Cross Validation

Introduction

The essence of prediction is discovering the extent to which our models can predict outcomes for data that does not come from our sample. Many times this process is temporal. We fit a model to data from one time period, then take predictors from a subsequent time period to come up with a prediction in the future. For instance, we might use data on team performance to predict the likely winners and losers for upcoming soccer games.

This process does not have to be temporal. We can also have data that is out of sample because it hadn't yet been collected when our first data was collected, or we can also have data that is out of sample because we designated it as out of sample.

The data that is used to generate our predictions is known as *training* data. The idea is that this is the data used to train our model, to let it know what the relationship is between our predictors and our outcome. So far, we have worked mostly with training data.

That data that is used to validate our predictions is known as *testing* data. With testing data, we take our trained model and see how good it is at predicting outcomes using out of sample data.

One very simple approach to this would be to cut our data in half. This is what we've done so far. We could then train our model on half the data, then test it on the other half. This would tell us whether our measure of model fit (e.g. rmse, auc) is similar or different when we apply our model to out of sample data. But this would only be a "one-shot" approach. It would be better to do this multiple times, cutting the data into two parts: training and testing, then fitting the model to the training data, and then checking its predictions against the testing data. That way, we could generate a large number of rmse's to see how well the model fits on lots of different possible out-of-sample predictions.

This process is called *cross-fold validation*, and it involves two important decisions: first, how will the data be cut, and how many times will the validation run.

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 3.5.3

## -- Attaching packages -------------------------------------------------------------------------- tidyve

## v ggplot2 3.2.1     v purrr   0.3.2
## v tibble  2.1.3     v dplyr   0.8.1
## v tidyr   0.8.3     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0

## Warning: package 'ggplot2' was built under R version 3.5.3

## Warning: package 'tibble' was built under R version 3.5.3

## Warning: package 'tidyr' was built under R version 3.5.3

## Warning: package 'purrr' was built under R version 3.5.3

## Warning: package 'dplyr' was built under R version 3.5.3

## Warning: package 'stringr' was built under R version 3.5.3

## Warning: package 'forcats' was built under R version 3.5.3

## -- Conflicts ----------------------------------------------------------------------------- tidyverse_co
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
library(modelr)
```

```
## Warning: package 'modelr' was built under R version 3.5.3
```

```r
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.5.3
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

