Potatoes

```
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.5.1     v tibble    3.2.1
v lubridate 1.9.3     v tidyr     1.3.1
v purrr     1.0.2
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becor
```

```
library(spmodel)
library(vroom)
```

```
Attaching package: 'vroom'

The following objects are masked from 'package:readr':

    as.col_spec, col_character, col_date, col_datetime, col_double,
    col_factor, col_guess, col_integer, col_logical, col_number,
    col_skip, col_time, cols, cols_condense, cols_only, date_names,
    date_names_lang, date_names_langs, default_locale, fwf_cols,
    fwf_empty, fwf_positions, fwf_widths, locale, output_column,
    problems, spec
```

```
library(viridis)
```

```
Loading required package: viridisLite
```

```
library(FRK)
```

```
Warning: package 'FRK' was built under R version 4.4.3
```

```

Attaching package: 'FRK'

The following object is masked from 'package:stats':

    simulate
```

```
library(fields)
```

```
Loading required package: spam
Spam version 2.11-0 (2024-10-03) is loaded.
Type 'help( Spam)' or 'demo( spam)' for a short introduction
and overview of this package.
Help for individual functions is also obtained by adding the
suffix '.spam' to the function name, e.g. 'help( chol.spam)'.

Attaching package: 'spam'

The following objects are masked from 'package:base':

    backsolve, forwardsolve


Try help(fields) to get started.
```

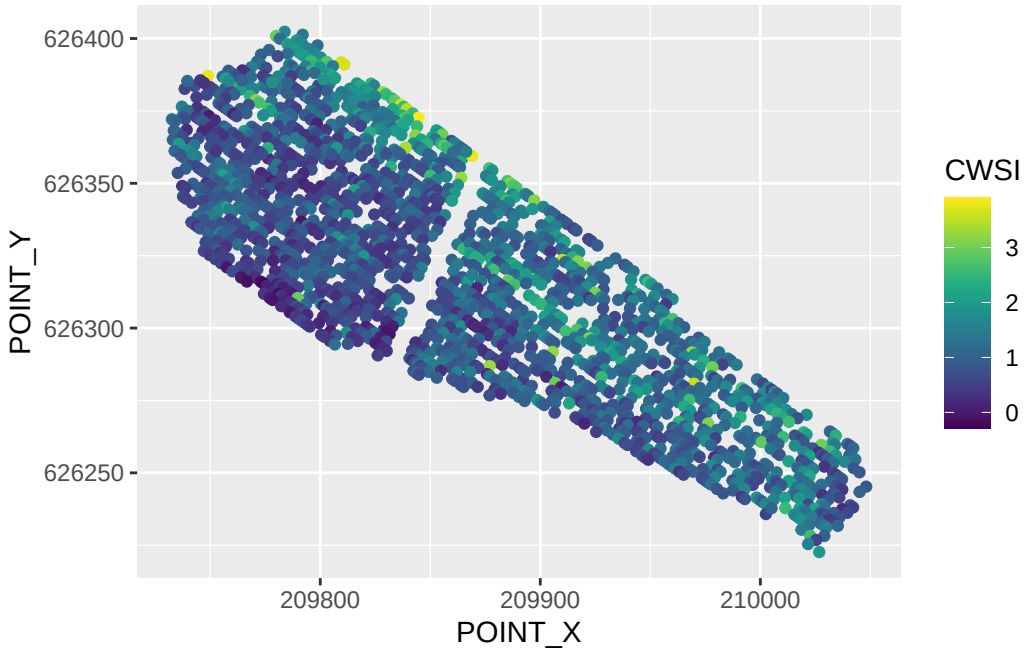
```
df <- vroom('PotatoCWSI.csv')
```

```
Rows: 3889 Columns: 8
-- Column specification --------------------------------------------------------
Delimiter: ","
dbl (8): POINT_X, POINT_Y, SLOPE, TWI, ASPECT, ECA_SHALLOW, NDVI, CWSI

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

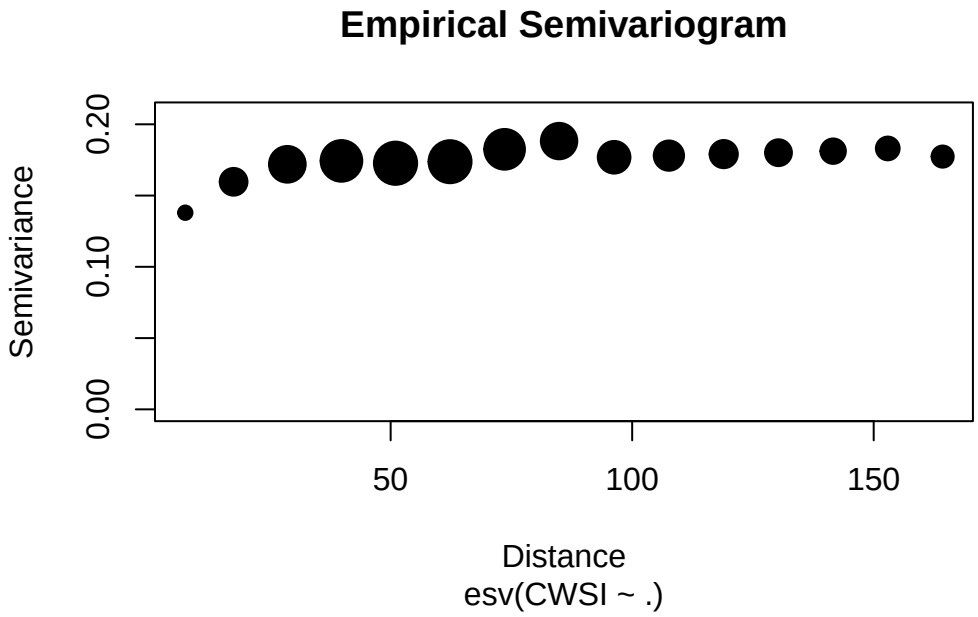
```
obs <- df %>%
  filter(!is.na(CWSI))
new <- df %>%
  filter(is.na(CWSI))
```

EDA

