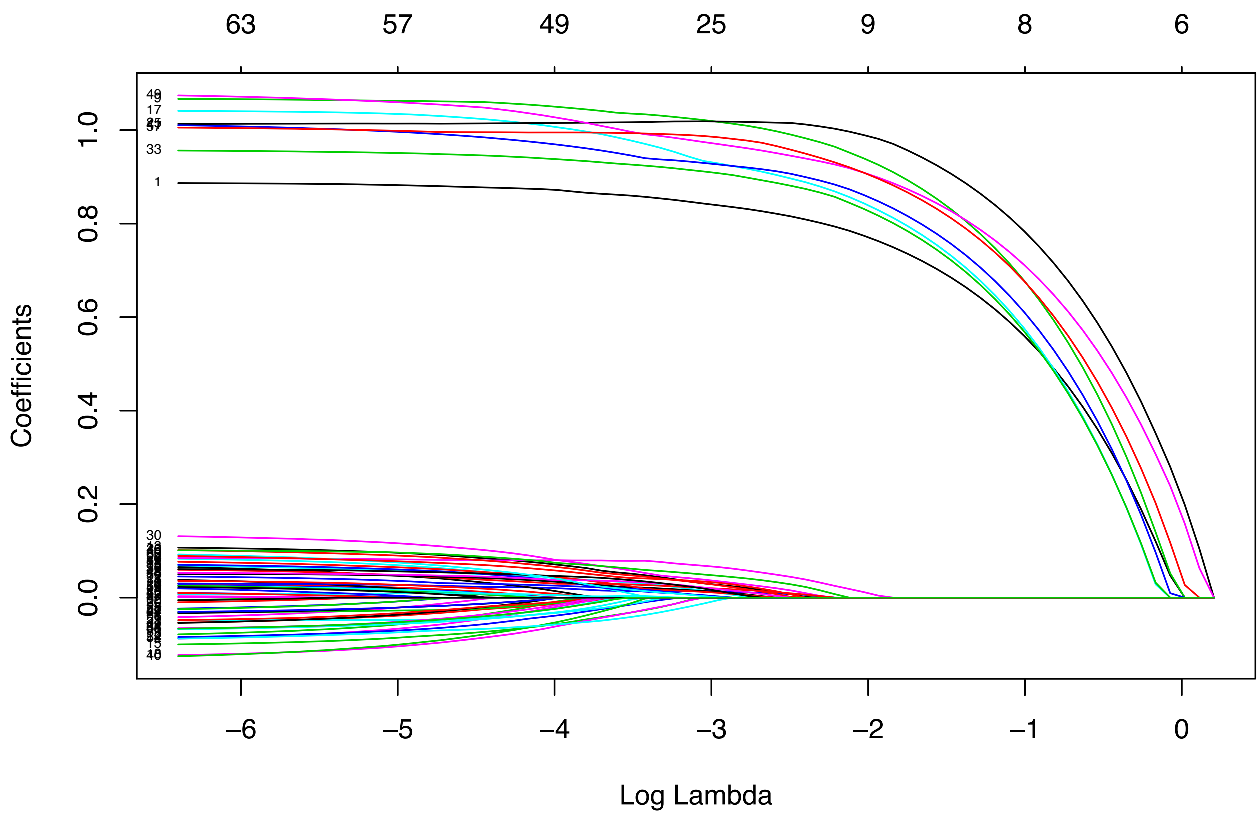
project3

Contents

```
load("~/Downloads/data.rdata")
#install.packages('/Library/gurobi702/mac64/R/gurobi_7.0-2.tgz', repos=NULL,type="source")
# change the directory if you are not using MAX OS
#install.packages("glmnet", repos = "http://cran.us.r-project.org")

# Indirect Feature Selection (Lasso Regression)
library(glmnet)

## Loading required package: Matrix

## Loading required package: foreach

## Loaded glmnet 2.0-5

fit = glmnet(X, y, alpha = 1)
plot(fit, xvar = "lambda", label = TRUE)
```




```
# The plot shows a cut-off at Log Lambda = -2; we will use e^(-2) as the penalty term of our Lasso model
lasso_beta = coef(fit,s=exp(-2))[-1]
lasso_real_comp = cbind(lasso_beta,beta_real)
# Comparing the real betas and the results returned by Lasso:
lasso_real_comp
```

```
##        lasso_beta beta_real
##  [1,] 0.770760978         1
##  [2,] 0.000000000         0
##  [3,] 0.000000000         0
##  [4,] 0.000000000         0
##  [5,] 0.000000000         0
##  [6,] 0.000000000         0
##  [7,] 0.000000000         0
##  [8,] 0.000000000         0
##  [9,] 0.935553531         1
## [10,] 0.000000000         0
## [11,] 0.000000000         0
## [12,] 0.000000000         0
## [13,] 0.000000000         0
## [14,] 0.000000000         0
## [15,] 0.000000000         0
## [16,] 0.000000000         0
## [17,] 0.839184926         1
## [18,] 0.000000000         0
## [19,] 0.000000000         0
## [20,] 0.000000000         0
## [21,] 0.000000000         0
## [22,] 0.000000000         0
## [23,] 0.000000000         0
## [24,] 0.009660727         0
## [25,] 0.987048372         1
## [26,] 0.000000000         0
## [27,] 0.000000000         0
## [28,] 0.000000000         0
## [29,] 0.000000000         0
## [30,] 0.000000000         0
## [31,] 0.000000000         0
## [32,] 0.000000000         0
## [33,] 0.827306844         1
## [34,] 0.000000000         0
## [35,] 0.000000000         0
## [36,] 0.000000000         0
## [37,] 0.000000000         0
## [38,] 0.000000000         0
## [39,] 0.000000000         0
## [40,] 