 0.20 (80% of the data was used to train and 20% to test the model) was used.

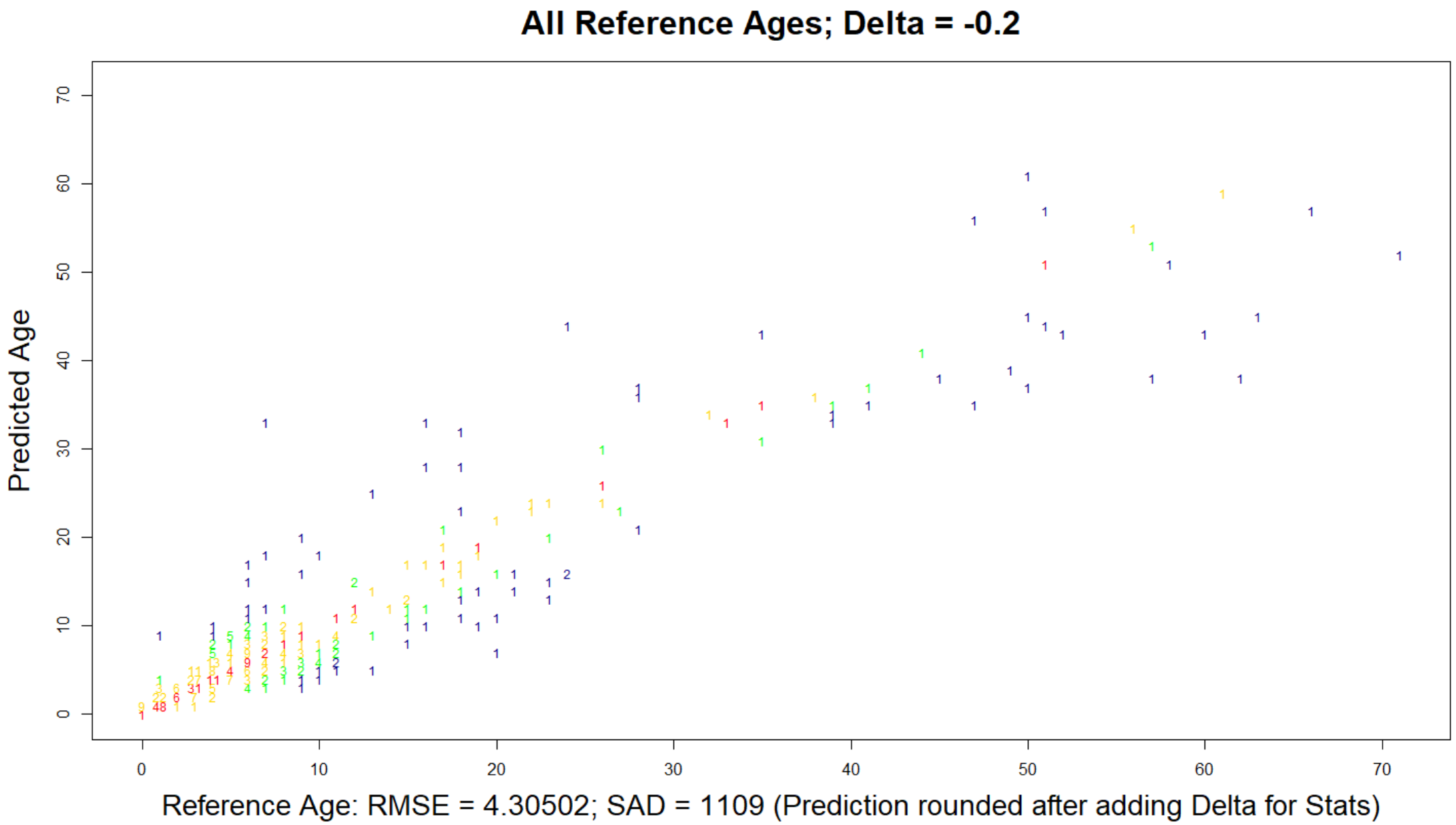
Loss and Accuracy for Training and Validation over the Epochs



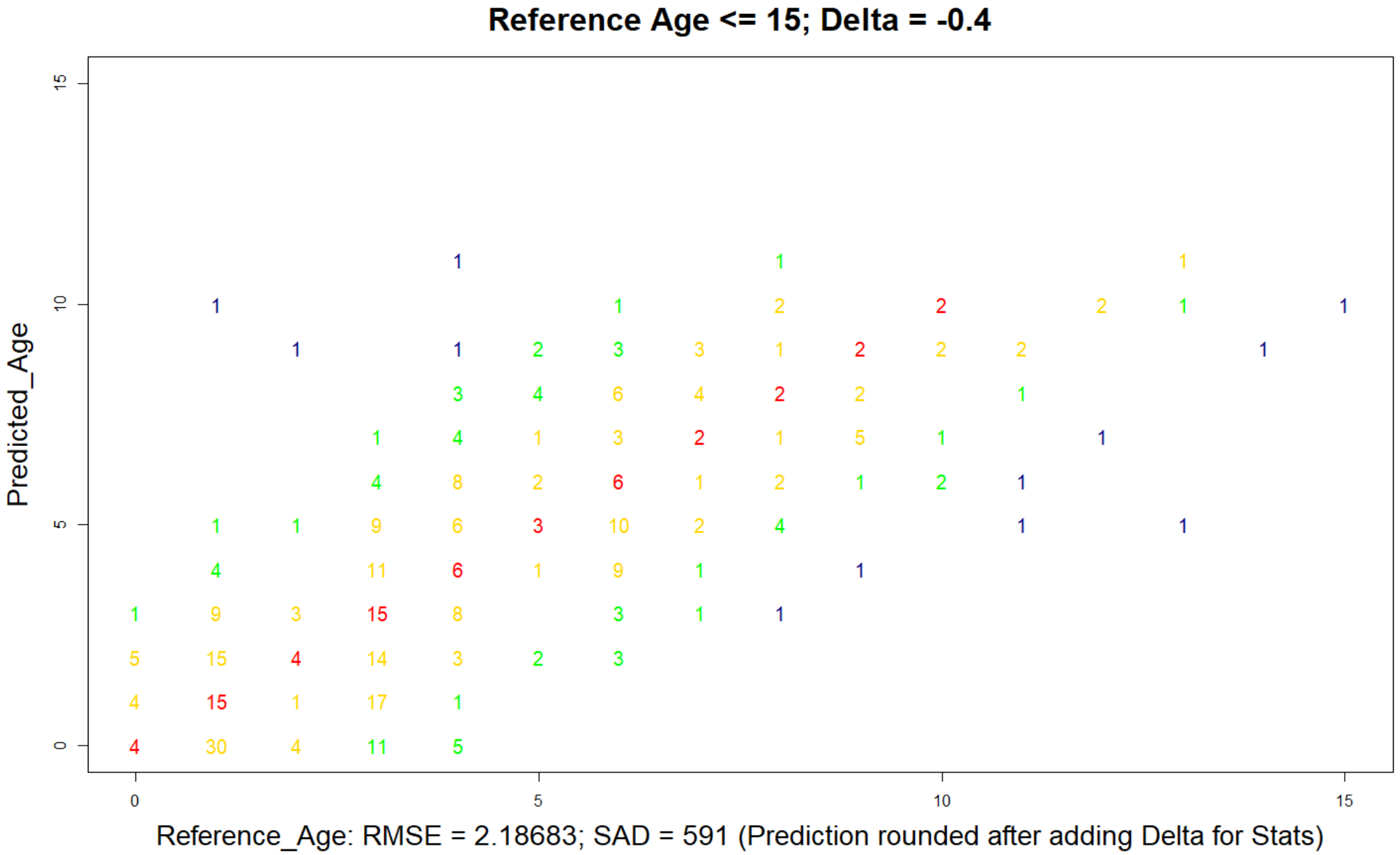
Sablefish, PLS Model Predicted onto Test Data (1/3 of the Total)



