Linear regression in R

May 2, 2016

Contents

Linear regression

In this tutorial we'll learn:

1. how to fit linear regression models
2. how to split data into test and train sets
3. how to tune our models and select features

Data preparation

As always when you open R, start by reloading the packages you expect to use. In the code below, we're loading the `dplyr` and `ggplot2` libraries. We're working with the Capital Bikeshare again, so start by reading in the data file.

```r
library(dplyr)
library(ggplot2)

# you might want to set the working directory first.
setwd('/path/to/your/data')

bikeshare = read.delim('bikeshare_2015.tsv',
                       sep = '\t',
                       header = TRUE)
```

The `lm()` function

The function for creating a linear model in R is `lm()` and the primary arguments are *formula* and *data*. Formulas in R are expressed with a tilde, *e.g.*`y ~ x`. Let's fit the model: $rentals = \beta_0 + \beta_1 * crossing$.

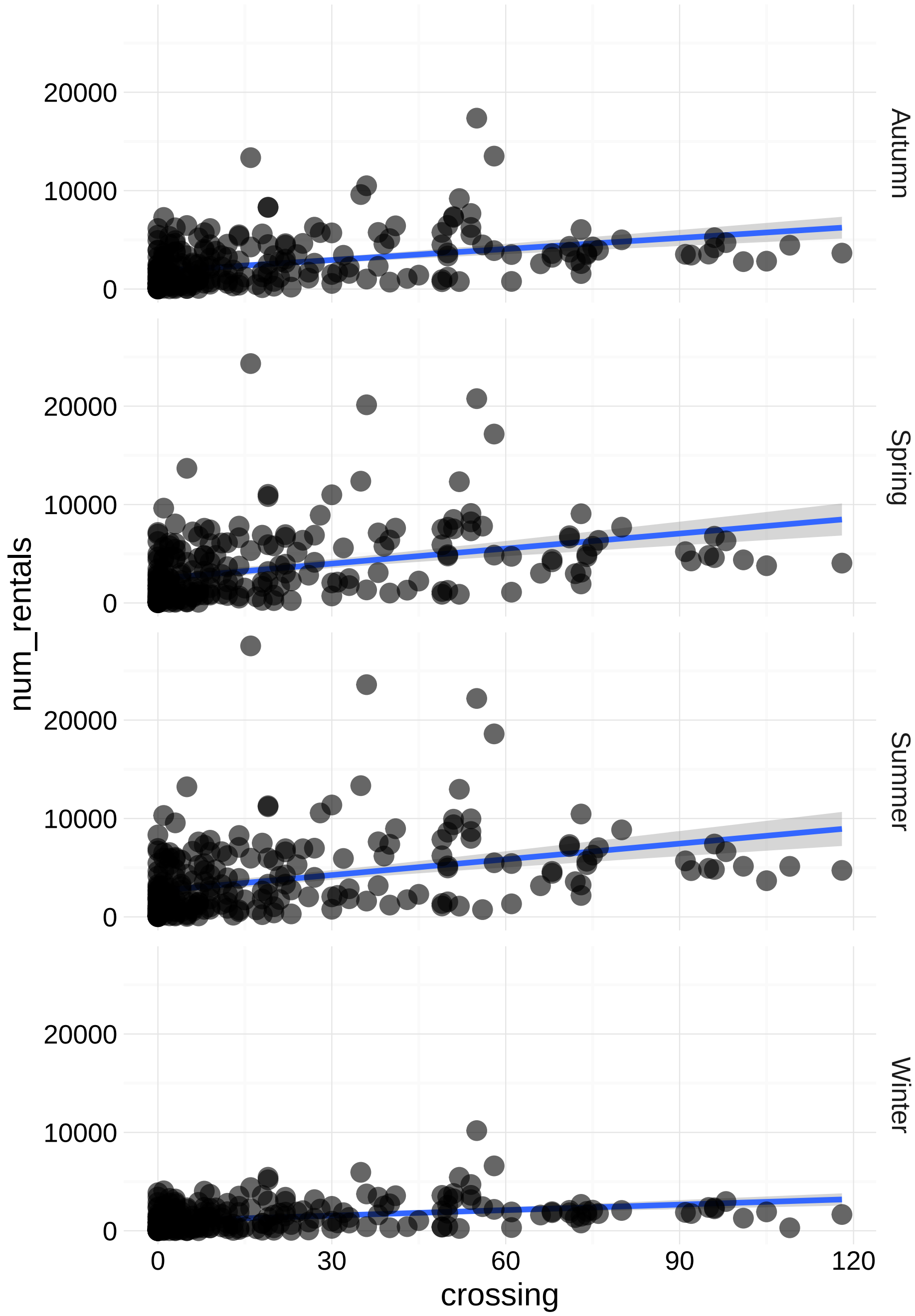
```r
model = lm(num_rentals ~ crossing, data = bikeshare)

# view what is returned in the lm object
attributes(model)
```

```
## $names
##  [1] "coefficients"  "residuals"     "effects"       "rank"
##  [5] "fitted.values" "assign"        "qr"            "df.residual"
##  [9] "xlevels"       "call"          "terms"         "model"
##
## $class
## [1] "lm"
```

```r
# get model output
summary(model)
```

```
##
## Call:
## lm(formula = num_rentals ~ crossing, data = bikeshare)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6020.3 -1826.9  -947.5   930.2 24912.3
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1986.512    116.721   17.02   <2e-16 ***
## crossing      39.824      3.536   11.26   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2968 on 999 degrees of freedom
## Multiple R-squared:  0.1127, Adjusted R-squared:  0.1118
## F-statistic: 126.8 on 1 and 999 DF,  p-value: < 2.2e-16
```

```r
# plot it
ggplot(bikeshare, aes(x = crossing, y = num_rentals)) +
  geom_smooth(method = 'lm') +
  geom_point(size = 3, alpha = 0.60) +
  facet_grid(season ~. ) +
  theme_minimal()
```



The `attributes()` function can be called on just about any object in R and it returns a list of all the things inside. It's a great way to explore objects and see what values are contained inside that could be used in other analysis. For example, extracting the residuals via `model$residuals` is useful if we want to print diagnostic plots like those above.

Run `summary()` on the `lm` object to see detailed results. The *Call* section prints the model specification, and the *Residuals* section contains a summary of the distribution of the errors. *Coefficients* section contains the estimated coefficients, standard errors, *t*- and *p*-values for each variable in the model. Our model ends up being `rentals = 1987 + 40*(crossings)`, which means that the average number of rentals is 1987 when there are no crosswalks, and the average increases by 40 rentals for every additional crosswalk within a quarter mile.

Multivariate regression

We can fit regressions with multiple covariates the same way by adding variables with the `+` sign. Let's fit another model that includes the number of crosswalks and the number of parking lots as predictors:

```r
model = lm(num_rentals ~ crossing + parking, data = bikeshare)
summary(model)
```

```
##
## Call:
## lm(formula = num_rentals ~ crossing + parking, data = bikeshare)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6018.3 -1827.1  -952.2   929.3 24902.8
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1995.254    122.161  16.333   <2e-16 ***
## crossing      39.874      3.544  11.252   <2e-16 ***
## parking      -16.191     66.431  -0.244    0.807
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2969 on 998 degrees of freedom
## Multiple R-squared:  0.1127, Adjusted R-squared:  0.1109
## F-statistic: 63.39 on 2 and 998 DF,  p-value: < 2.2e-16
```

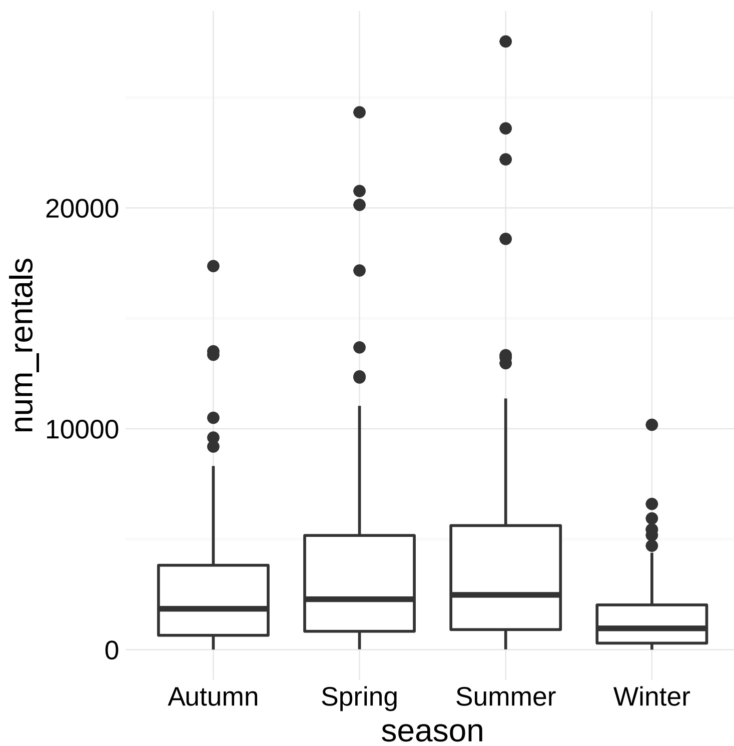
Let's try one more, this time we'll include `season`, a factor variable:

```r
# lets include season this time
model = lm(num_rentals ~ crossing + parking + season, data = bikeshare)
summary(model)
```

```
##
## Call:
## lm(formula = num_rentals ~ crossing + parking + season, data = bikeshare)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4609.5 -1783.6  -552.4  1061.3 23969.6
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1789.212    193.430   9.250  < 2e-16 ***
```

```
## crossing        39.926      3.372  11.840  < 2e-16 ***
## parking        -14.795     63.217  -0.234 0.815011
## seasonSpring   905.922    252.969   3.581 0.000359 ***
## seasonSummer  1138.356    252.457   4.509 7.28e-06 ***
## seasonWinter -1209.862    251.954  -4.802 1.81e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2825 on 995 degrees of freedom
## Multiple R-squared:  0.1989, Adjusted R-squared:  0.1949
## F-statistic: 49.41 on 5 and 995 DF,  p-value: < 2.2e-16
```

```r
ggplot(bikeshare, aes(x = season, y = num_rentals)) +
  geom_boxplot() +
  theme_minimal()
```



You might have noticed that one of the seasons is missing. By default R chooses a reference group that is represented in the intercept term. In this case `Autumn` is the reference group for the `season` variable because by default R orders factors alphabetically. If you want to order the `season` category differently, just specify the `levels` in the `factor` function.

```r
bikeshare$season = factor(bikeshare$season,
                          levels = c('Spring', 'Summer', 'Autumn', 'Winter'))

model = lm(num_rentals ~ crossing + season, data = bikeshare)
summary(model)
```

```
##
```

```
## Call:
## lm(formula = num_rentals ~ crossing + season, data = bikeshare)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4611.3 -1777.4  -548.2  1050.6 23978.3
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2687.188    190.841  14.081  < 2e-16 ***
## crossing        39.881      3.365  11.852  < 2e-16 ***
## seasonSummer   232.461    253.099   0.918 0.358602
## seasonAutumn  -906.036    252.848  -3.583 0.000356 ***
## seasonWinter -2115.866    252.598  -8.376  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2824 on 996 degrees of freedom
## Multiple R-squared:  0.1989, Adjusted R-squared:  0.1957
## F-statistic: 61.81 on 4 and 996 DF,  p-value: < 2.2e-16
```

The interpretation of the new model is that stations without crosswalks or parking have an average of 2,687 rentals in the spring. Those same stations can expect an additional 232 rentals in the summer, and about 906 less in the fall and 2,116 less in the winter.

The *caret* package

Remember last time when we did exploratory data analysis (EDA) with `ggpairs` because it allowed us to view relationships between many variable simultaneously? We're in a similar situation now because there are around 70 modeling variables to choose from, so how do we start developing models?

Lucky for us there's the `caret` package (short for **c**lassification **a**nd **re**gression **t**raining). `caret` is great for model development because it integrates many modeling methods in R into one unified syntax. That means more reusable code for us! *caret* contains helper functions that provide a unified framework for data cleaning/splitting, model training, and comparison. I highly recommend the optional reading this week which provides a great overview of the *caret* package.

```r
install.packages('caret', dependencies = TRUE)
library(caret)

set.seed(1234) # set a seed
```

Setting a seed in R insures that you get identical results each time you run your code. Since re-sampling methods are inherently probabilistic, every time we rerun them we'll get slightly different answers. Setting the seed to the same number insures that we get identical randomness each time the code is run, and that's helpful for debugging.

Train and test data

Before analysis we'll divide data into train and test sets. Check out this nice overview for more details. The *training* set is typically about 75% of the data and is used for all the model development. Once we have a model we're satisfied with, we use our *testing* set, the other 25% to generate model predictions. Splitting the data into the two groups, train and test, generates two types of errors, in-sample and out-of-sample errors. *In-sample* errors are the errors derived from same data the model was built with. *Out-of-sample* errors are derived from measuring the error on a fresh data set. We are interested in the out-of-sample error because this quantity represents how'd we'd expect the model to perform in the future on brand new data.

Here's how to split the data with *caret*:

```
# select the training observations
in_train = createDataPartition(y = bikeshare$num_rentals,
                               p = 0.75, # 75% in train, 25% in test
                               list = FALSE)
head(in_train) # row indices of observations in the training set
```

```
##      Resample1
## [1,]         1
## [2,]         2
## [3,]         3
## [4,]         4
## [5,]         6
## [6,]         7
```

```
train = bikeshare[in_train, ]
test = bikeshare[-in_train, ]

dim(train)
```

```
## [1] 753  76
```

```
dim(test)
```

```
## [1] 248  76
```

Note: I recommend doing all data processing and aggregation steps *before* splitting out your train/test sets.

Model training

Our workhorse function in the *caret* package in the `train` function. This function can be used to evaluate performance parameters, choose optimal models based on the values of those parameters, and estimate model performance. For regression we can use it in place of the `lm()` function. Here's our last regression model using the train function.

