Random Forest Model Training

Random forest hyperparameters tuned on full and observed datasets for each case separately

Amos Okutse

22 December, 2022

Contents

0.1 Introduction

Generate the data and see whether the two-step process results in less biased estimates? [We are adjusting for covariates that did not generate the outcome Y, at least not directly since the outcome was generated by $Z_i's$ which were not observed]

0.2 Data generating process

The data generating process is derived from Kang and Schaffer (2007) pg. 526. Assume we have a random sample, $i = 1, \cdots, n$ from an infinite population. Outcome variable, y_i, R_i is the response indicator corresponding to 1 if y_i is observed and 0 otherwise. x_i is the p-dimensional vector of covariates that may be related to y_i and R_i. The population response and non-response rates are denoted by $r^{(1)} = P(R_i = 1)$ and $P(R_i = 0)$, respectively, whereas the sample response rates are denoted by $\hat{r}^{(1)} = n^{(1)}/n$ and $\hat{r}^{(0)} = n^{(0)}/n$. For each i suppose that $(z_{i1}, z_{i2}, z_{i3}, z_{i4})^T$ are independently distributed as $N(0, I)$ where I is a 4×4 identity matrix. $y_i's$ are generated as:

$$y_i = 210 + 50A_i + 27.4Z_{i1} + 13.7z_{i2} + 13.7z_{i3} + 13.7z_{i4} + \epsilon_i \tag{1}$$

where $\epsilon \sim N(0, 1)$ and the true propensity scores are defined as:

$$\pi_i = \text{expit}(-z_{i1} + 0.5z_{i2} - 0.25z_{i3} - 0.1z_{i4}) \tag{2}$$

The average response rate is assumed as $r^{(1)} = 0.5$ and the difference in means is $E(y/R = 1) - E(y/R = 0) = 20$. The correct π model in this scenario is a logistic regression model of R_i on the z_{ij}'s whereas the correct y model is a linear regression of y_i on the z_{ij}'s. Suppose, however, that the covariates actually observed by the data analyst are:

$$\begin{aligned}
x_{i1} &= exp(z_{i1}/2), \\
x_{i2} &= z_{i2}/(1 + exp(Z_{i1})) + 10, \\
x_{i3} &= (z_{i1}z_{i3}/25 + 0.6)^3, \\
x_{i4} &= (z_2 + z_4 + 20)^2
\end{aligned}$$

such that $\text{logit}(\pi_i)$ and $E(y_i/x)$ are linear functions of $log(x_{i1}), x_2, x_1^2x_2, 1/log(x_1), x_3/log(x_1), x_4^{1/2}$. We proceed by sampling $n = 1000$ observations as follows:

- Set $A_i = 1$ for $i = 1, \cdots, 500$ and $A_i = 0$ for $i = 501, \cdots, 1000$.
- Draw $z_{i1}, \cdots, z_{i4} \sim N(0, I)$ where I is a 4×4 identity matrix.
- Calculate y_i as in (1).
- Calculate $\text{expit}(-z_{i1} + 0.5z_{i2} - 0.25z_{i3} - 0.1z_{i4})$
- Draw $R_i \sim Ber(\pi_i)$
- Denote $R_i = 0$ as those with missing outcome data y_i
- Calculate the $x_i's$ from the $z_i's$

```
setwd("C:\\Users\\aokutse\\OneDrive - Brown
 ↪  University\\ThesisResults\\[4]_random_forest\\train_rf")
## set up the simulation
rm(list = ls())
## load the saved single data files
load("C:\\Users\\aokutse\\OneDrive - Brown
 ↪  University\\ThesisResults\\data\\df_one.RData")
load("C:\\Users\\aokutse\\OneDrive - Brown
 ↪  University\\ThesisResults\\data\\df_two.RData")
```

```
load("C:\\Users\\aokutse\\OneDrive - Brown
↪  University\\ThesisResults\\data\\df_three.RData")
load("C:\\Users\\aokutse\\OneDrive - Brown
↪  University\\ThesisResults\\data\\df_four.RData")



## load the saved list data files
##load("C:\\Users\\aokutse\\OneDrive - Brown
↪  University\\ThesisResults\\data\\dsets1.RData")
##load("C:\\Users\\aokutse\\OneDrive - Brown
↪  University\\ThesisResults\\data\\dsets2.RData")
##load("C:\\Users\\aokutse\\OneDrive - Brown
↪  University\\ThesisResults\\data\\dsets3.RData")
##load("C:\\Users\\aokutse\\OneDrive - Brown
↪  University\\ThesisResults\\data\\dsets4.RData")
```

0.2.1 RANDOM FOREST MODELS

0.3 TUNING ON THE FULL DATASET

0.3.0.1 Full data modelling [case 1 when n = 500 and sd = 1]

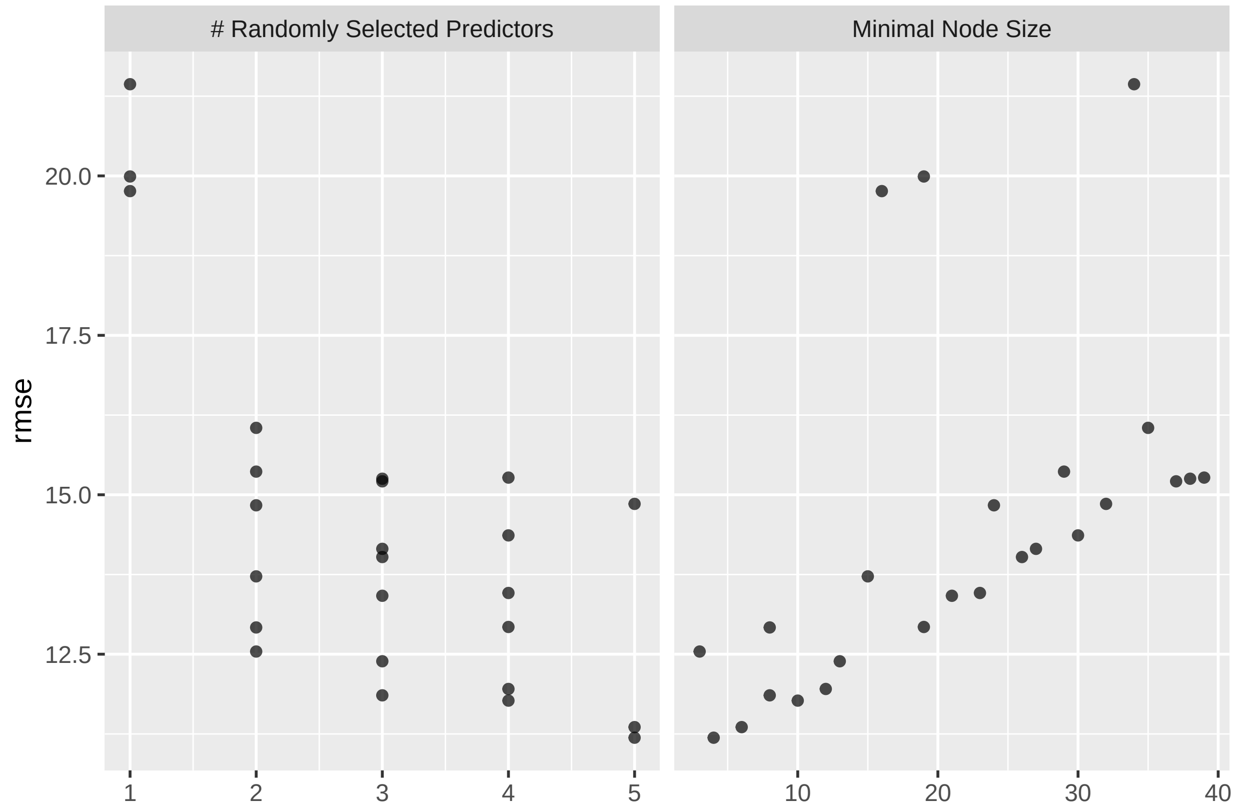
- All model parameter tuning are based on the full data.
- Parameters are explored for all possible scenarios on the full data set with 10-fold cross-validation.
- We use a space-filling grid design and search parameters across a grid of 25 models using cross-validation.

```
## using data with n = 500 and sd = 1 (df_one)

train <- df_one[, -7]


## create the random forest model object
rf_model <- rand_forest(trees = 1000, mtry = tune(), min_n = tune()) %>%
  set_engine("ranger", importance = "impurity") %>%
  set_mode("regression")

## create the workflow
rf_workflow <- workflow() %>%
  add_model(rf_model) %>%
  add_formula(y ~ A + x1 + x2 + x3 + x4)

## create the procedure for validating the model
val_set <- validation_split(train, prop = 0.80)

## set up the set of metrics to gather from the models [there is no mse; can't use
↪  accuracy too which is for class]
metrics <- metric_set(rmse, rsq) ## rsq=coefficient of determination =
```

```r
## create the re-sampling folds for hyper-parameter tuning
set.seed(345)
folds <- vfold_cv(train, v = 10)


## fit re-samples and estimate the hyper parameters
doParallel::registerDoParallel()  ## leverage parallel processing

set.seed(456)
rf_results <- rf_workflow %>%
  tune_grid(val_set,
            grid = 25,
            resamples = folds,
            control = control_grid(save_pred = TRUE), #saving preds allows collecting the
             ↪  metrics
            metrics = metric_set(rmse))

rf_results %>% show_best(metric = "rmse")
```

```
## # A tibble: 5 x 8
##    mtry min_n .metric .estimator  mean     n std_err .config
##   <int> <int> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1     5     4 rmse    standard    11.2    10   0.528 Preprocessor1_Model01
## 2     5     6 rmse    standard    11.4    10   0.525 Preprocessor1_Model20
## 3     4    10 rmse    standard    11.8    10   0.504 Preprocessor1_Model04
## 4     3     8 rmse    standard    11.9    10   0.472 Preprocessor1_Model25
## 5     4    12 rmse    standard    12.0    10   0.513 Preprocessor1_Model22
```

```r
## can plot the best models
autoplot(rf_results)
```




```
## rf best model based on accuracy
rf_best = rf_results %>%
  select_best(metric = "rmse")
rf_best
```

```
## # A tibble: 1 x 3
##    mtry min_n .config
##   <int> <int> <chr>
## 1     5     4 Preprocessor1_Model01
```

```
## variable importance??
final_md <- finalize_model(rf_model, rf_best)
options(scipen = 999)
full_one <- final_md %>%
 set_engine("ranger", importance = "impurity") %>%
  fit(y ~ .,
    data = df_one[, -7]) %>%
  vip::vip(geom = "col") + theme_minimal()
full_one
```

