Model Creation: Enviornmental Justice Index Training Random Forest and Geographically Weighted Random Forest Models

Thesis for a Master of Public Health, Epidemiology

Nathan Garcia-Diaz

Brown University, School of Public Health

August 21, 2024

Statement of Purpose

The purpose of the file is to build two final models: a traditional random forest model (RF) and a geographically weighted random forest model (GWFRF). The following two sentence provide a overarching description of the two models. In a RF model, each tree in the forest is built from a different bootstrap sample of the training data, and at each node, a random subset of predictors (features) is considered for splitting, rather than the full set of predictors. A GWRF model expands on this concept by incorporating spatial information by weighting the training samples based on their geographic proximity to the prediction location. The splitting process in a RF model is determined by the mean squared error and in a GWRF is influenced by the spatial weights (i.e., weighted mean squared error), which adjust the contribution of each sample based on its geographic distance.

Overview of Hyperparameters Definitions

In James et al 2021, Ch 8.2.2 Random Forests, James et al 2023, Ch 15.2 Definition of Random Forests and Garson 2021, Ch 5 Random Forest, the others highlight shared parameters between the RF and GWRF models:

- **Number of randomly selected predictors**: This is the number of predictors (p) considered for splitting at each node. It controls the diversity among the trees. A smaller m leads to greater diversity, while a larger m can make the trees more similar to each other.
    - for regression this defaults to $p/3$, where p is the total of predictor variables
- **Number of trees**: This is the total number of decision trees in the forest (m). More trees generally lead to a more stable and accurate model, but at the cost of increased computational resources and time.
    - for the `randomForest::randomForest()`, this defaults to 500

Additionally, GWRF involves an extra tuning spatial parameters:

- **Bandwidth parameter**: This controls the influence of spatial weights, determining how quickly the weight decreases with distance. A smaller bandwidth means only very close samples have significant influence, while a larger bandwidth allows more distant samples to also contribute to the model.

Outline of Hyper-parameter Tuning Process

4 RF models will be built, and they differ based on the different hyperparameters: (1) default settings, (2) first tune p, and subsequently tune, then m while keeping p constant, (3) simultaneously tune m and p with a grid search, (4) tuned with Out of Bag MSE Error Rates as described by Garson 2021. Two metrics will be implemented in the tuning process: Root Mean Squared Error and Out of Bag Error Rate.

In Garson 2021, Ch 5 Random Forest, Garson teaches Random Forest Models by using `randomForest::randomForest()`, and in chapter 5.5.9 (pg. 267), he provides methods for tuning both of these parameters simultaneously using the Out of Bag MSE Error Rates. This value is a measure of the prediction error for data points that were not used in training each tree, and it can be written as OOB Error Rate $= \frac{1}{n}\Sigma_{i=1}^{N}(y_i - \hat{y}_i^{\text{OOB}})^2$. $\hat{y}_i^{\text{OOB}}$ is the OOB prediction for the i-th observation, which is obtained by averaging the predictions from only those trees that did not include i in their bootstrap sample. To provide a high-level summary, since each tree in a Random Forest is trained on a bootstrap sample (a random sample with replacement) of the data, approximately one-third of the data is not used for training each tree. This subset of data is referred to as the "out-of-bag" data for that tree, and this value is calculated using the data points that were not included in the bootstrap sample used to build each tree.

Georganos et al (2019) created the `package(SpatialML)`, and subsequently the tuning is made possible by the `SpatialML::grf.bw()` function. The function uses an exhaustive approach (i.e., it tests sequential nearest neighbor bandwidths within a range and with a user defined step, and returns a list of goodness of fit statistics).

4 RF models will be built, and they differ based on the different hyperparameters: (1) default settings; (2) tuned by first tuning *mtry*, with *ntrees* set to default, and subsequently tuning then *ntrees* while keeping the newly defined *mtry* constant and both methods use RMSE as the metric; (3) tuned both *mtry* and *ntrees* with an Exhaustive Grid Search and both methods use RMSE as the metric, (4) tune tuned with Out of Bag MSE Error Rates as described by Garson 2021. For each model, MAE, MSE, RMSE, and R^2 will be calculated and the hyperparameters of the best model will continue onto the GWRF. To provide points of comparison in the GWRF, two additional models will be created. Thus, three GWRF models will be created: (1) default *mtry* and *ntrees* with optimized *bandwidth parameter*, (2) using the previously defined best hyperparameters, (3) using the optimized *bandwidth parameter* in step one, then tuning *mtry*, with *ntrees* set to default. The method for GWRF Model 3 uses Out of Bag Error Rate as the Metric. The same model evaluation metrics will be compared in addition to calculating the residual autocorrelation.

Lastly, the feature importance plots will be generated for the final, and local feature importance plots will also be created.

Preparation

Traditional Random Forest Model

Model Training and Hyperparameter Tuning

Models will be created and compared at the end of the section.

RF Model 1 - Default Settings

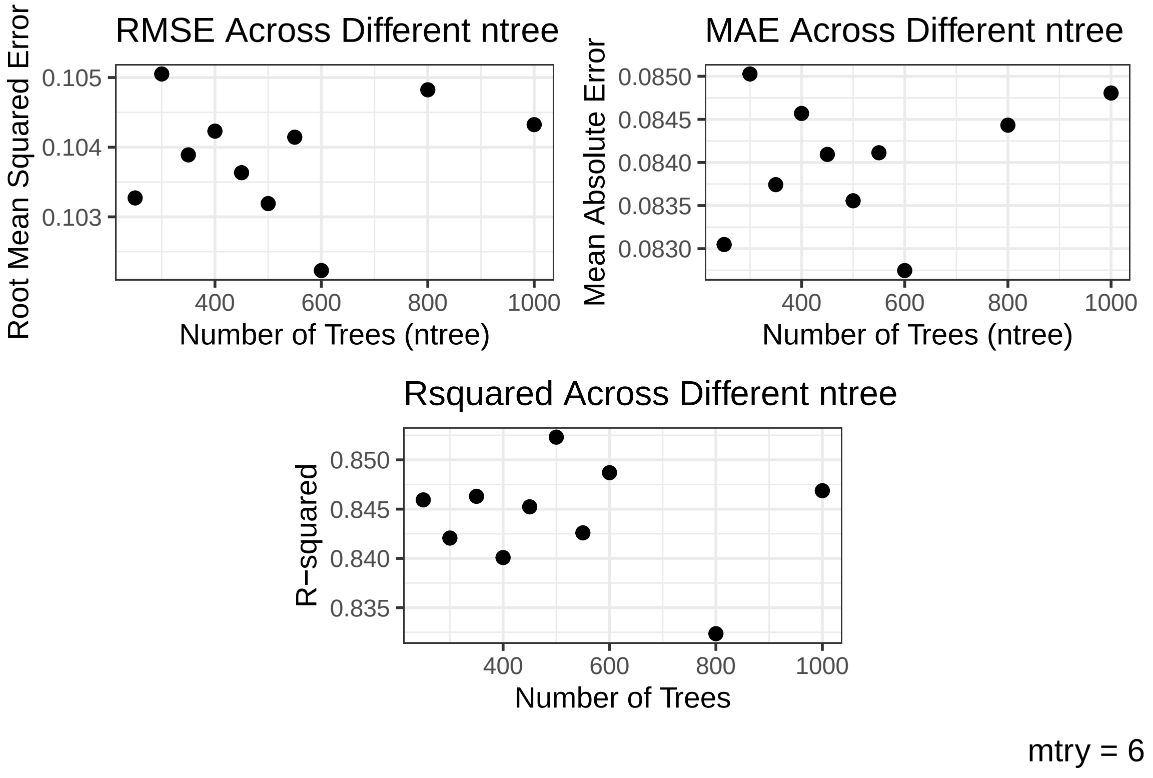
Background: The default settings for the RF model is $mtry$ = p/3, and ntrees = 500, where p is the number of predictors.

```
##
## Call:
##  randomForest(x = x, y = y, ntree = 500, mtry = param$mtry, importance = TRUE)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 12
##
##           Mean of squared residuals: 0.01051627
##                     % Var explained: 83.44
```

RF Model 2 - Sequential Processing With RMSE Metric

Background: This model training process uses a combination of sequential processing and cross-validation. First, tuning the `mtry` parameter by using cross-validation to find the best value for each iteration. The model runs 10 times (i.e., the for loop) because given the nature of the building random forest models, the value of m within the loop changes. Therefore, preforming the function 10 times and taking the average of the most optimal mtry value it calculates and prints the average of the best mtry values. During the second step, the ntree is changing and cross-validated while *mtry* is held constant.

```
##
## Call:
##  randomForest(x = x, y = y, ntree = 800, mtry = param$mtry, importance = TRUE)
##                Type of random forest: regression
##                      Number of trees: 800
## No. of variables tried at each split: 12
##
##           Mean of squared residuals: 0.01075198
##                     % Var explained: 83.07
```

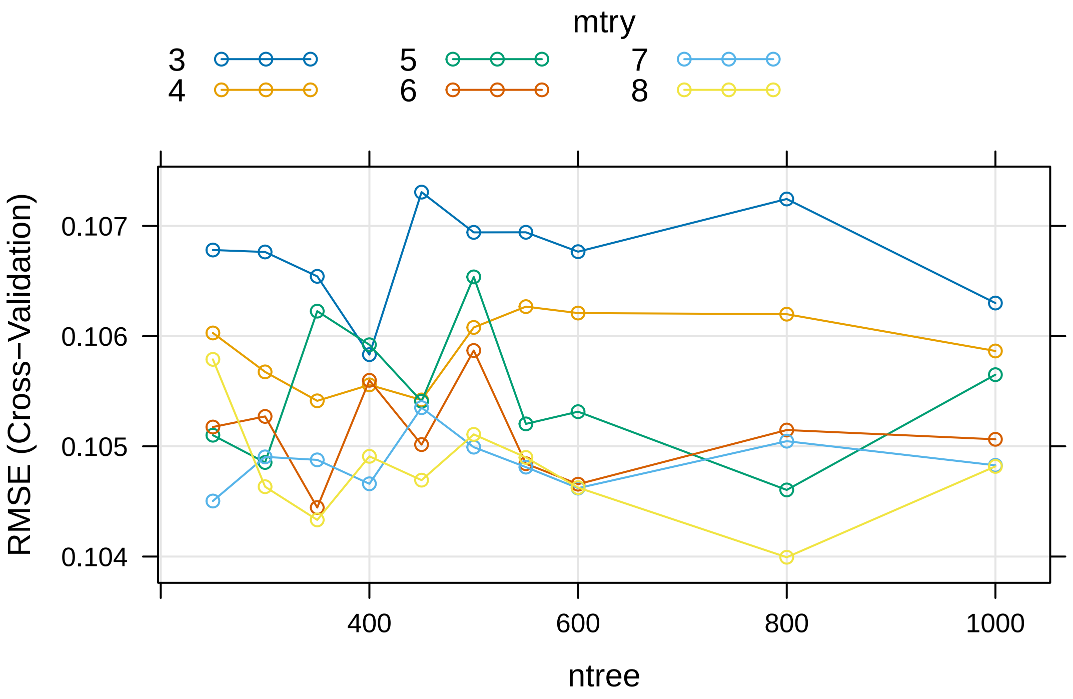


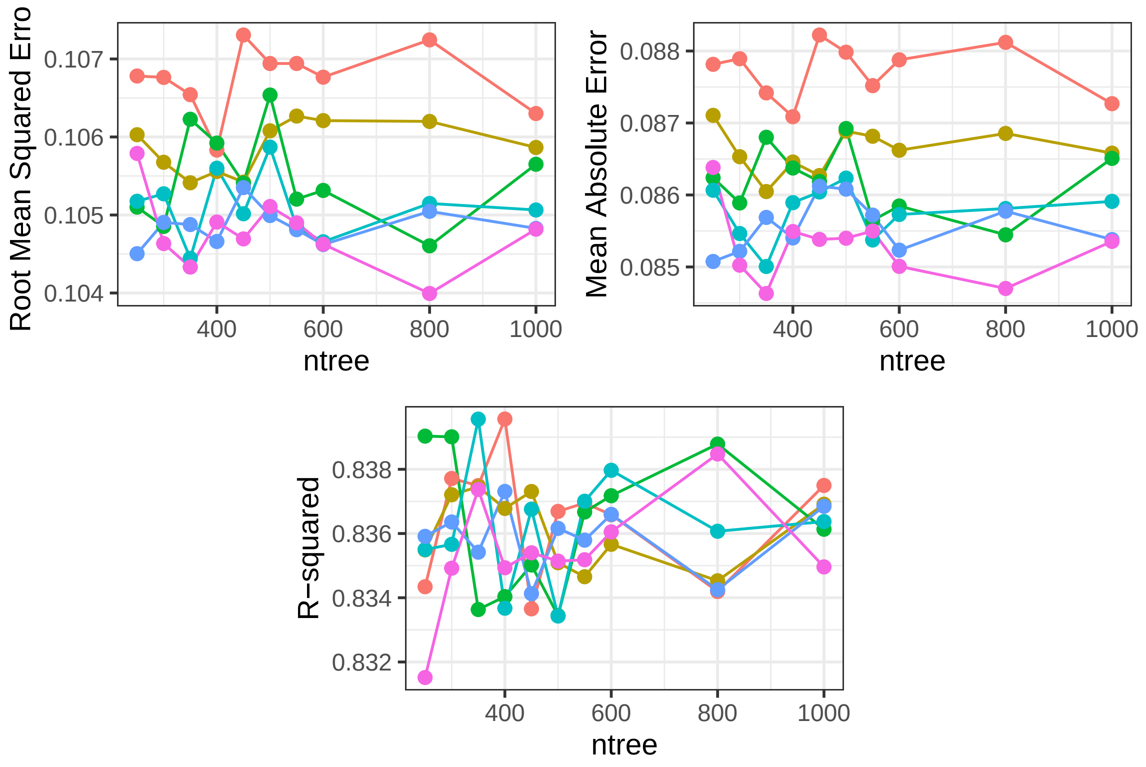
The second model hyperparameters have been set to $mtry = 12$, and $ntrees = 800$.