Now that you're familiar with how to specify model equations with the ~ character you should recognize the model:

```
model_fit = train(num_rentals ~ crossing + parking + season,
                  data = train,
                  method = 'lm',
                  metric = 'RMSE')
print(model_fit)
```

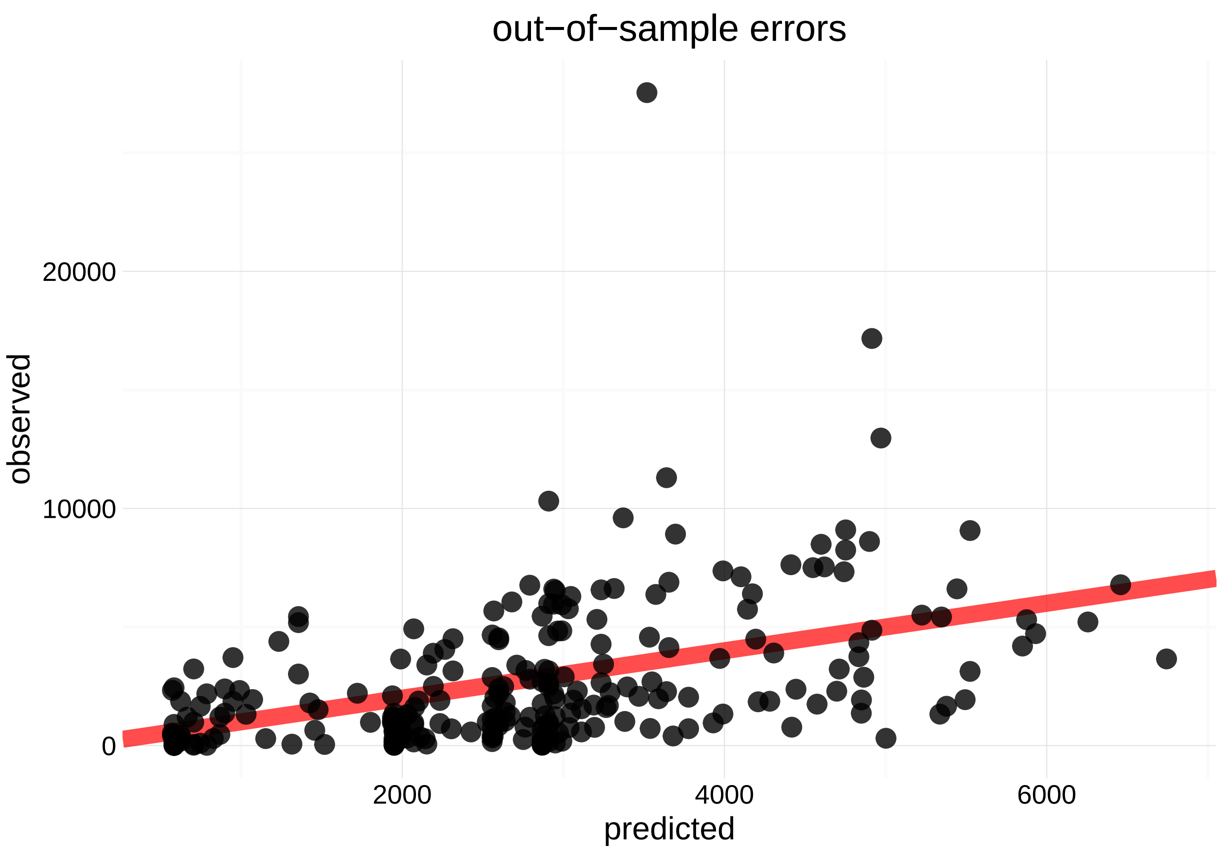
```
## Linear Regression
##
## 753 samples
##  75 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 753, 753, 753, 753, 753, 753, ...
## Resampling results
##
##   RMSE      Rsquared   RMSE SD  Rsquared SD
##   2812.186  0.1877356  203.556  0.03102951
##
##
```

```r
summary(model_fit)
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4373.3 -1863.5  -563.1  1022.1 21120.4
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2558.524    221.798  11.535  < 2e-16 ***
## crossing        40.630      3.968  10.238  < 2e-16 ***
## parking        -10.216     69.366  -0.147   0.8830
## seasonSummer   310.146    290.576   1.067   0.2862
## seasonAutumn  -608.945    294.571  -2.067   0.0391 *
## seasonWinter -1974.447    294.973  -6.694 4.27e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2851 on 747 degrees of freedom
## Multiple R-squared:  0.1939, Adjusted R-squared:  0.1886
## F-statistic: 35.95 on 5 and 747 DF,  p-value: < 2.2e-16
```

```r
# get predictions
out_of_sample_predictions = predict(model_fit, newdata = test)

# compare predictions against the observed values
errors = data.frame(predicted = out_of_sample_predictions,
                    observed = test$num_rentals,
                    error = out_of_sample_predictions - test$num_rentals)

# plot the out-of-sample errors
ggplot(data = errors, aes(x = predicted, y = observed)) +
  geom_abline(aes(intercept = 0, slope = 1),
              size = 3, alpha = 0.70, color = 'red') +
  geom_point(size = 3, alpha = 0.80) +
```

```r
  ggtitle('out-of-sample errors') +
  theme_minimal()
```



Our prediction accuracy is not so great for this model. The in-sample RMSE is about 2,863 which means that on average the predictions are off by about 2,863 rentals.

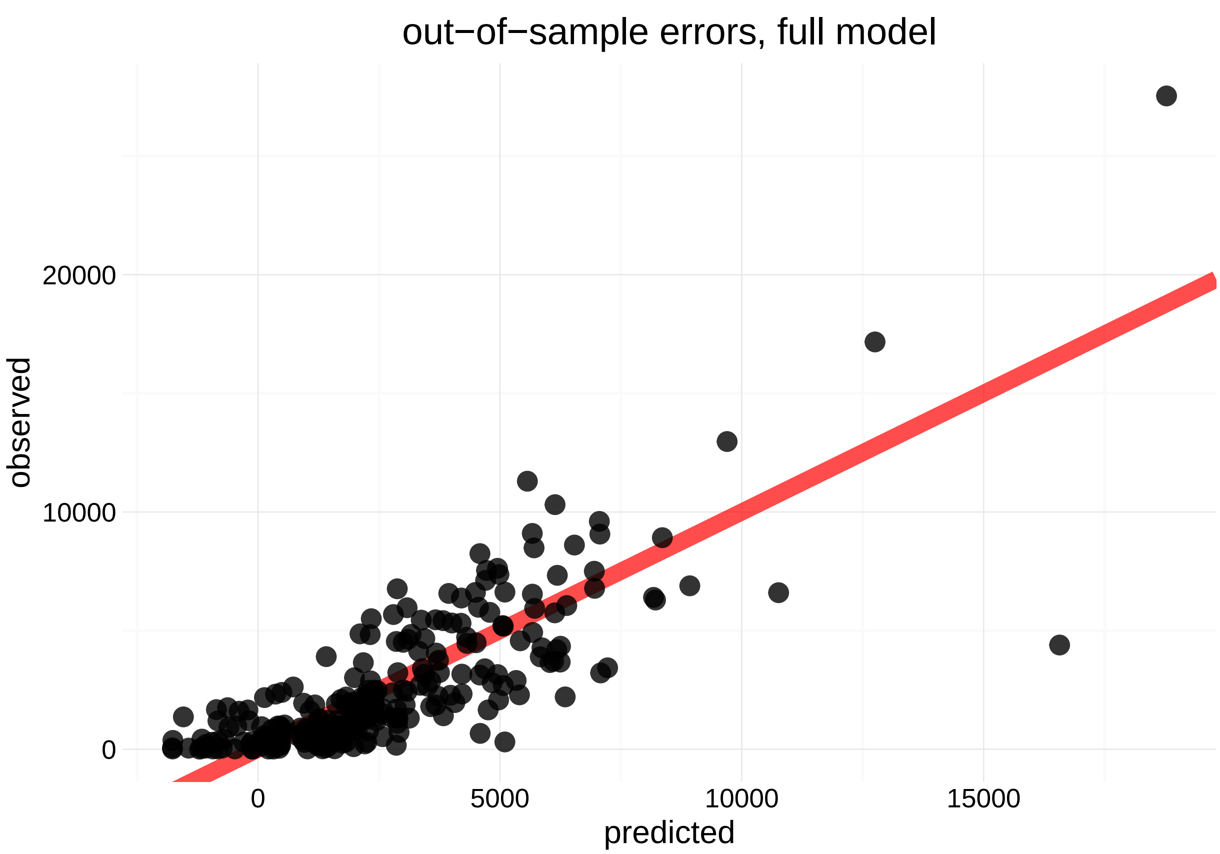
What happens if we build a model with all the variables in it? To do that, I'm using the `select` verb from `dplyr` to remove variables that aren't predictors, like the station name, id, and lat/long.

```r
full_model = train(num_rentals ~ .,
                   data = select(train, -station, -id, -lat, -long),
                   method = 'lm')
full_model
```

```
## Linear Regression
##
## 753 samples
##  71 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 753, 753, 753, 753, 753, 753, ...
## Resampling results
##
##   RMSE      Rsquared   RMSE SD   Rsquared SD
##   2130.272  0.5439089  227.4531  0.05816546
```

```
##
##
```

```r
# get predictions
out_of_sample_predictions = predict(full_model, newdata = test)

# compare predictions against the observed values
errors = data.frame(predicted = out_of_sample_predictions,
                    observed = test$num_rentals,
                    error = out_of_sample_predictions - test$num_rentals)

# plot the out-of-sample errors
ggplot(data = errors, aes(x = predicted, y = observed)) +
  geom_abline(aes(intercept = 0, slope = 1),
              size = 3, alpha = 0.70, color = 'red') +
  geom_point(size = 3, alpha = 0.80) +
  ggtitle('out-of-sample errors, full model') +
  theme_minimal()
```



The in-sample RMSE is about 2,281, so definitely an improvement over the previous model, but this model is really complex and probably not going to be usable by Pronto. How can we reduce the complexity of the model, but maintain reasonable predictive accuracy?

Assignment 3

1. Try a couple different models based on the hypotheses you tested in the first two assignments. Can you improve on the RMSE?

Preprocessing

Shrinkage methods require that the predictors are normalized to be on the same scale. We can accomplish this by centering and scaling the data. You center a variable by subtracting the mean of the variable from from each observation. To scale your observations you then divide the centered observation by the variable standard deviation. Now the variable follows a standard normal distribution with mean = 0 and standard deviation = 1.

The *caret* package has lots of convenient functions for preprocessing data, check 'em out!

```
full_model_scaled = train(num_rentals ~ .,
                          data = select(train, -station, -id, -lat, -long),
                          method = 'lm',
                          preProcess = c('center', 'scale'))
```

Coefficients estimated with normalized data have a different interpretation than coefficients from un-normalized data. In this case when the data are scaled the intercept has a better interpretation, it's the expected number of rentals when all the predictors are at their average value. So, in this case, when all the predictors are at their average values, we expect about 2813 rentals per season. In the previous full-model we had an intercept of about -878.261, which could be interpreted as the expected number of rentals when all the other predictors have a value of 0. That's pretty unsatisfying for a couple reasons. First, we can't have negative rentals! Second, for a lot of the predictors it doesn't make sense to plug in 0's. What does it mean to have a duration of 0? Centering and scaling fix the non-interpret ability of the previous models.

Since we divide by the standard deviation during scaling, the non-intercept coefficients in the centered and scaled model can be interpreted as the increase in y associated with a 1 standard deviation increase in x.

Model Selection

Variable combination

A simple method to reduce model complexity is to combine some of the variables. For example the data set contains a variable for *nightclub*, *pub* and *bar*, likewise there's a variable for *cafe*, *restaurant* and *fast_food*. Maybe we can retain information and remove some variables.

```
bikeshare$food = bikeshare$fast_food + bikeshare$restaurant + bikeshare$cafe

bikeshare$nightlife = bikeshare$bar + bikeshare$pub + bikeshare$nightclub

bikeshare$tourism =
  bikeshare$tourism_artwork +
  bikeshare$tourism_hotel +
  bikeshare$tourism_information +
  bikeshare$tourism_museum

# save new modeling dataset in new variable
```

```r
to_model =
  bikeshare %>%
    select(-station, -id, -lat, -long, -fast_food, -restaurant, -cafe, -bar,
           -pub, -nightclub, -tourism_artwork, -tourism_hotel,
           -tourism_information, -tourism_museum)
```

Try out your own categories, these are just a few to get you started. We'll learn how to make categories computationally when we cover clustering.

We've change the data frame, don't forget to redefine the train and test sets!

```r
train = bikeshare[in_train, ]
test = bikeshare[-in_train, ]

train = select(train, -station, -id, -lat, -long)
test = select(test, -station, -id, -lat, -long)

dim(train)
```

```
## [1] 753  75
```

```r
dim(test)
```

```
## [1] 248  75
```

```r
# how does our new full-model compare?
full_model = train(num_rentals ~ .,
                   data = train,
                   method = 'lm')
```