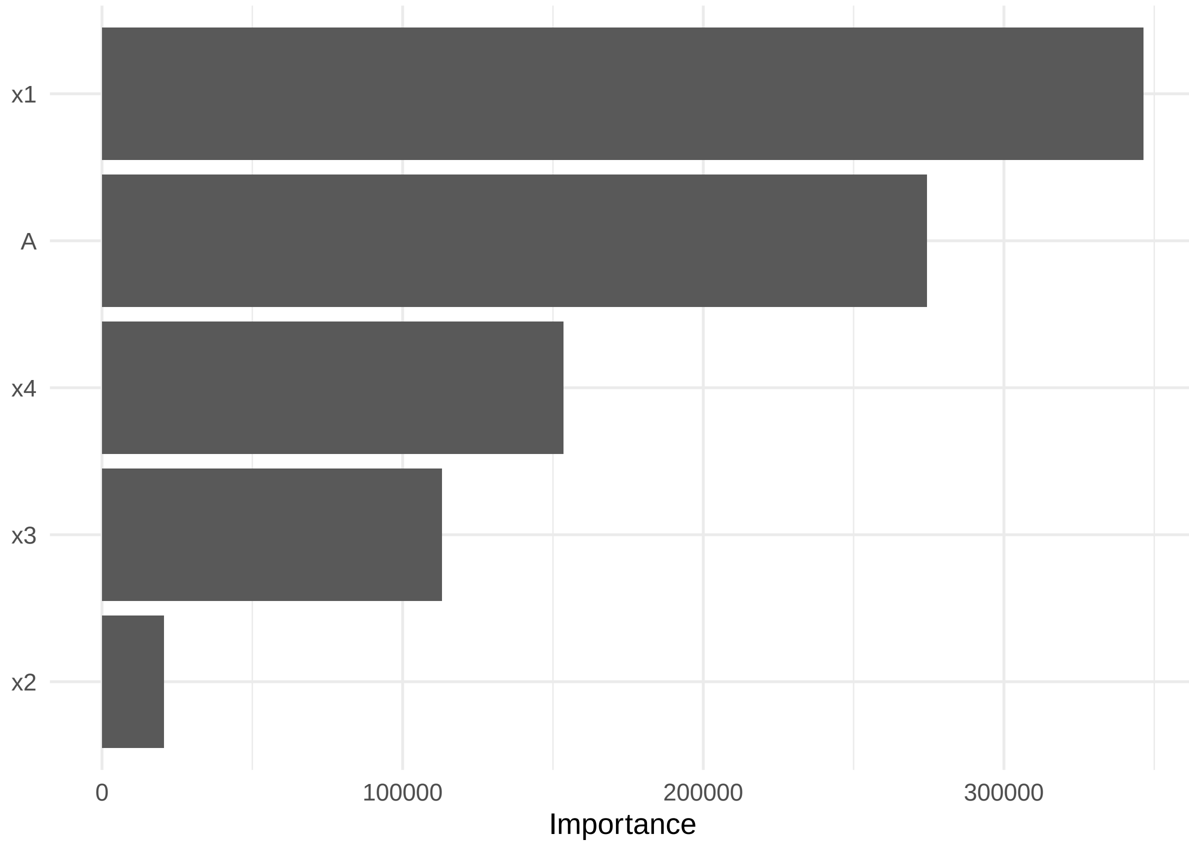


Figure 1: Variable importance for the full data case with $n = 500$ and $SD = 1$, dpi = 300

0.3.0.2 Full data modeling [case 2 when n = 500 and sd = 45]

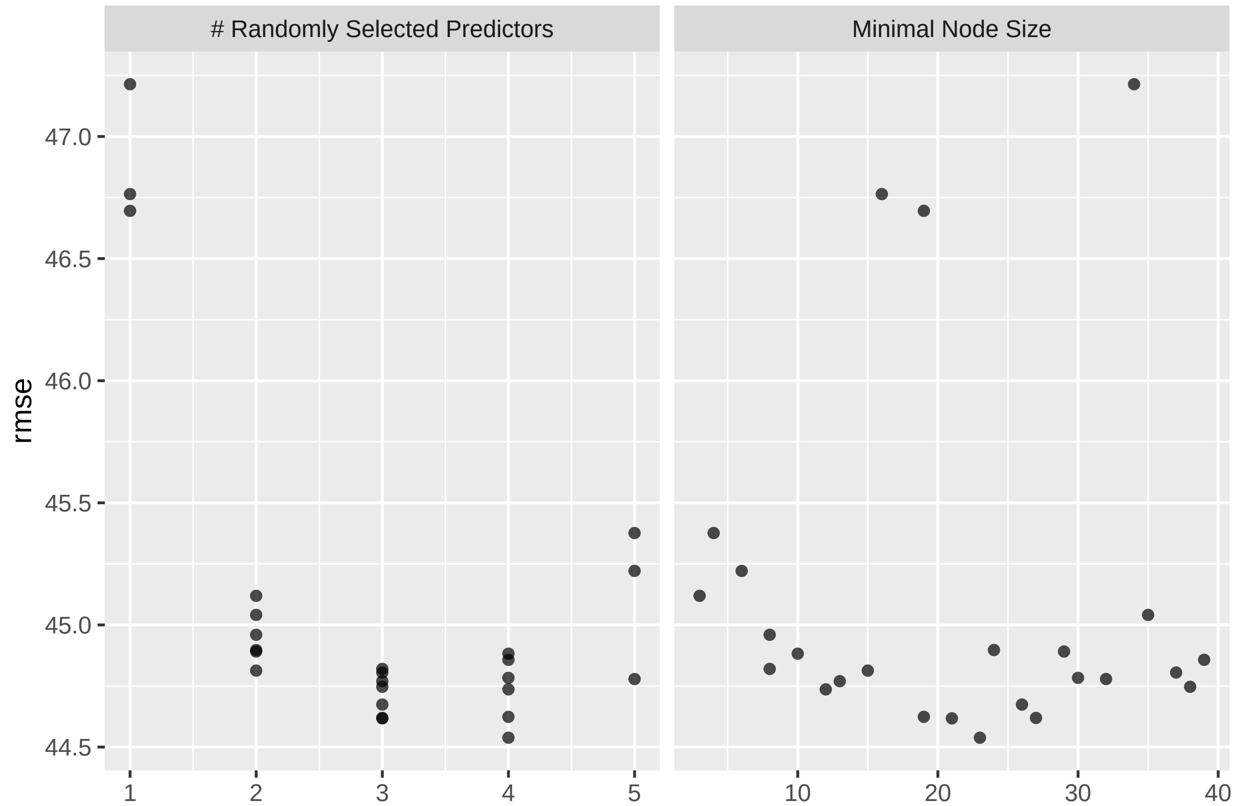
- We use a space-filling grid design and search parameters across a grid of 25 models using cross-validation.
- Based on this procedure, the best tuning parameters with n = 1000 trees were `min_n = 23` and `mtry = 4`

```
## using data with n = 500 and sd = 45 (df_two)

train <- df_two[, -7]


## create the random forest model object
rf_model <- rand_forest(trees = 1000, mtry = tune(), min_n = tune()) %>%
  set_engine("ranger", importance = "impurity") %>%
  set_mode("regression")

## create the workflow
rf_workflow <- workflow() %>%
  add_model(rf_model) %>%
  add_formula(y ~ A + x1 + x2 + x3 + x4)
```

```r
## create the procedure for validating the model
val_set <- validation_split(train, prop = 0.80)

## set up the set of metrics to gather from the models [there is no mse; can't use
↪  accuracy too which is for class]
metrics <- metric_set(rmse, rsq) ## rsq=coefficient of determination =


## create the re-sampling folds for hyper-parameter tuning
set.seed(345)
folds <- vfold_cv(train, v = 10)


## fit re-samples and estimate the hyper parameters
doParallel::registerDoParallel()  ## leverage parallel processing

set.seed(456)
rf_results <- rf_workflow %>%
  tune_grid(val_set,
            grid = 25,
            resamples = folds,
            control = control_grid(save_pred = TRUE), #saving preds allows collecting the
             ↪  metrics
            metrics = metric_set(rmse))

rf_results %>% show_best(metric = "rmse")
```

```
## # A tibble: 5 x 8
##    mtry min_n .metric .estimator  mean     n std_err .config
##   <int> <int> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1     4    23 rmse    standard    44.5    10   0.875 Preprocessor1_Model14
## 2     3    21 rmse    standard    44.6    10   0.911 Preprocessor1_Model11
## 3     3    27 rmse    standard    44.6    10   0.882 Preprocessor1_Model07
## 4     4    19 rmse    standard    44.6    10   0.873 Preprocessor1_Model08
## 5     3    26 rmse    standard    44.7    10   0.907 Preprocessor1_Model02
```

```r
## can plot the best models
autoplot(rf_results)
```




```
## rf best model based on accuracy
rf_best = rf_results %>%
  select_best(metric = "rmse")
rf_best
```

```
## # A tibble: 1 x 3
##    mtry min_n .config
##   <int> <int> <chr>
## 1     4    23 Preprocessor1_Model14
```

```
## summarize variable importance
final_md2 <- finalize_model(rf_model, rf_best)
options(scipen = 999)
full_two <- final_md2 %>%
 set_engine("ranger", importance = "impurity") %>%
  fit(y ~ .,
    data = df_two[, -7]) %>%
  vip::vip(geom = "col") + theme_minimal()
full_two
```