```
obs %>%
  ggplot(aes(x=POINT_X,y=POINT_Y, color=CWSI)) +
  geom_point() +
  scale_color_viridis()
```




```
# variogram
variogram <- esv(CWSI~., data=obs, xcoord=POINT_X, ycoord=POINT_Y)
plot(variogram)
```

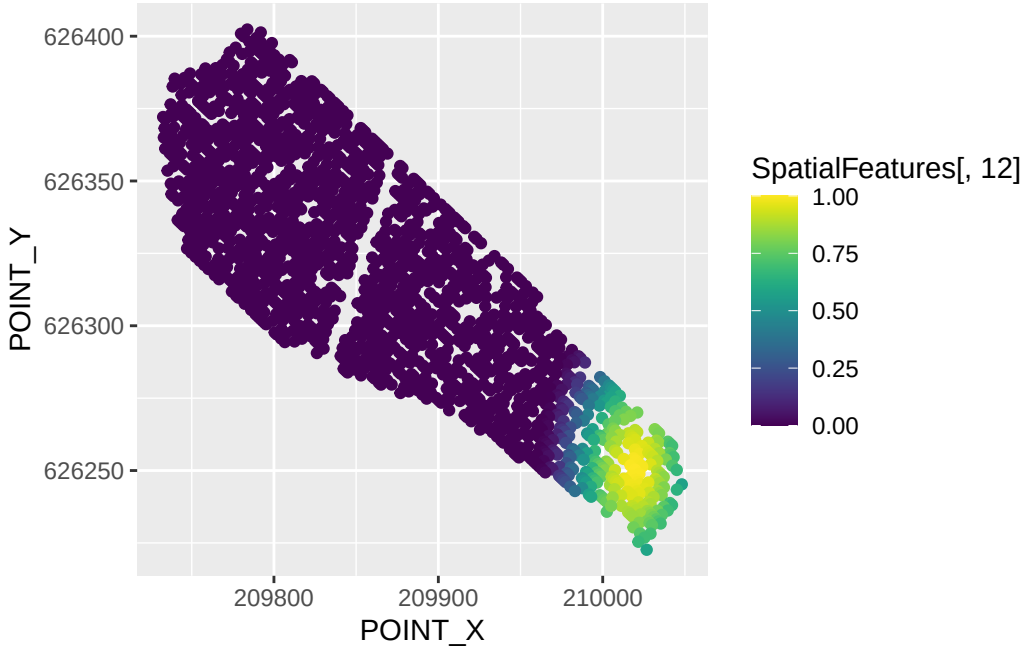


Feature Engineering

```
# Make obs loc matrix
observed_loc_matrix <- matrix(data=cbind(obs$POINT_X,obs$POINT_Y), ncol=2)