Subset selection

We haven't talked much about computational limitations yet, but it's a good time to start. Selection methods can be *extremely* slow. Why? Because we have $2^p = 2^{117}$ possible variable combinations. I recommend doing some combining before trying these methods. I'll leave the combining up to you, but to make sure these models can run in less than infinite time, I'm going to remove a bunch of predictors so you get the idea.

```r
# forward selection
forward_model = train(num_rentals ~ .,
                      data = train,
                      method = 'leapForward',
                      preProcess = c('center', 'scale'),
                      # try models of size 1 - 23
                      tuneGrid = expand.grid(nvmax = 1:23),
                      trControl = trainControl(method = 'cv', number = 5))
```

```
## Loading required package: leaps
```

```r
# what does this return?
attributes(forward_model)
```

```
## $names
##  [1] "method"       "modelInfo"    "modelType"    "results"
##  [5] "pred"         "bestTune"     "call"         "dots"
##  [9] "metric"       "control"      "finalModel"   "preProcess"
## [13] "trainingData" "resample"     "resampledCM"  "perfNames"
## [17] "maximize"     "yLimits"      "times"        "terms"
## [21] "coefnames"    "contrasts"    "xlevels"
##
## $class
## [1] "train"         "train.formula"
```

```
# what what should the number of variables, k, be?
forward_model$bestTune
```

```
##    nvmax
## 18    18
```

```
# what metric was used?
forward_model$metric
```

```
## [1] "RMSE"
```

```
# here's a handful of other useful plots and summaries
print(forward_model)
```

```
## Linear Regression with Forward Selection
##
## 753 samples
##  74 predictor
##
## Pre-processing: centered (76), scaled (76)
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 602, 602, 602, 604, 602
## Resampling results across tuning parameters:
##
##   nvmax  RMSE      Rsquared   RMSE SD   Rsquared SD
##    1     2702.307  0.2694288  484.3030  0.08518798
##    2     2579.851  0.3407042  469.3137  0.09615947
##    3     2522.512  0.3738656  515.2484  0.10931030
##    4     2371.326  0.4465397  472.3477  0.09849612
##    5     2317.669  0.4754220  456.3210  0.09188041
##    6     2230.125  0.5136850  462.5341  0.09392882
##    7     2232.760  0.5101060  467.8282  0.10111356
##    8     2237.314  0.5123174  474.4217  0.09846137
##    9     2185.135  0.5346181  502.8303  0.10808996
##   10     2166.635  0.5431605  504.9955  0.10019522
##   11     2129.376  0.5574488  516.8566  0.10516405
##   12     2117.728  0.5633520  494.5376  0.09705770
##   13     2108.599  0.5692042  510.8521  0.10211826
##   14     2102.275  0.5743338  511.7891  0.10697707
##   15     2092.176  0.5764592  513.1436  0.10650973
##   16     2089.927  0.5766539  527.6233  0.10911190
```

```
##   17     2094.813  0.5731897  507.4915  0.10326790
##   18     2083.553  0.5775245  456.5975  0.08241077
##   19     2092.014  0.5735088  452.1491  0.07799192
##   20     2091.838  0.5730770  449.0502  0.07436358
##   21     2098.275  0.5696468  458.8183  0.07363157
##   22     2086.126  0.5750714  458.0903  0.07302218
##   23     2089.007  0.5746407  452.6508  0.07275863
##
## RMSE was used to select the optimal model using  the smallest value.
## The final value used for the model was nvmax = 18.
```

```r
summary(forward_model)
```

```
## Subset selection object
## 76 Variables  (and intercept)
##                          Forced in Forced out
## seasonSummer                 FALSE      FALSE
## seasonAutumn                 FALSE      FALSE
## seasonWinter                 FALSE      FALSE
## duration_mean                FALSE      FALSE
## no_bikes                     FALSE      FALSE
## no_empty_docks               FALSE      FALSE
## atm                          FALSE      FALSE
## bank                         FALSE      FALSE
## bar                          FALSE      FALSE
## bench                        FALSE      FALSE
## bicycle_parking              FALSE      FALSE
## bicycle_rental               FALSE      FALSE
## cafe                         FALSE      FALSE
## doctors                      FALSE      FALSE
## drinking_water               FALSE      FALSE
## embassy                      FALSE      FALSE
## fast_food                    FALSE      FALSE
## fountain                     FALSE      FALSE
## fuel                         FALSE      FALSE
## kindergarten                 FALSE      FALSE
## nightclub                    FALSE      FALSE
## parking                      FALSE      FALSE
## parking_entrance             FALSE      FALSE
## pharmacy                     FALSE      FALSE
## place_of_worship             FALSE      FALSE
## police                       FALSE      FALSE
## post_box                     FALSE      FALSE
## post_office                  FALSE      FALSE
## pub                          FALSE      FALSE
## recycling                    FALSE      FALSE
## restaurant                   FALSE      FALSE
## school                       FALSE      FALSE
## theatre                      FALSE      FALSE
## waste_basket                 FALSE      FALSE
## artwork_type_statue          FALSE      FALSE
## diplomatic_embassy           FALSE      FALSE
## emergency_fire_hydrant       FALSE      FALSE
## bus_stop                     FALSE      FALSE
```

```
## crossing                     FALSE      FALSE
## motorway_junction            FALSE      FALSE
## stop                         FALSE      FALSE
## traffic_signals              FALSE      FALSE
## turning_circle               FALSE      FALSE
## historic_memorial            FALSE      FALSE
## historic_monument            FALSE      FALSE
## leisure_park                 FALSE      FALSE
## leisure_sports_centre        FALSE      FALSE
## office_government            FALSE      FALSE
## outdoor_seating              FALSE      FALSE
## parking_underground          FALSE      FALSE
## railway_level_crossing       FALSE      FALSE
## railway_station              FALSE      FALSE
## railway_subway_entrance      FALSE      FALSE
## shop_alcohol                 FALSE      FALSE
## shop_art                     FALSE      FALSE
## shop_bakery                  FALSE      FALSE
## shop_beauty                  FALSE      FALSE
## shop_books                   FALSE      FALSE
## shop_clothes                 FALSE      FALSE
## shop_convenience             FALSE      FALSE
## shop_dry_cleaning            FALSE      FALSE
## shop_electronics             FALSE      FALSE
## shop_gift                    FALSE      FALSE
## shop_hairdresser             FALSE      FALSE
## shop_mobile_phone            FALSE      FALSE
## shop_shoes                   FALSE      FALSE
## shop_stationery              FALSE      FALSE
## shop_supermarket             FALSE      FALSE
## station_subway               FALSE      FALSE
## tourism_artwork              FALSE      FALSE
## tourism_hotel                FALSE      FALSE
## tourism_information          FALSE      FALSE
## tourism_museum               FALSE      FALSE
## food                         FALSE      FALSE
## nightlife                    FALSE      FALSE
## tourism                      FALSE      FALSE
## 1 subsets of each size up to 18
## Selection Algorithm: forward
##           seasonSummer seasonAutumn seasonWinter duration_mean no_bikes
## 1  ( 1 )  " "          " "          " "          " "           " "
## 2  ( 1 )  " "          " "          "*"          " "           " "
## 3  ( 1 )  " "          " "          "*"          " "           " "
## 4  ( 1 )  " "          " "          "*"          " "           "*"
## 5  ( 1 )  " "          " "          "*"          " "           "*"
## 6  ( 1 )  " "          " "          "*"          " "           "*"
## 7  ( 1 )  " "          " "          "*"          " "           "*"
## 8  ( 1 )  " "          " "          "*"          " "           "*"
## 9  ( 1 )  " "          " "          "*"          " "           "*"
## 10  ( 1 ) " "          "*"          "*"          " "           "*"
## 11  ( 1 ) " "          "*"          "*"          " "           "*"
## 12  ( 1 ) " "          "*"          "*"          " "           "*"
## 13  ( 1 ) " "          "*"          "*"          " "           "*"
```

```
## 14  ( 1 ) " "          "*"          "*"          " "           "*"
## 15  ( 1 ) " "          "*"          "*"          " "           "*"
## 16  ( 1 ) " "          "*"          "*"          " "           "*"
## 17  ( 1 ) " "          "*"          "*"          " "           "*"
## 18  ( 1 ) " "          "*"          "*"          " "           "*"
##           no_empty_docks atm bank bar bench bicycle_parking bicycle_rental
## 1  ( 1 )  " "            " " " "  " " " "   " "             " "
## 2  ( 1 )  " "            " " " "  " " " "   " "             " "
## 3  ( 1 )  "*"            " " " "  " " " "   " "             " "
## 4  ( 1 )  "*"            " " " "  " " " "   " "             " "
## 5  ( 1 )  "*"            " " " "  " " " "   " "             " "
## 6  ( 1 )  "*"            " " " "  " " " "   " "             " "
## 7  ( 1 )  "*"            " " " "  " " " "   " "             " "
## 8  ( 1 )  "*"            " " " "  " " " "   " "             " "
## 9  ( 1 )  "*"            " " " "  "*" " "   " "             " "
## 10  ( 1 ) "*"            " " " "  "*" " "   " "             " "
## 11  ( 1 ) "*"            " " " "  "*" " "   " "             " "
## 12  ( 1 ) "*"            " " " "  "*" " "   " "             " "
## 13  ( 1 ) "*"            " " " "  "*" " "   " "             " "
## 14  ( 1 ) "*"            " " " "  "*" " "   " "             " "
## 15  ( 1 ) "*"            " " " "  "*" "*"   " "             " "
## 16  ( 1 ) "*"            " " " "  "*" "*"   " "             " "
## 17  ( 1 ) "*"            " " " "  "*" "*"   " "             " "
## 18  ( 1 ) "*"            " " " "  "*" "*"   " "             " "
##           cafe doctors drinking_water embassy fast_food fountain fuel
## 1  ( 1 )  " "  " "     " "            " "     " "       " "      " "
## 2  ( 1 )  " "  " "     " "            " "     " "       " "      " "
## 3  ( 1 )  " "  " "     " "            " "     " "       " "      " "
## 4  ( 1 )  " "  " "     " "            " "     " "       " "      " "
## 5  ( 1 )  " "  " "     " "            " "     " "       " "      " "
## 6  ( 1 )  " "  " "     " "            " "     " "       " "      " "
## 7  ( 1 )  " "  " "     " "            " "     " "       " "      " "
## 8  ( 1 )  " "  " "     " "            " "     " "       " "      " "
## 9  ( 1 )  " "  " "     " "            " "     " "       " "      " "
## 10  ( 1 ) " "  " "     " "            " "     " "       " "      " "
## 11  ( 1 ) " "  " "     " "            " "     "*"       " "      " "
## 12  ( 1 ) " "  " "     " "            " "     "*"       " "      " "
## 13  ( 1 ) " "  " "     " "            " "     "*"       " "      " "
## 14  ( 1 ) " "  " "     " "            " "     "*"       " "      " "
## 15  ( 1 ) " "  " "     " "            " "     "*"       " "      " "
## 16  ( 1 ) " "  " "     " "            " "     "*"       " "      " "
## 17  ( 1 ) " "  " "     " "            " "     "*"       " "      " "
## 18  ( 1 ) " "  " "     " "            " "     "*"       " "      " "
##           kindergarten nightclub parking parking_entrance pharmacy
## 1  ( 1 )  " "          " "       " "     " "              " "
## 2  ( 1 )  " "          " "       " "     " "              " "
## 3  ( 1 )  " "          " "       " "     " "              " "
## 4  ( 1 )  " "          " "       " "     " "              " "
## 5  ( 1 )  " "          " "       " "     " "              " "
## 6  ( 1 )  " "          " "       " "     " "              " "
## 7  ( 1 )  " "          " "       " "     " "              " "
## 8  ( 1 )  " "          " "       " "     " "              " "
## 9  ( 1 )  " "          " "       " "     " "              " "
## 10  ( 1 ) " "          " "       " "     " "              " "
```

```
## 11  ( 1 ) " "          " "       " "     " "               " "
## 12  ( 1 ) " "          " "       " "     " "               " "
## 13  ( 1 ) " "          " "       " "     " "               " "
## 14  ( 1 ) " "          " "       "*"     " "               " "
## 15  ( 1 ) " "          " "       "*"     " "               " "
## 16  ( 1 ) " "          " "       "*"     " "               " "
## 17  ( 1 ) " "          " "       "*"     " "               " "
## 18  ( 1 ) " "          " "       "*"     " "               " "
##           place_of_worship police post_box post_office pub recycling
## 1  ( 1 )  " "              " "    " "      " "         " " " "
## 2  ( 1 )  " "              " "    " "      " "         " " " "
## 3  ( 1 )  " "              " "    " "      " "         " " " "
## 4  ( 1 )  " "              " "    " "      " "         " " " "
## 5  ( 1 )  " "              " "    " "      " "         " " " "
## 6  ( 1 )  " "              " "    " "      " "         " " " "
## 7  ( 1 )  " "              " "    " "      " "         " " " "
## 8  ( 1 )  " "              " "    " "      " "         " " " "
## 9  ( 1 )  " "              " "    " "      " "         " " " "
## 10  ( 1 ) " "              " "    " "      " "         " " " "
## 11  ( 1 ) " "              " "    " "      " "         " " " "
## 12  ( 1 ) " "              " "    " "      " "         " " " "
## 13  ( 1 ) " "              " "    " "      " "         " " " "
## 14  ( 1 ) " "              " "    " "      " "         " " " "
## 15  ( 1 ) " "              " "    " "      " "         " " " "
## 16  ( 1 ) " "              " "    " "      " "         " " " "
## 17  ( 1 ) " "              " "    " "      " "         " " " "
## 18  ( 1 ) " "              " "    " "      " "         " " " "
##           restaurant school theatre waste_basket artwork_type_statue
## 1  ( 1 )  " "        " "    " "     " "          " "
## 2  ( 1 )  " "        " "    " "     " "          " "
## 3  ( 1 )  " "        " "    " "     " "          " "
## 4  ( 1 )  " "        " "    " "     " "          " "
## 5  ( 1 )  " "        " "    " "     " "          " "
## 6  ( 1 )  " "        " "    " "     " "          " "
## 7  ( 1 )  " "        " "    " "     " "          " "
## 8  ( 1 )  " "        " "    " "     " "          "*"
## 9  ( 1 )  " "        " "    " "     " "          "*"
## 10  ( 1 ) " "        " "    " "     " "          "*"
## 11  ( 1 ) " "        " "    " "     " "          "*"
## 12  ( 1 ) " "        " "    " "     " "          "*"
## 13  ( 1 ) " "        " "    " "     " "          "*"
## 14  ( 1 ) " "        " "    " "     " "          "*"
## 15  ( 1 ) " "        " "    " "     " "          "*"
## 16  ( 1 ) " "        " "    " "     " "          "*"
## 17  ( 1 ) " "        " "    " "     " "          "*"
## 18  ( 1 ) " "        " "    " "     " "          "*"
##           diplomatic_embassy emergency_fire_hydrant bus_stop crossing
## 1  ( 1 )  " "                " "                    " "      " "
## 2  ( 1 )  " "                " "                    " "      " "
## 3  ( 1 )  " "                " "                    " "      " "
## 4  ( 1 )  " "                " "                    " "      " "
## 5  ( 1 )  " "                " "                    " "      " "
## 6  ( 1 )  " "                " "                    " "      " "
## 7  ( 1 )  " "                " "                    " "      " "
```

