Multivariate Adaptive Regression Splines

Data Sets

```
attrition <- attrition %>% mutate_if(is.ordered, factor, order = F)
attrition.h2o <- as.h2o(attrition)
```

```
set.seed(123)

ames <- AmesHousing::make_ames()
ames.h2o <- as.h2o(ames)

ames.split <- initial_split(ames, prop =.7, strata = "Sale_Price")

ames.train <- training(ames.split)
ames.test <- testing(ames.split)
```

Overview

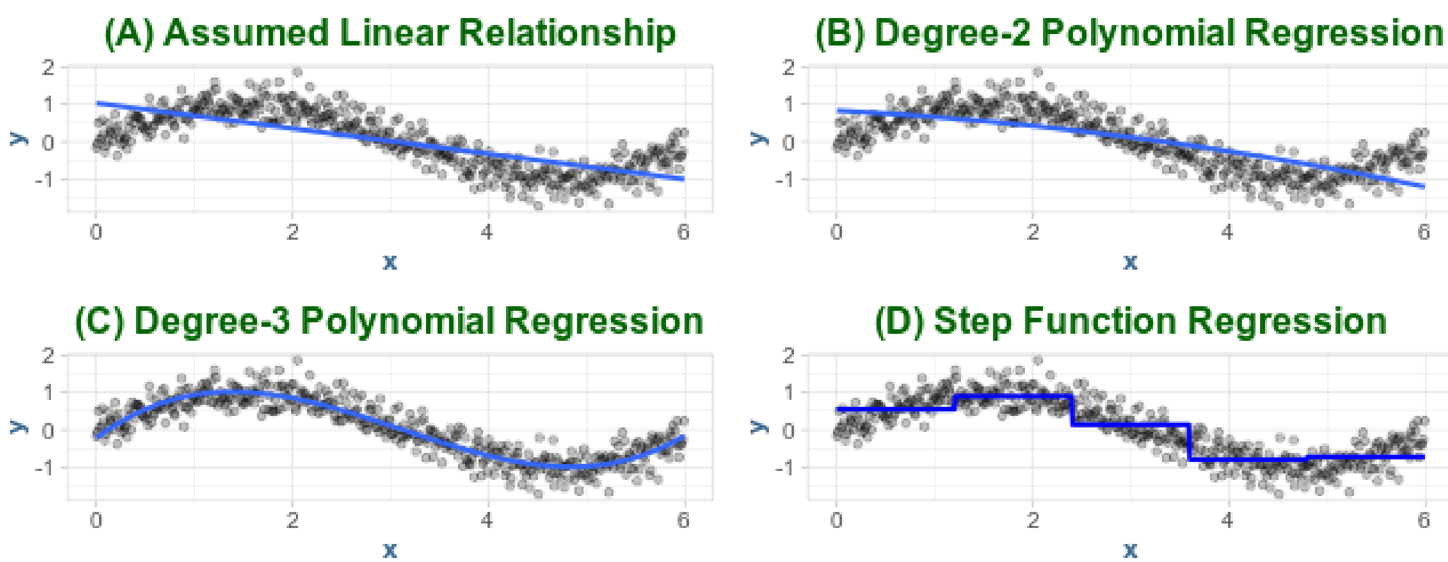
Linear models assume the underlying phenomena we are modeling is intrinsically linear which is not usually true. Multivariate adaptive regression splines (MARS) allow us to model non-linear relationships.

Basic strategies for modeling non-linear fits include polynomial regression and step-wise models.

Visually:

```
set.seed(123)

x <- seq(from = 0, to = 2 * pi, length = 500)
y <- sin(x) + rnorm(length(x), sd = .3)

df <- data.table(x, y) %>%
   filter(x < 6)

p1 <- ggplot(df, aes(x, y)) +
   geom_point(alpha = .25) +
   geom_smooth(method = "lm", se = F) +
   ggtitle("(A) Assumed Linear Relationship")

p2 <- ggplot(df, aes(x, y)) +
   geom_point(alpha = .25) +
   geom_smooth(method = "lm", se = F, formula = y ~ poly(x, 2, raw = T)) +
   ggtitle("(B) Degree-2 Polynomial Regression")
```

```
p3 <- ggplot(df, aes(x, y)) +
   geom_point(alpha = .25) +
   geom_smooth(method = "lm", se = F, formula = y ~ poly(x, 3, raw = T)) +
   ggtitle("(C) Degree-3 Polynomial Regression")

# fit step function model (6 steps)

step_fit <- lm(y ~ cut(x, 5), data = df)
step_pred <- predict(step_fit, df)

p4 <- ggplot(df, aes(x, y)) +
   geom_point(alpha = .25) +
   geom_line(aes(y = step_pred), size = 1, color = "blue") +
   ggtitle("(D) Step Function Regression")

gridExtra::grid.arrange(p1, p2, p3, p4, nrow = 2)
```



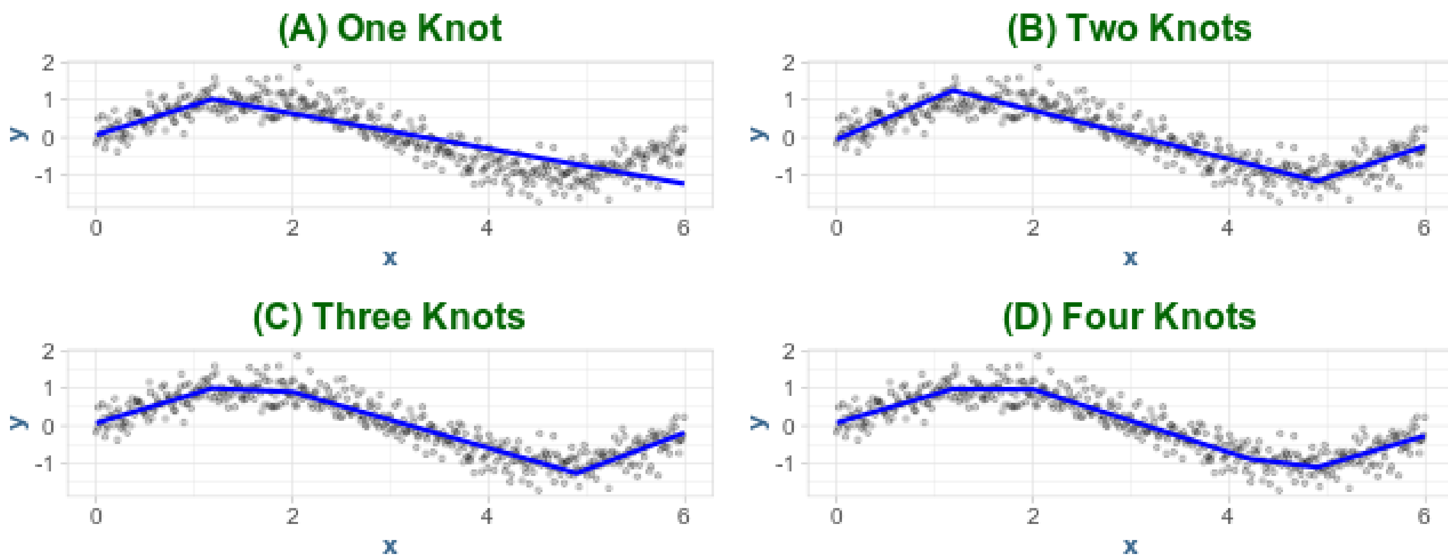
Multivariate Adaptive Regression Splines (MARS)