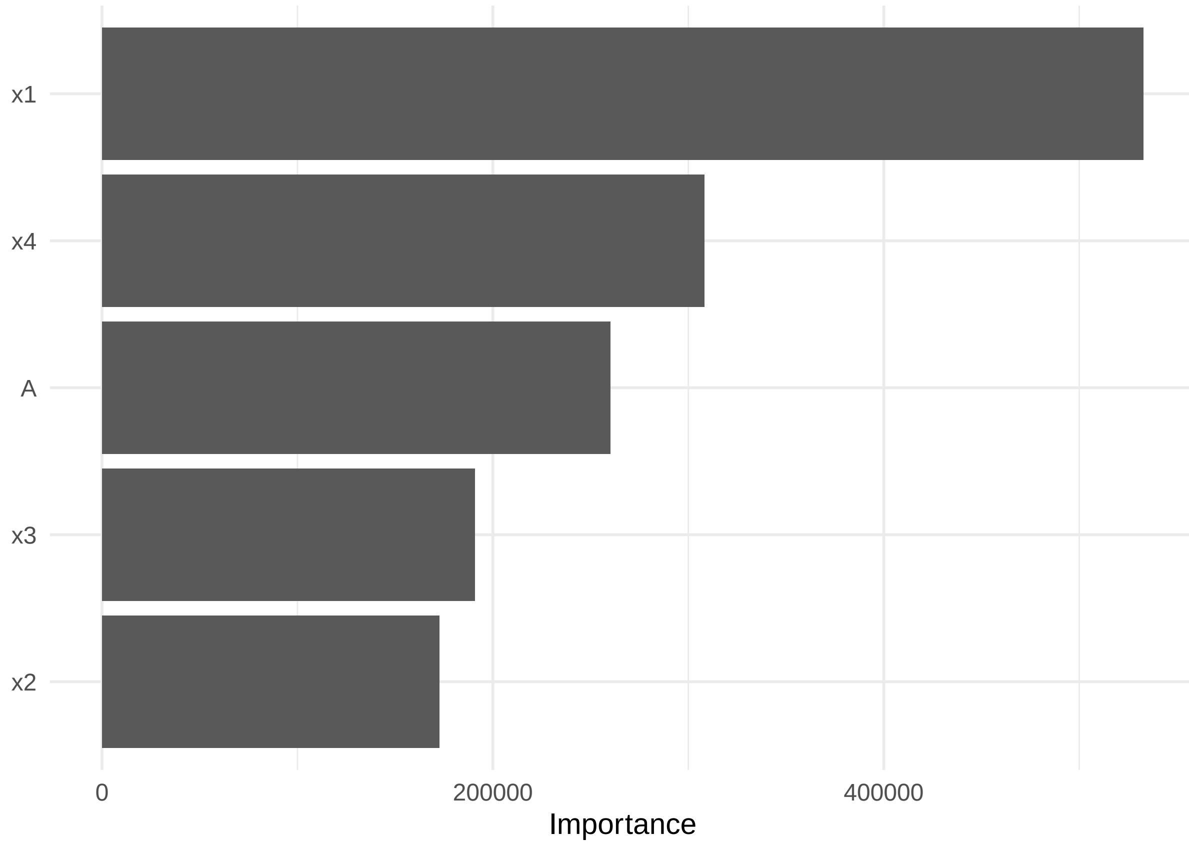


Figure 2: Variable importance for the full data case with $n = 500$ and $SD = 45$, dpi = 300

0.3.0.3 Full data modeling [case 3 when n = 2000 and sd = 1]

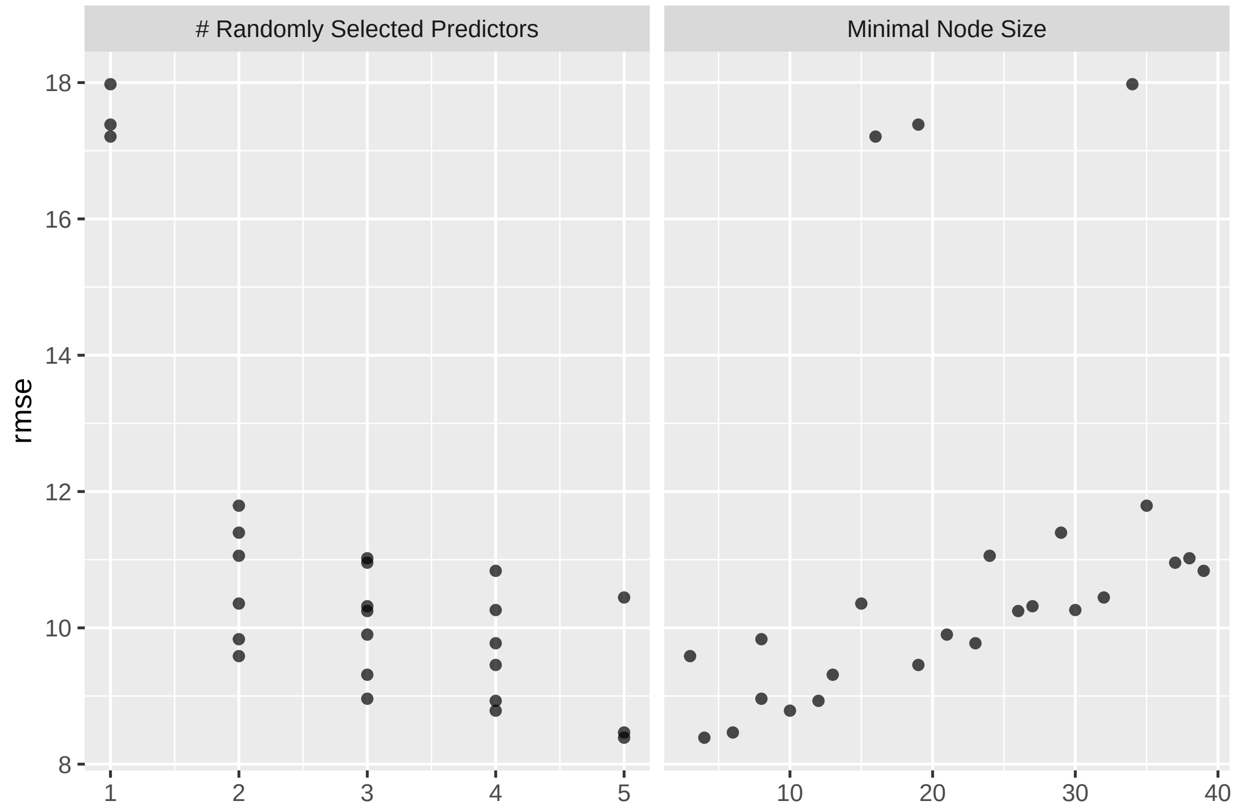
- We use a space-filling grid design and search parameters across a grid of 25 models using cross-validation.
- Based on this procedure, the best tuning parameters with n = 1000 trees were `min_n = 4` and `mtry = 5`

```r
## using data with n = 2000 and sd = 1 (df_three)

train <- df_three[, -7]


## create the random forest model object
rf_model <- rand_forest(trees = 1000, mtry = tune(), min_n = tune()) %>%
  set_engine("ranger", importance = "impurity") %>%
  set_mode("regression")

## create the workflow
rf_workflow <- workflow() %>%
  add_model(rf_model) %>%
  add_formula(y ~ A + x1 + x2 + x3 + x4)
```

```r
## create the procedure for validating the model
val_set <- validation_split(train, prop = 0.80)

## set up the set of metrics to gather from the models [there is no mse; can't use
↪  accuracy too which is for class]
metrics <- metric_set(rmse, rsq) ## rsq=coefficient of determination =


## create the re-sampling folds for hyper-parameter tuning
set.seed(345)
folds <- vfold_cv(train, v = 10)


## fit re-samples and estimate the hyper parameters
doParallel::registerDoParallel()  ## leverage parallel processing

set.seed(456)
rf_results <- rf_workflow %>%
  tune_grid(val_set,
            grid = 25,
            resamples = folds,
            control = control_grid(save_pred = TRUE), #saving preds allows collecting the
             ↪  metrics
            metrics = metric_set(rmse))

rf_results %>% show_best(metric = "rmse")
```

```
## # A tibble: 5 x 8
##    mtry min_n .metric .estimator  mean     n std_err .config
##   <int> <int> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1     5     4 rmse    standard    8.39    10   0.254 Preprocessor1_Model01
## 2     5     6 rmse    standard    8.46    10   0.253 Preprocessor1_Model20
## 3     4    10 rmse    standard    8.78    10   0.268 Preprocessor1_Model04
## 4     4    12 rmse    standard    8.93    10   0.275 Preprocessor1_Model22
## 5     3     8 rmse    standard    8.96    10   0.298 Preprocessor1_Model25
```

```r
## can plot the best models
autoplot(rf_results)
```




```r
## rf best model based on accuracy
rf_best = rf_results %>%
  select_best(metric = "rmse")
rf_best
```

```
## # A tibble: 1 x 3
##    mtry min_n .config
##   <int> <int> <chr>
## 1     5     4 Preprocessor1_Model01
```

```r
## summarize variable importance
final_md3 <- finalize_model(rf_model, rf_best)
options(scipen = 999)
full_three <- final_md3 %>%
 set_engine("ranger", importance = "impurity") %>%
  fit(y ~ .,
    data = df_three[, -7]) %>%
  vip::vip(geom = "col") + theme_minimal()
full_three
```