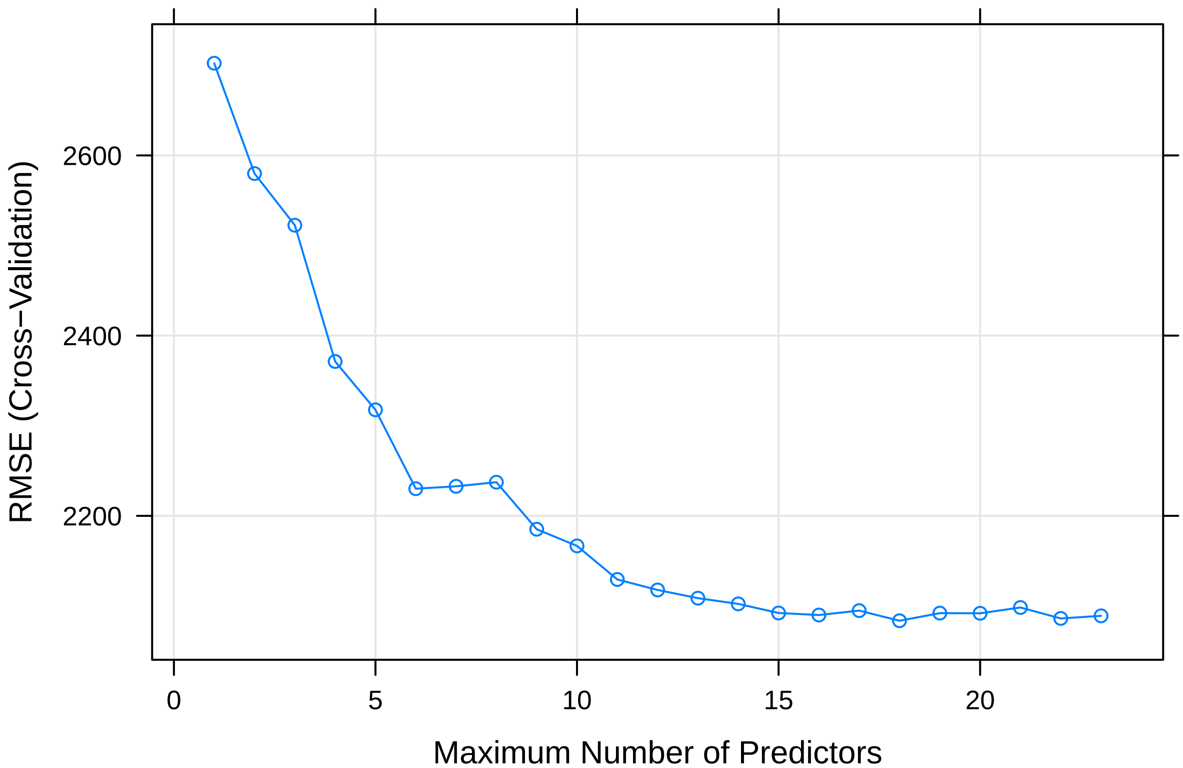
```
## 8  ( 1 )  " "                " "                     " "      " "
## 9  ( 1 )  " "                " "                     " "      " "
## 10  ( 1 ) " "                " "                     " "      " "
## 11  ( 1 ) " "                " "                     " "      " "
## 12  ( 1 ) " "                " "                     "*"      " "
## 13  ( 1 ) " "                " "                     "*"      " "
## 14  ( 1 ) " "                " "                     "*"      " "
## 15  ( 1 ) " "                " "                     "*"      " "
## 16  ( 1 ) " "                "*"                     "*"      " "
## 17  ( 1 ) " "                "*"                     "*"      " "
## 18  ( 1 ) " "                "*"                     "*"      " "
##           motorway_junction stop traffic_signals turning_circle
## 1  ( 1 )  " "               " "  " "             " "
## 2  ( 1 )  " "               " "  " "             " "
## 3  ( 1 )  " "               " "  " "             " "
## 4  ( 1 )  " "               " "  " "             " "
## 5  ( 1 )  " "               " "  " "             " "
## 6  ( 1 )  " "               "*"  " "             " "
## 7  ( 1 )  " "               "*"  " "             " "
## 8  ( 1 )  " "               "*"  " "             " "
## 9  ( 1 )  " "               "*"  " "             " "
## 10  ( 1 ) " "               "*"  " "             " "
## 11  ( 1 ) " "               "*"  " "             " "
## 12  ( 1 ) " "               "*"  " "             " "
## 13  ( 1 ) " "               "*"  " "             " "
## 14  ( 1 ) " "               "*"  " "             " "
## 15  ( 1 ) " "               "*"  " "             " "
## 16  ( 1 ) " "               "*"  " "             " "
## 17  ( 1 ) " "               "*"  " "             " "
## 18  ( 1 ) " "               "*"  " "             " "
##           historic_memorial historic_monument leisure_park
## 1  ( 1 )  " "               " "               " "
## 2  ( 1 )  " "               " "               " "
## 3  ( 1 )  " "               " "               " "
## 4  ( 1 )  " "               " "               " "
## 5  ( 1 )  " "               " "               " "
## 6  ( 1 )  " "               " "               " "
## 7  ( 1 )  " "               " "               " "
## 8  ( 1 )  " "               " "               " "
## 9  ( 1 )  " "               " "               " "
## 10  ( 1 ) " "               " "               " "
## 11  ( 1 ) " "               " "               " "
## 12  ( 1 ) " "               " "               " "
## 13  ( 1 ) " "               " "               " "
## 14  ( 1 ) " "               " "               " "
## 15  ( 1 ) " "               " "               " "
## 16  ( 1 ) " "               " "               " "
## 17  ( 1 ) " "               " "               " "
## 18  ( 1 ) " "               " "               " "
##           leisure_sports_centre office_government outdoor_seating
## 1  ( 1 )  " "                   " "               " "
## 2  ( 1 )  " "                   " "               " "
## 3  ( 1 )  " "                   " "               " "
## 4  ( 1 )  " "                   " "               " "
```

```
## 5  ( 1 )  " "                  " "              " "
## 6  ( 1 )  " "                  " "              " "
## 7  ( 1 )  " "                  "*"              " "
## 8  ( 1 )  " "                  "*"              " "
## 9  ( 1 )  " "                  "*"              " "
## 10  ( 1 ) " "                  "*"              " "
## 11  ( 1 ) " "                  "*"              " "
## 12  ( 1 ) " "                  "*"              " "
## 13  ( 1 ) " "                  "*"              " "
## 14  ( 1 ) " "                  "*"              " "
## 15  ( 1 ) " "                  "*"              " "
## 16  ( 1 ) " "                  "*"              " "
## 17  ( 1 ) " "                  "*"              " "
## 18  ( 1 ) " "                  "*"              " "
##           parking_underground railway_level_crossing railway_station
## 1  ( 1 )  " "                 " "                    " "
## 2  ( 1 )  " "                 " "                    " "
## 3  ( 1 )  " "                 " "                    " "
## 4  ( 1 )  " "                 " "                    " "
## 5  ( 1 )  " "                 " "                    " "
## 6  ( 1 )  " "                 " "                    " "
## 7  ( 1 )  " "                 " "                    " "
## 8  ( 1 )  " "                 " "                    " "
## 9  ( 1 )  " "                 " "                    " "
## 10  ( 1 ) " "                 " "                    " "
## 11  ( 1 ) " "                 " "                    " "
## 12  ( 1 ) " "                 " "                    " "
## 13  ( 1 ) " "                 " "                    " "
## 14  ( 1 ) " "                 " "                    " "
## 15  ( 1 ) " "                 " "                    " "
## 16  ( 1 ) " "                 " "                    " "
## 17  ( 1 ) " "                 " "                    " "
## 18  ( 1 ) " "                 " "                    " "
##           railway_subway_entrance shop_alcohol shop_art shop_bakery
## 1  ( 1 )  " "                     " "          " "      " "
## 2  ( 1 )  " "                     " "          " "      " "
## 3  ( 1 )  " "                     " "          " "      " "
## 4  ( 1 )  " "                     " "          " "      " "
## 5  ( 1 )  " "                     "*"          " "      " "
## 6  ( 1 )  " "                     "*"          " "      " "
## 7  ( 1 )  " "                     "*"          " "      " "
## 8  ( 1 )  " "                     "*"          " "      " "
## 9  ( 1 )  " "                     "*"          " "      " "
## 10  ( 1 ) " "                     "*"          " "      " "
## 11  ( 1 ) " "                     "*"          " "      " "
## 12  ( 1 ) " "                     "*"          " "      " "
## 13  ( 1 ) " "                     "*"          " "      " "
## 14  ( 1 ) " "                     "*"          " "      " "
## 15  ( 1 ) " "                     "*"          " "      " "
## 16  ( 1 ) " "                     "*"          " "      " "
## 17  ( 1 ) " "                     "*"          " "      " "
## 18  ( 1 ) " "                     "*"          " "      " "
##           shop_beauty shop_books shop_clothes shop_convenience
## 1  ( 1 )  " "         " "        " "          " "
```

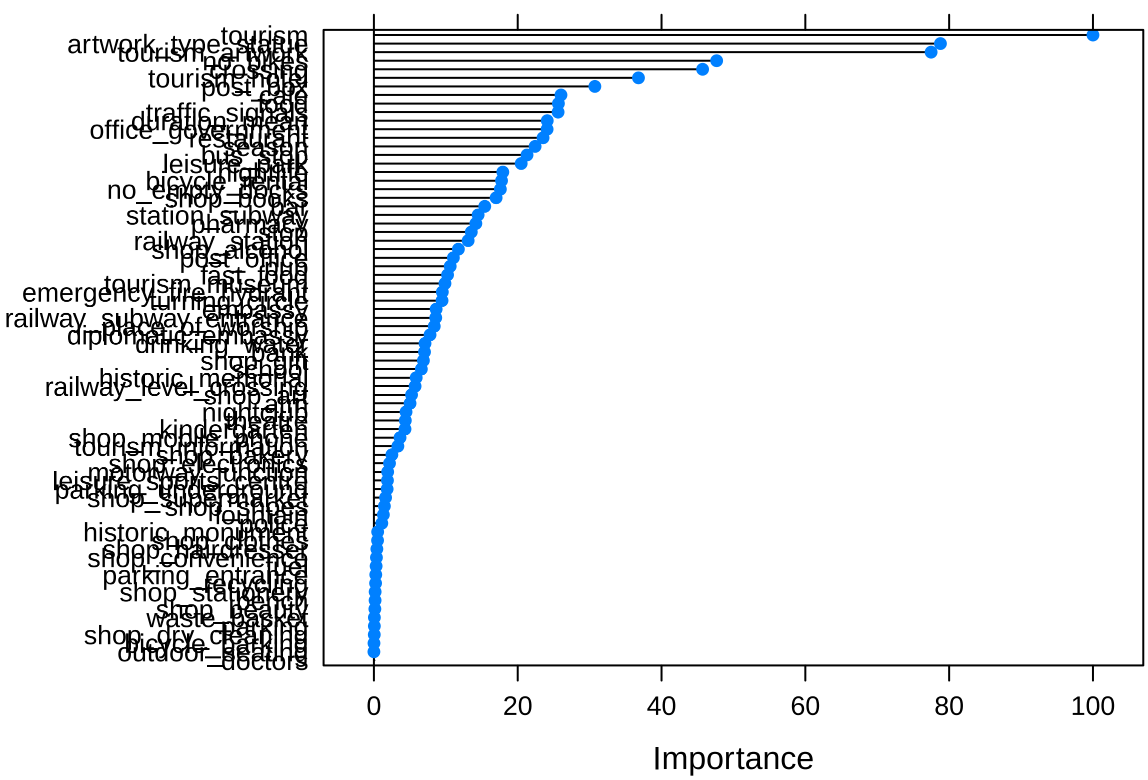
```
## 2  ( 1 )  " "          " "        " "          " "
## 3  ( 1 )  " "          " "        " "          " "
## 4  ( 1 )  " "          " "        " "          " "
## 5  ( 1 )  " "          " "        " "          " "
## 6  ( 1 )  " "          " "        " "          " "
## 7  ( 1 )  " "          " "        " "          " "
## 8  ( 1 )  " "          " "        " "          " "
## 9  ( 1 )  " "          " "        " "          " "
## 10  ( 1 ) " "          " "        " "          " "
## 11  ( 1 ) " "          " "        " "          " "
## 12  ( 1 ) " "          " "        " "          " "
## 13  ( 1 ) " "          " "        " "          " "
## 14  ( 1 ) " "          " "        " "          " "
## 15  ( 1 ) " "          " "        " "          " "
## 16  ( 1 ) " "          " "        " "          " "
## 17  ( 1 ) " "          " "        " "          " "
## 18  ( 1 ) " "          " "        " "          " "
##           shop_dry_cleaning shop_electronics shop_gift shop_hairdresser
## 1  ( 1 )  " "               " "              " "       " "
## 2  ( 1 )  " "               " "              " "       " "
## 3  ( 1 )  " "               " "              " "       " "
## 4  ( 1 )  " "               " "              " "       " "
## 5  ( 1 )  " "               " "              " "       " "
## 6  ( 1 )  " "               " "              " "       " "
## 7  ( 1 )  " "               " "              " "       " "
## 8  ( 1 )  " "               " "              " "       " "
## 9  ( 1 )  " "               " "              " "       " "
## 10  ( 1 ) " "               " "              " "       " "
## 11  ( 1 ) " "               " "              " "       " "
## 12  ( 1 ) " "               " "              " "       " "
## 13  ( 1 ) " "               " "              " "       " "
## 14  ( 1 ) " "               " "              " "       " "
## 15  ( 1 ) " "               " "              " "       " "
## 16  ( 1 ) " "               " "              " "       " "
## 17  ( 1 ) " "               " "              " "       " "
## 18  ( 1 ) " "               "*"              " "       " "
##           shop_mobile_phone shop_shoes shop_stationery shop_supermarket
## 1  ( 1 )  " "               " "        " "             " "
## 2  ( 1 )  " "               " "        " "             " "
## 3  ( 1 )  " "               " "        " "             " "
## 4  ( 1 )  " "               " "        " "             " "
## 5  ( 1 )  " "               " "        " "             " "
## 6  ( 1 )  " "               " "        " "             " "
## 7  ( 1 )  " "               " "        " "             " "
## 8  ( 1 )  " "               " "        " "             " "
## 9  ( 1 )  " "               " "        " "             " "
## 10  ( 1 ) " "               " "        " "             " "
## 11  ( 1 ) " "               " "        " "             " "
## 12  ( 1 ) " "               " "        " "             " "
## 13  ( 1 ) " "               " "        " "             " "
## 14  ( 1 ) " "               " "        " "             " "
## 15  ( 1 ) " "               " "        " "             " "
## 16  ( 1 ) " "               " "        " "             " "
## 17  ( 1 ) " "               " "        " "             " "
```

```
## 18  ( 1 ) " "                " "          " "              " "
##            station_subway tourism_artwork tourism_hotel tourism_information
## 1  ( 1 )  " "            " "             " "           " "
## 2  ( 1 )  " "            " "             " "           " "
## 3  ( 1 )  " "            " "             " "           " "
## 4  ( 1 )  " "            " "             " "           " "
## 5  ( 1 )  " "            " "             " "           " "
## 6  ( 1 )  " "            " "             " "           " "
## 7  ( 1 )  " "            " "             " "           " "
## 8  ( 1 )  " "            " "             " "           " "
## 9  ( 1 )  " "            " "             " "           " "
## 10  ( 1 ) " "            " "             " "           " "
## 11  ( 1 ) " "            " "             " "           " "
## 12  ( 1 ) " "            " "             " "           " "
## 13  ( 1 ) " "            " "             " "           "*"
## 14  ( 1 ) " "            " "             " "           "*"
## 15  ( 1 ) " "            " "             " "           "*"
## 16  ( 1 ) " "            " "             " "           "*"
## 17  ( 1 ) " "            " "             " "           "*"
## 18  ( 1 ) " "            " "             " "           "*"
##            tourism_museum food nightlife tourism
## 1  ( 1 )  " "            " "  " "       "*"
## 2  ( 1 )  " "            " "  " "       "*"
## 3  ( 1 )  " "            " "  " "       "*"
## 4  ( 1 )  " "            " "  " "       "*"
## 5  ( 1 )  " "            " "  " "       "*"
## 6  ( 1 )  " "            " "  " "       "*"
## 7  ( 1 )  " "            " "  " "       "*"
## 8  ( 1 )  " "            " "  " "       "*"
## 9  ( 1 )  " "            " "  " "       "*"
## 10  ( 1 ) " "            " "  " "       "*"
## 11  ( 1 ) " "            " "  " "       "*"
## 12  ( 1 ) " "            " "  " "       "*"
## 13  ( 1 ) " "            " "  " "       "*"
## 14  ( 1 ) " "            " "  " "       "*"
## 15  ( 1 ) " "            " "  " "       "*"
## 16  ( 1 ) " "            " "  " "       "*"
## 17  ( 1 ) " "            "*"  " "       "*"
## 18  ( 1 ) " "            "*"  " "       "*"
```

