LASSO Prediction on Churn Data

```
library(data.table) # Open the data.table library for use in this script
library(ggplot2)
library(glmnet)

set.seed(2093) # Set the seed so that all of our random operations produce the same results
```

CRM

CRM stands for Customer Retail Management. This refers to when the marketer has enough data to tailor the marketing experience of each customer. Here we will explore models that can be used to predict whether a customer will churn.

Data

The data is a development sample that consists of 100,000 randomly selected customers from the data base of a a credit card company in Singapore. The random sample represents only a small percentage of the whole data base.

Customers are identified by a unique `customer_id`. The data also include demographic information on each customer, including whether their gender is male (`genderM`), their employment classification (`employXX`), income (`hhIncome`), household size (`hhSize`), whether they own a home (`homeOwner`), marital status (`married`). Additionally, some partial information on each customer's purchase history is included: the share of past months they have made late payments (`avgLatePayment`), the number of months they have been a customer of the credit card company (`nMonthsCust`), their average monthly bill (`avgMoBill`) and their credit score (`creditScore`).

Load the `churnCC` data set. We will use the `fread` command from the `data.table` package which opens text files like ".csv" much, much faster than base-R commands like `read.csv`.

```
churn_DT = fread("~/Dropbox/LargeRepoFiles/teaching/Teaching Data/output/Churn CC/data/churnCC.csv")
churn_DT[,.N] # .N is a special character for calling the number of observations in the data.table
churn_DT = churn_DT[runif(.N) <.01]   # This creates a subset of the original data for testing/developm
churn_DT[,.N]
```

Data inspection

Summarize some key aspects of the `churned` variable. In particular, what is the churn incidence, and the average churn value? Note that data.table offers a special character that may be used in the `j` position, `.N` which we will use here. `.N` reports the number of observations in the data set. When `.N` is combined with a `by` variable, then data.table reports the number of observations for each `by` group.

```r
# Number of observations for each value of `churned`
churn_DT[, .N, churned]
```

```
   churned    N
1:       0 7650
2:       1 2372
```

```r
# Share of observations for which `churned` is zero or one.
churn_DT[, .N / churn_DT[,.N], churned]
```

```
   churned        V1
1:       0 0.7633207
2:       1 0.2366793
```

```r
# Average value of `churned` which matches the output above.-
churn_DT[, mean(churned)]
```

```
[1] 0.2366793
```

Summarize the data

```
summary(churn_DT)
```

```
   customerID          genderM       employSelfEmployed  employCrafts
 Min.   :10000054   Min.   :0.0000   Min.   :0.00000    Min.   :0.0000
 1st Qu.:10243584   1st Qu.:0.0000   1st Qu.:0.00000    1st Qu.:0.0000
 Median :10499934   Median :1.0000   Median :0.00000    Median :0.0000
 Mean   :10497001   Mean   :0.5019   Mean   :0.06735    Mean   :0.1107
 3rd Qu.:10744425   3rd Qu.:1.0000   3rd Qu.:0.00000    3rd Qu.:0.0000
 Max.   :10999971   Max.   :1.0000   Max.   :1.00000    Max.   :1.0000
 employProfessional employClerical    employRetired     employStudent
 Min.   :0.0000     Min.   :0.00000   Min.   :0.00000   Min.   :0.00000
 1st Qu.:0.0000     1st Qu.:0.00000   1st Qu.:0.00000   1st Qu.:0.00000
 Median :1.0000     Median :0.00000   Median :0.00000   Median :0.00000
 Mean   :0.6488     Mean   :0.07633   Mean   :0.05448   Mean   :0.03233
 3rd Qu.:1.0000     3rd Qu.:0.00000   3rd Qu.:0.00000   3rd Qu.:0.00000
 Max.   :1.0000     Max.   :1.00000   Max.   :1.00000   Max.   :1.00000
    hhIncome          hhSize        homeOwner         married
 Min.   :     0   Min.   :1.000   Min.   :0.0000   Min.   :0.0000
 1st Qu.: 40000   1st Qu.:5.000   1st Qu.:1.0000   1st Qu.:0.0000
 Median : 60000   Median :5.000   Median :1.0000   Median :1.0000
 Mean   : 60010   Mean   :4.561   Mean   :0.8639   Mean   :0.7094
 3rd Qu.: 80000   3rd Qu.:5.000   3rd Qu.:1.0000   3rd Qu.:1.0000
 Max.   :180000   Max.   :5.000   Max.   :1.0000   Max.   :1.0000
 avgLatePayment       nMonthsCust       avgMoBill      creditScore
 Min.   :3.901e-05   Min.   : 0.000   Min.   : 10.0   Min.   :200.0
 1st Qu.:5.016e-02   1st Qu.: 3.000   1st Qu.: 80.0   1st Qu.:440.0
 Median :1.008e-01   Median : 4.000   Median :100.0   Median :520.0
 Mean   :1.003e-01   Mean   : 3.977   Mean   :100.2   Mean   :520.2
 3rd Qu.:1.506e-01   3rd Qu.: 5.000   3rd Qu.:120.0   3rd Qu.:600.0
 Max.   :2.000e-01   Max.   :14.000   Max.   :240.0   Max.   :800.0
    churned
 Min.   :0.0000
 1st Qu.:0.0000
 Median :0.0000
 Mean   :0.2367
 3rd Qu.:0.0000
 Max.   :1.0000
```

In this analysis we would like to predict who will churn, or the `churned` variable. We should take a minute to think about which of these variables we want to use to predict `churned`. Most of the variables appear to have the potential to predict `churned`, however, there are a few that should not be included as predictive variables.

First, `customerID` should have no relationship to whether a customer churned; these values are just a way to track customers. Additionally, even if `customerID` was somehow able to predict `churned` we would not want to use it because when we apply our model to predict churn for a different group of customers, their `customerID` values will be different. In this case, a model that includes `customerID` will not be helpful. In general, we want to restrict our predictor variables to those that will be available to us, with the same types of values.

Add additional variables that are random numbers

To help illustrate the value of LASSO regression, we are going to add several new variables that are random and do not help predict churned. (Normally, we would not do this.)

```
# This syntax is more advanced data.table and not a core part of this course
# This code creates many "garbage variables" that are just random noise.

garbNum = 300
garbVars = paste0("X", 1:garbNum)
churn_DT[, (garbVars) := 1]
churn_DT[, (garbVars) := lapply(.SD, function (x) runif(.N) ), .SDcols = garbVars]
```

Holdout sample

Because we want to make predictions and assess the quality of those predictions, we need to produce a **confusion matrix** for our predictions. To produce this matrix we need observations which include both our model's predictions and the actual outcomes. How do we generate such data? Well, we already have observations with actual outcomes, so we just need to add model predictions.