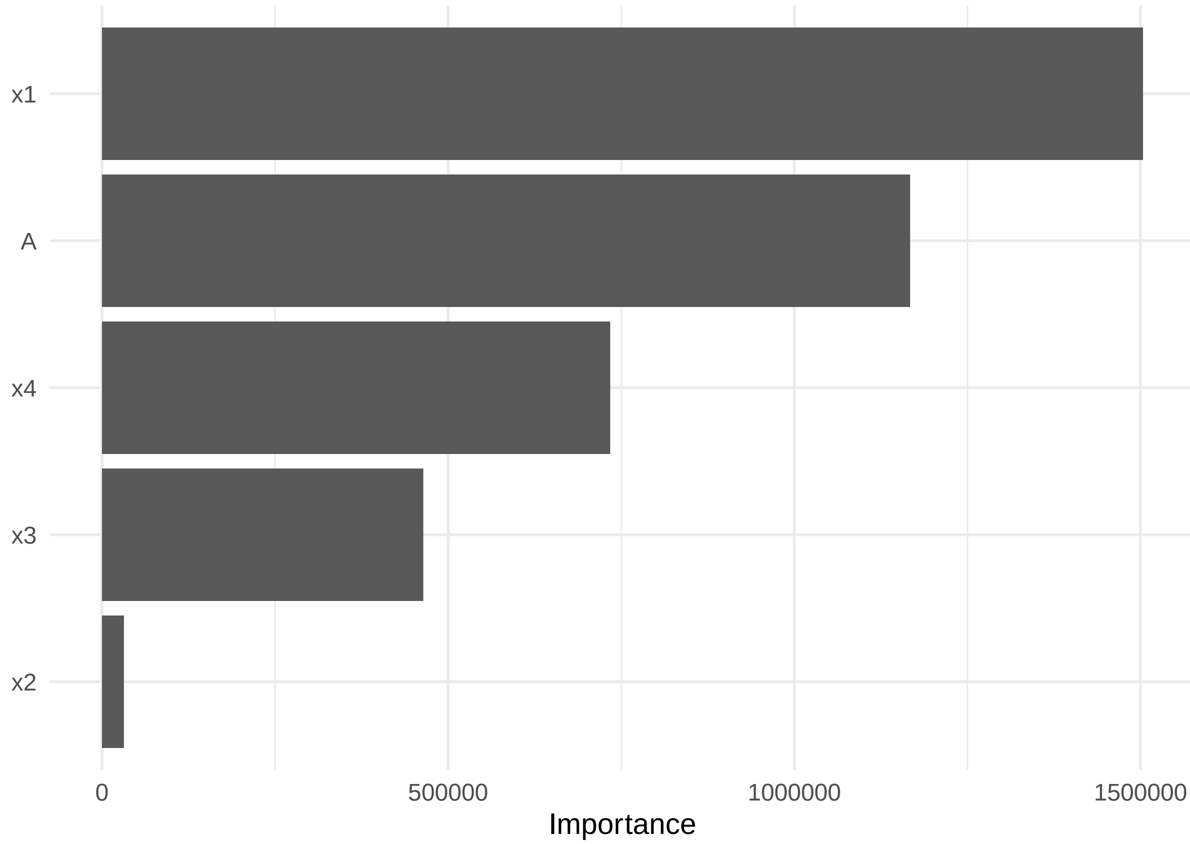


Figure 3: Variable importance for the full data case with $n = 2000$ and $SD = 1$, dpi = 300

0.3.0.4 Full data modeling [case 4 when n = 2000 and sd = 45]

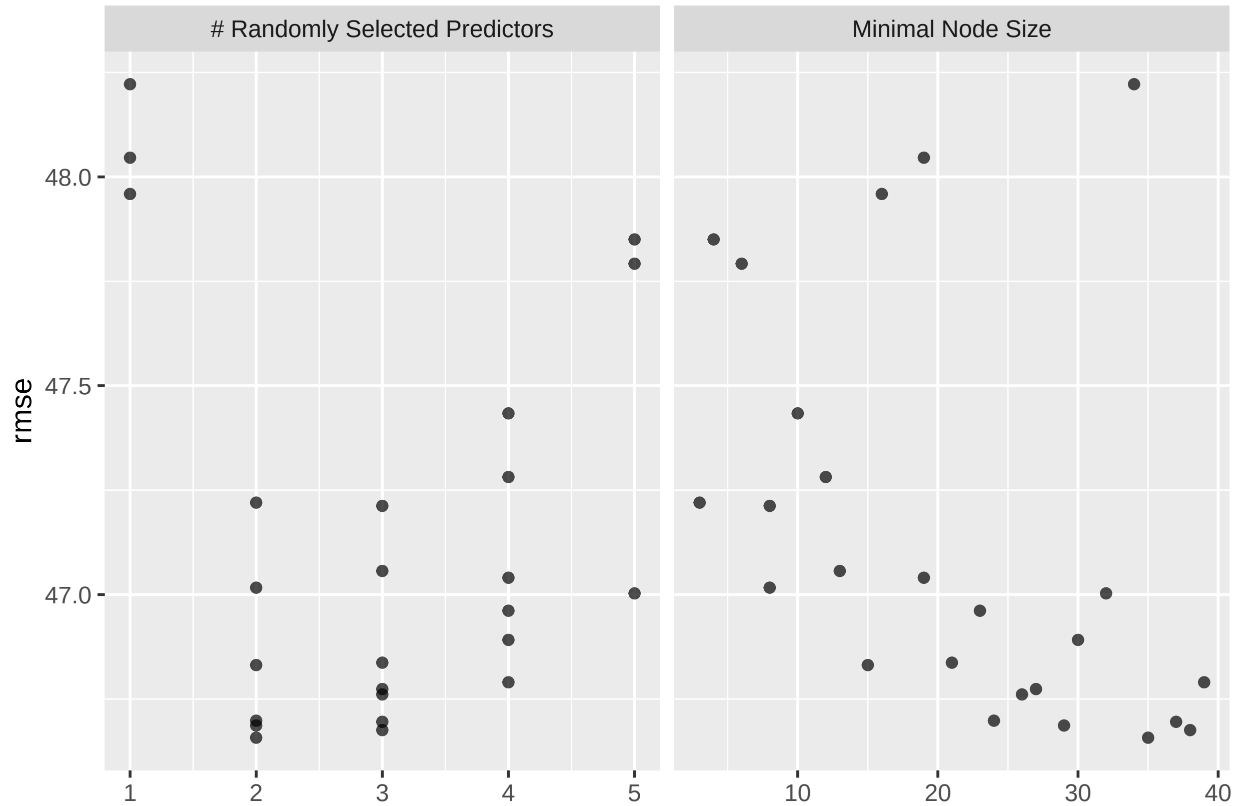
- We use a space-filling grid design and search parameters across a grid of 25 models using cross-validation.
- Based on this procedure, the best tuning parameters with n = 1000 trees were `min_n = 35` and `mtry = 2`

```
## using data with n = 2000 and sd = 45 (df_three)

train <- df_four[, -7]


## create the random forest model object
rf_model <- rand_forest(trees = 1000, mtry = tune(), min_n = tune()) %>%
  set_engine("ranger", importance = "impurity") %>%
  set_mode("regression")

## create the workflow
rf_workflow <- workflow() %>%
  add_model(rf_model) %>%
  add_formula(y ~ A + x1 + x2 + x3 + x4)
```

```r
## create the procedure for validating the model
val_set <- validation_split(train, prop = 0.80)

## set up the set of metrics to gather from the models [there is no mse; can't use
↪  accuracy too which is for class]
metrics <- metric_set(rmse, rsq) ## rsq=coefficient of determination =


## create the re-sampling folds for hyper-parameter tuning
set.seed(345)
folds <- vfold_cv(train, v = 10)


## fit re-samples and estimate the hyper parameters
doParallel::registerDoParallel()  ## leverage parallel processing

set.seed(456)
rf_results <- rf_workflow %>%
  tune_grid(val_set,
            grid = 25,
            resamples = folds,
            control = control_grid(save_pred = TRUE), #saving preds allows collecting the
             ↪  metrics
            metrics = metric_set(rmse))

rf_results %>% show_best(metric = "rmse")
```

```
## # A tibble: 5 x 8
##    mtry min_n .metric .estimator  mean     n std_err .config
##   <int> <int> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1     2    35 rmse    standard    46.7    10   0.637 Preprocessor1_Model23
## 2     3    38 rmse    standard    46.7    10   0.608 Preprocessor1_Model03
## 3     2    29 rmse    standard    46.7    10   0.635 Preprocessor1_Model15
## 4     3    37 rmse    standard    46.7    10   0.607 Preprocessor1_Model10
## 5     2    24 rmse    standard    46.7    10   0.632 Preprocessor1_Model18
```

```r
## can plot the best models
autoplot(rf_results)
```




```r
## rf best model based on accuracy
rf_best = rf_results %>%
  select_best(metric = "rmse")
rf_best
```

```
## # A tibble: 1 x 3
##    mtry min_n .config
##   <int> <int> <chr>
## 1     2    35 Preprocessor1_Model23
```

```r
## summarize variable importance
final_md4 <- finalize_model(rf_model, rf_best)
options(scipen = 999)
full_four <- final_md4 %>%
 set_engine("ranger", importance = "impurity") %>%
  fit(y ~ .,
    data = df_four[, -7]) %>%
  vip::vip(geom = "col") + theme_minimal()
full_four
```