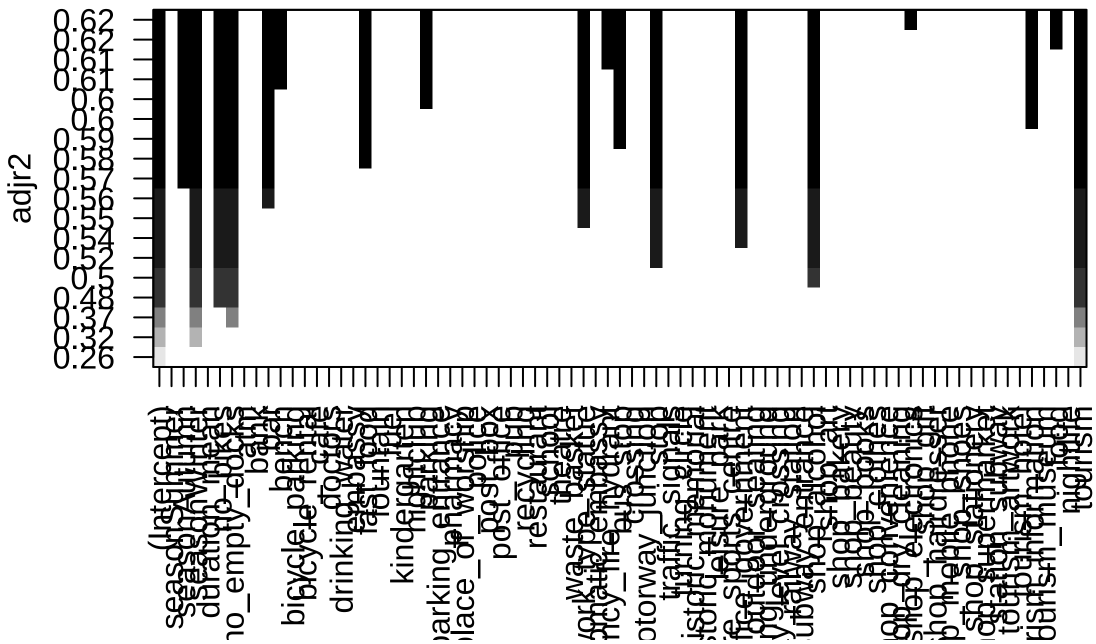
```r
plot(forward_model)
```



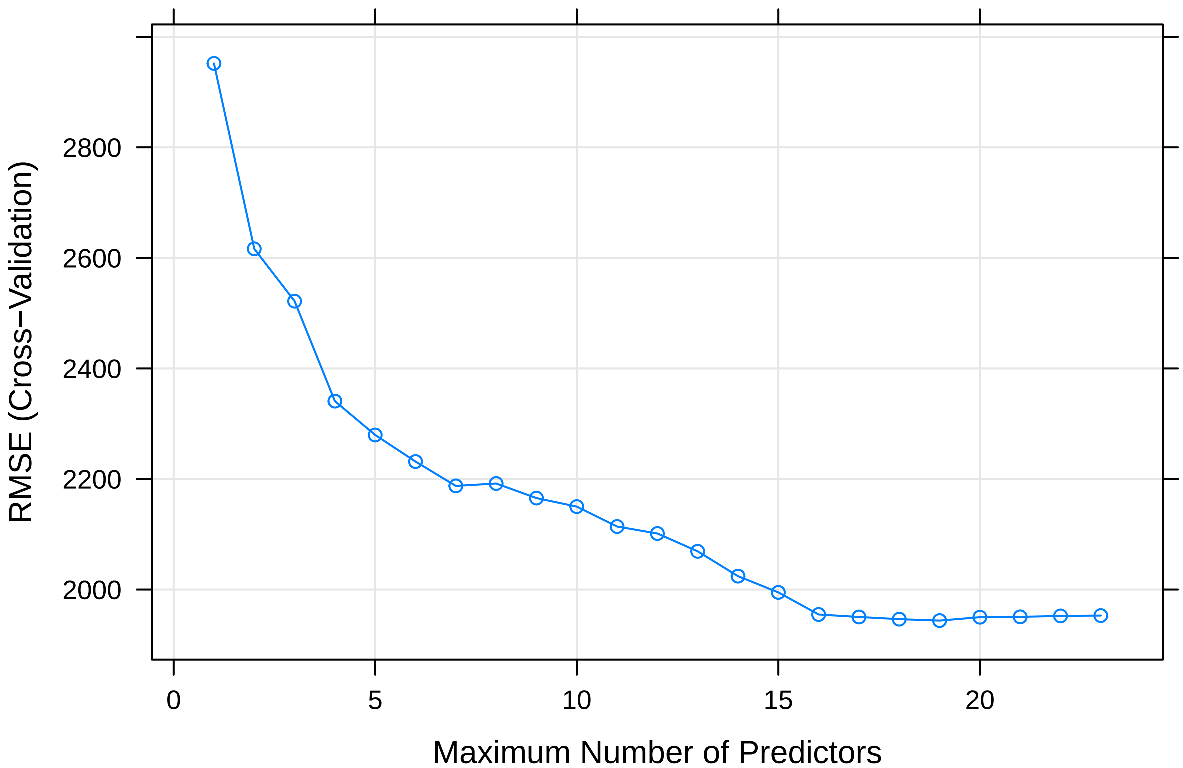

```r
plot(varImp(forward_model))
```



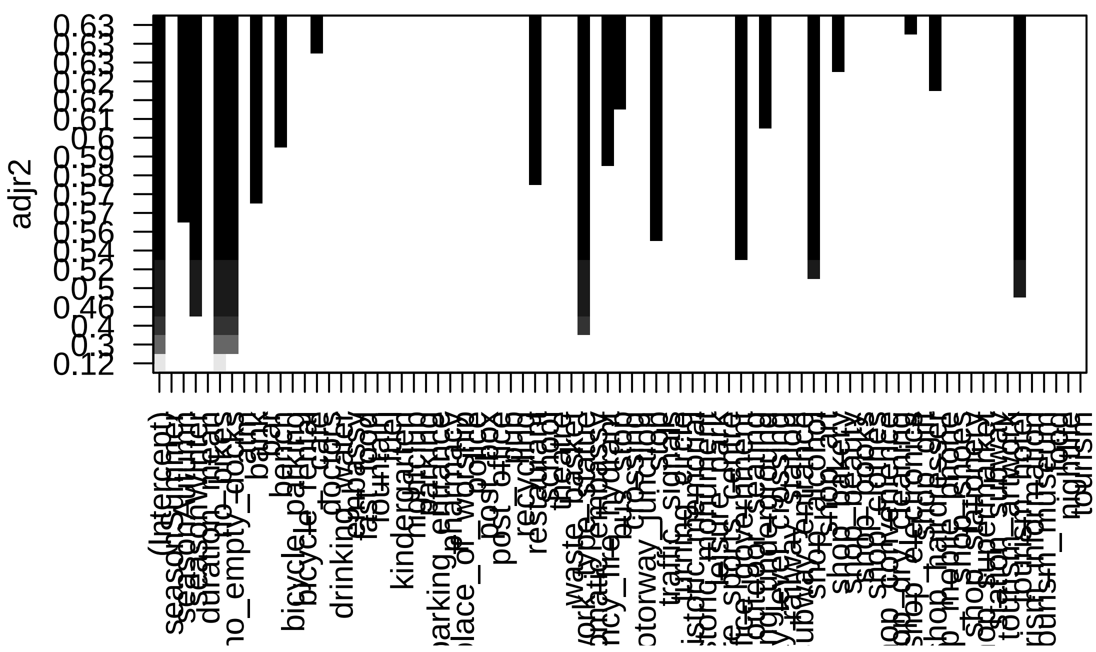
```r
# compare all the models
plot(forward_model$finalModel, scale = 'adjr2')
```



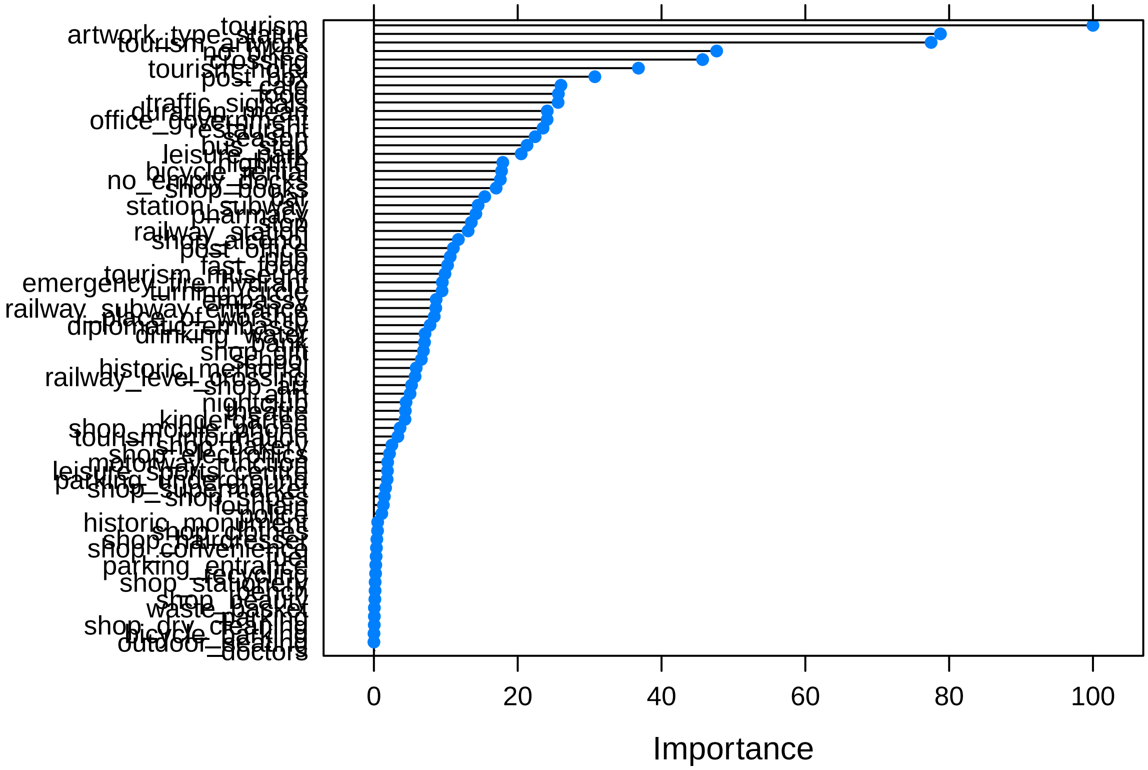

```r
# backward_selection
backward_model = train(num_rentals ~ .,
                       data = train,
                       method = 'leapBackward',
                       preProcess = c('center', 'scale'),
                       tuneGrid = expand.grid(nvmax = 1:23),
                       trControl = trainControl(method = 'cv', number = 5))

plot(backward_model)
```



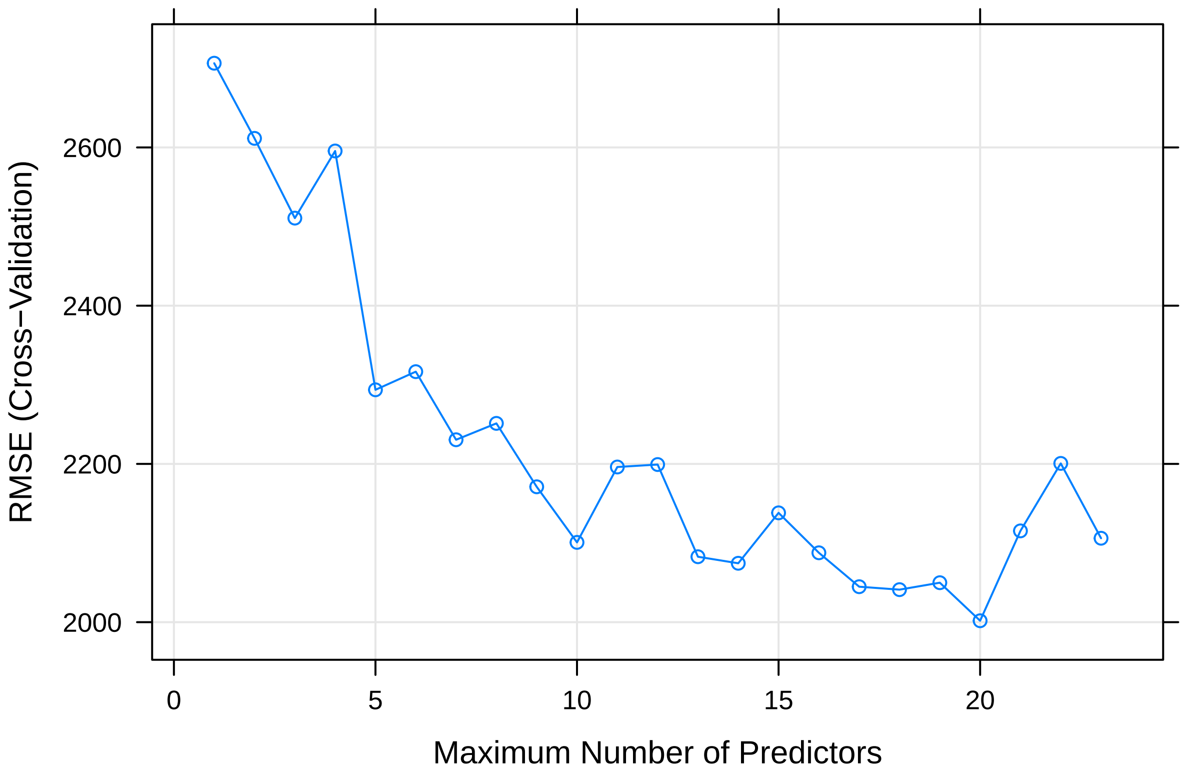

```r
plot(backward_model$finalModel, scale = 'adjr2')
```



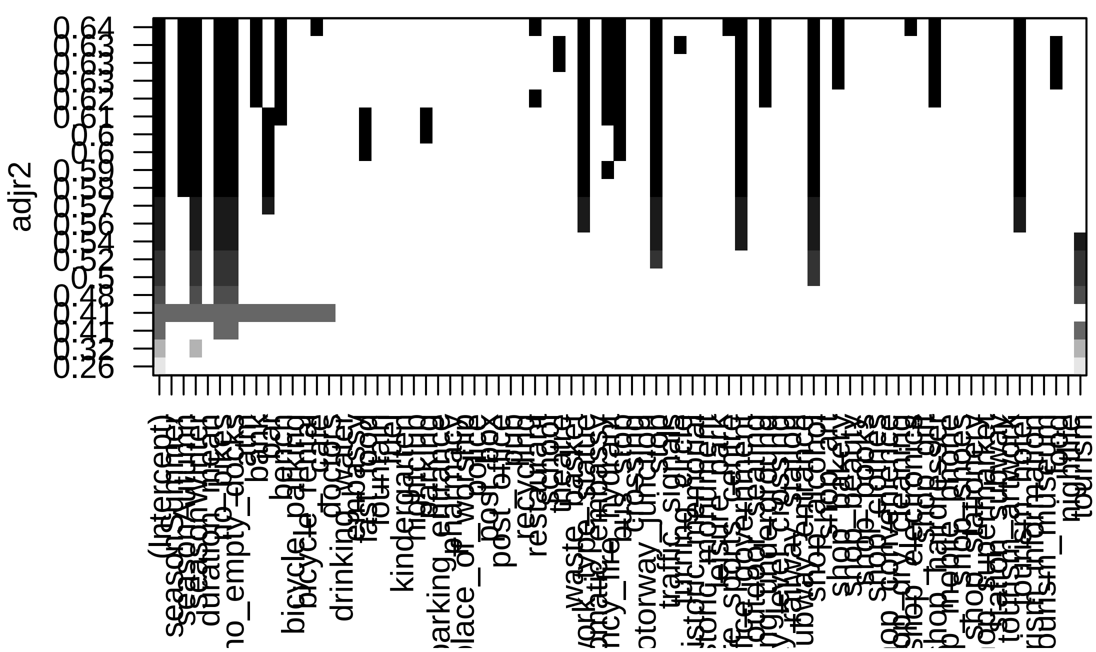

```r
plot(varImp(backward_model, scale = TRUE))
```