Similar to the step-wise approach, we can access "knots" in the data to model this behavior.

MARS models with *mda* package.

```
mars1 <- mda::mars(df$x, df$y, nk = 3, prune = F)

p1 <- df %>%
   mutate(predicted = as.vector(mars1$fitted.values)) %>%
   ggplot(aes(x, y)) +
```

```
  geom_point(size = 1, alpha = .2) +
  geom_line(aes(y = predicted), size = 1, color = "blue") +
  ggtitle("(A) One Knot")

mars2 <- mda::mars(df$x, df$y, nk = 5, prune = F)

p2 <- df %>%
  mutate(predicted = as.vector(mars2$fitted.values)) %>%
  ggplot(aes(x, y)) +
  geom_point(size = 1, alpha = .2) +
  geom_line(aes(y = predicted), size = 1, color = "blue") +
  ggtitle("(B) Two Knots")

mars3 <- mda::mars(df$x, df$y, nk = 7, prune = F)

p3 <- df %>%
  mutate(predicted = as.vector(mars3$fitted.values)) %>%
  ggplot(aes(x, y)) +
  geom_point(size = 1, alpha = .2) +
  geom_line(aes(y = predicted), size = 1, color = "blue") +
  ggtitle("(C) Three Knots")

mars4 <- mda::mars(df$x, df$y, nk = 9, prune = F)

p4 <- df %>%
  mutate(predicted = as.vector(mars4$fitted.values)) %>%
    ggplot(aes(x, y)) +
  geom_point(size = 1, alpha = .2) +
  geom_line(aes(y = predicted), size = 1, color = "blue") +
  ggtitle("(D) Four Knots")

gridExtra::grid.arrange(p1, p2, p3, p4, nrow = 2)
```



Fitting a basic MARS model

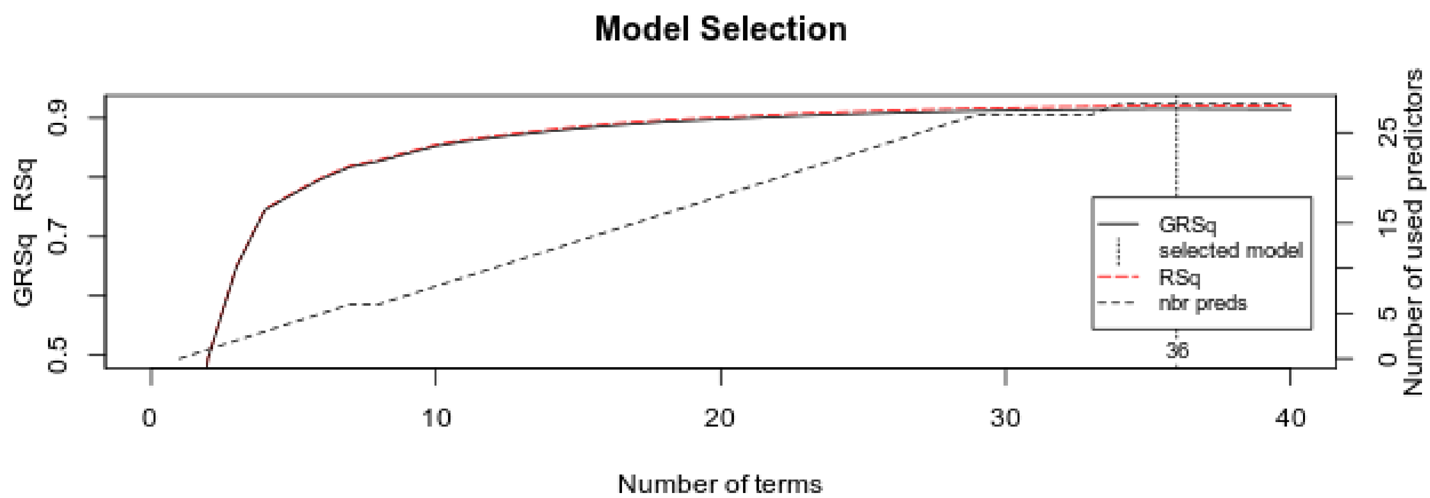
```
mars1 <- earth(
   Sale_Price ~ .,
   data = ames.train
)

print(mars1)
```

```
Selected 36 of 40 terms, and 28 of 307 predictors
Termination condition: RSq changed by less than 0.001 at 40 terms
Importance: Gr_Liv_Area, Year_Built, Total_Bsmt_SF, Overall_QualExcellent, ...
Number of terms at each degree of interaction: 1 35 (additive model)
GCV 547654257    RSS 1047912011488    GRSq 0.9150216    RSq 0.9207205
```