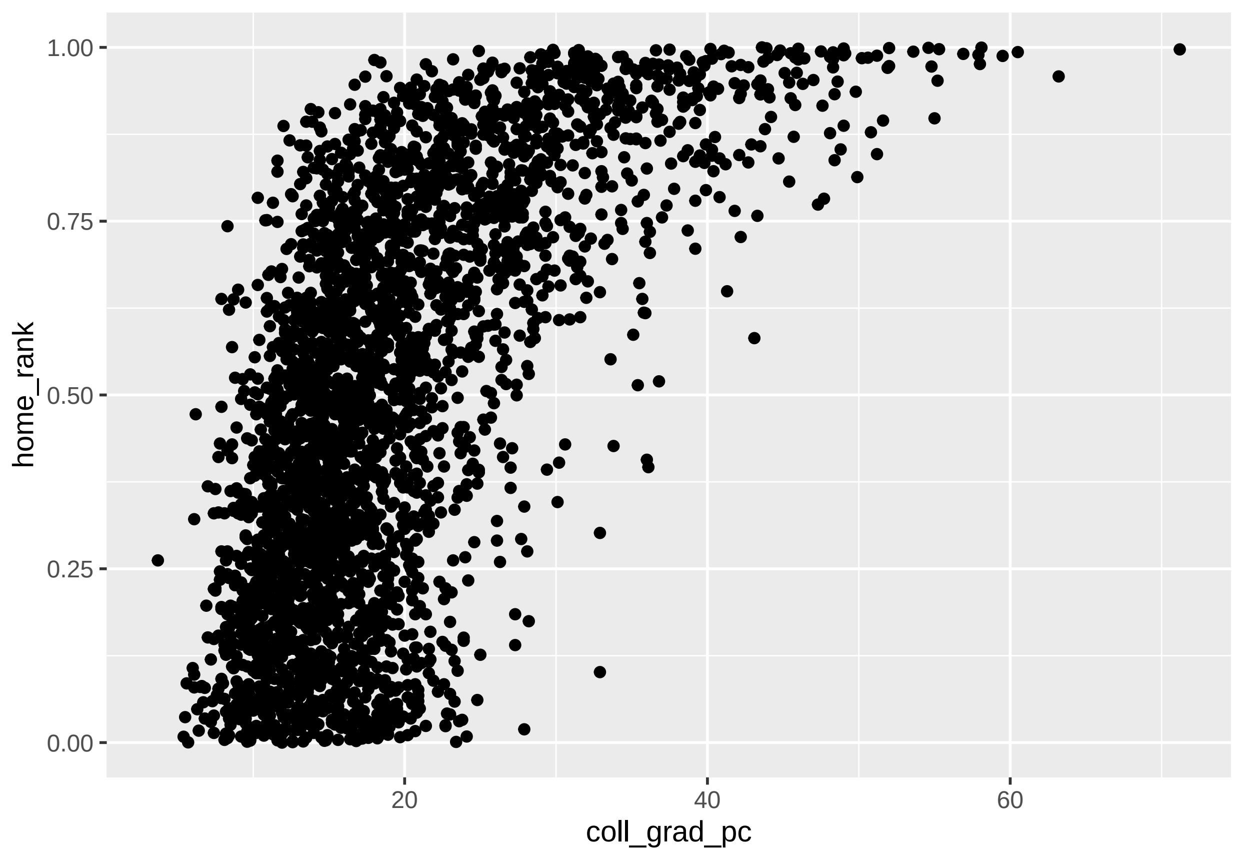
```r
library(tictoc)
```

Next we load the quickfacts data, which contains county-by-county information. We're going to create a simple model that predicts median home values in the county as a function of education, home ownership and income.

```r
load("pd.Rdata")
pd<-pd%>%
  select(median_home_val,median_hh_inc,coll_grad_pc,homeown_rate,per_capita_inc)%>%
  mutate_all(funs(as.numeric))%>%
  mutate(home_rank=percent_rank(median_home_val))%>%
  tbl_df()
```

```
## Warning: funs() is soft deprecated as of dplyr 0.8.0
## please use list() instead
##
##   # Before:
##   funs(name = f(.))
##
##   # After:
##   list(name = ~ f(.))
## This warning is displayed once per session.
```

```r
gg<-ggplot(pd, aes(x=coll_grad_pc,y=home_rank))
gg<-gg+geom_point()
gg
```



We can run this model on the full dataset, but we're not taking advantage of the idea of cross-validation.

```
## Define the model
mod1_formula<-formula(home_rank~coll_grad_pc+
                        per_capita_inc)
## Run the model against all of the data
basic.mod<-lm(mod1_formula,
              data=pd); summary(basic.mod)
```

```
##
## Call:
## lm(formula = mod1_formula, data = pd)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.81893 -0.15064  0.01061  0.15740  0.58125
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -1.298e-01  1.837e-02  -7.066 1.96e-12 ***
## coll_grad_pc    1.431e-02  7.278e-04  19.661  < 2e-16 ***
## per_capita_inc  1.512e-05  1.149e-06  13.156  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2141 on 3085 degrees of freedom
## Multiple R-squared:  0.4513, Adjusted R-squared:  0.451
## F-statistic:  1269 on 2 and 3085 DF,  p-value: < 2.2e-16
```

The `crossv_kfold` command creates a list of datasets from our original dataset, each of which contains a testing and training dataset. The proportion of cases held out for testing is determined by the number of folds: 10 folds would indicate 1/10 of the data to be held out.

```
pd_cf<-pd%>%
  crossv_kfold(20)
pd_cf
```

```
## # A tibble: 20 x 3
##    train        test         .id  
##    <named list> <named list> <chr>
##  1 <resample>   <resample>   01   
##  2 <resample>   <resample>   02   
##  3 <resample>   <resample>   03   
##  4 <resample>   <resample>   04   
##  5 <resample>   <resample>   05   
##  6 <resample>   <resample>   06   
##  7 <resample>   <resample>   07   
##  8 <resample>   <resample>   08   
##  9 <resample>   <resample>   09   
## 10 <resample>   <resample>   10   
## 11 <resample>   <resample>   11   
## 12 <resample>   <resample>   12   
## 13 <resample>   <resample>   13   
## 14 <resample>   <resample>   14   
## 15 <resample>   <resample>   15   
## 16 <resample>   <resample>   16   
## 17 <resample>   <resample>   17   
## 18 <resample>   <resample>   18   
## 19 <resample>   <resample>   19   
## 20 <resample>   <resample>   20
```

The `pd_cf` dataset is now a nested dataset, as described in (Chapter 25)[http://r4ds.had.co.nz/many-models.html] of the Wickham r4ds book.

The next bit of code is key. It starts by converting all of the individual training datasets to tibbles. Then the model is run on each training dataset. Then apply the predictions from the model to each testing dataset, and finally pull the rmse from each of the testing datasets.

```
tic()
rmse_mod1<-pd_cf %>%
  mutate(train = map(train, as_tibble)) %>% ## Convert to tibbles
  mutate(model = map(train, ~ lm(mod1_formula,
                                 data = .))) %>%
  mutate(rmse = map2_dbl(model, test, rmse)) %>% ## apply model, get rmse
  select(.id, rmse) ## pull just id and rmse
toc()
```

