Sentinel2 – Active Learning Regularization

Fall 2021 Co-op

Kate Harvey

CONTENTS

1 – Background **2**
1.1 SL2P Description 2
Table 1. Sentinel-2 bands used as SL2P input 2
1.1.1 – SL2P10_20m vs SL2P10_10m 2
1.2 Active Learning Regularization (ALR) 2
1.3 Regression Approaches 3
1.3.1 – Method 1 – SL2P10_10m 3
1.3.2 – Method 2 – LARS + Regression Tree 3
1.3.3 – Method 3 – LARS + Neural Network 3

2 – GitHub **4**
2.1 Github Files 4

3 – Sample Plots For Sites **5**
3.1 Geraldton, ON 5
3.2 Fox Creek, AB 11
3.3 Kouchibouguac, NB 16
3.4 Ottawa Region Test Image 21

Appendix A – Classification and Regression in Earth Engine **26**
A.1 Reading Resources 26
A.2 Classification 26
Supervised Learning 26
Classification Workflow 26
Random Forest vs CART 27
Implementation/Testing 27

I – BACKGROUND

1.1 SL2P Description

The SL2P algorithm uses a two-layer neural network to predict output values for level 2 biophysical variables (LAI, fAPAR, and fCOVER) from Sentinel-2. Currently, it uses 8 input bands: B3, B4, B5, B6, B7, B8A, B11, and B12 (bands are summarized in detail in Table 1 below).

Table 1. Sentinel-2 bands used as SL2P input

Acronym	Central (nm)	Width (nm)	Spatial resolution (m)
B3	560	35	10
B4	665	30	10
B5	705	15	20
B6	740	15	20
B7	783	20	20
B8a	865	20	20
B11	1610	90	20
B12	2190	180	20

1.1.1 – SL2P10_20m vs SL2P10_10m

The purpose of SL2P10 is to generate predictions as with SL2P, using only 10 m spectral bands (B2, B3, B4, and B8). Predictions generated using this algorithm will provide data at a higher resolution that may be required for certain applications.

There are two variations of SL2P10, which are named as **SL2P10_20m** and **SL2P10_10m**, respectively. The first approach takes the 10 m bands and averages the pixel values to a resolution of 20 m before applying the network. The second approach applies the network over the image before reducing the resolution to 20 m. **SL2P10_10m** generates an output map at 10 m resolution.

1.2 Active Learning Regularization (ALR)

The active learning component of this project involves taking the images created using the SL2P algorithm and discarding the 20 m bands, and then using a neural network in an attempt to increase the accuracy of the predictions. It is hypothesized that local calibration of the network will help to generate better predictions, despite reducing the number of input bands (since 20 m bands are discarded).

1.3 Regression Approaches

This task is concerned with comparing three different prediction methods of the test images to determine the preferred treatment.

1.3.1 – Method 1 – SL2P10_10m

This is the simplest of the three approaches, using only the SL2P10_10m algorithm to predict the biophysical variables. This will be applied to the raw input image (instead of SL2P) to create a prediction.

1.3.2 – Method 2 – LARS + Regression Tree

This approach will make use of the feature selection algorithm (as in ALR_client_side_10m.ipynb) combined with the smileCART regression function in GEE. The input image will be the output of SL2P.

1.3.3 – Method 3 – LARS + Neural Network

This approach has been implemented in the ALR_client_side_10m notebook (written by Hemit). This method also uses LARS for feature selection, and the selected features are then used to train a tensorflow (keras) model. As with the previous method (LARS + regression tree), this procedure will be applied to the output image of SL2P. This is the method described as Active Learning Regularization.

Fig. 1.3 (a) – Flow chart showing processing steps resulting in each of the outputs described

The results of all three methods will be applied to the selected images from the three test sites and compared. The R^2, RMSE, and bias will provide an indication of the preferred method.

2 – GITHUB

2.1 Github Files

Github can be accessed here: https://github.com/kateharvey/Sentinel2_ALR

- **SL2P10_control** script with modules:
    - ee_functions.py
    - feature_collections.py
    - image_bands.py
    - wrapper_nets.py
- **SL2P10_notebook** (complete version of **SL2P10_control** with all functions)
- **ALR_client_side_10m** script with module:
    - ALR_functions.py
- **ALR_client_side_notebook** (complete version of **ALR_client_side_10m** with all functions)
- py_server_side (server side script)
- **3_regression_approaches**, a combined script with code to generate all assets required for Methods 1, 2, and 3 (requires all modules listed above)

3 – SAMPLE PLOTS FOR SITES

The plots in the following section are intended to provide a visual explanation of the processing steps involved in obtaining the final output.

3.1 Geraldton, ON

Image ID: COPERNICUS/S2_SR/20200811T164849_20200811T165525_T16UEA

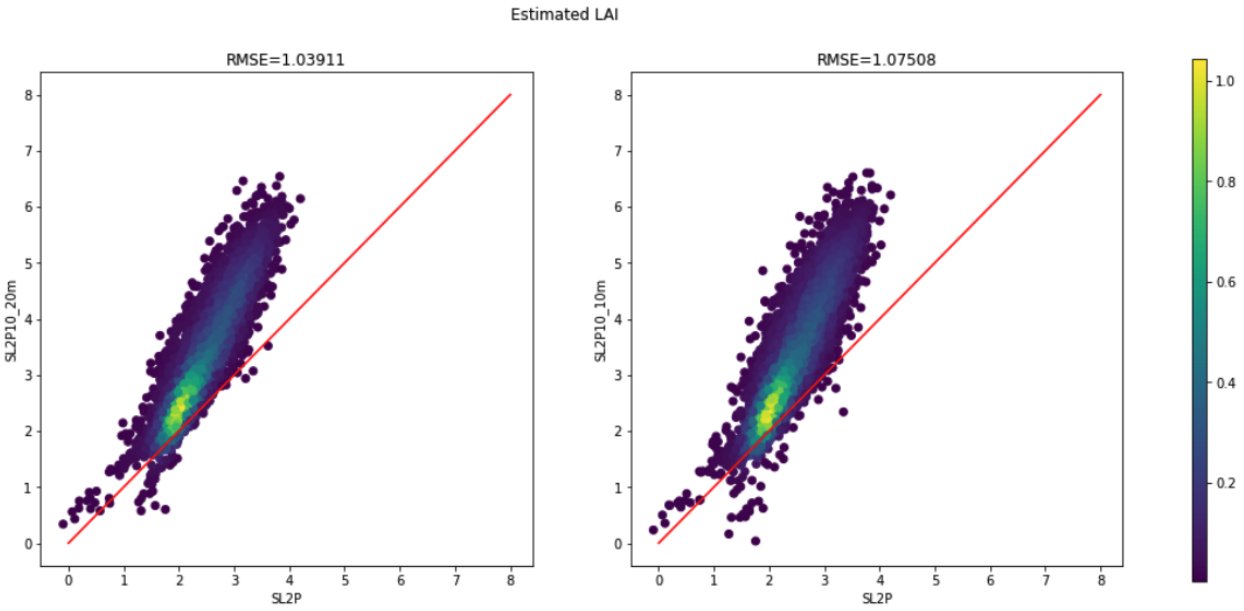

Fig. 3.1.a LAI predictions: SL2P vs. SL2P10 using 20m (L) and 10m (R) network

The plots in Fig. 3.1.a (above) compare the estimated LAI values created using the SL2P algorithm, versus the SL2P10_20m and SL2P10_10m networks. At this step, two versions of the plot are shown (in 20m and 10m resolutions). This is a preventative measure to confirm that there are no major discrepancies between the two products before proceeding – the product that we are interested in is represented by the 10m plot (on the right side). The predictions show a positive bias in both cases. The plots are created from a ~5000 random sample array (limited by user memory allowance in GEE).

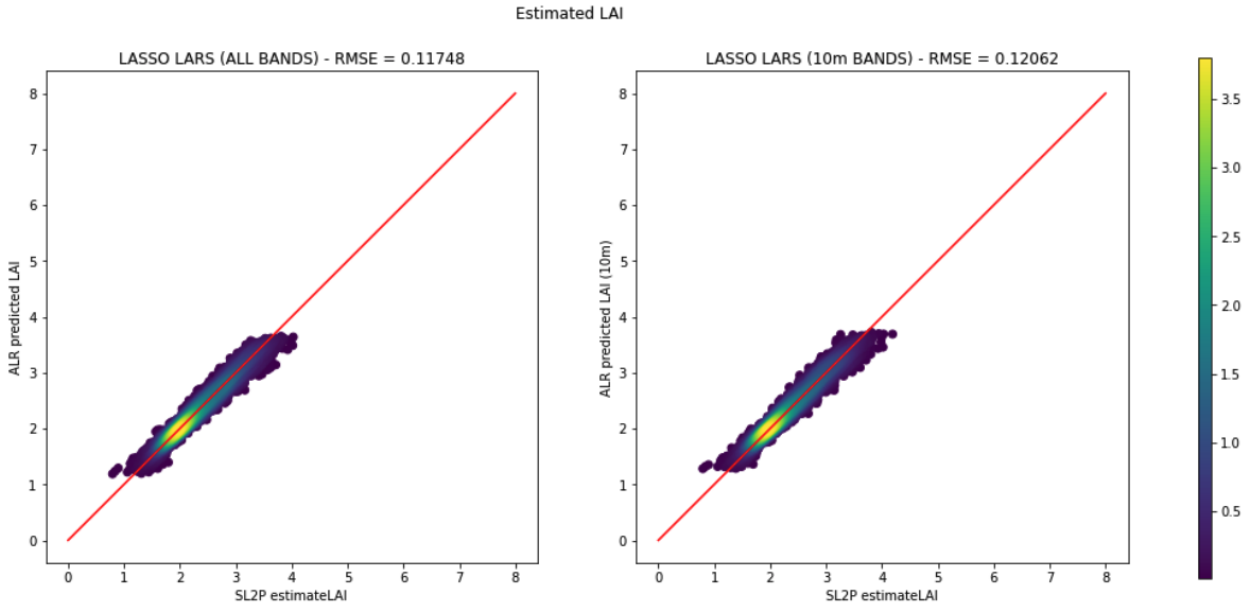

Fig. 3.1.b LAI predictions: SL2P vs. ALR using 20m (L) and 10m (R) bands

The plots in Fig. 3.1.b (above) compare the estimated LAI values from SL2P with the ALR-predicted values, in both 20 m and 10 m resolutions. The ALR predictions (y-axis values) are generated by using LARS feature selection on the SL2P output image and applying a tensorflow neural net to the selected features. These plots are based on a sample size of ~10000 random samples.

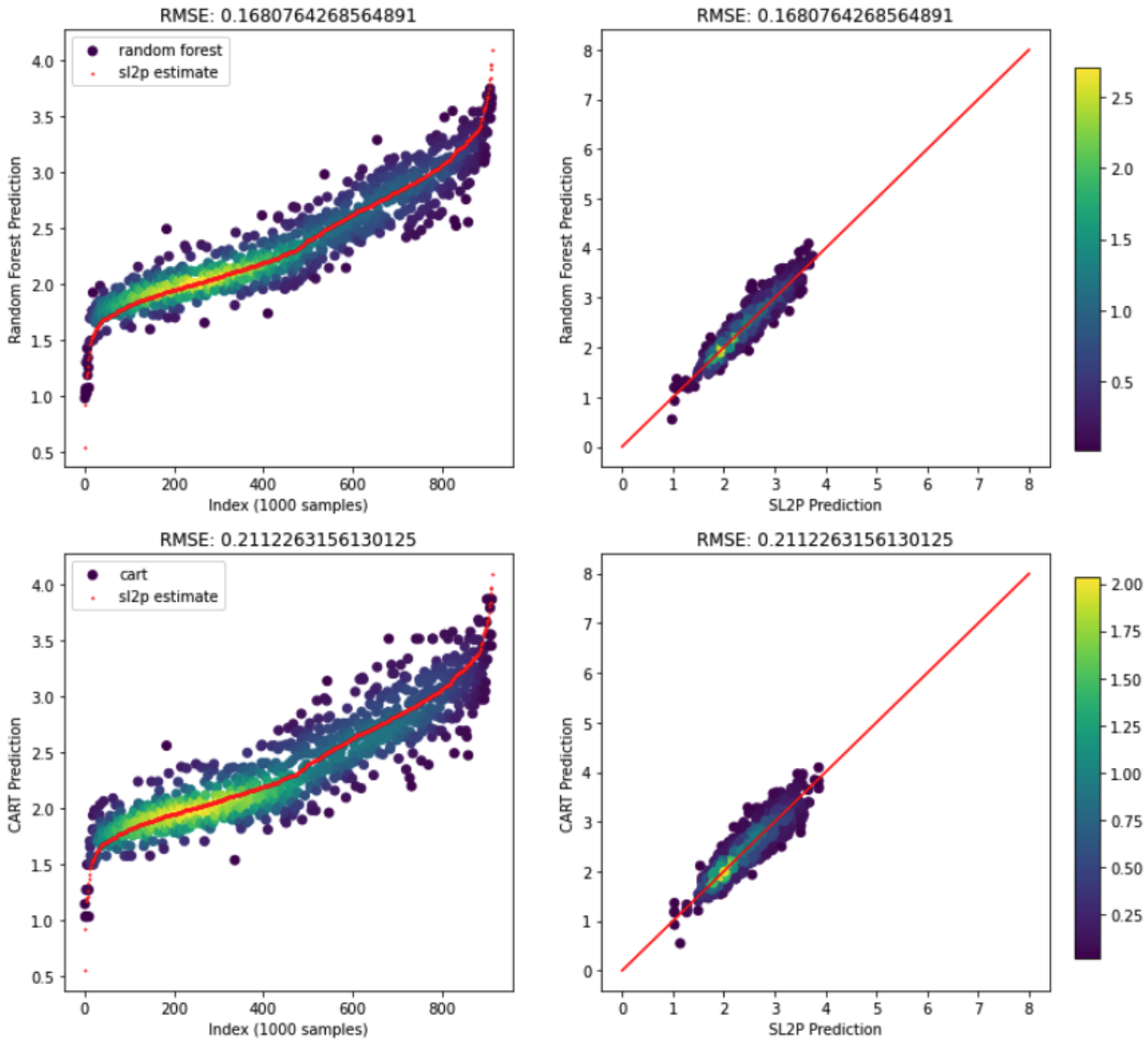

Fig. 3.1.c LAI predictions: SL2P vs. CART regression using 20m (L) and 10m (R) bands

