Linear regression in R

April 24, 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: $rentals = \beta_0 + \beta_1 * crossing$.

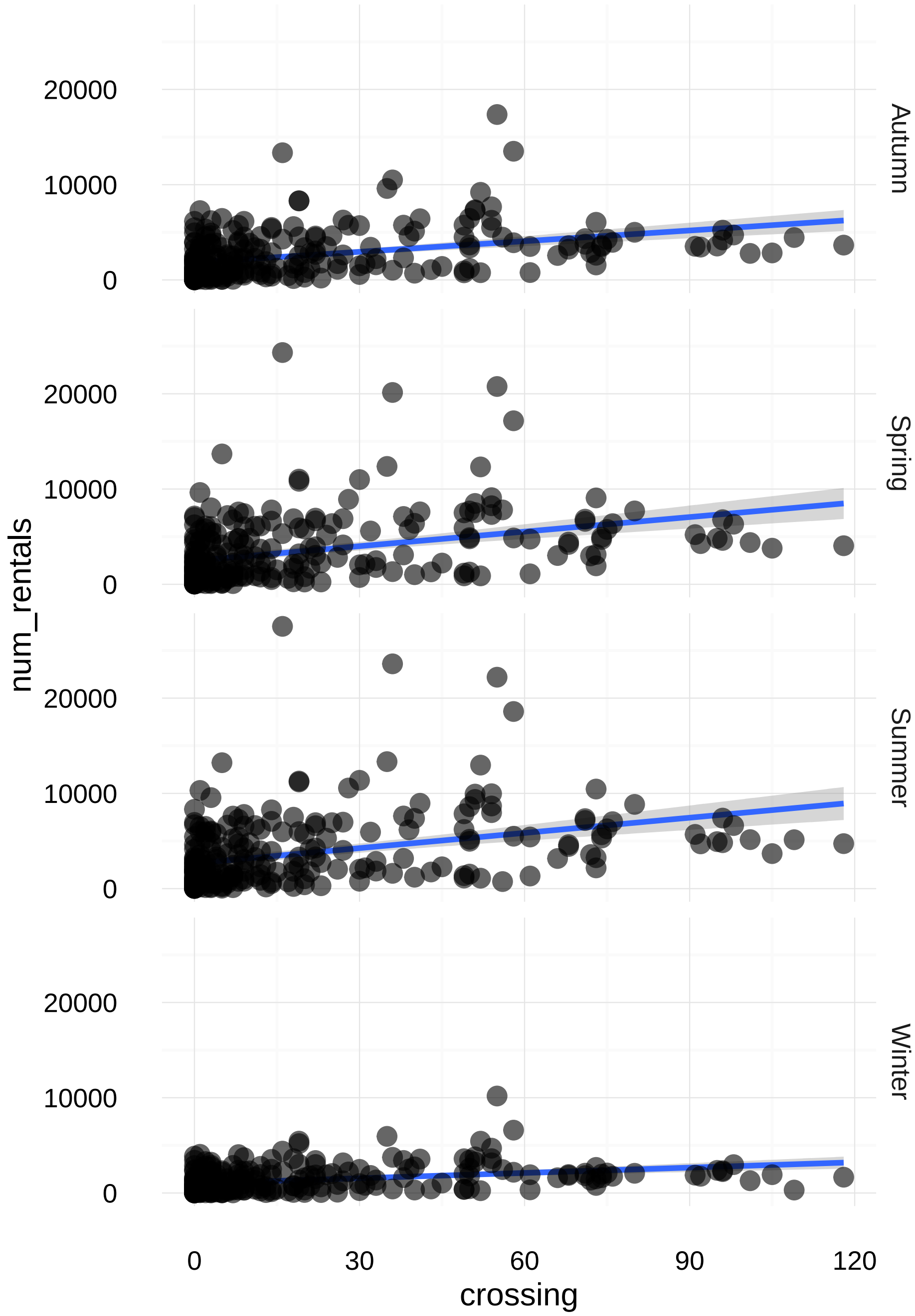
```r
model = lm(num_rentals ~ crossing, data = bikeshare)

# 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
```

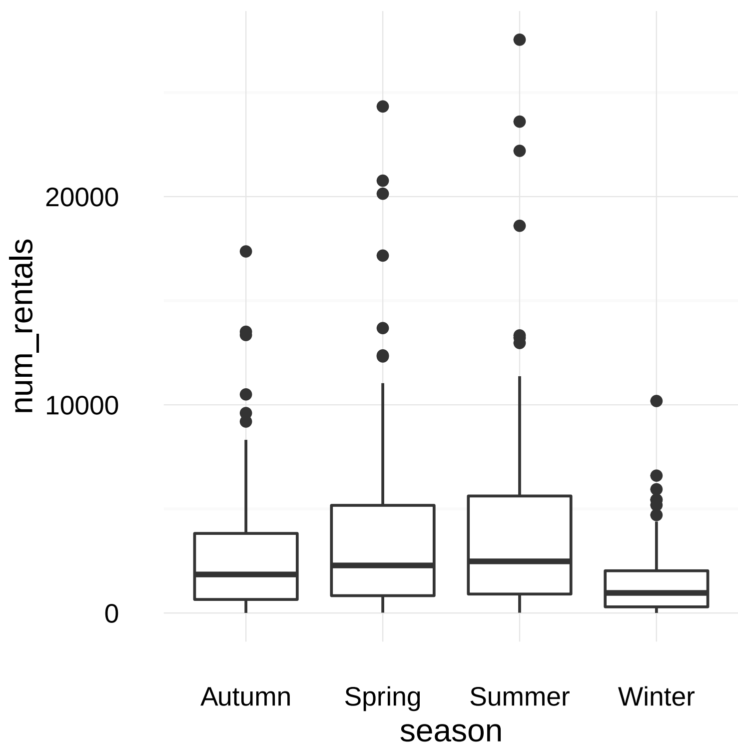
Let's try one more, this time we'll include `season`, a factor variable:

```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*:

```r
# 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,]         4
## [4,]         5
## [5,]         7
## [6,]         8
```

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

dim(train)
```

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

```r
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:

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

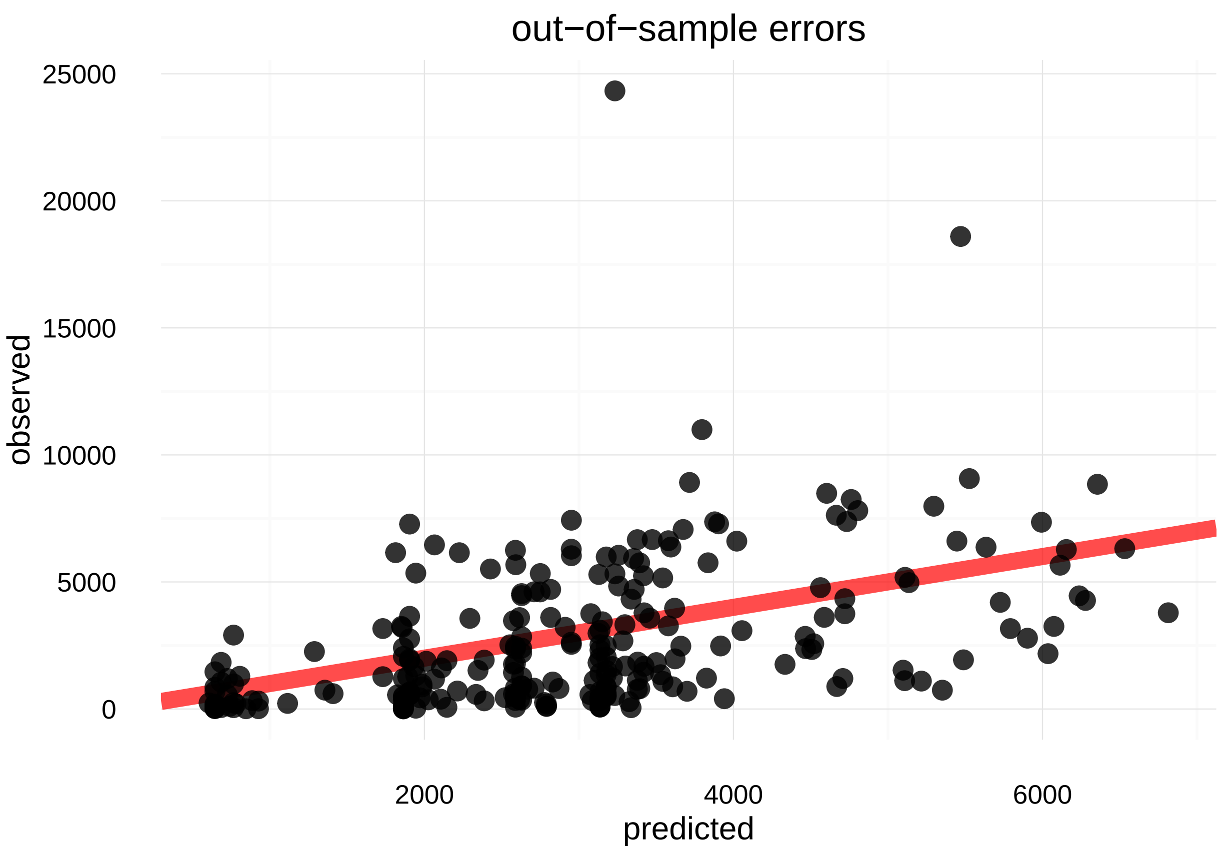
```
## Linear Regression
##
## 753 samples
##  75 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 753, 753, 753, 753, 753, 753, ...
## Resampling results:
##
##   RMSE      Rsquared
##   2853.694  0.2028326
##
##
```

```
summary(model_fit)
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4710.8 -1795.5  -599.3  1075.6 23755.7
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2589.10     229.84  11.265  < 2e-16 ***
## crossing        40.24       4.00  10.060  < 2e-16 ***
## parking        -12.61      73.05  -0.173   0.8630
## seasonSummer   547.34     302.85   1.807   0.0711 .
## seasonAutumn  -725.12     302.03  -2.401   0.0166 *
## seasonWinter -1944.78     292.70  -6.644 5.87e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2889 on 747 degrees of freedom
## Multiple R-squared:  0.1991, Adjusted R-squared:  0.1937
## F-statistic: 37.14 on 5 and 747 DF,  p-value: < 2.2e-16
```