0.000000000         0
## [41,] 0.857291714         1
## [42,] 0.000000000         0
## [43,] 0.000000000         0
## [44,] 0.000000000         0
## [45,] 0.000000000         0
## [46,] 0.000000000         0
## [47,] 0.000000000         0
## [48,] 0.000000000         0
## [49,] 0.905918040         1
## [50,] 0.000000000         0
## [51,] 0.000000000         0
## [52,] 0.000000000         0
## [53,] 0.000000000         0
```

```
## [54,] 0.000000000         0
## [55,] 0.000000000         0
## [56,] 0.000000000         0
## [57,] 0.905178846         1
## [58,] 0.000000000         0
## [59,] 0.000000000         0
## [60,] 0.000000000         0
## [61,] 0.000000000         0
## [62,] 0.000000000         0
## [63,] 0.000000000         0
## [64,] 0.000000000         0
```

```
# we can see that Lasso successfully regularized the irrelevant features by shrinking their correspondi

# Direct Selection (MIP)
construct_MIQP = function(X,y,k,M){
  # this function is to formulate the Mixed Interger Programming Problem
  n = dim(X)[2]
  # n = 64 in our case

  # we created 128 variables in total:
  # the first 64 (B1, B2,.., B64) are continuous variables representing each independent variable;
  # the last 64 (Z1, Z2,.., Z64) are binary variables: Zi (i in 1:64) indicates whether Bi is 0
  model <- list()
  model$vtype <- c(rep('C',n),rep('B',n))

  # Formulate Constraints
  A = matrix(0,2*n,2*n)
  # 1. for -M*Zi <= Bi <= M*Zi:
  A[1:n,1:n] = -1*diag(n)
  A[1:n,(n+1):(2*n)] = -M*diag(n)
  A[(n+1):(2*n),1:n] = diag(n)
  A[(n+1):(2*n),(n+1):(2*n)] = -M*diag(n)
  # 2. for Z1 + Z2 + ... + Z64 = k:
  A1 = c(rep(0,n),rep(1,n))
  model$A <- rbind(A1,A)
  model$sense <- rep("<=",2*n+1)
  model$rhs <- c(k,rep(0,2*n))

  # Formulate Objective (i.e. sum of squared residuals in our case)
  Q = matrix(0,2*n,2*n)
  # 1. quadratic component of the objective function
  Q[0:n,0:n] = t(X) %*% X
  model$Q = Q
  # 2. linear component of the objective function
  model$obj <- c(-2*y %*% X,rep(0,n))

  result <- gurobi(model, list(ResultFile='model.mps',OutputFlag=0))
  return(result)
}
```