The plots in Fig. 3.1.c (above) compare the results from applying ALR followed by a regression tree to the original SL2P prediction.

On the left, the Random Forest and CART predictions were sorted in ascending order in order to be plotted (the shape of the trend is not important). The red points in this plot represent the SL2P estimate, which is compared with the Random Forest and Cart regressions, respectively. For this particular image, the CART regressor matches the SL2P prediction more closely with smaller root mean squared error.

On the right, the Random Forest and CART predictions were plotted against the SL2P prediction similarly to Fig. 1.1 and 1.2. There does not appear to be a significant bias in either case, and again the CART regressor performs better.

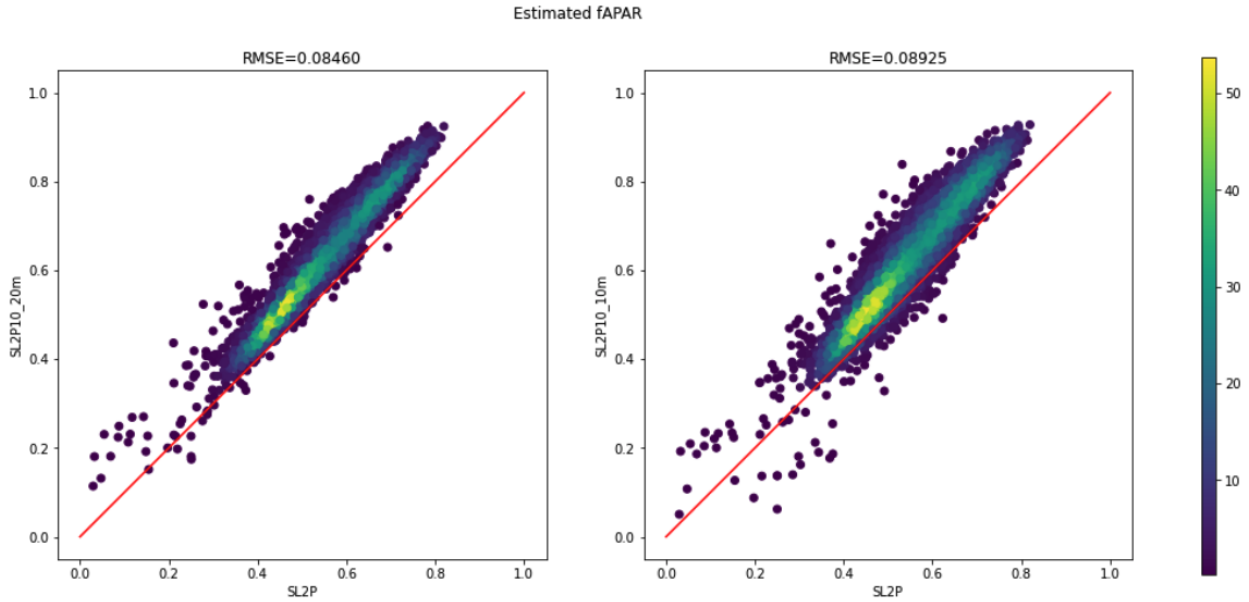

Fig. 3.1.d fAPAR predictions: SL2P vs. SL2P10 using 20m (L) and 10m (R) network

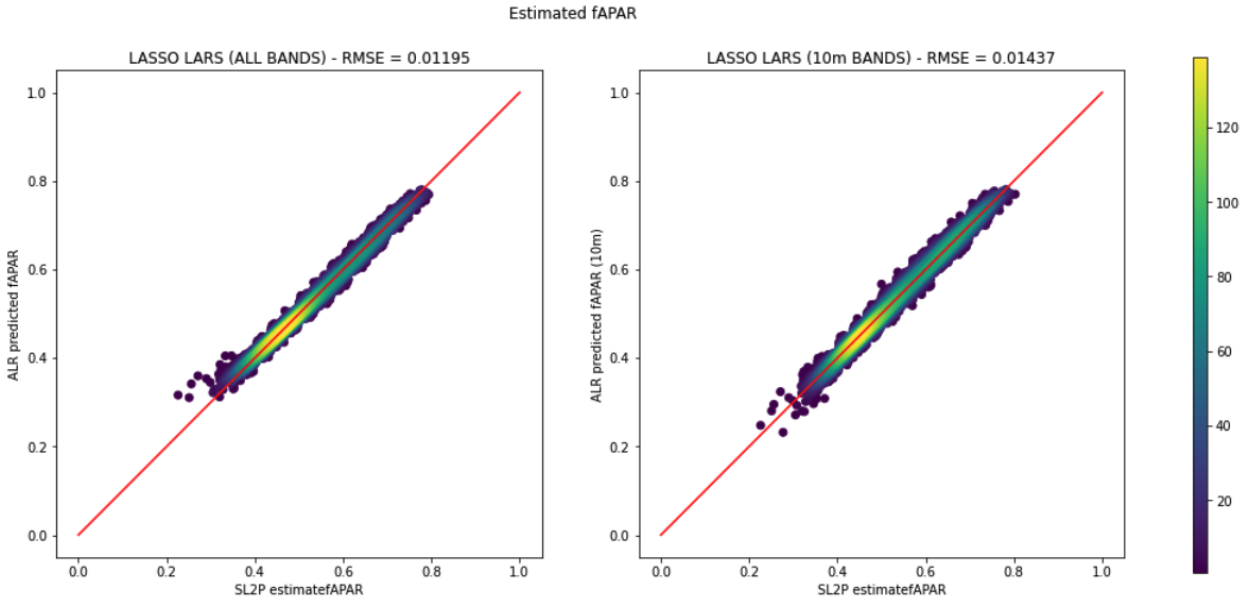

Fig. 3.1.e fAPAR predictions: SL2P vs. ALR using 20m (L) and 10m (R) bands

Fig. 3.1.f fAPAR predictions: SL2P vs. CART regression using 20m (L) and 10m (R) bands