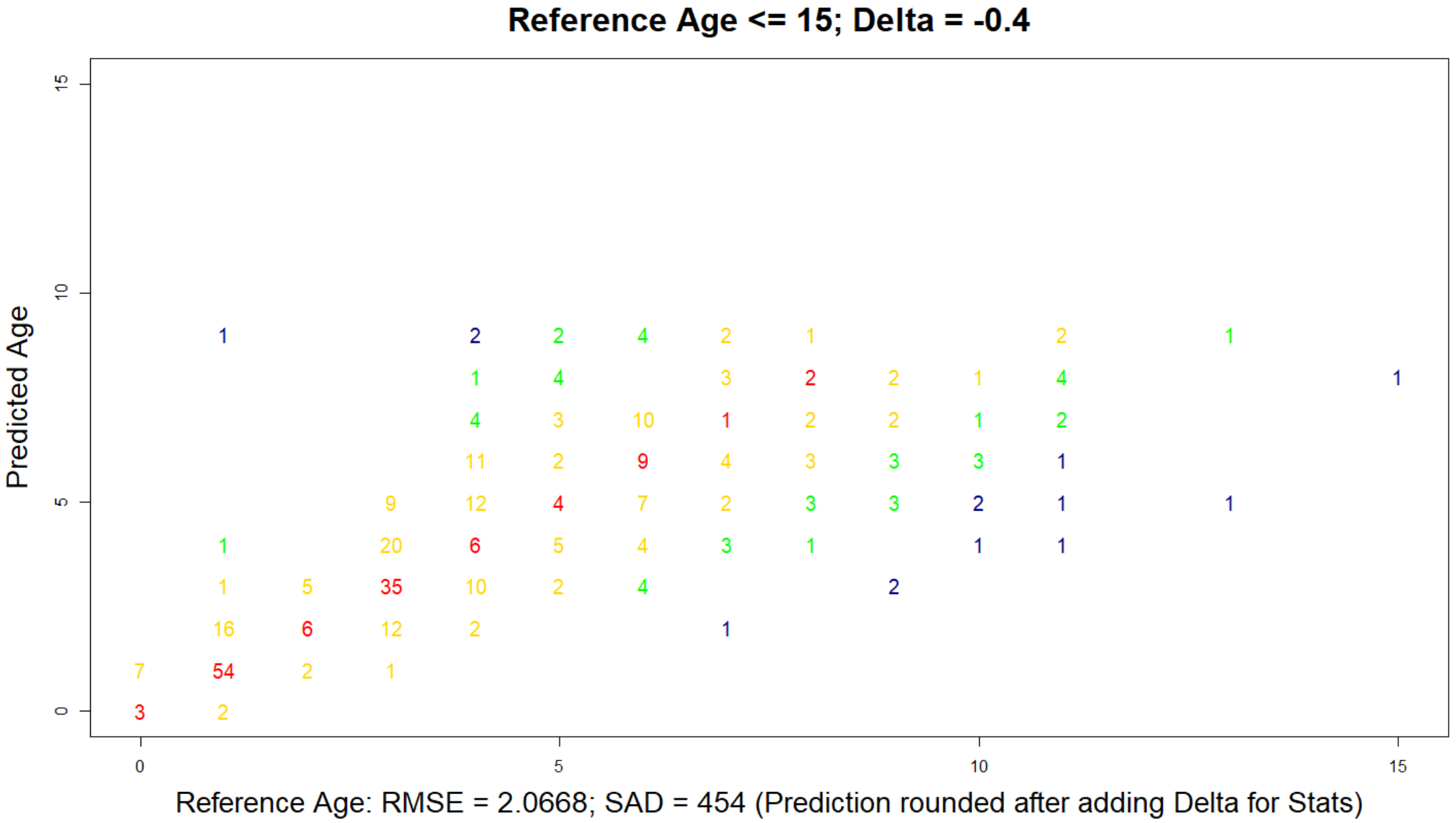
Sablefish, FCNN Model Predicted onto Test Data (1/3 of the Total)



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

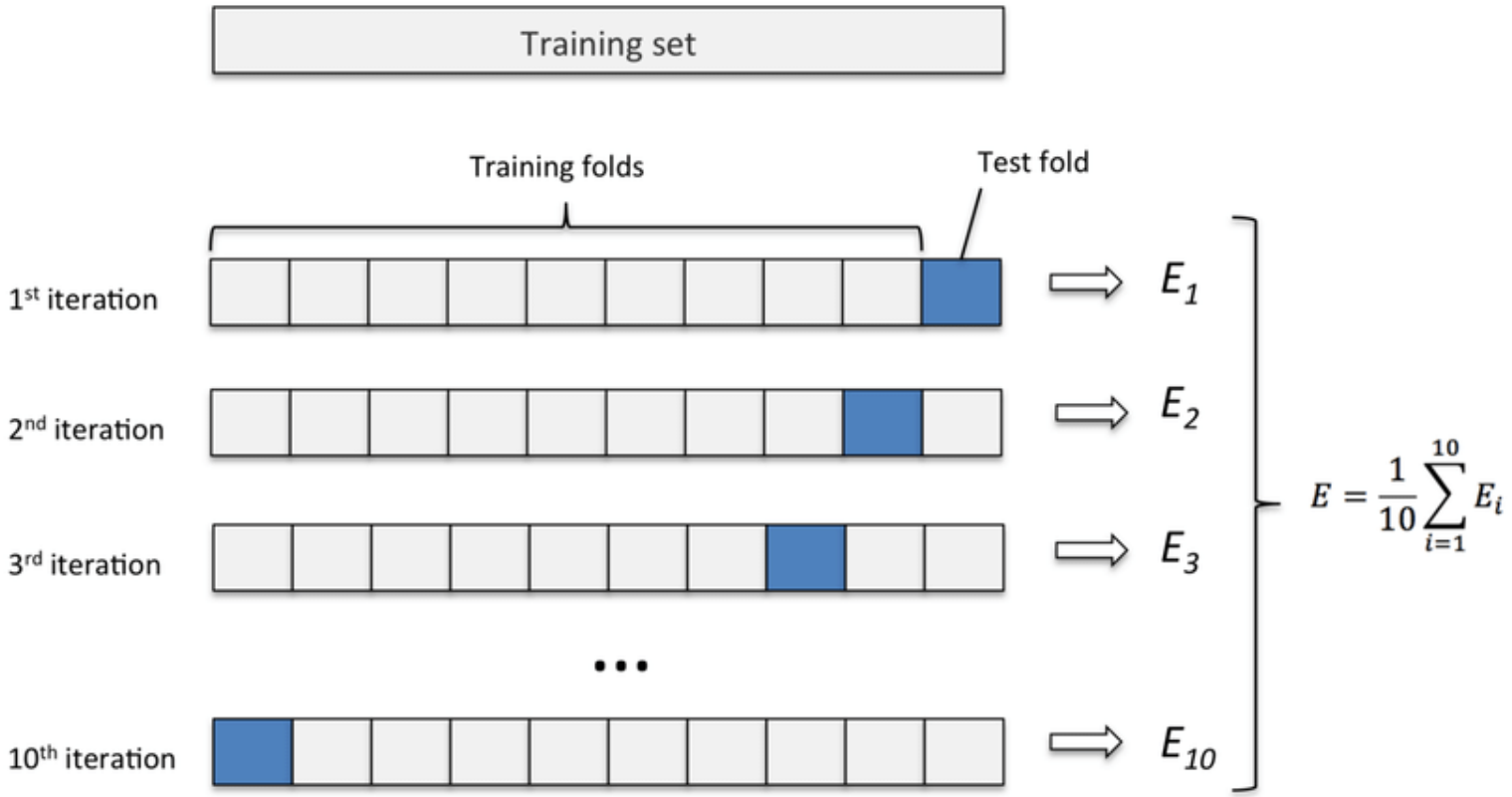


Sablefish, FCNN Predicted onto the Test Data, Ref Age <=15



Next Step: K-fold Modeling

- A 10-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.)

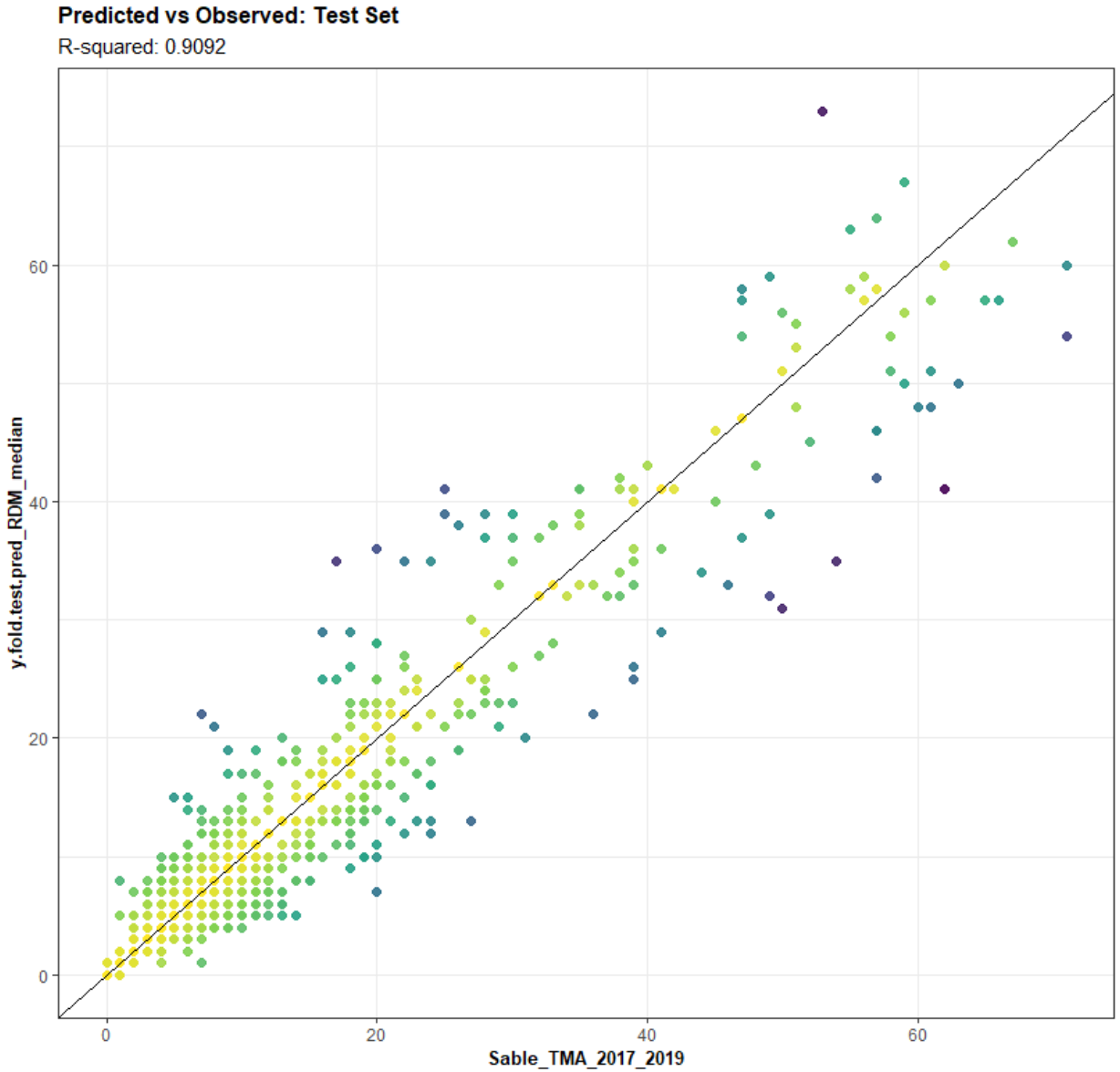
- Of that 90%, 2/3 was used for training and 1/3 for testing that particular model (as before).
- To train the sub-model, 500 neural net epochs were run on the training set and then tested against the 1/3 test set.
- Eight such iterations, of 500 epochs each were performed.
- As before, a validation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

K-fold Modeling (cont.)