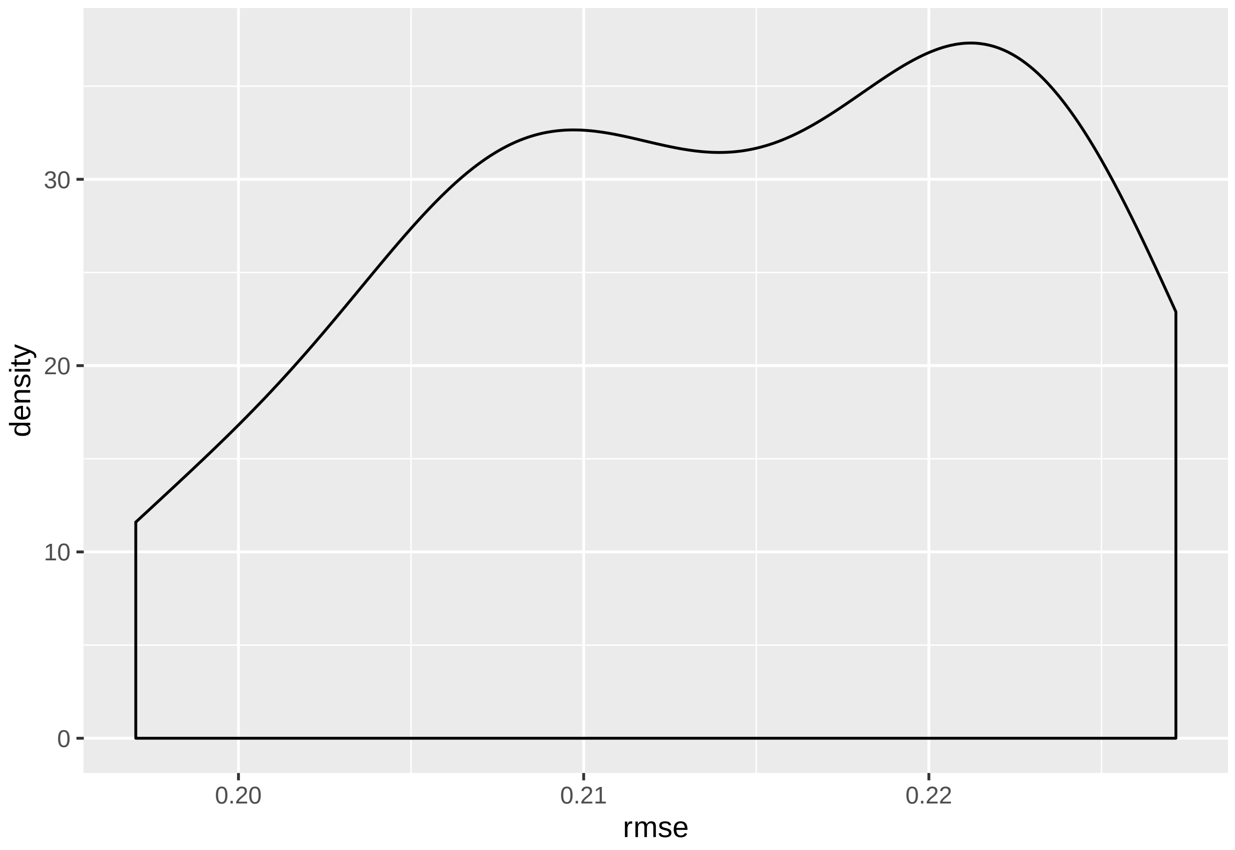
```
## 0.07 sec elapsed
```

The resulting dataset includes the id for the cross validation and the rmse. We can summarize and plot this new data frame to see what our likely range of rmse happens to be.

```
summary(rmse_mod1$rmse)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.1970  0.2076  0.2144  0.2140  0.2227  0.2272
```

```
gg<-ggplot(rmse_mod1,aes(rmse))
gg<-gg+geom_density()
gg
```



As this shows, the rmse for the crossfold validations goes from a minimum of 0.2 to a maximum of 0.23, with a median of 0.21.

Quick Exercise Run the crossfold command again, but this time only 5 times. Then run it again, but 20 times. What happens to the RMSE?

Full Cross Validation: Random Partition

The `crossv_mc` command provides for a generalization of the crossfold command. For this command, we can specify the proportion to be randomly held out in each iteration, via `test=p` where `p` is the proportion to be held out.

```
pd_cv<-pd%>%
  crossv_mc(n=1000,test=.2)
pd_cv
```

```
## # A tibble: 1,000 x 3
##    train      test       .id
##    <list>     <list>     <chr>
##  1 <resample> <resample> 0001
##  2 <resample> <resample> 0002
##  3 <resample> <resample> 0003
##  4 <resample> <resample> 0004
##  5 <resample> <resample> 0005
##  6 <resample> <resample> 0006
##  7 <resample> <resample> 0007
##  8 <resample> <resample> 0008
##  9 <resample> <resample> 0009
## 10 <resample> <resample> 0010
## # ... with 990 more rows
```

The pd_cv dataset is a dataset of 1000x2 datasets, with each row containing a training and testing dataset. The testing dataset is .2 of the sample, but it's different each time.

Now we use the same approach, but with the MUCH larger qf_cv dataset. This will take a bit of time.

```
tic()
mod1_rmse_cv<-pd_cv %>%
  mutate(train = map(train, as_tibble)) %>% ## Convert to tibbles
  mutate(model = map(train, ~ lm(mod1_formula, data = .)))%>%
  mutate(rmse = map2_dbl(model, test, rmse))%>%
  select(.id, rmse) ## pull just id and rmse

mod1_rmse_cv
```

```
## # A tibble: 1,000 x 2
##    .id    rmse
##    <chr> <dbl>
##  1 0001  0.216
##  2 0002  0.208
##  3 0003  0.207
##  4 0004  0.217
##  5 0005  0.206
##  6 0006  0.226
##  7 0007  0.214
##  8 0008  0.223
##  9 0009  0.222
## 10 0010  0.222
## # ... with 990 more rows
```

```
toc()
```

```
## 3.34 sec elapsed
```