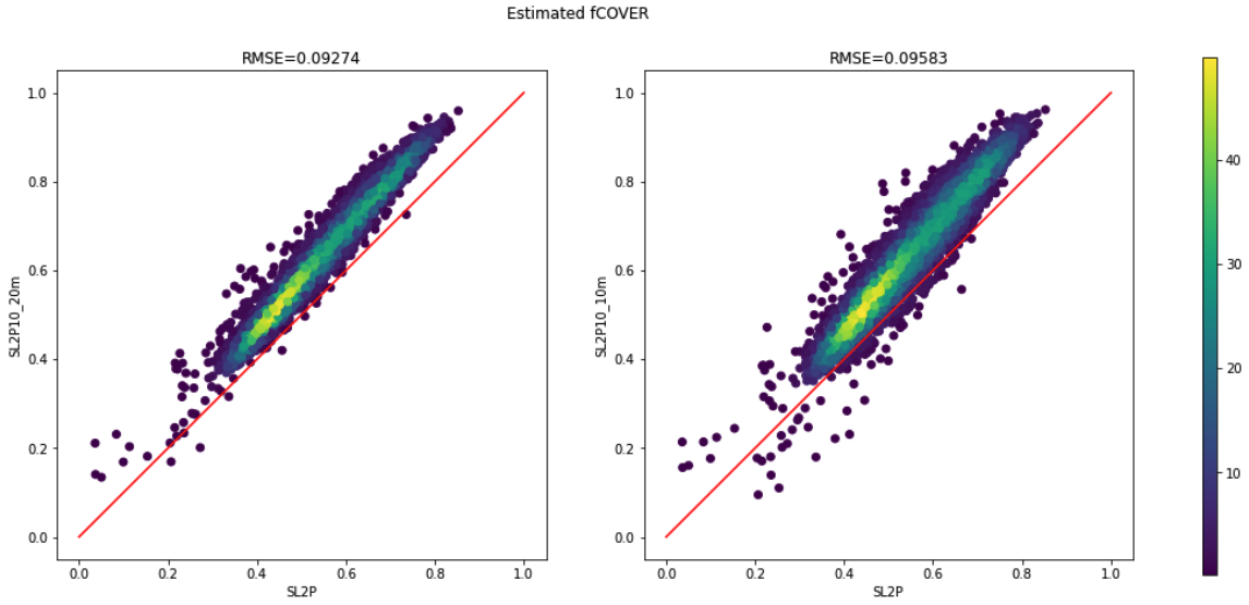

Fig. 3.1.g fCOVER predictions: SL2P vs. SL2P10 using 20m (L) and 10m (R) network

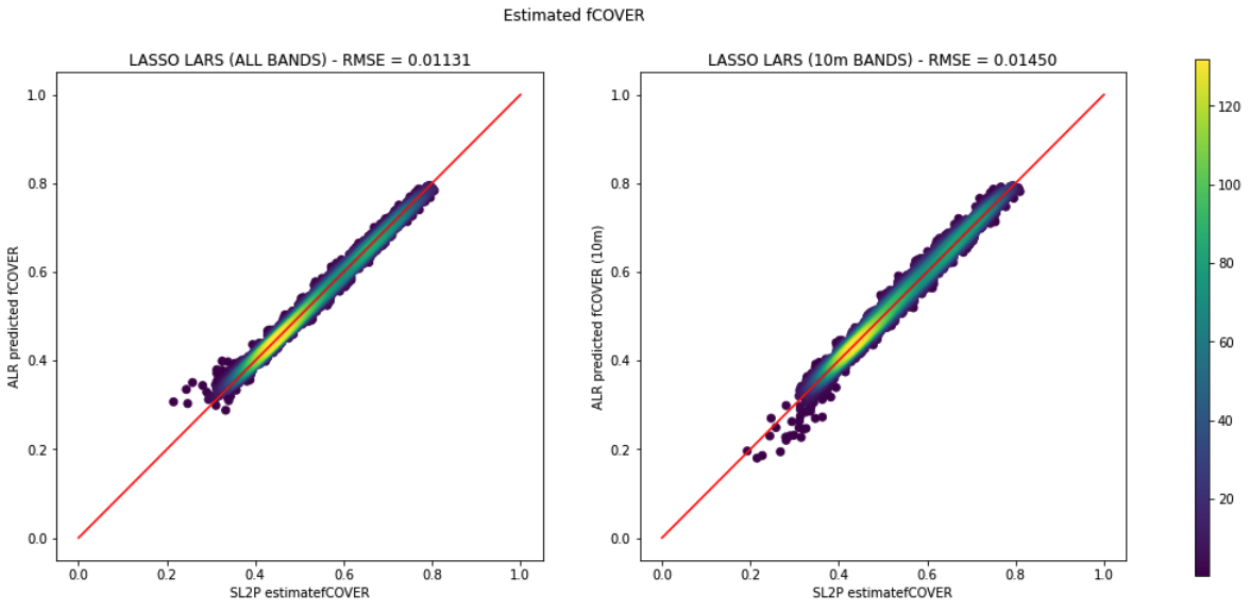

Fig. 3.1.h fCOVER predictions: SL2P vs. ALR using 20m (L) and 10m (R) bands

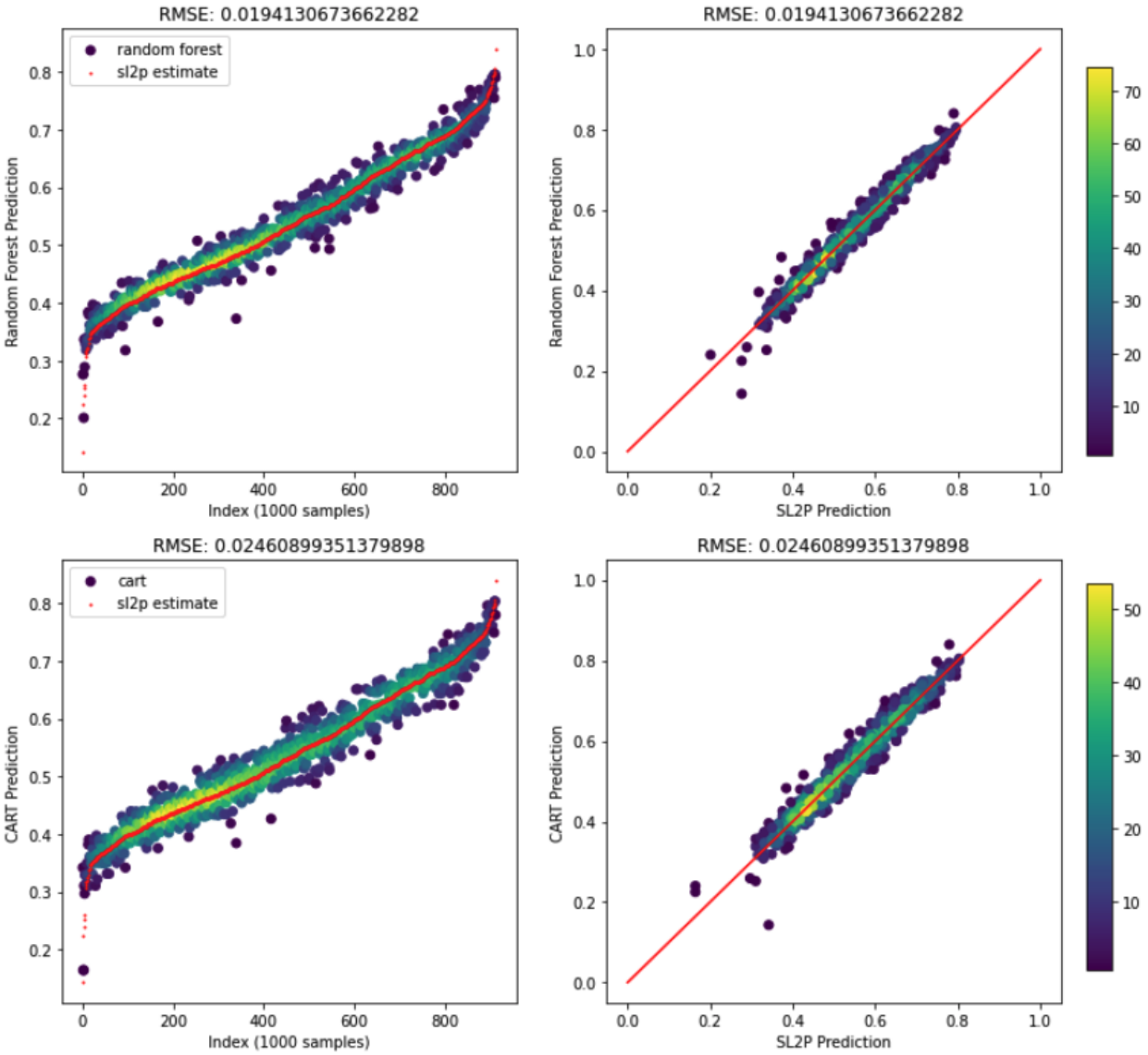

Fig. 3.1.i fCOVER predictions: SL2P vs. CART regression using 20m (L) and 10m (R) bands

3.2 Fox Creek, AB

Image ID: COPERNICUS/S2_SR/20210825T185919_20210825T190431_T11UNA

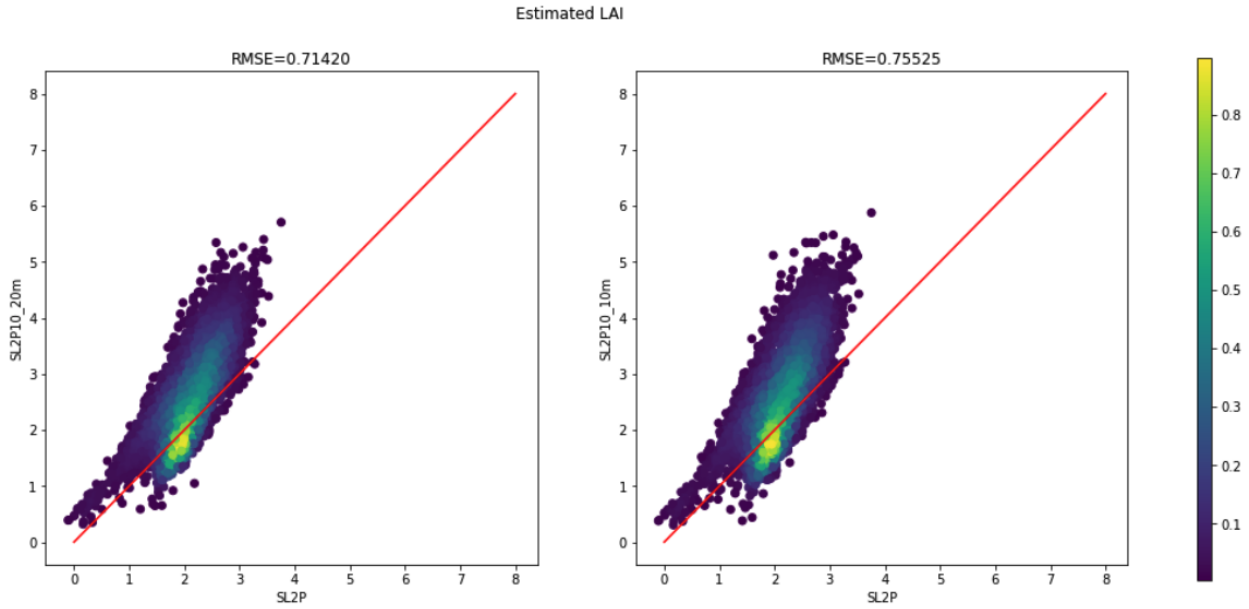

Fig. 3.2.a LAI predictions: SL2P vs. SL2P10 using 20m (L) and 10m (R) network

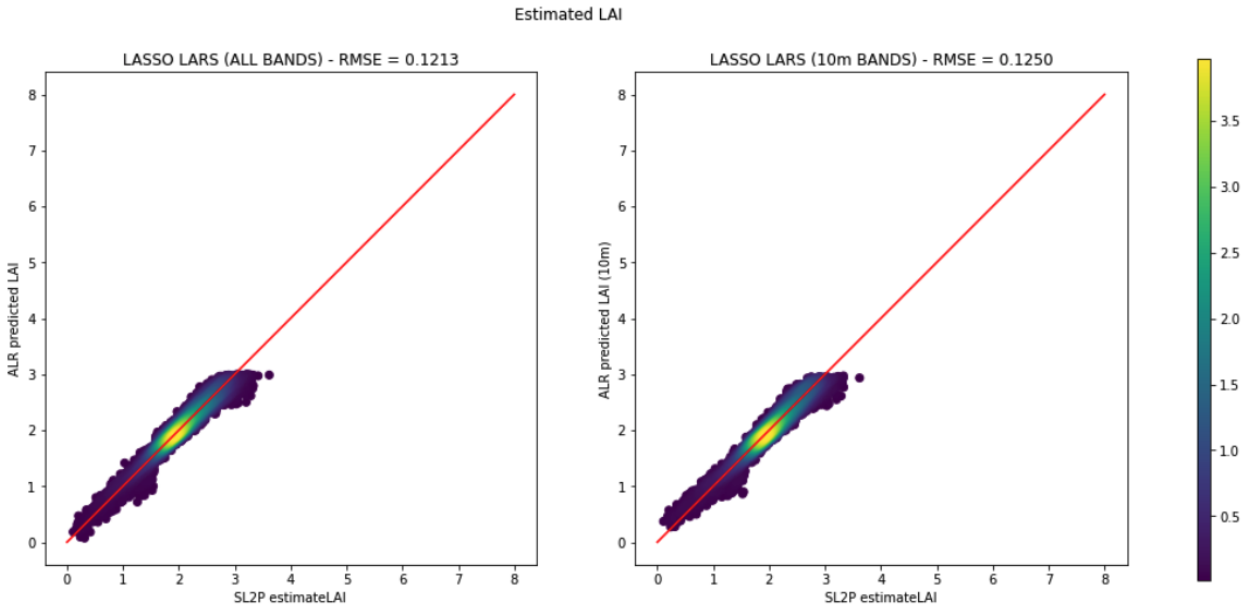

Fig. 3.2.b LAI predictions: SL2P vs. ALR using 20m (L) and 10m (R) bands

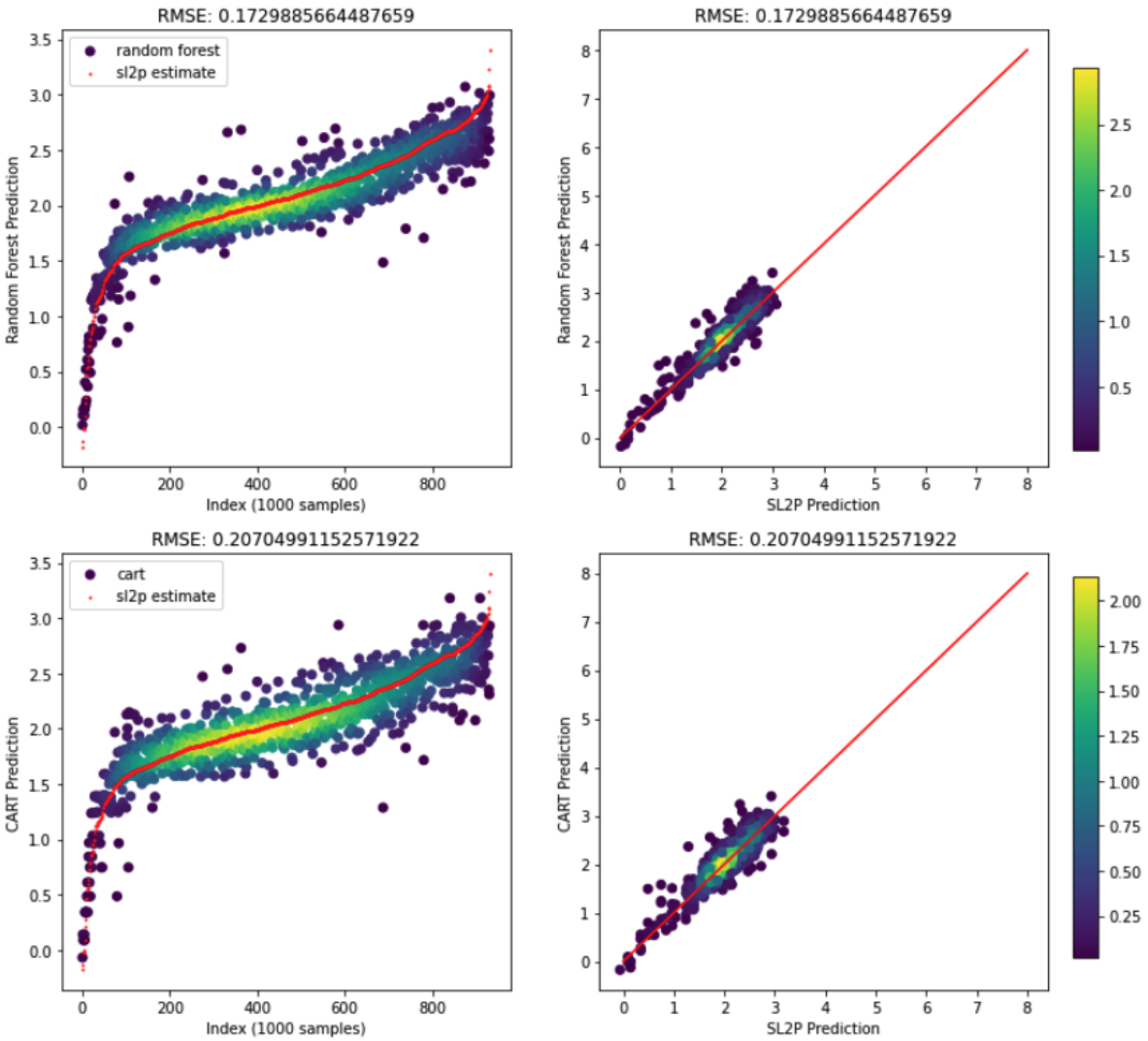

Fig. 3.2.c LAI predictions: SL2P vs. CART regression using 20m (L) and 10m (R) bands