```
# 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
##   2258.428  0.5342222
```

```
##
##
```

```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 21 rentals per day. In the previous full-model we had an intercept of about -28, 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? Or a temp 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 = to_model[in_train, ]
test = to_model[-in_train, ]

dim(train)
```

```
## [1] 753  65
```

```r
dim(test)
```

```
## [1] 248  65
```

```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"
```

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

```
##    nvmax
## 23    23
```

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

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

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

```
## Linear Regression with Forward Selection
##
## 753 samples
##  64 predictor
##
## Pre-processing: centered (66), scaled (66)
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 602, 603, 604, 601, 602
## Resampling results across tuning parameters:
##
##   nvmax  RMSE      Rsquared
##    1     2792.049  0.2678977
##    2     2653.999  0.3361394
##    3     2600.569  0.3621359
##    4     2433.095  0.4323640
##    5     2377.693  0.4546926
##    6     2308.525  0.4882945
##    7     2265.159  0.5063584
##    8     2259.752  0.5091623
##    9     2256.121  0.5109204
##   10     2238.630  0.5187352
##   11     2240.470  0.5181165
##   12     2235.475  0.5199525
##   13     2234.486  0.5210654
##   14     2211.794  0.5293513
##   15     2199.534  0.5338344
##   16     2177.552  0.5428467
##   17     2172.628  0.5446580
##   18     2169.248  0.5465543
```

```
##   19     2164.407  0.5491195
##   20     2149.442  0.5545740
##   21     2148.760  0.5547631
##   22     2146.925  0.5562826
##   23     2136.759  0.5604383
##
## RMSE was used to select the optimal model using  the smallest value.
## The final value used for the model was nvmax = 23.
```