However, we have to be careful not to use the same data for estimating the model and for producing the **confusion matrix**. Using the same data leads to **overfit** a topic we will discuss further. For now, suffice it to say that we want to evaluate the predictions our model on data we have never seen before. After all, this is how we plan to use the predictions, so this is how we should evaluate their quality.

A good way to estimate the **confusion matrix** is by generating a **holdout sample**. A holdout sample is a random subset of the data we have. We will "estimate" or "train" our model using the "training data." Then, with our model estimated, we will test its predictions using the holdout data. The holdout data will be used to produce the confusion matrix.

```
churn_DT[, holdout := sample(0:1, size=.N, prob=c(0.5, 0.5), replace=TRUE)]
```

Estimate the model

For now we will use a standard logit model to make our predictions. This step is also known as *training* the model or *calibrating* the model. In later classes we will introduce other methods for making predictions. First we create a vector of the names of variables we don't want included in the regression. Placing them in `noRegVar` allows us to refer to these variables later.

```
# Variables that are not to be used for estimation
names(churn_DT)
```

```
  [1] "customerID"         "genderM"            "employSelfEmployed"
  [4] "employCrafts"       "employProfessional" "employClerical"
  [7] "employRetired"      "employStudent"      "hhIncome"
 [10] "hhSize"             "homeOwner"          "married"
 [13] "avgLatePayment"     "nMonthsCust"        "avgMoBill"
 [16] "creditScore"        "churned"            "X1"
 [19] "X2"                 "X3"                 "X4"
 [22] "X5"                 "X6"                 "X7"
 [25] "X8"                 "X9"                 "X10"
 [28] "X11"                "X12"                "X13"
 [31] "X14"                "X15"                "X16"
 [34] "X17"                "X18"                "X19"
 [37] "X20"                "X21"                "X22"
 [40] "X23"                "X24"                "X25"
 [43] "X26"                "X27"                "X28"
```

```
 [46] "X29"                 "X30"                 "X31"
 [49] "X32"                 "X33"                 "X34"
 [52] "X35"                 "X36"                 "X37"
 [55] "X38"                 "X39"                 "X40"
 [58] "X41"                 "X42"                 "X43"
 [61] "X44"                 "X45"                 "X46"
 [64] "X47"                 "X48"                 "X49"
 [67] "X50"                 "X51"                 "X52"
 [70] "X53"                 "X54"                 "X55"
 [73] "X56"                 "X57"                 "X58"
 [76] "X59"                 "X60"                 "X61"
 [79] "X62"                 "X63"                 "X64"
 [82] "X65"                 "X66"                 "X67"
 [85] "X68"                 "X69"                 "X70"
 [88] "X71"                 "X72"                 "X73"
 [91] "X74"                 "X75"                 "X76"
 [94] "X77"                 "X78"                 "X79"
 [97] "X80"                 "X81"                 "X82"
[100] "X83"                 "X84"                 "X85"
[103] "X86"                 "X87"                 "X88"
[106] "X89"                 "X90"                 "X91"
[109] "X92"                 "X93"                 "X94"
[112] "X95"                 "X96"                 "X97"
[115] "X98"                 "X99"                 "X100"
[118] "X101"                "X102"                "X103"
[121] "X104"                "X105"                "X106"
[124] "X107"                "X108"                "X109"
[127] "X110"                "X111"                "X112"
[130] "X113"                "X114"                "X115"
[133] "X116"                "X117"                "X118"
[136] "X119"                "X120"                "X121"
[139] "X122"                "X123"                "X124"
[142] "X125"                "X126"                "X127"
[145] "X128"                "X129"                "X130"
[148] "X131"                "X132"                "X133"
[151] "X134"                "X135"                "X136"
[154] "X137"                "X138"                "X139"
[157] "X140"                "X141"                "X142"
[160] "X143"                "X144"                "X145"
[163] "X146"                "X147"                "X148"
[166] "X149"                "X150"                "X151"
[169] "X152"                "X153"                "X154"
[172] "X155"                "X156"                "X157"
[175] "X158"                "X159"                "X160"
[178] "X161"                "X162"                "X163"
[181] "X164"                "X165"                "X166"
[184] "X167"                "X168"                "X169"
[187] "X170"                "X171"                "X172"
[190] "X173"                "X174"                "X175"
[193] "X176"                "X177"                "X178"
[196] "X179"                "X180"                "X181"
[199] "X182"                "X183"                "X184"
[202] "X185"                "X186"                "X187"
[205] "X188"                "X189"                "X190"
```

```
[208] "X191"                 "X192"                 "X193"
[211] "X194"                 "X195"                 "X196"
[214] "X197"                 "X198"                 "X199"
[217] "X200"                 "X201"                 "X202"
[220] "X203"                 "X204"                 "X205"
[223] "X206"                 "X207"                 "X208"
[226] "X209"                 "X210"                 "X211"
[229] "X212"                 "X213"                 "X214"
[232] "X215"                 "X216"                 "X217"
[235] "X218"                 "X219"                 "X220"
[238] "X221"                 "X222"                 "X223"
[241] "X224"                 "X225"                 "X226"
[244] "X227"                 "X228"                 "X229"
[247] "X230"                 "X231"                 "X232"
[250] "X233"                 "X234"                 "X235"
[253] "X236"                 "X237"                 "X238"
[256] "X239"                 "X240"                 "X241"
[259] "X242"                 "X243"                 "X244"
[262] "X245"                 "X246"                 "X247"
[265] "X248"                 "X249"                 "X250"
[268] "X251"                 "X252"                 "X253"
[271] "X254"                 "X255"                 "X256"
[274] "X257"                 "X258"                 "X259"
[277] "X260"                 "X261"                 "X262"
[280] "X263"                 "X264"                 "X265"
[283] "X266"                 "X267"                 "X268"
[286] "X269"                 "X270"                 "X271"
[289] "X272"                 "X273"                 "X274"
[292] "X275"                 "X276"                 "X277"
[295] "X278"                 "X279"                 "X280"
[298] "X281"                 "X282"                 "X283"
[301] "X284"                 "X285"                 "X286"
[304] "X287"                 "X288"                 "X289"
[307] "X290"                 "X291"                 "X292"
[310] "X293"                 "X294"                 "X295"
[313] "X296"                 "X297"                 "X298"
[316] "X299"                 "X300"                 "holdout"
```

```r
noRegVars = c("holdout", "customerID")

churn.fit.base = glm(churned ~ ., family="binomial",
   data=churn_DT[holdout==0, -c(..noRegVars, ..garbVars)] )

# Estimate a logit model with garbage variables
churn.fit.garb = glm(churned ~ ., family="binomial",
   data=churn_DT[holdout==0, -..noRegVars] )
```

We use the `glm` command to estimate a logit model with the setting `family="binomial"`. Note that the first entry in `gml` is the formula for the regression we want to estimate. `churned ~ .` indicates that we want to predict `churned` using all of the other variables in the data set, represented by `.`.