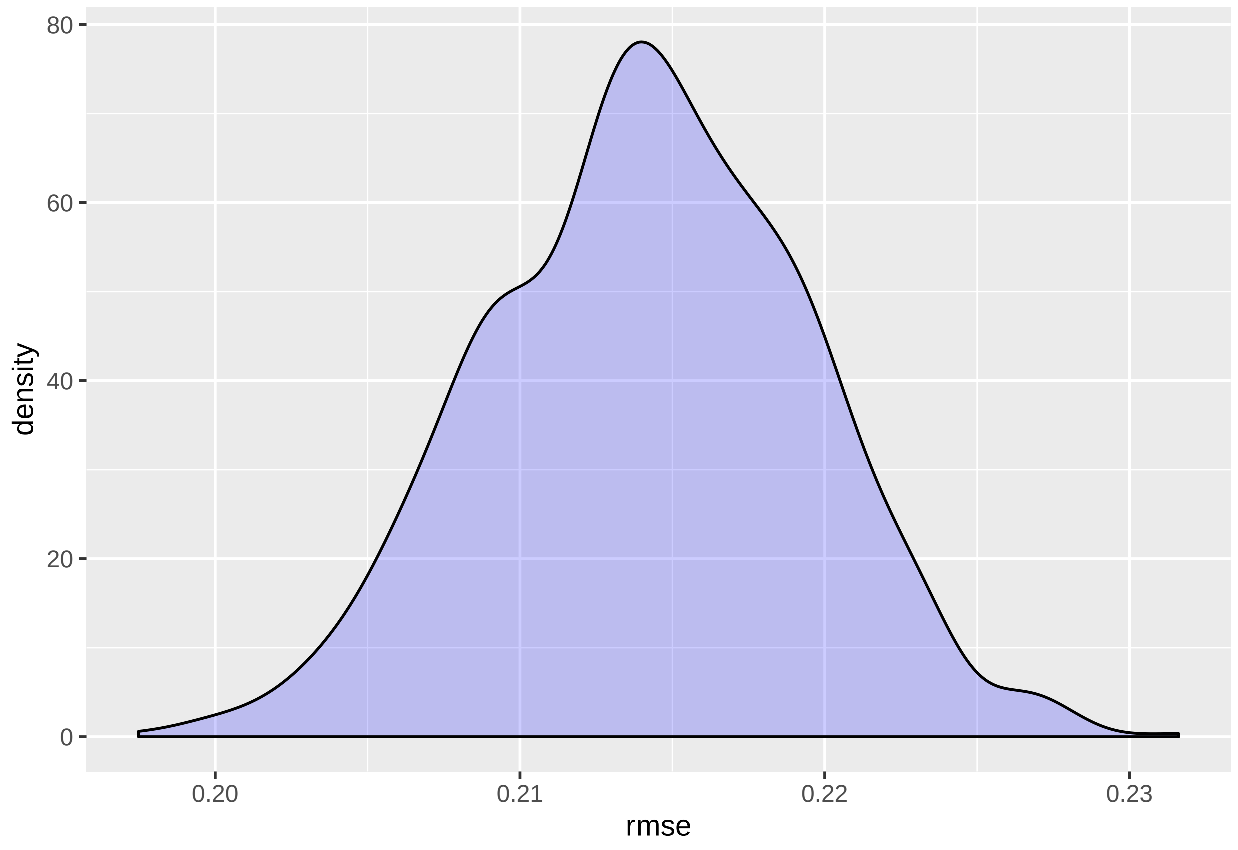
```
summary(mod1_rmse_cv$rmse)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.1975  0.2105  0.2142  0.2142  0.2178  0.2316
```

```
gg<-ggplot(mod1_rmse_cv,aes(rmse))
gg<-gg+geom_density(bins=50,fill="blue",alpha=.2)
```

```
## Warning: Ignoring unknown parameters: bins
```

```r
gg
```



Quick Question Run a cross validation with 10 and then 100 iterations. What happens to the distribution of RMSE when run with different numbers of iterations?

Selecting Between Models

It's the comparison between two different cross-validated models that we're really interested in. We want to know which model will perform best in predicting the future.

```r
tic()
## Define the model
mod2_formula<-formula("home_rank ~
                      coll_grad_pc +
                      per_capita_inc+
                      homeown_rate+
                      median_hh_inc")


mod2_rmse_cv<-pd_cv %>%
  mutate(train = map(train, as_tibble)) %>% ## Convert to tibbles
  mutate(model = map(train, ~ lm(mod2_formula, data = .)))%>%
  mutate(rmse = map2_dbl(model, test, rmse))%>%
  select(.id, rmse) ## pull just id and rmse

summary(mod2_rmse_cv$rmse)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.1853  0.1977  0.2007  0.2009  0.2043  0.2165
```