```r
summary(forward_model)
```

```
## Subset selection object
## 66 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
## bench                          FALSE      FALSE
## bicycle_parking                FALSE      FALSE
## bicycle_rental                 FALSE      FALSE
## doctors                        FALSE      FALSE
## drinking_water                 FALSE      FALSE
## embassy                        FALSE      FALSE
## fountain                       FALSE      FALSE
## fuel                           FALSE      FALSE
## kindergarten                   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
## recycling                      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
## food                        FALSE      FALSE
## nightlife                   FALSE      FALSE
## tourism                     FALSE      FALSE
## 1 subsets of each size up to 23
## 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 ) " "          "*"          "*"          " "           "*"     
## 19  ( 1 ) " "          "*"          "*"          " "           "*"     
## 20  ( 1 ) "*"          "*"          "*"          " "           "*"     
## 21  ( 1 ) "*"          "*"          "*"          " "           "*"     
## 22  ( 1 ) "*"          "*"          "*"          " "           "*"     
## 23  ( 1 ) "*"          "*"          "*"          " "           "*"     
##           no_empty_docks atm bank 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 ) "*"            " " " "  "*"   " "             " "
## 19  ( 1 ) "*"            " " " "  "*"   " "             " "
## 20  ( 1 ) "*"            " " " "  "*"   " "             " "
## 21  ( 1 ) "*"            " " "*"  "*"   " "             " "
## 22  ( 1 ) "*"            " " "*"  "*"   " "             " "
## 23  ( 1 ) "*"            " " "*"  "*"   " "             " "
##           doctors drinking_water embassy fountain fuel kindergarten
## 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 ) " "     " "            " "     " "      " "  " "
## 19  ( 1 ) "*"     " "            " "     " "      " "  " "
## 20  ( 1 ) "*"     " "            " "     " "      " "  " "
## 21  ( 1 ) "*"     " "            " "     " "      " "  " "
## 22  ( 1 ) "*"     " "            " "     " "      " "  " "
## 23  ( 1 ) "*"     " "            " "     " "      " "  " "
##           parking parking_entrance pharmacy place_of_worship police
## 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 ) " "      " "               " "      " "              " "
## 19  ( 1 ) " "      " "               " "      " "              " "
## 20  ( 1 ) " "      " "               " "      " "              " "
## 21  ( 1 ) " "      " "               " "      " "              " "
## 22  ( 1 ) " "      " "               " "      " "              " "
## 23  ( 1 ) " "      " "               " "      " "              " "
##           post_box post_office recycling school theatre waste_basket
## 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 ) "*"      " "         " "       " "    " "     " "
## 19  ( 1 ) "*"      " "         " "       " "    " "     " "
## 20  ( 1 ) "*"      " "         " "       " "    " "     " "
## 21  ( 1 ) "*"      " "         " "       " "    " "     " "
## 22  ( 1 ) "*"      " "         " "       " "    " "     " "
## 23  ( 1 ) "*"      " "         " "       " "    " "     " "
##           artwork_type_statue diplomatic_embassy emergency_fire_hydrant
## 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 ) "*"                " "                "*"
## 19  ( 1 ) "*"                " "                "*"
## 20  ( 1 ) "*"                " "                "*"
## 21  ( 1 ) "*"                " "                "*"
## 22  ( 1 ) "*"                " "                "*"
## 23  ( 1 ) "*"                " "                "*"
##           bus_stop crossing motorway_junction stop traffic_signals
## 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 ) "*"      " "      " "               "*"  " "
## 19  ( 1 ) "*"      " "      " "               "*"  " "
## 20  ( 1 ) "*"      " "      " "               "*"  " "
## 21  ( 1 ) "*"      " "      " "               "*"  " "
## 22  ( 1 ) "*"      " "      " "               "*"  " "
## 23  ( 1 ) "*"      " "      " "               "*"  " "
##           turning_circle 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 ) " "            " "               " "               " "
## 19  ( 1 ) " "            " "               " "               " "
```

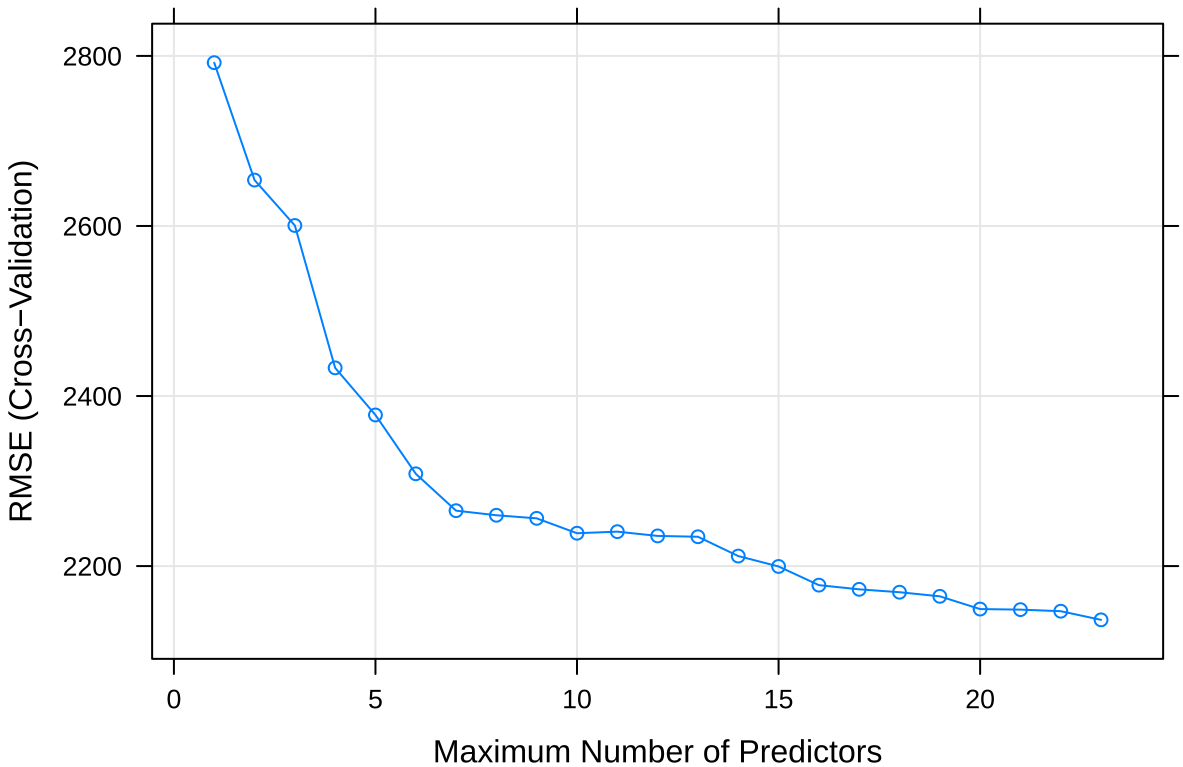
```
## 20  ( 1 ) " "            " "              " "              " "
## 21  ( 1 ) " "            " "              " "              " "
## 22  ( 1 ) " "            " "              " "              " "
## 23  ( 1 ) " "            " "              " "              " "
##           leisure_sports_centre 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 ) " "                   "*"               " "
## 19  ( 1 ) " "                   "*"               " "
## 20  ( 1 ) " "                   "*"               " "
## 21  ( 1 ) " "                   "*"               " "
## 22  ( 1 ) " "                   "*"               " "
## 23  ( 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 ) "*"                 " "                    " "
## 19  ( 1 ) "*"                 " "                    " "
## 20  ( 1 ) "*"                 " "                    " "
## 21  ( 1 ) "*"                 " "                    " "
## 22  ( 1 ) "*"                 " "                    " "
## 23  ( 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 ) " "                    "*"          " "       " "
## 19  ( 1 ) " "                    "*"          " "       " "
## 20  ( 1 ) " "                    "*"          " "       " "
## 21  ( 1 ) " "                    "*"          " "       " "
## 22  ( 1 ) " "                    "*"          "*"       " "
## 23  ( 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 ) " "         " "        " "          " "
## 19  ( 1 ) " "         " "        " "          " "
## 20  ( 1 ) " "         " "        " "          " "
## 21  ( 1 ) " "         " "        " "          " "
## 22  ( 1 ) " "         " "        " "          " "
## 23  ( 1 ) " "         " "        " "          " "
##           shop_dry_cleaning shop_electronics shop_gift shop_hairdresser
## 1  ( 1 )  " "               " "              " "       " "
## 2  ( 1 )  " "               " "              " "       " "
## 3  ( 1 )  " "               " "              " "       " "
## 4  ( 1 )  " "               " "              " "       " "
## 5  ( 1 )  " "               " "              " "       " "
## 6  ( 1 )  " "               " "              " "       " "
## 7  ( 1 )  " "               " "              " "       " "
```

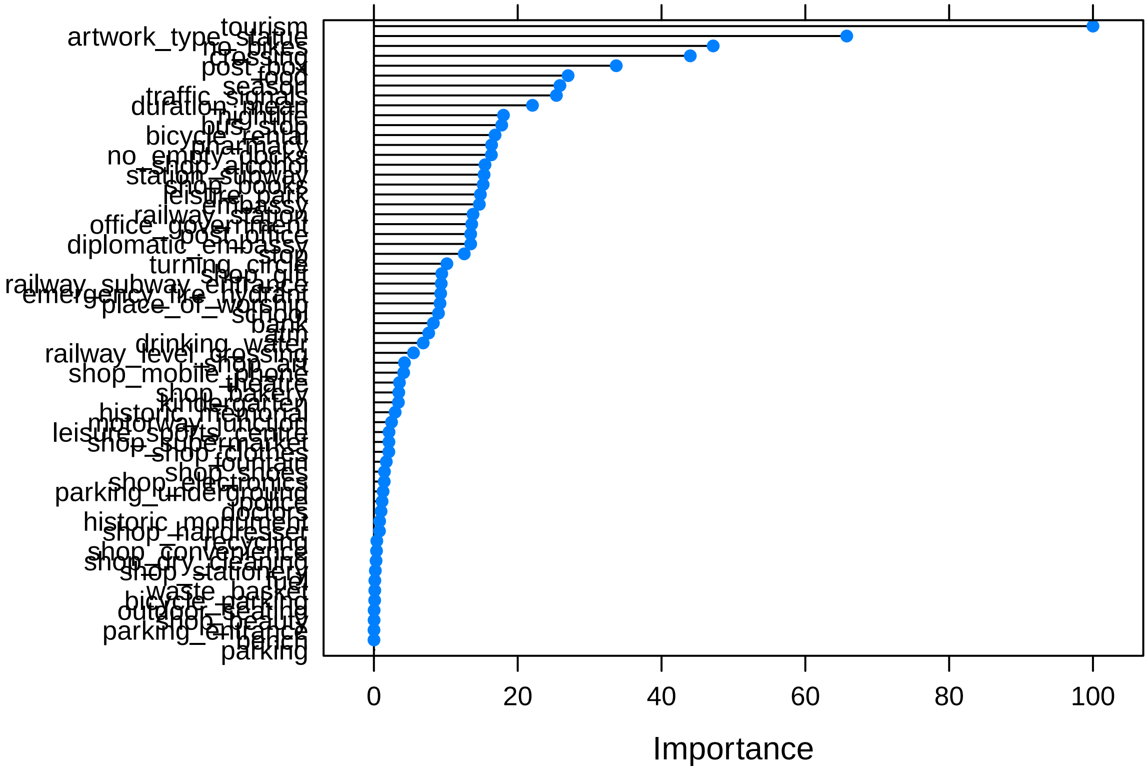
```
## 8  ( 1 )  " "               " "               " "        " "
## 9  ( 1 )  " "               " "               " "        " "
## 10  ( 1 ) " "               " "               " "        " "
## 11  ( 1 ) " "               " "               " "        " "
## 12  ( 1 ) " "               " "               " "        "*"
## 13  ( 1 ) " "               " "               " "        "*"
## 14  ( 1 ) " "               " "               " "        "*"
## 15  ( 1 ) " "               " "               " "        "*"
## 16  ( 1 ) " "               " "               " "        "*"
## 17  ( 1 ) " "               " "               " "        "*"
## 18  ( 1 ) " "               " "               " "        "*"
## 19  ( 1 ) " "               " "               " "        "*"
## 20  ( 1 ) " "               " "               " "        "*"
## 21  ( 1 ) " "               " "               " "        "*"
## 22  ( 1 ) " "               " "               " "        "*"
## 23  ( 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 ) " "               " "        "*"             "*"
## 19  ( 1 ) " "               " "        "*"             "*"
## 20  ( 1 ) " "               " "        "*"             "*"
## 21  ( 1 ) " "               " "        "*"             "*"
## 22  ( 1 ) " "               " "        "*"             "*"
## 23  ( 1 ) "*"               " "        "*"             "*"
##           station_subway 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 ) " "            " "  "*"        "*"
## 19  ( 1 ) " "            " "  "*"        "*"
## 20  ( 1 ) " "            " "  "*"        "*"
## 21  ( 1 ) " "            " "  "*"        "*"
## 22  ( 1 ) " "            " "  "*"        "*"
## 23  ( 1 ) " "            " "  "*"        "*"
```

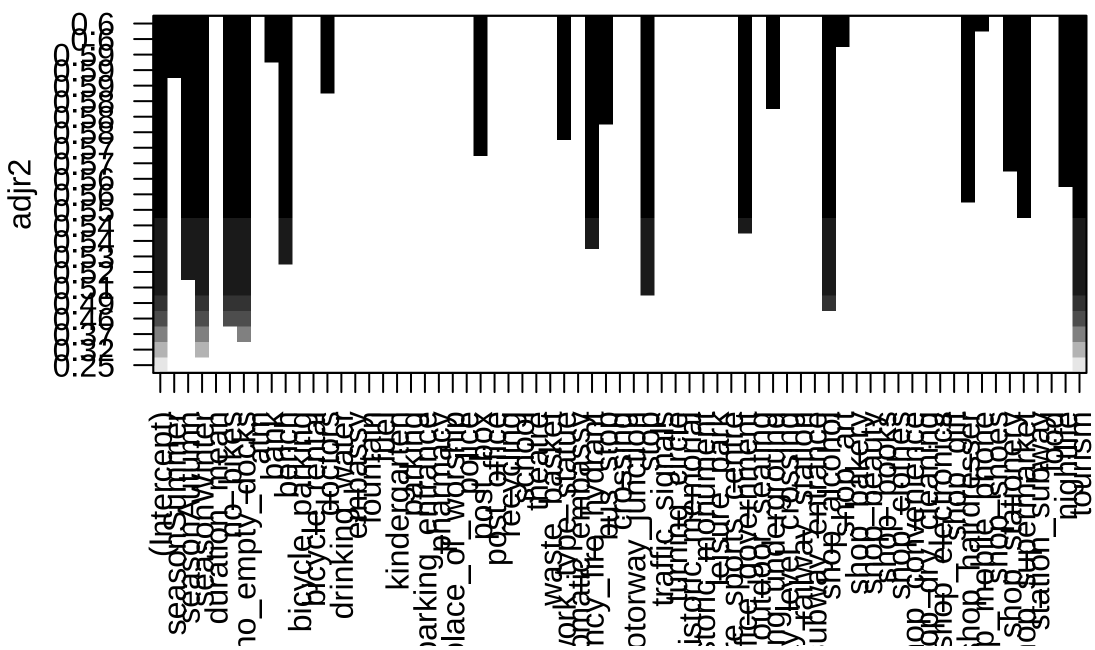
```r
plot(forward_model)
```




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

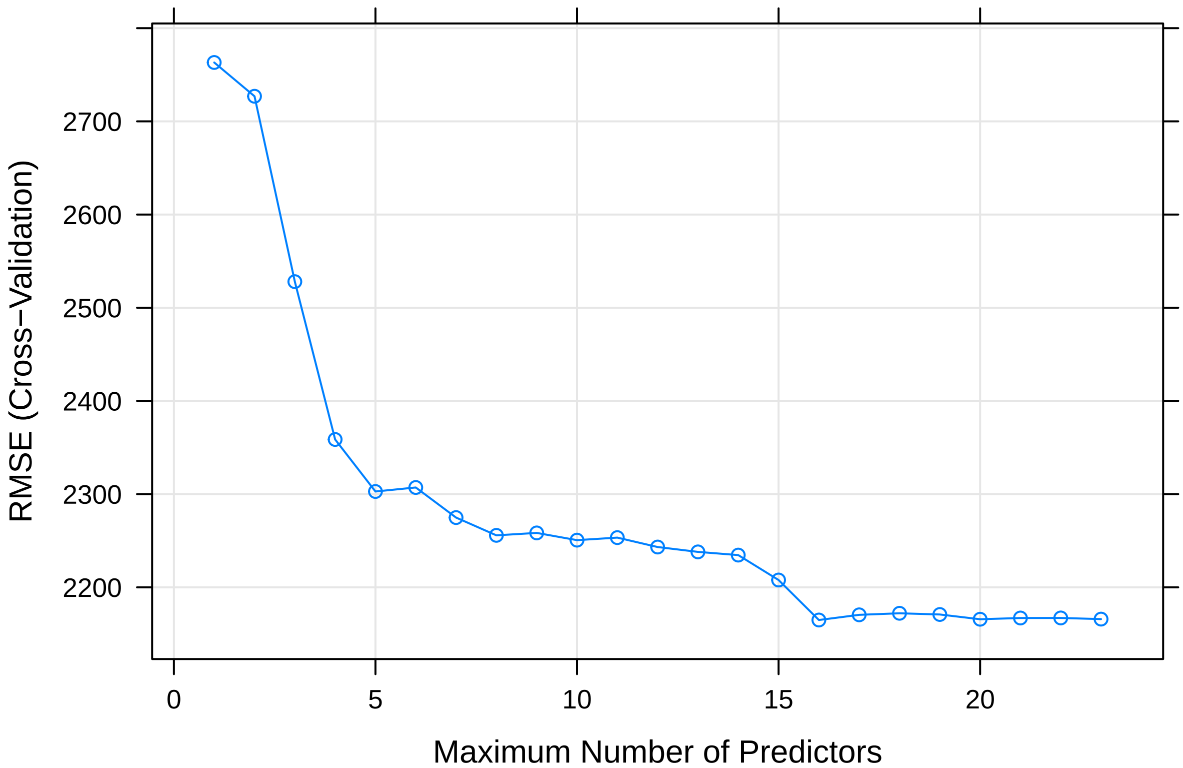