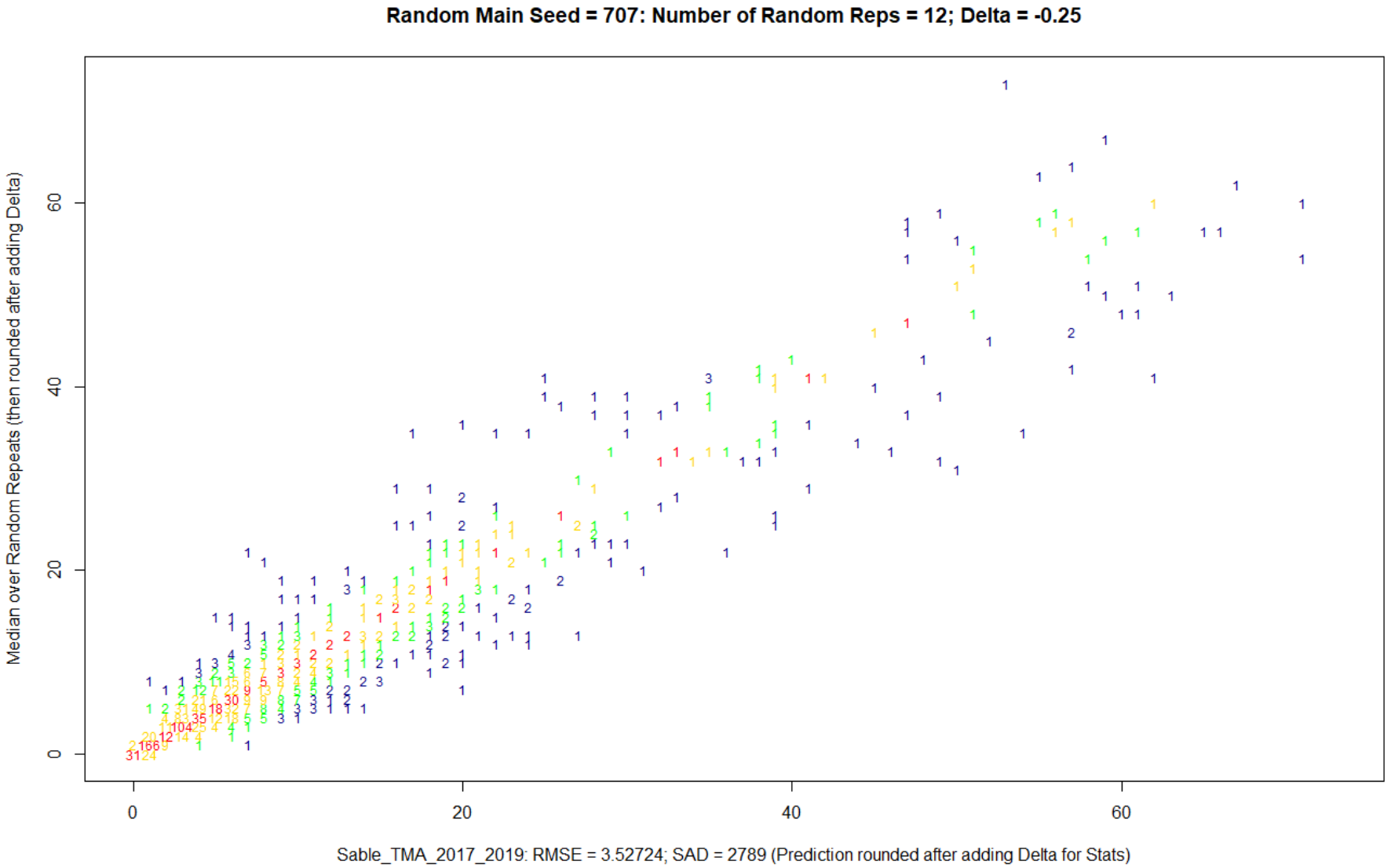
- After all 10% folds are set aside in turn, a complete fold set is finished, and each predicted point was never inside a model that predicted it.

- Twenty complete k-fold models were run, each with a different pseudo random number seed, controlled by a main seed for repeatability, restarting , or for running consecutive models.