For the data set, we entered into to `glm` we have used some `data.table` commands to produce a subset of the original data. First, we selected only observations that are *not* in the holdout sample using `holdout==0`. In other words, we have selected the "training" data to estimate the model. Second, we removed the variables included in `noRegVars` using some `data.table` syntax. The minus sign `-` indicates that we are removing the variables. The double dots, `..` indicate that we are entering a vector of variable names and want to (de-)select those variables from the data set.

Inspect the estimates

```
summary(churn.fit.garb)
```

```
Call:
glm(formula = churned ~ ., family = "binomial", data = churn_DT[holdout ==
    0, -..noRegVars])

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.9476  -0.6948  -0.4356  -0.1656   3.1015

Coefficients:
                     Estimate Std. Error z value Pr(>|z|)
(Intercept)        -3.537e+00  1.261e+00  -2.805  0.00504 **
genderM            -1.752e-01  7.793e-02  -2.249  0.02454 *
employSelfEmployed -3.130e-01  3.964e-01  -0.790  0.42969
employCrafts       -4.044e-01  3.871e-01  -1.045  0.29620
employProfessional -6.853e-01  3.734e-01  -1.835  0.06644 .
employClerical     -3.302e-01  3.937e-01  -0.839  0.40170
employRetired      -2.941e-02  4.010e-01  -0.073  0.94153
employStudent      -5.278e-01  4.297e-01  -1.228  0.21929
hhIncome           -4.135e-06  1.585e-06  -2.609  0.00907 **
hhSize              3.907e-02  3.749e-02   1.042  0.29736
homeOwner           1.021e-01  1.164e-01   0.877  0.38051
married            -1.388e-01  8.572e-02  -1.620  0.10534
avgLatePayment      5.782e-01  6.689e-01   0.865  0.38731
nMonthsCust         9.718e-03  1.949e-02   0.499  0.61811
avgMoBill           3.242e-02  1.396e-03  23.223  < 2e-16 ***
creditScore        -4.242e-04  3.516e-04  -1.207  0.22761
X1                 -2.765e-02  1.354e-01  -0.204  0.83819
X2                 -3.111e-01  1.352e-01  -2.301  0.02138 *
X3                  6.729e-02  1.342e-01   0.501  0.61620
X4                 -1.848e-01  1.369e-01  -1.350  0.17692
X5                 -3.662e-03  1.342e-01  -0.027  0.97823
X6                  1.772e-01  1.347e-01   1.316  0.18834
X7                  1.272e-02  1.354e-01   0.094  0.92517
X8                 -1.852e-01  1.345e-01  -1.377  0.16856
X9                 -1.396e-01  1.368e-01  -1.020  0.30753
X10                -5.693e-02  1.360e-01  -0.419  0.67544
X11                -2.117e-01  1.362e-01  -1.555  0.11999
X12                 1.671e-02  1.344e-01   0.124  0.90105
X13                -3.274e-02  1.334e-01  -0.245  0.80612
X14                -1.335e-01  1.339e-01  -0.997  0.31867
X15                 2.141e-02  1.374e-01   0.156  0.87613
X16                -2.615e-01  1.345e-01  -1.944  0.05188 .
X17                 2.513e-02  1.371e-01   0.183  0.85459
X18                 2.054e-01  1.353e-01   1.518  0.12905
X19                 2.595e-01  1.354e-01   1.917  0.05525 .
X20                 1.114e-01  1.362e-01   0.818  0.41337
X21                -7.357e-02  1.352e-01  -0.544  0.58633
```

X22	9.944e-02	1.353e-01	0.735	0.46225	
X23	3.903e-02	1.327e-01	0.294	0.76870	
X24	3.275e-01	1.347e-01	2.432	0.01502	*
X25	1.543e-01	1.339e-01	1.153	0.24906	
X26	-1.220e-01	1.351e-01	-0.903	0.36649	
X27	-7.936e-02	1.362e-01	-0.583	0.56015	
X28	-1.167e-01	1.360e-01	-0.857	0.39119	
X29	1.250e-01	1.359e-01	0.920	0.35747	
X30	-8.931e-02	1.358e-01	-0.658	0.51062	
X31	9.830e-02	1.345e-01	0.731	0.46490	
X32	2.600e-02	1.357e-01	0.192	0.84807	
X33	-4.366e-02	1.351e-01	-0.323	0.74663	
X34	-1.921e-01	1.350e-01	-1.422	0.15488	
X35	1.584e-01	1.329e-01	1.192	0.23308	
X36	1.080e-01	1.348e-01	0.801	0.42304	
X37	5.450e-02	1.342e-01	0.406	0.68458	
X38	-4.427e-03	1.359e-01	-0.033	0.97401	
X39	-4.531e-02	1.356e-01	-0.334	0.73826	
X40	2.142e-01	1.364e-01	1.570	0.11632	
X41	9.877e-02	1.350e-01	0.732	0.46431	
X42	-2.809e-02	1.344e-01	-0.209	0.83441	
X43	-1.619e-01	1.361e-01	-1.190	0.23423	
X44	-6.595e-02	1.359e-01	-0.485	0.62754	
X45	2.759e-03	1.352e-01	0.020	0.98372	
X46	3.426e-01	1.382e-01	2.479	0.01318	*
X47	-1.789e-01	1.337e-01	-1.338	0.18084	
X48	8.038e-02	1.353e-01	0.594	0.55232	
X49	2.036e-01	1.337e-01	1.523	0.12783	
X50	4.018e-03	1.366e-01	0.029	0.97653	
X51	2.591e-02	1.369e-01	0.189	0.84992	
X52	-9.330e-02	1.359e-01	-0.687	0.49239	
X53	-1.223e-02	1.349e-01	-0.091	0.92776	
X54	2.443e-02	1.347e-01	0.181	0.85608	
X55	1.639e-01	1.355e-01	1.209	0.22660	
X56	1.704e-01	1.352e-01	1.261	0.20734	
X57	3.320e-02	1.349e-01	0.246	0.80560	
X58	-1.738e-01	1.343e-01	-1.294	0.19577	
X59	2.159e-01	1.335e-01	1.617	0.10595	
X60	-1.317e-01	1.353e-01	-0.974	0.33018	
X61	9.889e-02	1.336e-01	0.740	0.45900	
X62	2.310e-02	1.360e-01	0.170	0.86515	
X63	1.947e-01	1.346e-01	1.446	0.14817	
X64	5.929e-02	1.359e-01	0.436	0.66263	
X65	-1.779e-01	1.340e-01	-1.328	0.18428	
X66	4.269e-02	1.375e-01	0.310	0.75622	
X67	-6.873e-02	1.348e-01	-0.510	0.61026	
X68	-1.561e-02	1.363e-01	-0.115	0.90878	
X69	-9.168e-02	1.344e-01	-0.682	0.49527	
X70	3.430e-01	1.362e-01	2.519	0.01178	*
X71	-1.370e-01	1.362e-01	-1.006	0.31443	
X72	8.402e-02	1.356e-01	0.620	0.53553	
X73	-1.168e-01	1.365e-01	-0.856	0.39215	
X74	8.782e-02	1.350e-01	0.651	0.51525	
X75	6.272e-02	1.377e-01	0.455	0.64886	