```r
# compare all the models
plot(forward_model$finalModel, scale = 'adjr2')
```

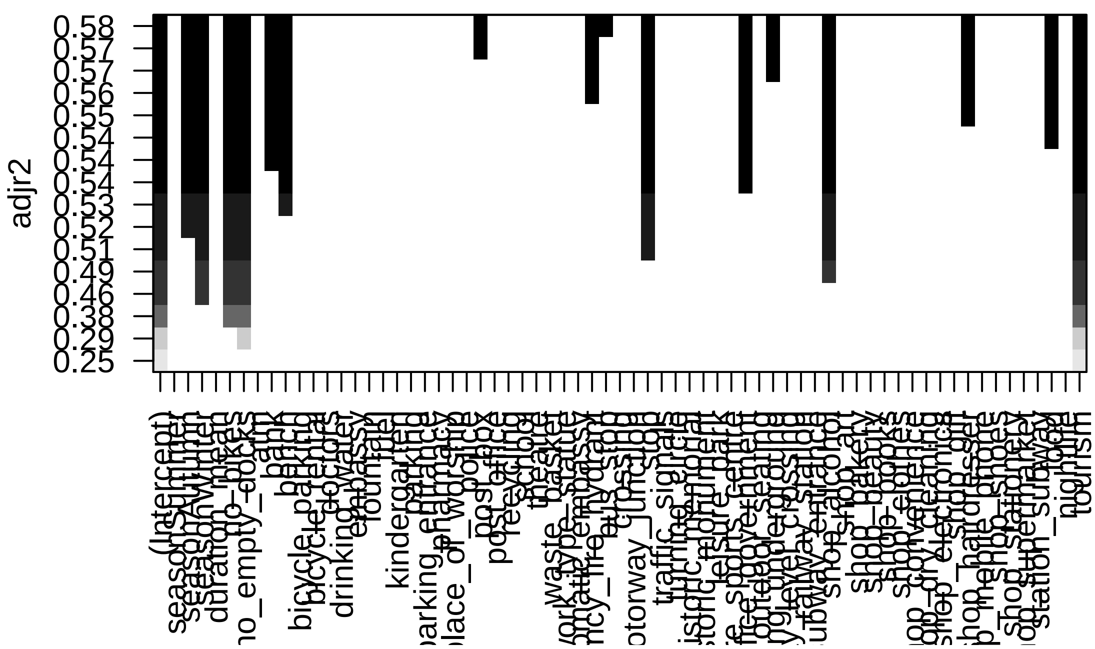



```r
# backward_selection
backward_model = train(num_rentals ~ .,
                       data = train,
                       method = 'leapBackward',
                       preProcess = c('center', 'scale'),
```

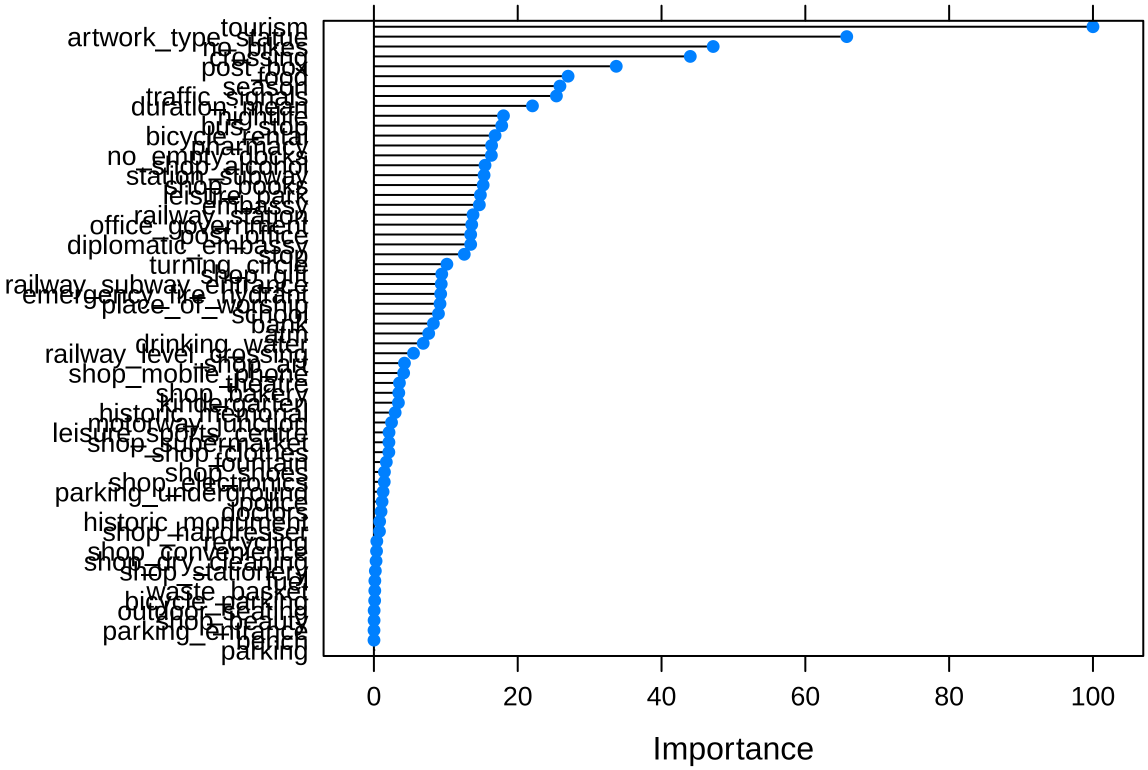
```r
                             tuneGrid = expand.grid(nvmax = 1:23),
                             trControl = trainControl(method = 'cv', number = 5))