Model 3 - Exhaustive Grid Search with RMSE as Metric

Background: To preform an exhaustive Grid Search, Brownlee (2020) created a custom function that preforms the grid search. This function checks every combination of *mtry* and *ntree* values determines the final values with RMSE.

```
##
## Call:
##  randomForest(x = x, y = y, ntree = param$ntree, mtry = param$mtry,      importance = TRUE)
##                Type of random forest: regression
##                      Number of trees: 800
## No. of variables tried at each split: 8
##
##           Mean of squared residuals: 0.01074548
##                     % Var explained: 83.08
```





The final values used for the model were $mtry = 8$ and $ntree = 800$. Note that the variation between each of the combinations is minimal.

Model 4

This code snippet is designed to optimize the hyperparameters mtry and ntree in a Random Forest model and by examining the OOB MSE across these combinations, the code identifies which parameters yield the lowest error, helping to optimize the Random Forest model. Here's how the code meets this objective:

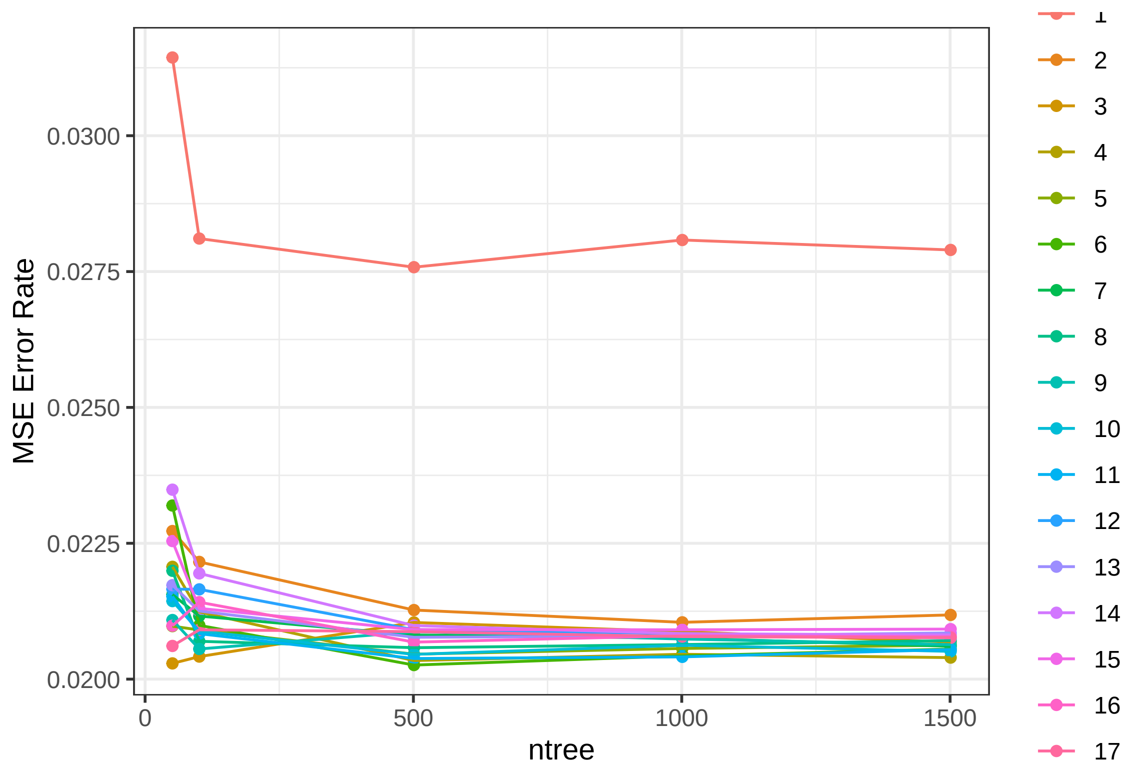
- Iterative Search for mtry: The mtry_iter function generates an iterable sequence of mtry values, starting from 1 up to the number of predictors, incremented by a step factor. This allows the code to explore different numbers of predictors used at each split in the trees.
- Specification of ntree Values: A predefined vector vntree contains different values for the number of trees to be grown in the forest. This allows the code to assess how the number of trees impacts the model performance.
- Error Calculation Across Hyperparameter Combinations: The tune function performs a grid search over the specified mtry values and the maximum number of trees specified in vntree. For each combination, the function trains a Random Forest model and calculates the OOB error rate (MSE if y is continuous).
- Parallel Processing: The foreach loop with the .dopar argument allows for parallel execution of the grid search, which speeds up the computation.
- Result Aggregation: The results are combined into a data frame, which can then be analyzed to identify the optimal combination of mtry and ntree that minimizes the OOB error rate.

This approach ensures that both hyperparameters are tuned simultaneously, leading to a more efficient model optimization process. The final model hyperparameters have been set to $m = 9$, and $ntrees = 501$. The graph below illustrates that the errors across the hyperparameters used with this method are very similar.

```r
# create an interaction function to search over different values of mtry
mtry_iter = function(from, to, stepFactor = 1.05){
  nextEl = function(){
    if (from > to) stop('StopIteration')
    i = from
    from <<- ceiling(from * stepFactor)
    i
  }
  obj = list(nextElem = nextEl)
  class(obj) = c('abstractiter', 'iter')
  obj
}

# create a vector of ntree values of interest
vntree = c(51, 101, 501, 1001, 1501)

# specify the predictor (x) and outcome (y) object
x = df %>% select(starts_with("e_")) %>% st_drop_geometry()
y = df %>% pull(rpl_eji)

# Create a function to get random forest error information for different mtry values
tune = function(x, y, ntree = vntree, mtry = NULL, keep.forest = FALSE, ...) {
  # Define the combination function to aggregate results
  comb = function(a, b) {
    if (is.null(a)) return(b)
    rbind(a, b)
  }
  results = foreach(mtry = mtry_iter(1, ncol(x)), .combine = comb, .packages = 'randomForest')
    model = randomForest::randomForest(x, y, ntree = max(ntree), mtry = mtry, keep.forest = FA
    if (is.factor(y)) {
      errors = data.frame(ntree = ntree, mtry = mtry, error = model$err.rate[ntree, 1])
    } else {
      errors = data.frame(ntree = ntree, mtry = mtry, error = model$mse[ntree])
    }
    return(errors)
  }
  return(results)
}

# running the tuning
results = tune(x,y) %>%
  mutate(MSE = error) %>%
  select(-error)

# examinations of other hyperparameters
# table
temp = results %>%
  arrange(MSE) %>%
  head()

kable(temp, caption = "Model 4 Preformance Metrics", digits = 4, align = c("l", "l", "c"))
```

Table 1: Model 4 Preformance Metrics

ntree	mtry	MSE
501	6	0.0203
51	3	0.0203
501	4	0.0203
501	11	0.0204
1501	4	0.0204
1001	11	0.0204



```
##
## Call:
##  randomForest(x = x, y = y, ntree = ..1, mtry = param$mtry, importance = TRUE)
##                Type of random forest: regression
##                      Number of trees: 501
## No. of variables tried at each split: 6
##
##           Mean of squared residuals: 0.01104803
##                     % Var explained: 82.6
```

The final model hyperparameters have been set to $mtry = 6$, and $ntrees = 501$. The graph below illustrates that the errors across the hyperparameters used with this method are very similar.

RF Model Evaluation

Despite variations in the *m* and *ntrees* parameters across different models, the overall prediction performance remains consistent. The relatively low MSE and RMSE values across the models indicate that the predictions are generally close to the actual values. The high R-Squared values suggest that each model explains a significant portion of the variance in the target variable. However, since model 4 produced code that is lowest MSE and RMSE, and highest R-squared value, these are the parameters that will be head contains for the GWRF.

- Mean Absolute Error (MAE):$\frac{1}{n}\Sigma_{i=1}^{n}|y_i - \hat{y}_i|$
- Mean Squared Error (MSE): $\frac{1}{n}\Sigma_{i=1}^{n}(y_i - \hat{y}_i)^2$
- Root Mean Squared Error (RMSE):$\sqrt{\frac{1}{n}\Sigma_{i=1}^{n}(y_i - \hat{y}_i)^2}$
- R-Squared Value: $\frac{\Sigma(y-\hat{y})^2}{\Sigma(y-\bar{y})^2}$

Table 2: Performance Metrics for Each Model

Model	mtry	ntree	MAE	RMSE	R_Squared
Model 1	12	500	0.085	0.105	0.835
Model 2	12	550	0.083	0.103	0.844
Model 3	8	540	0.086	0.105	0.836
Model 4	6	501	0.085	0.105	0.842

Training a Geographically Weighted Random Forest Model

GWRF Model 1

This model has hyperparameters defined with mtry and trees by the default, and optimized bandwith.

```
## Ranger result
##
## Call:
##  ranger(rpl_eji ~ e_ozone + e_pm + e_dslpm + e_totcr + e_npl +      e_tri + e_tsd + e_rmp +
##
## Type:                             Regression
## Number of trees:                  500
## Sample size:                      240
## Number of independent variables:  36
## Mtry:                             12
## Target node size:                 5
## Variable importance mode:         impurity
## Splitrule:                        variance
## OOB prediction error (MSE):       0.01061433
## R squared (OOB):                  0.8335626
##     e_ozone        e_pm     e_dslpm     e_totcr       e_npl       e_tri
##  0.04468540  0.30852189  0.21565943  1.33741723  0.17726889  1.96279490
##       e_tsd       e_rmp      e_coal      e_lead      e_park    e_houage
##  0.02250228  0.31256788  0.00000000  0.00000000  0.04120770  0.09948134
##    e_wlkind      e_rail      e_road    e_airprt    e_impwtr   ep_minrty
##  0.11227869  0.57724027  0.20292897  0.11768580  0.11070844  0.77473053
##   ep_pov200   ep_nohsdp    ep_unemp   ep_renter   ep_houbdn  ep_uninsur
##  0.83559929  1.10849097  0.10344450  0.87337408  0.24156240  0.16052974
##    ep_noint    ep_age65    ep_age17   ep_disabl   ep_limeng   ep_mobile
##  0.51251325  0.09177855  0.08898825  0.21740933  1.22067357  0.05030199
##   ep_groupq   ep_bphigh   ep_asthma   ep_cancer    ep_mhlth ep_diabetes
##  0.12791546  0.22442021  0.56808811  0.10148957  0.23322023  1.87406328
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -0.253629 -0.070803  0.001584  0.003669  0.071375  0.316642
##       Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
## -3.863e-02 -3.135e-03  4.204e-05  5.830e-05  4.502e-03  2.329e-02
##                       Min        Max       Mean         StD
## e_ozone     0.000000e+00 0.16657977 0.01166416 0.026373238
## e_pm        1.627322e-03 0.47345524 0.03212233 0.059937163
## e_dslpm     2.396070e-03 0.79956503 0.04733299 0.098673361
## e_totcr     1.966467e-03 1.07312138 0.09164882 0.189239347
## e_npl       0.000000e+00 0.43877849 0.02837049 0.071846153
## e_tri       1.539254e-04 0.47476457 0.09407893 0.115816166
## e_tsd       0.000000e+00 0.05233505 0.00450023 0.007861019
## e_rmp       3.445814e-05 0.28235947 0.03991593 0.056364100
## e_coal      0.000000e+00 0.00000000 0.00000000 0.000000000
## e_lead      0.000000e+00 0.00000000 0.00000000 0.000000000
## e_park      0.000000e+00 0.15076461 0.01144035 0.017145026
## e_houage    3.005441e-03 0.17551236 0.02523248 0.023476700
## e_wlkind    4.775245e-03 0.18024159 0.02018668 0.022363100
## e_rail      8.706061e-04 0.45194974 0.03523789 0.064190025
## e_road      2.949998e-05 0.30995353 0.03423681 0.048140191
## e_airprt    0.000000e+00 0.26025327 0.02327458 0.052681604
## e_impwtr    2.805287e-03 0.13521078 0.02259338 0.022197326
## ep_minrty   5.905723e-03 0.29238514 0.06513552 0.049927779
## ep_pov200   9.197471e-03 0.45906601 0.13195498 0.111725501
## ep_nohsdp   1.322510e-02 0.47373983 0.13495005 0.105650122
## ep_unemp    3.943823e-03 0.16508108 0.02620598 0.028528941
## ep_renter   5.409651e-03 0.58838003 0.10014596 0.085300880
## ep_houbdn   7.083068e-03 0.35046171 0.06751131 0.055193361
## ep_uninsur  3.893328e-03 0.24840528 0.03383773 0.031639436
## ep_noint    5.965929e-03 0.45348962 0.12575910 0.113320837
## ep_age65    2.182992e-03 0.13800869 0.01937324 0.023311159
## ep_age17    3.108043e-03 0.15401943 0.03134260 0.031446758
## ep_disabl   6.137318e-03 0.36984917 0.06056298 0.061305564
## ep_limeng   4.142503e-03 0.36895952 0.06785540 0.059552523
## ep_mobile   5.503401e-06 0.04987863 0.00990896 0.010889849
## ep_groupq   2.408794e-03 0.06868492 0.01556478 0.009585813
## ep_bphigh   4.544709e-03 0.38436169 0.06286123 0.078786554
## ep_asthma   8.226793e-03 0.43963538 0.08033395 0.106647229
## ep_cancer   2.194027e-03 0.06714326 0.01277356 0.011041326
## ep_mhlth    4.473070e-03 0.31947635 0.08074732 0.091290628
## ep_diabetes 1.444889e-02 0.52508913 0.13682725 0.088253480
```

