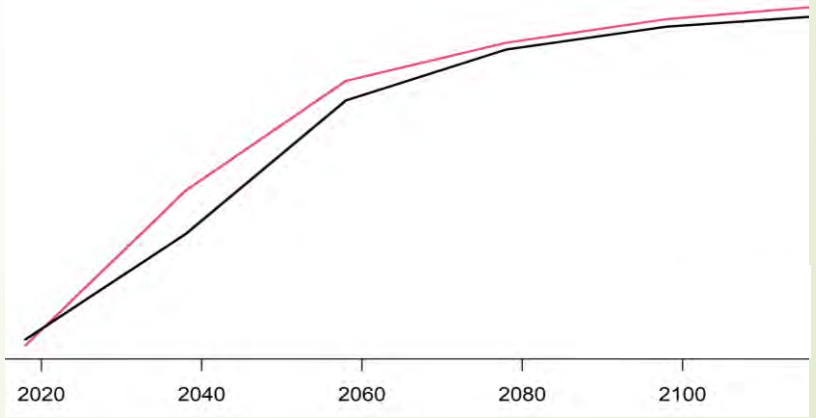
FVS Growth from kNN Lidar Inventory

Jacob Strunk

4/06/2022



Outline

- Site and Context
- Study Overview
- kNN explained
- FVS Results
  - "Tract"
  - Stand
  - Stand vs Plot growth approach
  - Other strategies
- Future work

Study Site

- Savannah River Site in South Carolina
- Department of Energy facility but non-facility lands are managed by the Forest Service
- Total area is ~200,000 acres with ~170,000 acres of forested land
- Lands managed for wood products and habitat for various species
- Some areas are restricted or require special precautions for entry
- Earlier lidar project in 2009-2010

Study Context

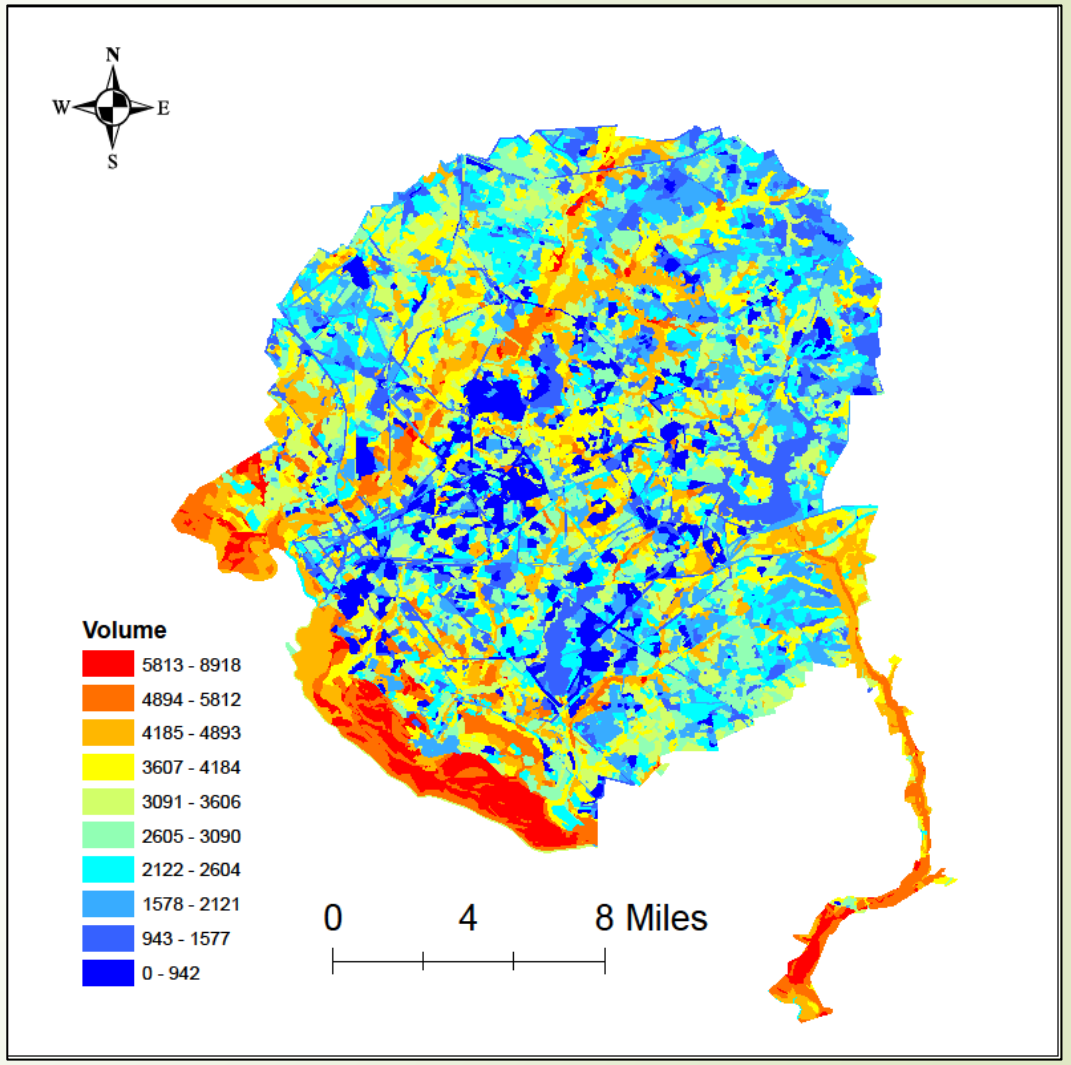
Lidar inventory completed in 2021

Also interested in:

- Management effects
  - Rotation length
  - Thinning / not
  - Planting density
- Sustainable harvest levels
- Various growth-related questions…

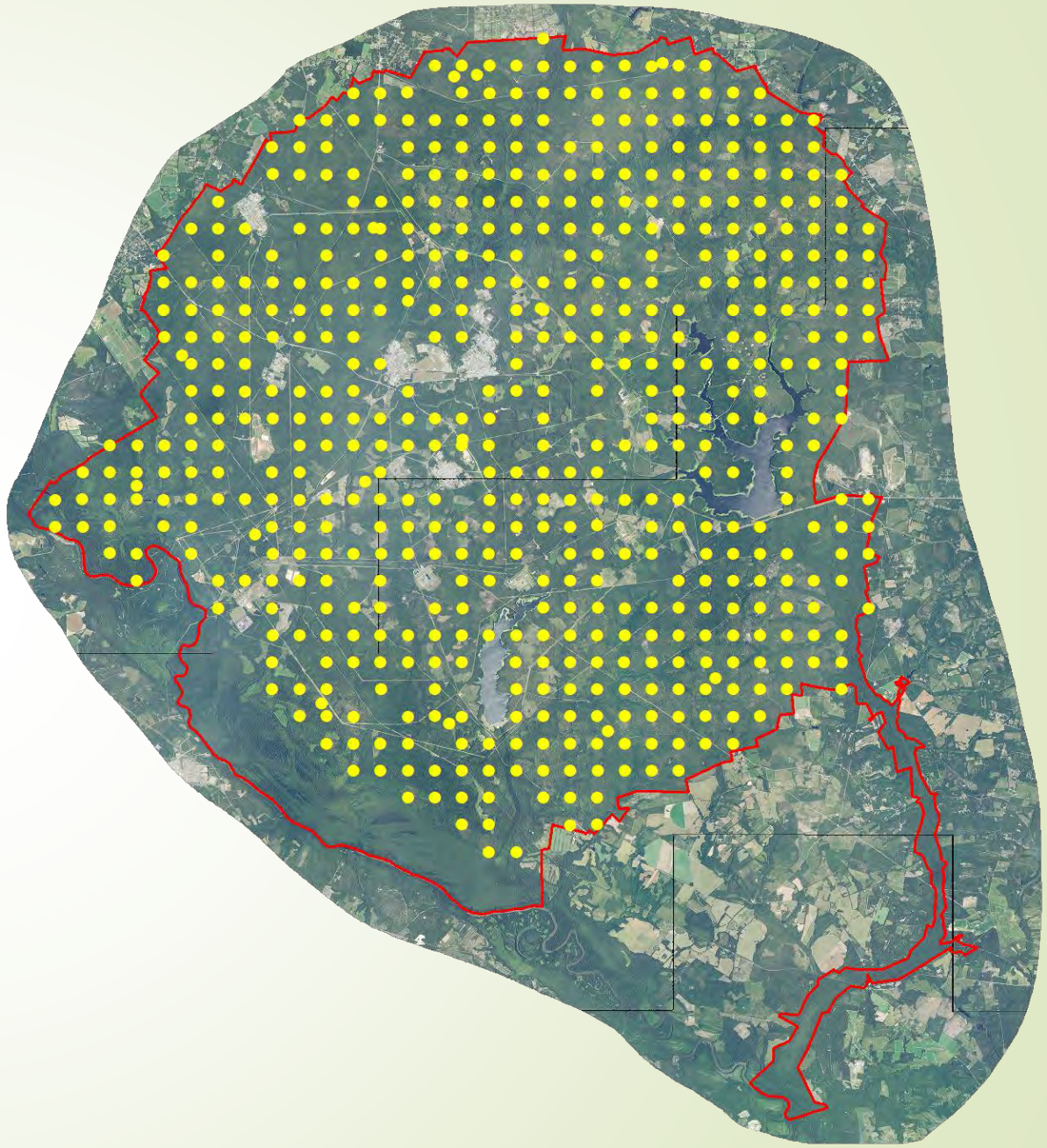
Question:

Answer with lidar inventory?