Unfortunately, the GUROBI library works in RStudio but not in RMarkdown. So please forgive the awkward copy-and-paste solution

```
library(gurobi)
# the number of variables is specified as 8 in this project
k=8
# M is the bound for the absolute value of coefficients; start with a small M for regularization
M = 0.1
initial_sol = construct_MIQP(X,y,k,M)
# check whether the largest absolute value of our current solution is strictly smaller than M
max = max(abs(initial_sol$x[1:64]))
max
```

Output:

```
> max
[1] 0.1
```

```
# doubling the M until it exceeds the largest absolute value of the optimal solution
while (M<=max){
  M = M*2
  current_sol = construct_MIQP(X,y,k,M)
  max = max(abs(current_sol$x[1:64]))
  print(M)
}
```

Output:

```
[1] 0.2
[1] 0.4
[1] 0.8
[1] 1.6
# M = 1.6 satisfied the condition and quit the loop
```

```
MIQP_beta = current_sol$x[1:64]

sol_compare = cbind(lasso_beta,MIQP_beta,beta_real)
write.csv(sol_compare, 'sol_compare.csv')
```

Comparison of results using the LASSO and MIQP methods with the real betas:

	lasso_beta	MIQP_beta	beta_real
1	0.77076	0.89306	1.00000
2	0.00000	0.00000	0.00000
3	0.00000	0.00000	0.00000
4	0.00000	0.00000	0.00000
5	0.00000	0.00000	0.00000
6	0.00000	0.00000	0.00000
7	0.00000	0.00000	0.00000
8	0.00000	0.00000	0.00000
9	0.93555	1.08924	1.00000
10	0.00000	0.00000	0.00000
11	0.00000	0.00000	0.00000
12	0.00000	0.00000	0.00000
13	0.00000	0.00000	0.00000
14	0.00000	0.00000	0.00000
15	0.00000	0.00000	0.00000
16	0.00000	0.00000	0.00000
17	0.83918	0.99210	1.00000
18	0.00000	0.00000	0.00000
19	0.00000	0.00000	0.00000
20	0.00000	0.00000	0.00000
21	0.00000	0.00000	0.00000
22	0.00000	0.00000	0.00000
23	0.00000	0.00000	0.00000
24	0.00966	0.00000	0.00000
25	0.98705	1.11583	1.00000
26	0.00000	0.00000	0.00000
27	0.00000	0.00000	0.00000
28	0.00000	0.00000	0.00000
29	0.00000	0.00000	0.00000
30	0.00000	0.00000	0.00000
31	0.00000	0.00000	0.00000
32	0.00000	0.00000	0.00000
33	0.82731	0.97974	1.00000
34	0.00000	0.00000	0.00000
35	0.00000	0.00000	0.00000
36	0.00000	0.00000	0.00000
37	0.00000	0.00000	0.00000
38	0.00000	0.00000	0.00000
39	0.00000	0.00000	0.00000

40	0.00000	0.00000	0.00000
41	0.85729	1.00299	1.00000
42	0.00000	0.00000	0.00000
43	0.00000	0.00000	0.00000
44	0.00000	0.00000	0.00000
45	0.00000	0.00000	0.00000
46	0.00000	0.00000	0.00000
47	0.00000	0.00000	0.00000
48	0.00000	0.00000	0.00000
49	0.90592	1.02020	1.00000
50	0.00000	0.00000	0.00000
51	0.00000	0.00000	0.00000
52	0.00000	0.00000	0.00000
53	0.00000	0.00000	0.00000
54	0.00000	0.00000	0.00000
55	0.00000	0.00000	0.00000
56	0.00000	0.00000	0.00000
57	0.90518	1.03892	1.00000
58	0.00000	0.00000	0.00000
59	0.00000	0.00000	0.00000
60	0.00000	0.00000	0.00000
61	0.00000	0.00000	0.00000
62	0.00000	0.00000	0.00000
63	0.00000	0.00000	0.00000
64	0.00000	0.00000	0.00000

In the end, we compare the prediction error of the two regressions:

```
compute_error -> function(sol_beta, real_beta, X){
    return(sum((X%*%sol_beta - X%*%real_beta)^2)/sum((X%*%real_beta)^2))
}
compute_error(MIQP_beta, beta_real, X)
compute_error(lasso_beta, beta_real, X)
```

Output:

```
> compute_error(MIQP_beta, beta_real, X)
[1] 0.004456055
> compute_error(lasso_beta, beta_real, X)
[1] 0.01835347
```