plot(backward_model)
```



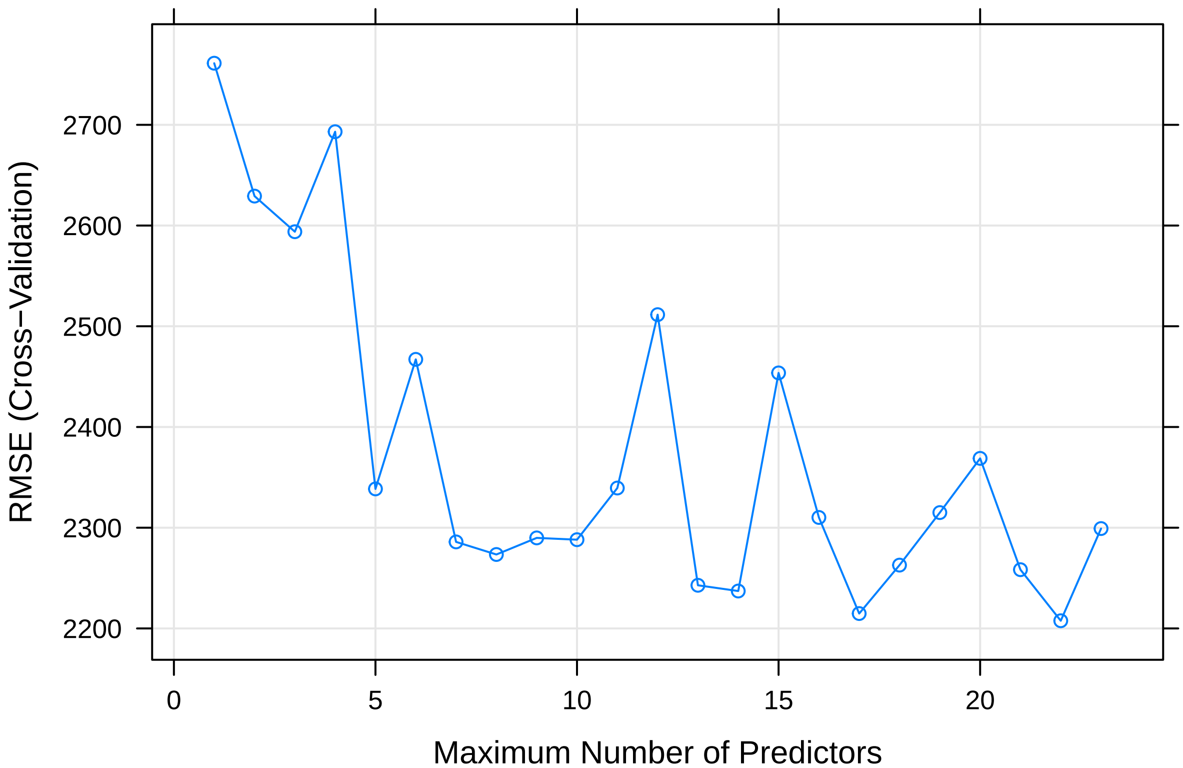

```r
plot(backward_model$finalModel, scale = 'adjr2')
```



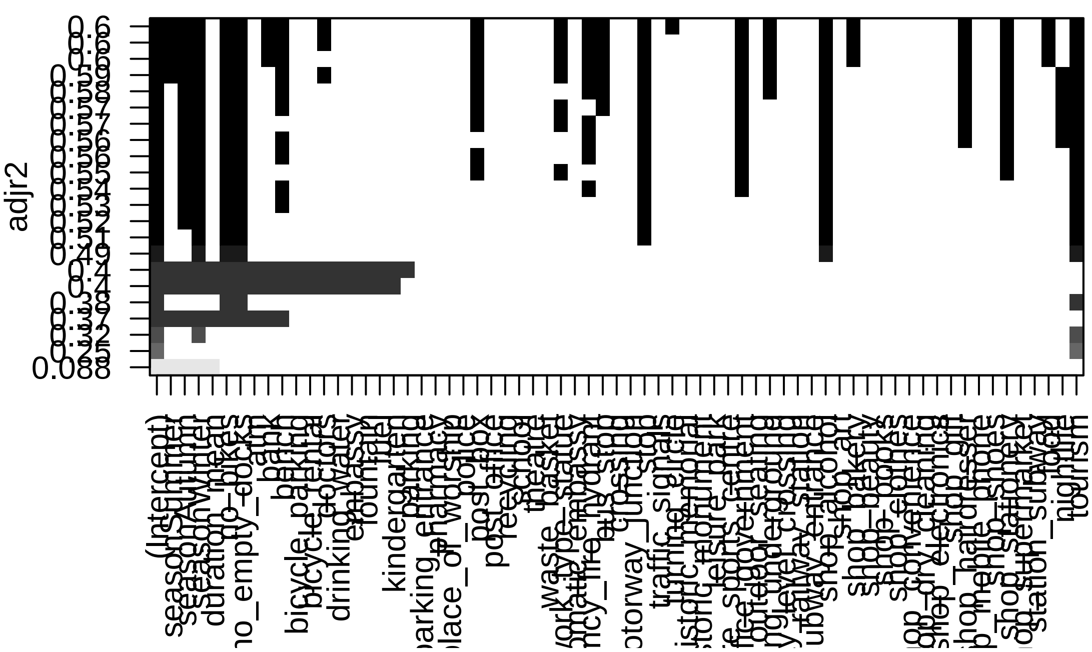
```
plot(varImp(backward_model, scale = TRUE))
```



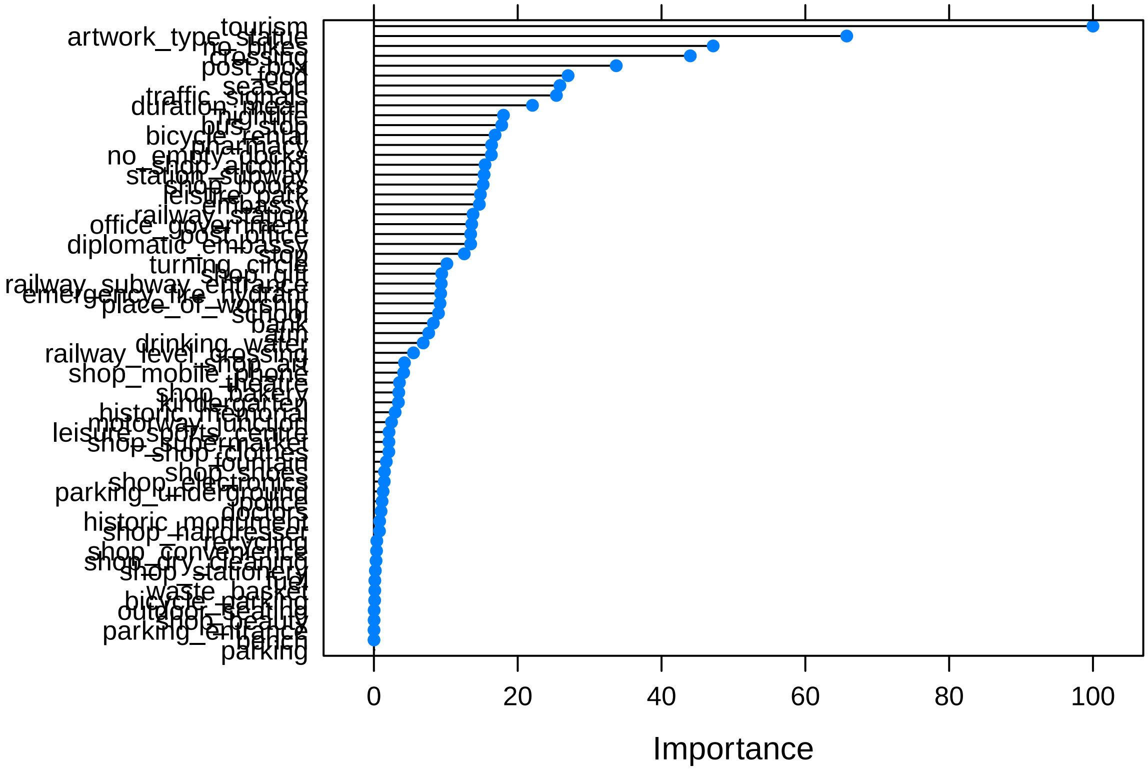

```
# steps in both directions
hybrid_model = train(num_rentals ~ .,
                     data = train,
                     method = 'leapSeq',
                     preProcess = c('center', 'scale'),
                     tuneGrid = expand.grid(nvmax = 1:23),
                     trControl = trainControl(method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

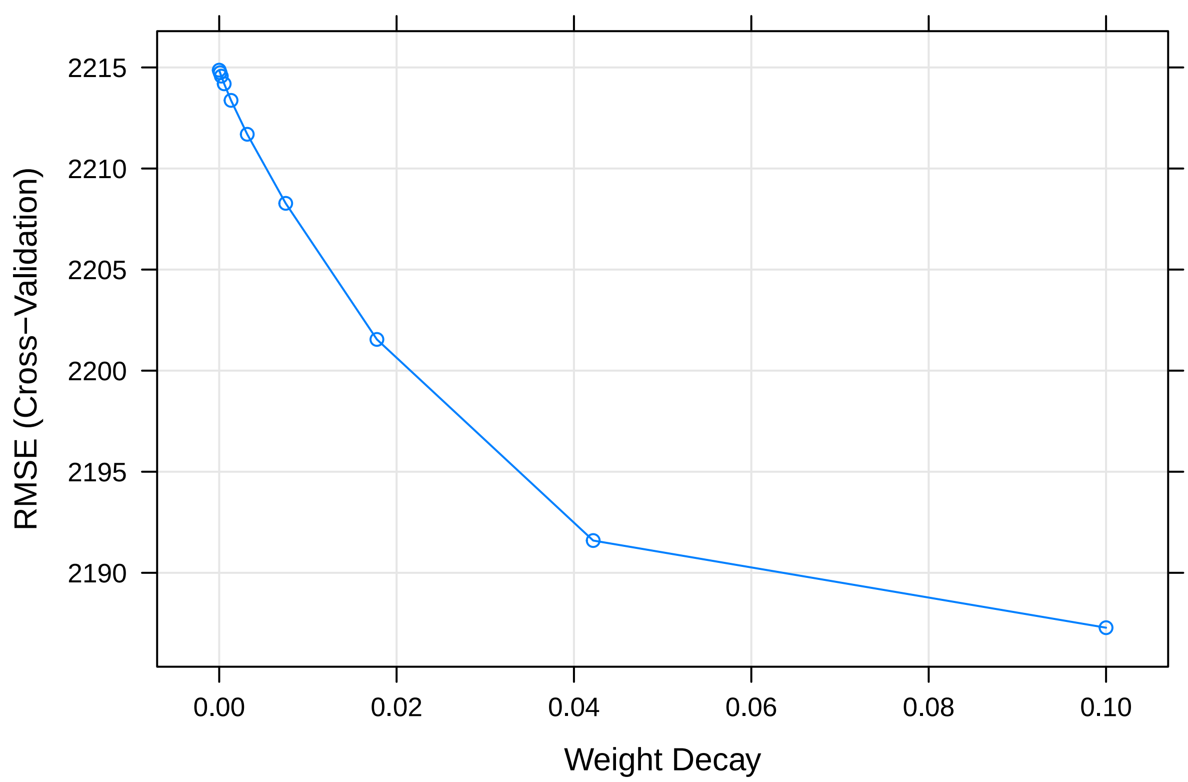
```
## Loaded lars 1.2
```

```r
print(ridge_model)
```

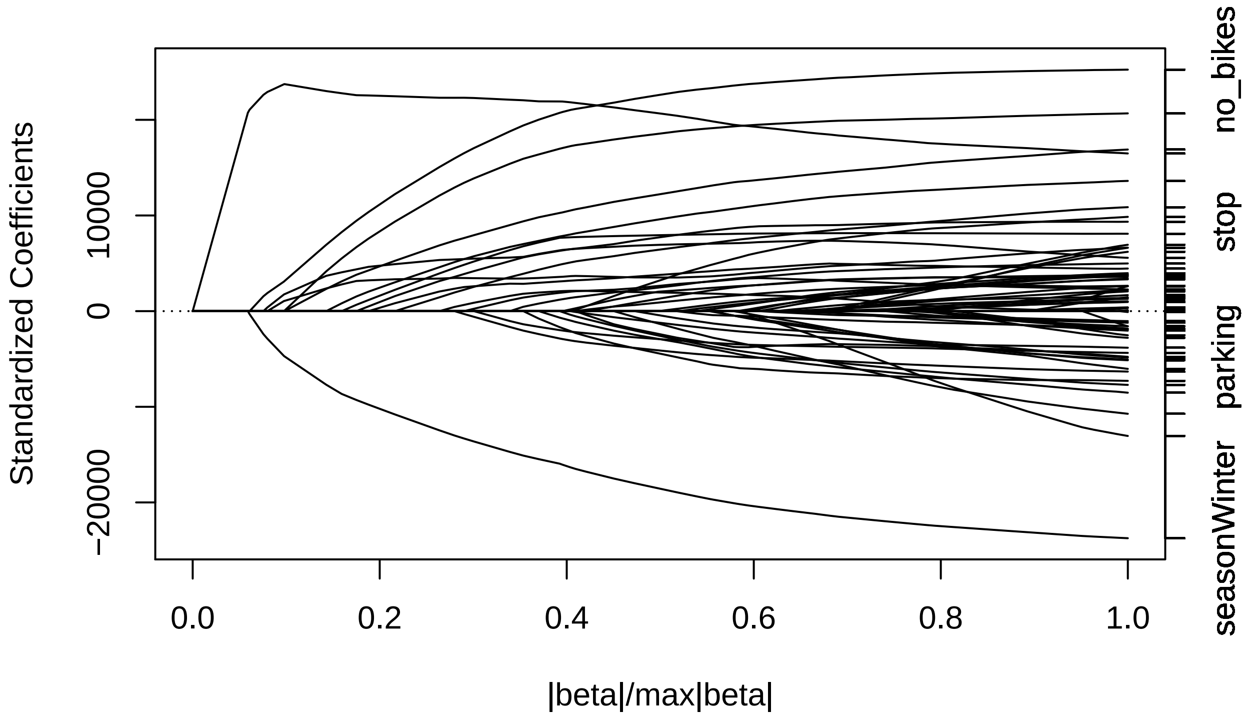
```
## Ridge Regression
##
## 753 samples
##  64 predictor
```

```
##
## Pre-processing: centered (66), scaled (66)
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 602, 602, 603, 602, 603
## Resampling results across tuning parameters:
##
##   lambda        RMSE      Rsquared
##   0.0000000000  2214.865  0.5512942
##   0.0001000000  2214.738  0.5513465
##   0.0002371374  2214.570  0.5514161
##   0.0005623413  2214.189  0.5515732
##   0.0013335214  2213.370  0.5519102
##   0.0031622777  2211.693  0.5525912
##   0.0074989421  2208.277  0.5539373
##   0.0177827941  2201.544  0.5565410
##   0.0421696503  2191.596  0.5605930
##   0.1000000000  2187.284  0.5638092
##
## RMSE was used to select the optimal model using  the smallest value.
## The final value used for the model was lambda = 0.1.
```