The final model hyperparameters have been set to $bandwidth = 49$, $mtry = 12$, and $ntrees = 500$.

GWRF Model 2

This model contains the hyperparameters defined in the RF building section.

```
## Ranger result
##
## Call:
##  ranger(rpl_eji ~ e_ozone + e_pm + e_dslpm + e_totcr + e_npl +      e_tri + e_tsd + e_rmp +
##
## Type:                             Regression
## Number of trees:                  800
## Sample size:                      240
## Number of independent variables:  36
## Mtry:                             8
## Target node size:                 5
## Variable importance mode:         impurity
## Splitrule:                        variance
## OOB prediction error (MSE):       0.01071509
## R squared (OOB):                  0.8319826
##     e_ozone        e_pm     e_dslpm     e_totcr       e_npl       e_tri
##  0.06381356  0.47740993  0.26766543  1.06015833  0.15450469  1.74188107
##       e_tsd       e_rmp      e_coal      e_lead      e_park    e_houage
##  0.02514420  0.31598480  0.00000000  0.00000000  0.05552358  0.11281819
##    e_wlkind      e_rail      e_road    e_airprt    e_impwtr   ep_minrty
##  0.12064289  0.57352822  0.21782886  0.08956095  0.12463339  0.69339834
##   ep_pov200   ep_nohsdp    ep_unemp   ep_renter   ep_houbdn  ep_uninsur
##  0.85506604  1.14218798  0.11662667  0.88467008  0.38736471  0.19069234
##    ep_noint    ep_age65    ep_age17   ep_disabl   ep_limeng   ep_mobile
##  0.54539434  0.09710374  0.09483521  0.23049902  1.18096697  0.04770333
##   ep_groupq   ep_bphigh   ep_asthma   ep_cancer    ep_mhlth ep_diabetes
##  0.12268079  0.24307777  0.66219145  0.12382942  0.26530645  1.68080758
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -0.261473 -0.075487  0.005867  0.001862  0.068819  0.315302
##       Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
## -0.0308604 -0.0033013  0.0005673  0.0001896  0.0043426  0.0288374
##                      Min        Max        Mean         StD
## e_ozone     0.0000000000 0.18117006 0.014457231 0.029591942
## e_pm        0.0020986042 0.43497196 0.036860523 0.059503875
## e_dslpm     0.0043000119 0.60718261 0.048764918 0.083467206
## e_totcr     0.0037681251 0.79656877 0.079992166 0.139964135
## e_npl       0.0000000000 0.33642448 0.025479229 0.056979475
## e_tri       0.0004295002 0.39486178 0.086727932 0.097596941
## e_tsd       0.0000000000 0.05997935 0.005102481 0.008682967
## e_rmp       0.0005180813 0.23197652 0.040274236 0.051471239
## e_coal      0.0000000000 0.00000000 0.000000000 0.000000000
## e_lead      0.0000000000 0.00000000 0.000000000 0.000000000
## e_park      0.0000000000 0.18186292 0.013271870 0.020353617
## e_houage    0.0044427244 0.16969165 0.029361122 0.025969413
## e_wlkind    0.0059190365 0.18448696 0.024183604 0.023122595
## e_rail      0.0010441108 0.38315748 0.037020544 0.058193155
## e_road      0.0003097620 0.26639724 0.035553062 0.044888112
## e_airprt    0.0000000000 0.21988199 0.021976896 0.047143529
## e_impwtr    0.0045507812 0.13915360 0.024447494 0.021721302
## ep_minrty   0.0077163066 0.30599237 0.074127225 0.052014263
## ep_pov200   0.0122693081 0.42102764 0.120422876 0.080974172
## ep_nohsdp   0.0192753275 0.35645576 0.123197239 0.077726424
## ep_unemp    0.0054092285 0.13739037 0.031304983 0.027806488
## ep_renter   0.0083628798 0.38783979 0.097605599 0.065222796
## ep_houbdn   0.0129528340 0.24972488 0.073733332 0.047111214
## ep_uninsur  0.0049589407 0.27540449 0.040950802 0.033101485
## ep_noint    0.0131526456 0.35762103 0.114012543 0.084757083
## ep_age65    0.0031540796 0.13477704 0.022595917 0.025079364
## ep_age17    0.0040455006 0.13526952 0.034623467 0.027153301
## ep_disabl   0.0102541315 0.26847837 0.061347662 0.051174821
## ep_limeng   0.0034557730 0.34994150 0.072440951 0.053672203
## ep_mobile   0.0001481956 0.05435700 0.010857069 0.011991083
## ep_groupq   0.0031971931 0.05634757 0.017101618 0.009872093
## ep_bphigh   0.0071042942 0.32160118 0.064237818 0.068223728
## ep_asthma   0.0091518792 0.34923752 0.077563560 0.080063952
## ep_cancer   0.0035182091 0.08225959 0.017028374 0.013449289
## ep_mhlth    0.0058690834 0.27665581 0.077878924 0.074813617
## ep_diabetes 0.0176545623 0.44690265 0.127320112 0.071652368
```

The final model hyperparameters have been set to $bandwidth = 49$, $mtry = 6$, and $ntrees = 501$.

GWRF Model Evaluation

The models both preform nearly identically because the hyperparameters preform nearly identically. Therefore, the model define by the previous traditional random forest model.

Table 3: Performance Metrics for Each Model

Model	bw	mtry	ntree	MAE	RMSE	R_Squared
Model 5	49	5	500	0.082	0.103	0.833
Model 6	49	9	501	0.083	0.104	0.831

Random Forest Model Comparisons

Model 4 shows a lower MAE and MSE, indicating that it generally makes smaller errors in prediction. The RMSE is also relatively low, and the high R^2 value (0.791) suggests that this model explains a significant portion of the variance in the SVI. Overall, Model 4 performs well and is effective in predicting SVI using the selected predictors.

Model 6 has higher MAE and MSE values, indicating that it makes larger errors on average compared to Model 4. The RMSE is also higher, and the R^2 is lower (0.638), suggesting that Model 6 explains less variance in the SVI. This could mean that while the Geographically Weighted Random Forest accounts for spatial autocorrelation, it may not perform as well in terms of overall prediction accuracy as the traditional Random Forest.

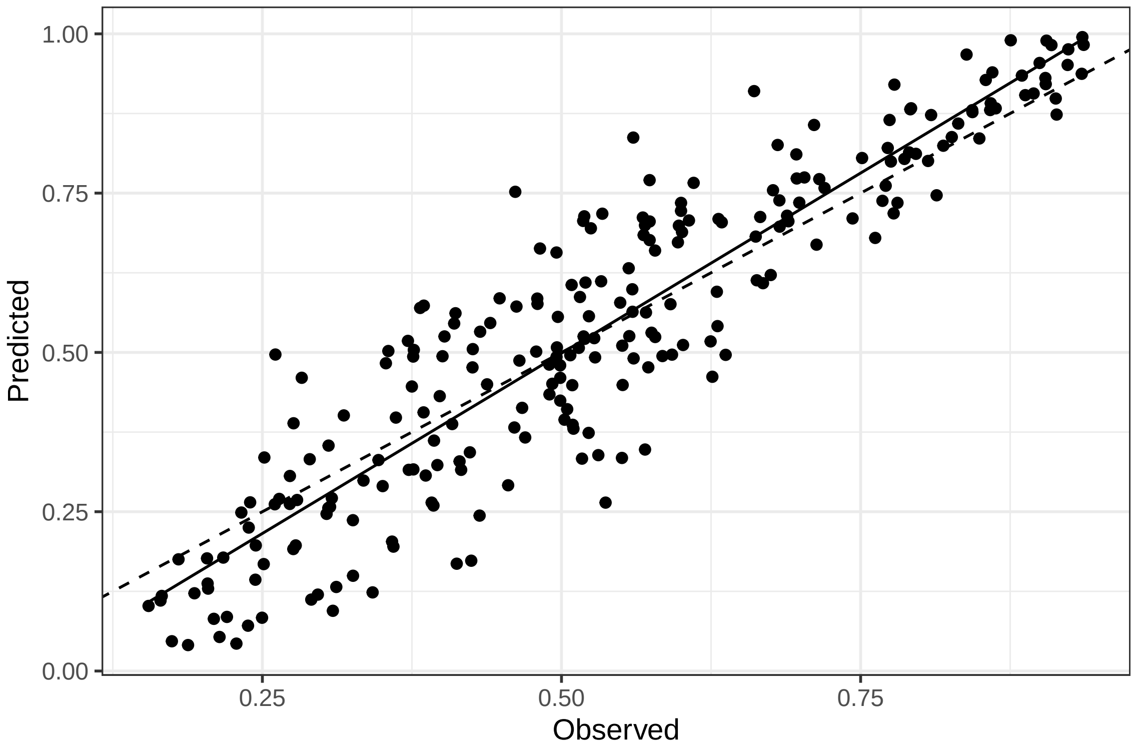
Model 4 (Traditional Random Forest) outperforms Model 6 (Geographically Weighted Random Forest) in terms of accuracy and explained variance. However, Model 6 is still valuable because it accounts for spatial dependencies. Model 6 might be more appropriate despite its lower overall performance metrics, particularly if the goal is to understand regional variations in the SVI. However, if the focus is purely on predictive accuracy, Model 4 appears to be the better choice.