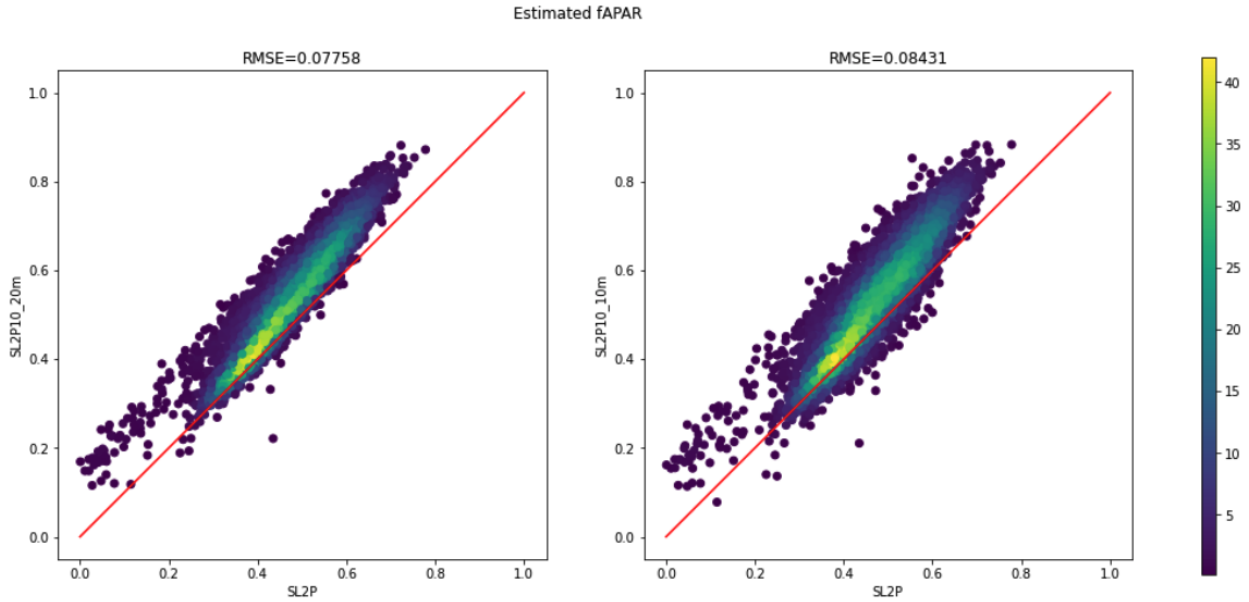

Fig. 3.2.d fAPAR predictions: SL2P vs. SL2P10 using 20m (L) and 10m (R) network

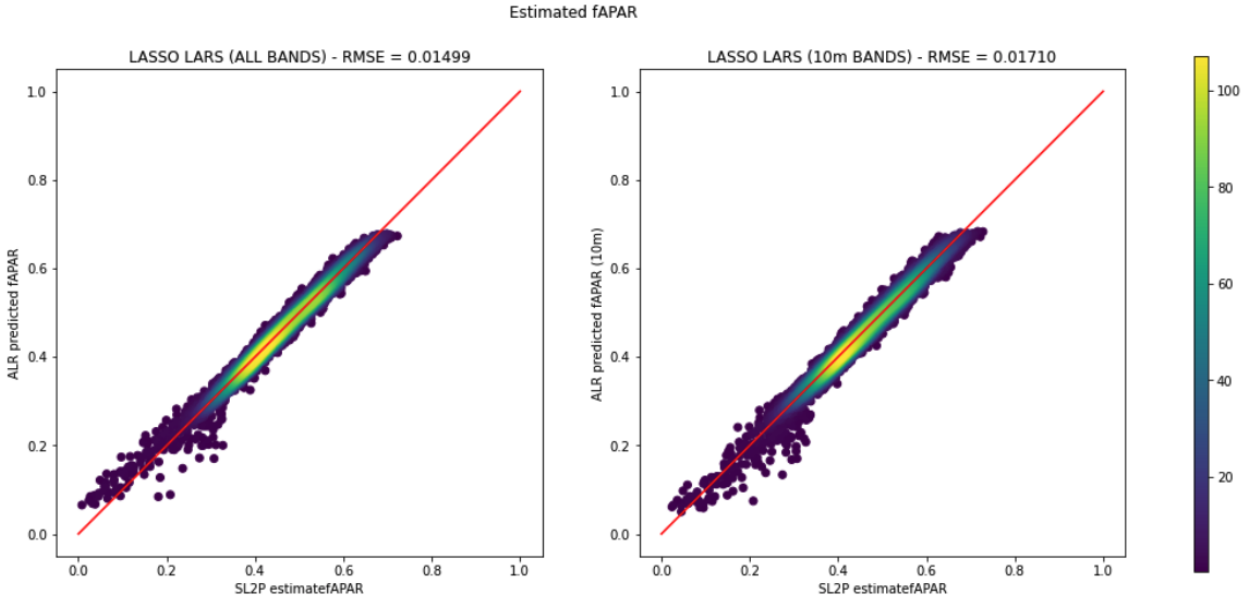

Fig. 3.2.e fAPAR predictions: SL2P vs. ALR using 20m (L) and 10m (R) bands

Fig. 3.2.f fAPAR predictions: SL2P vs. CART regression using 20m (L) and 10m (R) bands