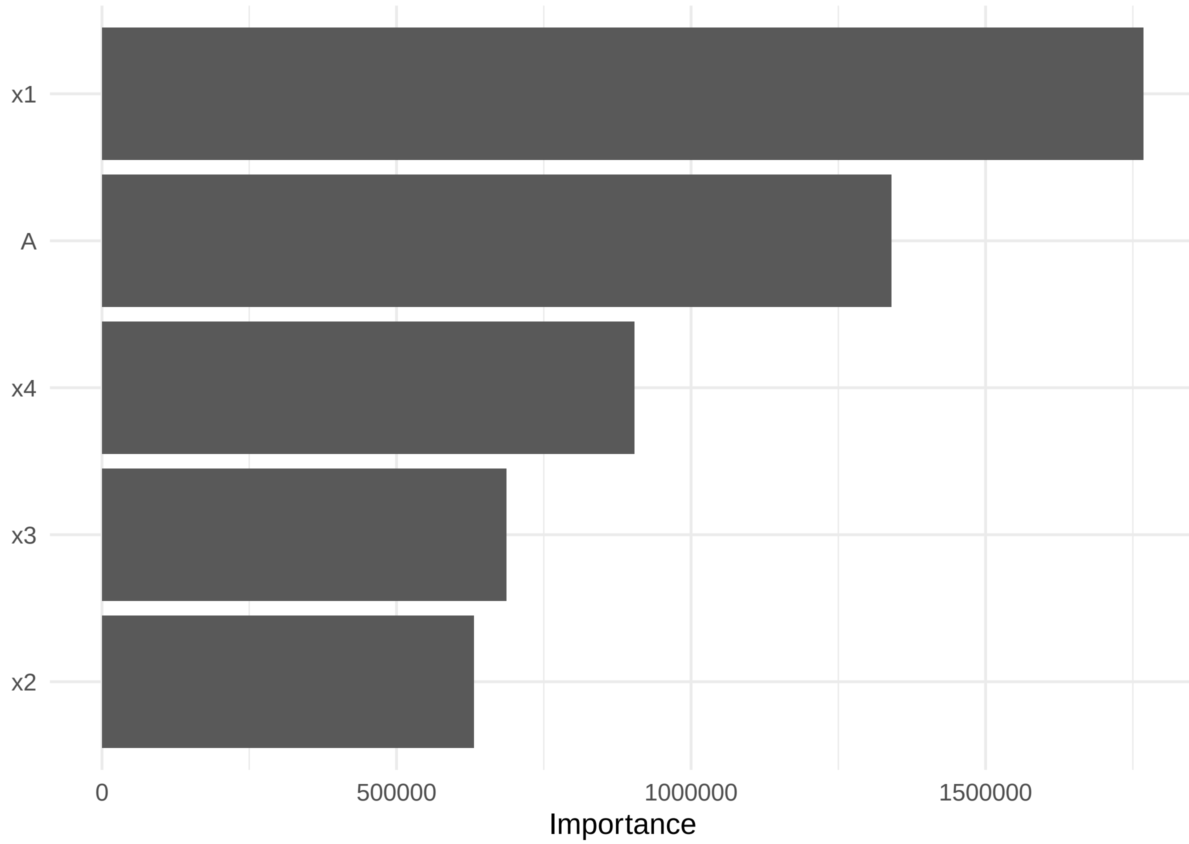


Figure 4: Variable importance for the full data case with $n = 2000$ and $SD = 45$, dpi = 300

```
## create a grid of variable importance across all the cases under full data
↪ hyper-parameter tuning
#jpeg("full_vip.jpeg", width = 4, height = 4, units = 'in', res = 300)
#plot(x, y) # Make plot
#full_vip = grid.arrange(full_one, full_two, full_three, full_four, ncol = 2)
#full_vip
#dev.off()
```

0.4 TUNING ON OBSERVED DATA

0.4.0.1 [case 1 when n = 500 and sd = 1]

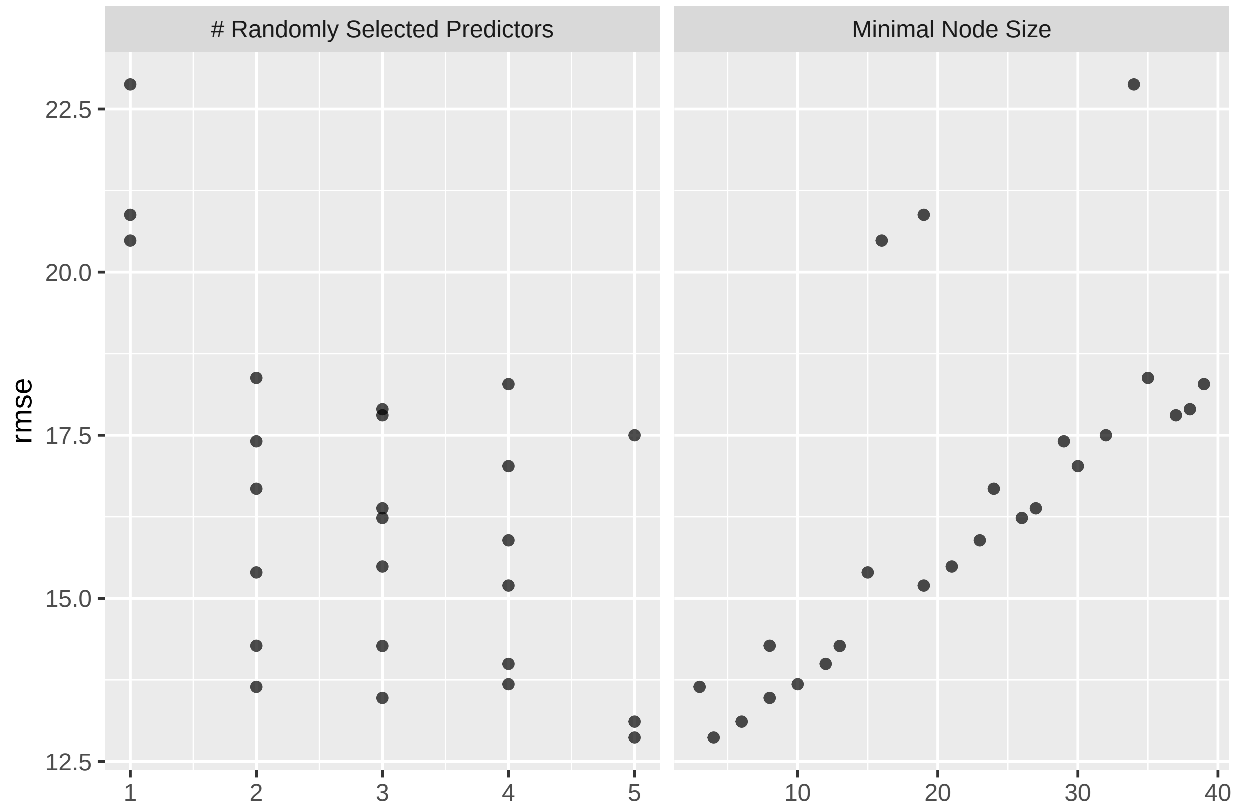
- All model parameter tuning are based on the full data.
- Parameters are explored for all possible scenarios on the full data set with 10-fold cross-validation.
- We use a space-filling grid design and search parameters across a grid of 25 models using cross-validation.

```
## using data with n = 500 and sd = 1 (df_one)

train1 <- base::subset(df_one, R == 1)
```

```
train <- train1[, -7]


## create the random forest model object
rf_model <- rand_forest(trees = 1000, mtry = tune(), min_n = tune()) %>%
  set_engine("ranger", importance = "impurity") %>%
  set_mode("regression")

## create the workflow
rf_workflow <- workflow() %>%
  add_model(rf_model) %>%
  add_formula(y ~ A + x1 + x2 + x3 + x4)

## create the procedure for validating the model
val_set <- validation_split(train, prop = 0.80)

## set up the set of metrics to gather from the models [there is no mse; can't use
↪  accuracy too which is for class]
metrics <- metric_set(rmse, rsq) ## rsq=coefficient of determination =


## create the re-sampling folds for hyper-parameter tuning
set.seed(345)
folds <- vfold_cv(train, v = 10)


## fit re-samples and estimate the hyper parameters
doParallel::registerDoParallel()  ## leverage parallel processing

set.seed(456)
rf_results <- rf_workflow %>%
  tune_grid(val_set,
            grid = 25,
            resamples = folds,
            control = control_grid(save_pred = TRUE), #saving preds allows collecting the
             ↪  metrics
            metrics = metric_set(rmse))

rf_results %>% show_best(metric = "rmse")
```

```
## # A tibble: 5 x 8
##    mtry min_n .metric .estimator  mean     n std_err .config
##   <int> <int> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1     5     4 rmse    standard    12.9    10   0.792 Preprocessor1_Model01
## 2     5     6 rmse    standard    13.1    10   0.805 Preprocessor1_Model20
## 3     3     8 rmse    standard    13.5    10   0.782 Preprocessor1_Model25
## 4     2     3 rmse    standard    13.6    10   0.724 Preprocessor1_Model13
## 5     4    10 rmse    standard    13.7    10   0.801 Preprocessor1_Model04
```

```
## can plot the best models
autoplot(rf_results)
```




```
## rf best model based on accuracy
rf_best = rf_results %>%
  select_best(metric = "rmse")
rf_best
```

```
## # A tibble: 1 x 3
##    mtry min_n .config
##   <int> <int> <chr>
## 1     5     4 Preprocessor1_Model01
```

```
## summarize variable importance
final_obs <- finalize_model(rf_model, rf_best)
options(scipen = 999)
obs_one <- final_obs %>%
 set_engine("ranger", importance = "impurity") %>%
  fit(y ~ .,
    data = base::subset(df_one, R == 1)[, -7]) %>%
  vip::vip(geom = "col") + theme_minimal()
obs_one
```

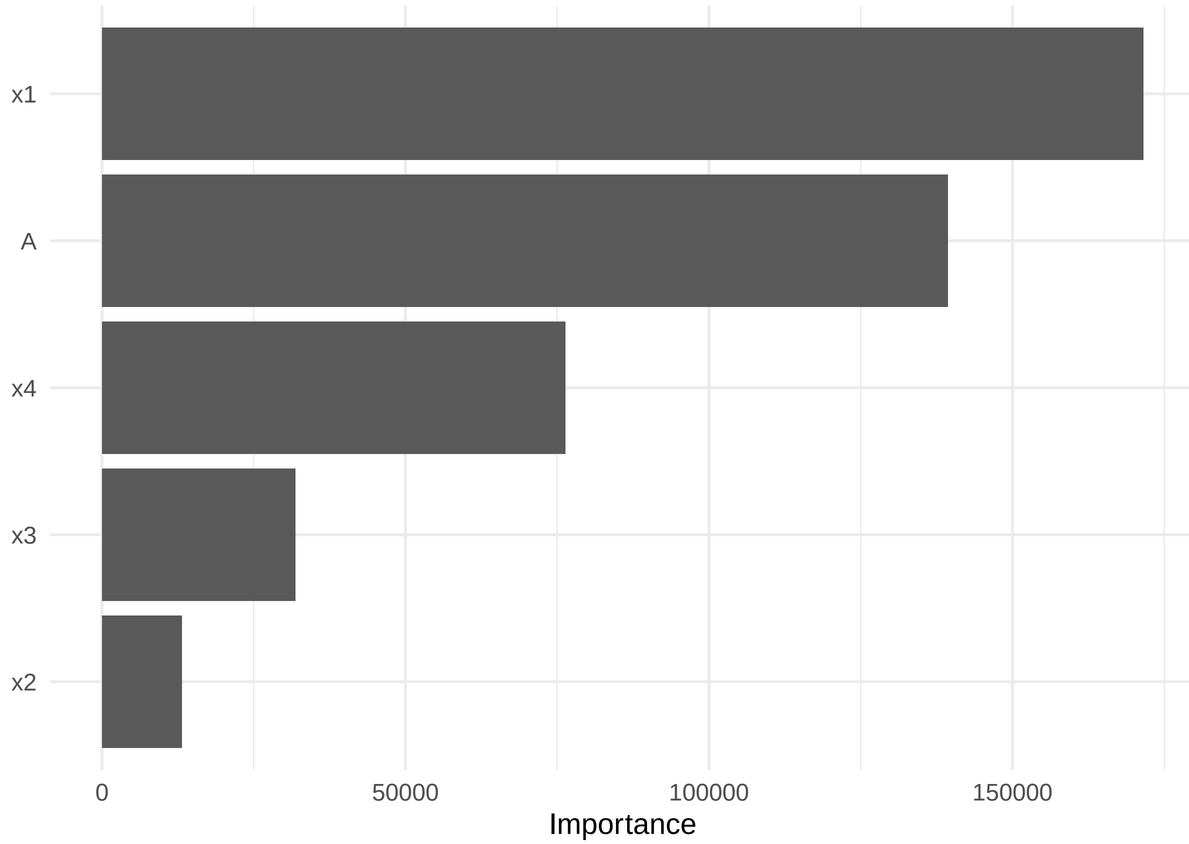


Figure 5: Variable importance for the observed data case with $n = 500$ and $SD = 1$, dpi = 300