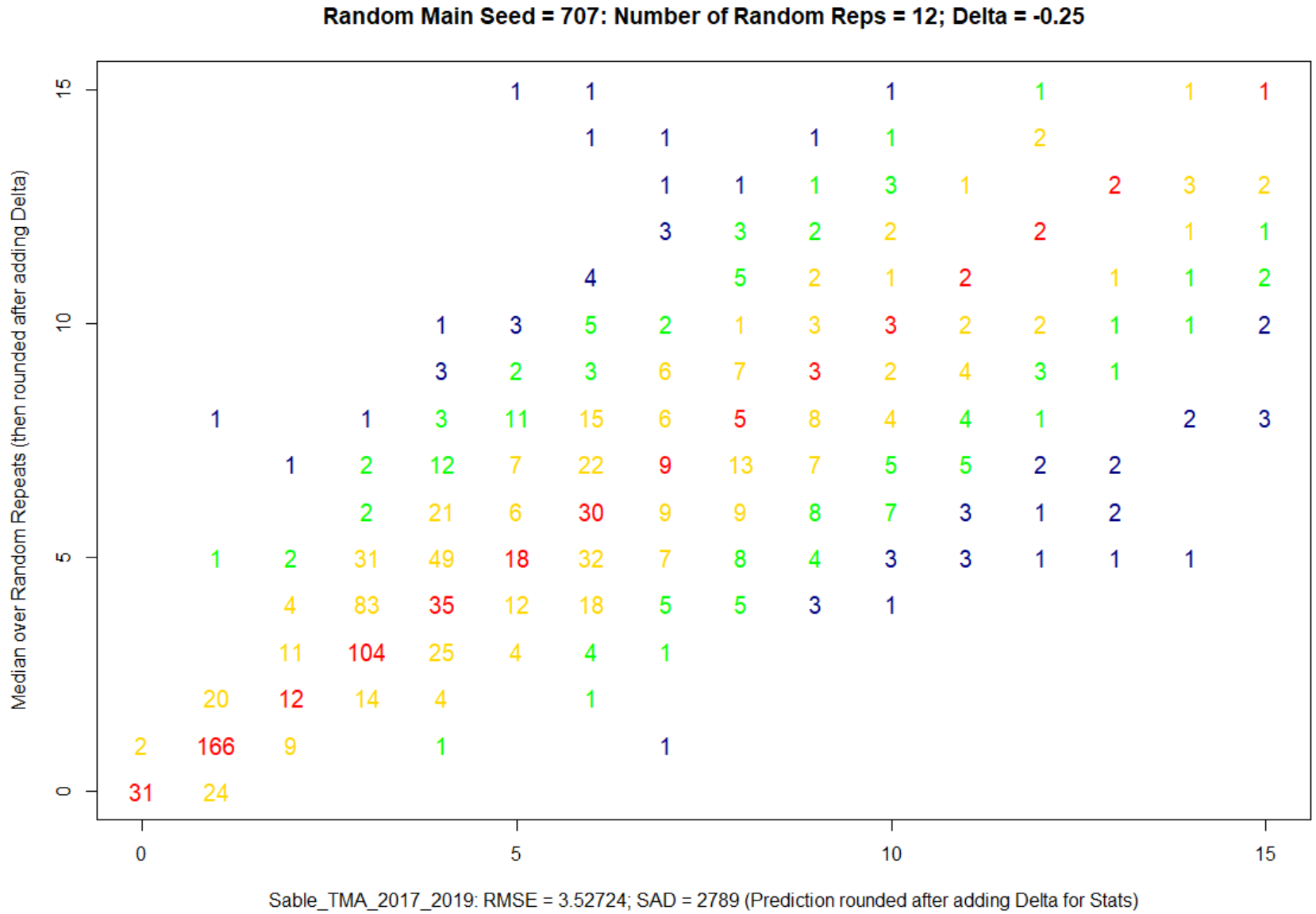
Median over 12 k-fold models vs TMA



Median over 12 k-fold models vs TMA



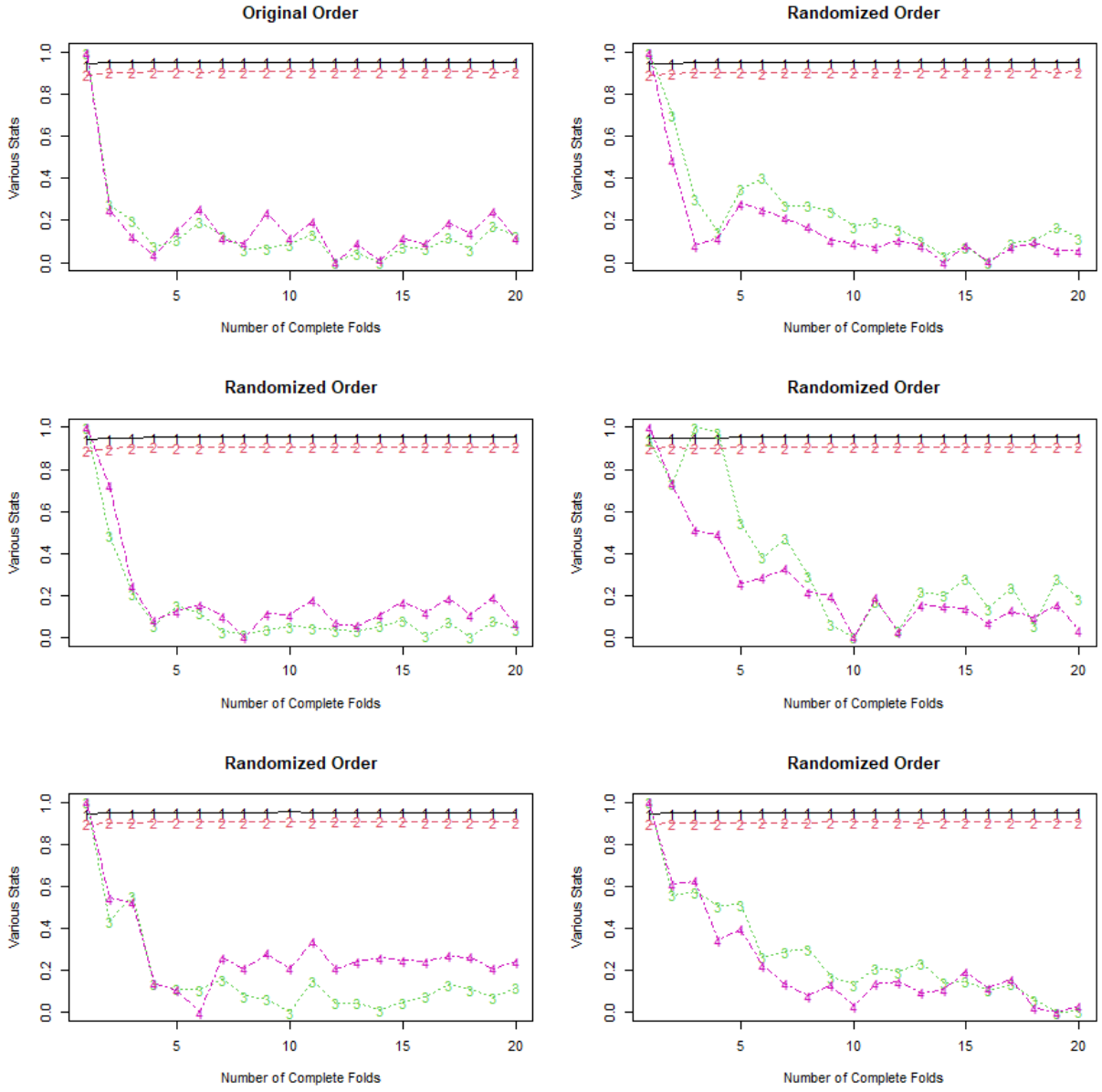
Median over 12 k-fold models vs TMA (zoomed)



Lastly

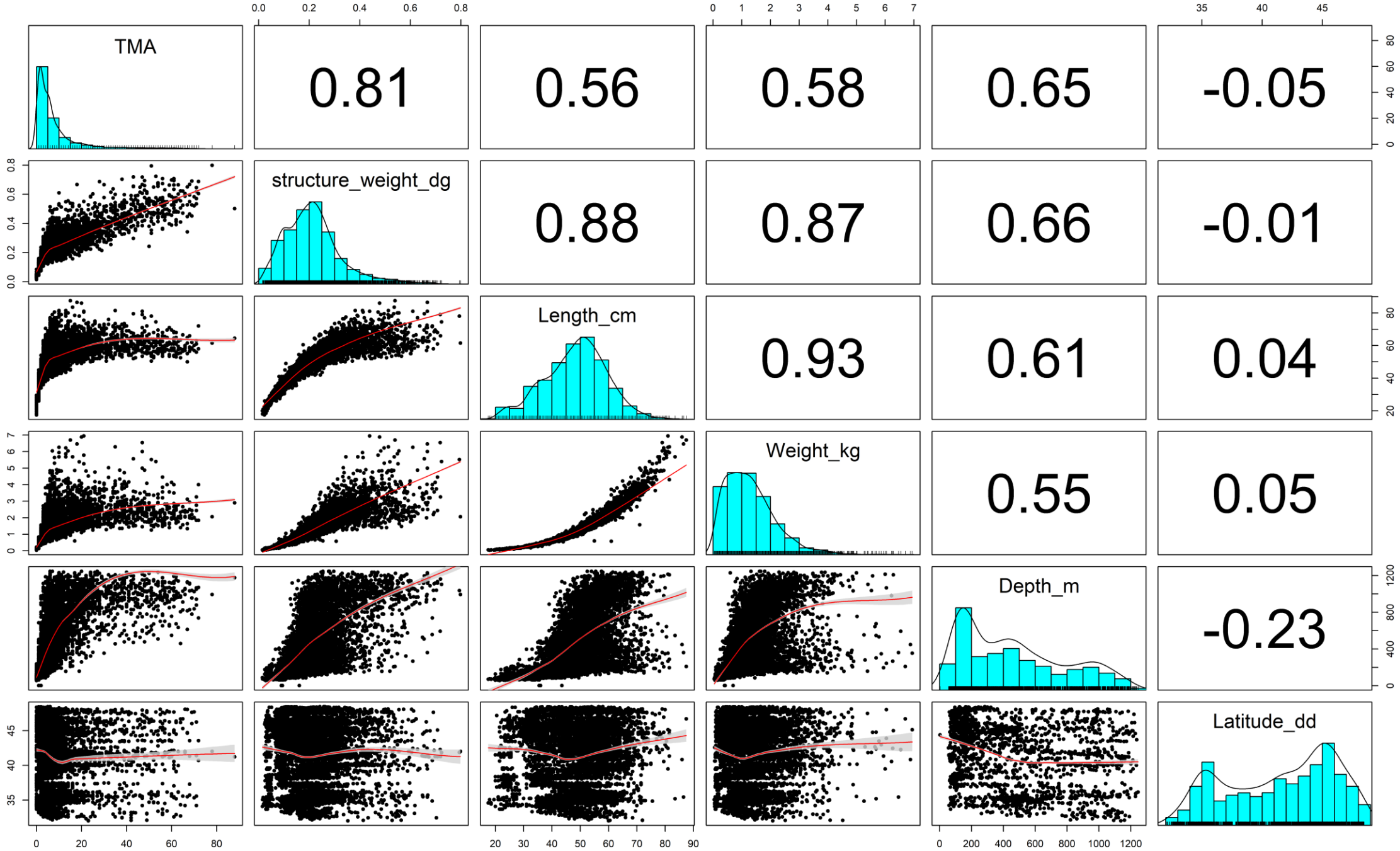
- Medians over predicted ages where taken for each additional k-fold model added at each step from 1 to 20; first in the original order and then in randomized order.