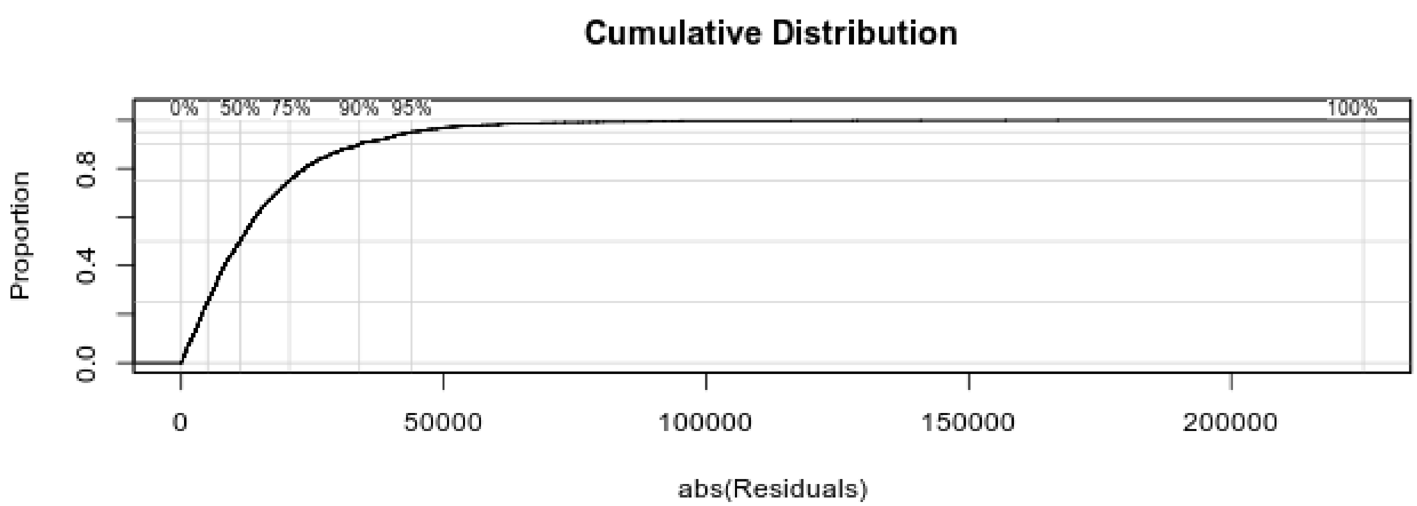
```
summary(mars1) %>% .$coefficients %>% head(10)
```

```
                              Sale_Price
(Intercept)                 234803.84945
h(2787-Gr_Liv_Area)            -50.67882
h(Year_Built-2004)            3595.39940
h(2004-Year_Built)            -373.40103
h(Total_Bsmt_SF-1298)           56.30424
h(1298-Total_Bsmt_SF)          -29.95470
h(Bsmt_Unf_SF-536)             -24.47153
h(536-Bsmt_Unf_SF)              16.28784
Overall_QualExcellent        79769.47075
Overall_QualVery_Excellent 117138.64127
```

```r
plot(mars1, which = 1)
```




```r
plot(mars1, which = 2)
```

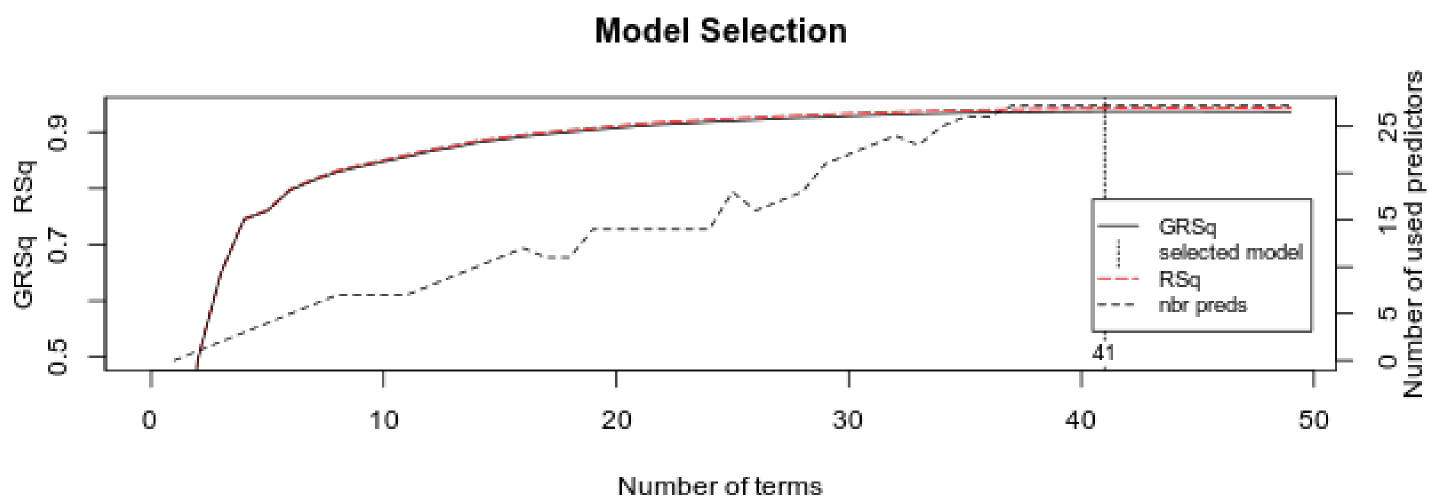



```r
mars2 <- earth(
   Sale_Price ~.,
   data = ames.train,
   degree = 2
)

summary(mars2) %>% .$coefficients %>% head(10)
```

```
                                               Sale_Price
(Intercept)                                256834.408425081
```

```
h(Gr_Liv_Area-2787)                       -144.934131073
h(2787-Gr_Liv_Area)                        -49.666036718
h(Year_Built-2004)                        4548.490902341
h(2004-Year_Built)                        -724.920051940
h(Total_Bsmt_SF-1298)                       80.606680445
h(1298-Total_Bsmt_SF)                      -41.967572528
h(Bsmt_Unf_SF-1017)*h(2787-Gr_Liv_Area)     -0.023524460
h(1017-Bsmt_Unf_SF)*h(2787-Gr_Liv_Area)      0.008535269
Condition_1Norm*h(Gr_Liv_Area-2787)        278.504549507
```

```r
plot(mars2, which = 1)
```



Tuning

As always, we will use a cross-validated grid search procedure to tune the hyperparameters.