0.4.0.2 [case 2 when n = 500 and sd = 45]

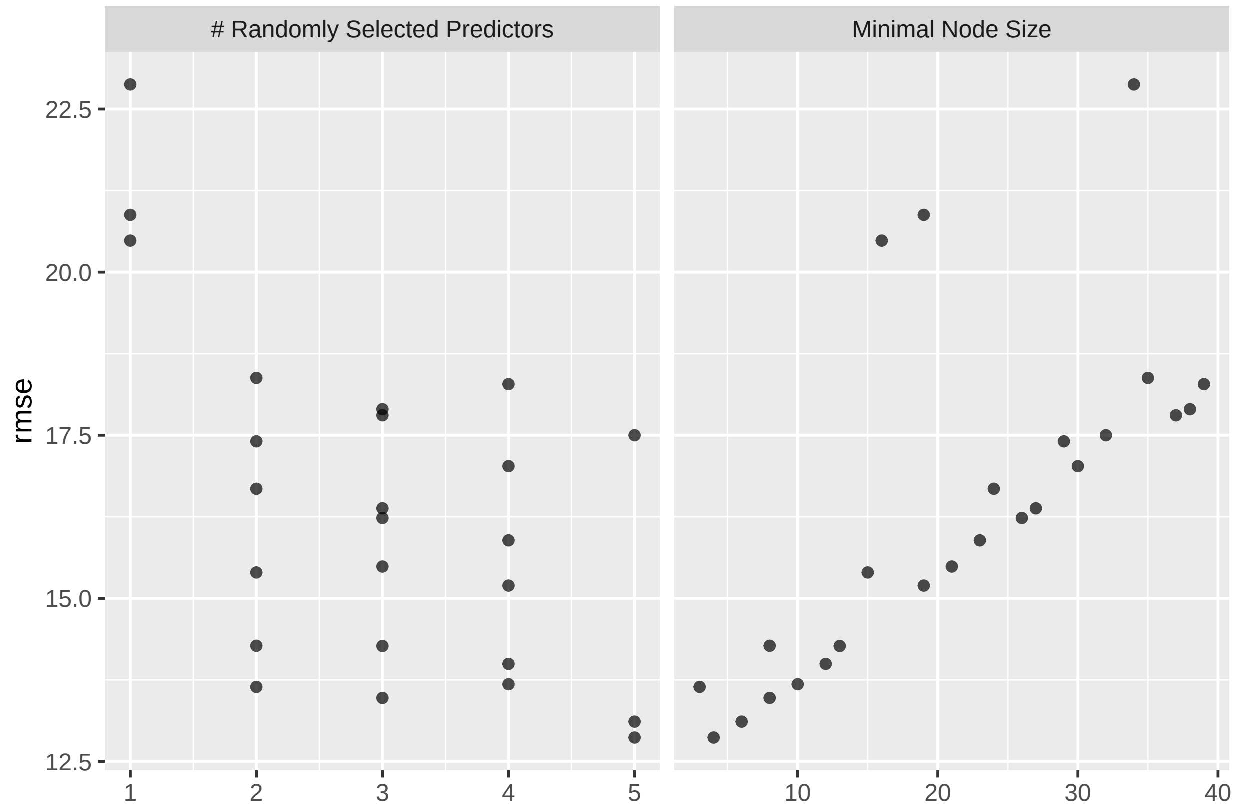
- We use a space-filling grid design and search parameters across a grid of 25 models using cross-validation.
- Based on this procedure, the best tuning parameters with n = 1000 trees were `min_n = 23` and `mtry = 4`

```
## using data with n = 500 and sd = 45 (df_two)

train1 <- base::subset(df_two, R == 1)
train = train[, -7]


## create the random forest model object
rf_model <- rand_forest(trees = 1000, mtry = tune(), min_n = tune()) %>%
  set_engine("ranger", importance = "impurity") %>%
  set_mode("regression")

## create the workflow
rf_workflow <- workflow() %>%
  add_model(rf_model) %>%
  add_formula(y ~ A + x1 + x2 + x3 + x4)
```

```r
## create the procedure for validating the model
val_set <- validation_split(train, prop = 0.80)

## set up the set of metrics to gather from the models [there is no mse; can't use
↪  accuracy too which is for class]
metrics <- metric_set(rmse, rsq) ## rsq=coefficient of determination =


## create the re-sampling folds for hyper-parameter tuning
set.seed(345)
folds <- vfold_cv(train, v = 10)


## fit re-samples and estimate the hyper parameters
doParallel::registerDoParallel()  ## leverage parallel processing

set.seed(456)
rf_results <- rf_workflow %>%
  tune_grid(val_set,
            grid = 25,
            resamples = folds,
            control = control_grid(save_pred = TRUE), #saving preds allows collecting the
             ↪  metrics
            metrics = metric_set(rmse))

rf_results %>% show_best(metric = "rmse")
```

```
## # A tibble: 5 x 8
##    mtry min_n .metric .estimator  mean     n std_err .config
##   <int> <int> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1     5     4 rmse    standard    12.9    10   0.792 Preprocessor1_Model01
## 2     5     6 rmse    standard    13.1    10   0.805 Preprocessor1_Model20
## 3     3     8 rmse    standard    13.5    10   0.782 Preprocessor1_Model25
## 4     2     3 rmse    standard    13.6    10   0.724 Preprocessor1_Model13
## 5     4    10 rmse    standard    13.7    10   0.801 Preprocessor1_Model04
```

```r
## can plot the best models
autoplot(rf_results)
```




```
## rf best model based on accuracy
rf_best = rf_results %>%
  select_best(metric = "rmse")
rf_best
```

```
## # A tibble: 1 x 3
##    mtry min_n .config
##   <int> <int> <chr>
## 1     5     4 Preprocessor1_Model01
```

```
## summarize variable importance
final_obs <- finalize_model(rf_model, rf_best)
options(scipen = 999)
obs_two <- final_obs %>%
 set_engine("ranger", importance = "impurity") %>%
  fit(y ~ .,
    data = base::subset(df_two, R == 1)[, -7]) %>%
  vip::vip(geom = "col") + theme_minimal()
obs_two
```

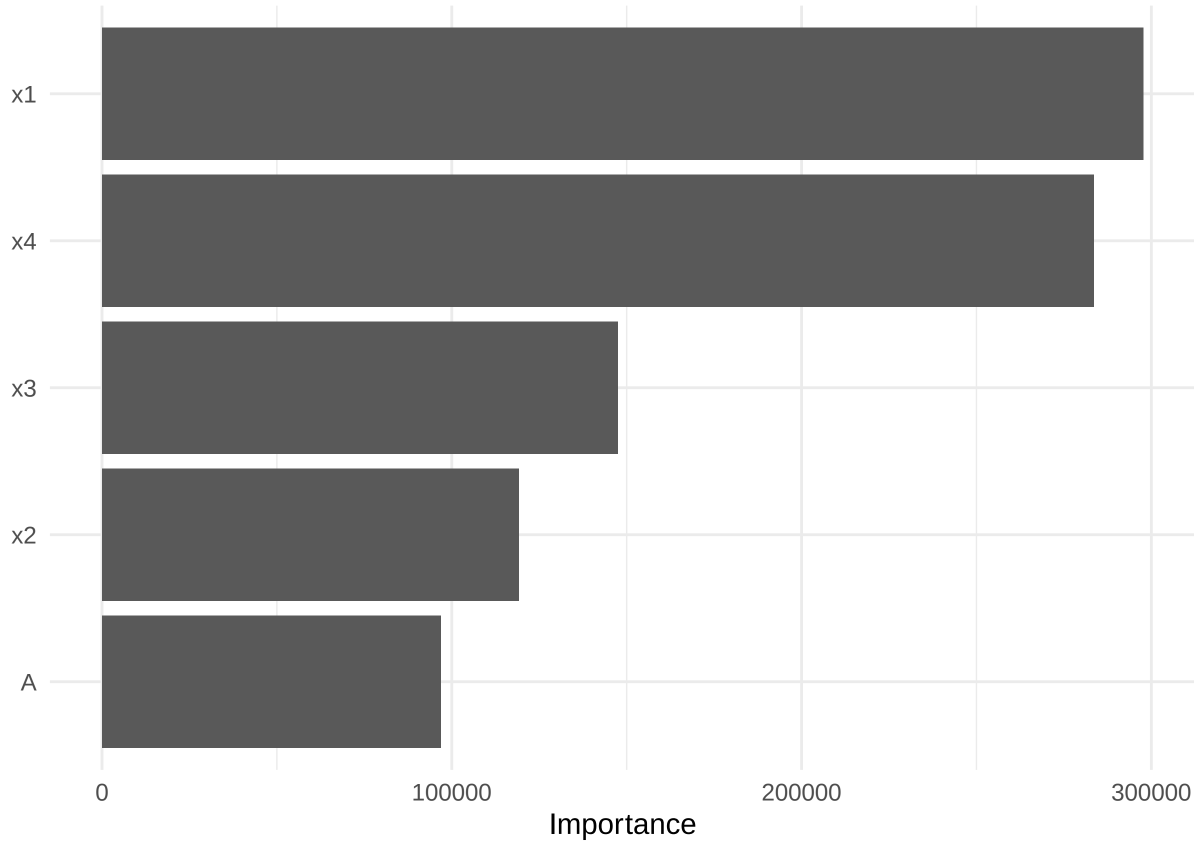


Figure 6: Variable importance for the observed data case with $n = 500$ and $SD = 45$, dpi = 300