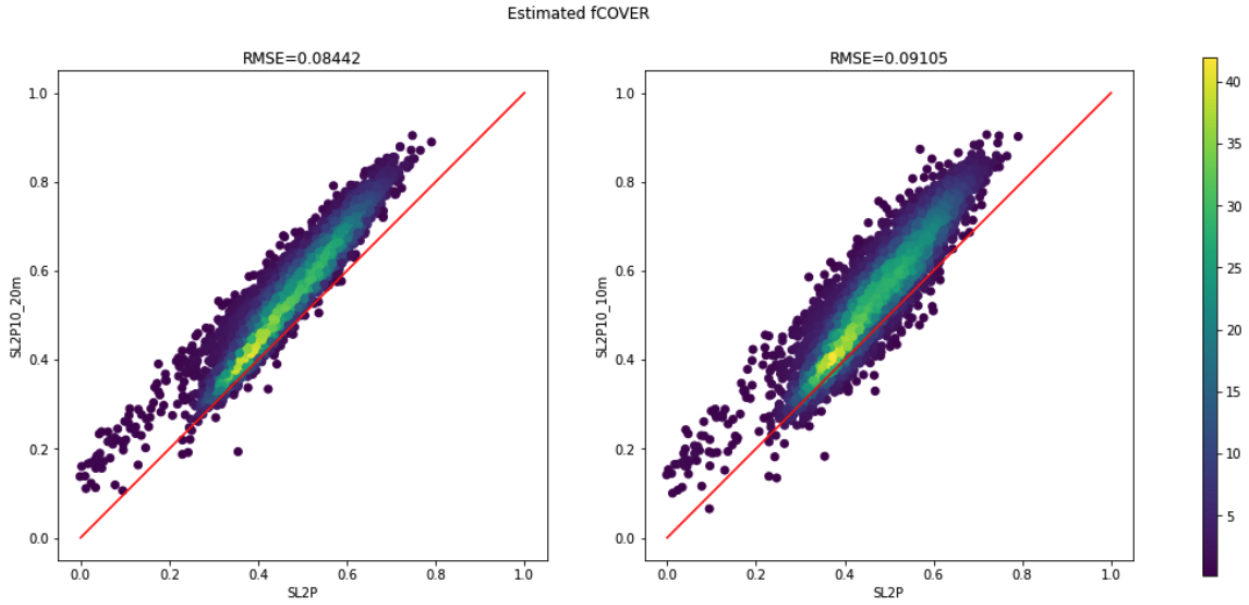

Fig. 3.2.g fCOVER predictions: SL2P vs. SL2P10 using 20m (L) and 10m (R) network

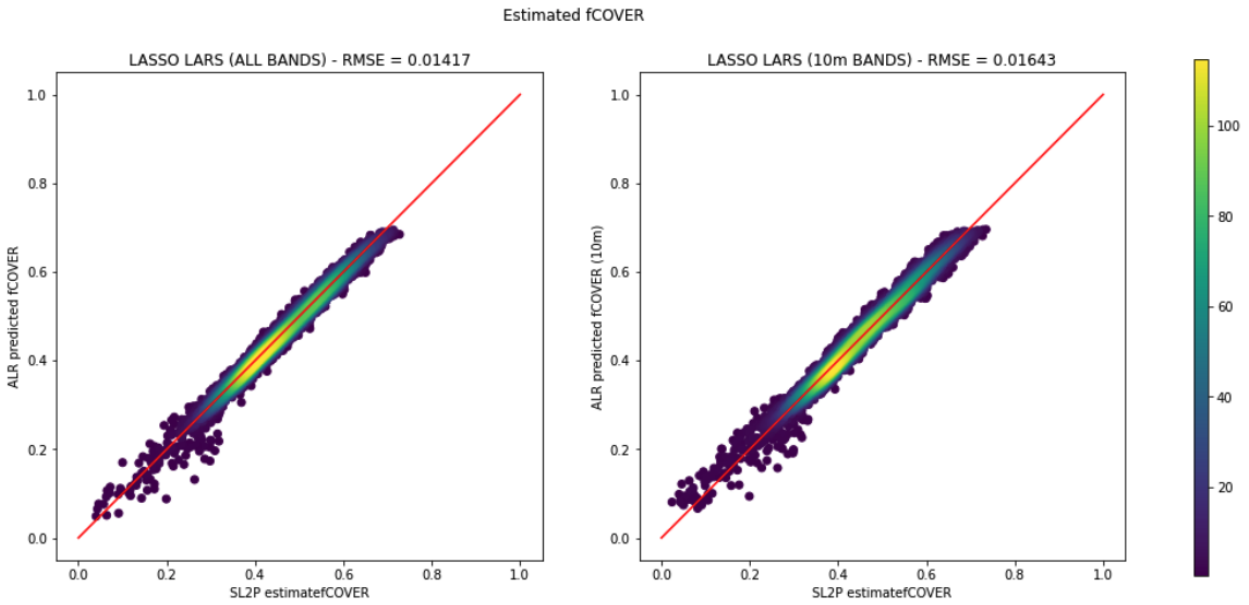

Fig. 3.2.h fCOVER predictions: SL2P vs. ALR using 20m (L) and 10m (R) bands

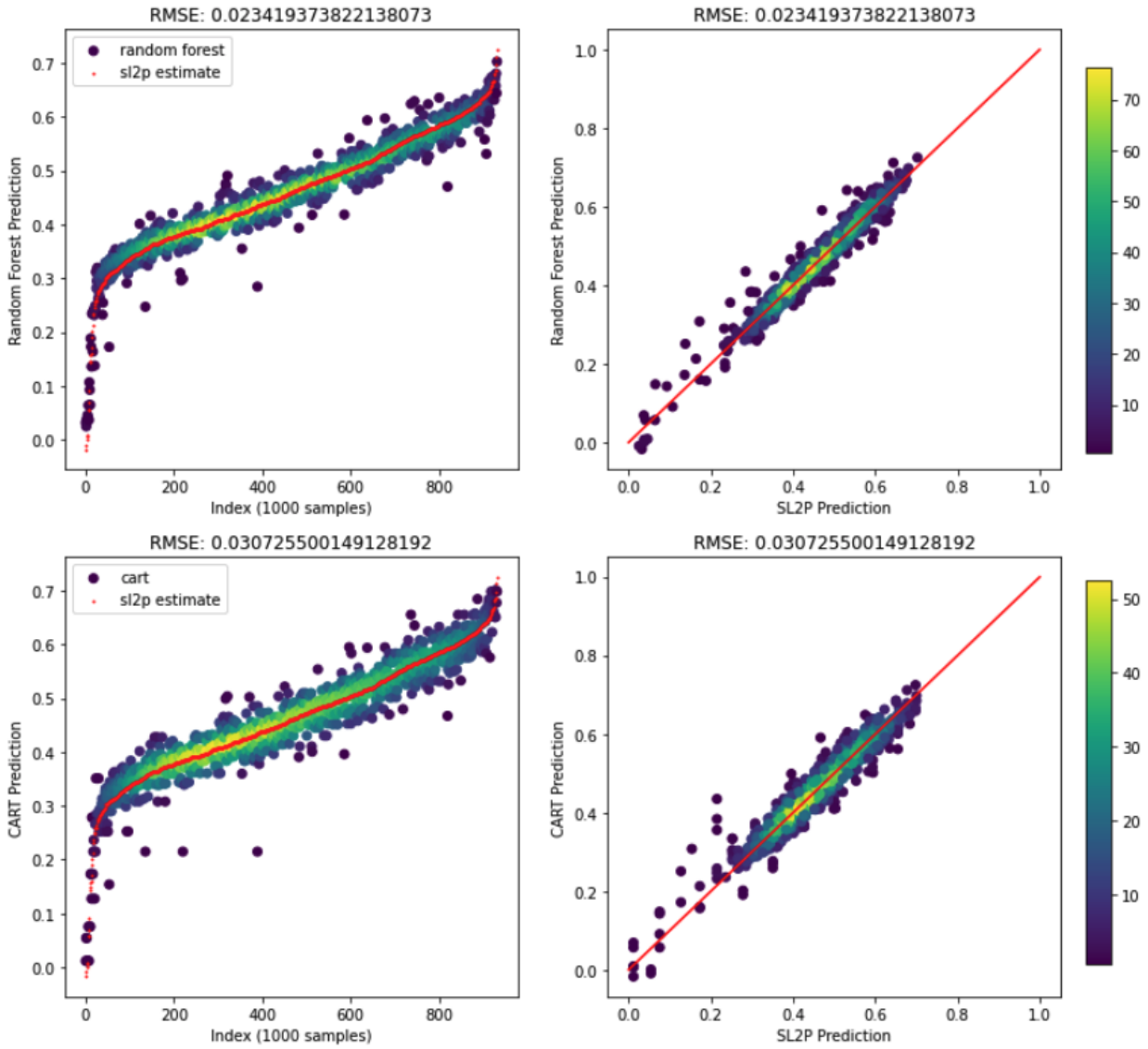

Fig. 3.2.i fCOVER predictions: SL2P vs. CART regression using 20m (L) and 10m (R) bands

3.3 Kouchibouguac, NB

Image ID: COPERNICUS/S2_SR/20200905T151701_20200905T151829_T20TLS

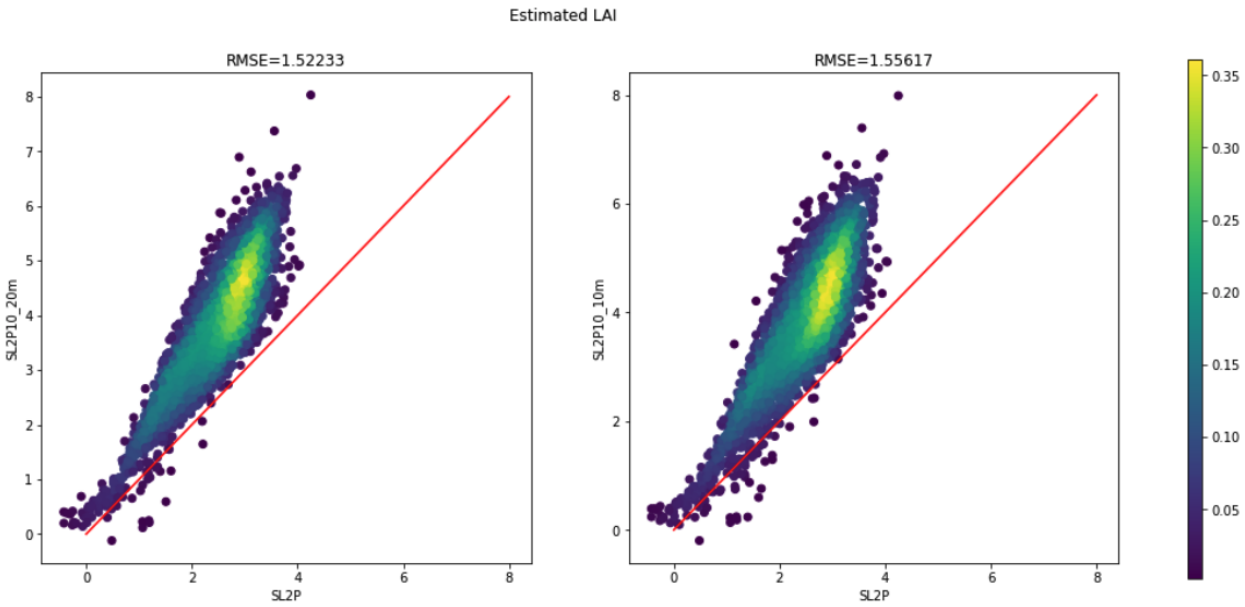

Fig. 3.3.a LAI predictions: SL2P vs. SL2P10 using 20m (L) and 10m (R) network

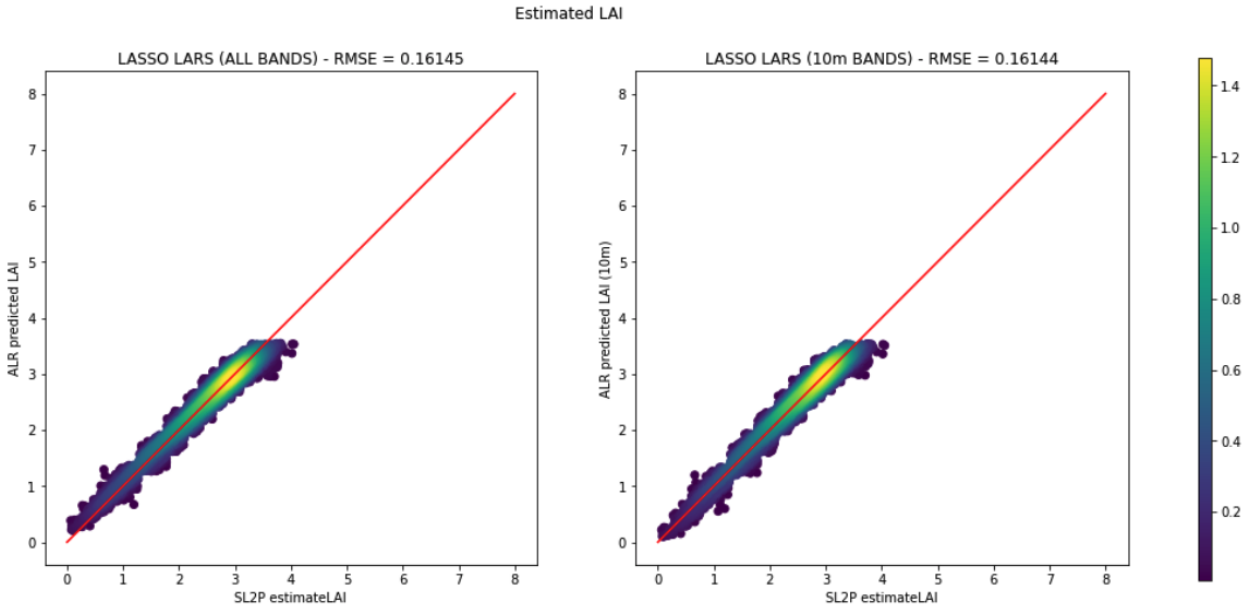

Fig. 3.3.b LAI predictions: SL2P vs. ALR using 20m (L) and 10m (R) bands

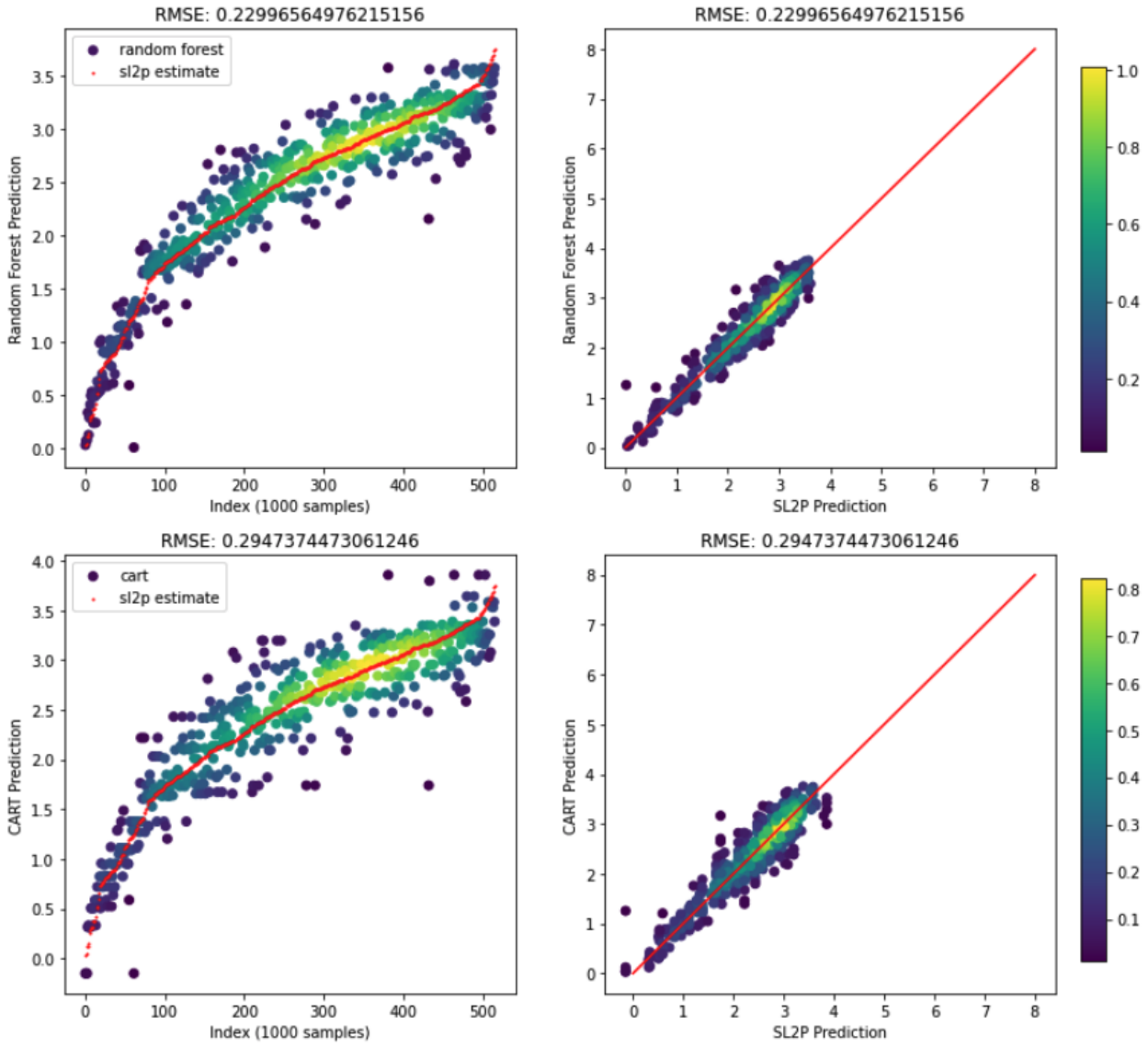

Fig. 3.3.c LAI predictions: SL2P vs. CART regression using 20m (L) and 10m (R) bands