X76	7.533e-02	1.351e-01	0.558	0.57709	
X77	2.167e-01	1.352e-01	1.602	0.10913	
X78	-8.331e-02	1.331e-01	-0.626	0.53127	
X79	7.415e-02	1.355e-01	0.547	0.58410	
X80	1.445e-01	1.329e-01	1.087	0.27691	
X81	1.443e-01	1.354e-01	1.066	0.28656	
X82	2.105e-01	1.358e-01	1.550	0.12112	
X83	1.741e-01	1.349e-01	1.290	0.19705	
X84	2.701e-02	1.364e-01	0.198	0.84304	
X85	2.875e-03	1.347e-01	0.021	0.98298	
X86	-2.175e-02	1.343e-01	-0.162	0.87133	
X87	-8.284e-02	1.380e-01	-0.600	0.54831	
X88	3.203e-01	1.352e-01	2.369	0.01785	*
X89	-6.149e-02	1.350e-01	-0.455	0.64887	
X90	9.841e-02	1.351e-01	0.728	0.46652	
X91	-1.398e-01	1.355e-01	-1.032	0.30208	
X92	4.724e-02	1.369e-01	0.345	0.73002	
X93	-1.805e-01	1.327e-01	-1.360	0.17378	
X94	-1.824e-01	1.342e-01	-1.359	0.17417	
X95	2.789e-01	1.349e-01	2.068	0.03862	*
X96	2.724e-01	1.363e-01	1.999	0.04563	*
X97	-8.091e-02	1.361e-01	-0.595	0.55214	
X98	-5.001e-02	1.369e-01	-0.365	0.71492	
X99	-5.776e-02	1.383e-01	-0.418	0.67625	
X100	-3.793e-03	1.351e-01	-0.028	0.97761	
X101	1.979e-01	1.342e-01	1.474	0.14043	
X102	2.936e-01	1.347e-01	2.179	0.02932	*
X103	-2.271e-01	1.354e-01	-1.677	0.09354	.
X104	1.109e-01	1.335e-01	0.831	0.40593	
X105	5.727e-02	1.356e-01	0.422	0.67272	
X106	9.075e-02	1.361e-01	0.667	0.50488	
X107	4.682e-02	1.378e-01	0.340	0.73396	
X108	1.555e-01	1.361e-01	1.143	0.25312	
X109	-5.101e-02	1.380e-01	-0.370	0.71162	
X110	1.184e-01	1.341e-01	0.883	0.37713	
X111	-1.372e-01	1.336e-01	-1.027	0.30443	
X112	-9.802e-02	1.361e-01	-0.720	0.47151	
X113	9.058e-02	1.348e-01	0.672	0.50148	
X114	5.747e-03	1.367e-01	0.042	0.96646	
X115	3.999e-02	1.337e-01	0.299	0.76483	
X116	5.331e-02	1.353e-01	0.394	0.69363	
X117	9.449e-03	1.343e-01	0.070	0.94392	
X118	7.377e-02	1.368e-01	0.539	0.58958	
X119	2.029e-01	1.350e-01	1.503	0.13285	
X120	2.152e-01	1.352e-01	1.592	0.11147	
X121	-4.866e-02	1.346e-01	-0.361	0.71777	
X122	-6.296e-02	1.349e-01	-0.467	0.64065	
X123	-6.825e-02	1.339e-01	-0.510	0.61014	
X124	-1.560e-02	1.374e-01	-0.113	0.90964	
X125	-2.022e-01	1.368e-01	-1.479	0.13922	
X126	1.231e-01	1.349e-01	0.913	0.36139	
X127	-1.730e-01	1.346e-01	-1.285	0.19872	
X128	-5.017e-02	1.352e-01	-0.371	0.71047	
X129	7.600e-02	1.355e-01	0.561	0.57494	

X130	1.302e-01	1.363e-01	0.955	0.33933	
X131	-7.610e-02	1.350e-01	-0.564	0.57308	
X132	2.339e-02	1.359e-01	0.172	0.86335	
X133	-6.303e-02	1.343e-01	-0.469	0.63889	
X134	-5.348e-05	1.357e-01	0.000	0.99969	
X135	-6.914e-02	1.354e-01	-0.511	0.60957	
X136	-3.773e-01	1.356e-01	-2.783	0.00538	**
X137	-1.381e-01	1.345e-01	-1.027	0.30436	
X138	-1.549e-01	1.364e-01	-1.135	0.25625	
X139	-2.884e-01	1.329e-01	-2.170	0.02998	*
X140	2.922e-02	1.328e-01	0.220	0.82590	
X141	-2.116e-01	1.335e-01	-1.585	0.11300	
X142	1.510e-01	1.337e-01	1.129	0.25874	
X143	-2.394e-01	1.357e-01	-1.764	0.07766	.
X144	-2.581e-01	1.355e-01	-1.905	0.05676	.
X145	8.600e-02	1.371e-01	0.627	0.53060	
X146	1.761e-01	1.342e-01	1.312	0.18955	
X147	-1.284e-02	1.348e-01	-0.095	0.92409	
X148	-6.447e-02	1.352e-01	-0.477	0.63332	
X149	-1.832e-01	1.356e-01	-1.351	0.17681	
X150	-1.070e-01	1.375e-01	-0.778	0.43644	
X151	1.154e-01	1.362e-01	0.847	0.39700	
X152	1.316e-01	1.350e-01	0.975	0.32956	
X153	-1.630e-01	1.347e-01	-1.210	0.22630	
X154	1.600e-01	1.359e-01	1.177	0.23914	
X155	1.406e-01	1.348e-01	1.044	0.29669	
X156	-4.950e-02	1.358e-01	-0.364	0.71549	
X157	-1.349e-01	1.342e-01	-1.005	0.31468	
X158	2.067e-02	1.353e-01	0.153	0.87852	
X159	3.584e-01	1.393e-01	2.572	0.01010	*
X160	-8.826e-02	1.350e-01	-0.654	0.51324	
X161	-3.438e-01	1.340e-01	-2.565	0.01032	*
X162	6.777e-02	1.357e-01	0.499	0.61746	
X163	1.503e-01	1.374e-01	1.094	0.27395	
X164	-9.397e-02	1.365e-01	-0.689	0.49105	
X165	-6.023e-02	1.345e-01	-0.448	0.65428	
X166	2.557e-02	1.369e-01	0.187	0.85190	
X167	-2.447e-01	1.341e-01	-1.825	0.06800	.
X168	1.789e-01	1.352e-01	1.324	0.18556	
X169	3.149e-01	1.344e-01	2.344	0.01909	*
X170	2.024e-01	1.356e-01	1.493	0.13540	
X171	6.335e-02	1.360e-01	0.466	0.64144	
X172	2.204e-02	1.360e-01	0.162	0.87125	
X173	-2.009e-01	1.343e-01	-1.496	0.13467	
X174	-1.721e-01	1.362e-01	-1.264	0.20632	
X175	-8.245e-02	1.353e-01	-0.609	0.54230	
X176	-2.279e-01	1.371e-01	-1.662	0.09642	.
X177	2.816e-02	1.368e-01	0.206	0.83687	
X178	-1.184e-01	1.347e-01	-0.879	0.37920	
X179	-1.546e-01	1.340e-01	-1.154	0.24852	
X180	-4.854e-02	1.352e-01	-0.359	0.71961	
X181	1.095e-01	1.346e-01	0.814	0.41593	
X182	-5.384e-02	1.350e-01	-0.399	0.69010	
X183	2.687e-01	1.354e-01	1.985	0.04717	*