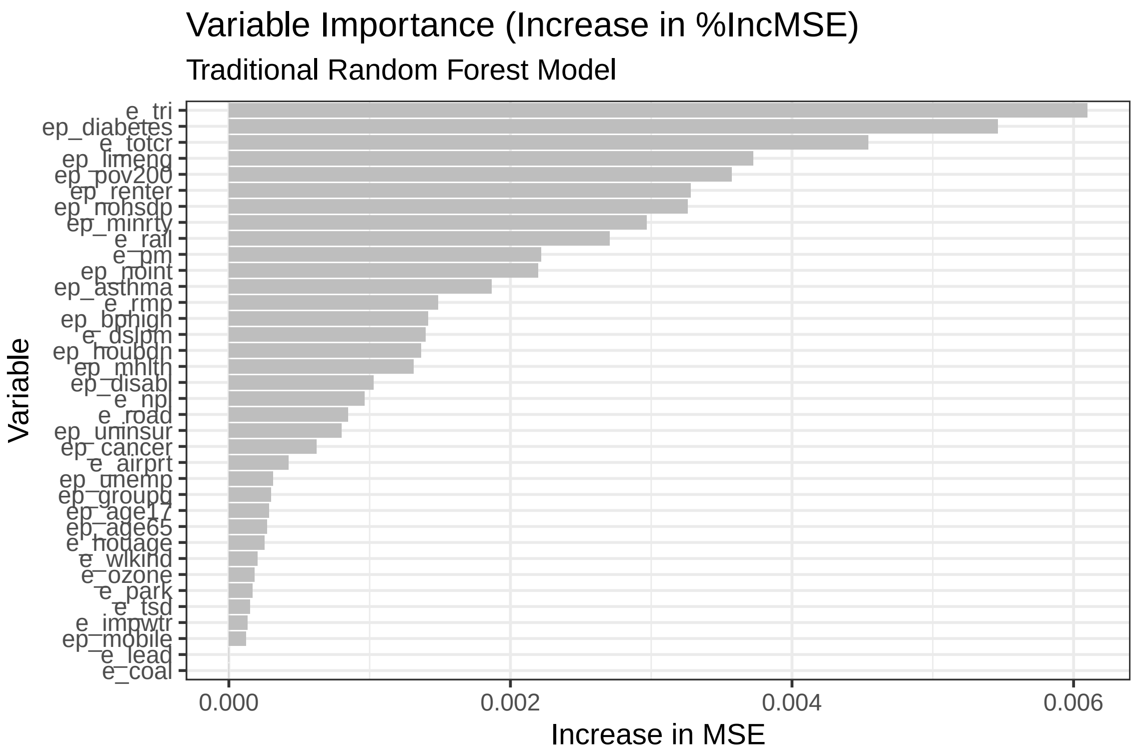
Table 4: Performance Metrics for Each Model

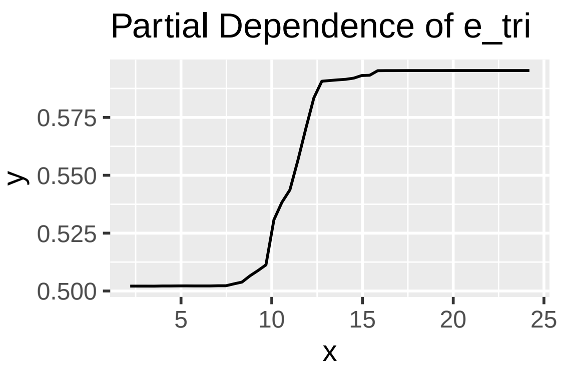
	Model	bw	mtry	ntree	MAE	RMSE	R_Squared
4	Model 4	-	6	501	0.085	0.105	0.842
2	Model 6	49	9	501	0.083	0.104	0.831