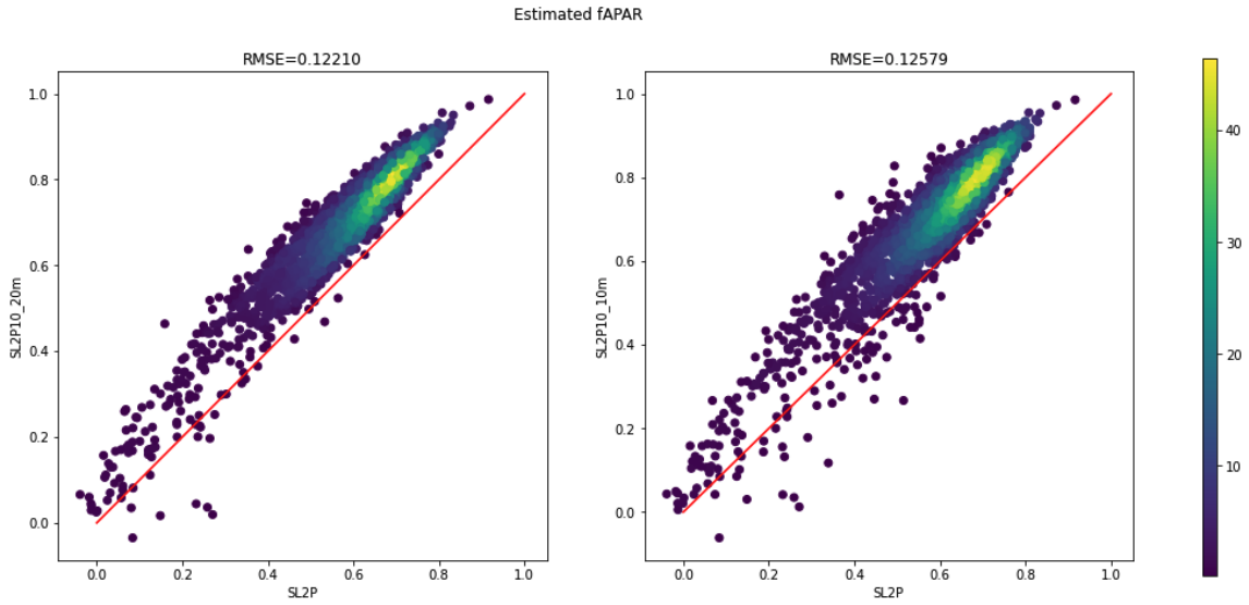

Fig. 3.3.d fAPAR predictions: SL2P vs. SL2P10 using 20m (L) and 10m (R) network

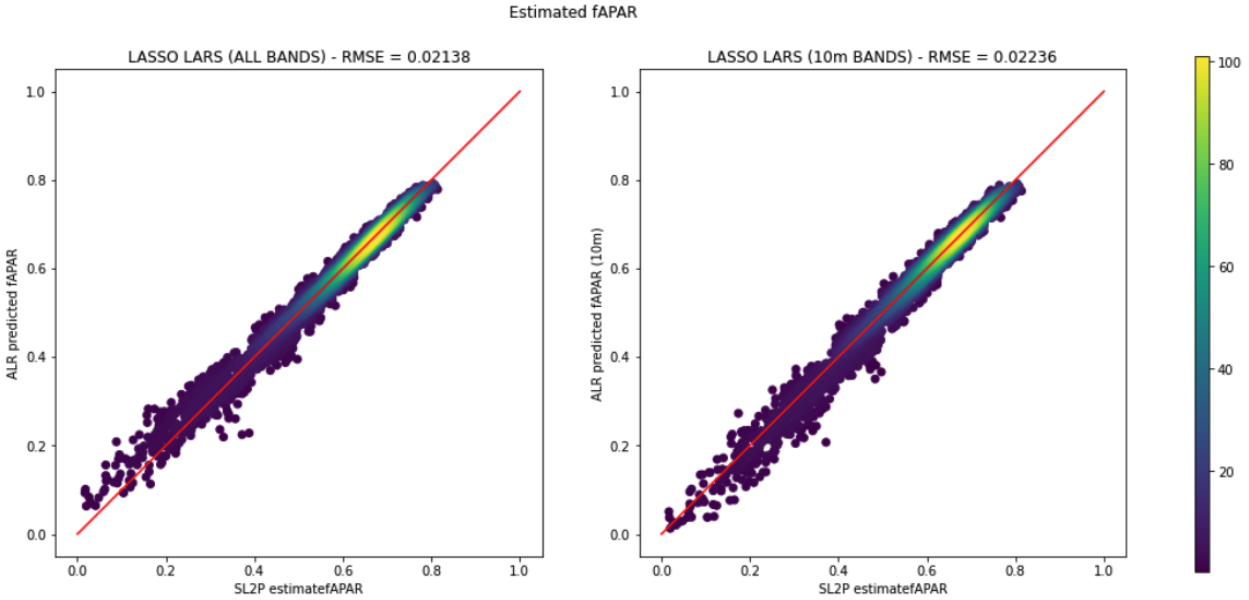

Fig. 3.3.e fAPAR predictions: SL2P vs. ALR using 20m (L) and 10m (R) bands

Fig. 3.3.f fAPAR predictions: SL2P vs. CART regression using 20m (L) and 10m (R) bands

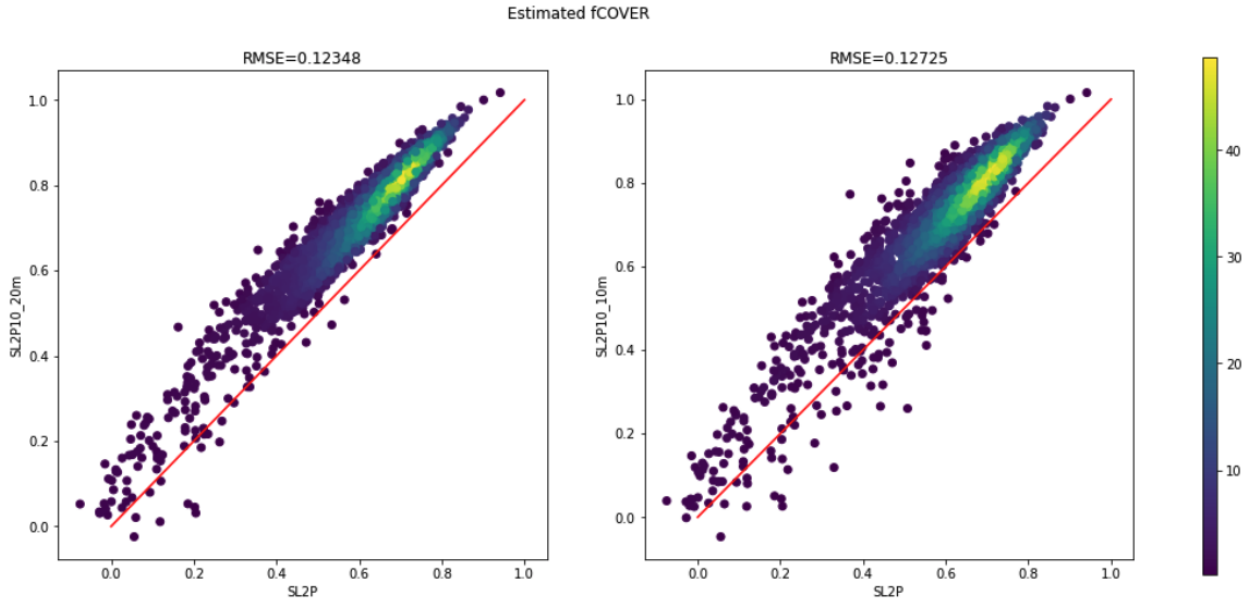

Fig. 3.3.g fCOVER predictions: SL2P vs. SL2P10 using 20m (L) and 10m (R) network

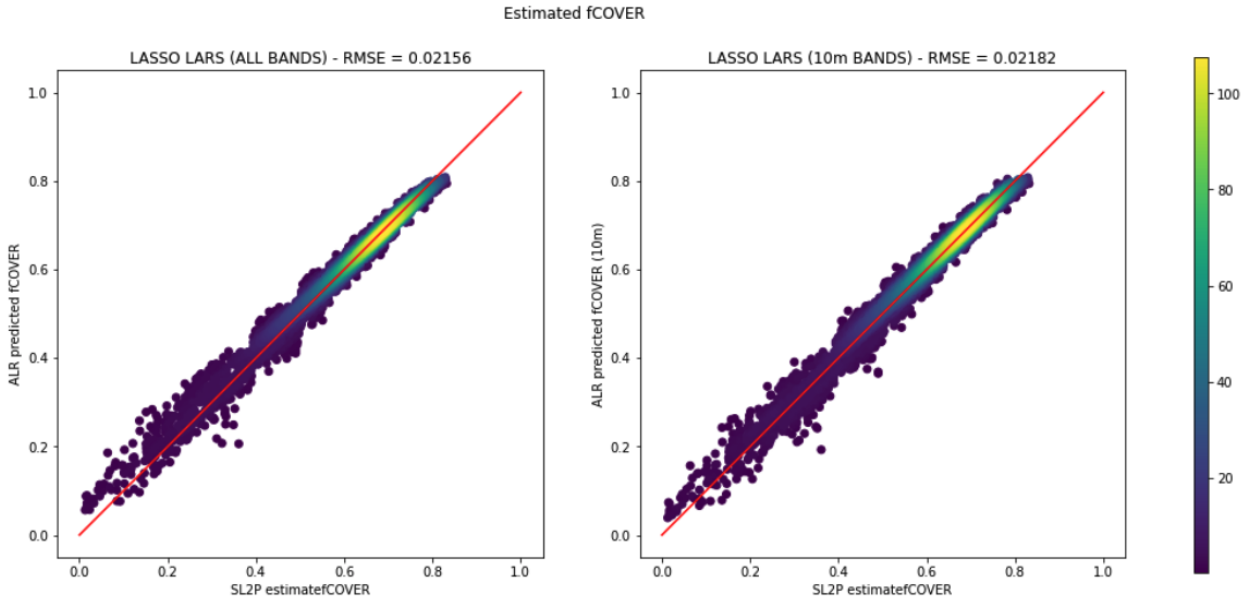

Fig. 3.3.h fCOVER predictions: SL2P vs. ALR using 20m (L) and 10m (R) bands

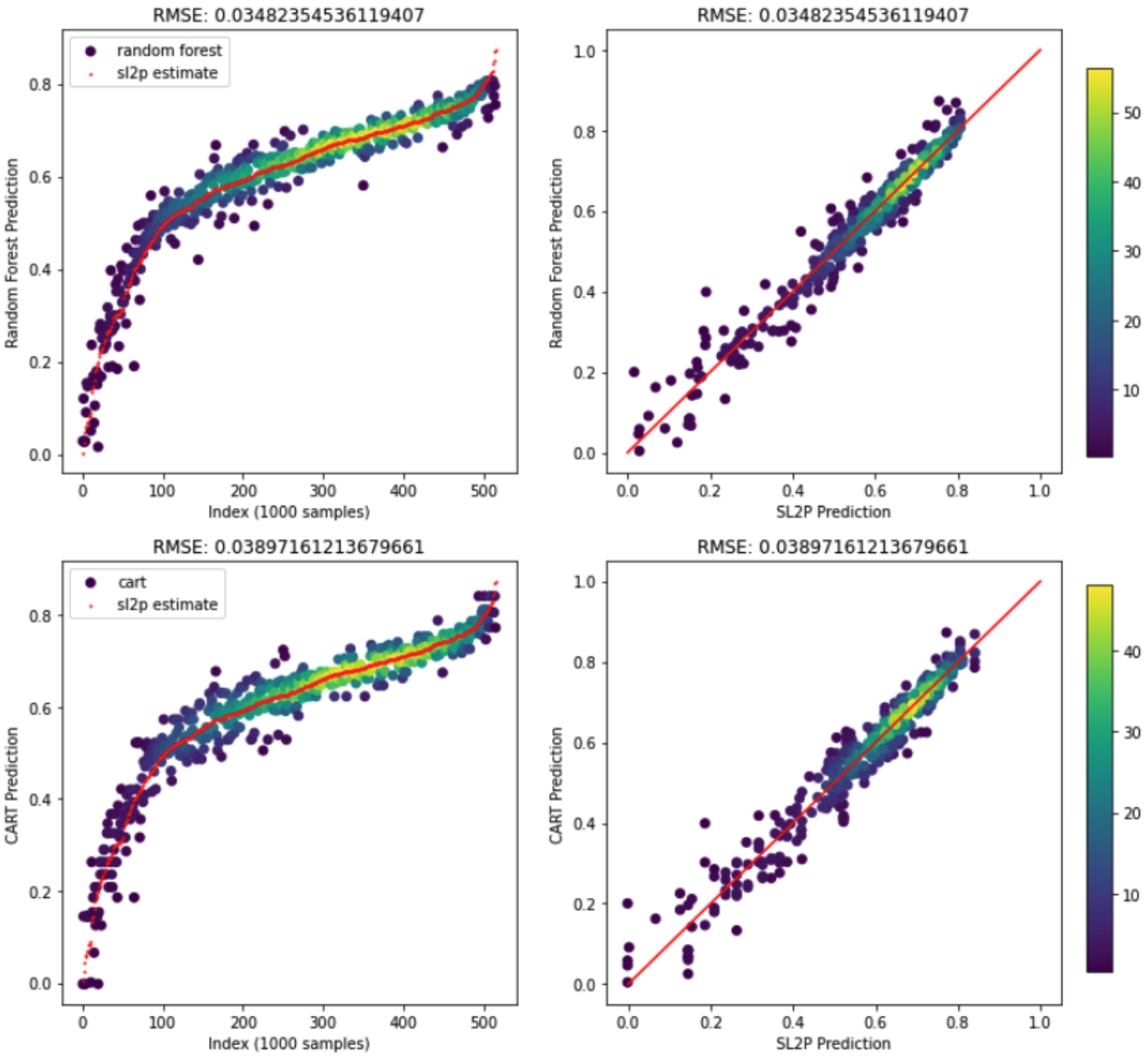

Fig. 3.3.i fCOVER predictions: SL2P vs. CART regression using 20m (L) and 10m (R) bands