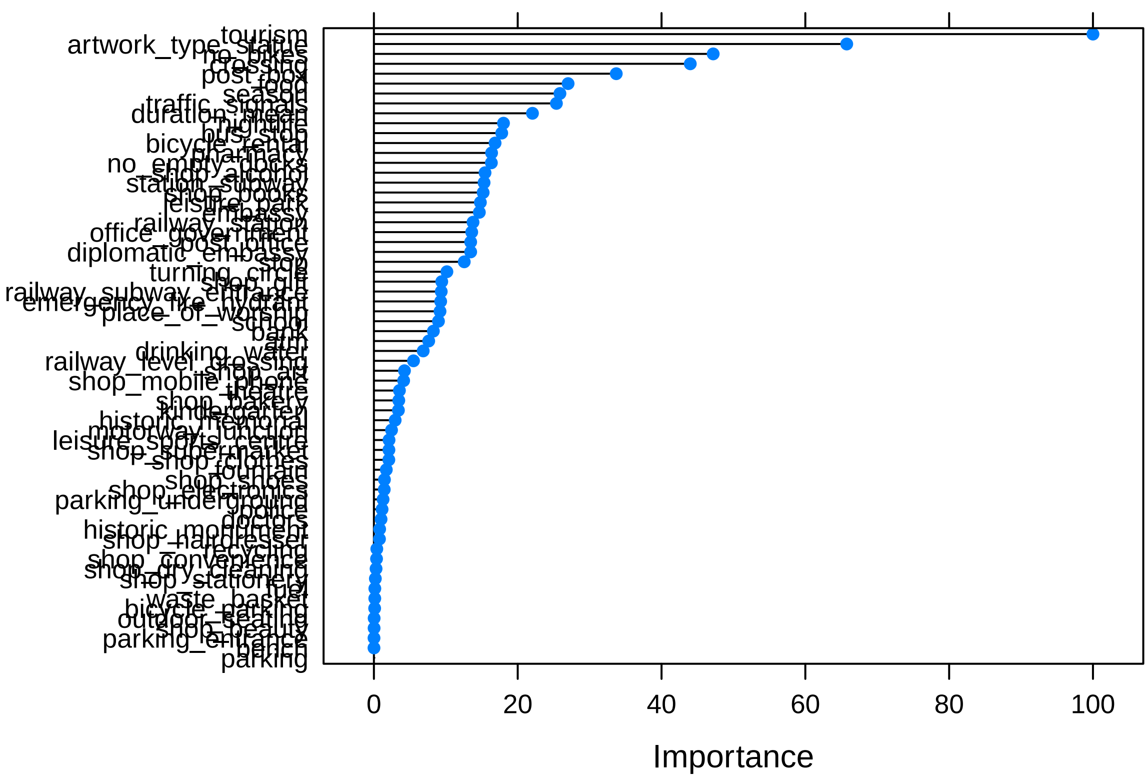
```r
plot(ridge_model)
```




```r
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
```

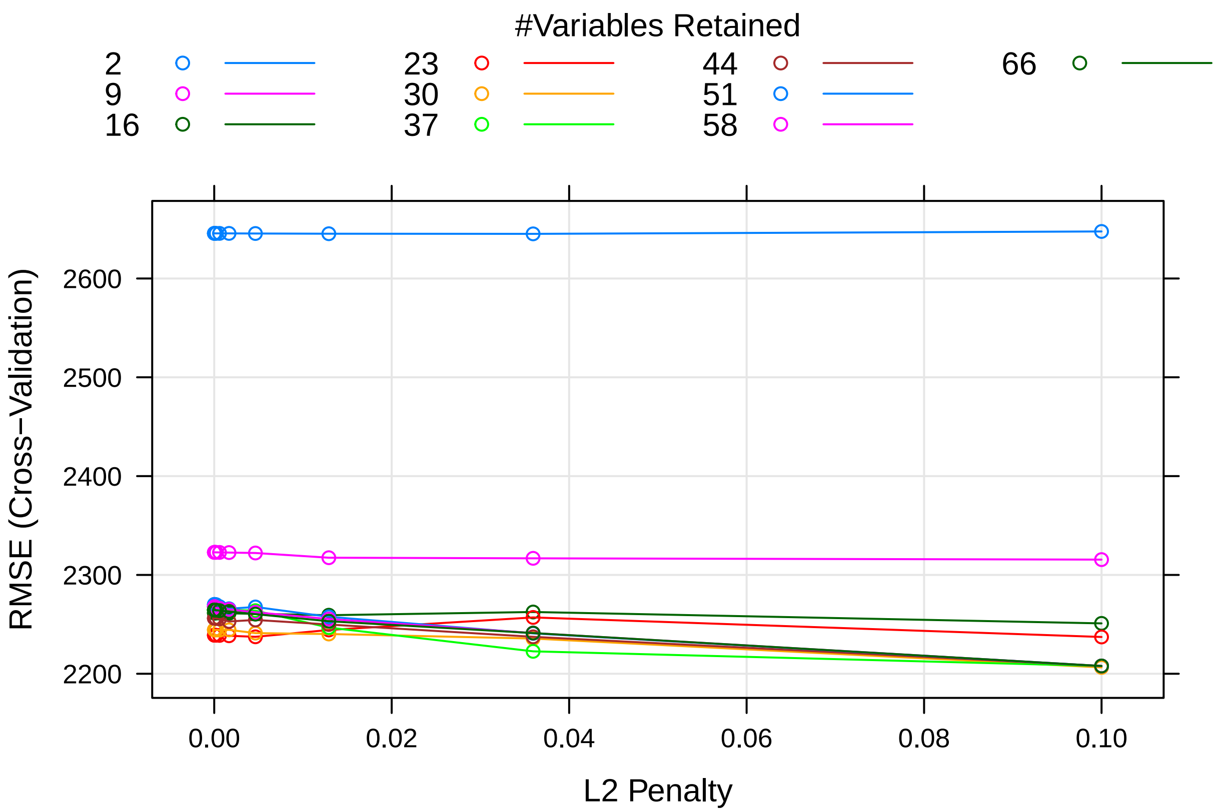
```r
print(ridge_model2)
```

```
## Ridge Regression with Variable Selection
##
## 753 samples
##  64 predictor
##
## Pre-processing: centered (66), scaled (66)
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 601, 603, 604, 602, 602
## Resampling results across tuning parameters:
##
##   lambda        k   RMSE      Rsquared
##   1.000000e-05   2  2645.671  0.3338878
##   1.000000e-05   9  2322.888  0.4969939
##   1.000000e-05  16  2261.485  0.5316621
##   1.000000e-05  23  2239.021  0.5449244
##   1.000000e-05  30  2244.435  0.5435895
##   1.000000e-05  37  2264.603  0.5394048
##   1.000000e-05  44  2256.158  0.5450384
##   1.000000e-05  51  2270.167  0.5420337
##   1.000000e-05  58  2267.821  0.5430421
##   1.000000e-05  66  2264.553  0.5440262
##   2.782559e-05   2  2645.671  0.3338878
##   2.782559e-05   9  2322.885  0.4969941
##   2.782559e-05  16  2261.481  0.5316618
##   2.782559e-05  23  2239.015  0.5449250
##   2.782559e-05  30  2244.425  0.5435908
##   2.782559e-05  37  2264.588  0.5394072
##   2.782559e-05  44  2256.138  0.5450423
##   2.782559e-05  51  2270.145  0.5420377
##   2.782559e-05  58  2267.802  0.5430441
##   2.782559e-05  66  2264.532  0.5440288
##   7.742637e-05   2  2645.669  0.3338878
##   7.742637e-05   9  2322.877  0.4969947
##   7.742637e-05  16  2261.472  0.5316610
##   7.742637e-05  23  2238.997  0.5449266
##   7.742637e-05  30  2244.398  0.5435944
##   7.742637e-05  37  2264.546  0.5394139
##   7.742637e-05  44  2256.082  0.5450529
##   7.742637e-05  51  2270.084  0.5420491
##   7.742637e-05  58  2267.749  0.5430495
```