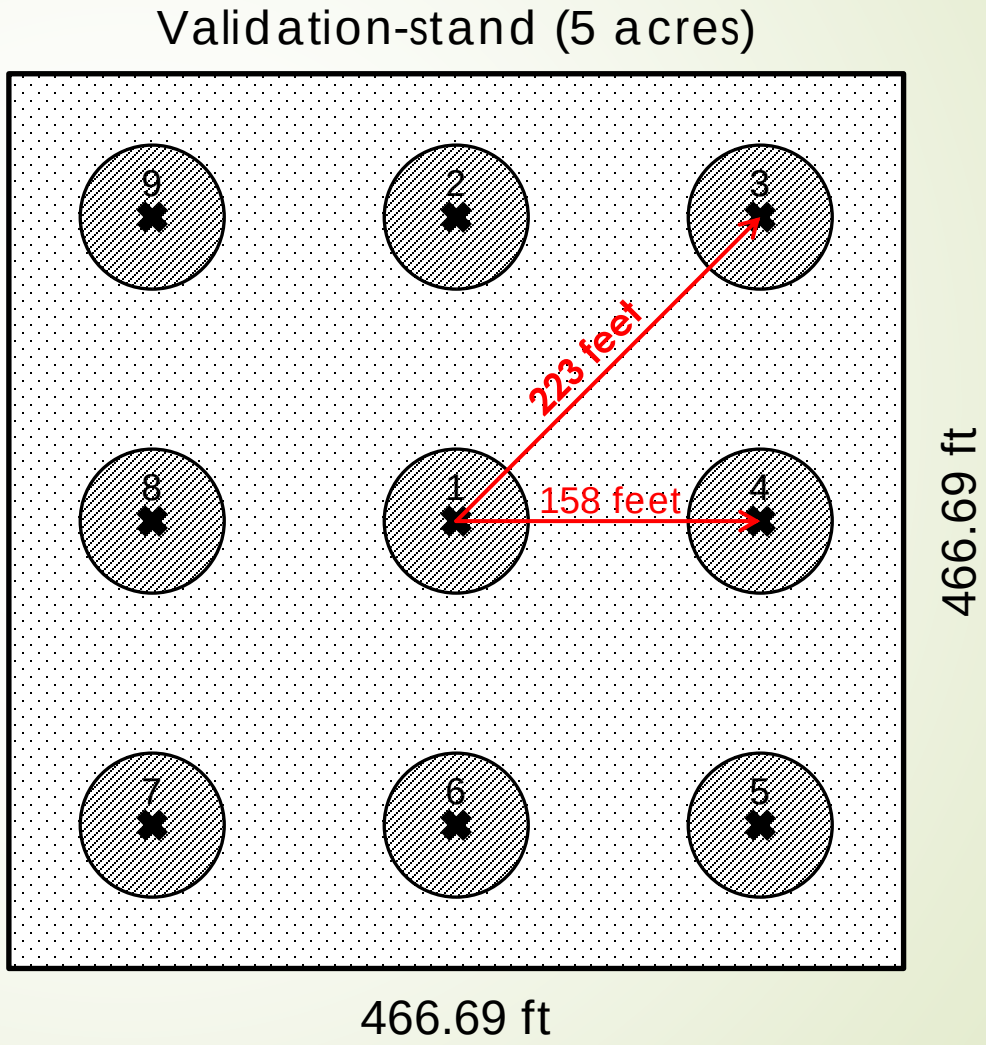


Study Overview

- Lidar Inventory with 500+ Gridded plots
- 50 validation stands
- Evaluate FVS projections fromLIDAR
  - Predicted (imputed) stands
  - Observed stand measurements

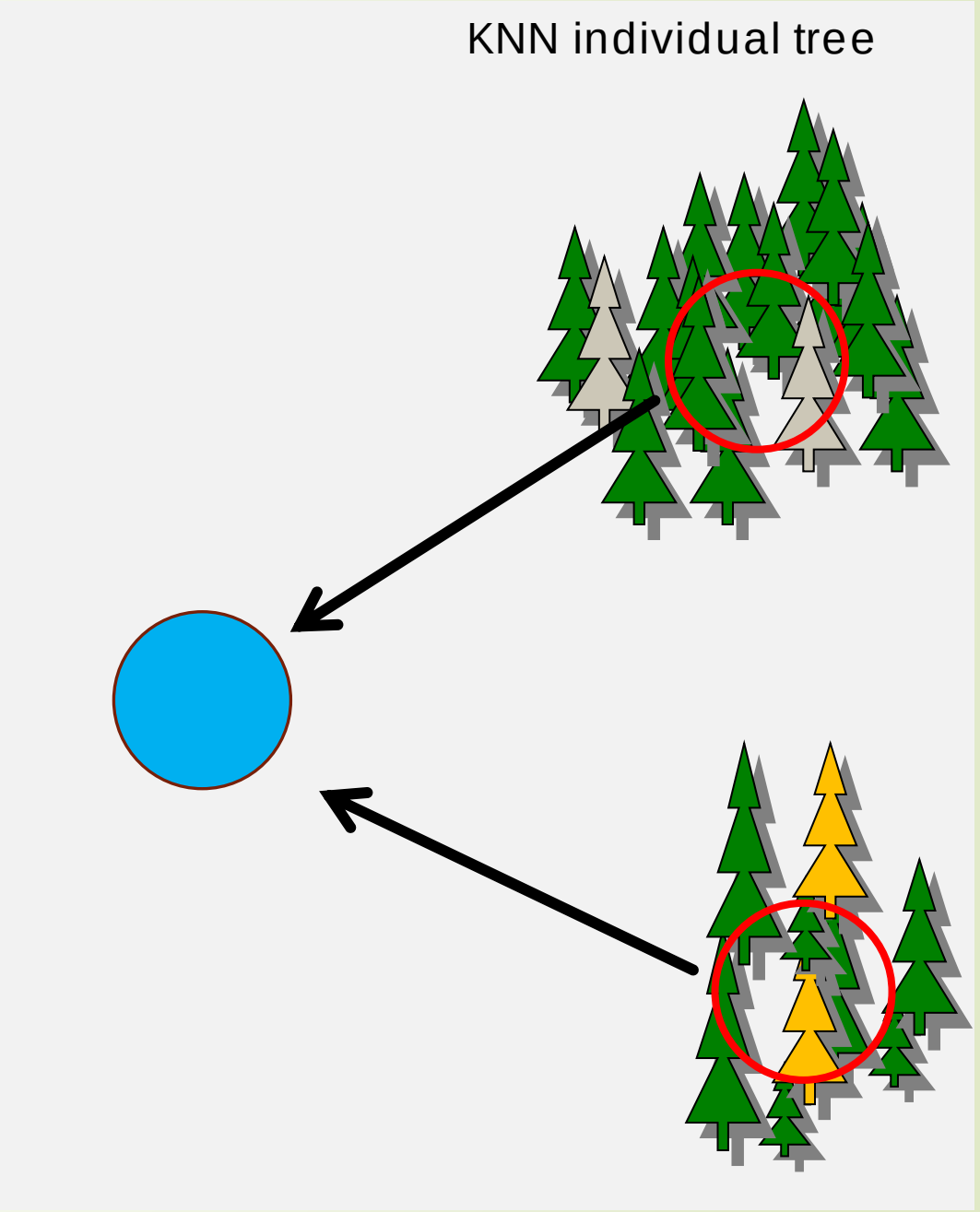


50 Validation Stands (9 X $\frac{1}{10}$ acre plots, 5 acres)

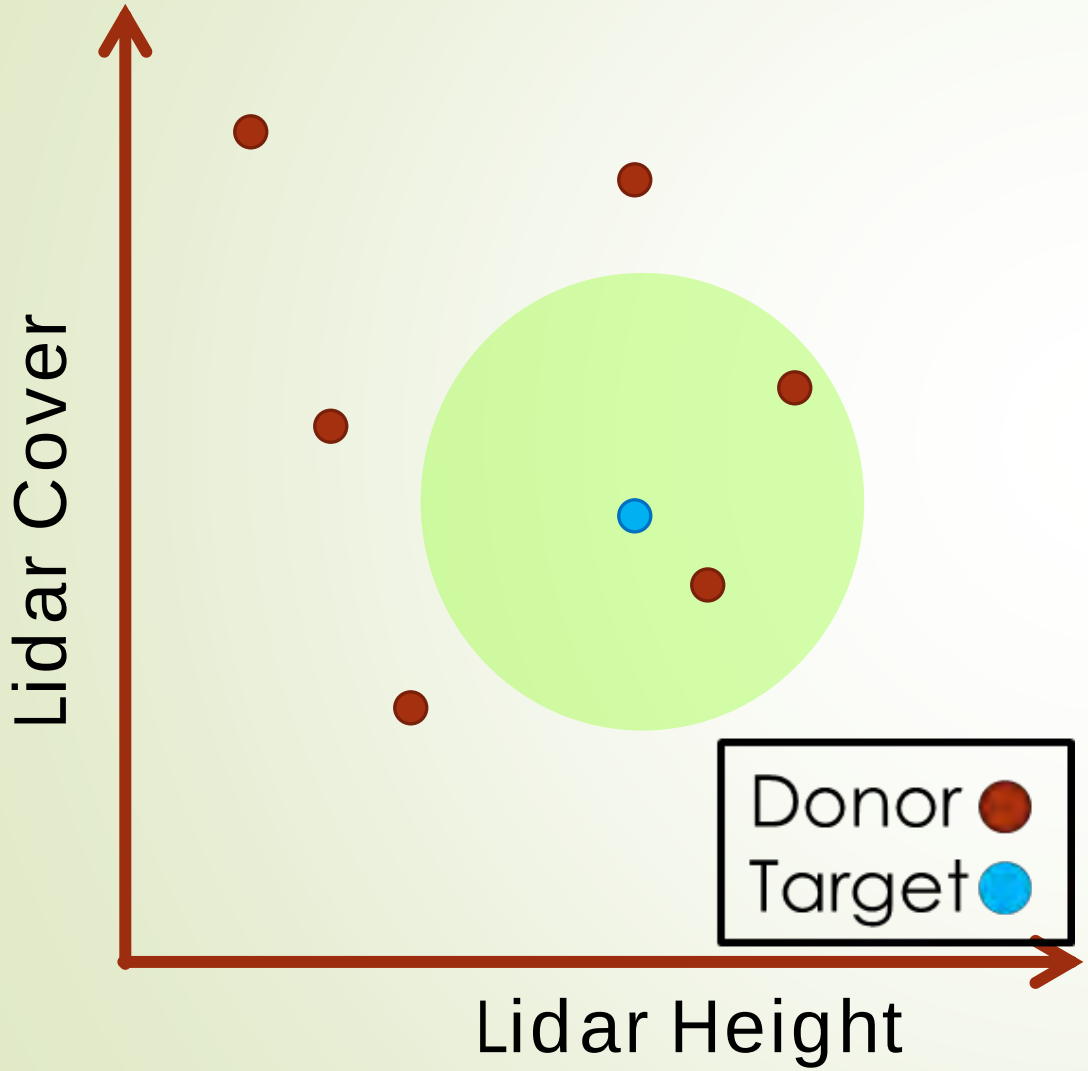


Generating Tree Lists with kNN

- Find similar measured plots
  - Lidar height
  - Lidar Cover
- Lots of ways to compute distance
  - Euclidean
  - Mahalanobis
  - MSN
  - Random Forest
  - GNN