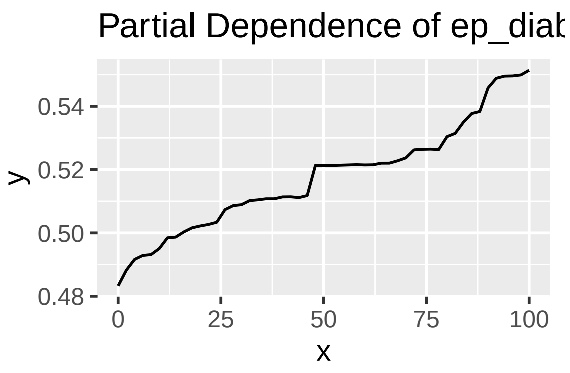
Graphs for the Final Models

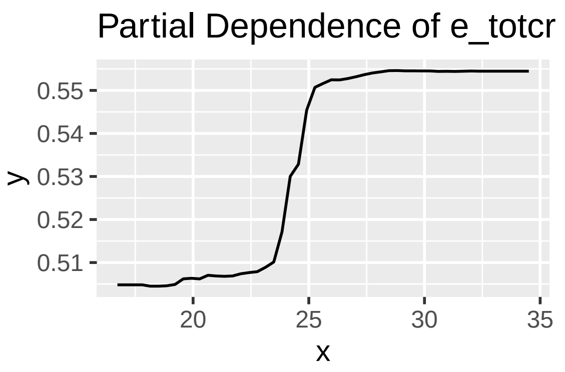
Traditional Random Forest

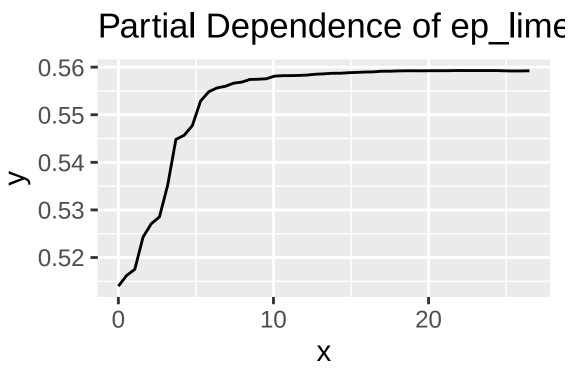




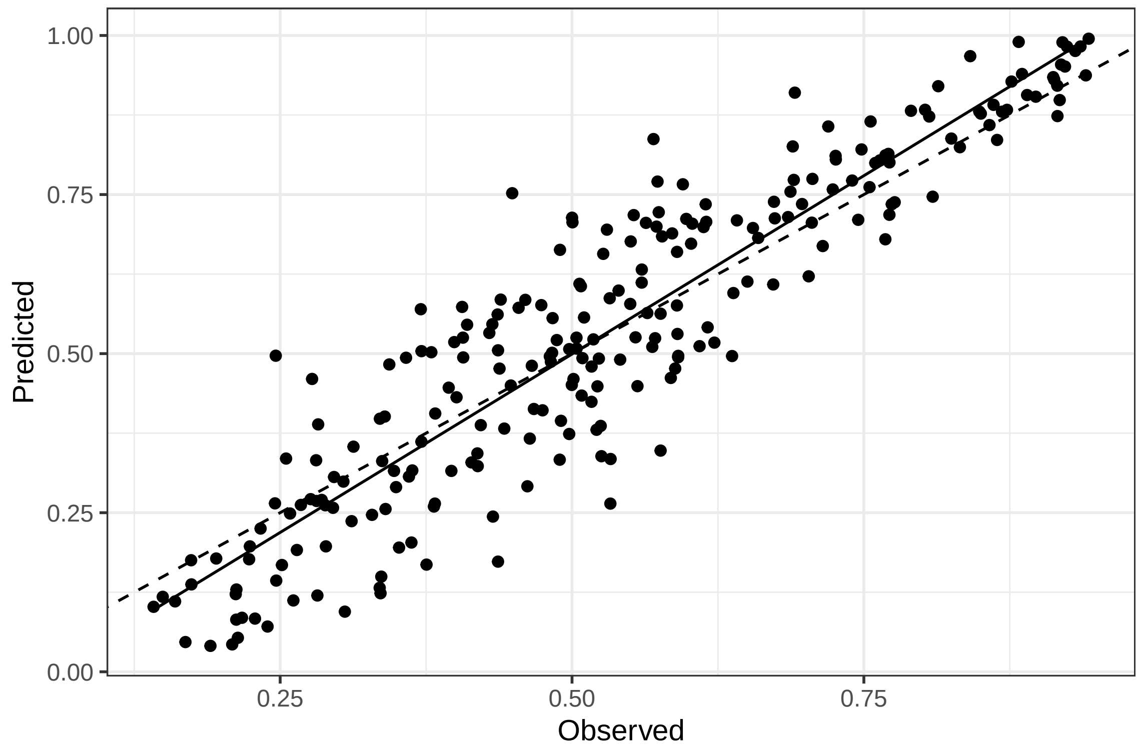

Partial Dependence of e_tri
y
0.500
0.525
0.550
0.575
x
5
10
15
20
25

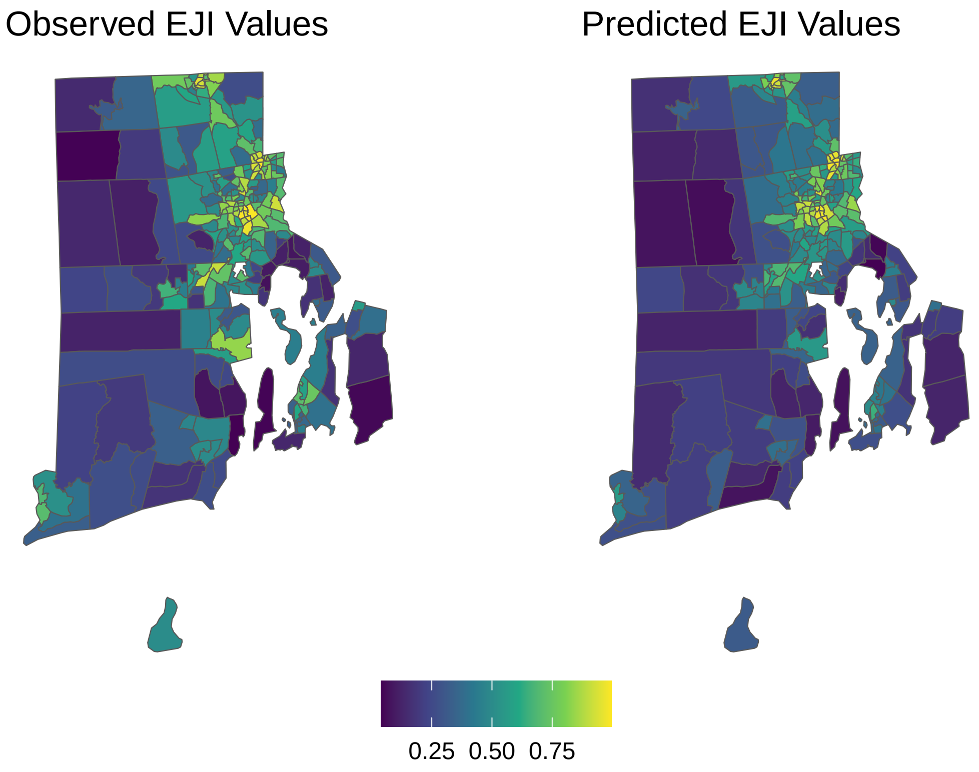

Partial Dependence of ep_diab
y
0.48
0.50
0.52
0.54
x
0
25
50
75
100

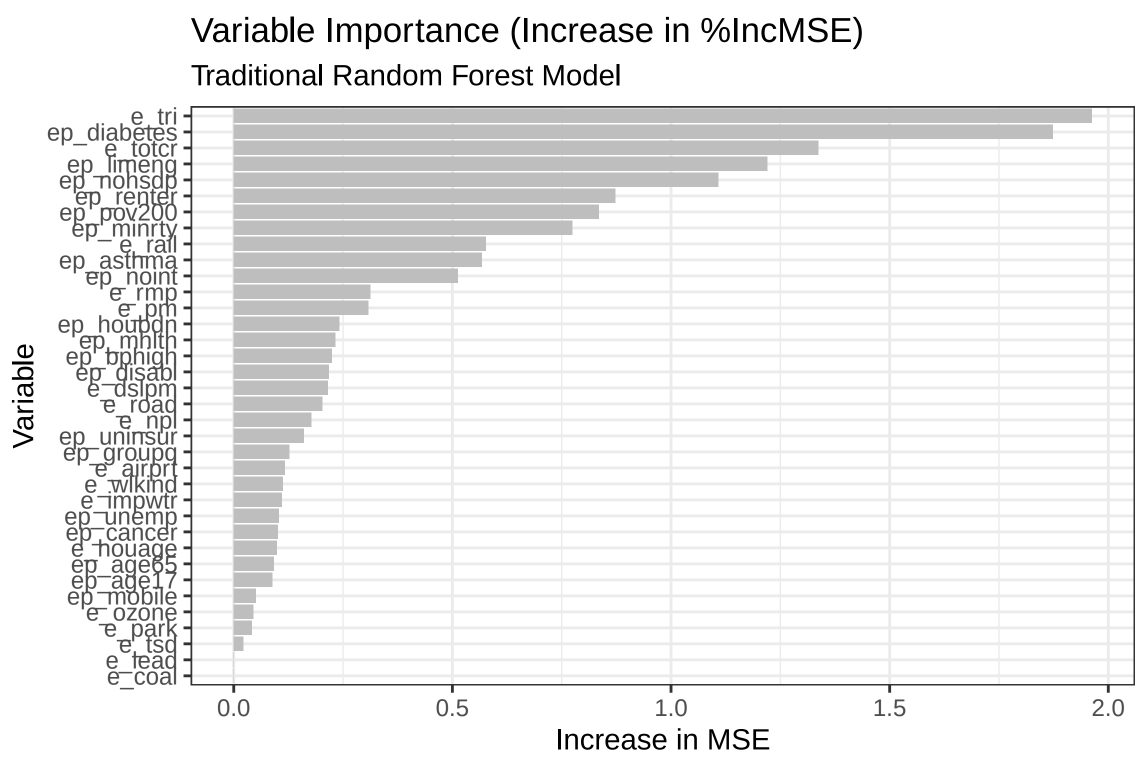

Partial Dependence of e_totcr
y
0.51
0.52
0.53
0.54
0.55
x
20
25
30
35

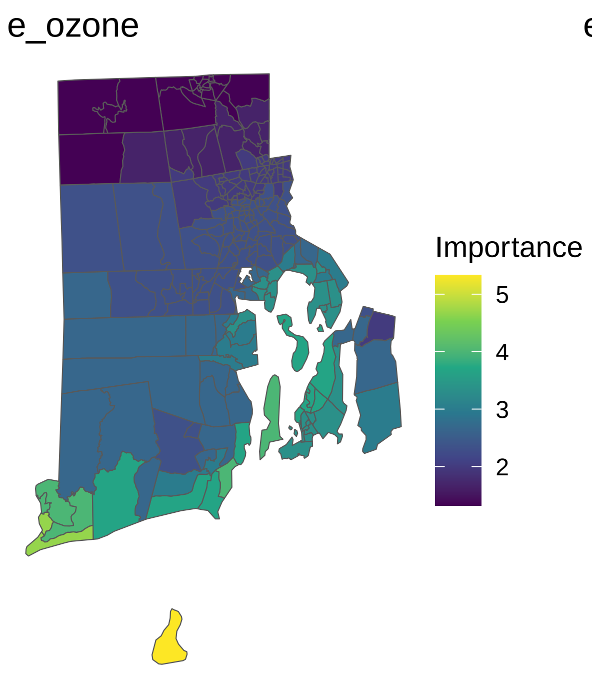

Partial Dependence of ep_lime
y
0.52
0.53
0.54
0.55
0.56
x
0
10
20

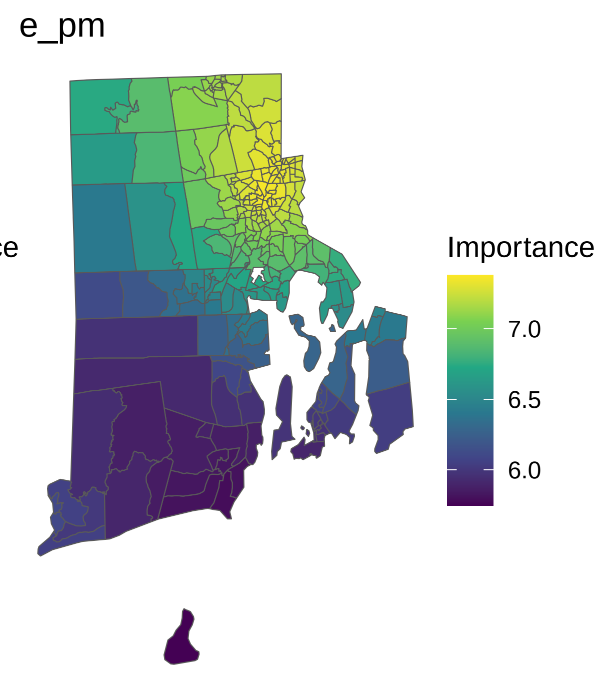
Geographically Weighted Random Forest

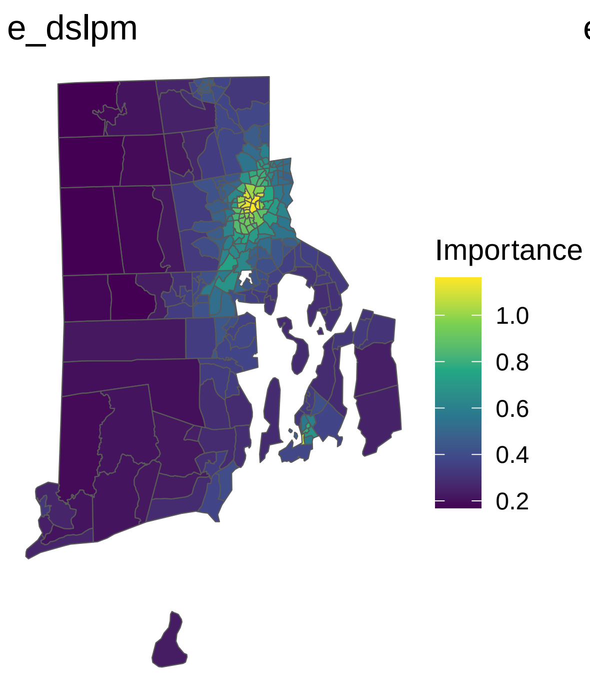


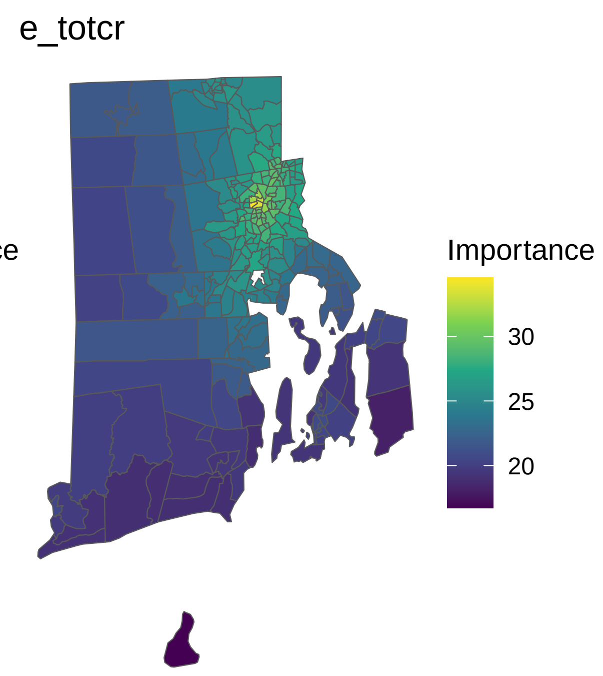


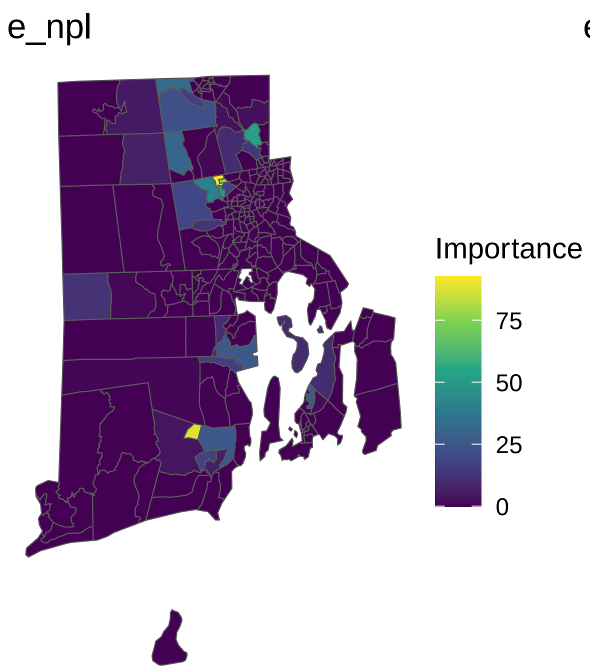

Variable Importance (Increase in %IncMSE)
Traditional Random Forest Model
Variable
e_tri
ep_diabetes
e_totcr
ep_limeng
ep_nohsdp
ep_renter
ep_pov200
ep_minrty
e_rail
ep_asthma
ep_noint
e_rmp
e_pm
ep_houbdn
ep_mhlth
ep_bphigh
ep_disabl
e_dslpm
e_road
e_npl
ep_uninsur
ep_groupq
e_airprt
e_wlkind
e_impwtr
ep_unemp
ep_cancer
e_houage
ep_age65
ep_age17
ep_mobile
e_ozone
e_park
e_tsd
e_lead
e_coal
0.0
0.5
1.0
1.5
2.0
Increase in MSE