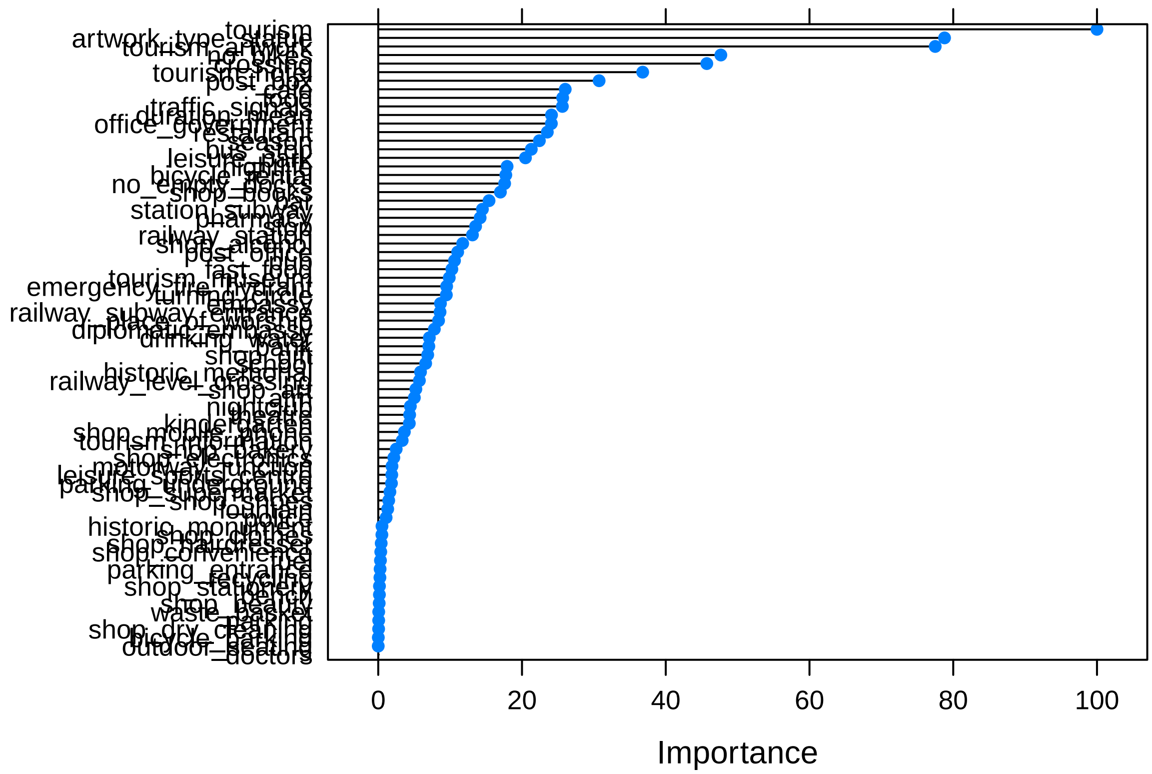

```r
# steps in both directions
hybrid_model = train(num_rentals ~ .,
                     data = train,
                     method = 'leapSeq',
                     preProcess = c('center', 'scale'),
                     tuneGrid = expand.grid(nvmax = 1:23),
                     trControl = trainControl(method = 'cv', number = 5))

plot(hybrid_model)
```




```r
plot(hybrid_model$finalModel, scale = 'adjr2')
```




```r
plot(varImp(hybrid_model))
```



Shrinkage

Ridge regression

```r
# ridge regression
ridge_model = train(num_rentals ~ .,
                    data = train,
                    method = 'ridge',
                    preProcess = c('center', 'scale'),
                    tuneLength = 10,
                    # reducing the cv for speed
                    trControl = trainControl(method = 'cv', number = 5))
```

```
## Loading required package: elasticnet
```

```
## Loading required package: lars
```

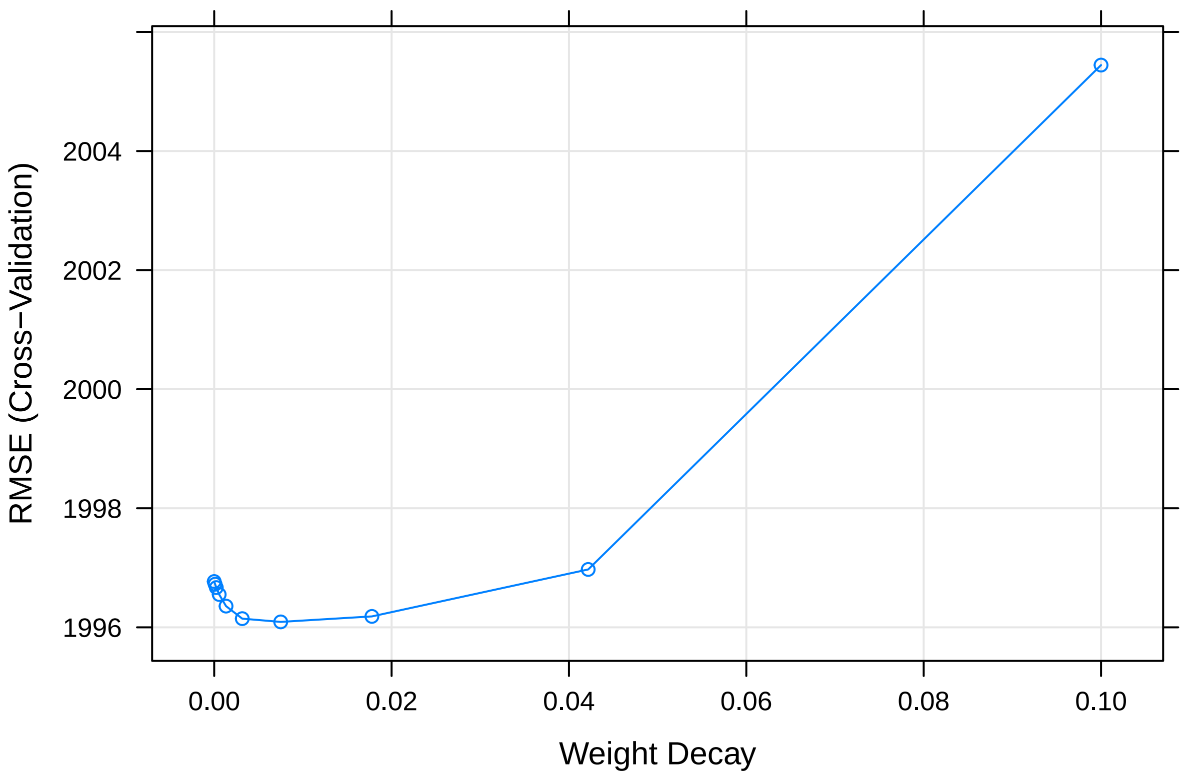
```
## Loaded lars 1.2
```

```r
print(ridge_model)
```

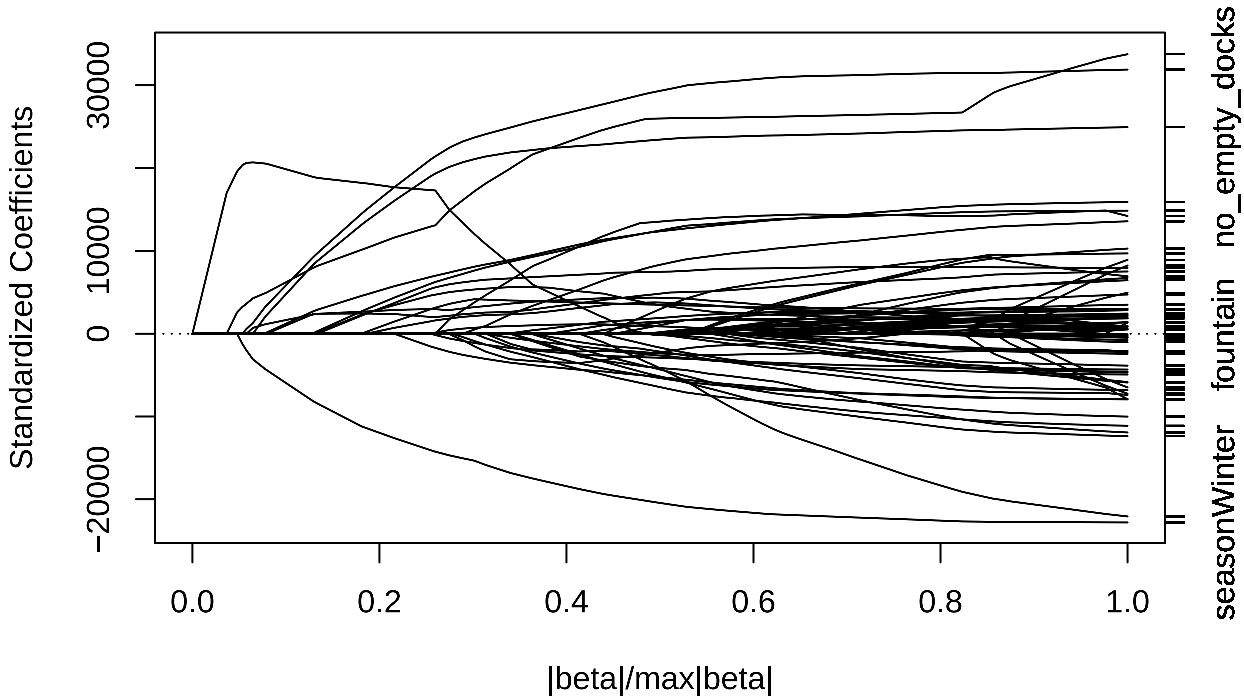
```
## Ridge Regression
##
## 753 samples
##  74 predictor
```

```
##
## Pre-processing: centered (76), scaled (76)
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 603, 602, 603, 603, 601
## Resampling results across tuning parameters:
##
##   lambda        RMSE      Rsquared   RMSE SD   Rsquared SD
##   0.0000000000  1996.770  0.6071052  317.6141  0.1040422
##   0.0001000000  1996.725  0.6071134  317.6939  0.1040771
##   0.0002371374  1996.668  0.6071234  317.7993  0.1041236
##   0.0005623413  1996.551  0.6071419  318.0324  0.1042279
##   0.0013335214  1996.356  0.6071651  318.5070  0.1044478
##   0.0031622777  1996.146  0.6071611  319.3251  0.1048632
##   0.0074989421  1996.091  0.6070881  320.3504  0.1055194
##   0.0177827941  1996.184  0.6070350  320.9800  0.1063080
##   0.0421696503  1996.972  0.6069992  321.0851  0.1071478
##   0.1000000000  2005.444  0.6050756  324.0219  0.1089584
##
## RMSE was used to select the optimal model using  the smallest value.
## The final value used for the model was lambda = 0.007498942.
```

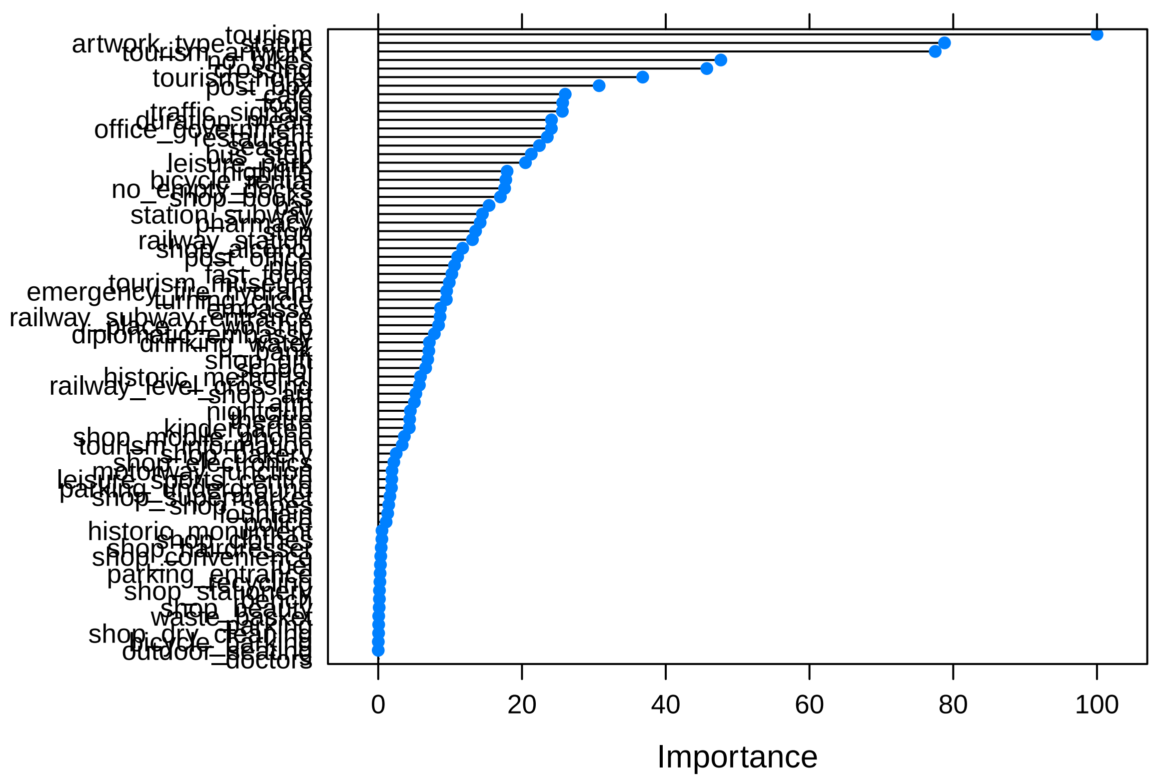
```
plot(ridge_model)
```




```
plot(ridge_model$finalModel)
```




```r
plot(varImp(ridge_model))
```




```r
# get the coefficients for the model
# NOTE: shrinkage methods don't have intercept terms
ridge_coefs = predict(ridge_model$finalModel, type = 'coef', mode = 'norm')$coefficients
```

```r
# ridge regression with variable selection
ridge_model2 = train(num_rentals ~ .,
                     data = train,
                     method = 'foba',
                     preProcess = c('center', 'scale'),
                     tuneLength = 10,
                     trControl = trainControl(method = 'cv', number = 5))
```

```
## Loading required package: foba
```