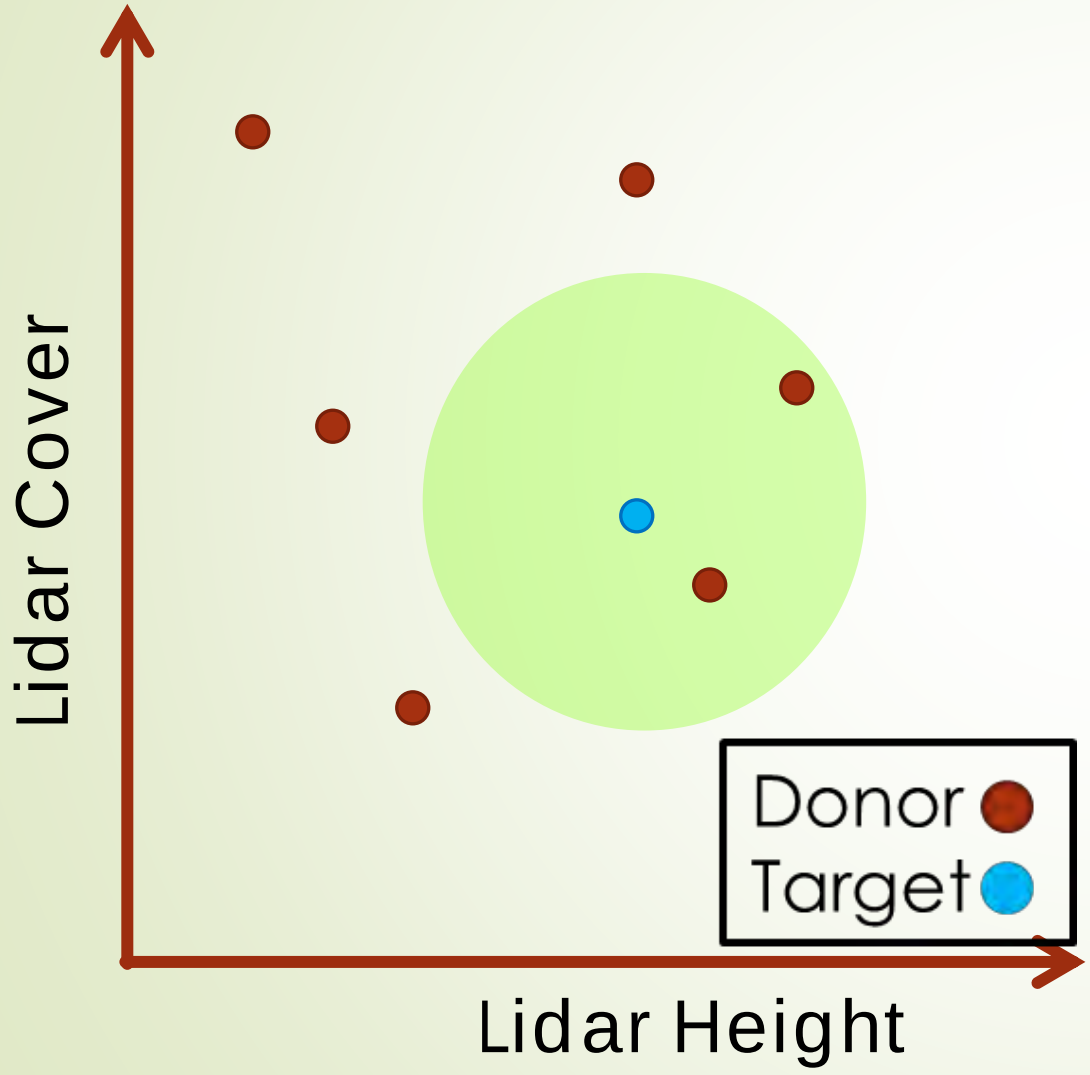


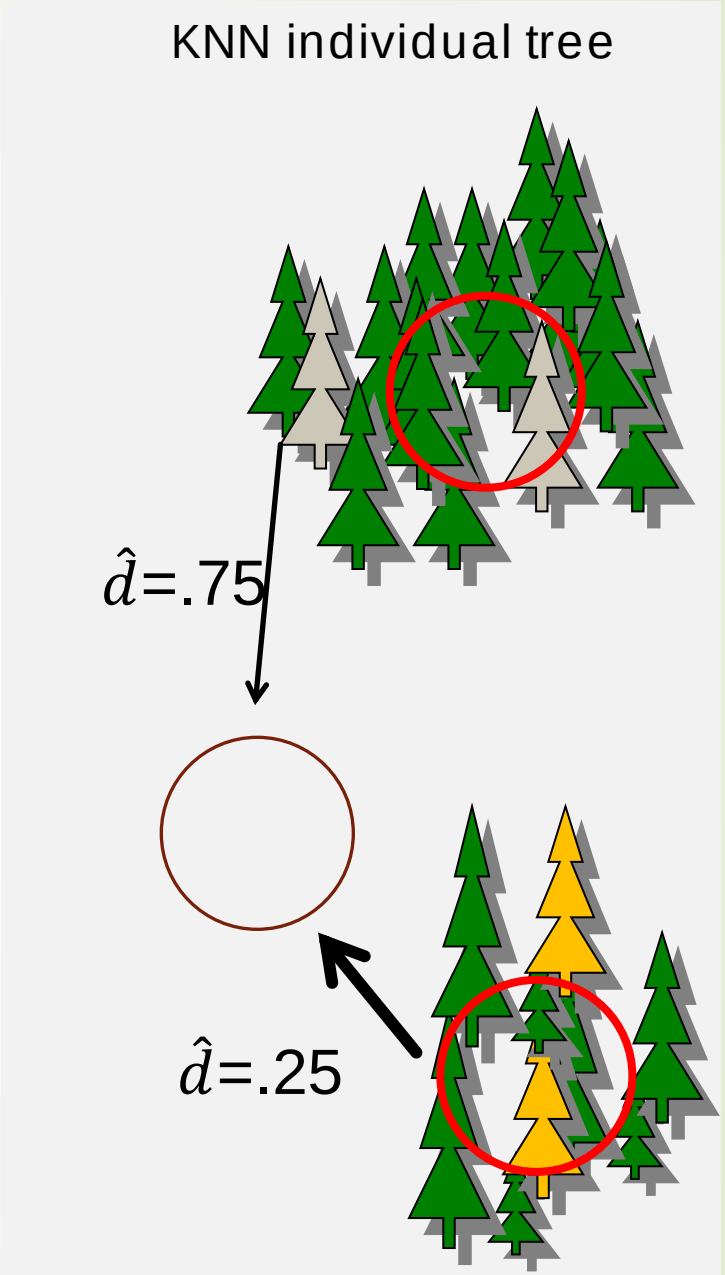
Generating Tree Lists with kNN



kNN: Select k= 2
(e.g.) Nearest Neighbors

kNN Tree List





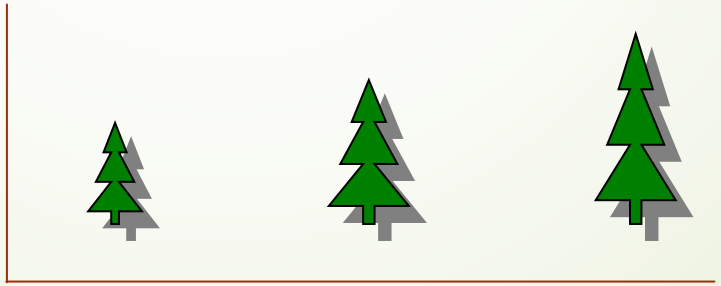
Growth Projections

- Benchmarks: no management
- USFS FVS Software
  - FVSsn.exe
  - Called from R
  - Package to help? –
    - github.com/jstrunk001/RForInvt
    - Wrapper for NVEL
    - "Wrapper" for FVS

```
#assume a typical inventory dataset and prepare fvs parameters
#the fvs_make_keyfiles will take this list of parameters and make a
#separate key file for each record in the FVS_StandInit table
  start=2018
  finish = 2118
  interval = 20
  num = ceiling((finish - start) /interval)
  print(num)

  df_params = fvs_protype_params()
  df_params[1:nrow(fvs_stands),]=NA
  df_params[,"std_cn"] = fvs_stands$stand_id
  df_params[,"std_id"] = fvs_stands$stand_id
  df_params[,"invyr"] = fvs_stands$inv_year
  df_params[,"timeint"] = interval
  df_params[,"numcycle"] = num
  df_params[,"input_db"] = dir_fvs_in
  df_params[,"fvs_path"] = "C:/FVS/FVSbin/FVSsn.exe"
  df_params[,"tree_table"] = "FVS_TREEINIT"
  df_params[,"stand_table"] = "FVS_STANDINIT"
  df_params

#prepare prototype key file - the fvs_prototype_keyfile function just put
#series of text strings. Some common keywords are suggested (e.g. notripl
#If the parameter is NULL, then the keyword is not included, and other ke

  #here we override the defualt to used std_id and use std_cn instead
  key_proto = fvs_prototype_keyfile(
                          std_id = "@std_id@"
                        , std_cn = "@std_id@"
                        , other_keywords = kcp
                        )

#convert prototype key file into series of key files associated with each
  try(unlink(paste0("c:\\FVS\\FVS_knn_",this_version)))
  df_keys = fvs_make_keyfiles(df_params, key_proto = key_proto, cluster =
                              ,processing_dir = paste0("c:\\FVS\\FVS_knn_

#make sure to look at a few keyword files to see that they are  (especial
  df_keys

#lastly, actually run fvs - if processing in parallel this function creat
  fvs_run(df_keys, cluster = clus1, merge_dbs=T , delete_temp_db = F)
```