First pass:

```r
hyper.grid <- expand.grid(
   degree = 1:3,
   nprune = seq(2, 100, length.out = 10) %>% floor()
)

head(hyper.grid)
```

```
  degree nprune
1      1      2
2      2      2
3      3      2
4      1     12
```

```
5      2     12
6      3     12
```

```r
set.seed(123)

suppressWarnings(print({
cv.mars <- train(
   x = ames.train %>% select(-Sale_Price),
   y = ames.train$Sale_Price,
   method = "earth",
   metric = "RMSE",
   trControl = trainControl(method = "cv", number = 10),
   tuneGrid = hyper.grid
)}))
```

```
Multivariate Adaptive Regression Spline

2053 samples
  80 predictor

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 1846, 1848, 1848, 1848, 1848, 1848, ...
Resampling results across tuning parameters:

  degree  nprune  RMSE      Rsquared   MAE
  1         2     56943.09  0.4964264  39557.80
  1        12     31634.16  0.8447279  20207.96
  1        23     28407.87  0.8732735  17936.68
  1        34     28235.91  0.8753639  17250.10
  1        45     28140.74  0.8759693  17065.57
  1        56     28140.74  0.8759693  17065.57
  1        67     28140.74  0.8759693  17065.57
  1        78     28140.74  0.8759693  17065.57
  1        89     28140.74  0.8759693  17065.57
  1       100     28140.74  0.8759693  17065.57
  2         2     56150.99  0.5111693  39532.38
  2        12     31572.67  0.8450372  20993.65
  2        23     29873.41  0.8629176  17870.00
  2        34     28410.99  0.8722307  16630.36
  2        45     28051.85  0.8752016  16189.05
  2        56     27899.28  0.8762783  16171.49
  2        67     27899.28  0.8762783  16171.49
  2        78     27899.28  0.8762783  16171.49
  2        89     27899.28  0.8762783  16171.49
  2       100     27899.28  0.8762783  16171.49
```

```
  3         2     56571.66  0.5061590  39916.57
  3        12     33411.42  0.8279242  22001.04
  3        23     30829.04  0.8545617  18542.74
  3        34     29795.30  0.8621228  17312.10
  3        45     29288.71  0.8667649  16845.19
  3        56     29322.13  0.8665347  16811.13
  3        67     29322.13  0.8665347  16811.13
  3        78     29322.13  0.8665347  16811.13
  3        89     29322.13  0.8665347  16811.13
  3       100     29322.13  0.8665347  16811.13

RMSE was used to select the optimal model using the smallest value.
The final values used for the model were nprune = 56 and degree = 2.
```

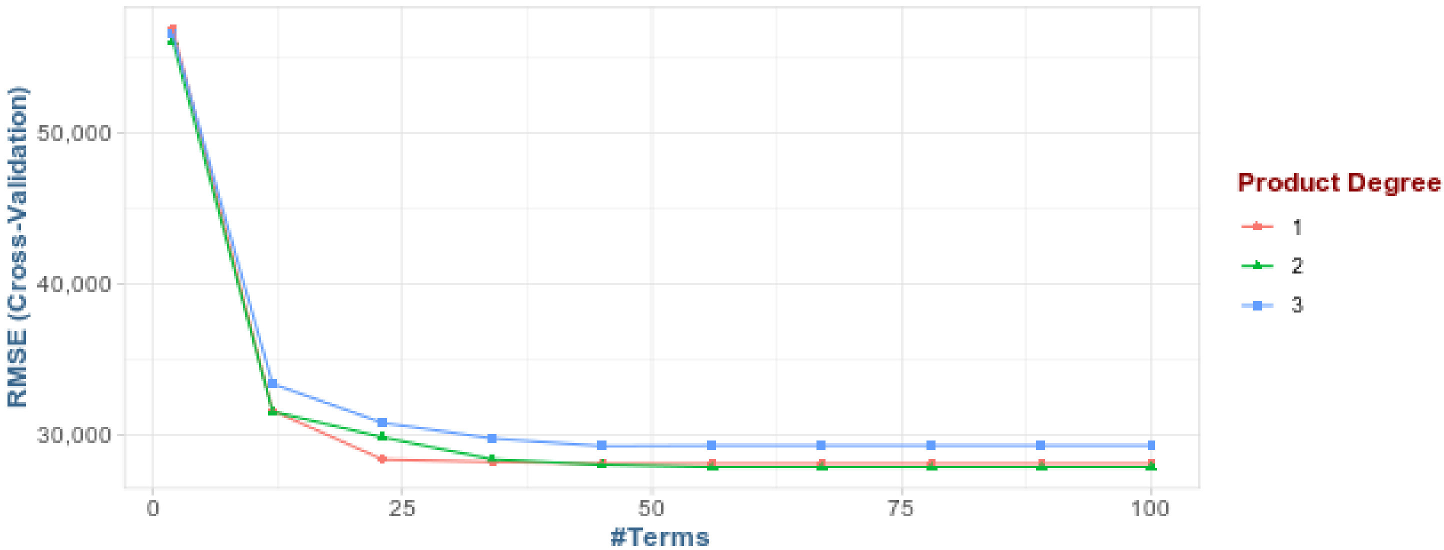
```r
cv.mars$bestTune
```

```
   nprune degree
16     56      2
```

```r
cv.mars$results %>%
   as_tibble() %>%
   arrange(RMSE)
```

```
# A tibble: 30 x 8
   degree nprune   RMSE Rsquared    MAE RMSESD RsquaredSD MAESD
    <int>  <dbl>  <dbl>    <dbl>  <dbl>  <dbl>      <dbl> <dbl>
 1      2     56 27899.    0.876 16171. 15103.     0.129  2354.
 2      2     67 27899.    0.876 16171. 15103.     0.129  2354.
 3      2     78 27899.    0.876 16171. 15103.     0.129  2354.
 4      2     89 27899.    0.876 16171. 15103.     0.129  2354.
 5      2    100 27899.    0.876 16171. 15103.     0.129  2354.
 6      2     45 28052.    0.875 16189. 15525.     0.132  2378.
 7      1     45 28141.    0.876 17066.  8845.     0.0757 1915.
 8      1     56 28141.    0.876 17066.  8845.     0.0757 1915.
 9      1     67 28141.    0.876 17066.  8845.     0.0757 1915.
10      1     78 28141.    0.876 17066.  8845.     0.0757 1915.
# ... with 20 more rows
```

```r
ggplot(cv.mars) +
   scale_y_continuous(labels = scales::comma)
```