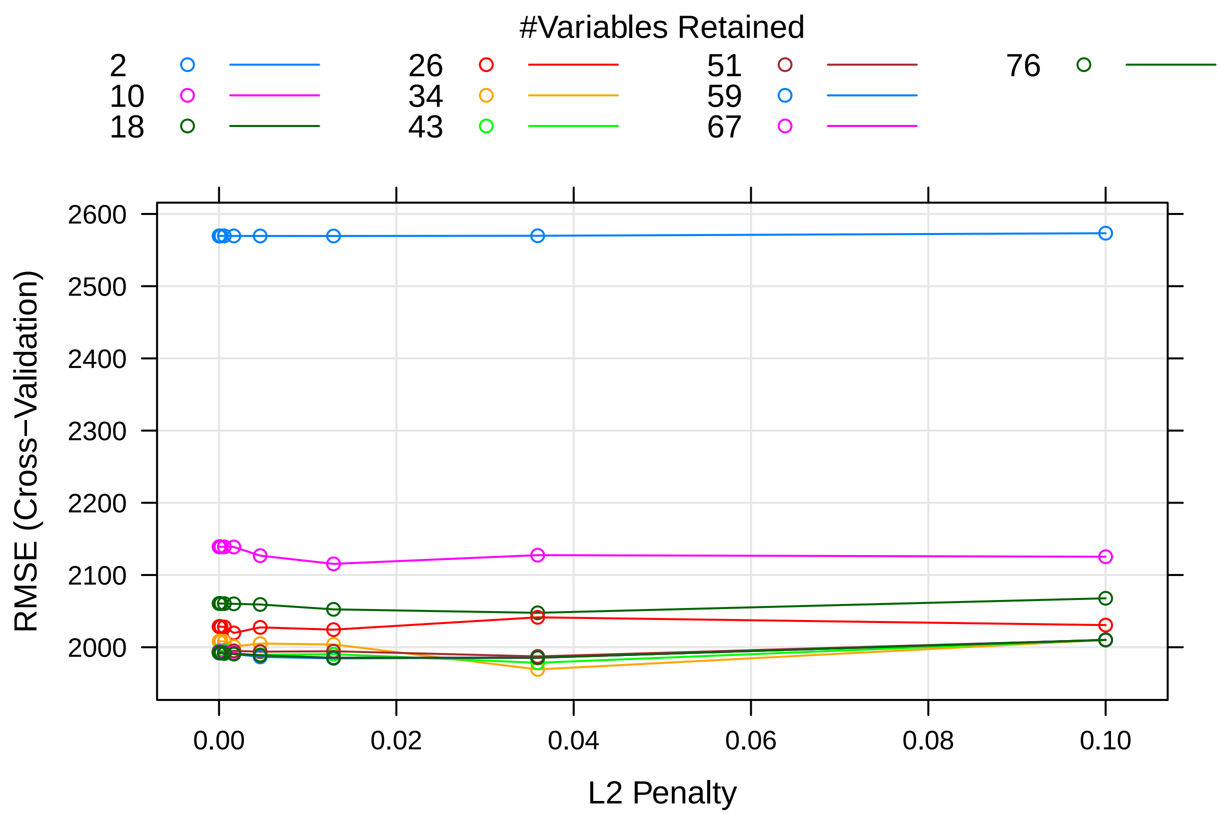
```r
print(ridge_model2)
```

```
## Ridge Regression with Variable Selection
##
## 753 samples
##  74 predictor
##
## Pre-processing: centered (76), scaled (76)
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 602, 603, 601, 605, 601
## Resampling results across tuning parameters:
##
##   lambda        k   RMSE      Rsquared   RMSE SD   Rsquared SD
##   1.000000e-05   2  2569.586  0.3329591  530.3167  0.1396605
##   1.000000e-05  10  2139.131  0.5395007  557.0840  0.1365352
##   1.000000e-05  18  2060.620  0.5774391  548.0785  0.1228851
##   1.000000e-05  26  2028.582  0.5936032  509.2843  0.1070831
##   1.000000e-05  34  2008.246  0.6016328  512.4131  0.1118143
##   1.000000e-05  43  1995.268  0.6062579  496.3489  0.1113152
##   1.000000e-05  51  1993.653  0.6080117  500.9346  0.1128202
##   1.000000e-05  59  1992.089  0.6081461  504.4870  0.1135275
##   1.000000e-05  67  1993.856  0.6082945  497.6157  0.1120051
##   1.000000e-05  76  1992.365  0.6087346  497.2352  0.1120933
##   2.782559e-05   2  2569.586  0.3329591  530.3174  0.1396605
##   2.782559e-05  10  2139.127  0.5395006  557.0890  0.1365365
##   2.782559e-05  18  2060.615  0.5774392  548.0855  0.1228862
##   2.782559e-05  26  2028.572  0.5936048  509.2953  0.1070838
##   2.782559e-05  34  2008.239  0.6016330  512.4204  0.1118136
##   2.782559e-05  43  1995.258  0.6062580  496.3671  0.1113170
##   2.782559e-05  51  1993.642  0.6080118  500.9517  0.1128235
##   2.782559e-05  59  1992.070  0.6081491  504.4880  0.1135241
##   2.782559e-05  67  1993.811  0.6083041  497.6386  0.1120061
##   2.782559e-05  76  1992.341  0.6087367  497.2649  0.1120993
##   7.742637e-05   2  2569.585  0.3329591  530.3193  0.1396605
##   7.742637e-05  10  2139.117  0.5395005  557.1031  0.1365399
##   7.742637e-05  18  2060.599  0.5774395  548.1050  0.1228891
##   7.742637e-05  26  2028.543  0.5936092  509.3258  0.1070857
##   7.742637e-05  34  2008.221  0.6016333  512.4405  0.1118119
##   7.742637e-05  43  1995.228  0.6062583  496.4179  0.1113218
##   7.742637e-05  51  1993.612  0.6080119  500.9991  0.1128328
##   7.742637e-05  59  1992.018  0.6081572  504.4911  0.1135147
##   7.742637e-05  67  1993.689  0.6083303  497.7019  0.1120089
##   7.742637e-05  76  1992.250  0.6087543  497.2966  0.1120911
```

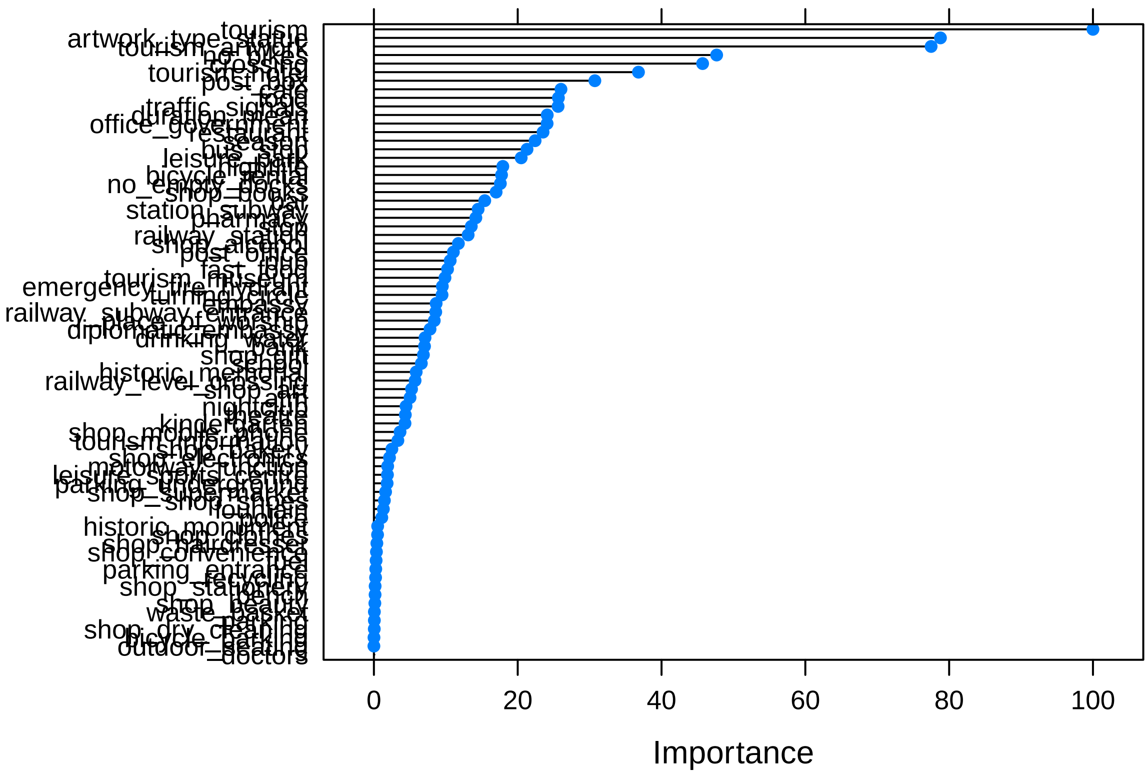
```
## 2.154435e-04  2 2569.583 0.3329592 530.3245 0.1396605
## 2.154435e-04 10 2139.090 0.5395000 557.1423 0.1365495
## 2.154435e-04 18 2060.556 0.5774405 548.1593 0.1228971
## 2.154435e-04 26 2028.462 0.5936213 509.4105 0.1070909
## 2.154435e-04 34 2008.171 0.6016342 512.4965 0.1118071
## 2.154435e-04 43 1995.146 0.6062586 496.5577 0.1113351
## 2.154435e-04 51 1993.531 0.6080120 501.1294 0.1128582
## 2.154435e-04 59 1991.877 0.6081784 504.5024 0.1134903
## 2.154435e-04 67 1993.362 0.6083998 497.8740 0.1120170
## 2.154435e-04 76 1992.005 0.6087991 497.3832 0.1120687
## 5.994843e-04  2 2569.577 0.3329592 530.3389 0.1396606
## 5.994843e-04 10 2139.015 0.5394985 557.2510 0.1365761
## 5.994843e-04 18 2060.435 0.5774430 548.3101 0.1229193
## 5.994843e-04 26 2028.090 0.5937610 509.3501 0.1068709
## 5.994843e-04 34 2008.034 0.6016362 512.6521 0.1117939
## 5.994843e-04 43 1995.586 0.6058435 498.3117 0.1122690
## 5.994843e-04 51 1992.506 0.6083422 502.4535 0.1132952
## 5.994843e-04 59 1991.711 0.6081299 504.9620 0.1136480
## 5.994843e-04 67 1993.325 0.6081286 497.8797 0.1122540
## 5.994843e-04 76 1991.355 0.6089100 497.6409 0.1120131
## 1.668101e-03  2 2569.560 0.3329595 530.3791 0.1396606
## 1.668101e-03 10 2138.809 0.5394942 557.5519 0.1366497
## 1.668101e-03 18 2060.105 0.5774496 548.7282 0.1229812
## 1.668101e-03 26 2019.786 0.5962261 519.1975 0.1078813
## 1.668101e-03 34 2001.034 0.6040981 517.7566 0.1129356
## 1.668101e-03 43 1990.451 0.6086122 500.4724 0.1138543
## 1.668101e-03 51 1995.012 0.6072857 502.3423 0.1125678
## 1.668101e-03 59 1990.594 0.6084215 505.2267 0.1136724
## 1.668101e-03 67 1991.782 0.6083818 497.9173 0.1118950
## 1.668101e-03 76 1990.661 0.6088797 497.0956 0.1117008
## 4.641589e-03  2 2569.522 0.3329600 530.4904 0.1396607
## 4.641589e-03 10 2126.810 0.5445914 571.5325 0.1375220
## 4.641589e-03 18 2059.223 0.5774645 549.8784 0.1231527
## 4.641589e-03 26 2027.499 0.5936173 512.2053 0.1065778
## 4.641589e-03 34 2005.090 0.6024558 520.1899 0.1139344
## 4.641589e-03 43 1989.555 0.6078112 496.3706 0.1110752
## 4.641589e-03 51 1993.838 0.6067362 501.2802 0.1120877
## 4.641589e-03 59 1986.624 0.6094074 502.7198 0.1119519
## 4.641589e-03 67 1988.543 0.6090916 499.3859 0.1117300
## 4.641589e-03 76 1988.771 0.6089542 499.1182 0.1117095
## 1.291550e-02  2 2569.474 0.3329616 530.7965 0.1396610
## 1.291550e-02 10 2115.447 0.5488463 578.7667 0.1415599
## 1.291550e-02 18 2052.463 0.5774236 553.7848 0.1246777
## 1.291550e-02 26 2024.346 0.5935269 509.1768 0.1006142
## 1.291550e-02 34 2003.893 0.6022598 522.9257 0.1135997
## 1.291550e-02 43 1990.388 0.6062695 508.6311 0.1137296
## 1.291550e-02 51 1994.453 0.6051133 507.9125 0.1138498
## 1.291550e-02 59 1984.534 0.6086893 505.1924 0.1118076
## 1.291550e-02 67 1985.390 0.6084336 504.1237 0.1118646
## 1.291550e-02 76 1985.390 0.6084336 504.1237 0.1118646
## 3.593814e-02  2 2569.767 0.3329659 531.6197 0.1396618
## 3.593814e-02 10 2127.538 0.5412244 569.1162 0.1436079
## 3.593814e-02 18 2047.669 0.5783605 575.9643 0.1304433
## 3.593814e-02 26 2041.409 0.5826229 548.0318 0.1147410
```

```
##    3.593814e-02  34  1969.306  0.6127369  519.7692  0.1100842
##    3.593814e-02  43  1978.472  0.6094304  514.7985  0.1123574
##    3.593814e-02  51  1987.263  0.6058990  512.0406  0.1136882
##    3.593814e-02  59  1985.453  0.6065234  513.2441  0.1139776
##    3.593814e-02  67  1985.453  0.6065234  513.2441  0.1139776
##    3.593814e-02  76  1985.453  0.6065234  513.2441  0.1139776
##    1.000000e-01   2  2573.331  0.3329767  533.6973  0.1396636
##    1.000000e-01  10  2125.321  0.5417912  563.7766  0.1300092
##    1.000000e-01  18  2067.778  0.5685217  569.1134  0.1271967
##    1.000000e-01  26  2030.678  0.5845239  557.9117  0.1322013
##    1.000000e-01  34  2010.148  0.5947831  517.1842  0.1092050
##    1.000000e-01  43  2010.148  0.5947831  517.1842  0.1092050
##    1.000000e-01  51  2010.148  0.5947831  517.1842  0.1092050
##    1.000000e-01  59  2010.148  0.5947831  517.1842  0.1092050
##    1.000000e-01  67  2010.148  0.5947831  517.1842  0.1092050
##    1.000000e-01  76  2010.148  0.5947831  517.1842  0.1092050
##
## RMSE was used to select the optimal model using  the smallest value.
## The final values used for the model were k = 34 and lambda = 0.03593814.
```

```r
plot(ridge_model2)
```




```r
plot(varImp(ridge_model2))
```



Selection, ridge regression, and lasso are just a couple techniques at our disposal for decreasing our model size. See this page for a list of other available options to try out if you like.