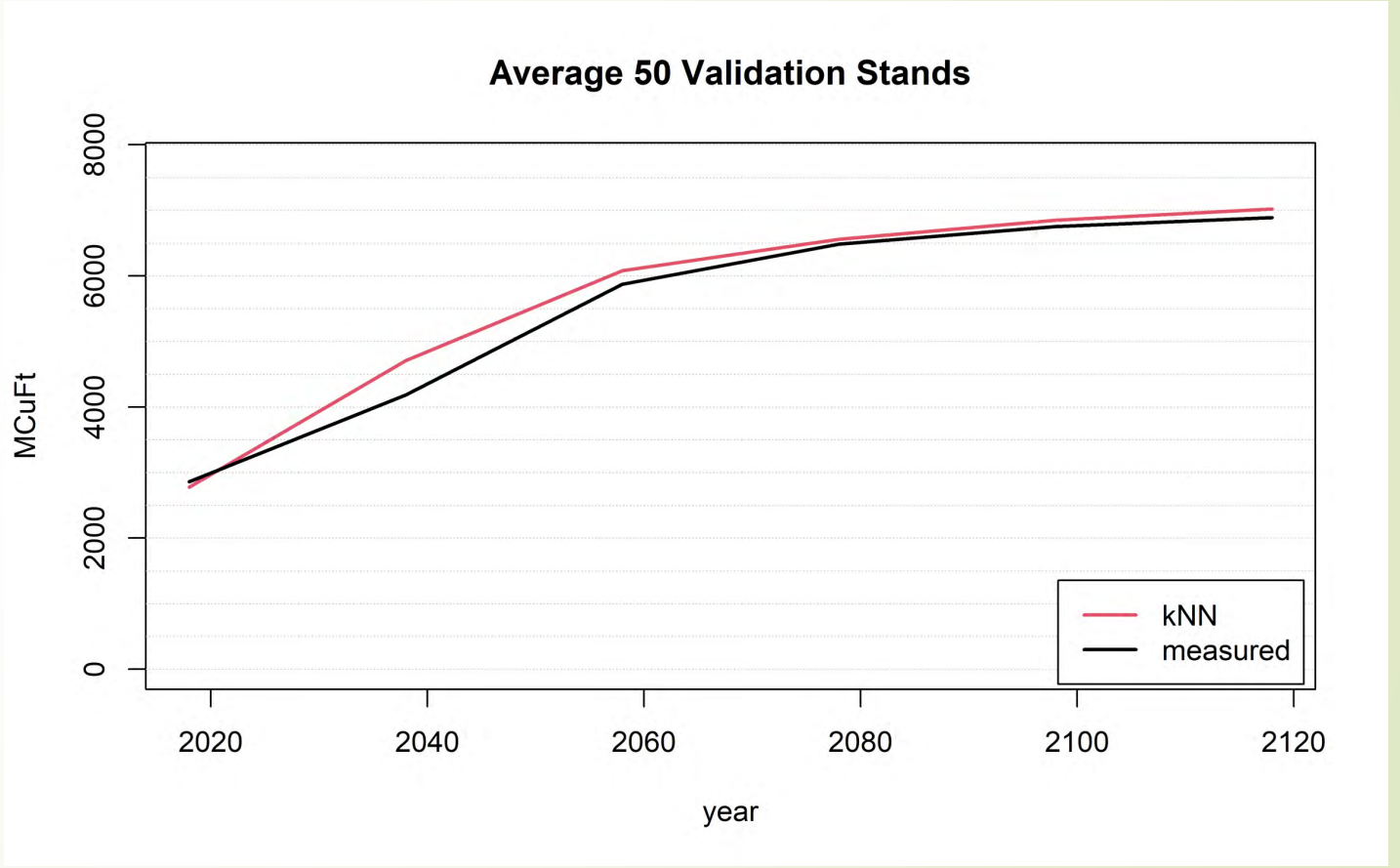
Initial FVS Results

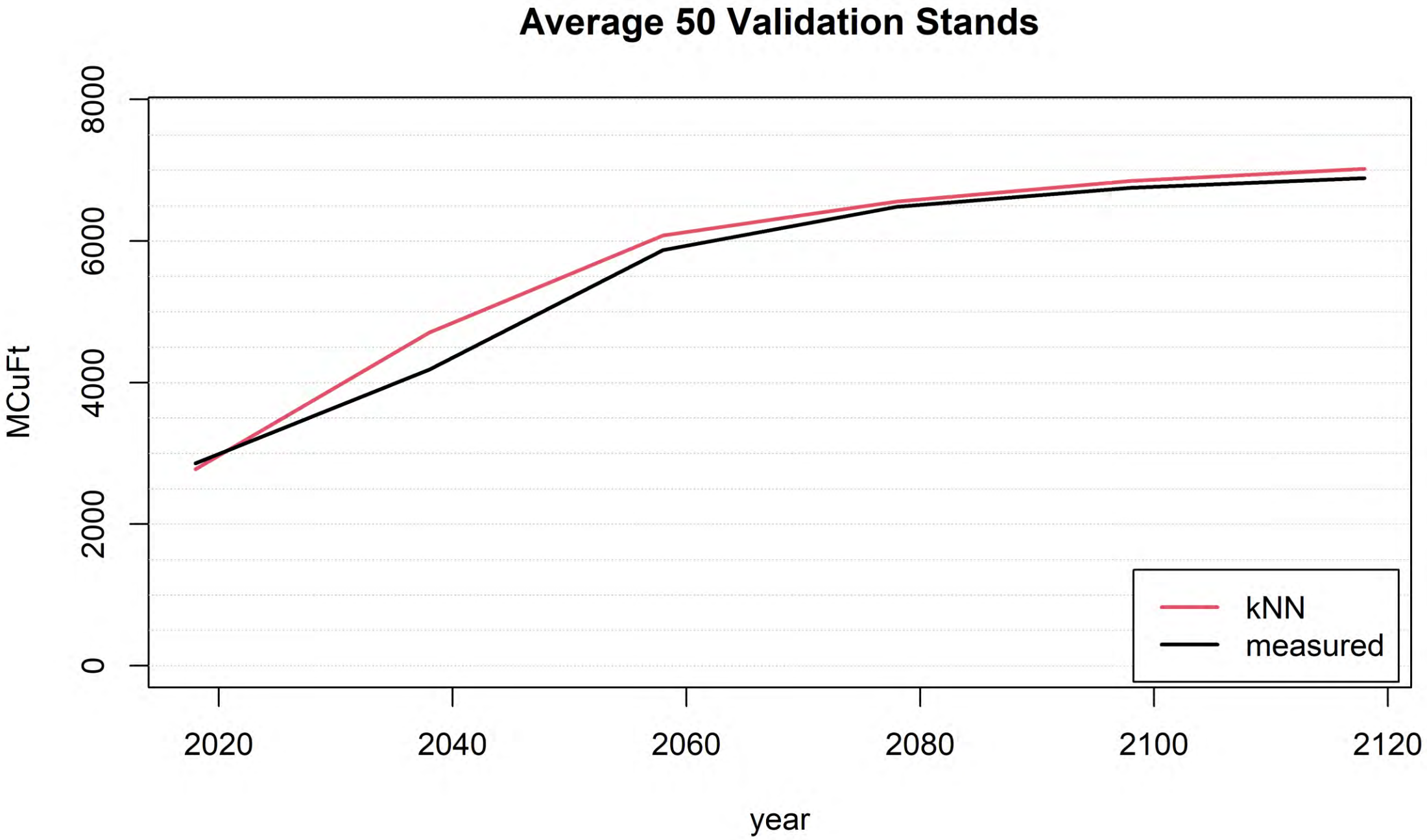
FVS Projections for Validation Stands



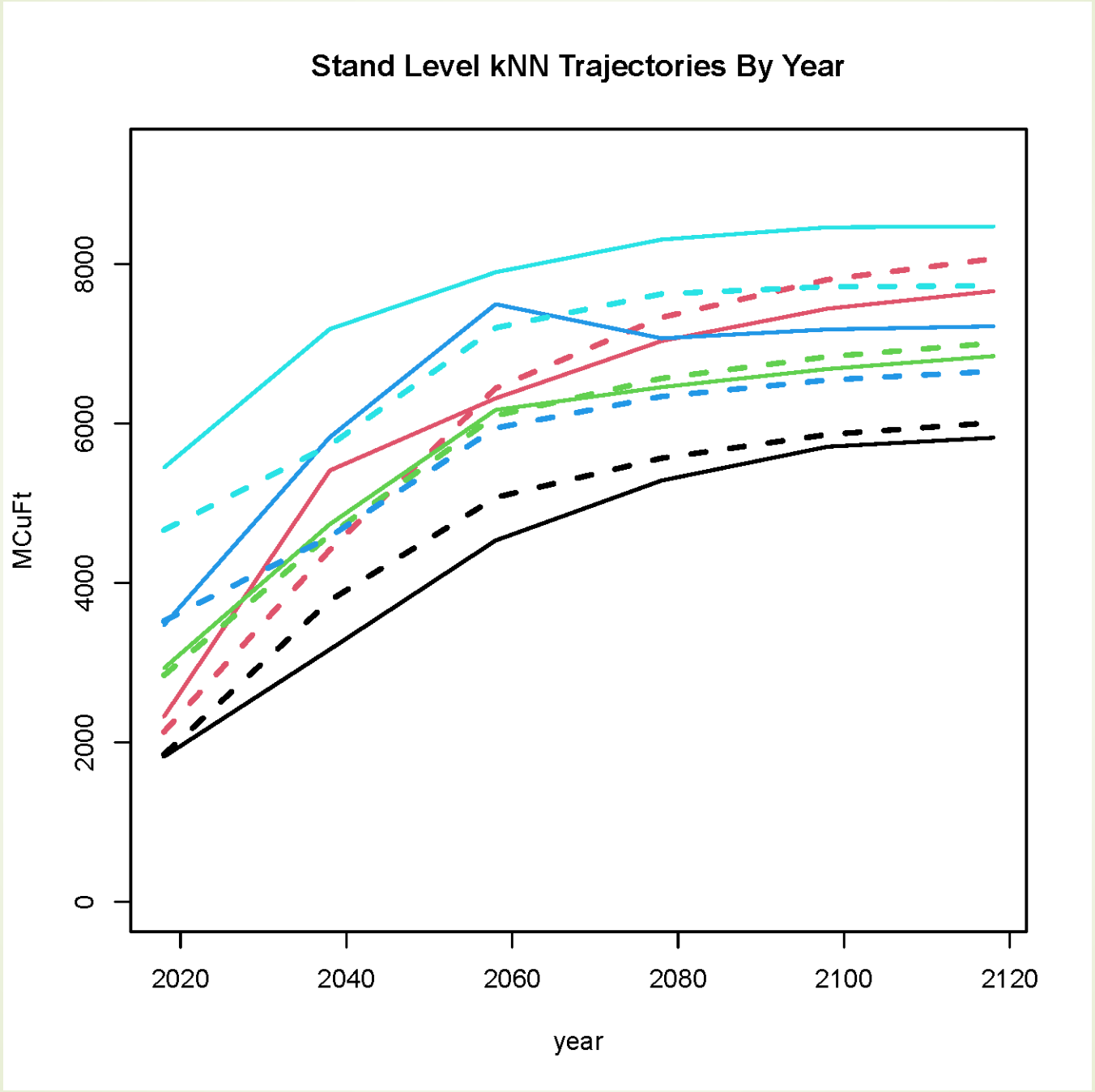
“Tract” or Average Stand performance

- kNN results reasonable
- 500 cubic feet of error max
- ?product distributions?

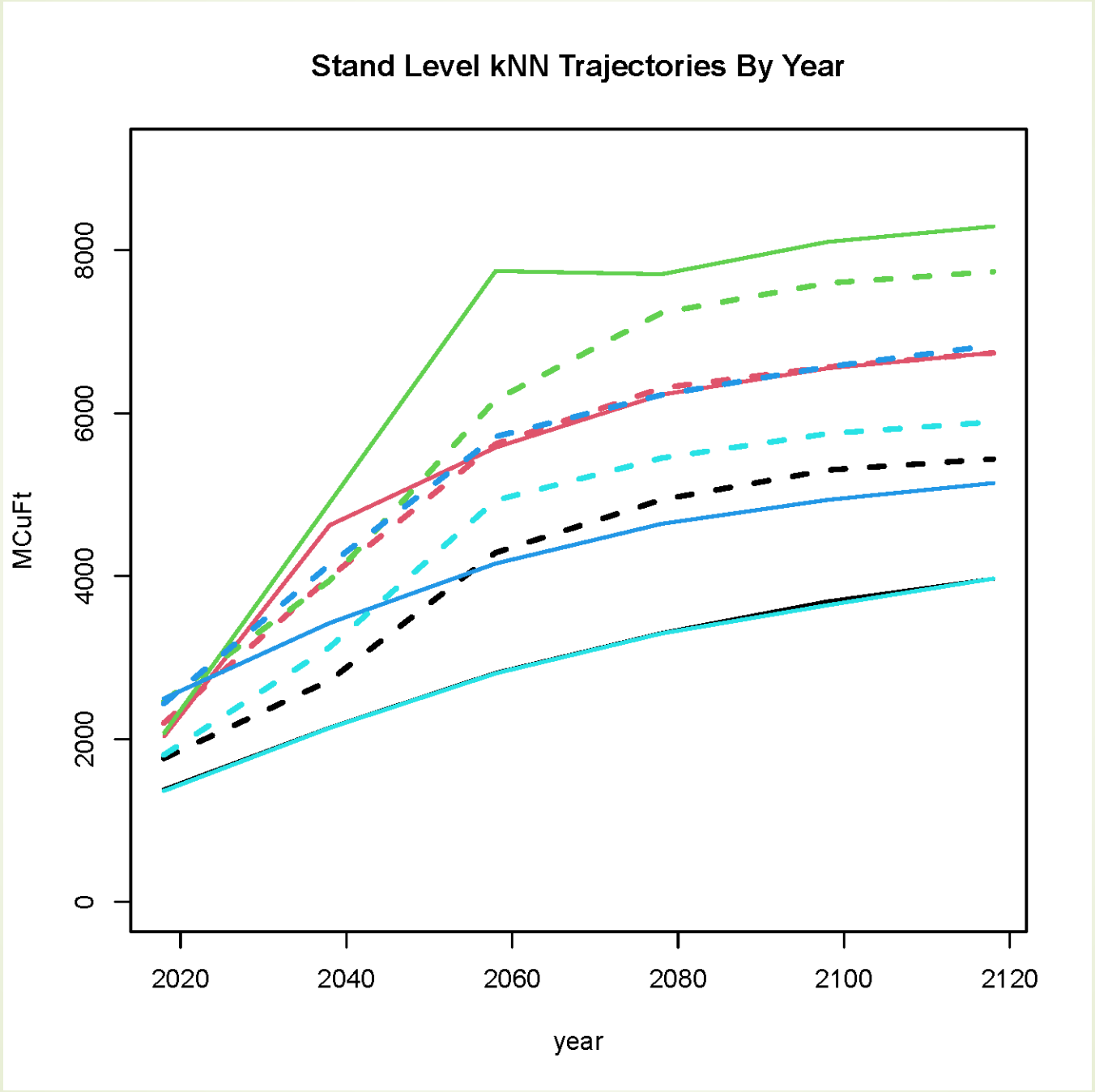



Average 50 Validation Stands
MCuFt
0
2000
4000
6000
8000
2020
2040
2060
2080
2100
2120
year
kNN
measured