Randomized Additions of a Full k-Fold

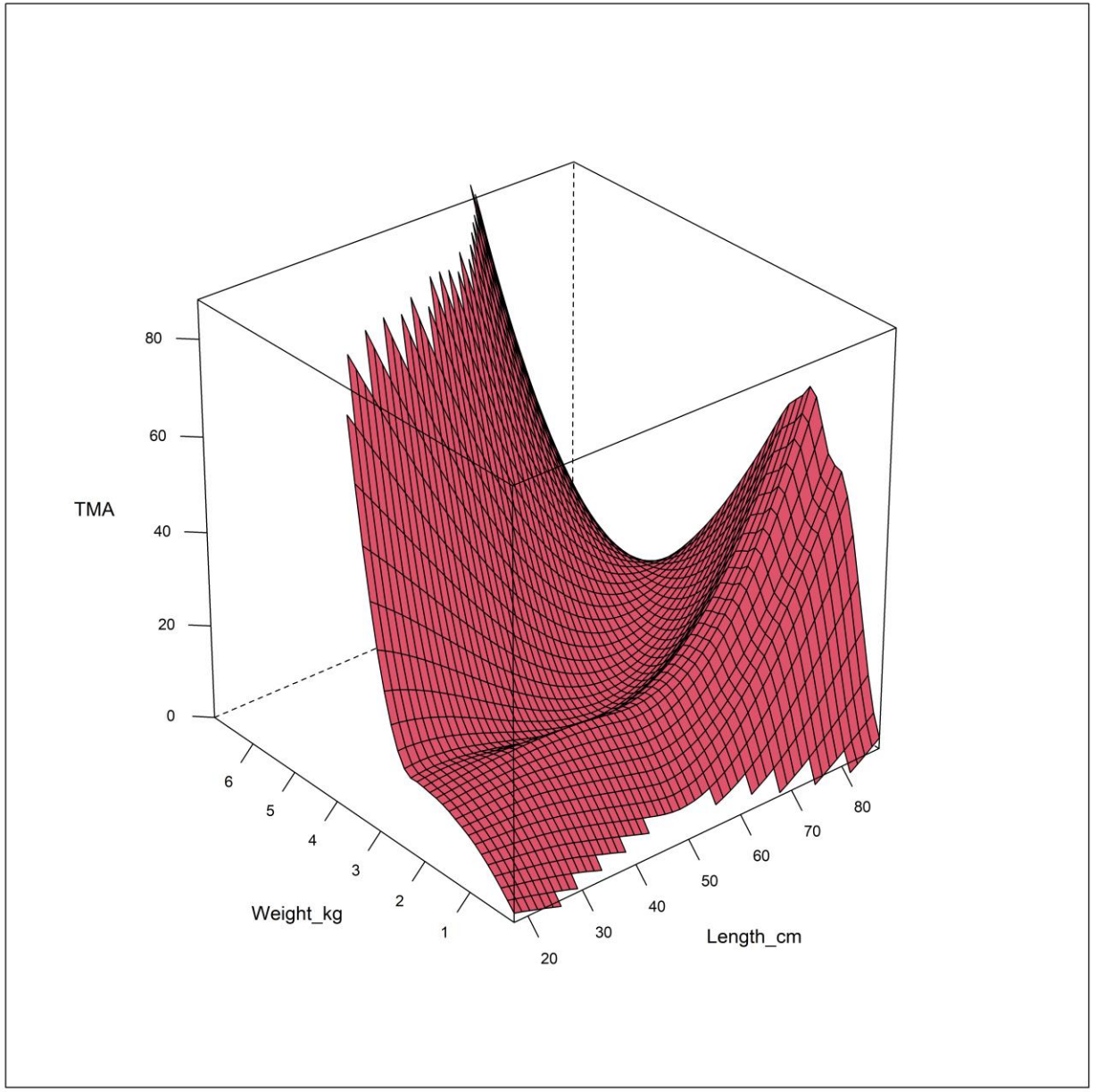


1: Correlation: Black
2: R squared: Red
3: Standardized RMSE: Green
4: Standardized SAD: Purple

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

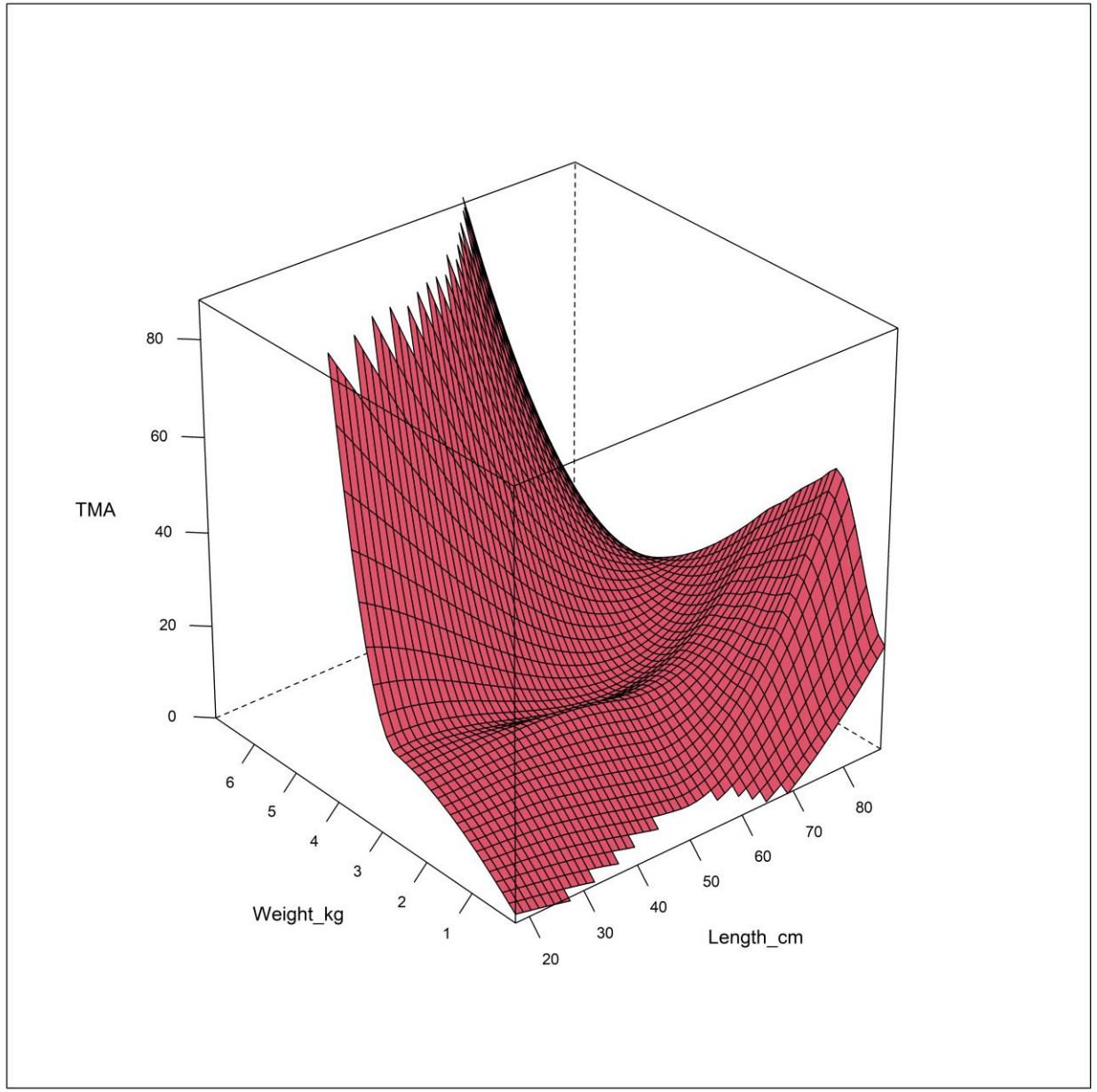

TMA
0.81
0.56
0.58
0.65
-0.05
structure_weight_dg
0.88
0.87
0.66
-0.01
Length_cm
0.93
0.61
0.04
Weight_kg
0.55
0.05
Depth_m
-0.23
Latitude_dd