Refinement:

```
refine.grid <- expand.grid(
   degree = 1,
   nprune = seq(from = 30, to = 45)
)

suppressWarnings(print({
   cv.mars2 <- train(
      x = ames.train %>% select(-Sale_Price),
      y = ames.train$Sale_Price,
      method = "earth",
      metric = "RMSE",
      trControl = trainControl(method = "cv", number = 10),
      tuneGrid = refine.grid
   )
}))
```

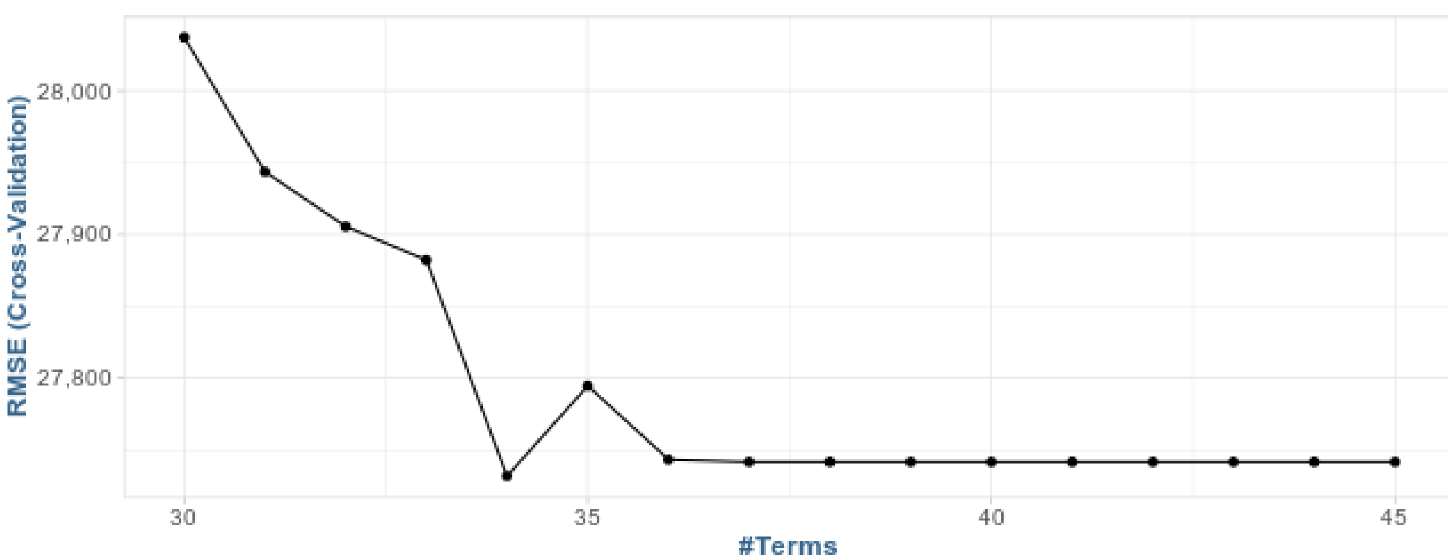
```
Multivariate Adaptive Regression Spline

2053 samples
  80 predictor

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 1846, 1848, 1848, 1847, 1848, 1849, ...
Resampling results across tuning parameters:

  nprune  RMSE      Rsquared   MAE
  30      28037.08  0.8768351  17239.50
```

```
  31      27943.54  0.8777220  17170.66
  32      27905.51  0.8780454  17059.47
  33      27882.12  0.8786504  17016.62
  34      27731.86  0.8798819  16922.83
  35      27794.61  0.8793995  16910.55
  36      27743.34  0.8798244  16872.24
  37      27741.71  0.8798384  16872.96
  38      27741.71  0.8798384  16872.96
  39      27741.71  0.8798384  16872.96
  40      27741.71  0.8798384  16872.96
  41      27741.71  0.8798384  16872.96
  42      27741.71  0.8798384  16872.96
  43      27741.71  0.8798384  16872.96
  44      27741.71  0.8798384  16872.96
  45      27741.71  0.8798384  16872.96

Tuning parameter 'degree' was held constant at a value of 1
RMSE was used to select the optimal model using the smallest value.
The final values used for the model were nprune = 34 and degree = 1.
```

```r
cv.mars2$bestTune
```

```
  nprune degree
5     34      1
```

```r
ggplot(cv.mars2) +
   scale_y_continuous(labels = scales::comma)
```