3.4 Ottawa Region Test Image

Image ID: COPERNICUS/S2_SR/20200801T155911_20200801T160644_T18TVQ

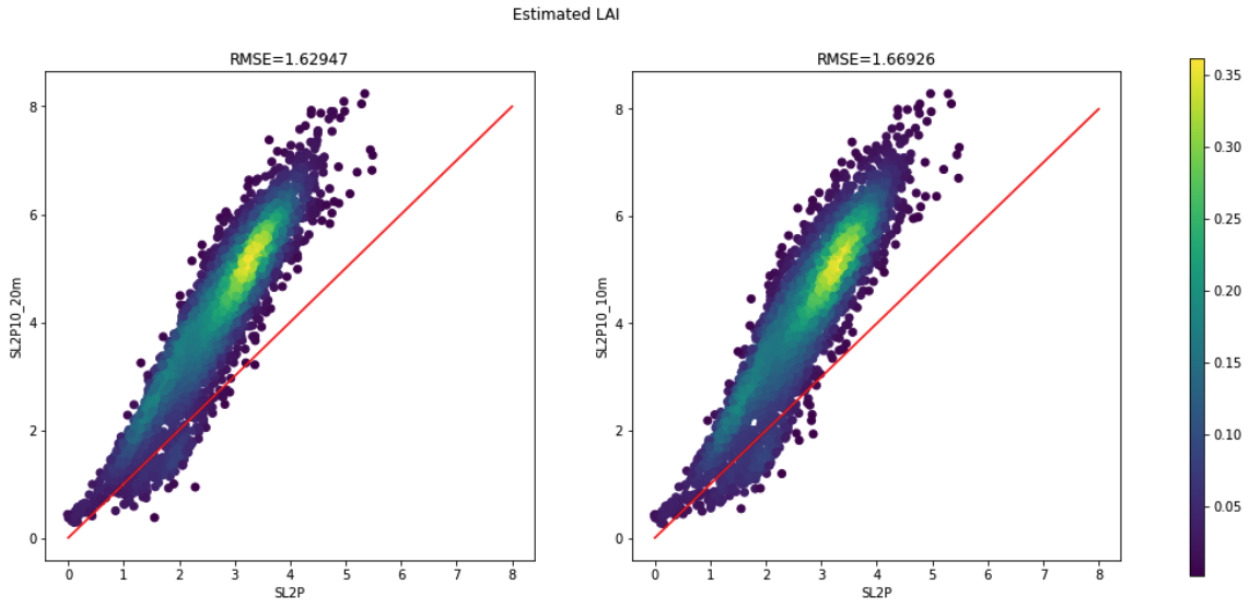

Fig. 3.4.a LAI predictions: SL2P vs. SL2P10 using 20m (L) and 10m (R) network

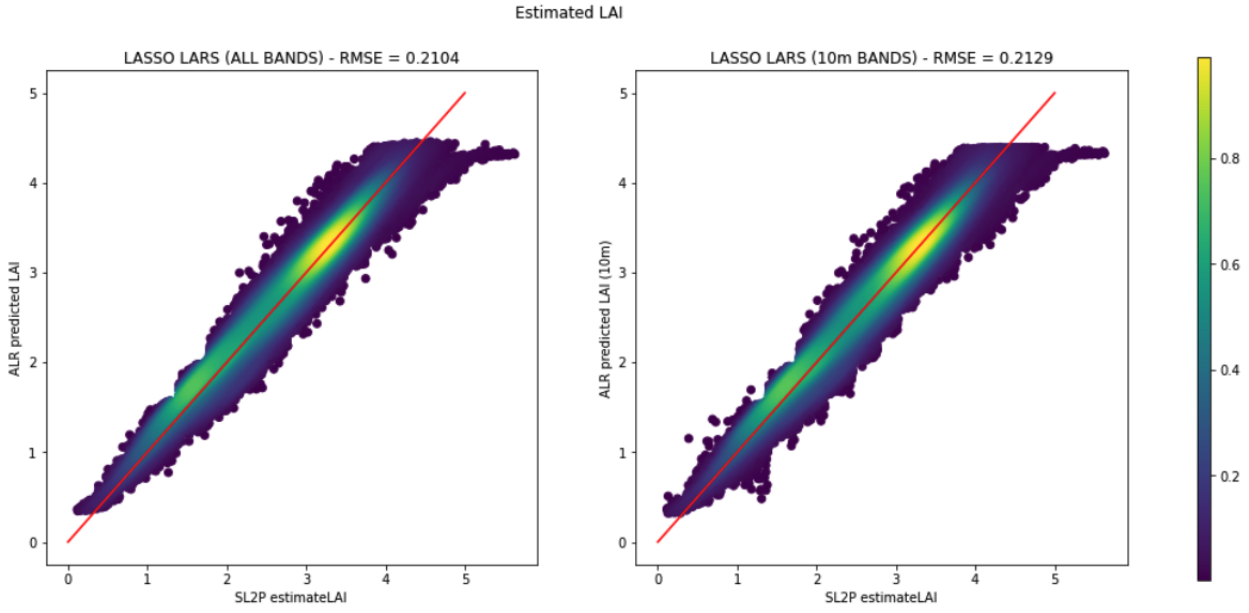

Fig. 3.4.b LAI predictions: SL2P vs. ALR using 20m (L) and 10m (R) bands

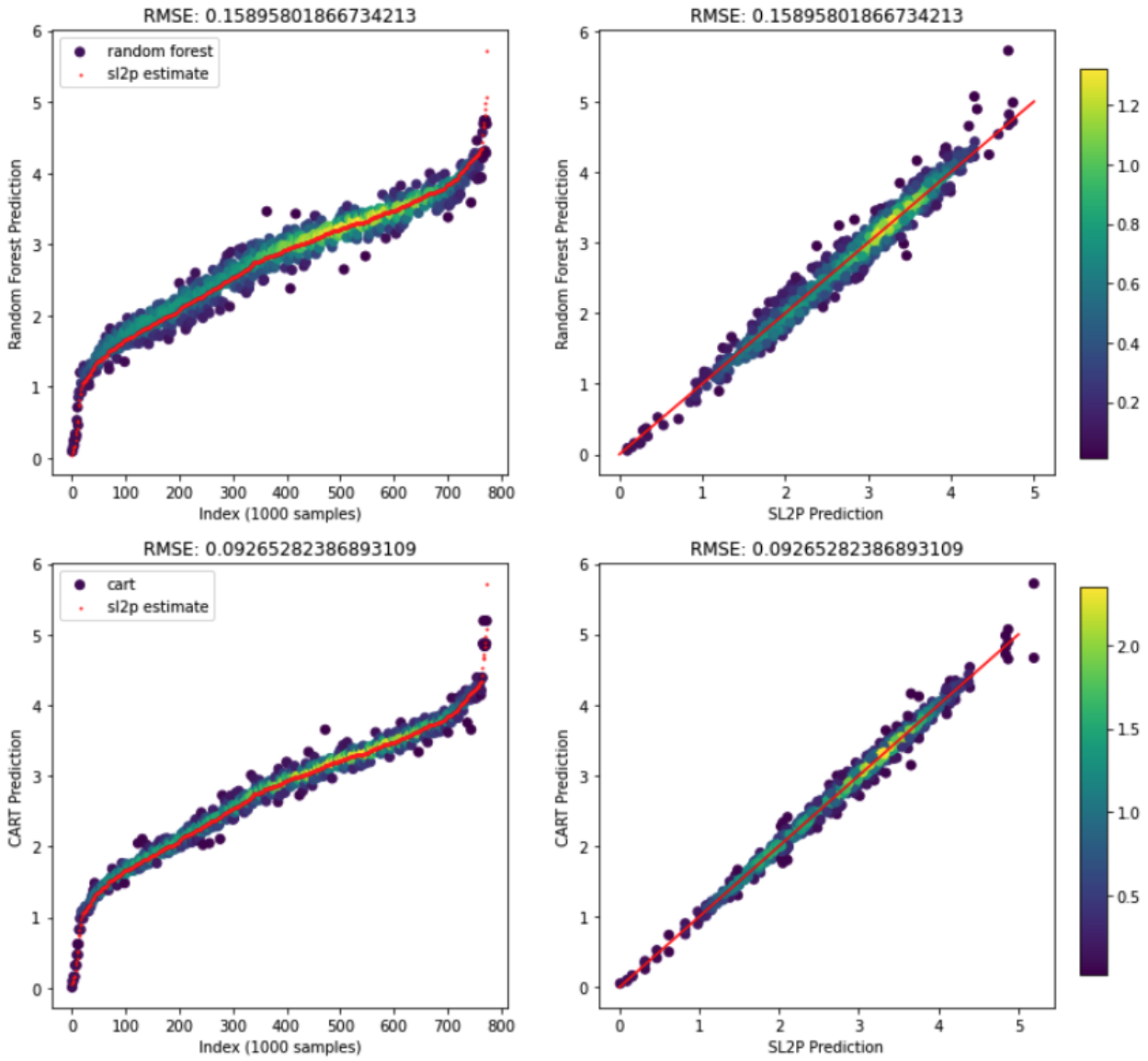

Fig. 3.4.c LAI predictions: SL2P vs. CART regression using 20m (L) and 10m (R) bands

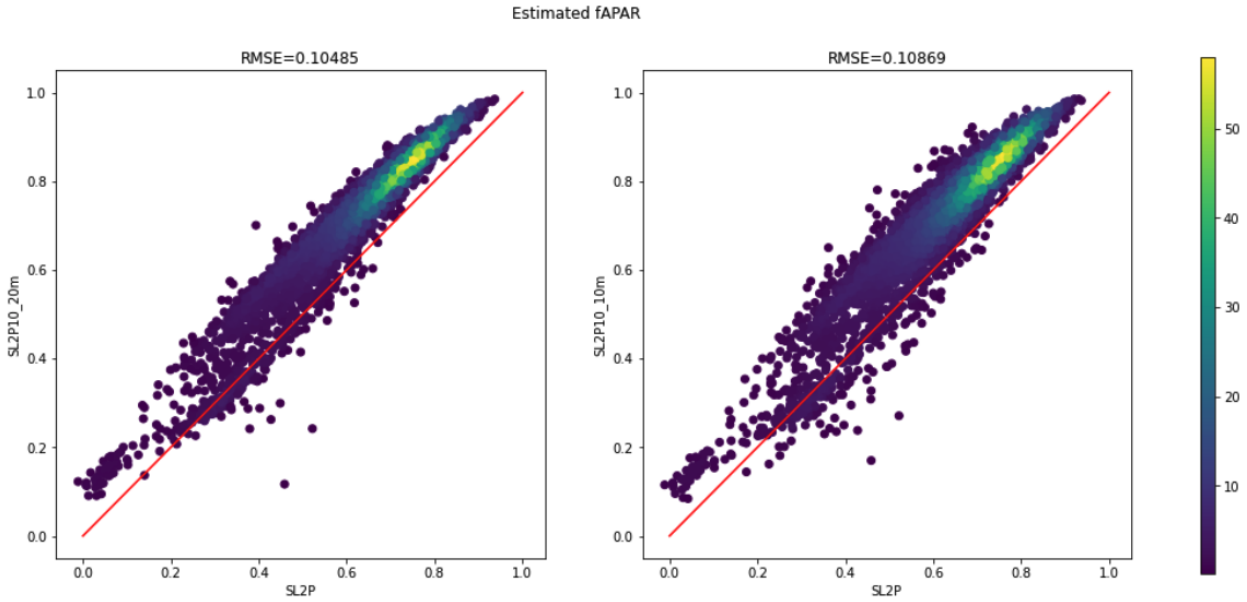

Fig. 3.4.d fAPAR predictions: SL2P vs. SL2P10 using 20m (L) and 10m (R) network

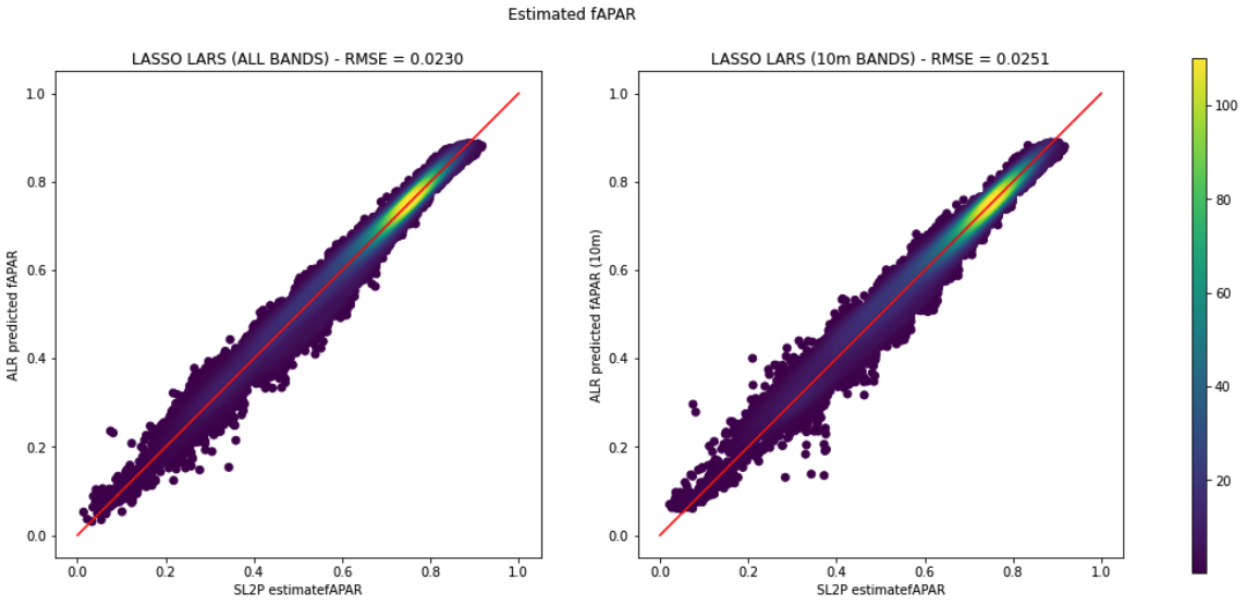

Fig. 3.4.e fAPAR predictions: SL2P vs. ALR using 20m (L) and 10m (R) bands

Fig. 3.4.f fAPAR predictions: SL2P vs. CART regression using 20m (L) and 10m (R) bands