X184	-2.407e-01	1.358e-01	-1.773	0.07629	.
X185	-2.421e-02	1.354e-01	-0.179	0.85810	
X186	-1.033e-01	1.343e-01	-0.769	0.44175	
X187	2.220e-02	1.369e-01	0.162	0.87112	
X188	-6.012e-02	1.367e-01	-0.440	0.66002	
X189	1.338e-01	1.356e-01	0.987	0.32355	
X190	-1.052e-01	1.365e-01	-0.771	0.44098	
X191	-2.343e-01	1.340e-01	-1.749	0.08028	.
X192	9.628e-02	1.359e-01	0.709	0.47860	
X193	-5.260e-02	1.355e-01	-0.388	0.69792	
X194	-1.919e-01	1.357e-01	-1.414	0.15736	
X195	-1.551e-01	1.356e-01	-1.144	0.25272	
X196	3.749e-01	1.339e-01	2.799	0.00513	**
X197	1.466e-01	1.351e-01	1.085	0.27784	
X198	-2.490e-02	1.347e-01	-0.185	0.85334	
X199	-4.442e-02	1.337e-01	-0.332	0.73977	
X200	1.096e-01	1.361e-01	0.805	0.42080	
X201	-8.679e-03	1.360e-01	-0.064	0.94913	
X202	2.938e-02	1.367e-01	0.215	0.82988	
X203	-1.396e-01	1.340e-01	-1.041	0.29770	
X204	4.186e-02	1.372e-01	0.305	0.76021	
X205	-2.550e-02	1.310e-01	-0.195	0.84566	
X206	8.049e-02	1.348e-01	0.597	0.55037	
X207	2.970e-01	1.367e-01	2.173	0.02980	*
X208	-4.290e-02	1.340e-01	-0.320	0.74892	
X209	4.961e-03	1.342e-01	0.037	0.97050	
X210	-1.150e-02	1.339e-01	-0.086	0.93156	
X211	7.622e-02	1.356e-01	0.562	0.57398	
X212	1.305e-01	1.358e-01	0.961	0.33664	
X213	-1.462e-01	1.361e-01	-1.075	0.28254	
X214	8.141e-02	1.351e-01	0.603	0.54664	
X215	1.021e-01	1.354e-01	0.754	0.45094	
X216	5.403e-02	1.351e-01	0.400	0.68915	
X217	1.047e-01	1.342e-01	0.781	0.43505	
X218	6.697e-02	1.384e-01	0.484	0.62857	
X219	-1.301e-01	1.358e-01	-0.958	0.33819	
X220	9.297e-02	1.342e-01	0.693	0.48836	
X221	1.533e-02	1.349e-01	0.114	0.90954	
X222	-5.824e-02	1.341e-01	-0.434	0.66412	
X223	-5.004e-02	1.365e-01	-0.367	0.71395	
X224	-2.098e-01	1.362e-01	-1.541	0.12339	
X225	1.886e-01	1.357e-01	1.390	0.16450	
X226	3.033e-02	1.350e-01	0.225	0.82222	
X227	-1.000e-01	1.338e-01	-0.747	0.45478	
X228	-2.412e-01	1.361e-01	-1.773	0.07626	.
X229	-1.776e-01	1.357e-01	-1.309	0.19062	
X230	2.791e-02	1.364e-01	0.205	0.83790	
X231	-1.416e-01	1.355e-01	-1.045	0.29623	
X232	-1.635e-01	1.353e-01	-1.209	0.22669	
X233	1.920e-03	1.355e-01	0.014	0.98870	
X234	-1.036e-01	1.356e-01	-0.764	0.44465	
X235	2.863e-02	1.331e-01	0.215	0.82968	
X236	-1.071e-01	1.349e-01	-0.793	0.42752	
X237	1.001e-01	1.375e-01	0.728	0.46660	