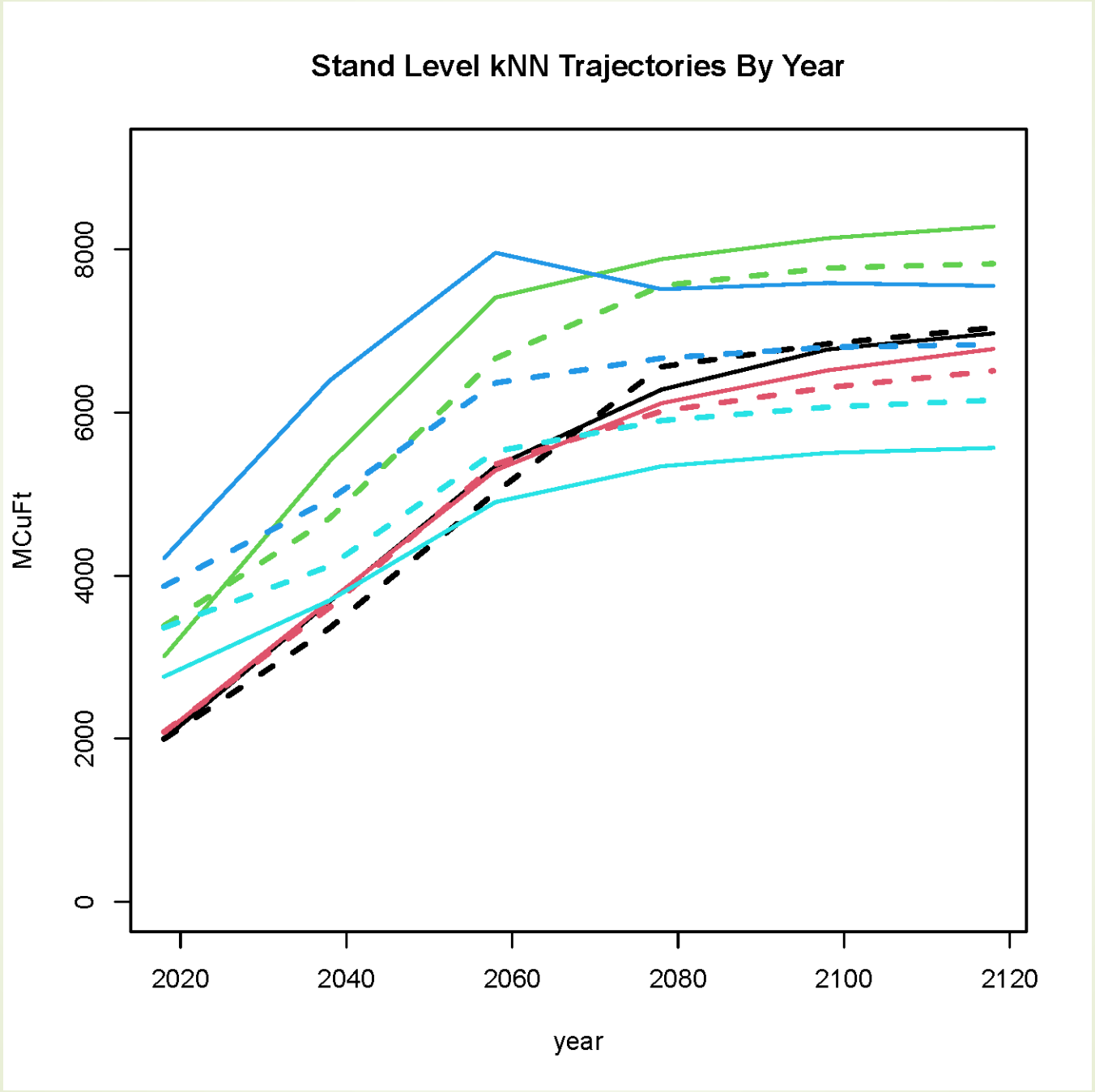
A.


Stand Level kNN Trajectories By Year
MCuFt
0
2000
4000
6000
8000
2020
2040
2060
2080
2100
2120
year

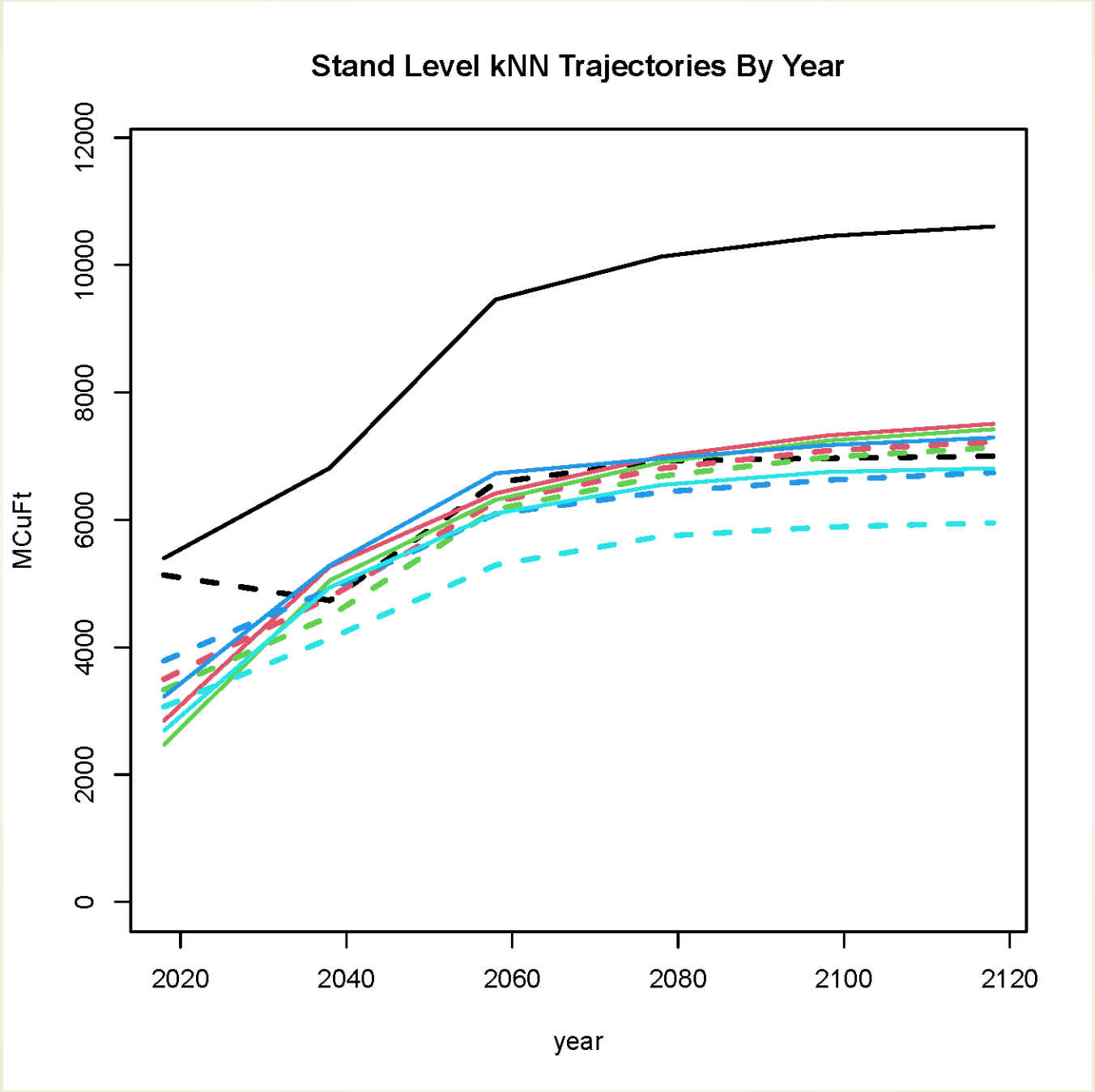
B.