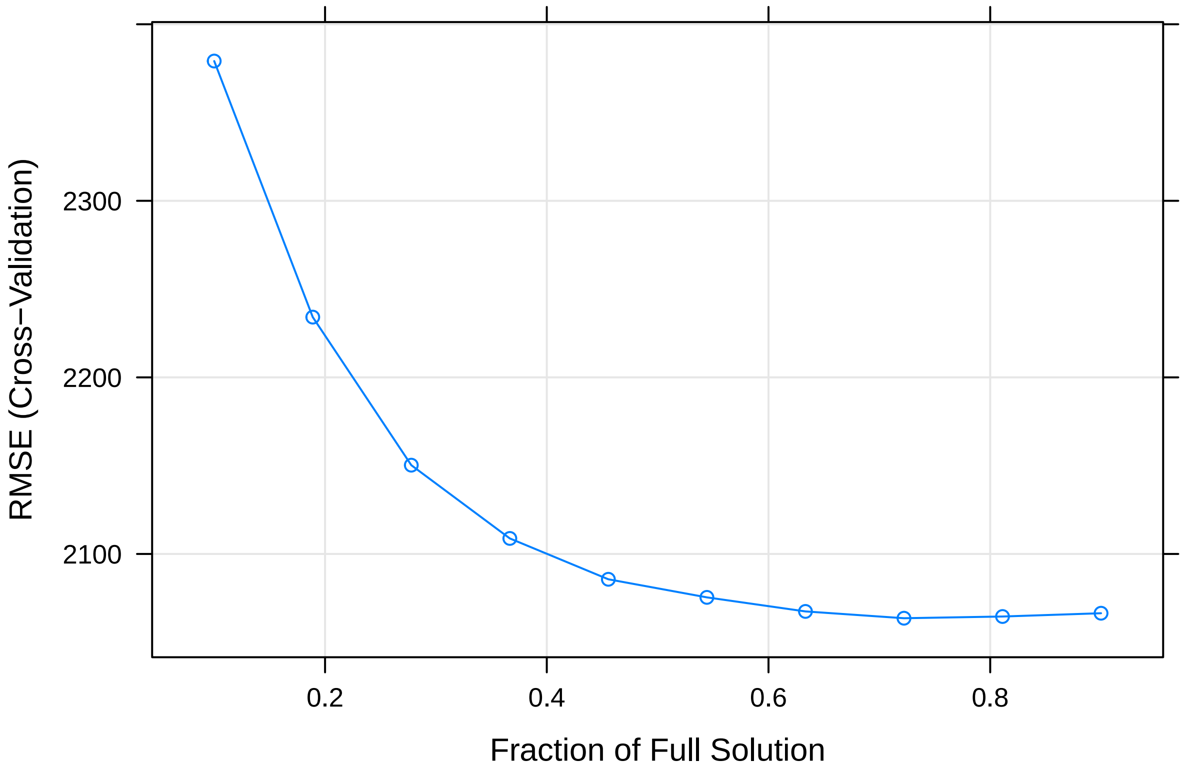
Lasso

```
lasso_model = train(num_rentals ~ .,
                    data = train,
                    method = 'lasso',
                    preProc = c('scale', 'center'),
                    tuneLength = 10,
                    trControl = trainControl(method = 'cv', number = 5))

print(lasso_model)
```

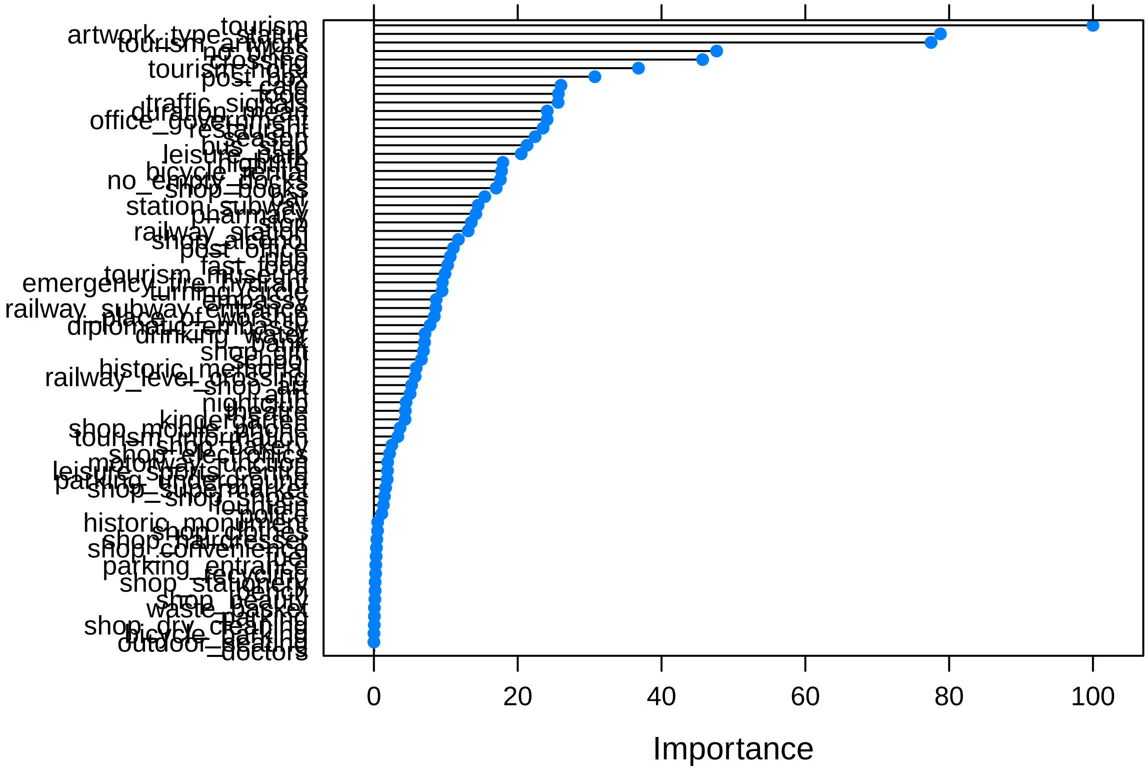
```
## The lasso
##
## 753 samples
##  74 predictor
##
## Pre-processing: scaled (76), centered (76)
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 602, 601, 604, 601, 604
## Resampling results across tuning parameters:
##
##   fraction   RMSE      Rsquared   RMSE SD   Rsquared SD
##   0.1000000  2379.154  0.4620301  654.0488  0.14682933
##   0.1888889  2234.123  0.5184435  515.9106  0.09789831
```

```
##    0.2777778  2150.268  0.5479330  476.0123  0.08260864
##    0.3666667  2108.790  0.5637786  457.7262  0.06992674
##    0.4555556  2085.648  0.5727649  441.2429  0.06083687
##    0.5444444  2075.402  0.5785919  434.7210  0.05801578
##    0.6333333  2067.445  0.5838539  433.4509  0.05722407
##    0.7222222  2063.596  0.5874240  434.9315  0.05773185
##    0.8111111  2064.560  0.5880592  432.7686  0.05712464
##    0.9000000  2066.423  0.5881793  430.4743  0.05674076
##
## RMSE was used to select the optimal model using  the smallest value.
## The final value used for the model was fraction = 0.7222222.
```

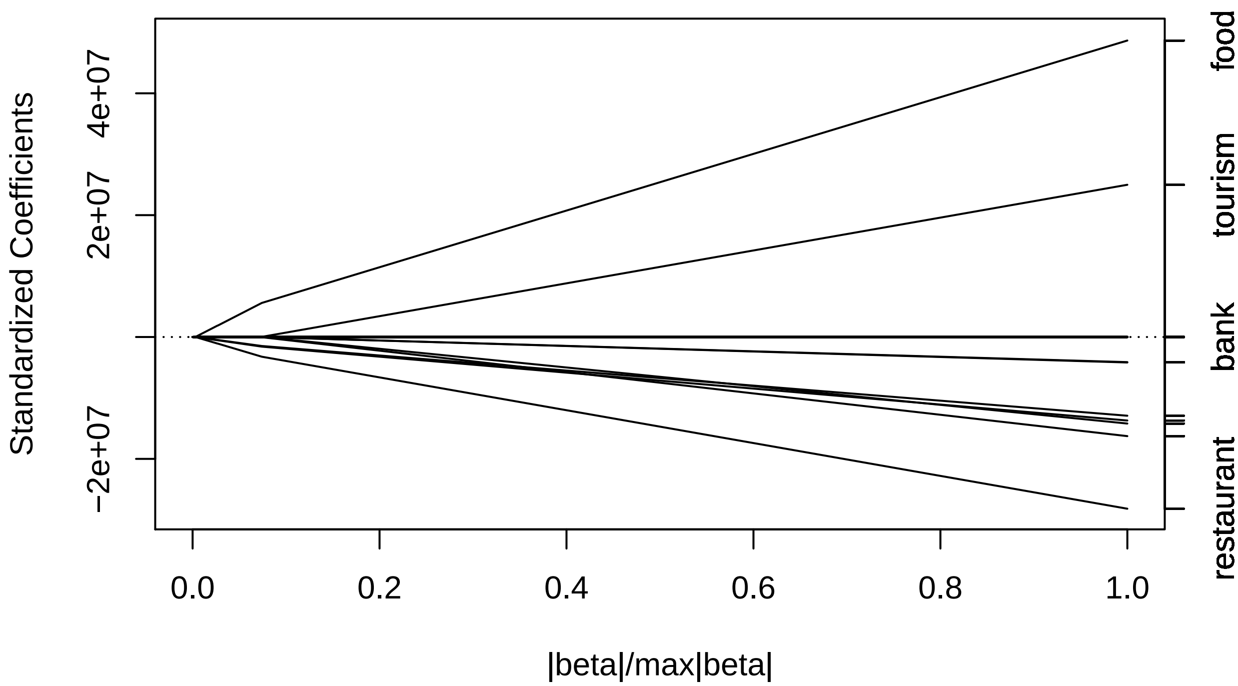
```r
plot(lasso_model)
```




```r
plot(varImp(lasso_model))
```




```r
plot(lasso_model$finalModel)
```




```r
# get the model coefficients
lasso_coefs = predict(lasso_model$finalModel, type = 'coef', mode = 'norm')$coefficients
```

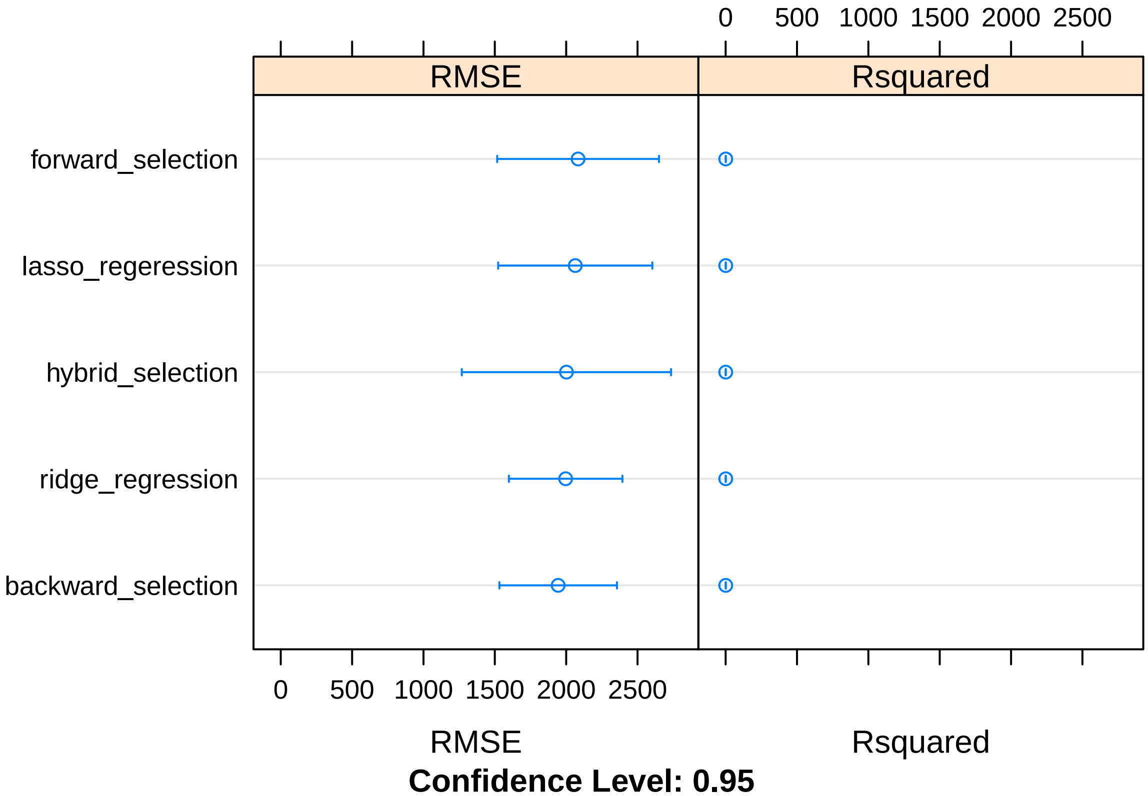
Measuring predictive accuracy

All right, now we've got a nice collection of models. Which one should we report?

```
results = resamples(list(forward_selection = forward_model,
                         backward_selection = backward_model,
                         hybrid_selection = hybrid_model,
                         ridge_regression = ridge_model,
                         lasso_regeression = lasso_model))

# compare RMSE and R-squared
summary(results)
```

```
##
## Call:
## summary.resamples(object = results)
##
## Models: forward_selection, backward_selection, hybrid_selection, ridge_regression, lasso_regeression
## Number of resamples: 5
##
## RMSE
##                    Min. 1st Qu. Median Mean 3rd Qu. Max. NA's
## forward_selection  1665    1867   1992 2084    2035 2859    0
## backward_selection 1522    1810   1887 1944    2084 2415    0
## hybrid_selection   1628    1653   1796 2002    1891 3040    0
## ridge_regression   1628    1844   1871 1996    2199 2438    0
## lasso_regeression  1524    1771   2027 2064    2498 2499    0
##
## Rsquared
##                      Min. 1st Qu. Median   Mean 3rd Qu.   Max. NA's
## forward_selection  0.4539  0.5412 0.6011 0.5775  0.6268 0.6646    0
## backward_selection 0.6046  0.6126 0.6376 0.6312  0.6397 0.6616    0
## hybrid_selection   0.3825  0.6079 0.6349 0.6043  0.6926 0.7038    0
## ridge_regression   0.4736  0.5178 0.6441 0.6071  0.6919 0.7080    0
## lasso_regeression  0.5381  0.5478 0.5520 0.5874  0.6367 0.6626    0
```

```
# plot results
dotplot(results)
```



Those are in-sample statistics however, so if we want to compare the model's out-of-sample prediction accuracy, we need to compute the RMSE using the test data we held out. Let's compare two models: backward selection and lasso:

```
backward_predictions = predict(backward_model, test)
sqrt(mean((backward_predictions - test$rentals)^2 , na.rm = TRUE))
```

```
## [1] NaN
```

```
lasso_predictions = predict(lasso_model, test)
sqrt(mean((lasso_predictions - test$rentals)^2 , na.rm = TRUE))
```

```
## [1] NaN
```

Assignment 4