X238	-3.696e-01	1.369e-01	-2.700	0.00694	**
X239	-2.478e-01	1.347e-01	-1.840	0.06584	.
X240	-2.434e-01	1.348e-01	-1.806	0.07088	.
X241	2.603e-02	1.357e-01	0.192	0.84791	
X242	-1.484e-01	1.352e-01	-1.098	0.27235	
X243	1.799e-01	1.356e-01	1.327	0.18458	
X244	-1.907e-02	1.349e-01	-0.141	0.88763	
X245	1.120e-01	1.359e-01	0.824	0.40977	
X246	6.810e-02	1.361e-01	0.500	0.61675	
X247	-4.389e-01	1.375e-01	-3.193	0.00141	**
X248	2.010e-01	1.344e-01	1.495	0.13482	
X249	-2.590e-02	1.363e-01	-0.190	0.84934	
X250	-3.207e-01	1.335e-01	-2.403	0.01626	*
X251	-2.310e-02	1.360e-01	-0.170	0.86509	
X252	9.967e-03	1.360e-01	0.073	0.94158	
X253	-5.352e-02	1.343e-01	-0.398	0.69029	
X254	4.234e-01	1.350e-01	3.137	0.00171	**
X255	-2.090e-01	1.361e-01	-1.536	0.12462	
X256	-8.626e-02	1.351e-01	-0.638	0.52332	
X257	2.149e-02	1.328e-01	0.162	0.87144	
X258	1.929e-01	1.355e-01	1.424	0.15452	
X259	-8.838e-03	1.340e-01	-0.066	0.94742	
X260	-9.629e-02	1.348e-01	-0.714	0.47495	
X261	-2.989e-02	1.361e-01	-0.220	0.82613	
X262	-2.606e-01	1.356e-01	-1.922	0.05461	.
X263	-2.954e-01	1.340e-01	-2.204	0.02755	*
X264	1.966e-01	1.354e-01	1.452	0.14647	
X265	1.427e-01	1.363e-01	1.046	0.29534	
X266	5.816e-02	1.359e-01	0.428	0.66858	
X267	-2.174e-01	1.355e-01	-1.604	0.10866	
X268	6.107e-03	1.335e-01	0.046	0.96352	
X269	-6.670e-02	1.348e-01	-0.495	0.62067	
X270	3.654e-02	1.371e-01	0.267	0.78974	
X271	-1.101e-01	1.343e-01	-0.820	0.41229	
X272	4.710e-02	1.359e-01	0.347	0.72889	
X273	-4.377e-02	1.362e-01	-0.321	0.74786	
X274	3.096e-01	1.345e-01	2.301	0.02138	*
X275	-1.129e-01	1.361e-01	-0.830	0.40650	
X276	1.841e-01	1.363e-01	1.350	0.17692	
X277	-3.917e-02	1.352e-01	-0.290	0.77202	
X278	-9.402e-02	1.340e-01	-0.702	0.48275	
X279	-2.305e-01	1.355e-01	-1.701	0.08896	.
X280	6.473e-02	1.363e-01	0.475	0.63480	
X281	6.566e-02	1.350e-01	0.486	0.62663	
X282	-7.138e-02	1.353e-01	-0.528	0.59784	
X283	1.968e-01	1.354e-01	1.454	0.14603	
X284	2.173e-01	1.348e-01	1.612	0.10687	
X285	3.216e-02	1.379e-01	0.233	0.81556	
X286	2.506e-02	1.343e-01	0.187	0.85203	
X287	-4.410e-02	1.345e-01	-0.328	0.74294	
X288	1.182e-01	1.371e-01	0.862	0.38850	
X289	-1.698e-01	1.340e-01	-1.267	0.20503	
X290	-2.758e-01	1.356e-01	-2.035	0.04189	*
X291	-3.469e-02	1.358e-01	-0.255	0.79834	

```
X292                1.148e-01  1.351e-01   0.849  0.39561
X293                1.044e-01  1.338e-01   0.780  0.43533
X294                2.119e-02  1.352e-01   0.157  0.87549
X295               -4.318e-01  1.349e-01  -3.202  0.00137 **
X296               -2.274e-02  1.332e-01  -0.171  0.86444
X297                7.367e-02  1.356e-01   0.543  0.58683
X298               -1.379e-01  1.360e-01  -1.014  0.31049
X299                2.090e-01  1.354e-01   1.544  0.12256
X300                8.687e-02  1.344e-01   0.646  0.51814
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 5520.5  on 5021  degrees of freedom
Residual deviance: 4431.0  on 4706  degrees of freedom
AIC: 5063

Number of Fisher Scoring iterations: 5
```

Based on the coefficient estimates and their z-values, only a few of the variables we used in the model are statistically significant. This is okay, we didn't have clear sense of which variables would be important. Additionally, because we are primarily interested in making prediction with this model, we are not especially interested in the coefficient estimates.

Make predictions in a Holdout Sample

Now that we have estimated the model, we can generate the confusion matrix. Let's create a new data.table just for this purpose:

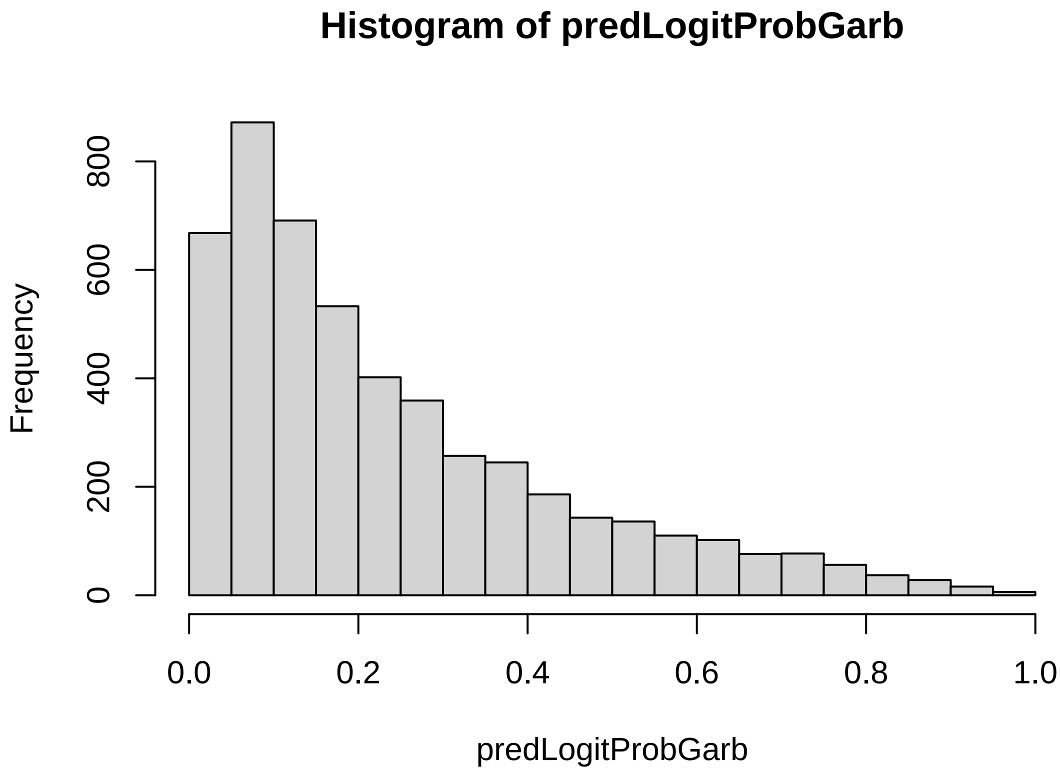
```r
churn_HO = churn_DT[holdout==1]
chHOmat = model.matrix( churned ~ ., churn_HO[, -..noRegVars])
```

Next, let's make predictions for this data set. Recall that none of this data was used to estimate the model, which is important for the validity of the predictions. We want to test our "out-of-sample" predictions (predictions on data other than the training data) since that is how we will use the model.

```r
churn_HO[, predLogitProbBase := predict(churn.fit.base, newdata=churn_HO, type = "response")]
churn_HO[, predLogitProbGarb := predict(churn.fit.garb, newdata=churn_HO, type = "response")]
```

This step used the estimated logit model to predict the probability that each customer in the holdout data would churn. Let's have a look at the distribution of these probabilities

```r
churn_HO[, hist(predLogitProbGarb)]
```




```
$breaks
 [1] 0.00 0.05 0.10 0.15 0.20 0.25 0.30 0.35 0.40 0.45 0.50 0.55 0.60 0.65 0.70
[16] 0.75 0.80 0.85 0.90 0.95 1.00

$counts
 [1] 668 872 691 533 402 359 257 245 186 143 136 110 102  76  77  56  37  28  16
[20]   6

$density
 [1] 2.672 3.488 2.764 2.132 1.608 1.436 1.028 0.980 0.744 0.572 0.544 0.440
[13] 0.408 0.304 0.308 0.224 0.148 0.112 0.064 0.024

$mids
 [1] 0.025 0.075 0.125 0.175 0.225 0.275 0.325 0.375 0.425 0.475 0.525 0.575
[13] 0.625 0.675 0.725 0.775 0.825 0.875 0.925 0.975

$xname
[1] "predLogitProbGarb"

$equidist
[1] TRUE

attr(,"class")
[1] "histogram"
```

Plot the Profit Curve

A function for calculating expected profits.