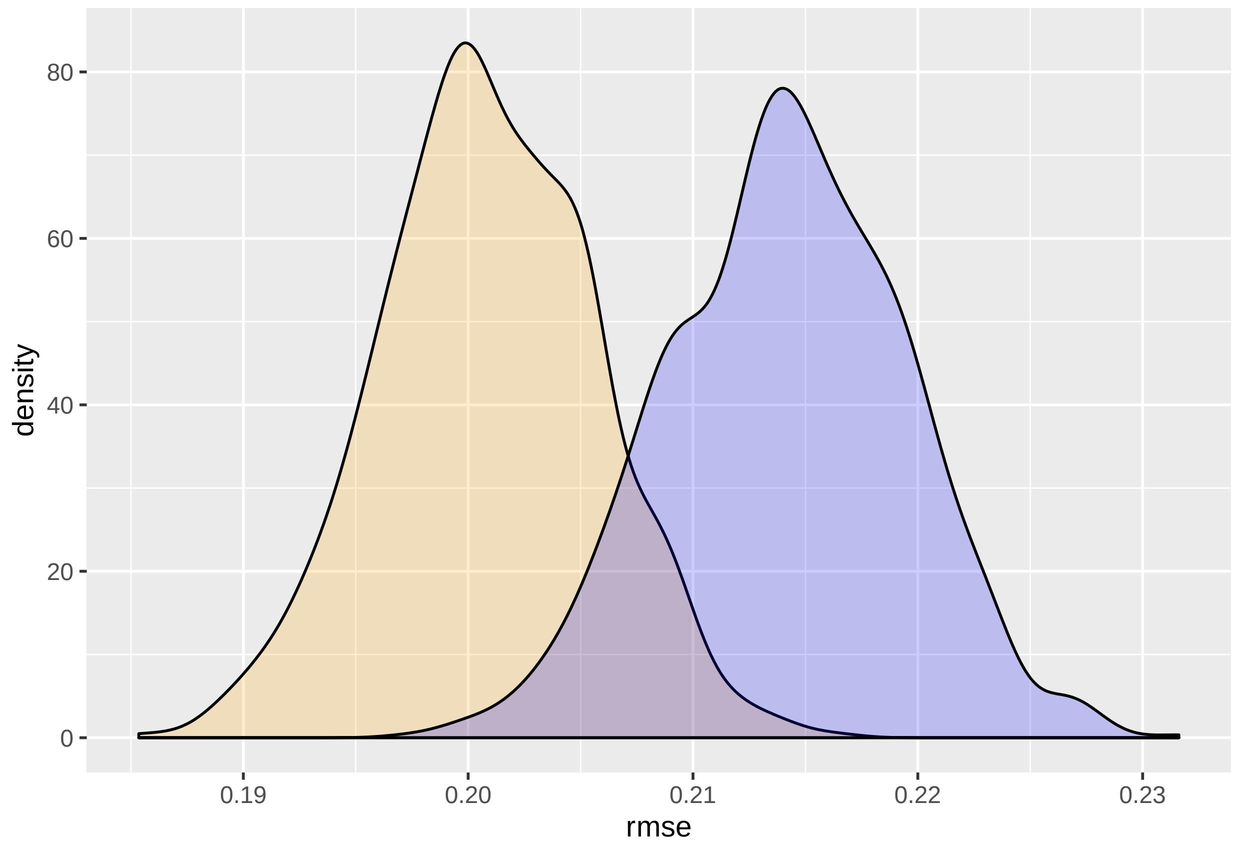
```
summary(mod1_rmse_cv$rmse)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.1975  0.2105  0.2142  0.2142  0.2178  0.2316
```

```
toc()
```

```
## 3.08 sec elapsed
```

```
gg<-ggplot(mod2_rmse_cv,aes(x=rmse))
gg<-gg+geom_density(fill="orange",alpha=.2)
gg<-gg+geom_density(data=mod1_rmse_cv,aes(x=rmse),fill="blue",alpha=.2)
gg
```



This graphic gives us a sense of model performance: While model 2 (orange) generally has lower rmse for out-of-sample predictions, there's still overlap in the performance between the two models. *Not So quick exercise* From the qf dataset, choose new variables to add to your model to predict median home values. Compare the distribution of rmse from your model to the ones I obtained.

Model Tuning

In model tuning, we try to find the best possible model among various candidates. There are two basic ways to acomplish this. One is using a model that the analyst has identified, either through theory or practice or both, as a good model to predict the outcome. This has been our basic perspective in this class.