TMA by Fish Length and Fish Weight



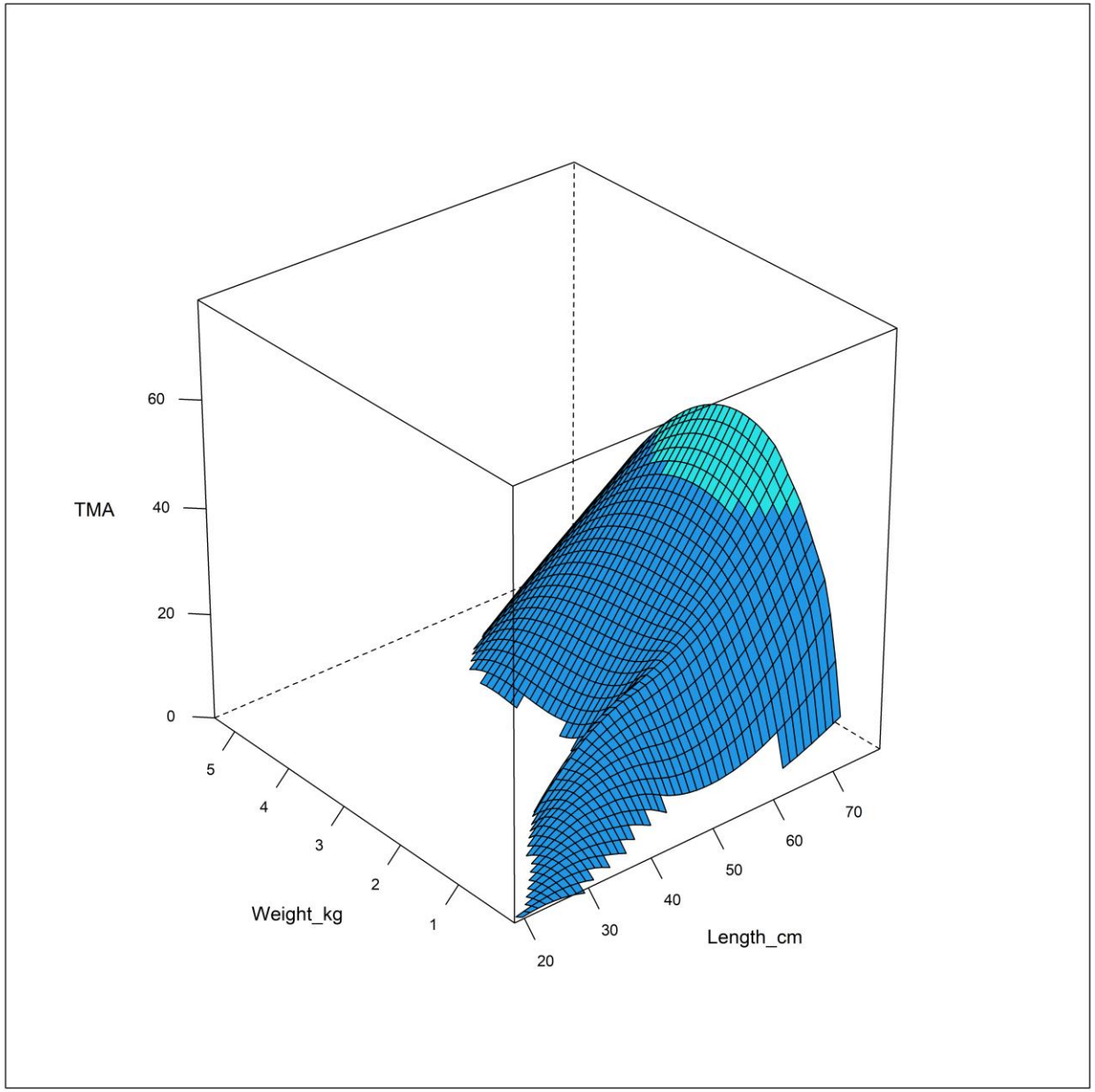
Females, Males, and Unknown

TMA by Fish Length and Fish Weight



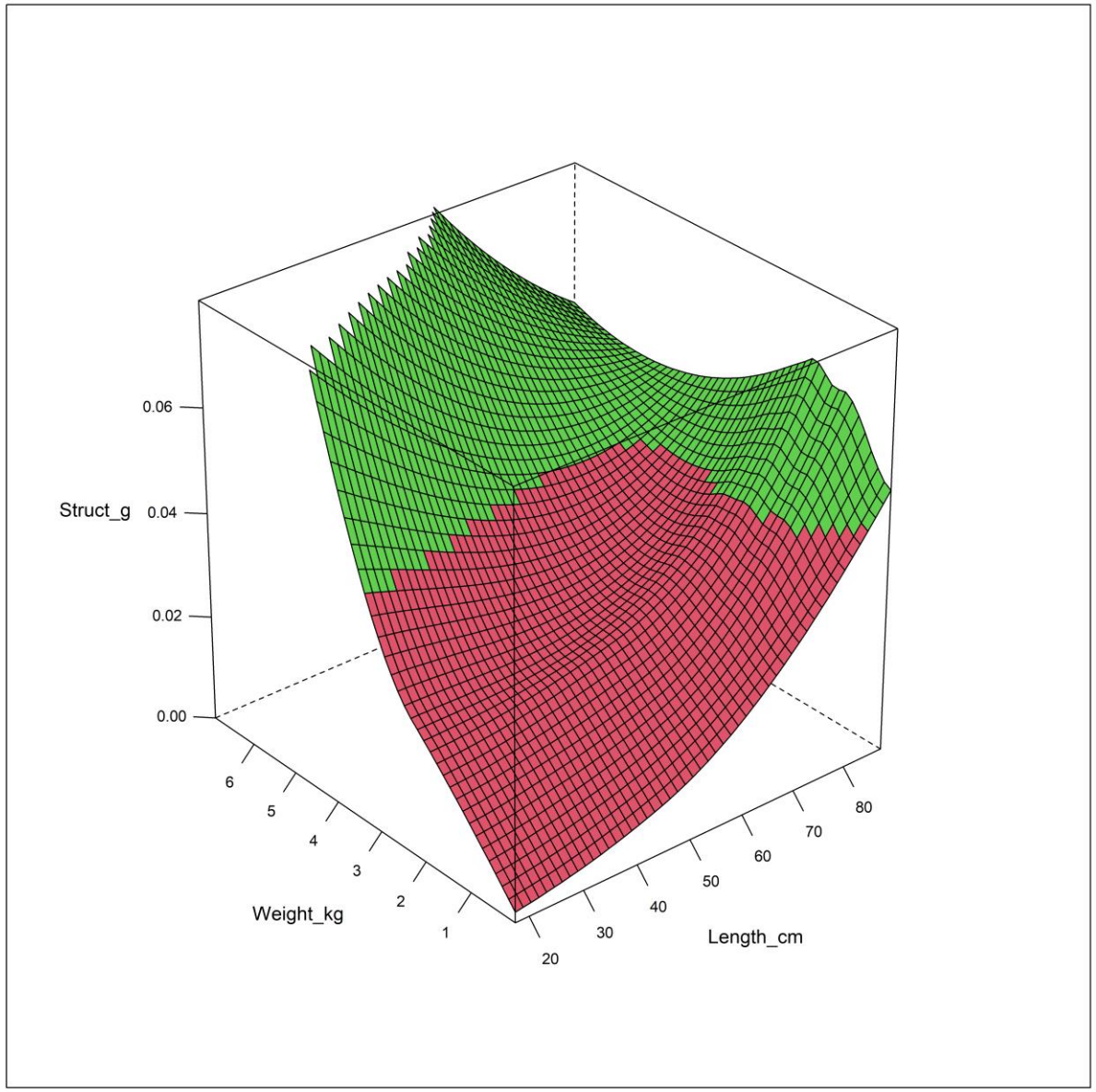
Females

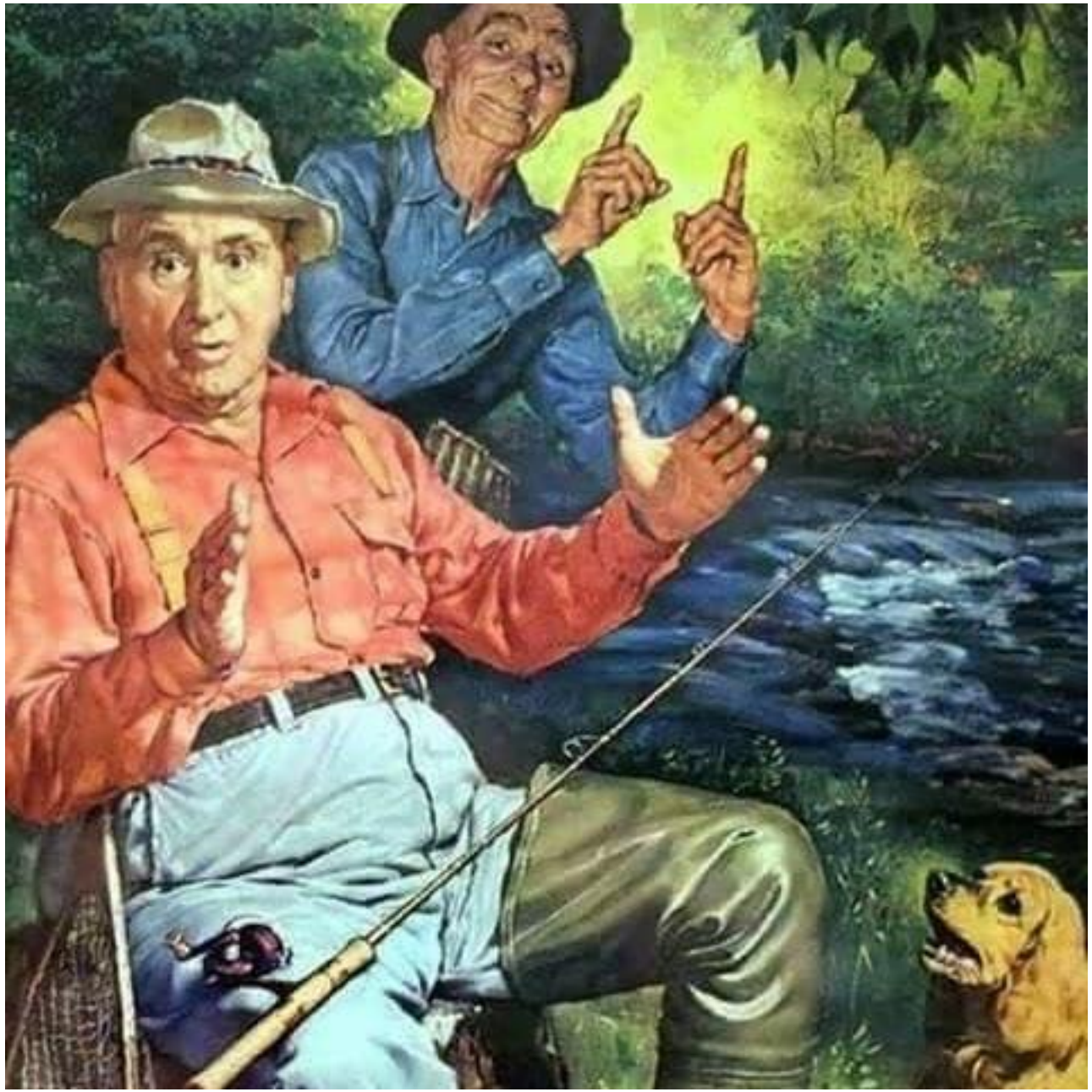
TMA by Fish Length and Fish Weight



Males

Structure Weight by Fish Length and Fish Weight



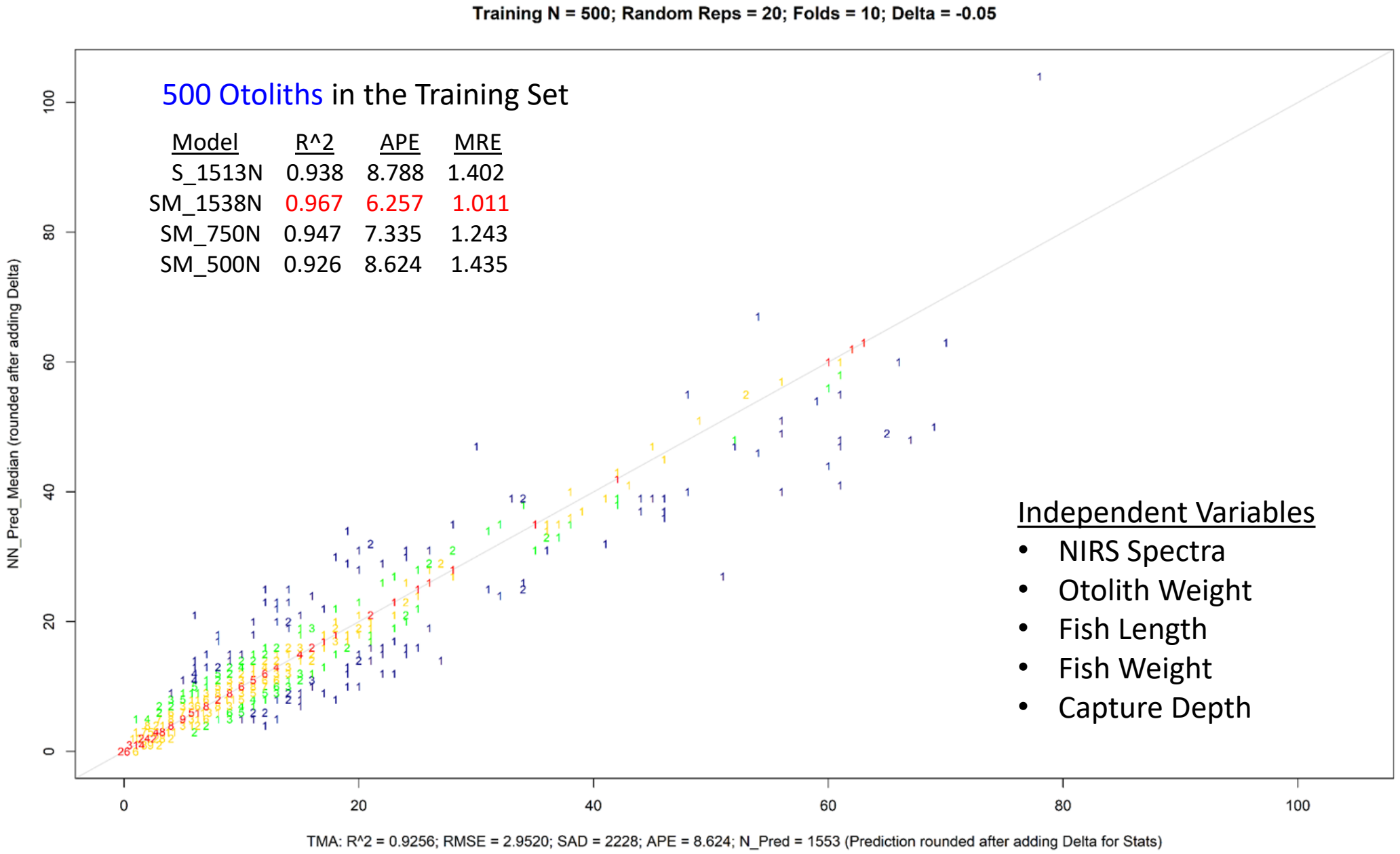


Credit: Henry Major

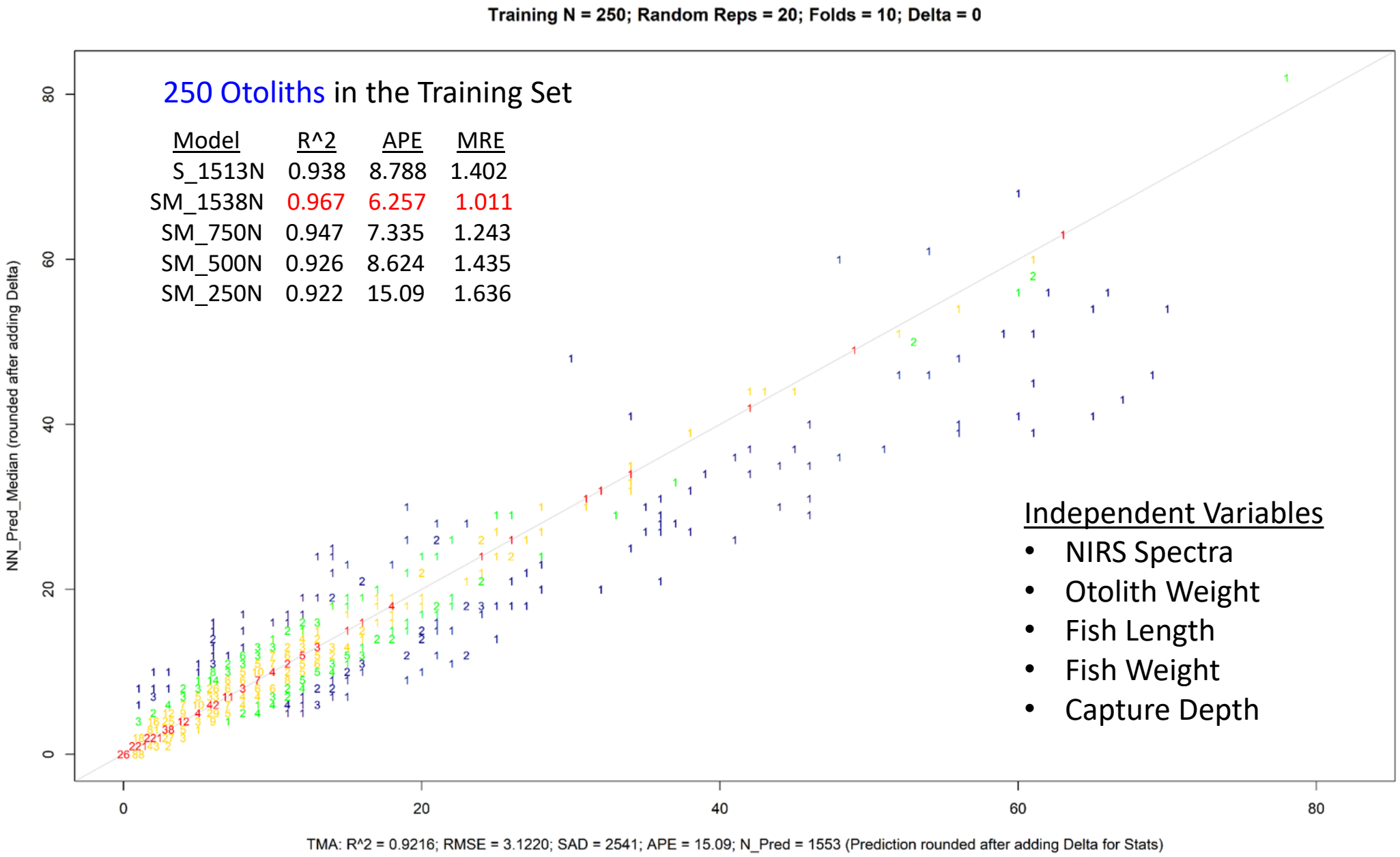
Sablefish 2022 WCGBTS Survey

500 and 250 Training N

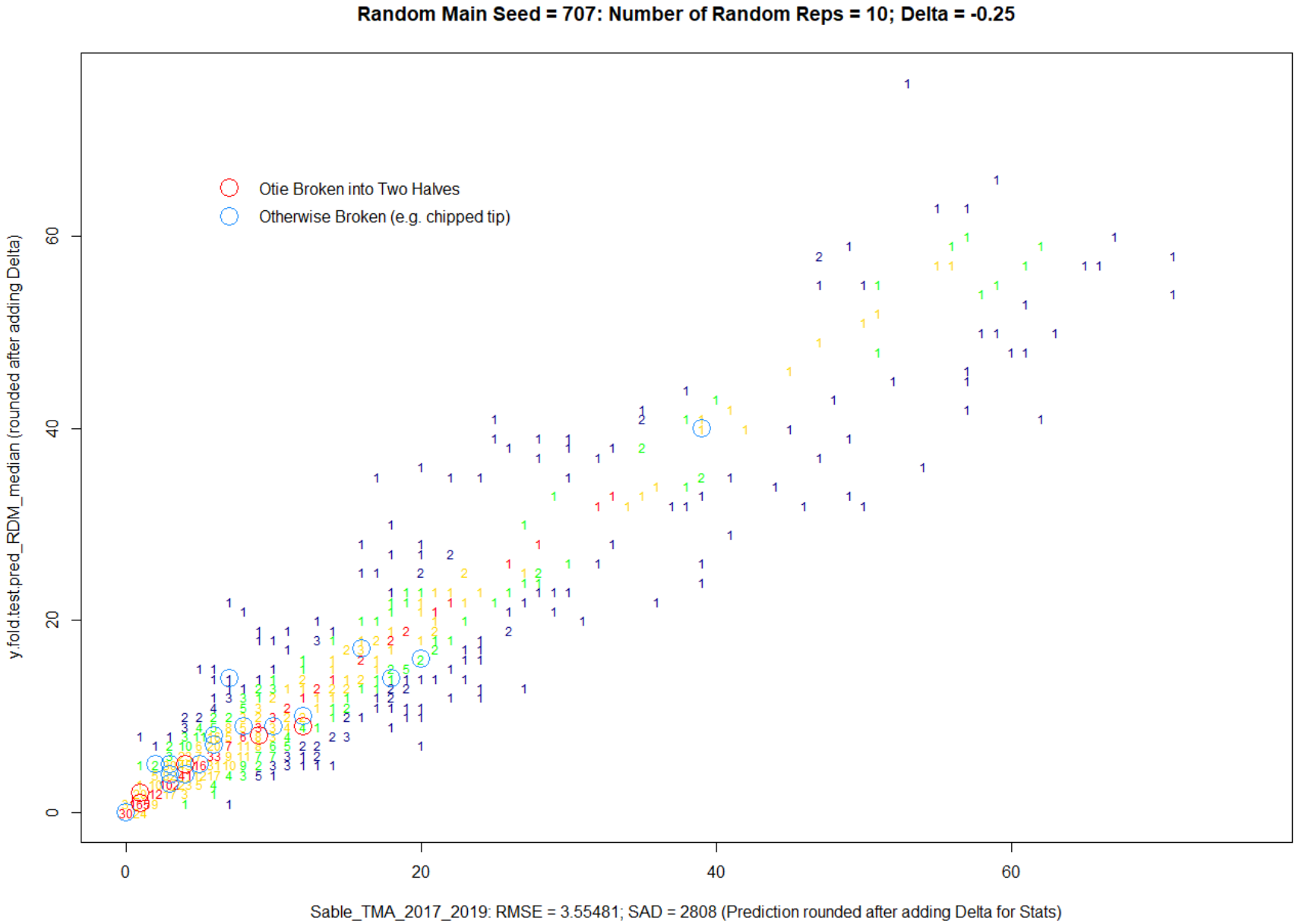
Sablefish 2022 WCGBTS Survey



Sablefish 2022 WCGBTS Survey



Broken Oties Intermixed with Unbroken Ones



Interactive Plotly Figures

- 2D plotly figures have zoom, interactive information for each line, and each item in the legend can removed or viewed separately (start with double-clicking on the last item in the legend).
- 3D plotly figures have 2 types of rotation: Orbital and Turntable (see the upper right legend).
- URL's for interactive plotly Hake figures
  - https://soundbirds.github.io/Hake_Spectra_plotly/