## Number of Spatial Features
K <- 20

## "Centers" of Features
## cover.design() choose K centers spread out over spatial region
centers <- cover.design(observed_loc_matrix, nd=K)$design

## Use 1.5 rule of thumb for scale
dist_between_centers <- rdist(centers)
the_scale <- 1.5*max(apply(dist_between_centers,1,function(x){sort(x)[2]}))

## Define spatial features
spatial_features <- local_basis(
manifold=plane(),
loc=centers,
scale=rep(the_scale, K),
```

```
type="bisquare" #or "Gaussian" or "exp"
)

## Mutate new spatial features in data frame
obs <- obs %>%
  mutate(SpatialFeatures=as.matrix(eval_basis(spatial_features, observed_loc_matrix)))
```

```
# Plot
obs %>%
  ggplot(aes(x=POINT_X,y=POINT_Y, color=SpatialFeatures[,12])) +
  geom_point() +
  scale_color_viridis()
```



Model

```
obs.sp <- splm(CWSI~SLOPE+TWI+ASPECT+ECA_SHALLOW+NDVI+SpatialFeatures[,1]+SpatialFeatures[,2]
xcoord=POINT_X, ycoord=POINT_Y)
summary(obs.sp)
```

```
Call:
splm(formula = CWSI ~ SLOPE + TWI + ASPECT + ECA_SHALLOW + NDVI +
    SpatialFeatures[, 1] + SpatialFeatures[, 2] + SpatialFeatures[,
    3] + SpatialFeatures[, 4] + SpatialFeatures[, 5] + SpatialFeatures[,
    6] + SpatialFeatures[, 7] + SpatialFeatures[, 8] + SpatialFeatures[,
    9] + SpatialFeatures[, 10] + SpatialFeatures[, 11] + SpatialFeatures[,
    12] + SpatialFeatures[, 13] + SpatialFeatures[, 14] + SpatialFeatures[,
    15] + SpatialFeatures[, 16] + SpatialFeatures[, 17] + SpatialFeatures[,
    18] + SpatialFeatures[, 19] + SpatialFeatures[, 20], data = obs,
    spcov_type = "exponential", xcoord = POINT_X, ycoord = POINT_Y)

Residuals:
    Min      1Q  Median      3Q     Max
-2.2543 -0.4032 -0.1314  0.1753  2.0094