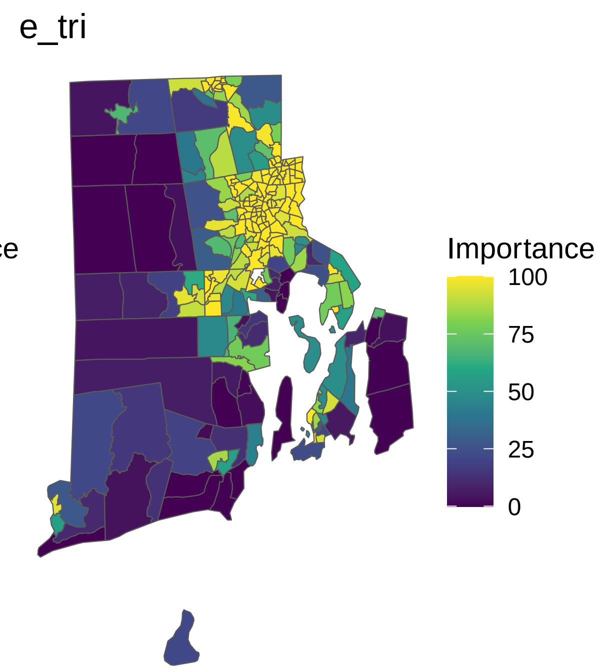

e_ozone
Importance
5
4
3
2

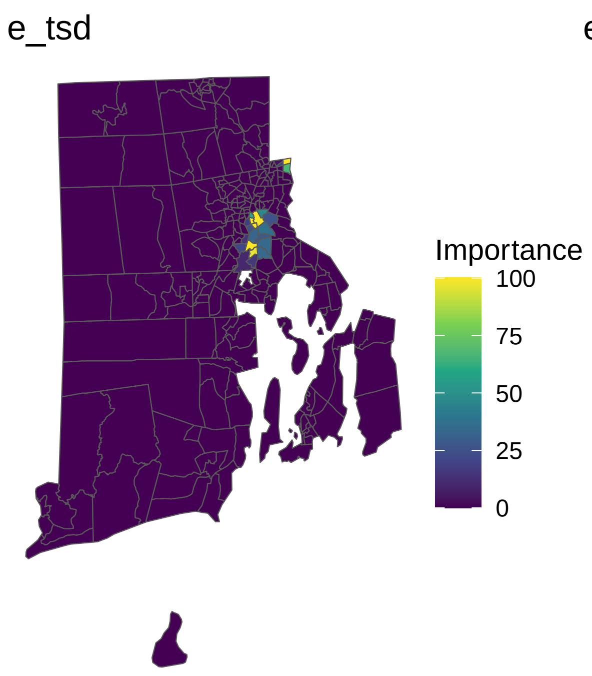

e_pm
Importance
7.0
6.5
6.0

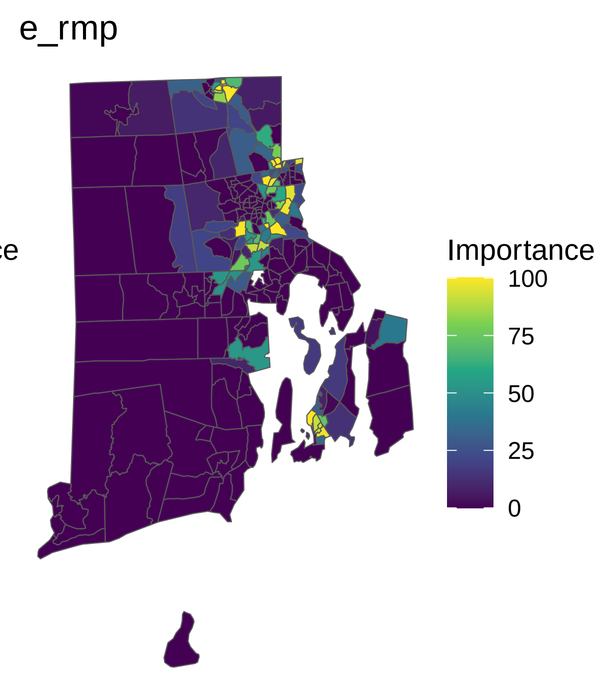

e_dslpm
Importance
1.0
0.8
0.6
0.4
0.2

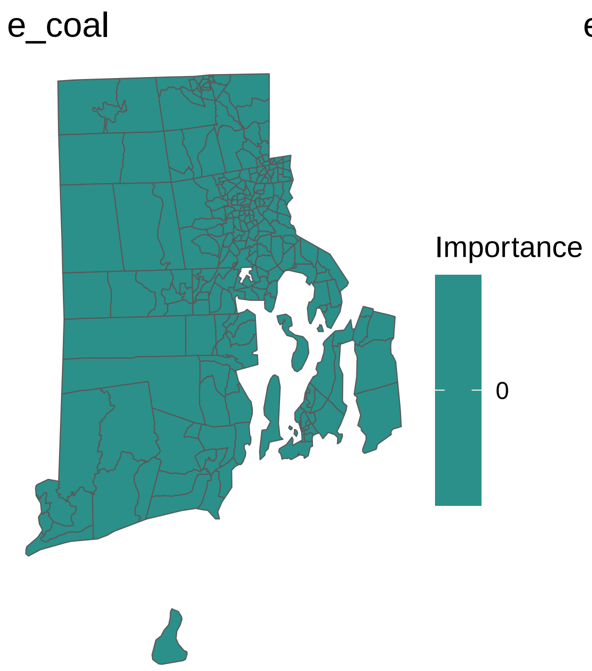

e_totcr
Importance
30
25
20

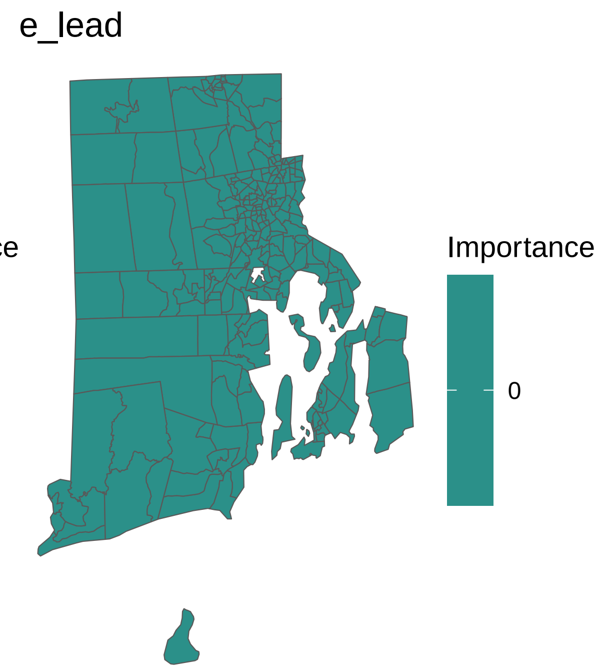

e_npl
Importance
75
50
25
0

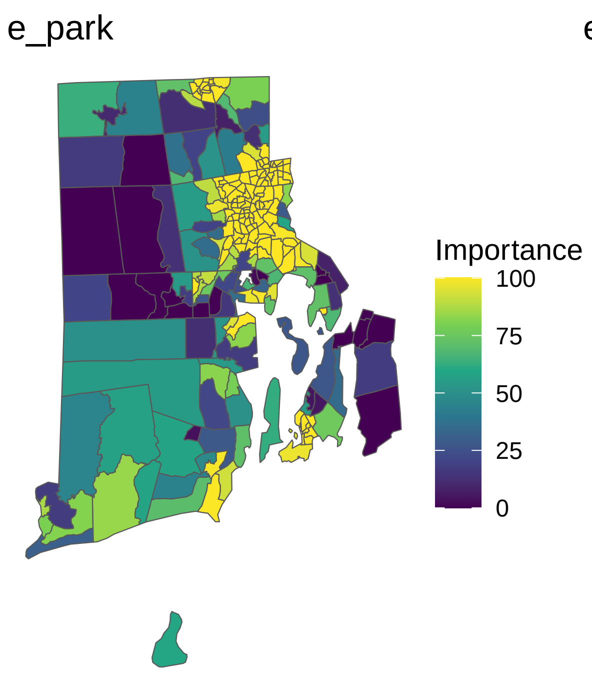

e_tri
Importance
100
75
50
25
0

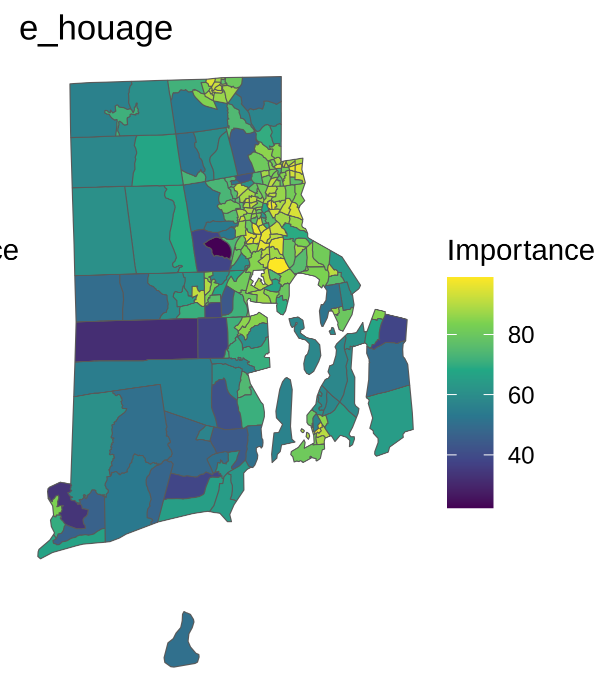

e_tsd
Importance
100
75
50
25
0

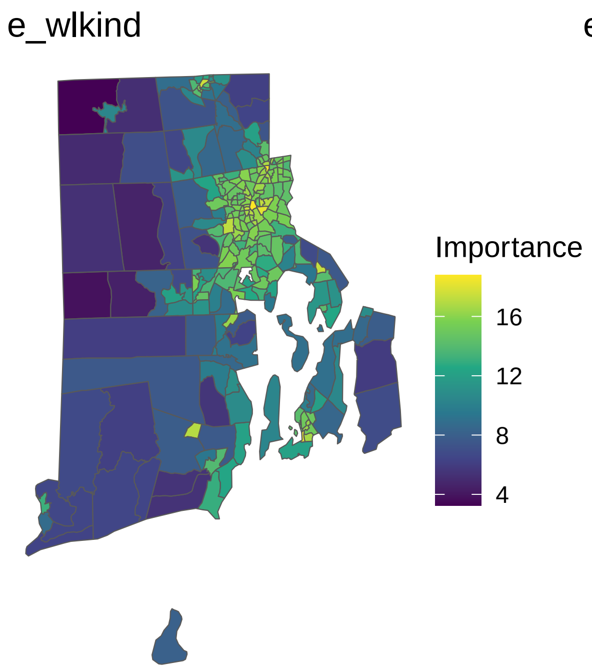

e_rmp
Importance
100
75
50
25
0

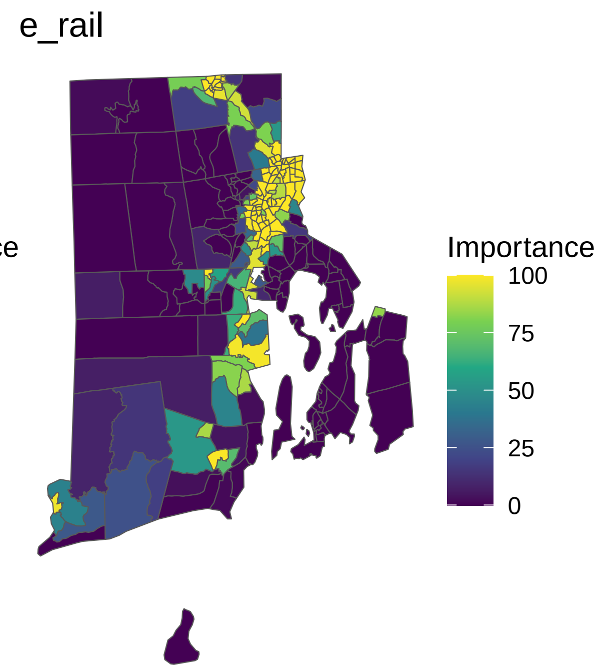

e_coal
Importance
0

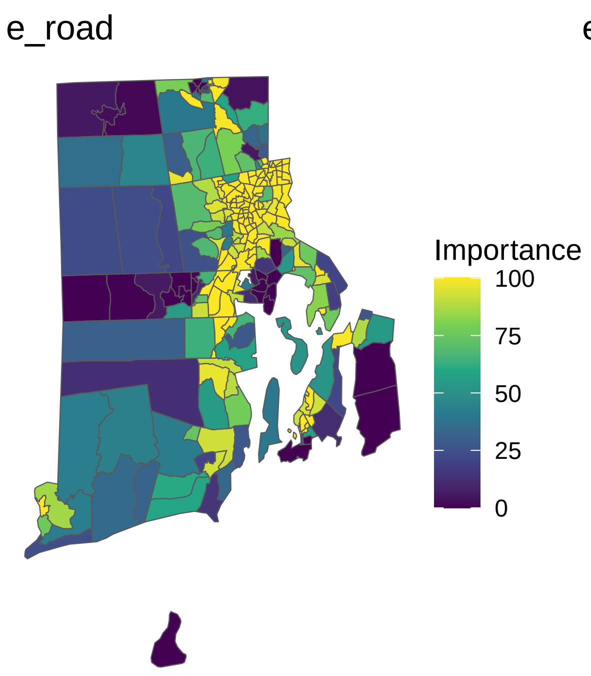

e_lead
Importance
0

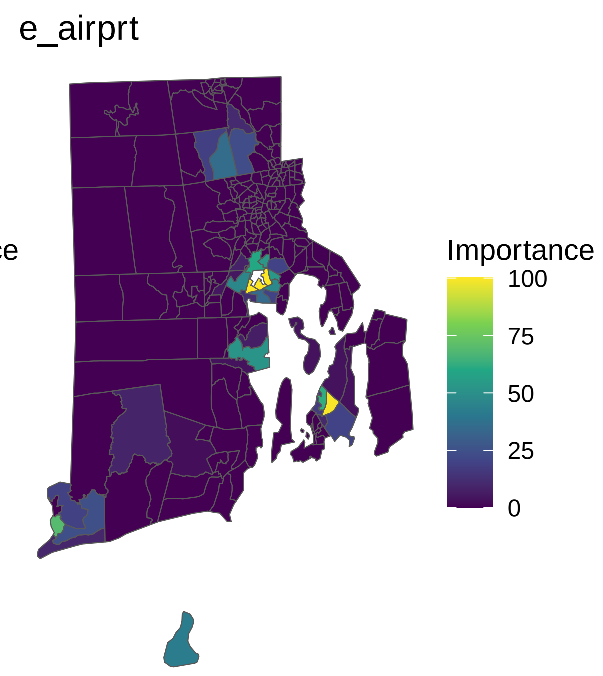

e_park
Importance
100
75
50
25
0

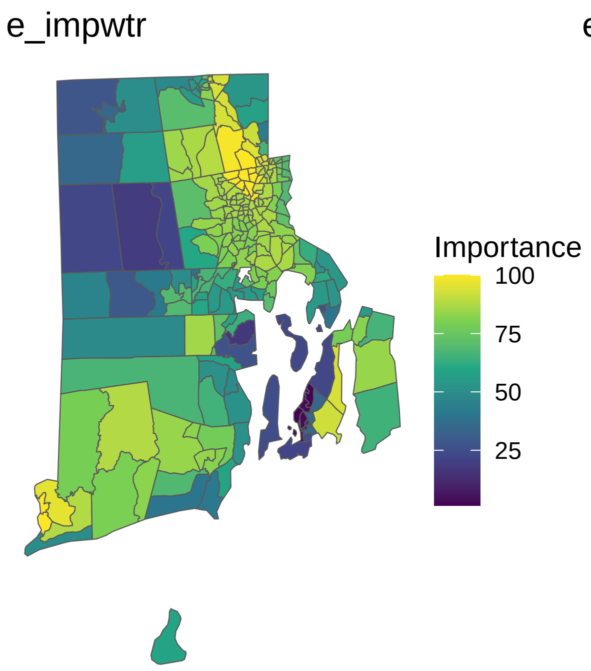