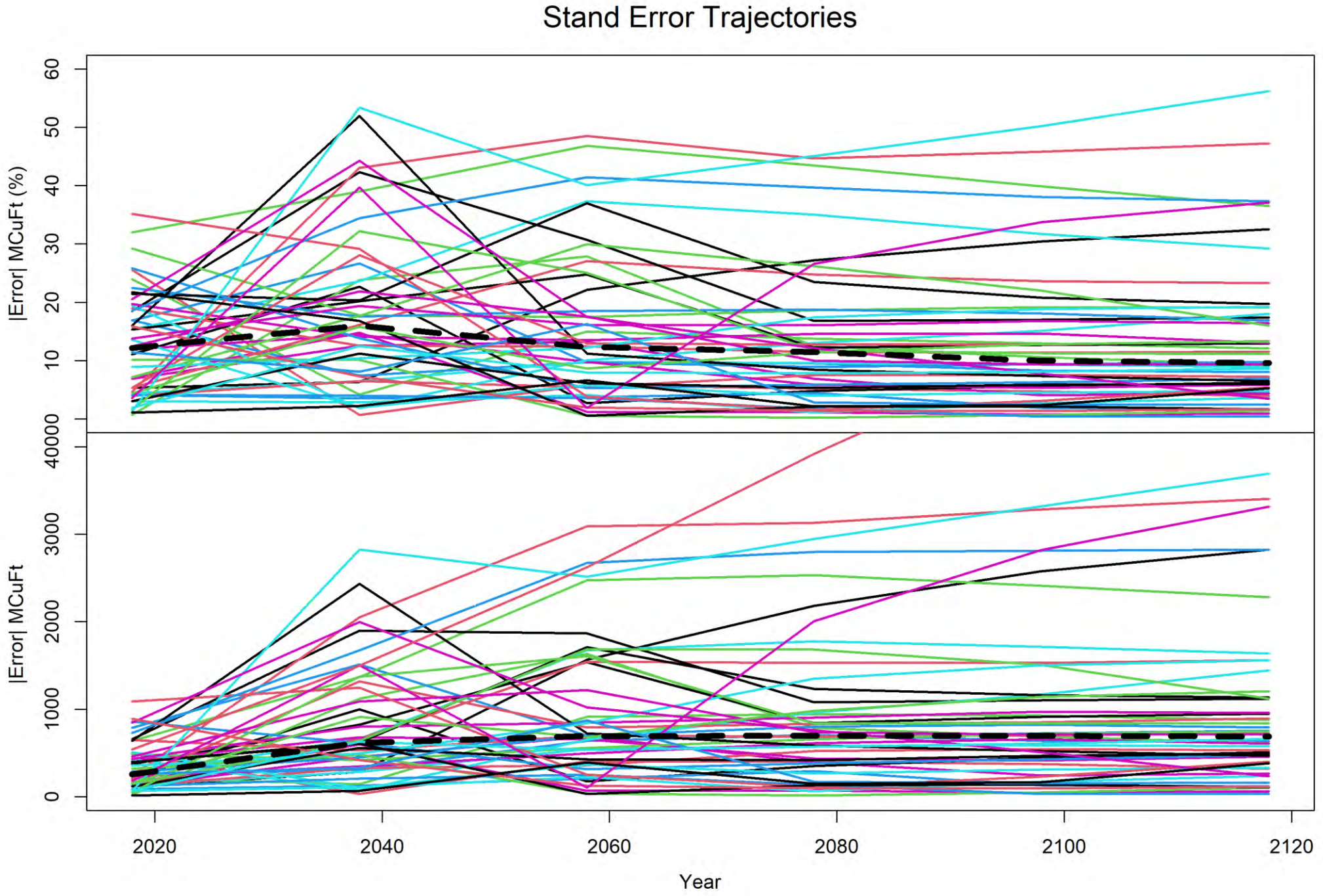

Stand Level kNN Trajectories By Year
MCuFt
0
2000
4000
6000
8000
2020
2040
2060
2080
2100
2120
year

C.


Stand Level kNN Trajectories By Year
MCuFt
0
2000
4000
6000
8000
2020
2040
2060
2080
2100
2120
year

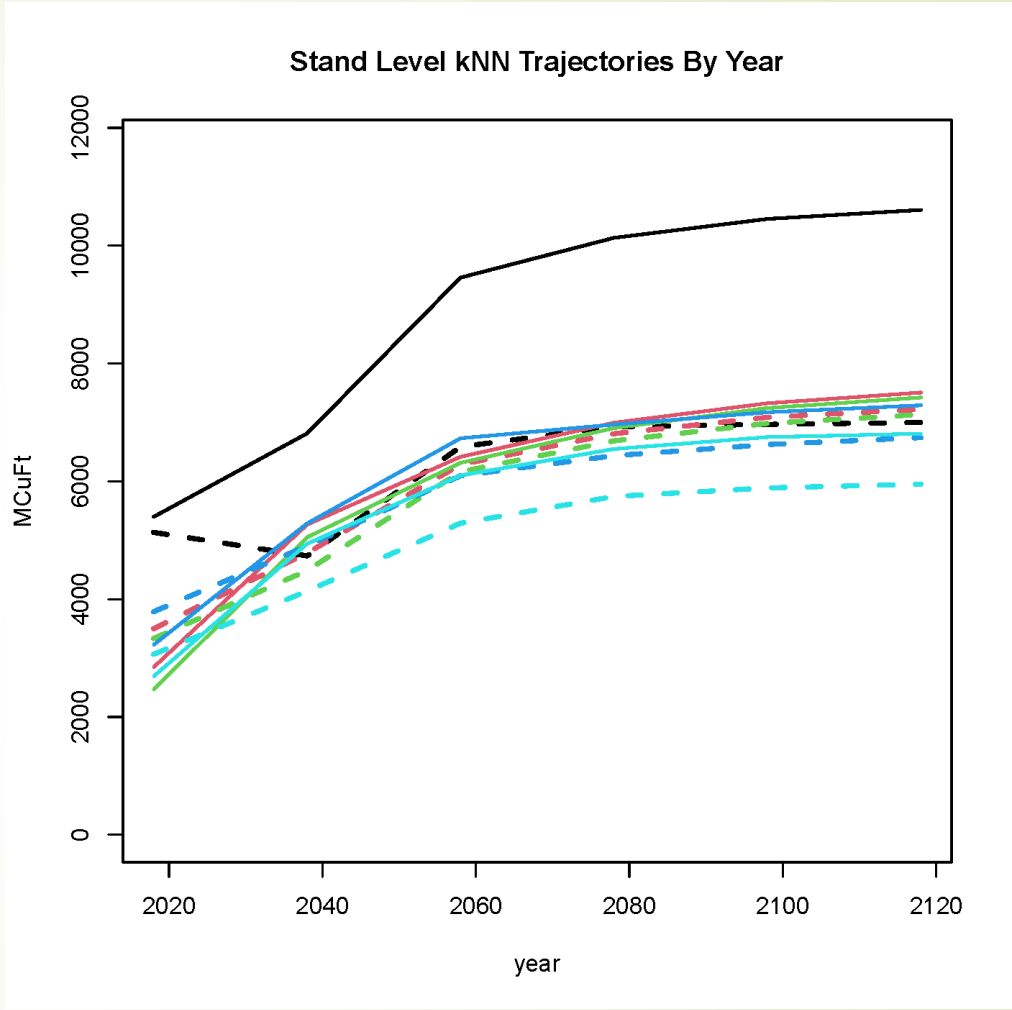
D.


Stand Level kNN Trajectories By Year
MCuFt
0
2000
4000
6000
8000
10000
12000
2020
2040
2060
2080
2100
2120
year


Stand Error Trajectories
|Error| MCuFt (%)
0
10
20
30
40
50
60
|Error| MCuFt
0
1000
2000
3000
40000
2020
2040
2060
2080
2100
2120
Year

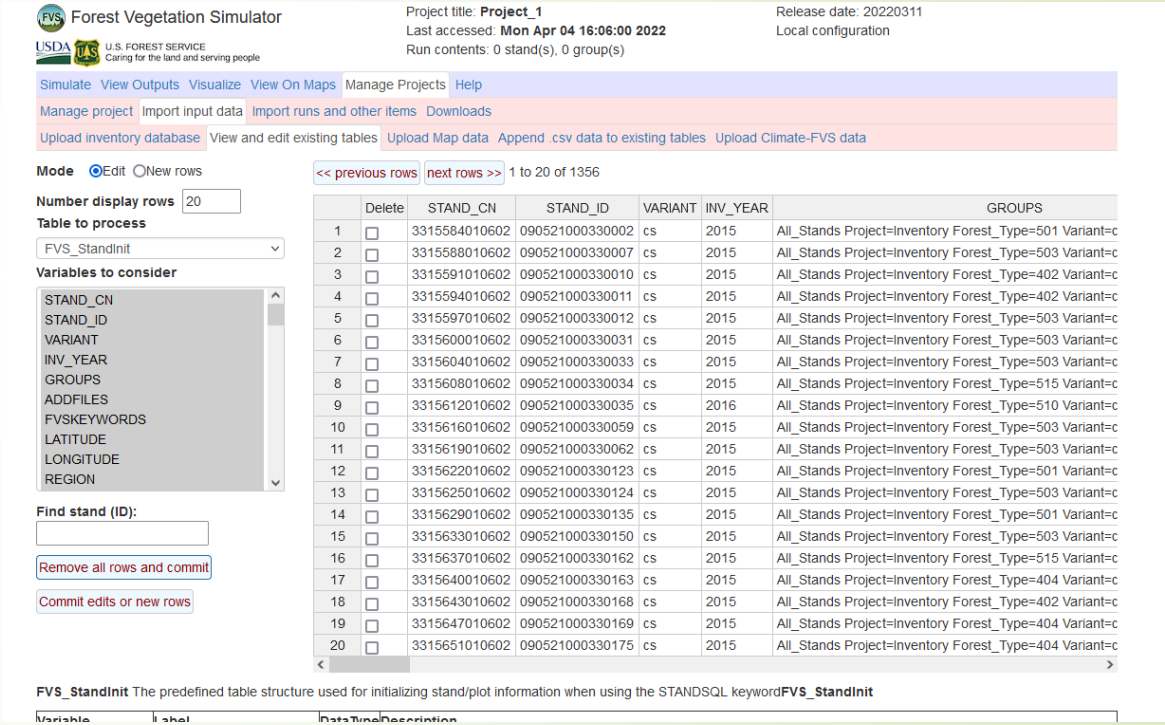
Stand level performance mixed

- Generally "good" results
- Some spectacular failures
- Median error
  - Starts at 13%
  - Ends at 10 %



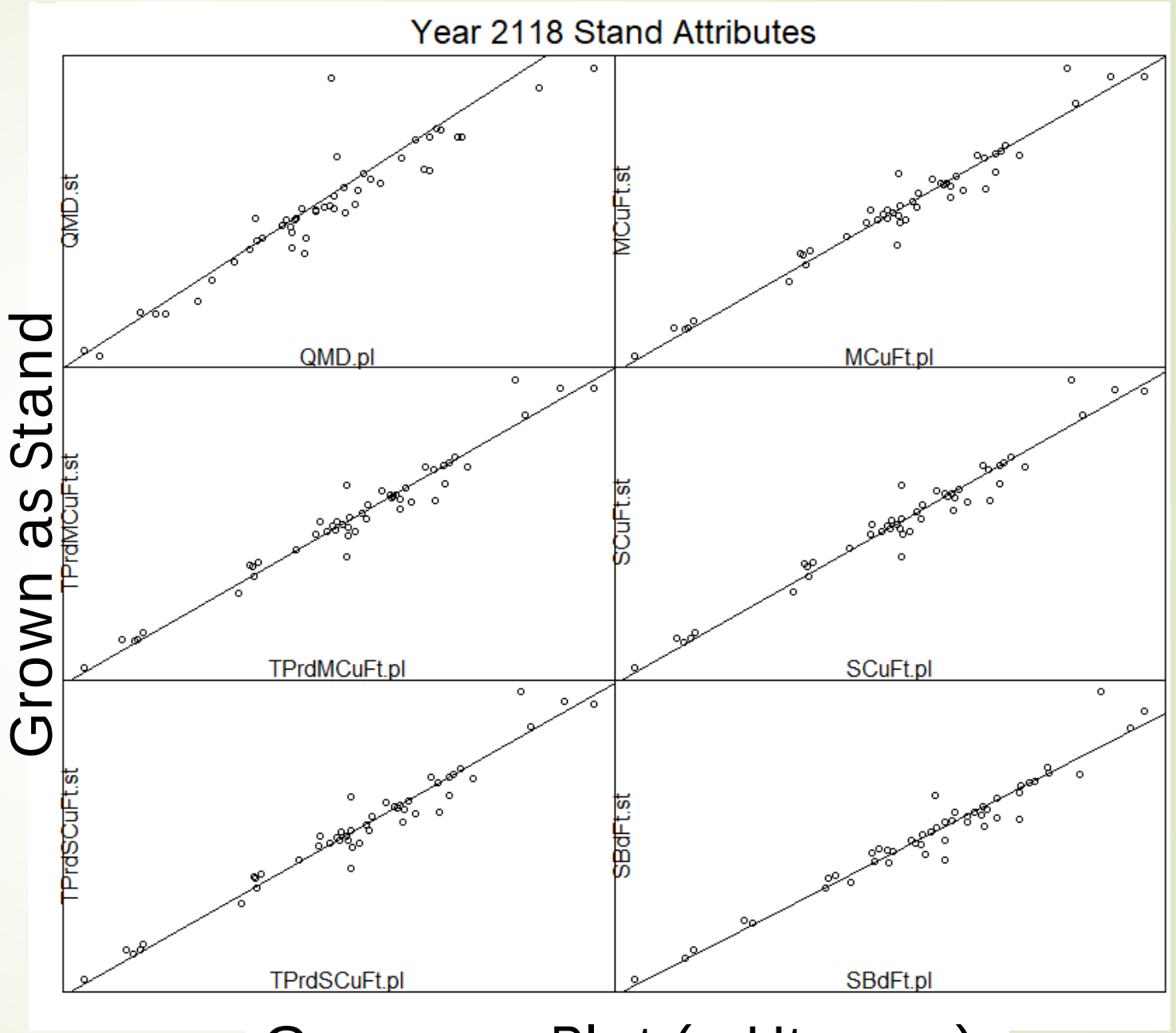
Two More Questions

- How to run the data
  - As plots?
  - As stands?
- Effect of height predictions (e.g., add height errors)
  - Distribution of heights compressed
  - Volume residuals upward biased?