0.4.0.3 [case 3 when n = 2000 and sd = 1]

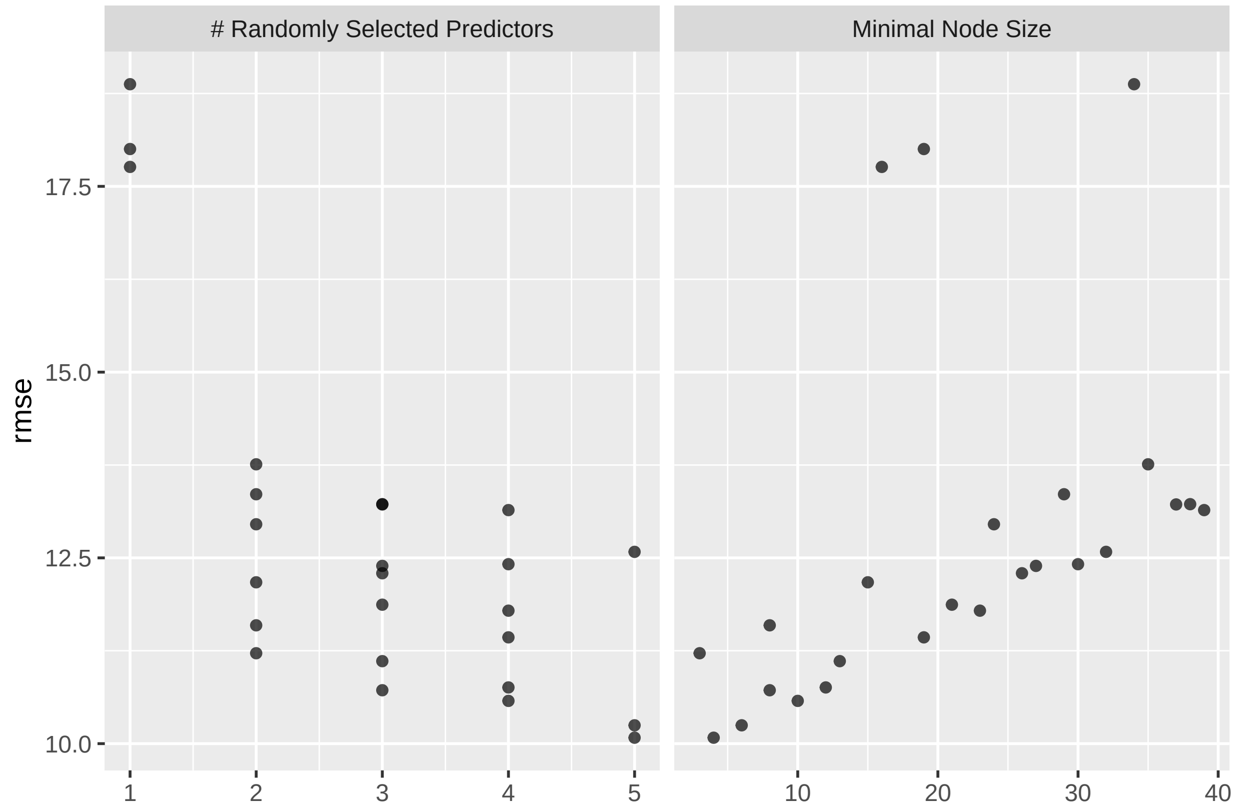
- We use a space-filling grid design and search parameters across a grid of 25 models using cross-validation.
- Based on this procedure, the best tuning parameters with n = 1000 trees were `min_n = 4` and `mtry = 5`

```
## using data with n = 2000 and sd = 1 (df_three)

train1 <- base::subset(df_three, R == 1)
train = train1[, -7]


## create the random forest model object
rf_model <- rand_forest(trees = 1000, mtry = tune(), min_n = tune()) %>%
  set_engine("ranger", importance = "impurity") %>%
  set_mode("regression")

## create the workflow
rf_workflow <- workflow() %>%
  add_model(rf_model) %>%
  add_formula(y ~ A + x1 + x2 + x3 + x4)
```

```r
## create the procedure for validating the model
val_set <- validation_split(train, prop = 0.80)

## set up the set of metrics to gather from the models [there is no mse; can't use
↪  accuracy too which is for class]
metrics <- metric_set(rmse, rsq) ## rsq=coefficient of determination =


## create the re-sampling folds for hyper-parameter tuning
set.seed(345)
folds <- vfold_cv(train, v = 10)


## fit re-samples and estimate the hyper parameters
doParallel::registerDoParallel()  ## leverage parallel processing

set.seed(456)
rf_results <- rf_workflow %>%
  tune_grid(val_set,
            grid = 25,
            resamples = folds,
            control = control_grid(save_pred = TRUE), #saving preds allows collecting the
             ↪  metrics
            metrics = metric_set(rmse))

rf_results %>% show_best(metric = "rmse")
```

```
## # A tibble: 5 x 8
##    mtry min_n .metric .estimator  mean     n std_err .config
##   <int> <int> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1     5     4 rmse    standard    10.1    10   0.342 Preprocessor1_Model01
## 2     5     6 rmse    standard    10.2    10   0.330 Preprocessor1_Model20
## 3     4    10 rmse    standard    10.6    10   0.359 Preprocessor1_Model04
## 4     3     8 rmse    standard    10.7    10   0.357 Preprocessor1_Model25
## 5     4    12 rmse    standard    10.8    10   0.369 Preprocessor1_Model22
```

```r
## can plot the best models
autoplot(rf_results)
```




```
## rf best model based on accuracy
rf_best = rf_results %>%
  select_best(metric = "rmse")
rf_best
```

```
## # A tibble: 1 x 3
##    mtry min_n .config
##   <int> <int> <chr>
## 1     5     4 Preprocessor1_Model01
```

```
## summarize variable importance
final_obs <- finalize_model(rf_model, rf_best)
options(scipen = 999)
obs_three <- final_obs %>%
 set_engine("ranger", importance = "impurity") %>%
  fit(y ~ .,
    data = base::subset(df_three, R == 1)[, -7]) %>%
  vip::vip(geom = "col") + theme_minimal()
obs_three
```

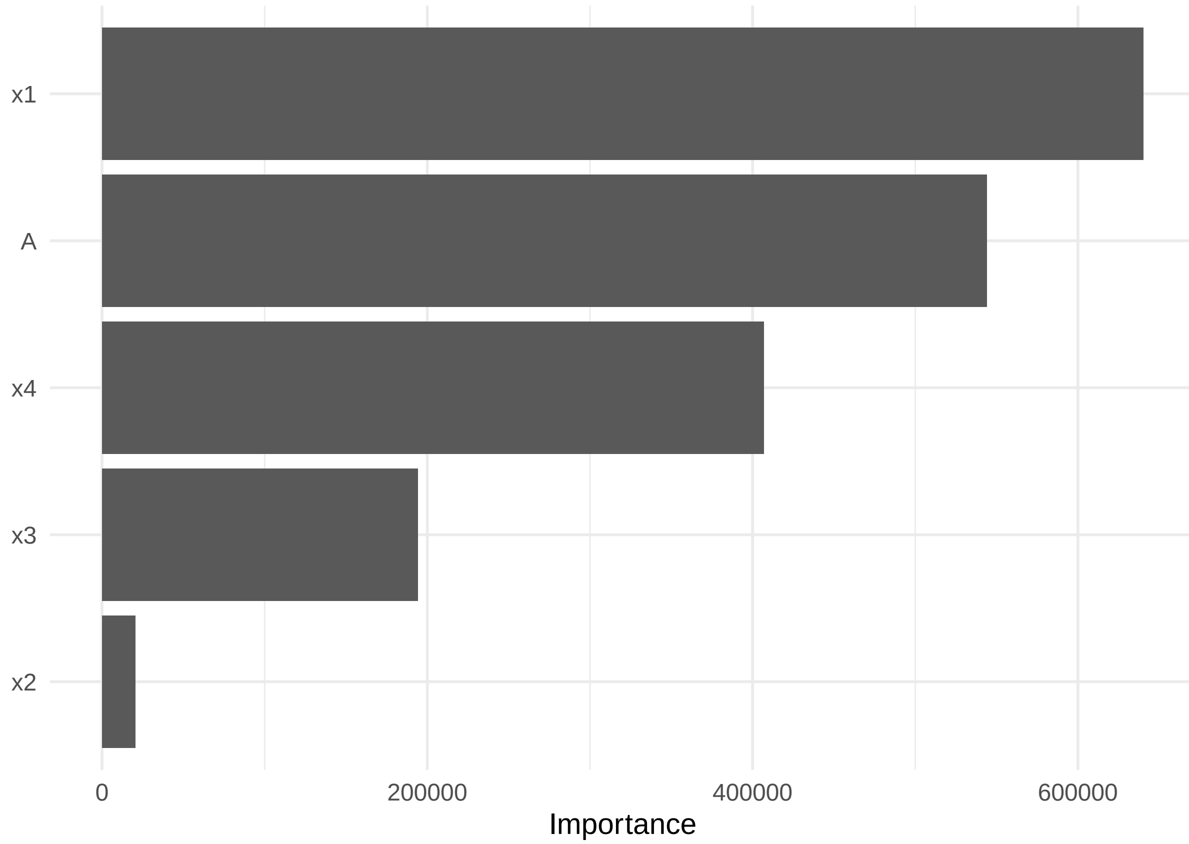


Figure 7: Variable importance for the observed data case with $n = 2000$ and $SD = 1$, dpi = 300