Feature Interpretation

Variable Importance Plots

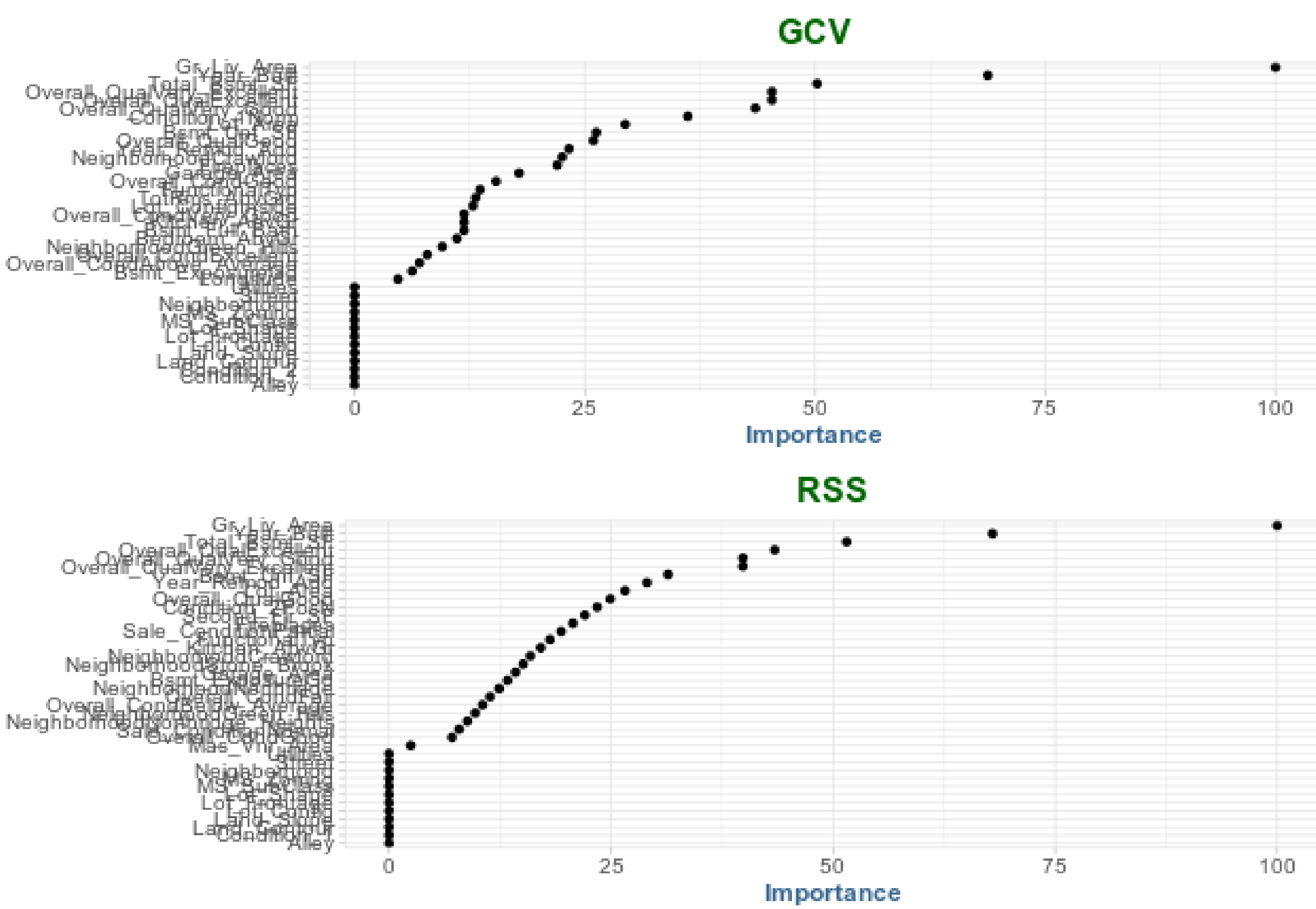
```r
p1 <- vip::vip(cv.mars, num_features = 40, bar = F, value = "gcv") +
  ggtitle("GCV")
```

```
Warning in vip.default(cv.mars, num_features = 40, bar = F, value = "gcv"): The
`bar` argument has been deprecated in favor of the new `geom` argument. It will
be removed in version 0.3.0.
```

```r
p2 <- vip::vip(cv.mars2, num_features = 40, bar = F, value = "rss") +
  ggtitle("RSS")
```

```
Warning in vip.default(cv.mars2, num_features = 40, bar = F, value = "rss"): The
`bar` argument has been deprecated in favor of the new `geom` argument. It will
be removed in version 0.3.0.
```

```r
gridExtra::grid.arrange(p1, p2, nrow = 2)
```



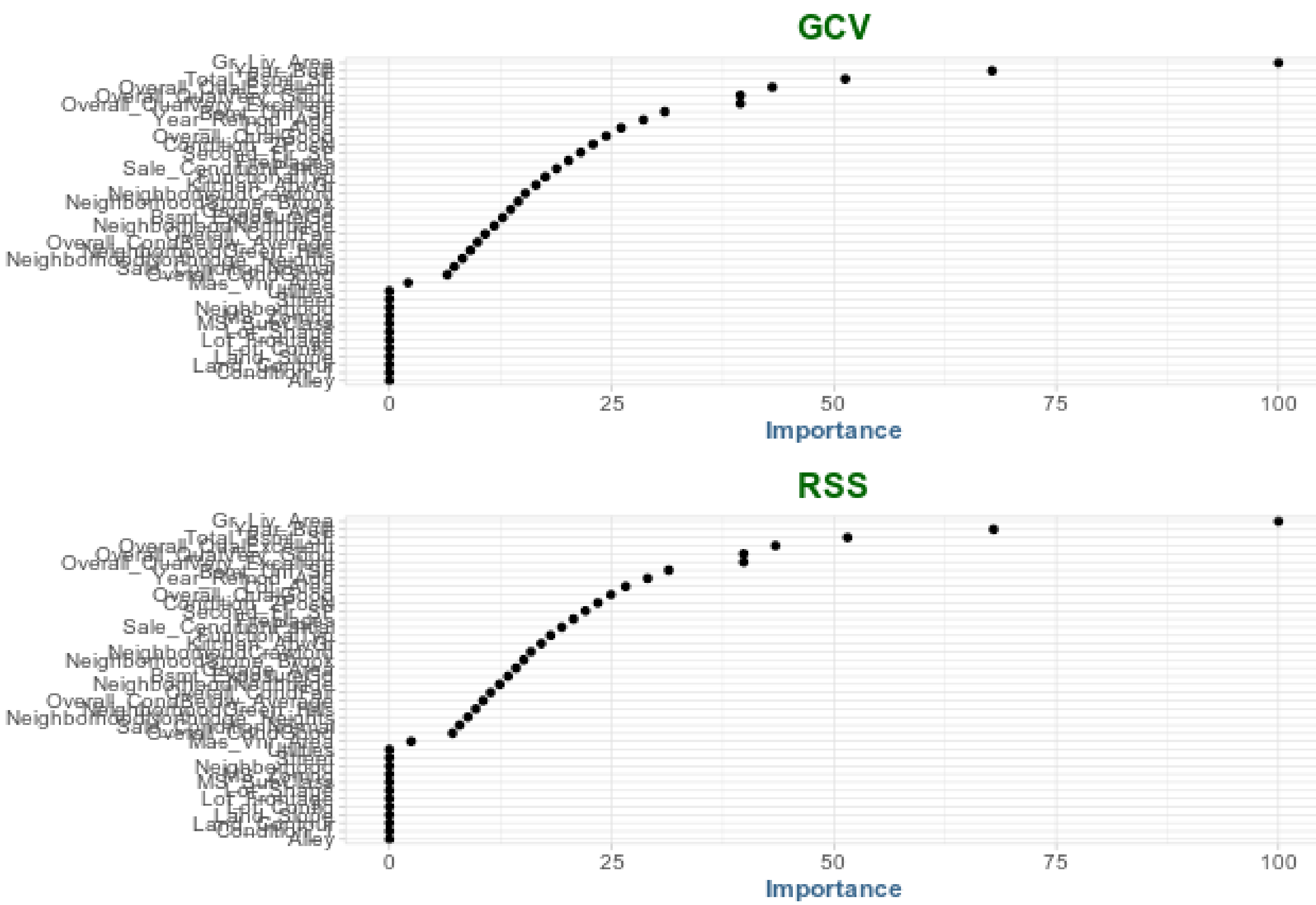
```r
p1 <- vip::vip(cv.mars2, num_features = 40, bar = F, value = "gcv") +
   ggtitle("GCV")
```

```
Warning in vip.default(cv.mars2, num_features = 40, bar = F, value = "gcv"): The
`bar` argument has been deprecated in favor of the new `geom` argument. It will
be removed in version 0.3.0.
```

```r
p2 <- vip::vip(cv.mars2, num_features = 40, bar = F, value = "rss") +
   ggtitle("RSS")
```

```
Warning in vip.default(cv.mars2, num_features = 40, bar = F, value = "rss"): The
`bar` argument has been deprecated in favor of the new `geom` argument. It will
be removed in version 0.3.0.
```

```r
gridExtra::grid.arrange(p1, p2, nrow = 2)
```