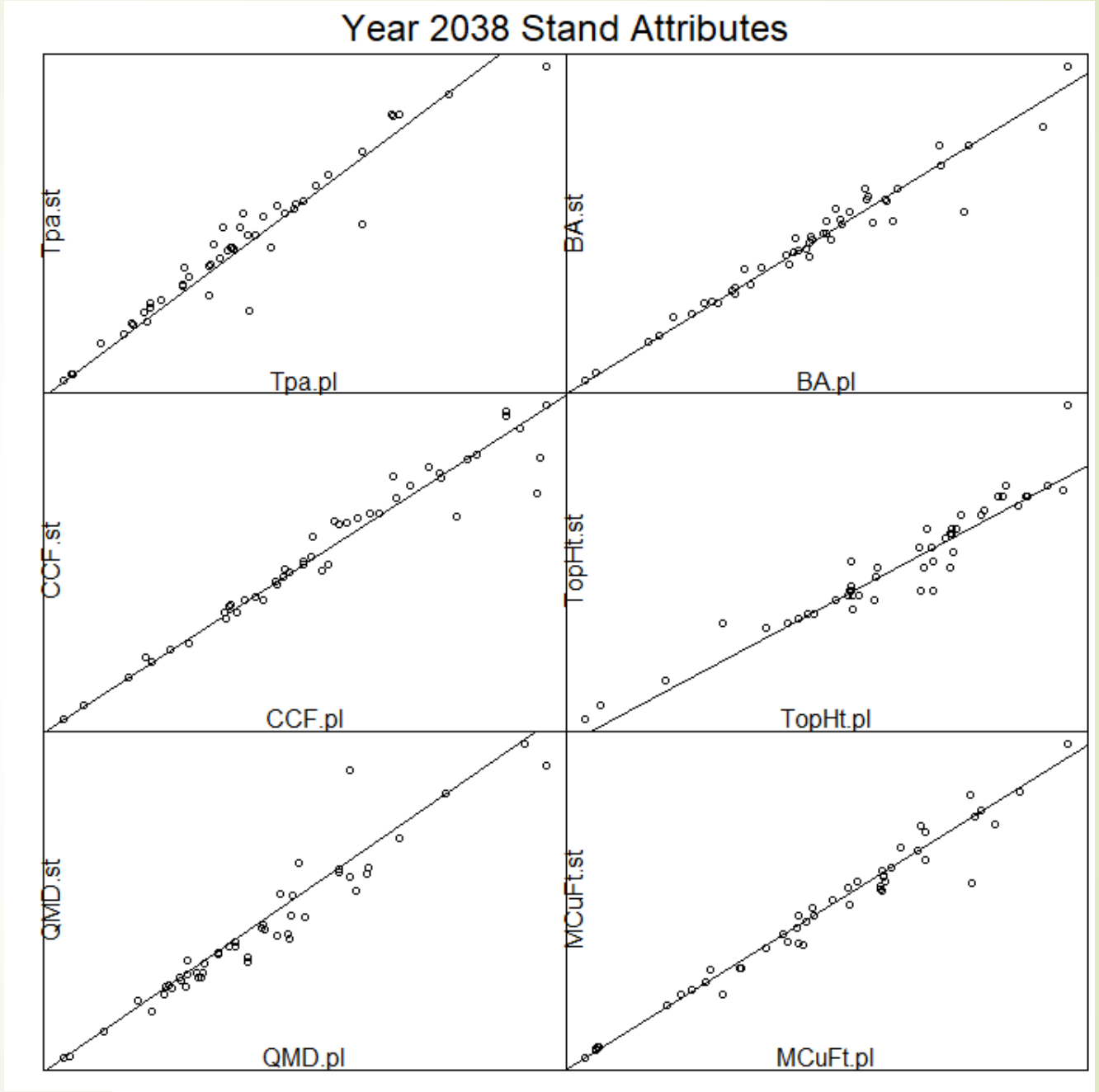


FVS
Effect of Stand vs Plot

- Grow tree
  - 100 years
  - As plots (+error)
  - As stands
- Yes: Stand level differences
- ~Unbiased at 100 years
- No effect from ht errors
- Results same at 2038



Year	mcft_Bias%	mcft_CV%
2018	-0.12	0.59
2038	-1.06	5.00
2058	-5.46	8.28
2078	-0.24	4.51
2098	-0.38	4.64
2118	-0.41	4.85

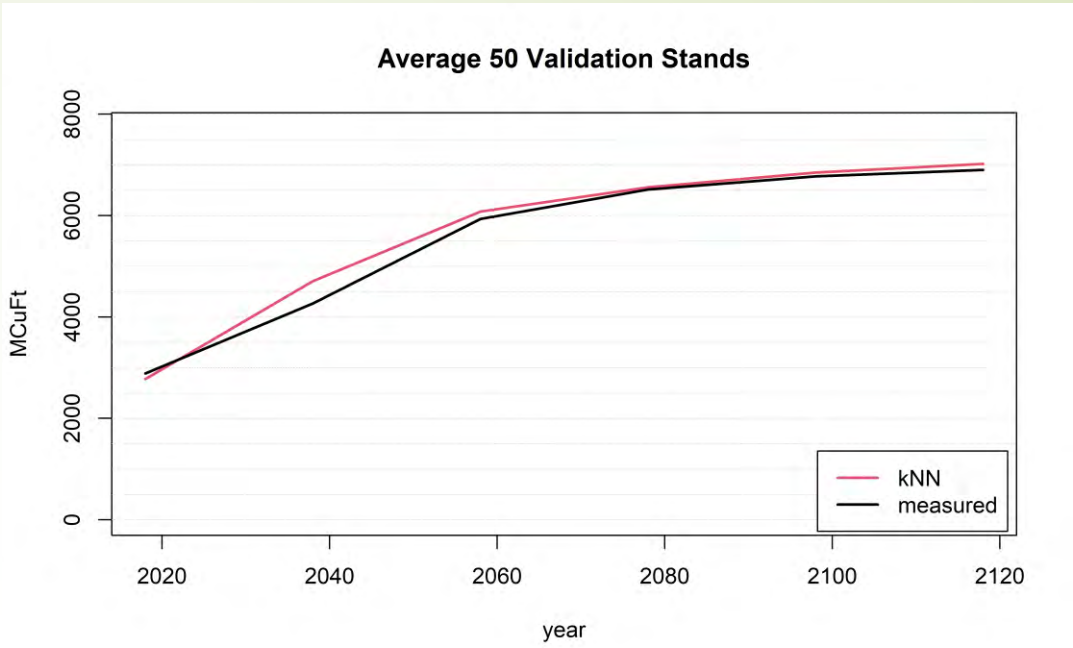


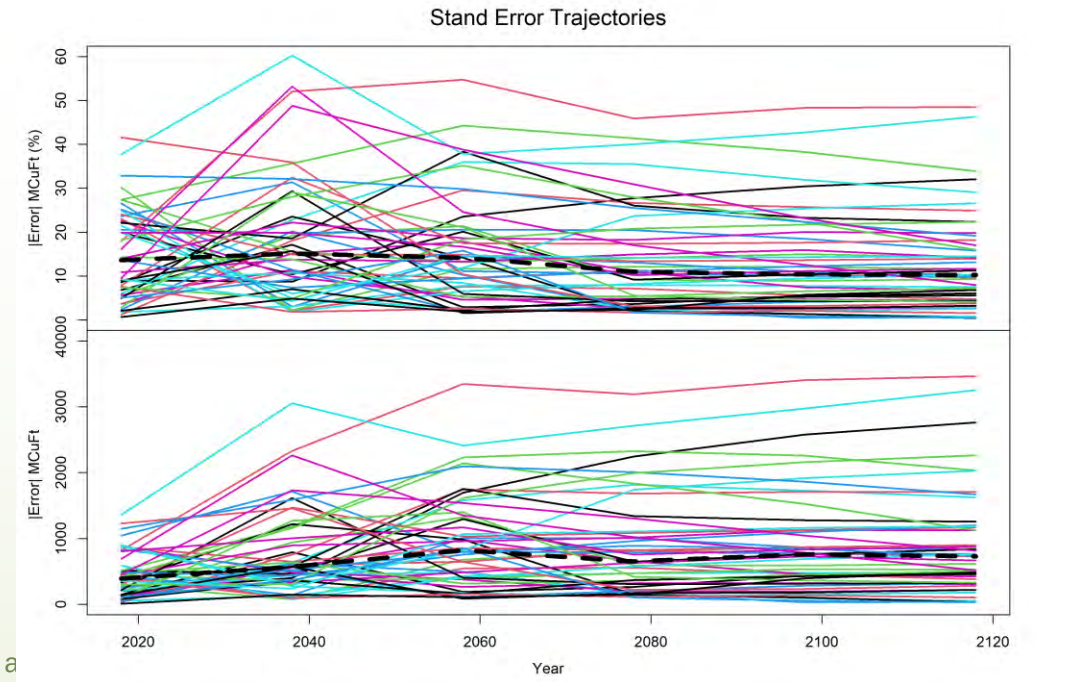
Grown as Stand

Grown as Plot (No Ht error)

Improved kNN strategies?