Coefficients (fixed):
                        Estimate Std. Error z value Pr(>|z|)
(Intercept)            3.0025855  0.7524782   3.990  6.6e-05 ***
SLOPE                  0.0059379  0.0177375   0.335    0.738
TWI                    0.0023441  0.0133474   0.176    0.861
ASPECT                 0.0002179  0.0003477   0.627    0.531
ECA_SHALLOW            0.0005355  0.0052826   0.101    0.919
NDVI                  -4.8873120  0.1350701 -36.184  < 2e-16 ***
SpatialFeatures[, 1]   0.7298364  0.7616013   0.958    0.338
SpatialFeatures[, 2]   0.0544470  0.6432089   0.085    0.933
SpatialFeatures[, 3]  -0.0581519  0.6840561  -0.085    0.932
SpatialFeatures[, 4]  -0.0274428  0.8541085  -0.032    0.974
SpatialFeatures[, 5]   0.4862680  0.5870894   0.828    0.408
SpatialFeatures[, 6]  -0.5598174  0.9798524  -0.571    0.568
SpatialFeatures[, 7]  -0.6117252  0.6037053  -1.013    0.311
SpatialFeatures[, 8]  -0.5796139  0.7498070  -0.773    0.440
SpatialFeatures[, 9]  -0.1678815  0.6721241  -0.250    0.803
SpatialFeatures[, 10]  0.1274221  0.8309479   0.153    0.878
SpatialFeatures[, 11] -0.0415590  0.5812196  -0.072    0.943
SpatialFeatures[, 12]  0.2837621  0.6381664   0.445    0.657
SpatialFeatures[, 13] -0.0502747  0.5399202  -0.093    0.926
SpatialFeatures[, 14]  0.9546484  0.6939404   1.376    0.169
SpatialFeatures[, 15]  0.4152727  0.6339386   0.655    0.512
SpatialFeatures[, 16]  0.8469551  0.5781594   1.465    0.143
SpatialFeatures[, 17]  0.0283579  0.7525846   0.038    0.970
SpatialFeatures[, 18]  0.0494055  0.8215509   0.060    0.952
SpatialFeatures[, 19]  0.5591974  0.9579389   0.584    0.559
SpatialFeatures[, 20] -0.1740933  0.8850562  -0.197    0.844
```

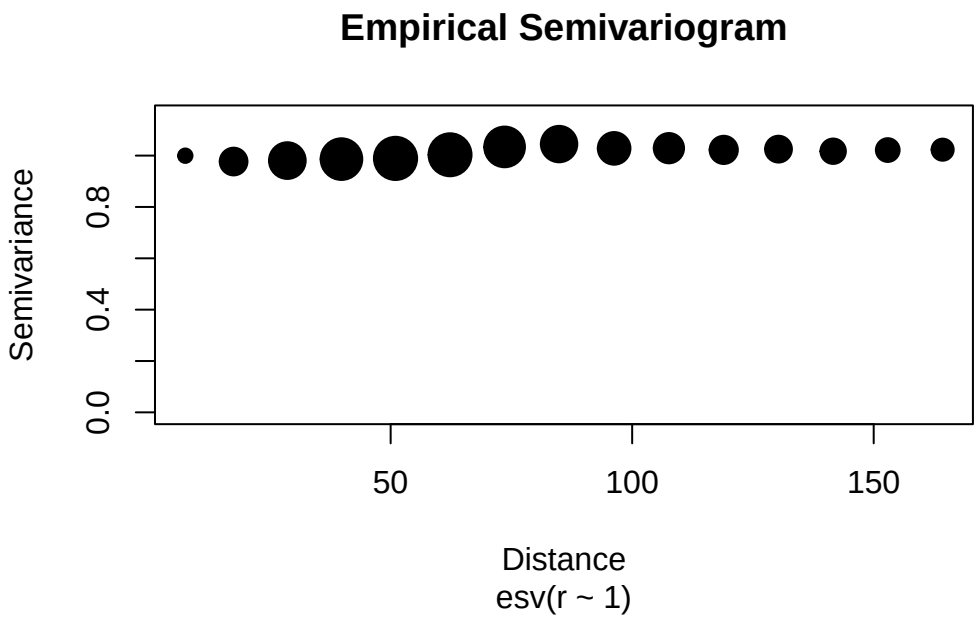
```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Pseudo R-squared: 0.4083

Coefficients (exponential spatial covariance):
       de        ie     range
  1.06403   0.09373 138.86010
```