0.4.0.4 [case 4 when n = 2000 and sd = 45]

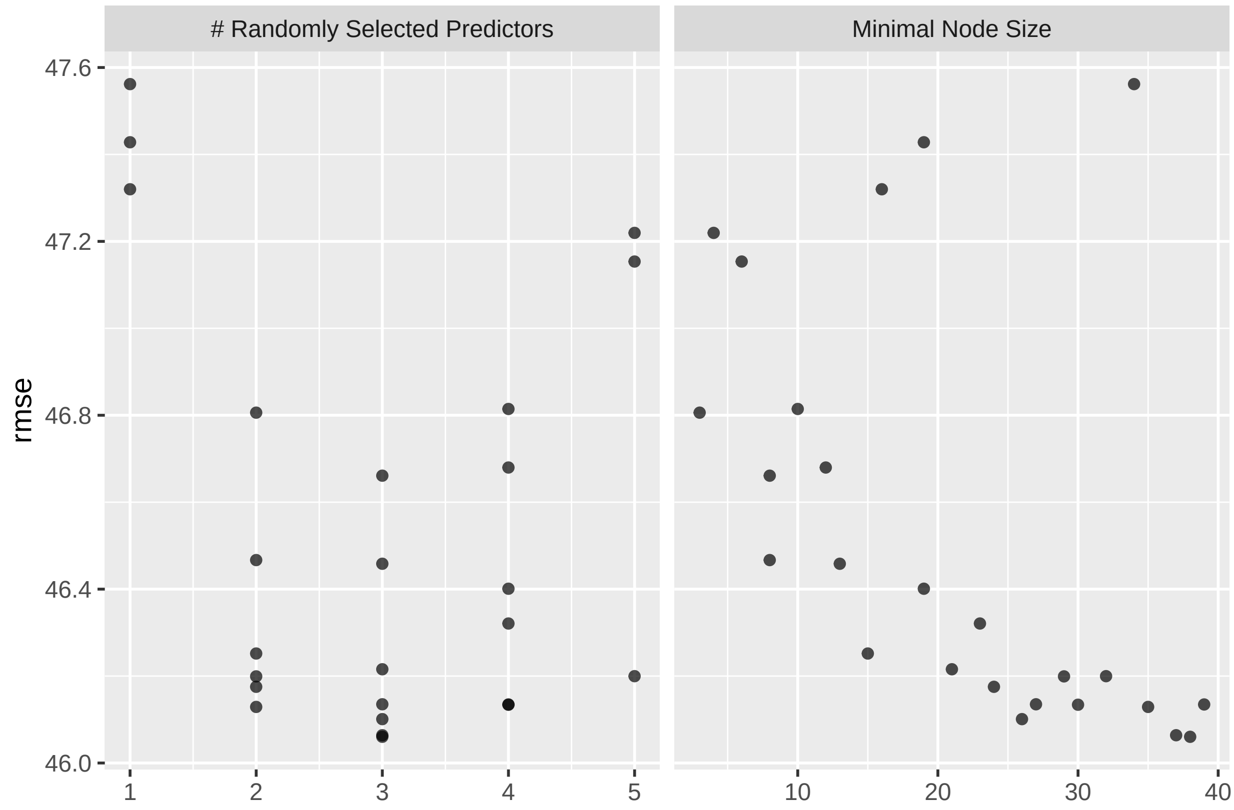
- We use a space-filling grid design and search parameters across a grid of 25 models using cross-validation.
- Based on this procedure, the best tuning parameters with n = 1000 trees were `min_n = 35` and `mtry = 2`

```
## using data with n = 2000 and sd = 45 (df_three)

train1 <- base::subset(df_four, R == 1)
train = train1[, -7]


## create the random forest model object
rf_model <- rand_forest(trees = 1000, mtry = tune(), min_n = tune()) %>%
  set_engine("ranger", importance = "impurity") %>%
  set_mode("regression")

## create the workflow
rf_workflow <- workflow() %>%
  add_model(rf_model) %>%
  add_formula(y ~ A + x1 + x2 + x3 + x4)
```

```r
## create the procedure for validating the model
val_set <- validation_split(train, prop = 0.80)

## set up the set of metrics to gather from the models [there is no mse; can't use
↪  accuracy too which is for class]
metrics <- metric_set(rmse, rsq) ## rsq=coefficient of determination =


## create the re-sampling folds for hyper-parameter tuning
set.seed(345)
folds <- vfold_cv(train, v = 10)


## fit re-samples and estimate the hyper parameters
doParallel::registerDoParallel()  ## leverage parallel processing

set.seed(456)
rf_results <- rf_workflow %>%
  tune_grid(val_set,
            grid = 25,
            resamples = folds,
            control = control_grid(save_pred = TRUE), #saving preds allows collecting the
             ↪  metrics
            metrics = metric_set(rmse))

rf_results %>% show_best(metric = "rmse")
```

```
## # A tibble: 5 x 8
##    mtry min_n .metric .estimator  mean     n std_err .config
##   <int> <int> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1     3    38 rmse    standard    46.1    10   0.605 Preprocessor1_Model03
## 2     3    37 rmse    standard    46.1    10   0.599 Preprocessor1_Model10
## 3     3    26 rmse    standard    46.1    10   0.628 Preprocessor1_Model02
## 4     2    35 rmse    standard    46.1    10   0.668 Preprocessor1_Model23
## 5     4    30 rmse    standard    46.1    10   0.598 Preprocessor1_Model19
```

```r
## can plot the best models
autoplot(rf_results)
```




```r
## rf best model based on accuracy
rf_best = rf_results %>%
  select_best(metric = "rmse")
rf_best
```

```
## # A tibble: 1 x 3
##    mtry min_n .config
##   <int> <int> <chr>
## 1     3    38 Preprocessor1_Model03
```

```r
## summarize variable importance
final_obs <- finalize_model(rf_model, rf_best)
options(scipen = 999)
obs_four <- final_obs %>%
 set_engine("ranger", importance = "impurity") %>%
  fit(y ~ .,
    data = base::subset(df_four, R == 1)[, -7]) %>%
  vip::vip(geom = "col") + theme_minimal()
obs_four
```

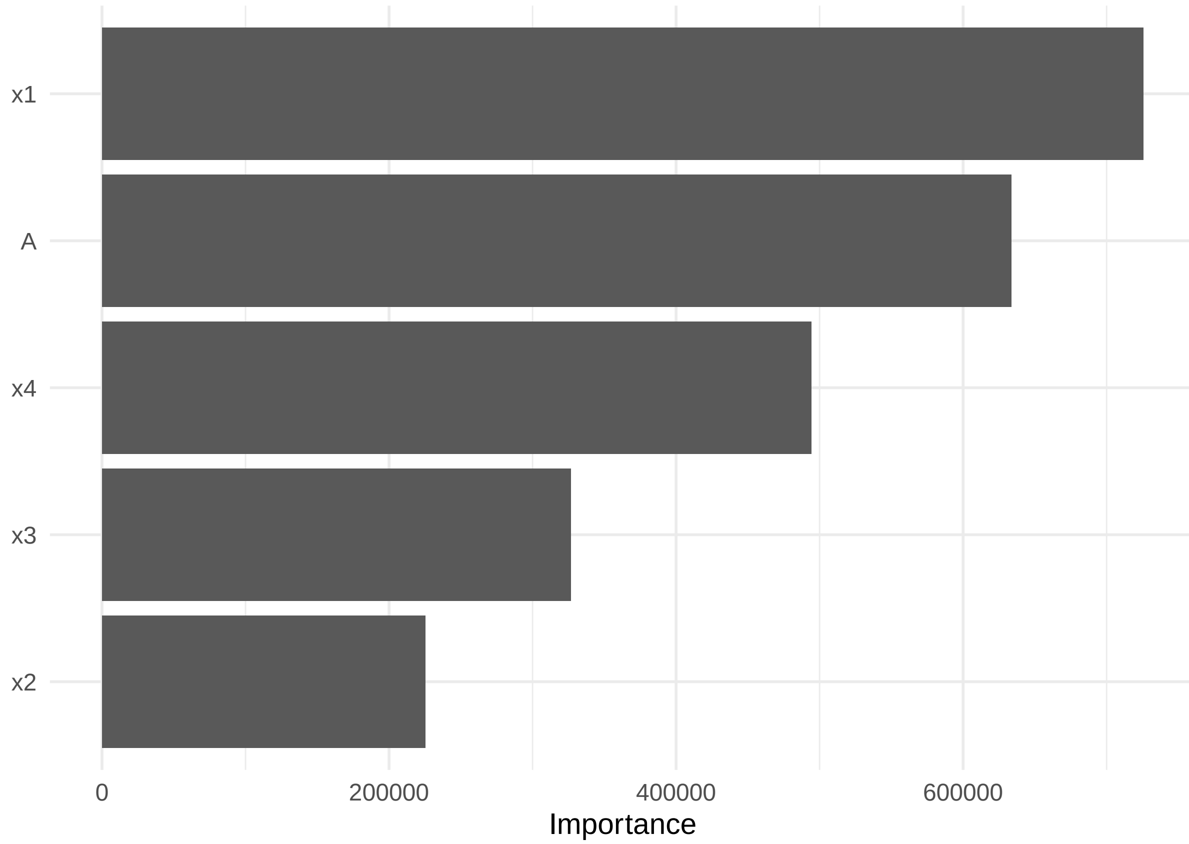


Figure 8: Variable importance for the observed data case with $n = 2000$ and $SD = 45$, dpi = 300

0.5 Variable Importance

0.5.1 Case when n = 500, SD = 1

```
#library(vip)

#final_rf %>%
#  set_engine("ranger", importance = "impurity") %>%
#  fit(y ~ .,
#    data = df_one[, -7]) %>%
#  vip(geom = "point")
```

0.5.2 Case when n = 500, SD = 45

```
#final_rf %>%
#  set_engine("ranger", importance = "impurity") %>%
#  fit(y ~ .,
#    data = df_one[, -7]) %>%
#  vip(geom = "point")
```