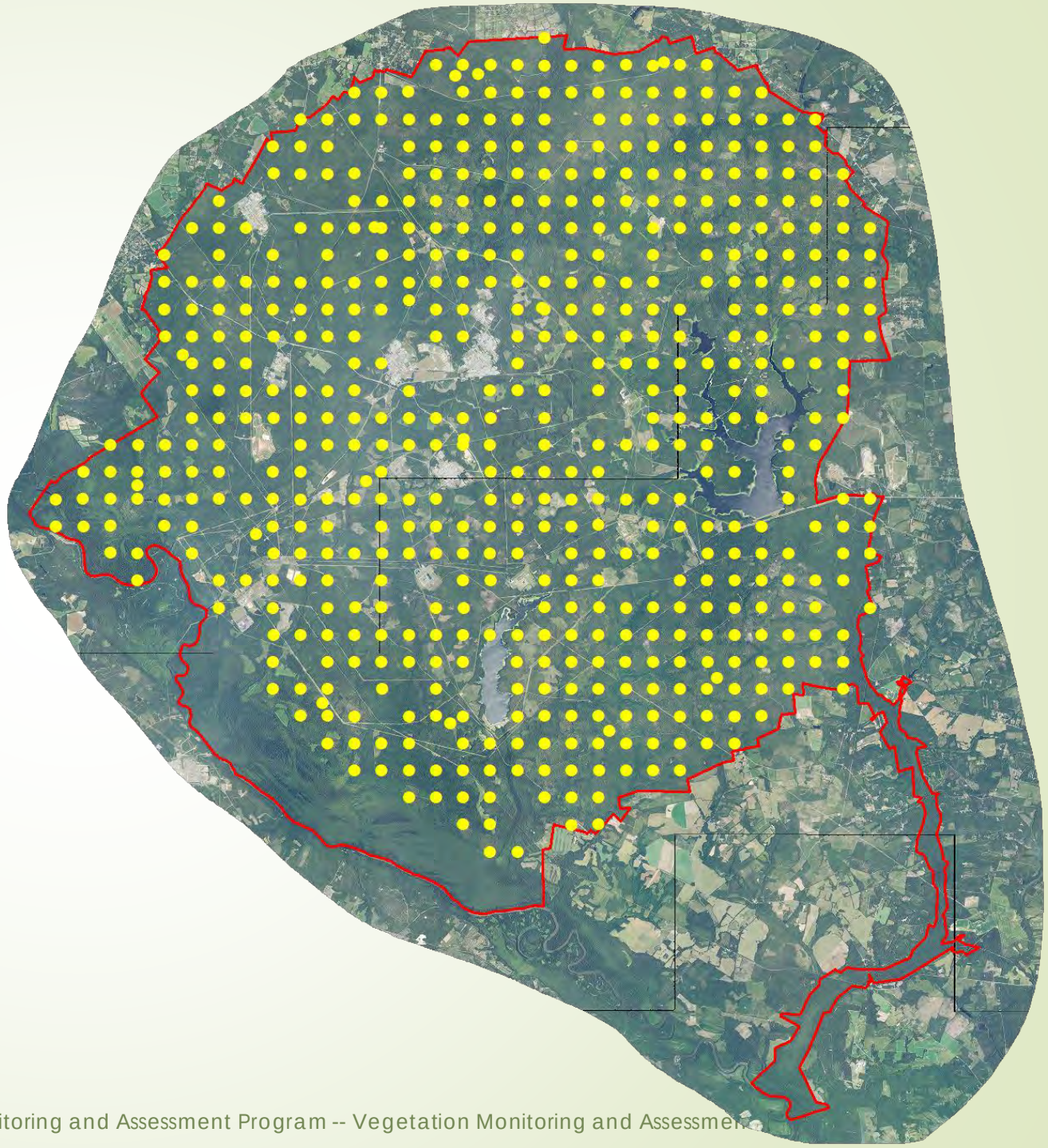
- Impute on predicted attributes (2118) ?
  - results worse
- Match stand level X variation?
- Pre-stratify based on species composition





Whats next?

- Diagnose "failed" stands
  - QMD?
  - Species?
- Grow lidar inventory
  - CFI grid (measured)
  - CFI grid (predictions)
  - kNN Tract
- Evaluate agreement
  - CFI as benchmark
  - Compare product distributions



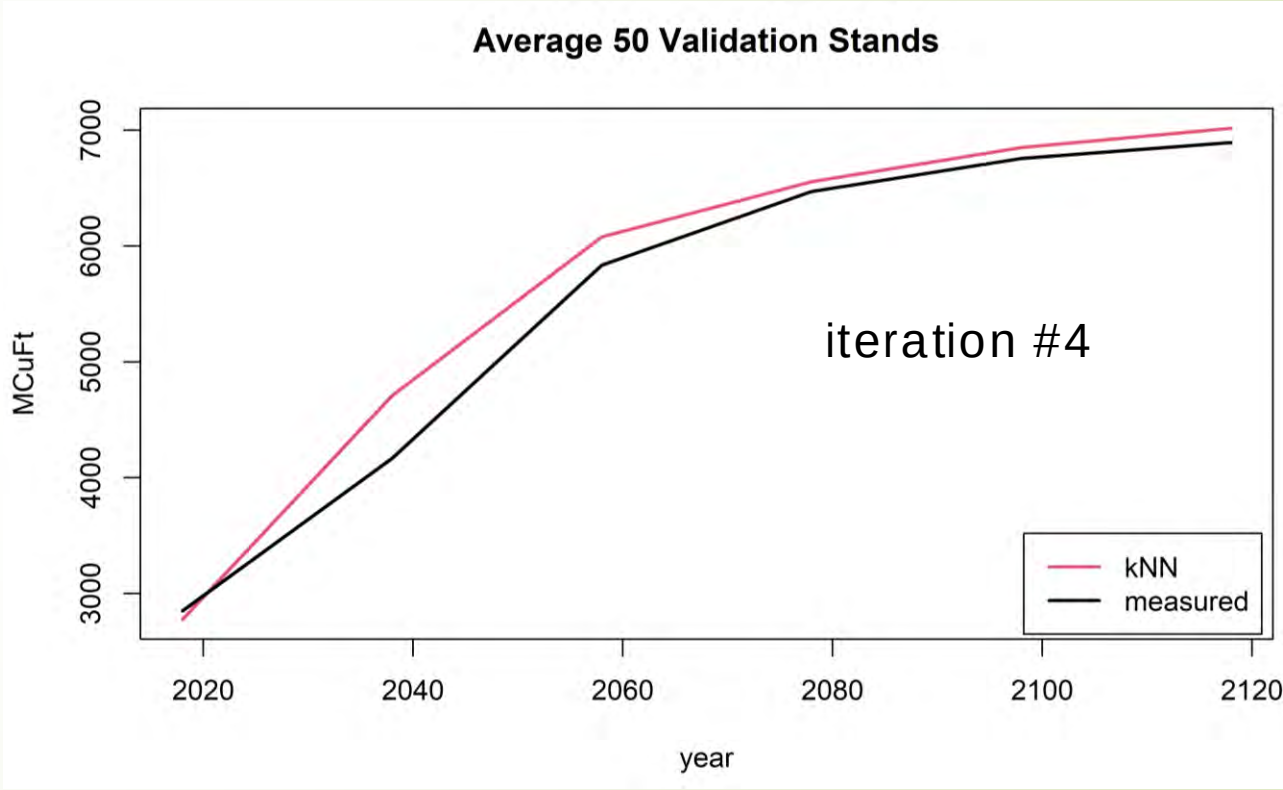
Thank you!

- Questions?

Old slides

Study Overview

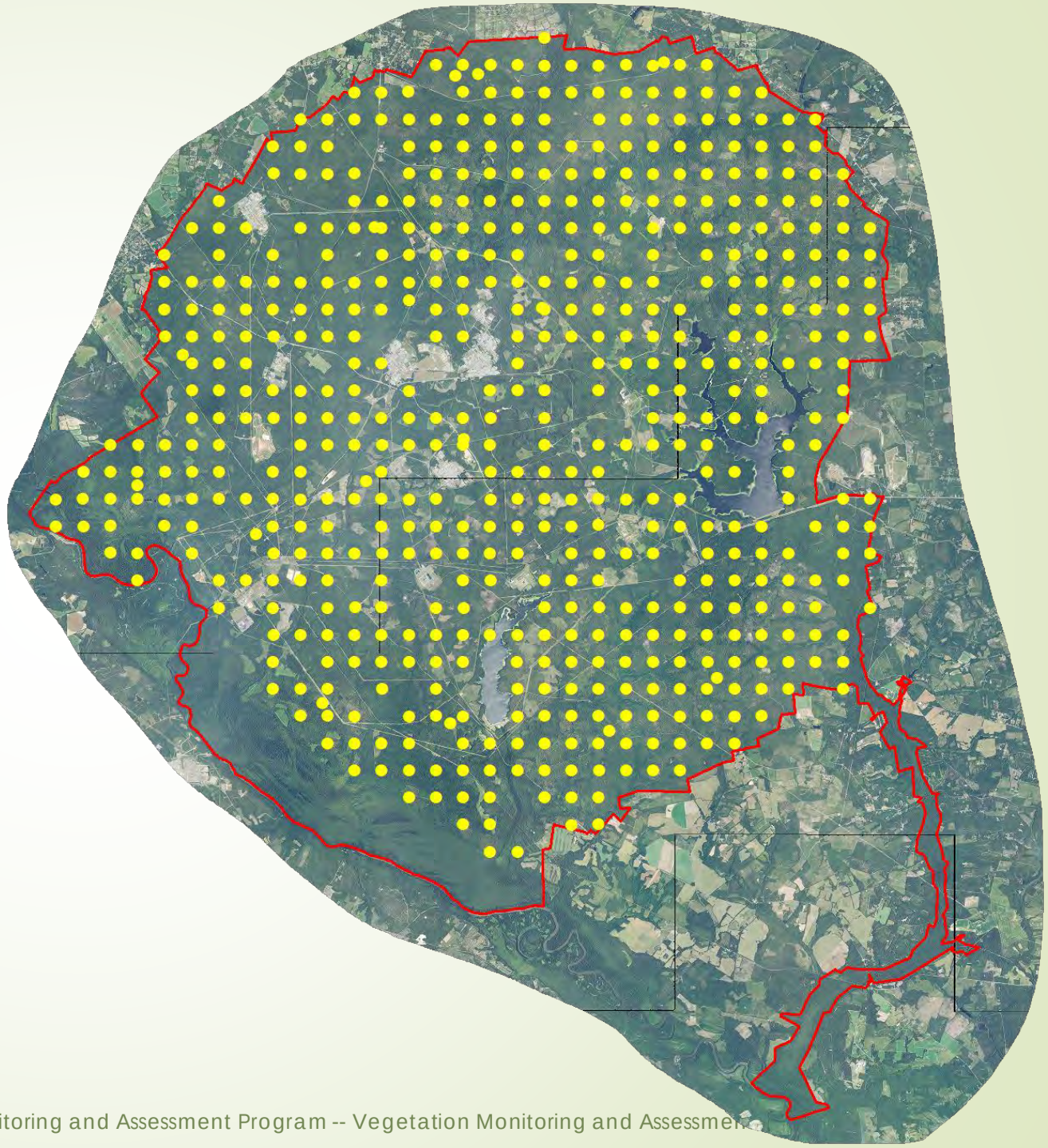
- Lidar Inventory with 500+ Gridded plots
- 50 validation stands
- Evaluate LIDAR FVS projections
  - Predicted (imputed) stands
  - Observed stand measurements



Sample Design

- 548 Plot Fixed Radius Plots
- 50 Validation Stands
- Lidar (leaf off)

- Objectives:
  - Explore modeling alternatives
  - Model hw/sw, components
  - Build functional inventory
  - Growth projection



Generating Tree Lists

- kNN to find plots