e_houage
Importance
80
60
40

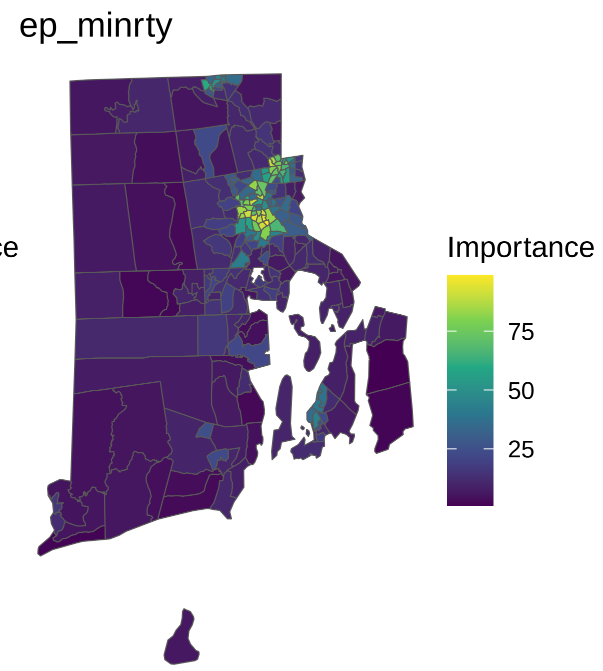

e_wlkind
Importance
16
12
8
4

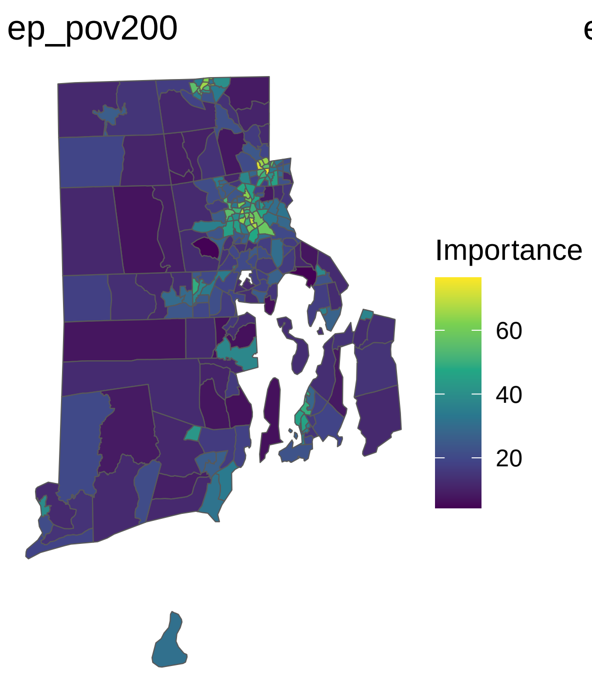

e_rail
Importance
100
75
50
25
0

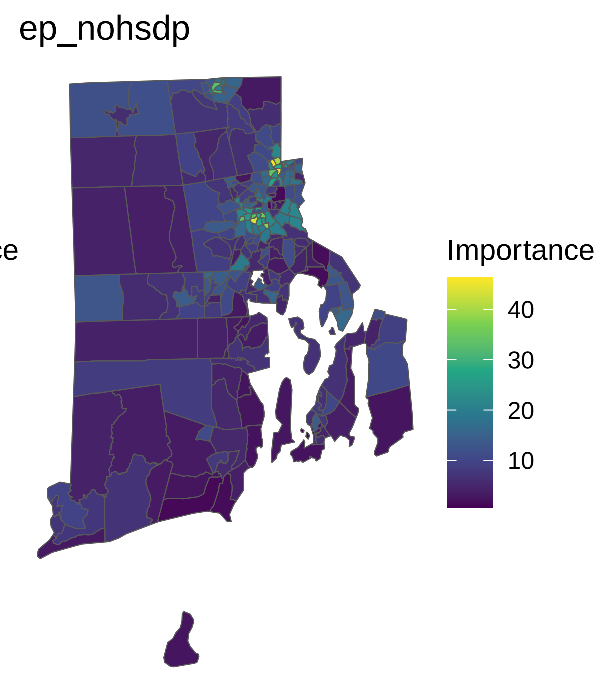

e_road
Importance
100
75
50
25
0

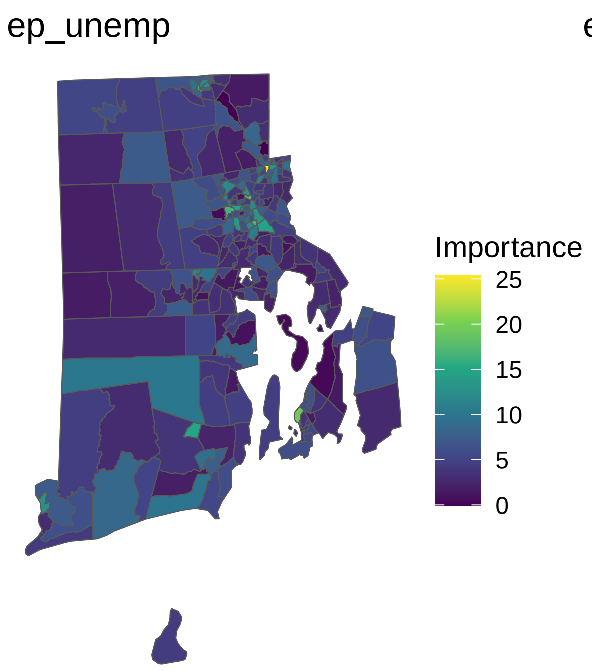

e_airprt
Importance
100
75
50
25
0

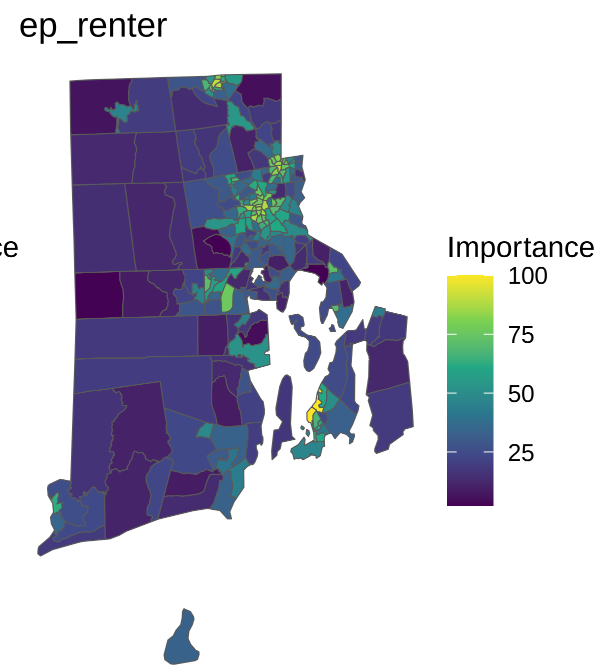

e_impwtr
Importance
100
75
50
25

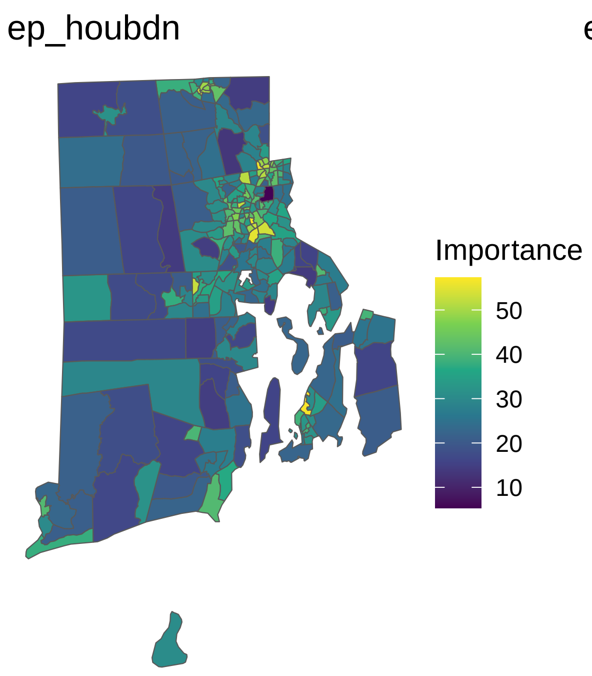

ep_minrty
Importance
75
50
25

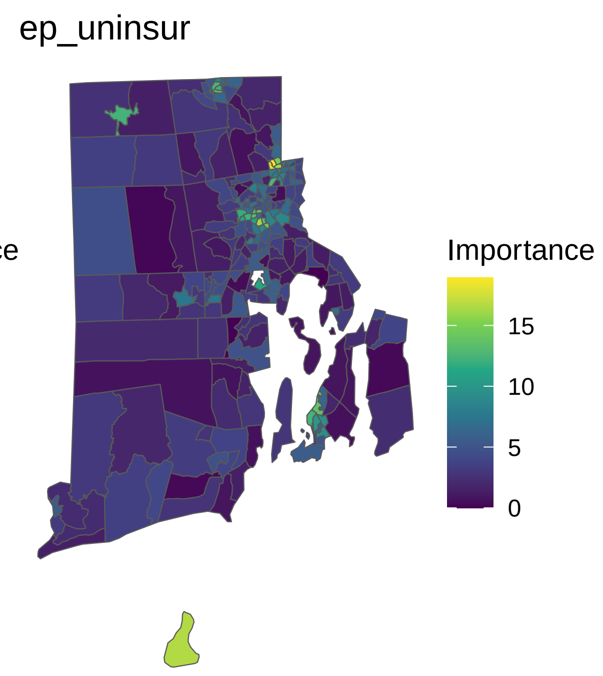

ep_pov200
Importance
60
40
20

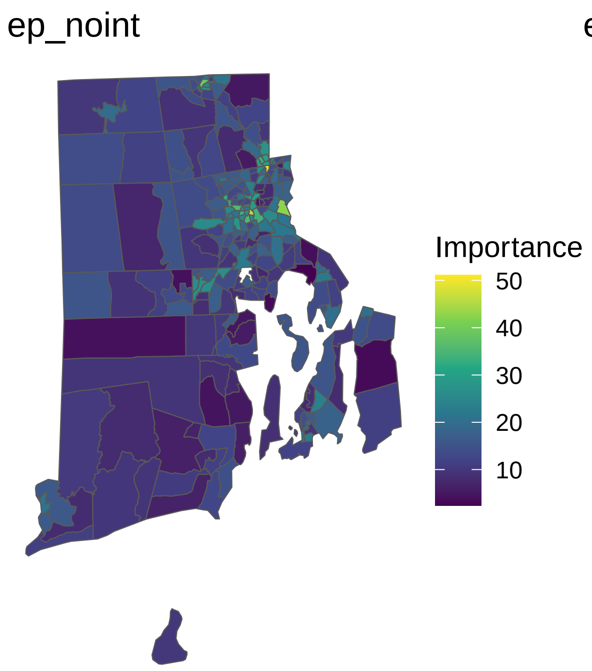

ep_nohsdp
Importance
40
30
20
10

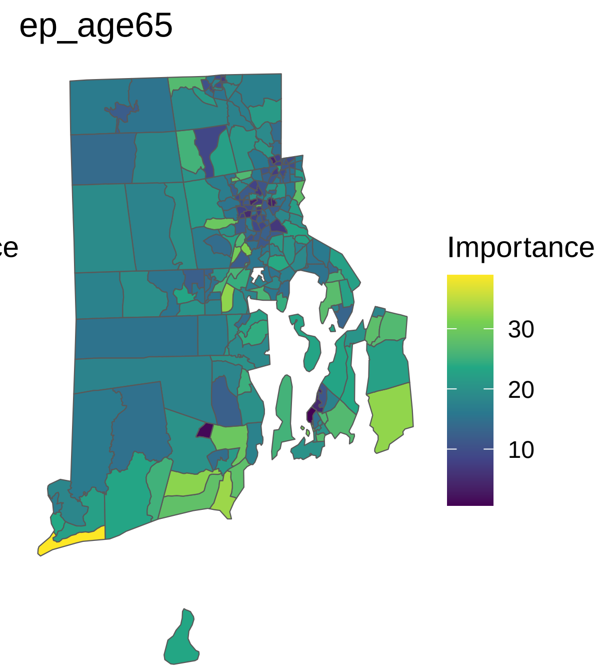

ep_unemp
Importance
25
20
15
10
5
0

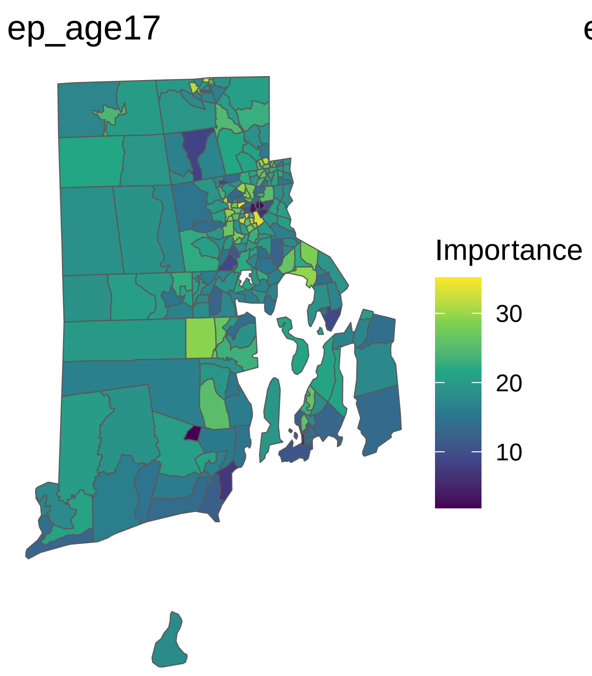

ep_renter
Importance
100
75
50
25

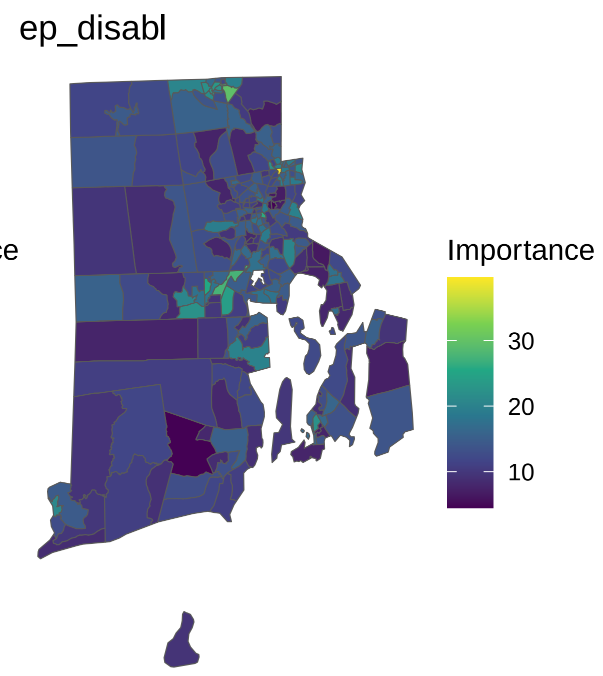