```
# obs.sp2 <- splm(CWSI ~ SLOPE+TWI+ASPECT+ECA_SHALLOW+NDVI+SpatialFeatures, data = obs,
#                 spcov_type = 'exponential', xcoord = POINT_X, ycoord = POINT_Y)
# summary(obs.sp2)
```

Validation

Line Assumptions

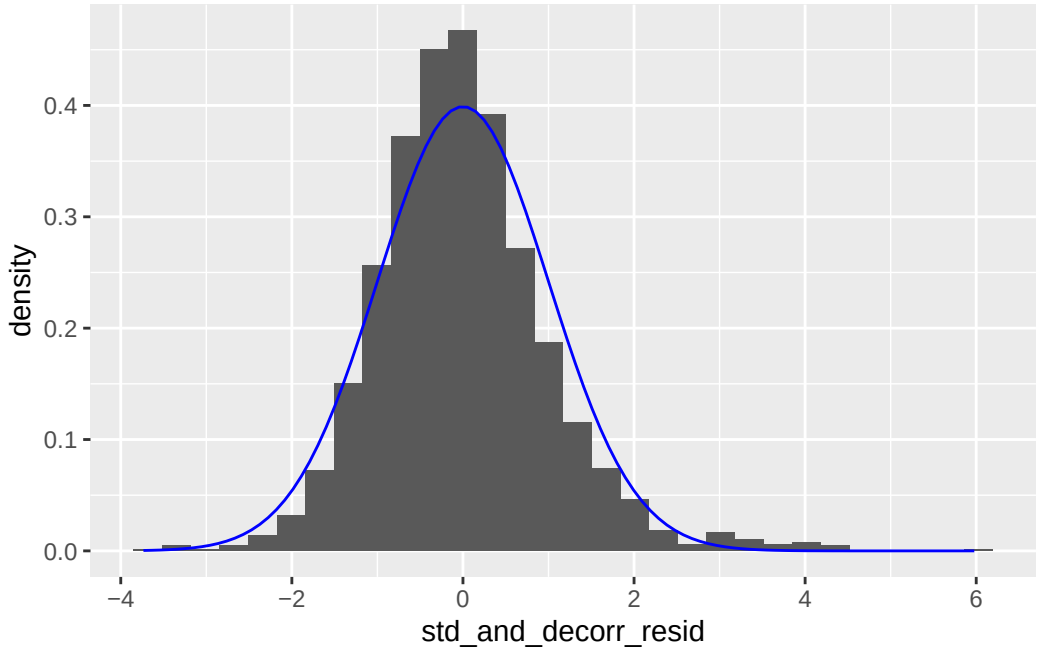
```
std_and_decorr_resid <- rstandard(obs.sp)
# Independence
resids.sp <- data.frame(r=std_and_decorr_resid,
                        x = obs$POINT_X,
                        y = obs$POINT_Y)
variogram_sp <- esv(r~1, data=resids.sp, xcoord=x, ycoord=y)
plot(variogram_sp)
```



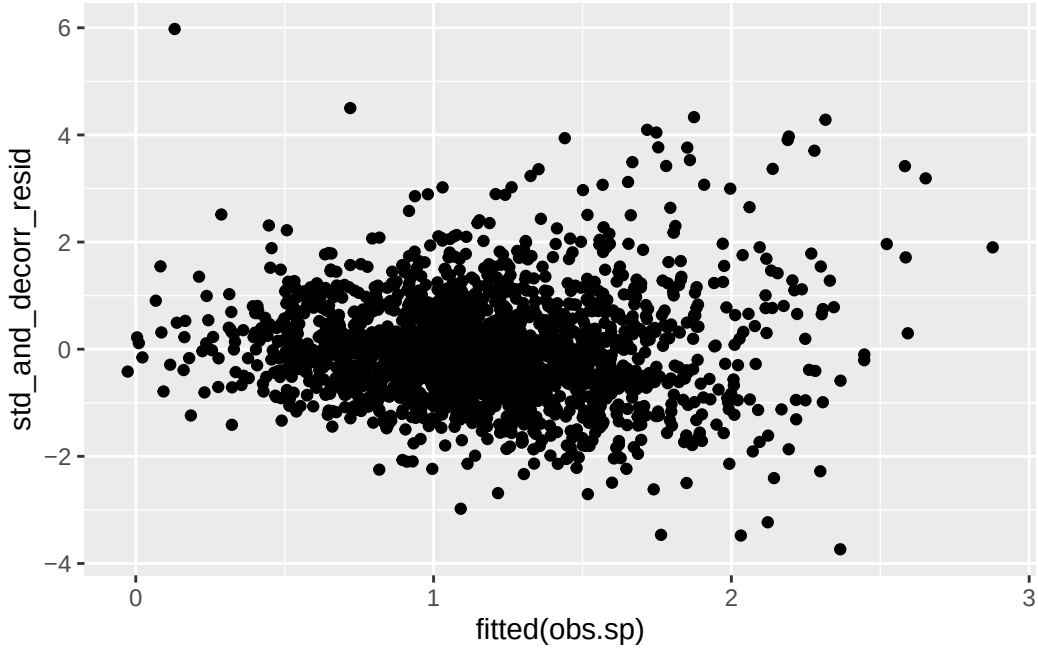

```
# Normality
ggplot() +
  geom_histogram(mapping = aes(x=std_and_decorr_resid, y=..density..)) +
  stat_function(fun = dnorm, color='blue')
```

```
Warning: The dot-dot notation (`..density..`) was deprecated in ggplot2 3.4.0.
i Please use `after_stat(density)` instead.
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