  - https://soundbirds.github.io/Hake_Spectra_plotly_3D/
- Plotly R Open Source Graphing Library
  - https://plotly.com/r/

Loss vs Accuracy

- **Loss** is defined as:
  - The difference between the predicted value by your model and the true value.

- **Accuracy** is defined as:
  - *Accuracy* = Number of correct predictions/Total number of predictions

On NN Models Getting Worse Over Time

- The central challenge in machine learning is that we must perform well on new, previously unseen inputs - not just those on which our model was trained. The ability to perform well on previously unobserved inputs is called generalization.
  - Page 110, Deep Learning, 2016.

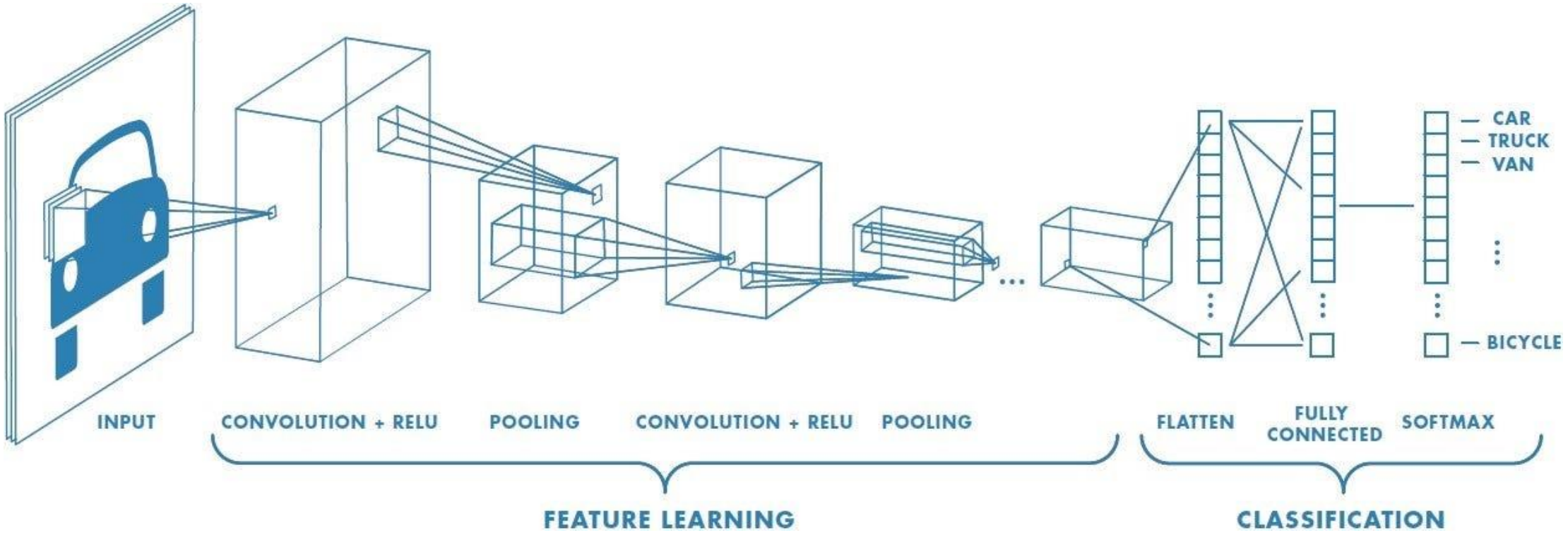
- As training progresses, the generalization error may decrease to a minimum and then increase again as the network adapts to idiosyncrasies of the training data.
  - Page 250, Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks, 1999.

Other Thoughts

- All combinations of 10 full k-fold models (1k+) was looked at, from 1 model only to all 10. Some of best combinations had only 4 models.