Machine Learning

Of course, we can also just let the computer choose a model from a set of candidate variables. Below, I use stepwise regression, which involves proposing candidate variables and evaluating their ability to lower RMSE, as the basis for choosing a "best" model.

```r
#Tuning model parameters
pd<-pd%>%select(-median_home_val)


fitControl<-trainControl(method="boot",
                         p=.2)


fit1<-train(home_rank~per_capita_inc+
              coll_grad_pc,
            method="lm",
            data=pd,
            trControl=fitControl)

summary(fit1)
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.81893 -0.15064  0.01061  0.15740  0.58125
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)     -1.298e-01  1.837e-02  -7.066 1.96e-12 ***
## per_capita_inc   1.512e-05  1.149e-06  13.156  < 2e-16 ***
## coll_grad_pc     1.431e-02  7.278e-04  19.661  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2141 on 3085 degrees of freedom
## Multiple R-squared:  0.4513, Adjusted R-squared:  0.451
## F-statistic:  1269 on 2 and 3085 DF,  p-value: < 2.2e-16
```

```r
fit1$results
```

```
##   intercept      RMSE  Rsquared       MAE      RMSESD RsquaredSD
## 1      TRUE 0.2145718 0.4510114 0.1754029 0.002605736 0.01162468
##         MAESD
## 1 0.002346507
```

```r
## Stepwise Regression
fit2<-train(home_rank~.,
            data=pd,
            method="glmStepAIC",
            trControl=fitControl)
```

```
## Start:  AIC=-1150.95
## .outcome ~ median_hh_inc + coll_grad_pc + homeown_rate + per_capita_inc
##
##                  Df Deviance      AIC
## <none>                124.06 -1150.95
## - per_capita_inc  1   124.80 -1134.58
## - homeown_rate    1   126.23 -1099.48
## - coll_grad_pc    1   133.19  -933.71
## - median_hh_inc   1   141.86  -739.01
## Start:  AIC=-1106.84
## .outcome ~ median_hh_inc + coll_grad_pc + homeown_rate + per_capita_inc
##
##                  Df Deviance      AIC
## <none>                125.85 -1106.84
## - per_capita_inc  1   127.29 -1073.76
## - homeown_rate    1   128.05 -1055.32
## - coll_grad_pc    1   136.37  -860.88
## - median_hh_inc   1   146.20  -646.04
## Start:  AIC=-1178.79
## .outcome ~ median_hh_inc + coll_grad_pc + homeown_rate + per_capita_inc
##
##                  Df Deviance      AIC
## <none>                122.95 -1178.79
## - per_capita_inc  1   123.66 -1163.15
## - homeown_rate    1   123.75 -1160.90
## - coll_grad_pc    1   135.23  -886.83
## - median_hh_inc   1   138.57  -811.57
## Start:  AIC=-1120.59
## .outcome ~ median_hh_inc + coll_grad_pc + homeown_rate + per_capita_inc
##
##                  Df Deviance      AIC
## - per_capita_inc  1   125.35 -1121.15
## <none>                125.29 -1120.59
## - homeown_rate    1   128.11 -1053.94
## - coll_grad_pc    1   133.77  -920.43
## - median_hh_inc   1   139.09  -799.88
##
## Step:  AIC=-1121.15
## .outcome ~ median_hh_inc + coll_grad_pc + homeown_rate
##
##                 Df Deviance      AIC
## <none>               125.35 -1121.15
## - homeown_rate   1   128.49 -1046.76
## - coll_grad_pc   1   136.47  -860.60
## - median_hh_inc  1   151.39  -540.27
## Start:  AIC=-1319.1
## .outcome ~ median_hh_inc + coll_grad_pc + homeown_rate + per_capita_inc
##
##                  Df Deviance      AIC
## <none>                117.49 -1319.10
## - homeown_rate    1   118.22 -1301.83
## - per_capita_inc  1   118.23 -1301.78
## - coll_grad_pc    1   127.08 -1078.70
## - median_hh_inc   1   135.48  -881.22
## Start:  AIC=-970.17
## .outcome ~ median_hh_inc + coll_grad_pc + homeown_rate + per_capita_inc
##
##                  Df Deviance     AIC
## <none>                131.54 -970.17
## - per_capita_inc  1   132.53 -949.09
## - homeown_rate    1   133.59 -924.39
## - coll_grad_pc    1   140.27 -773.92
## - median_hh_inc   1   150.53 -555.95
## Start:  AIC=-1235.89
## .outcome ~ median_hh_inc + coll_grad_pc + homeown_rate + per_capita_inc
##
##                  Df Deviance      AIC
## <none>                120.70 -1235.89
## - per_capita_inc  1   121.07 -1228.49
## - homeown_rate    1   122.36 -1195.80
## - coll_grad_pc    1   128.30 -1049.32
## - median_hh_inc   1   138.69  -808.76
## Start:  AIC=-1214.66
## .outcome ~ median_hh_inc + coll_grad_pc + homeown_rate + per_capita_inc
##
##                  Df Deviance      AIC
## <none>                121.53 -1214.66
## - per_capita_inc  1   122.04 -1203.72
## - homeown_rate    1   122.95 -1180.72
## - coll_grad_pc    1   131.39  -975.84
## - median_hh_inc   1   139.31  -795.15
## Start:  AIC=-1220.54
## .outcome ~ median_hh_inc + coll_grad_pc + homeown_rate + per_capita_inc
##
##                  Df Deviance      AIC
## <none>                121.30 -1220.54
## - per_capita_inc  1   121.77 -1210.70
## - homeown_rate    1   122.44 -1193.59
## - coll_grad_pc    1   132.82  -942.34
## - median_hh_inc   1   136.49  -858.11
## Start:  AIC=-1060.35
## .outcome ~ median_hh_inc + coll_grad_pc + homeown_rate + per_capita_inc
##
##                  Df Deviance      AIC
## <none>                127.76 -1060.35
## - per_capita_inc  1   128.17 -1052.45
## - homeown_rate    1   129.49 -1020.86
## - coll_grad_pc    1   136.95  -847.75
## - median_hh_inc   1   143.46  -704.48
## Start:  AIC=-1194.96
## .outcome ~ median_hh_inc + coll_grad_pc + homeown_rate + per_capita_inc
##
##                  Df Deviance      AIC
## <none>                122.31 -1194.96