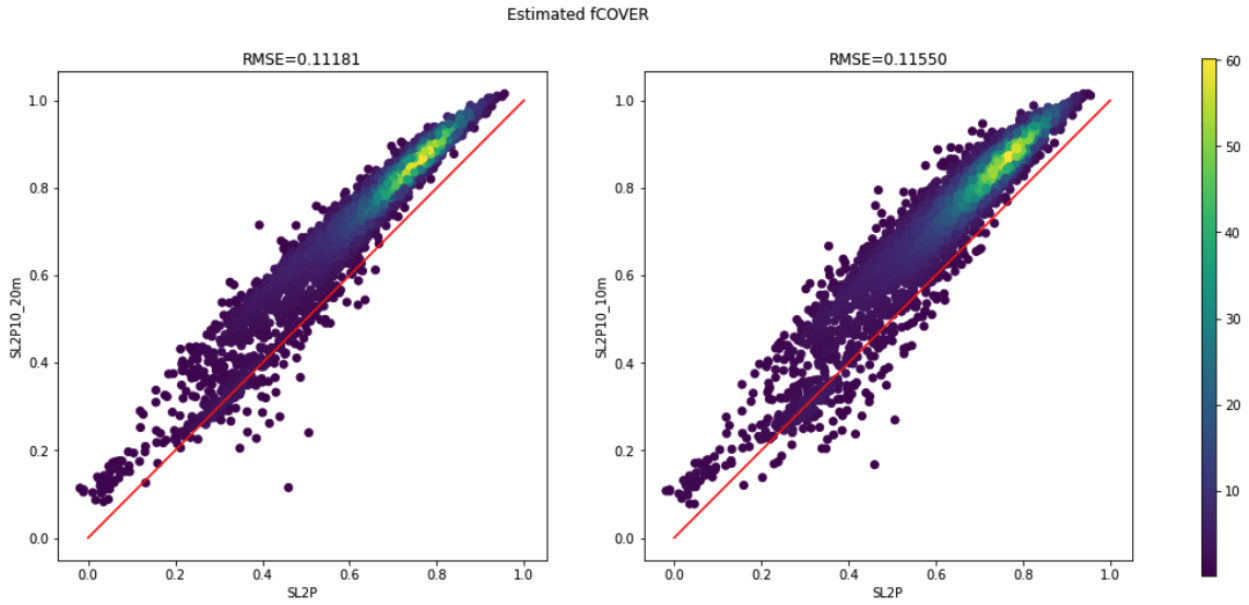

Fig. 3.4.g fCOVER predictions: SL2P vs. SL2P10 using 20m (L) and 10m (R) network

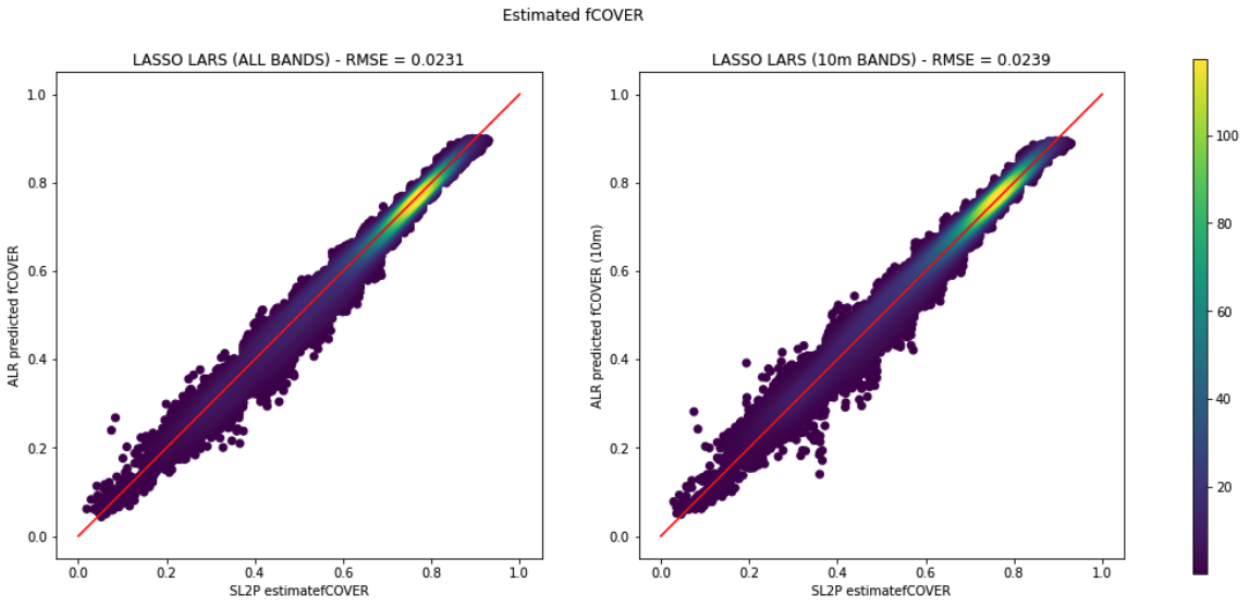

Fig. 3.4.h fCOVER predictions: SL2P vs. ALR using 20m (L) and 10m (R) bands

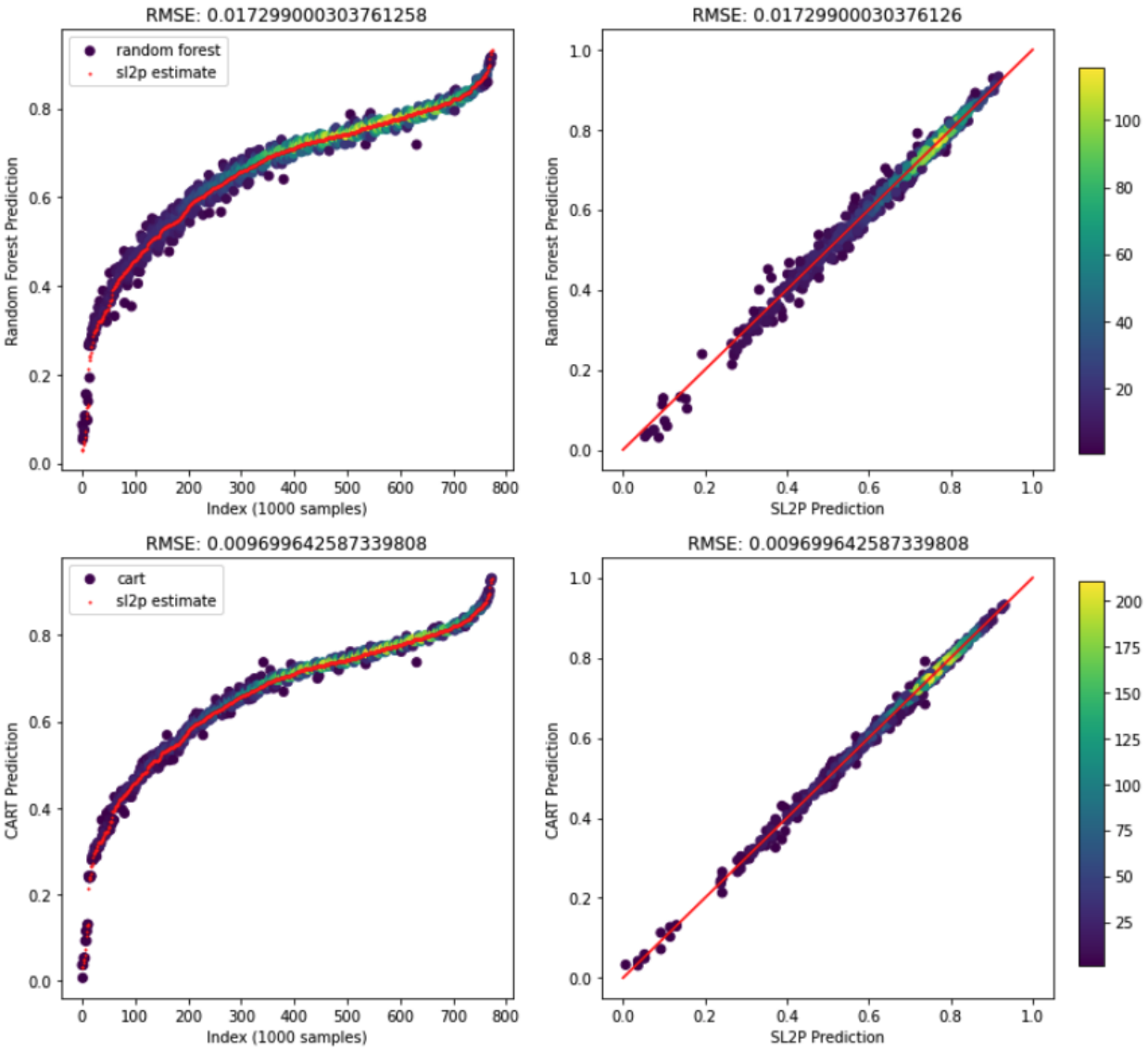

Fig. 3.4.i fAPAR predictions: SL2P vs. CART regression using 20m (L) and 10m (R) bands

APPENDIX A – CLASSIFICATION AND REGRESSION IN EARTH ENGINE

A.1 Reading Resources

- Regression in EE walkthrough: https://docs.google.com/document/d/1bRCEB8ybRUp2D6vFc7mjqDMG2bpmCd2iTPQIEern0mk/edit
- Classification example: https://www.analyticsvidhya.com/blog/2021/04/google-earth-engine-machine-learning-for-land-cover-classification-with-code/
- **Video tutorial**: https://developers.google.com/earth-engine/guides/classification

A.2 Classification

Supervised Learning

- Training and classifying **feature vectors** (tables)
- Feature vectors (see table as example)
    - Rows: features
    - Columns: properties

Feature	Time(DOY)	NDVI	Elevation	Class
1	32	0.60	230	0
2	45	0.65	242	0
3	63	0.72	400	1
4	63	0.8	500	1
...	...	...	...	

Classification Workflow

1. Build
    a. Requires feature collection (table using image.sample()) with training data
2. Train
    a. Classifier.train()
3. Apply
    a. Image.classify()
    b. Can be applied to images or tables

4. Assess
    a. Classifier.confusionMatrix() (cross-validation with training data)
        i. Re-classify based on training data
    b. confusionMatrix.accuracy()

Random Forest vs CART

- Random Forest
    - Pros: uses multiple CART models → better accuracy in general
    - Cons: not able to interpret the model (is that important in this case?)

Implementation/Testing

- Example script using random forest classifier in EE: https://code.earthengine.google.com/08939bea01a8419560caeb1efa81000d
- How to tune hyperparameters for best model performance: https://towardsdatascience.com/hyperparameter-tuning-the-random-forest-in-python-using-scikit-learn-28d2aa77dd74
- To consider:
    - Implementation in scikit-learn – easy evaluation methods
    -