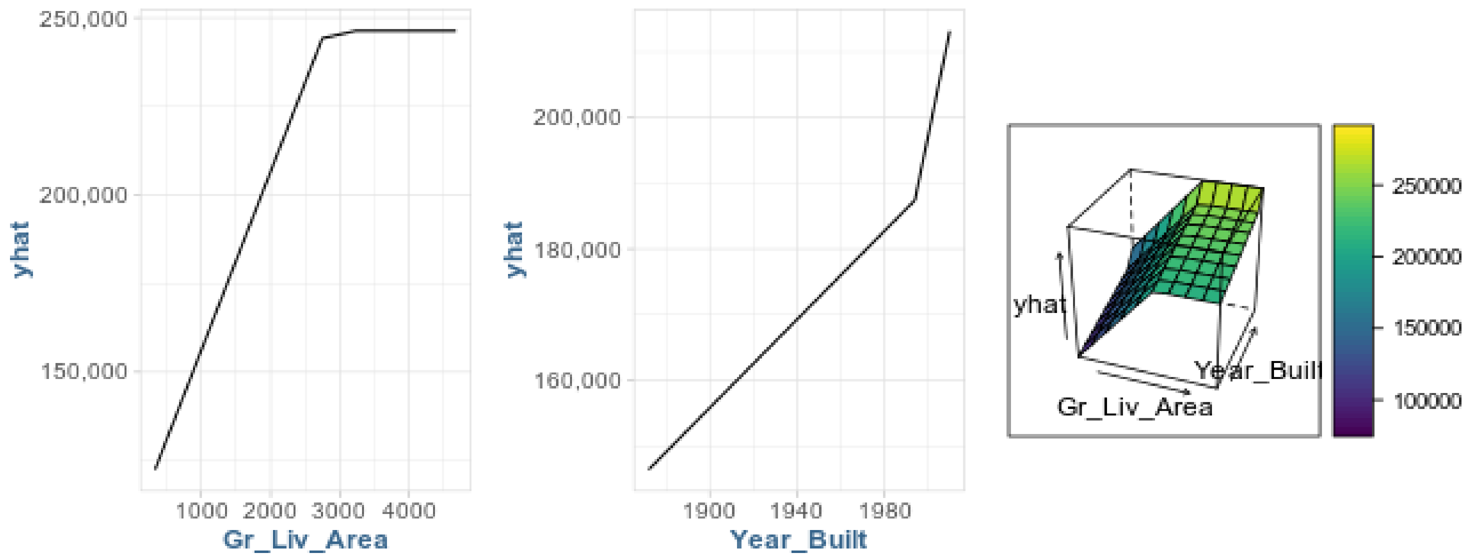

```r
# extract coefficients, covert to tidy & filter for interaction

cv.mars2$finalModel %>%
   coef() %>%
```

```r
  broom::tidy()
```

```
Warning: 'tidy.numeric' is deprecated.
See help("Deprecated")
```

```
# A tibble: 34 x 2
   names                             x
   <chr>                         <dbl>
 1 (Intercept)                 238598.
 2 h(2787-Gr_Liv_Area)            -50.5
 3 h(Year_Built-2004)            3735.
 4 h(2004-Year_Built)            -336.
 5 h(Total_Bsmt_SF-1298)           56.8
 6 h(1298-Total_Bsmt_SF)          -28.5
 7 h(Bsmt_Unf_SF-536)             -24.2
 8 h(536-Bsmt_Unf_SF)              15.4
 9 Overall_QualExcellent        80797.
10 Overall_QualVery_Excellent 119191.
# ... with 24 more rows
```

```r
# Construct partial dependence plots
p1 <- pdp::partial(cv.mars2, pred.var = "Gr_Liv_Area", grid.resolution = 10) %>%
  autoplot() +
   scale_y_continuous(labels = scales::comma)

p2 <- pdp::partial(cv.mars2, pred.var = "Year_Built", grid.resolution = 10) %>%
  autoplot() +
   scale_y_continuous(labels = scales::comma)

p3 <- pdp::partial(cv.mars2, pred.var = c("Gr_Liv_Area", "Year_Built"),
              grid.resolution = 10) %>%
  plotPartial(levelplot = FALSE, zlab = "yhat", drape = TRUE, colorkey = TRUE,
              screen = list(z = -20, x = -60))

# Display plots side by side
gridExtra::grid.arrange(p1, p2, p3, ncol = 3)
```