## - per_capita_inc  1   123.05 -1178.29
## - homeown_rate    1   123.54 -1165.99
## - coll_grad_pc    1   132.66  -946.07
## - median_hh_inc   1   139.18  -797.88
## Start:  AIC=-1260.91
## .outcome ~ median_hh_inc + coll_grad_pc + homeown_rate + per_capita_inc
##
##                  Df Deviance      AIC
## <none>                119.72 -1260.91
## - per_capita_inc  1   120.51 -1242.78
## - homeown_rate    1   120.89 -1232.99
## - coll_grad_pc    1   132.22  -956.38
## - median_hh_inc   1   135.98  -869.72
## Start:  AIC=-1076.32
## .outcome ~ median_hh_inc + coll_grad_pc + homeown_rate + per_capita_inc
##
##                  Df Deviance      AIC
## <none>                127.10 -1076.32
## - per_capita_inc  1   127.70 -1063.82
## - homeown_rate    1   129.86 -1012.01
## - coll_grad_pc    1   136.07  -867.81
## - median_hh_inc   1   141.82  -739.99
## Start:  AIC=-1062.53
## .outcome ~ median_hh_inc + coll_grad_pc + homeown_rate + per_capita_inc
##
##                  Df Deviance      AIC
## <none>                127.67 -1062.53
## - homeown_rate    1   128.89 -1035.07
## - per_capita_inc  1   129.03 -1031.78
## - coll_grad_pc    1   136.99  -846.84
## - median_hh_inc   1   147.46  -619.39
## Start:  AIC=-1123.59
## .outcome ~ median_hh_inc + coll_grad_pc + homeown_rate + per_capita_inc
##
##                  Df Deviance      AIC
## <none>                125.17 -1123.59
## - per_capita_inc  1   125.48 -1118.01
## - homeown_rate    1   127.25 -1074.59
## - coll_grad_pc    1   133.81  -919.48
## - median_hh_inc   1   140.07  -778.18
## Start:  AIC=-1163.38
## .outcome ~ median_hh_inc + coll_grad_pc + homeown_rate + per_capita_inc
##
##                  Df Deviance      AIC
## <none>                123.57 -1163.38
## - per_capita_inc  1   124.03 -1153.76
## - homeown_rate    1   125.19 -1125.03
## - coll_grad_pc    1   134.43  -905.17
## - median_hh_inc   1   139.97  -780.52
## Start:  AIC=-1225.47
## .outcome ~ median_hh_inc + coll_grad_pc + homeown_rate + per_capita_inc
##
##                  Df Deviance      AIC
## <none>                121.11 -1225.47
## - per_capita_inc  1   122.85 -1183.37
## - homeown_rate    1   123.09 -1177.35
## - coll_grad_pc    1   131.16  -981.12
## - median_hh_inc   1   142.89  -716.72
## Start:  AIC=-1127.9
## .outcome ~ median_hh_inc + coll_grad_pc + homeown_rate + per_capita_inc
##
##                  Df Deviance      AIC
## <none>                124.99 -1127.90
## - homeown_rate    1   125.57 -1115.60
## - per_capita_inc  1   125.80 -1110.04
## - coll_grad_pc    1   136.96  -847.61
## - median_hh_inc   1   141.63  -743.98
## Start:  AIC=-1178.45
## .outcome ~ median_hh_inc + coll_grad_pc + homeown_rate + per_capita_inc
##
##                  Df Deviance      AIC
## <none>                122.96 -1178.45
## - per_capita_inc  1   123.57 -1165.35
## - homeown_rate    1   125.25 -1123.69
## - coll_grad_pc    1   133.18  -934.08
## - median_hh_inc   1   139.47  -791.39
## Start:  AIC=-1162.02
## .outcome ~ median_hh_inc + coll_grad_pc + homeown_rate + per_capita_inc
##
##                  Df Deviance      AIC
## <none>                123.62 -1162.02
## - per_capita_inc  1   124.09 -1152.34
## - homeown_rate    1   124.50 -1142.14
## - coll_grad_pc    1   134.97  -892.87
## - median_hh_inc   1   138.04  -823.21
## Start:  AIC=-1183.3
## .outcome ~ median_hh_inc + coll_grad_pc + homeown_rate + per_capita_inc
##
##                  Df Deviance      AIC
## <none>                122.77 -1183.30
## - homeown_rate    1   123.32 -1171.64
## - per_capita_inc  1   123.99 -1154.80
## - coll_grad_pc    1   135.76  -874.71
## - median_hh_inc   1   139.00  -802.01
## Start:  AIC=-1182.2
## .outcome ~ median_hh_inc + coll_grad_pc + homeown_rate + per_capita_inc
##
##                  Df Deviance      AIC
## <none>                122.81 -1182.20
## - per_capita_inc  1   123.48 -1167.58
## - homeown_rate    1   124.21 -1149.26
## - coll_grad_pc    1   134.25  -909.24
## - median_hh_inc   1   136.83  -850.43
## Start:  AIC=-1215.13
## .outcome ~ median_hh_inc + coll_grad_pc + homeown_rate + per_capita_inc
##
##                  Df Deviance      AIC
## <none>                121.51 -1215.13
## - per_capita_inc  1   122.48 -1192.54
## - homeown_rate    1   122.69 -1187.43
## - coll_grad_pc    1   132.21  -956.49
## - median_hh_inc   1   139.50  -790.75
## Start:  AIC=-1180.07
## .outcome ~ median_hh_inc + coll_grad_pc + homeown_rate + per_capita_inc
##
##                  Df Deviance      AIC
## <none>                122.90 -1180.07
## - per_capita_inc  1   124.23 -1148.72
## - homeown_rate    1   124.71 -1136.93
## - coll_grad_pc    1   134.47  -904.22
## - median_hh_inc   1   142.21  -731.52
## Start:  AIC=-1110.7
## .outcome ~ median_hh_inc + coll_grad_pc + homeown_rate + per_capita_inc
##
##                  Df Deviance      AIC
## <none>                125.69 -1110.70
## - per_capita_inc  1   126.38 -1095.92
## - homeown_rate    1   126.54 -1091.95
## - coll_grad_pc    1   138.98  -802.37
## - median_hh_inc   1   140.87  -760.72
## Start:  AIC=-1142.64
## .outcome ~ median_hh_inc + coll_grad_pc + homeown_rate + per_capita_inc
##
##                  Df Deviance      AIC
## <none>                124.40 -1142.64
## - per_capita_inc  1   125.12 -1126.71
## - homeown_rate    1   125.85 -1108.88
## - coll_grad_pc    1   134.79  -896.92
## - median_hh_inc   1   141.02  -757.31
```

```
summary(fit2)
```