ep_houbdn
Importance
50
40
30
20
10

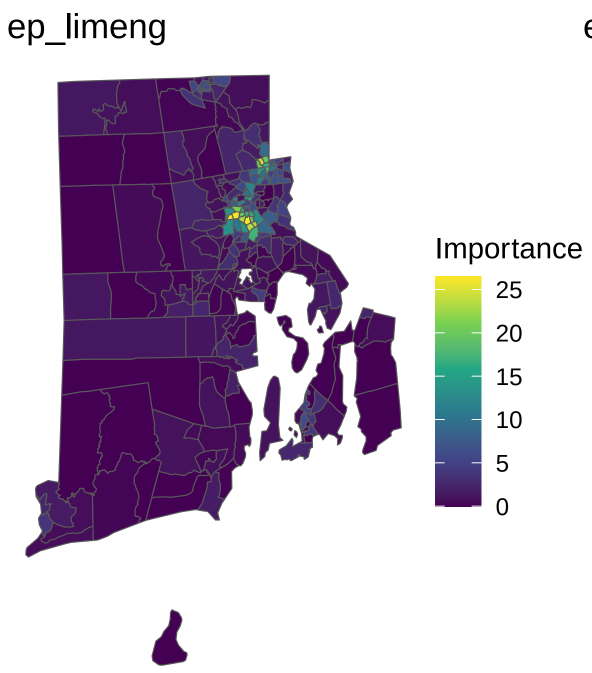

ep_uninsur
Importance
15
10
5
0

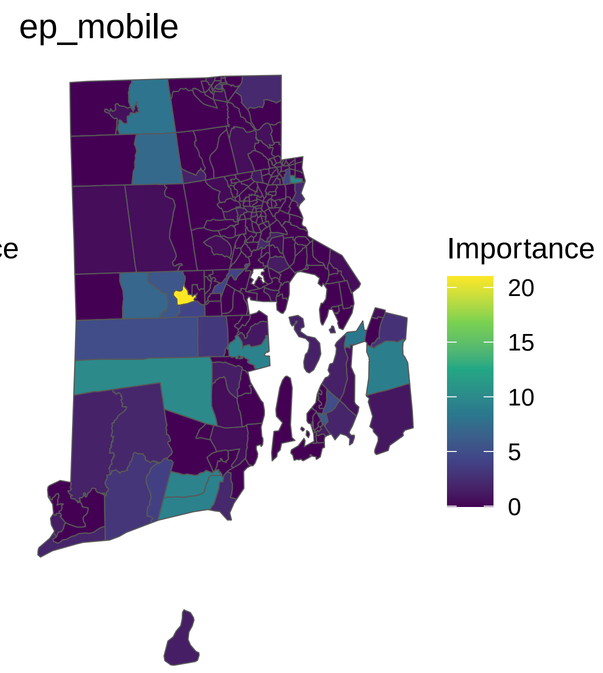

ep_noint
Importance
50
40
30
20
10

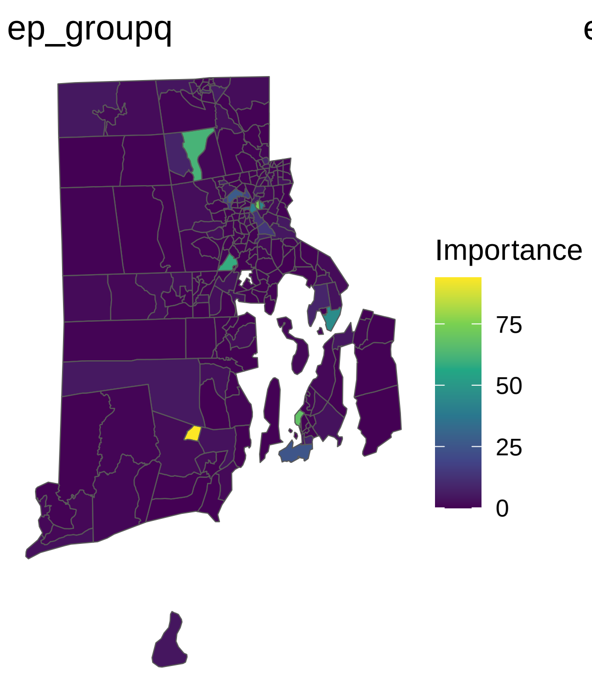

ep_age65
Importance
30
20
10

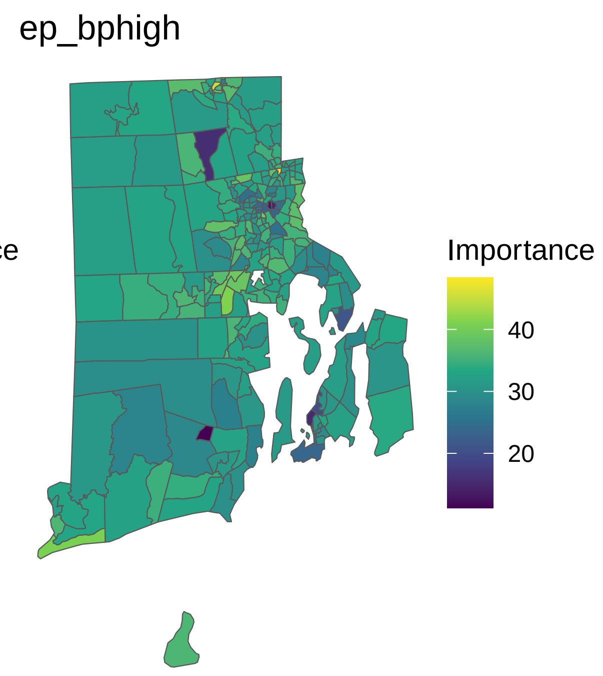

ep_age17
Importance
30
20
10

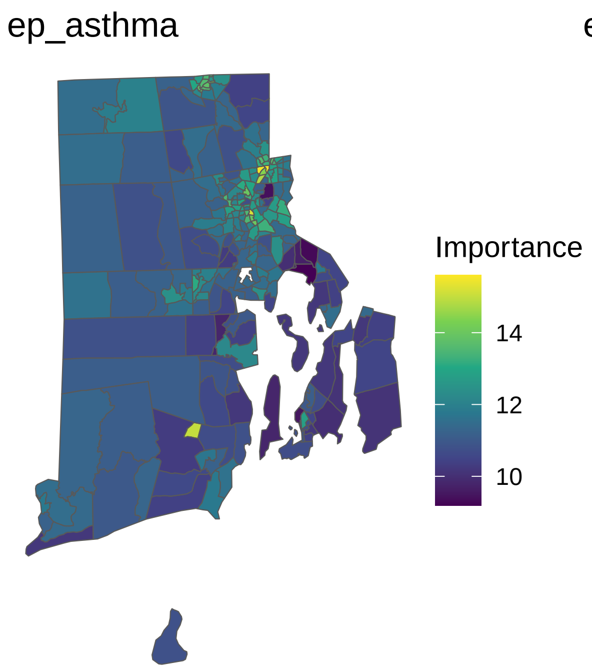

ep_disabl
Importance
30
20
10

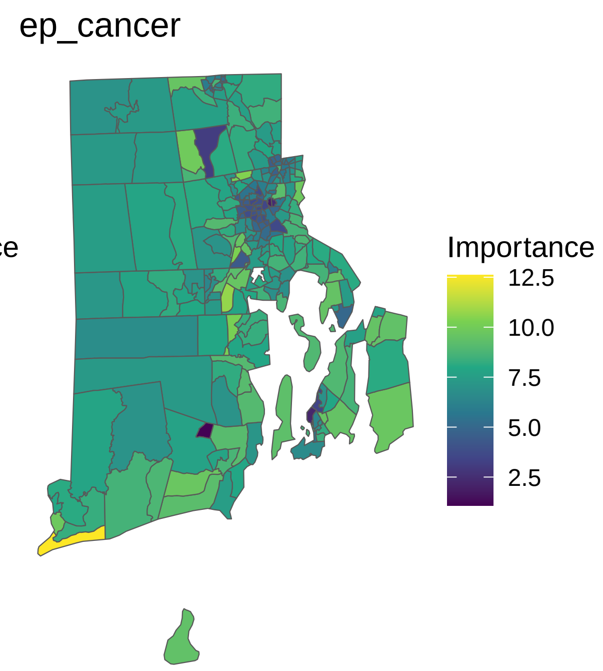

ep_limeng
Importance
25
20
15
10
5
0

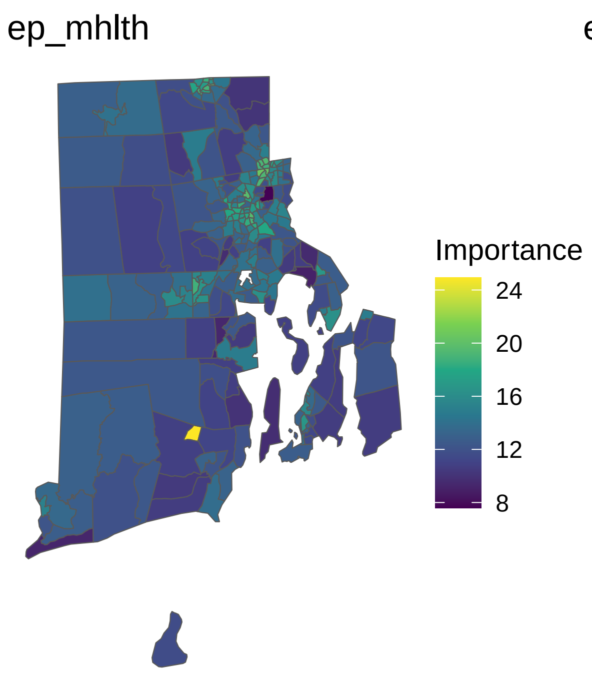

ep_mobile
Importance
20
15
10
5
0

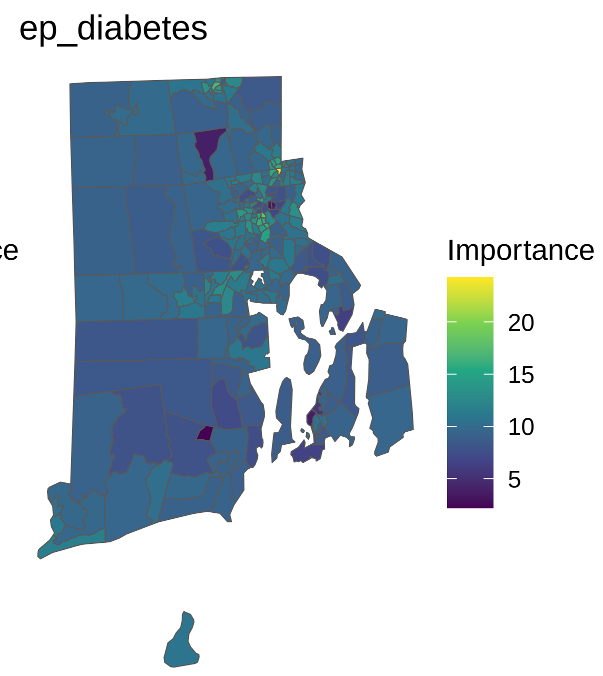

ep_groupq
Importance
75
50
25
0


ep_bphigh
Importance
40
30
20


ep_asthma
Importance
14
12
10


ep_cancer
Importance
12.5
10.0
7.5
5.0
2.5


ep_mhlth
Importance
24
20
16
12
8


ep_diabetes
Importance
20
15
10
5