The following function, called `expProfit` provides a way to compute the expected profits associated with targeting different portions of the population. It should be applied to the holdout sample.

`expProfit` requires five input values. First is the name of the data.table that contains your holdout sample. Second is the name of the variable in the holdout sample that contains the **score** variables. In this application, this is `predLogitProb`. Third is the name of the variable that contains the actual outcomes that were observed. In this application, this is `churned`. Fourth is the value from the **Cost-Benefit Matrix** of correct predictions that the consumer will take an action. Fifth is the value from the **Cost-Benefit Matrix**

You are not responsible for understanding the coding details in this function, but I am happy to discuss them with you if interested.

```
# Function to calculate the expected profit at each score level
expProfit <- function(DT, score, actual, v11, v10) {
    # prepare copied data set for operations within the function with standardized names
    setnames(DT, c(actual, score), c("actual", "score"))
    DTloc <- copy(DT[, .(actual, score)]) # Copy a new version of the data set to work with in this fun
    setnames(DT, c("actual", "score"), c(actual, score)) # Return original variable names to original d
    DTloc[, origOrder := 1:.N] # Capture the order given for returning the output

    # Sort by score, highest to lowest.
    setorder( DTloc, -score )
    DTloc[, p11 := cumsum(actual==1) /.N] # Probability of true positives at a given score threshold
    DTloc[, p10 := cumsum(actual==0) /.N] # Probability of false positives at a given score threshold
    DTloc[, expProfit := v11*p11 + v10*p10] # Profits

    # Output expected profit estimates
    setorder(DTloc, origOrder)
    return(DTloc$expProfit)
}
```

Apply the `expProfit` function

```
### Base version of the model
# Apply the function to calculate expected profits for each threshold
churn_HO[, expProfitLogitBase := expProfit(
  DT = churn_HO,           # data.table use for the calculation
  score = "predLogitProbBase", # Which variable corresponds to the score
  actual = "churned",      # Which variable corresponds to the actual outcomes
  v11 = 4,                 # Value of a true positive
  v10 = -1)]               # Value of a false positive.


### Version of the model with extra "garbage" variables
# Apply the function to calculate expected profits for each threshold
churn_HO[, expProfitLogitGarb := expProfit(
  DT = churn_HO,           # data.table use for the calculation
  score = "predLogitProbGarb", # Which variable corresponds to the score
  actual = "churned",      # Which variable corresponds to the actual outcomes
  v11 = 4,                 # Value of a true positive
  v10 = -1)]               # Value of a false positive.
```

Estimate using LASSO

We will use the `glmnet` function for LASSO and ridge regressions. `glmnet` does not take data.tables as its input, but instead requires that the X data be converted to a matrix and the y data a vector.

We will use the "cross-validated" version of `glmnet`, which is called `cv.glmnet`. It uses the **cross validation** method we described in class to help discover the optimal value for `lambda`.

```
# Convert the data.table to a matrix and vector to input to glmnet
X = model.matrix( churned ~ . , churn_DT[holdout==0, -..noRegVars] ) # Creates a matrix of the X variab
y = churn_DT[holdout==0, churned] # Creates a vector of the y variables
lasso.fit <- cv.glmnet(X, y, family="binomial", alpha=1.0) # Estimates the model for binary outcomes. a
# (Ridge can be estimated with alpha=0.)
coef(lasso.fit, s = "lambda.min") # Report the estimates for each coefficients estimated
```

```
317 x 1 sparse Matrix of class "dgCMatrix"
                              s1
(Intercept)        -3.813829e+00
(Intercept)         .
genderM            -5.970516e-02
employSelfEmployed  .
employCrafts        .
employProfessional -2.220383e-01
employClerical      .
employRetired       9.774171e-02
employStudent       .
hhIncome           -1.234730e-06
hhSize              .
homeOwner           .
married             .
avgLatePayment      .
nMonthsCust         .
avgMoBill           2.775950e-02
creditScore         .
X1                  .
X2                 -4.351818e-02
X3                  .
X4                  .
X5                  .
X6                  .
X7                  .
X8                  .
X9                  .
X10                 .
X11                 .
X12                 .
X13                 .
X14                 .
X15                 .
X16                -9.094375e-03
X17                 .
X18                 .
X19                 4.839712e-02
X20                 .
X21                 .
```

X22	.
X23	.
X24	8.513802e-02
X25	.
X26	.
X27	.
X28	.
X29	.
X30	.
X31	.
X32	.
X33	.
X34	.
X35	.
X36	.
X37	.
X38	.
X39	.
X40	.
X41	.
X42	.
X43	.
X44	.
X45	.
X46	1.109828e-01
X47	.
X48	.
X49	1.541288e-02
X50	.
X51	.
X52	.
X53	.
X54	.
X55	.
X56	.
X57	.
X58	.
X59	.
X60	.
X61	.
X62	.
X63	.
X64	.
X65	.
X66	.
X67	.
X68	.
X69	.
X70	8.799158e-02
X71	.
X72	.
X73	.
X74	.
X75	.

```
X76          .
X77           3.533740e-02
X78          .
X79          .
X80          .
X81          .
X82           2.576636e-03
X83          .
X84          .
X85          .
X86          .
X87          .
X88           1.033319e-01
X89          .
X90          .
X91          .
X92          .
X93          .
X94          -4.833706e-03
X95          .
X96           2.333157e-02
X97          .
X98          .
X99          .
X100         .
X101          1.262619e-03
X102          9.272279e-02
X103         .
X104         .
X105         .
X106         .
X107         .
X108          8.462068e-03
X109         .
X110         .
X111         .
X112         .
X113         .
X114         .
X115         .
X116         .
X117         .
X118         .
X119         .
X120         .
X121         .
X122         .
X123         .
X124         .
X125         .
X126         .
X127         .
X128         .
X129         .
```