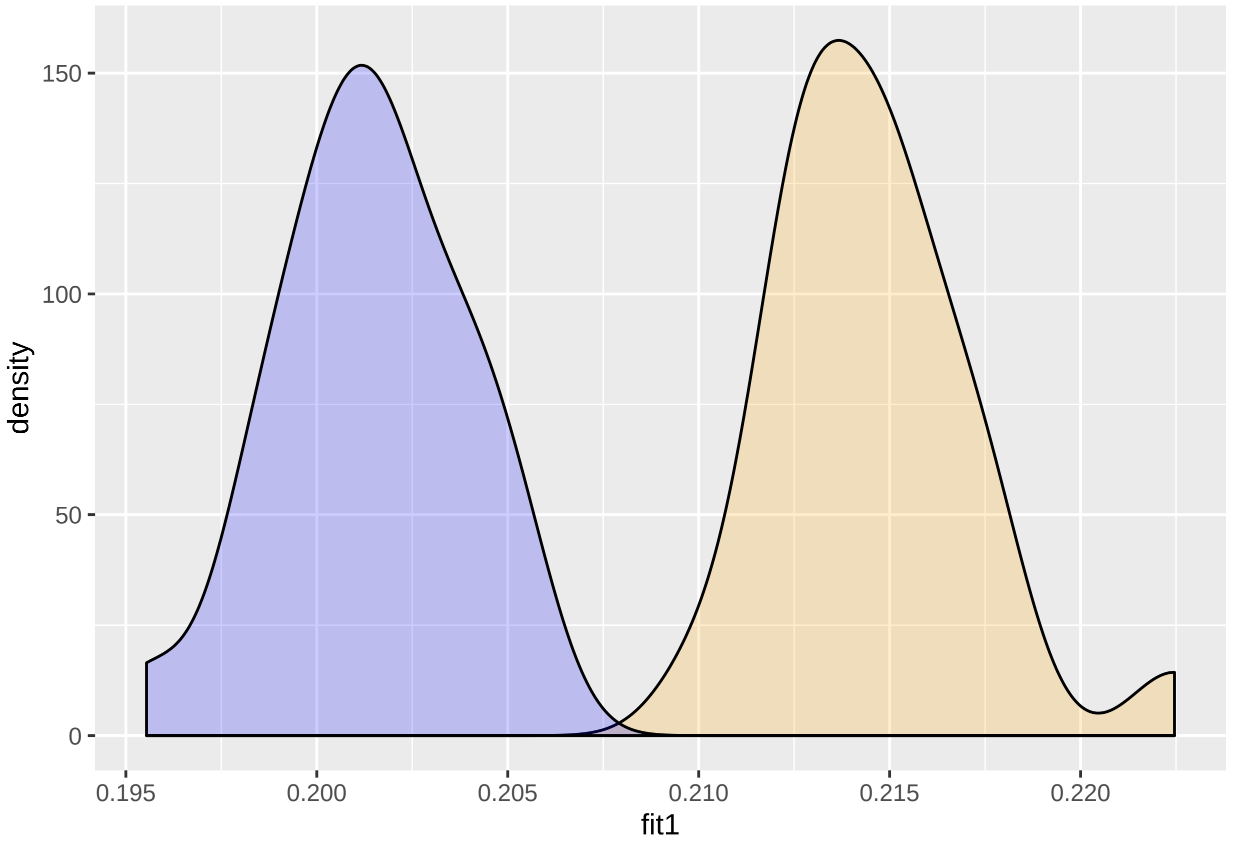
```
##
## Call:
## NULL
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -0.80967  -0.13518   0.00342   0.14635   0.55492
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)     5.712e-02  3.917e-02   1.458    0.145
## median_hh_inc   1.315e-05  6.480e-07  20.298  < 2e-16 ***
## coll_grad_pc    1.224e-02  7.628e-04  16.047  < 2e-16 ***
## homeown_rate   -3.188e-03  5.319e-04  -5.993 2.30e-09 ***
## per_capita_inc -6.916e-06  1.632e-06  -4.238 2.32e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.04034976)
##
##     Null deviance: 257.74  on 3087  degrees of freedom
## Residual deviance: 124.40  on 3083  degrees of freedom
## AIC: -1142.6
##
## Number of Fisher Scoring iterations: 2
```

```r
fit2$results
```

```
##   parameter      RMSE  Rsquared       MAE      RMSESD RsquaredSD
## 1      none 0.2013846 0.5172199 0.1635594 0.002442614   0.011531
##         MAESD
## 1 0.001769706
```

```r
rmse_data<-tbl_df(data.frame(fit1$resample$RMSE,fit2$resample$RMSE))
names(rmse_data)<-c("fit1","fit2")

gg<-ggplot(rmse_data,aes(x=fit1))
gg<-gg+geom_density(fill="orange",alpha=.2)
gg<-gg+geom_density(aes(x=fit2),fill="blue",alpha=.2)
gg
```



What this shows is that the second model—using stepwise regression— has superior out-of-sample results to the first model.

In different situations we may care more about WHY something predicts an outcome, and in other situations we care more about WHETHER something predicts an outcome. The key is to be clear with yourself about what you're interested in. Model selection via stepwise regression or other algorithms is not a panacea.