Attrition data

```
df <- rsample::attrition %>% mutate_if(is.ordered, factor, order = F)

# Create training (70%) and test (30%) sets for the attrition data.
set.seed(123)

churn.split <- initial_split(df, prop = .7, strata = "Attrition")
churn.train <- training(churn.split)
churn.test <- testing(churn.split)

set.seed(123)

suppressWarnings(print({
tuned.mars <- train(
   x = subset(churn.train, select = -Attrition),
   y = churn.train$Attrition,
   method = "earth",
   trControl = trainControl(method = "cv", number = 10),
   tuneGrid = hyper.grid
)}))
```

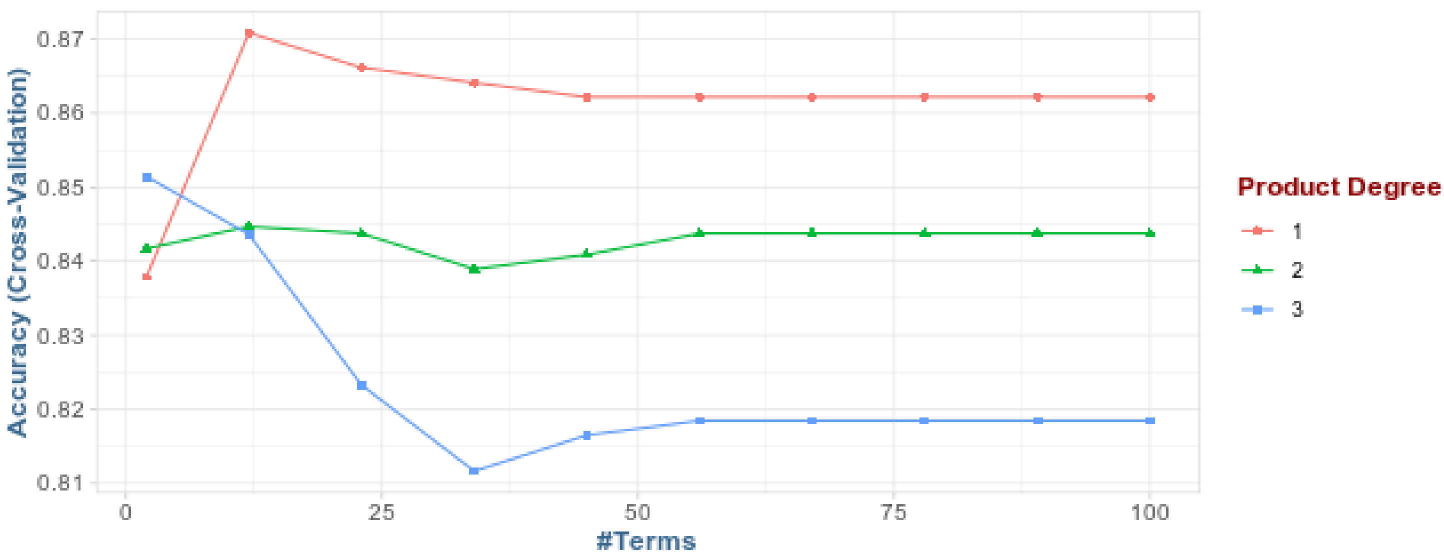
```
Multivariate Adaptive Regression Spline

1030 samples
  30 predictor
   2 classes: 'No', 'Yes'
```

```
No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 926, 926, 927, 928, 928, 927, ...
Resampling results across tuning parameters:

  degree  nprune  Accuracy   Kappa
  1         2     0.8378862  0.005358948
  1        12     0.8708500  0.401114326
  1        23     0.8661373  0.430216328
  1        34     0.8641099  0.428656510
  1        45     0.8621777  0.426509019
  1        56     0.8621777  0.426509019
  1        67     0.8621777  0.426509019
  1        78     0.8621777  0.426509019
  1        89     0.8621777  0.426509019
  1       100     0.8621777  0.426509019
  2         2     0.8417225  0.224053300
  2        12     0.8446920  0.299911342
  2        23     0.8437495  0.332054852
  2        34     0.8389424  0.325019482
  2        45     0.8408937  0.342780742
  2        56     0.8437503  0.359336395
  2        67     0.8437503  0.359336395
  2        78     0.8437503  0.359336395
  2        89     0.8437503  0.359336395
  2       100     0.8437503  0.359336395
  3         2     0.8514312  0.266403262
  3        12     0.8436073  0.280383943
  3        23     0.8232658  0.251074784
  3        34     0.8116433  0.231032715
  3        45     0.8165072  0.237656881
  3        56     0.8184490  0.250682779
  3        67     0.8184490  0.250682779
  3        78     0.8184490  0.250682779
  3        89     0.8184490  0.250682779
  3       100     0.8184490  0.250682779

Accuracy was used to select the optimal model using the largest value.
The final values used for the model were nprune = 12 and degree = 1.
```

```r
tuned.mars$bestTune
```

```
  nprune degree
2     12      1
```

```r
ggplot(tuned.mars)
```




```r
# train logistic regression model
set.seed(123)

glm.mod <- train(
  Attrition ~ .,
  data = churn.train,
  method = "glm",
  family = "binomial",
  preProc = c("zv", "center", "scale"),
  trControl = trainControl(method = "cv", number = 10)
)

# train regularized logistic regression model
set.seed(123)

penalized.mod <- train(
  Attrition ~ .,
  data = churn.train,
  method = "glmnet",
  family = "binomial",
  preProc = c("zv", "center", "scale"),
  trControl = trainControl(method = "cv", number = 10),
  tuneLength = 10
)

# extract out of sample performance measures
summary(resamples(list(
```

Table 1: Cross-validated accuracy results for tuned MARS and regression models.

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	NA's
Logistic_model	0.8365385	0.8495146	0.8792476	0.8757893	0.8907767	0.9313725	0
Elastic_net	0.8446602	0.8759280	0.8834951	0.8835759	0.8915469	0.9411765	0
MARS_model	0.8155340	0.8578463	0.8780697	0.8708500	0.8907767	0.9029126	0

```r
  Logistic_model = glm.mod,
  Elastic_net = penalized.mod,
  MARS_model = tuned.mars
  )))$statistics$Accuracy %>%
  kableExtra::kable(caption = "Cross-validated accuracy results for tuned MARS and regression m
  kableExtra::kable_styling(bootstrap_options = c("striped", "hover"))
```

```r
# clean up
rm(list = ls())
```