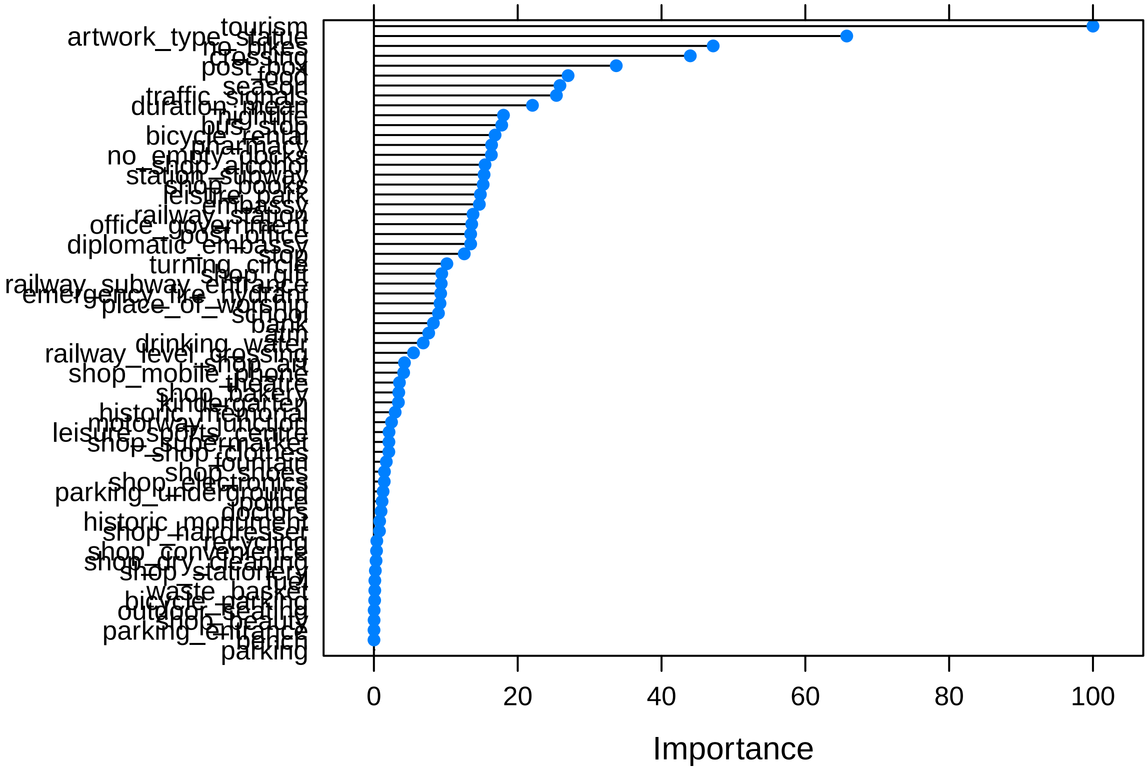
```
## 7.742637e-05 66 2264.473 0.5440361
## 2.154435e-04  2 2645.664 0.3338878
## 2.154435e-04  9 2322.855 0.4969962
## 2.154435e-04 16 2261.445 0.5316588
## 2.154435e-04 23 2238.946 0.5449311
## 2.154435e-04 30 2244.323 0.5436043
## 2.154435e-04 37 2264.431 0.5394326
## 2.154435e-04 44 2255.927 0.5450824
## 2.154435e-04 51 2269.914 0.5420805
## 2.154435e-04 58 2266.627 0.5433335
## 2.154435e-04 66 2264.312 0.5440559
## 5.994843e-04  2 2645.650 0.3338878
## 5.994843e-04  9 2322.795 0.4970004
## 5.994843e-04 16 2261.370 0.5316525
## 5.994843e-04 23 2238.807 0.5449435
## 5.994843e-04 30 2244.115 0.5436318
## 5.994843e-04 37 2264.111 0.5394842
## 5.994843e-04 44 2255.498 0.5451638
## 5.994843e-04 51 2267.879 0.5423687
## 5.994843e-04 58 2266.202 0.5433843
## 5.994843e-04 66 2263.885 0.5441067
## 1.668101e-03  2 2645.612 0.3338877
## 1.668101e-03  9 2322.631 0.4970120
## 1.668101e-03 16 2261.166 0.5316349
## 1.668101e-03 23 2238.423 0.5449775
## 1.668101e-03 30 2244.427 0.5436669
## 1.668101e-03 37 2264.667 0.5391718
## 1.668101e-03 44 2252.904 0.5459261
## 1.668101e-03 51 2265.760 0.5429067
## 1.668101e-03 58 2265.106 0.5435132
## 1.668101e-03 66 2262.988 0.5441520
## 4.641589e-03  2 2645.515 0.3338877
## 4.641589e-03  9 2322.193 0.4970426
## 4.641589e-03 16 2260.618 0.5315843
## 4.641589e-03 23 2237.388 0.5450674
## 4.641589e-03 30 2241.123 0.5446593
## 4.641589e-03 37 2263.836 0.5394392
## 4.641589e-03 44 2254.439 0.5455554
## 4.641589e-03 51 2267.584 0.5415980
## 4.641589e-03 58 2262.213 0.5438697
## 4.641589e-03 66 2260.314 0.5444660
## 1.291550e-02  2 2645.305 0.3338875
## 1.291550e-02  9 2317.435 0.4984124
## 1.291550e-02 16 2259.253 0.5314318
## 1.291550e-02 23 2244.292 0.5412263
## 1.291550e-02 30 2240.203 0.5425428
## 1.291550e-02 37 2246.710 0.5451318
## 1.291550e-02 44 2249.850 0.5457491
## 1.291550e-02 51 2257.636 0.5438645
## 1.291550e-02 58 2255.376 0.5447947
## 1.291550e-02 66 2253.150 0.5455092
## 3.593814e-02  2 2645.161 0.3338870
## 3.593814e-02  9 2316.799 0.4978885
## 3.593814e-02 16 2262.498 0.5289563
```

```
##   3.593814e-02  23  2256.984  0.5350388
##   3.593814e-02  30  2235.621  0.5427647
##   3.593814e-02  37  2222.680  0.5513630
##   3.593814e-02  44  2237.375  0.5472937
##   3.593814e-02  51  2240.862  0.5456792
##   3.593814e-02  58  2241.191  0.5455398
##   3.593814e-02  66  2241.191  0.5455398
##   1.000000e-01   2  2647.600  0.3338857
##   1.000000e-01   9  2315.517  0.4997552
##   1.000000e-01  16  2250.978  0.5288298
##   1.000000e-01  23  2237.221  0.5405829
##   1.000000e-01  30  2206.454  0.5521106
##   1.000000e-01  37  2207.847  0.5513755
##   1.000000e-01  44  2207.847  0.5513755
##   1.000000e-01  51  2207.847  0.5513755
##   1.000000e-01  58  2207.847  0.5513755
##   1.000000e-01  66  2207.847  0.5513755
##
## RMSE was used to select the optimal model using  the smallest value.
## The final values used for the model were k = 30 and lambda = 0.1.
```

```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.

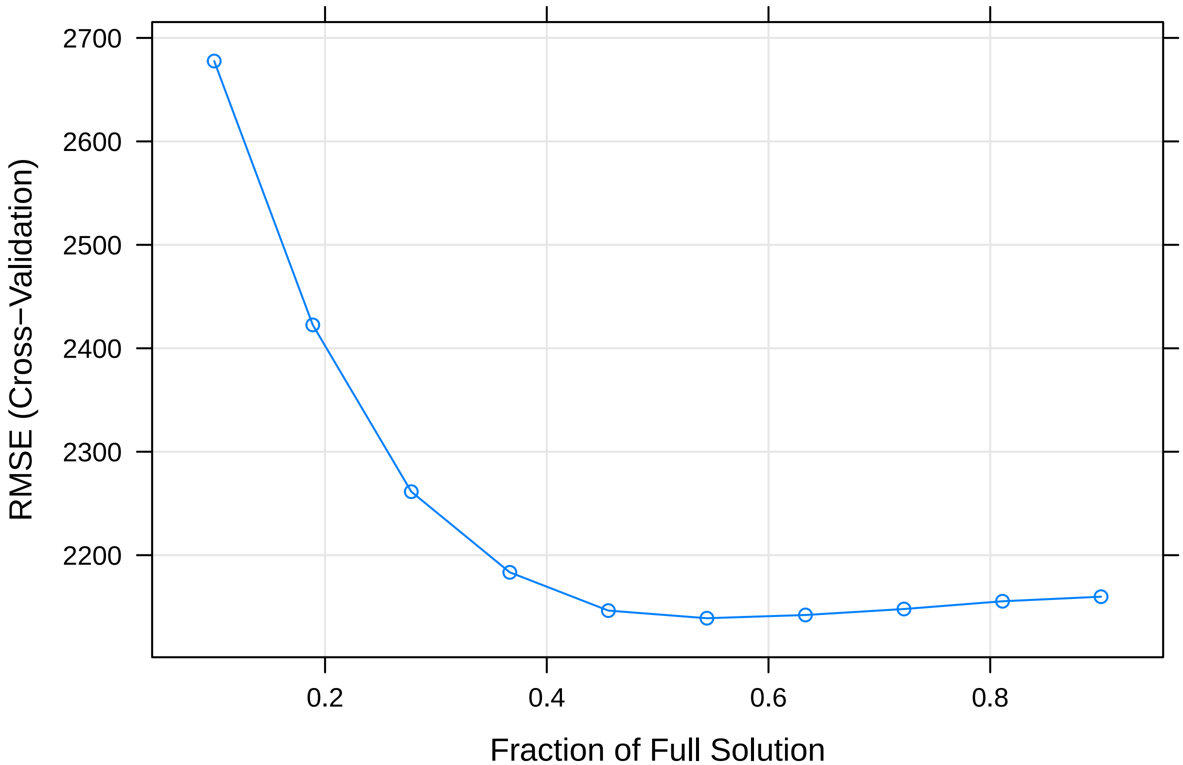
Lasso

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

print(lasso_model)
```