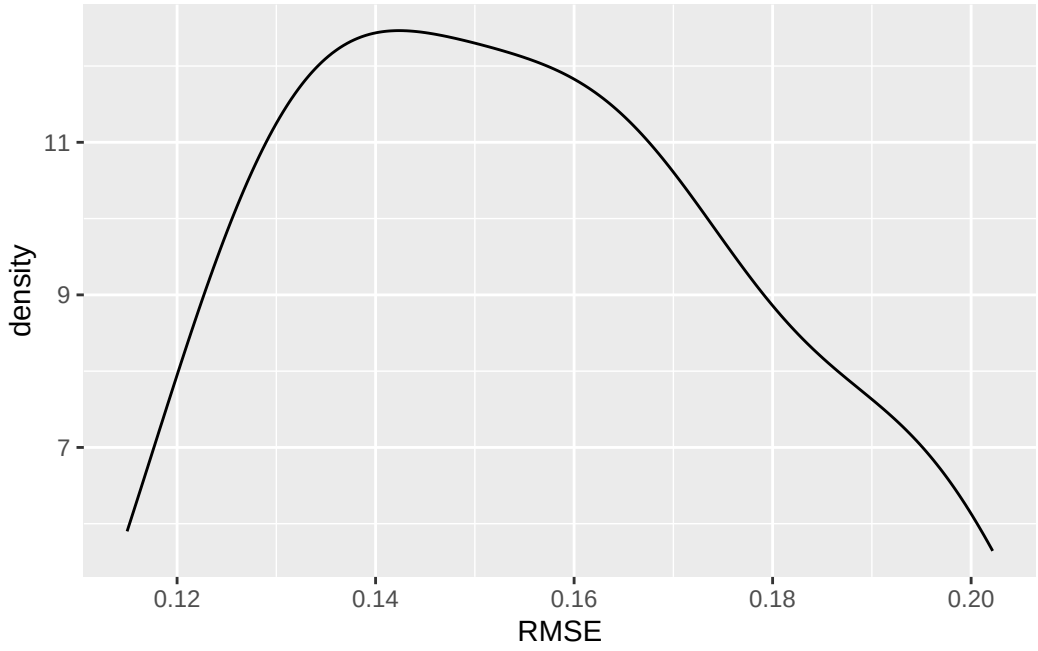



```
# Equal Variance
ggplot() +
  geom_point(mapping = aes(y=std_and_decorr_resid, x=fitted(obs.sp)))
```



Cross Validation

```
cv_sp <- function(fold_num){
  ## Split into train-validation sets
  validationSet <- obs %>%
    filter(folds==fold_num)
  trainSet <- obs %>%
    filter(folds!=fold_num)

  # Ensure factors have at least two levels (drop unused levels)
  trainSet <- droplevels(trainSet)
  validationSet <- droplevels(validationSet)

  ## Fit a model and predict
  trainModel <- splm(CWSI~SLOPE+TWI+ASPECT+ECA_SHALLOW+NDVI+SpatialFeatures[,1]+SpatialFeatu
xcoord=POINT_X, ycoord=POINT_Y, local=TRUE)
  preds <- predict(trainModel, newdata=validationSet)

  ## Validate the predictions
  rmse <- yardstick::rmse_vec(truth=validationSet$CWSI, estimate=preds)

  return(rmse)
}

K <- 20
folds <- rep(1:K, length = nrow(obs)) %>% #Rep K folds n times
  sample()
## CV for spatial linear model w/ spatial features
rmse_results_splm <- sapply(1:K, FUN=cv_sp) %>%
  unlist()
rmse_results <- data.frame(RMSE = rmse_results_splm)
ggplot(rmse_results, aes(x=RMSE)) +
  stat_density(geom="line",position="identity")
```




```
# average rmse across folds
mean(rmse_results_splm)
```

```
[1] 0.1569842
```

```
# standard deviation (approximate errror if we just predicted the mean for everything)
sd(obs$CWSI) # about 4 times worse than our model
```

```
[1] 0.6007423
```

R-Squared

```
pseudoR2(obs.sp)
```

```
[1] 0.4082934
```

Save Model

```
saveRDS(obs.sp, 'PotatoModel.rds')
```

Predict to New Locations

```
# Make obs loc matrix
new_loc_matrix <- matrix(data=cbind(new$POINT_X,new$POINT_Y), ncol=2)

## Number of Spatial Features
K <- 20

## "Centers" of Features
## cover.design() choose K centers spread out over spatial region
centers <- cover.design(new_loc_matrix, nd=K)$design

## Use 1.5 rule of thumb for scale
dist_between_centers <- rdist(centers)
the_scale <- 1.5*max(apply(dist_between_centers,1,function(x){sort(x)[2]}))

## Define spatial features
spatial_features <- local_basis(
manifold=plane(),
loc=centers,
scale=rep(the_scale, K),
type="bisquare" #or "Gaussian" or "exp"
)

## Mutate new spatial features in data frame
new <- new %>%
  mutate(SpatialFeatures=as.matrix(eval_basis(spatial_features, new_loc_matrix)))
```

```
# Read in model
sp_model <- readRDS('PotatoModel.rds')

# Save predictions
new$preds <- predict(sp_model, newdata = new)

# Plot predictions
```

```
new %>%
  ggplot(aes(x=POINT_X,y=POINT_Y, color = preds)) +
  geom_point() +
  scale_color_viridis()
```