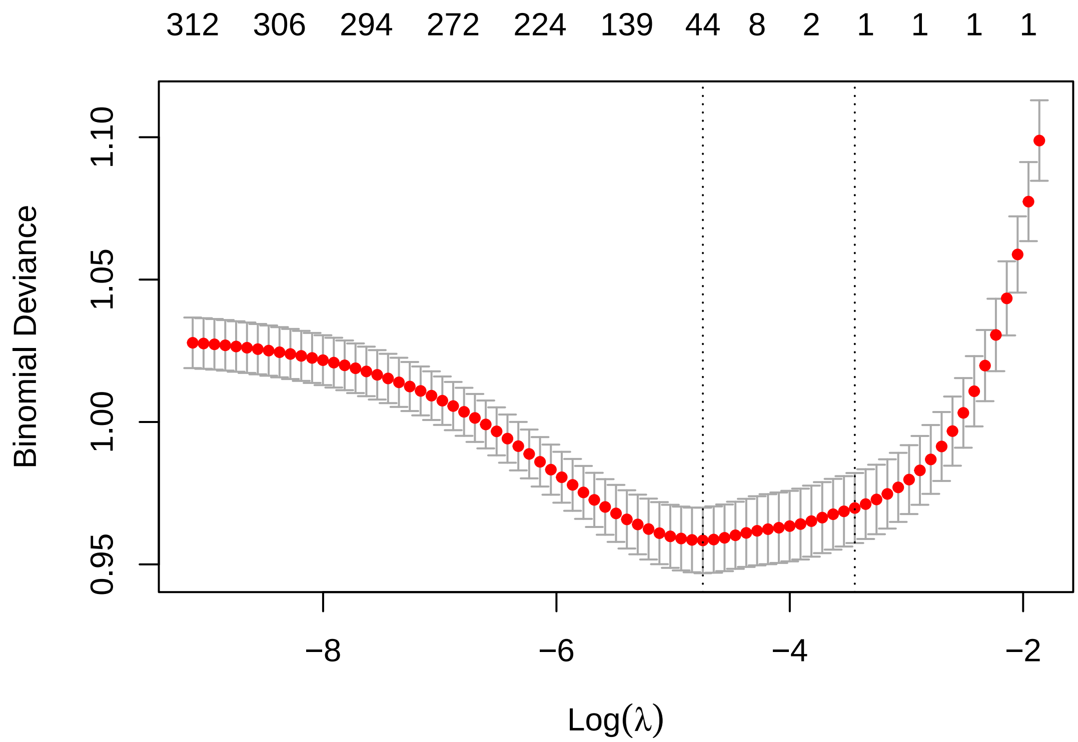
```
X130             .
X131             .
X132             .
X133             .
X134             .
X135             .
X136            -1.014284e-01
X137             .
X138             .
X139            -4.815335e-02
X140             .
X141             .
X142             .
X143             .
X144             .
X145             .
X146             .
X147             .
X148             .
X149             .
X150             .
X151             .
X152             .
X153             .
X154             .
X155             .
X156             .
X157             .
X158             .
X159             8.493787e-02
X160             .
X161            -1.470461e-01
X162             .
X163             .
X164             .
X165             .
X166             .
X167            -7.960225e-02
X168             .
X169             4.623966e-02
X170             5.844784e-05
X171             .
X172             .
X173             .
X174             .
X175             .
X176             .
X177             .
X178             .
X179             .
X180             .
X181             .
X182             .
X183             3.040603e-02
```

X184	-2.731483e-02
X185	.
X186	.
X187	.
X188	.
X189	.
X190	.
X191	-3.834900e-02
X192	.
X193	.
X194	.
X195	.
X196	1.260179e-01
X197	.
X198	.
X199	.
X200	.
X201	.
X202	.
X203	.
X204	.
X205	.
X206	.
X207	7.062899e-02
X208	.
X209	.
X210	.
X211	.
X212	.
X213	.
X214	.
X215	.
X216	.
X217	.
X218	.
X219	.
X220	.
X221	.
X222	.
X223	.
X224	.
X225	.
X226	.
X227	.
X228	.
X229	.
X230	.
X231	.
X232	.
X233	.
X234	.
X235	.
X236	.
X237	.

X238	-1.033301e-01
X239	-6.660676e-02
X240	-3.570075e-02
X241	.
X242	.
X243	.
X244	.
X245	.
X246	.
X247	-1.890971e-01
X248	.
X249	.
X250	-5.483952e-02
X251	.
X252	.
X253	.
X254	1.758450e-01
X255	-3.400405e-02
X256	.
X257	.
X258	.
X259	.
X260	.
X261	.
X262	-6.914809e-02
X263	-7.893707e-02
X264	.
X265	.
X266	.
X267	.
X268	.
X269	.
X270	.
X271	.
X272	.
X273	.
X274	5.608535e-02
X275	.
X276	.
X277	.
X278	.
X279	.
X280	.
X281	.
X282	.
X283	.
X284	.
X285	.
X286	.
X287	.
X288	.
X289	.
X290	-1.258814e-02
X291	.

```
X292                  .
X293                  .
X294                  .
X295                 -1.977191e-01
X296                  .
X297                  .
X298                  .
X299                  .
X300                  .
```

```
plot(lasso.fit) # Plot the quality of the predictions for different lambda values.
```



Calculate the expected profits based on the LASSO model

```
churn_HO[, predLASSOProb := predict(lasso.fit, chHOmat, s= "lambda.min", type="response")]

# Apply the function to calculate expected profits for each threshold
churn_HO[, expProfitLASSO := expProfit(
  DT = churn_HO,            # data.table use for the calculation
  score = "predLASSOProb", # Which variable corresponds to the score
  actual = "churned",       # Which variable corresponds to the actual outcomes
  v11 = 4,                  # Value of a true positive
  v10 = -1)]                # Value of a false positive.
```

Reformat the data for plotting

`ggplot` requires that the data be in a specific format for plotting multiple lines. In particular, the variables on the x and y-axes need to each correspond to a single variable name. In this case, x corresponds to `shareTest` and y corresponds to `expProfit`. In order to plot multiple lines on the same plot, we require a third variable that indicates how the different plots are different. In this case, I have called this third variable `method` and it takes values "LASSO" and "Logit."

To put the data in the required shape, I have made two new subsets of the original holdout data. The first reports the x and y values for our "Logit" estimates and the second the values for the "LASSO" estimates. These two new data.tables are then stacked one on top of the other. To accomplish this stacking I place each of the two new data.tables as different objects in the list and then call `rbindlist` which "binds" the the two data.tables by row.

The resulting data.table is called `profitDT` and can be plot with `ggplot`.

```
# Stack the data from Logit and LASSO profits
chHOlogitBase = churn_HO[order(-predLogitProbBase), .(shareTest=(1:.N)/.N, expProfit = expProfitLogitBa
chHOlogitGarb = churn_HO[order(-predLogitProbGarb), .(shareTest=(1:.N)/.N, expProfit = expProfitLogitGa
chHOlasso =churn_HO[order(-predLASSOProb), .(shareTest=(1:.N)/.N, expProfit = expProfitLASSO, method="L

profitDT = rbindlist( list(chHOlogitBase,chHOlogitGarb, chHOlasso ))

# Plot the two profit curves on one plot
ggplot(data=profitDT, aes(x=shareTest, y= expProfit, color=method)) +
  geom_line() + theme_bw()
```