  - All combinations of 20 k-fold models is over a million.
- Neural networks (multi-layered perceptron) performed better than a random forest model in this reference:

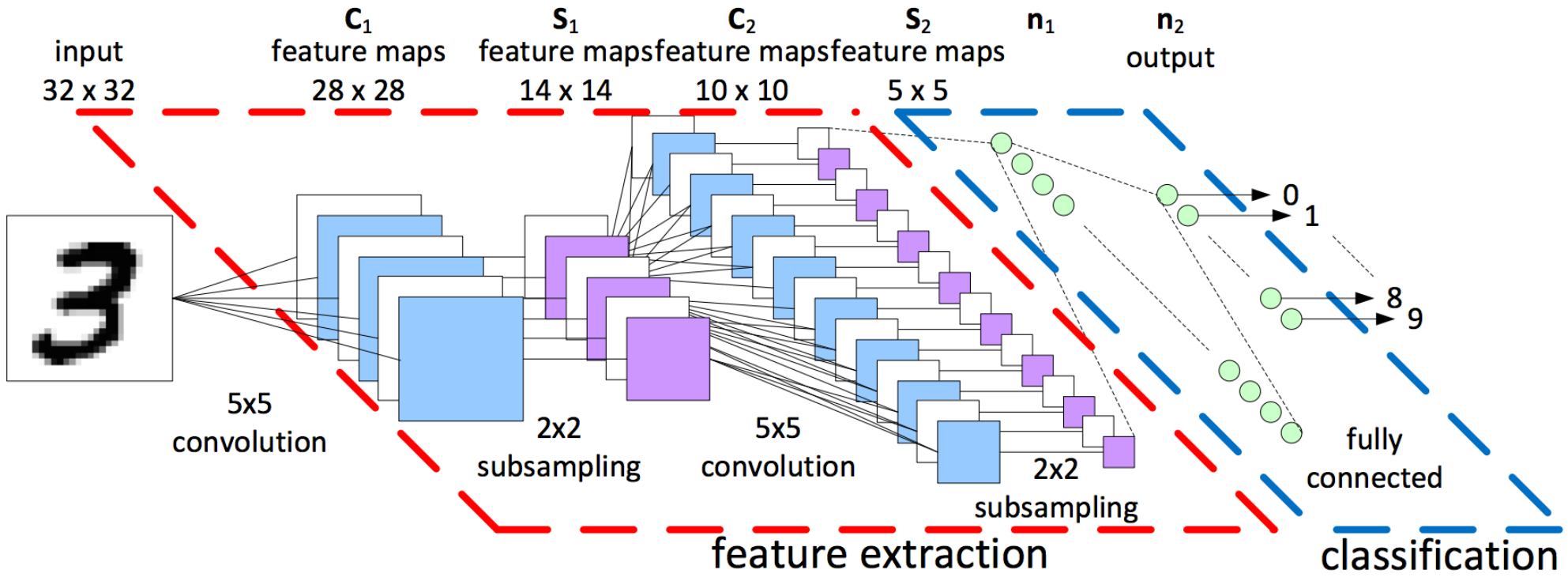
  Spectral deep learning for prediction and prospective validation of functional groups. Fine et al. Chem. Sci., 2020, 11, 4618.
- Many references show NN models outperforming PLS models.
- Future direction may include ensemble models
  - PLS, NN, and other models would be used in an ensemble approach

Convolutional Neural Network (2D example)



Basic CNN structure

- A typical CNN design begins with feature extraction and finishes with classification. Feature extraction is performed by alternating convolution layers with subsambling layers. Classification is performed with dense layers followed by a final softmax layer. For image classification, this architecture performs better than an entirely fully connected feed forward neural network. (https://www.kaggle.com/code/cdeotte/how-to-choose-cnn-architecture-mnist)



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".