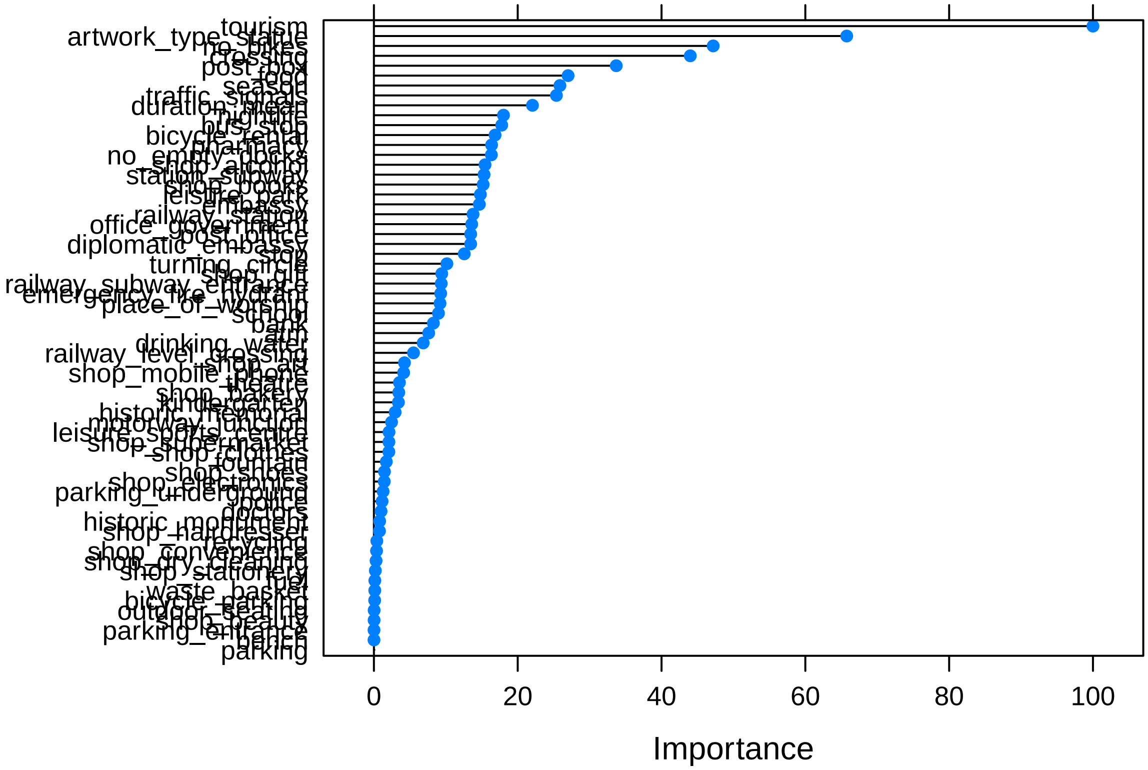
```
## The lasso
##
## 753 samples
##  64 predictor
##
## Pre-processing: scaled (66), centered (66)
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 602, 602, 603, 602, 603
## Resampling results across tuning parameters:
##
##   fraction   RMSE      Rsquared
##   0.1000000  2677.648  0.3832204
##   0.1888889  2422.590  0.4817487
```

```
##    0.2777778  2261.370  0.5256737
##    0.3666667  2183.387  0.5480690
##    0.4555556  2146.424  0.5602249
##    0.5444444  2139.059  0.5635707
##    0.6333333  2142.178  0.5637707
##    0.7222222  2147.950  0.5636487
##    0.8111111  2155.467  0.5623881
##    0.9000000  2159.869  0.5614911
##
## RMSE was used to select the optimal model using  the smallest value.
## The final value used for the model was fraction = 0.5444444.
```

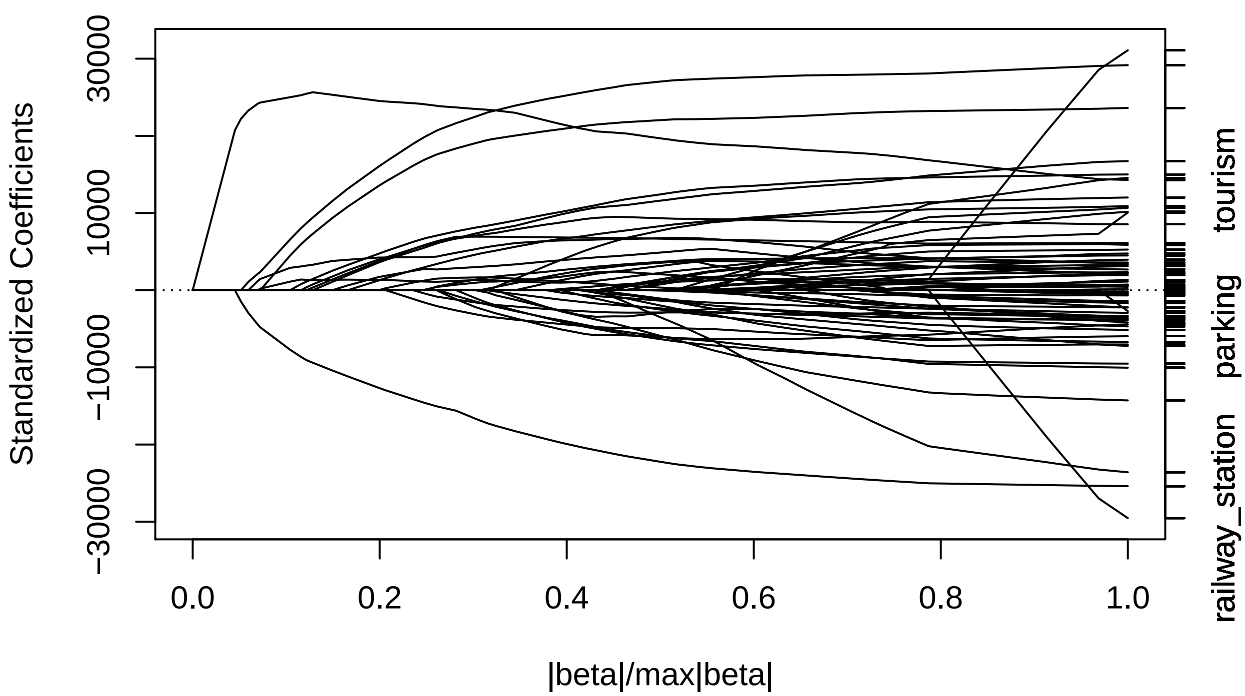
```r
plot(lasso_model)
```




```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
```

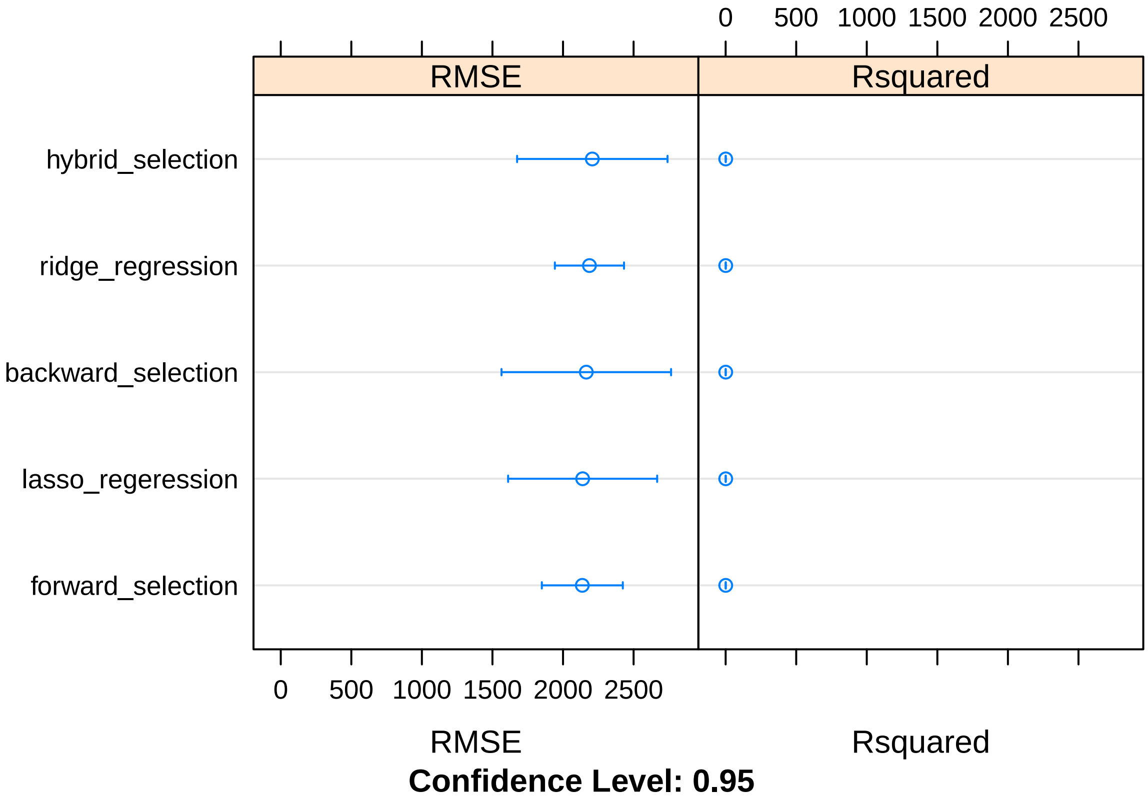
Measuring predictive accuracy

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

```r
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  1800    1991   2274 2137    2297 2322    0
## backward_selection 1748    1761   1940 2165    2660 2716    0
## hybrid_selection   1747    2027   2061 2208    2316 2885    0
## ridge_regression   1846    2198   2263 2187    2295 2335    0
## lasso_regeression  1731    1910   2063 2139    2147 2845    0
##
## Rsquared
##                      Min. 1st Qu. Median   Mean 3rd Qu.   Max. NA's
## forward_selection  0.4875  0.5082 0.5175 0.5604  0.6420 0.6470    0
## backward_selection 0.4445  0.5848 0.5916 0.5659  0.5946 0.6138    0
## hybrid_selection   0.5012  0.5126 0.5322 0.5380  0.5562 0.5879    0
## ridge_regression   0.5117  0.5131 0.5552 0.5638  0.5951 0.6439    0
## lasso_regeression  0.4630  0.5273 0.5642 0.5636  0.6202 0.6432    0
```

```r
# 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