

ETF Casestudy

Peter Kempthorne

2024-09-23

Contents

0. Load Libraries	2
1. Load data into R session	4
1.1 Daily Stock Price Data from Yahoo	4
1.2 Weekly Stock Price Data	6
1.3 Save data objects for casestudy in R data file	9
2. Analysis of returns for specific period (2013-2024)	10
2.1 Subset data objects for specific period	10
2.2 Define y0 equal to weekly returns of specific sector etf (XLB)	10
2.3 Define df0 equal to data frame with y0 and index etfs	11
2.4 Print out summary statistics for weekly returns	11
2.5 Compute/plot cumulative returns	12
3. Regression of Sector ETF on Index ETFs	14
3.1 Fit regression of y0 on index etf returns	14
3.2 Plot Actual versus Fitted Values	14
4. Use R package car to analyse residuals/influence	16
4.1 plot() method gives 4-panel plots of residuals and Leverage	16
4.3 influencePlot() output	17
4.4 residualPlots()	18
5. Compare Regression Using Z-scores of Predictors	20
5.1 Reproduce Linear Regression on Predictors	20
5.2 Repeat regression with Z-scores of index etfs	21
6. PCA of Explanatory Variables	22
6.1 Extract scaled x matrix (except intercept)	22
6.2 Compute PCA of x	22
6.3 The Screeplot: Barplot of Principal Component Variances	22
7. Regressions on PC variables	26
7.1 Univariate regressions	26
7.2 Regression model on all PC variables	30
7.3 Use PCA regression to recompute regression parameters	31
7.4 Compute regression parameter based on only first 3 pc vars	31
7.5 Compute regression parameter based on significant pc vars	31
7.6 Create table of betas from these three fits	31
7.7 Demonstrate equality of LS and PCA Regression Fitted Values	32

8. Ridge and Lasso Regression Fits	34
8.1 Load glmnet package for ridge/lasso regressions	34
8.2 Define y vector and x matrix for ridge/lasso fits	34
8.3 Plot coefficient trace of ridge regression model	34
8.4 Plot coefficient trace of lasso regression model —	34
8.5 Apply Cross Validation to choose ridge parameters	35
	39
8.6 Apply Cross Validation to choose lasso parameters	39

0. Load Libraries

```
library("quantmod")
```

```
## Loading required package: xts
```

```
## Loading required package: zoo
```

```
##
```

```
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
```

```
##
```

```
##      as.Date, as.Date.numeric
```

```
## Loading required package: TTR
```

```
## Registered S3 method overwritten by 'quantmod':
```

```
##      method          from
```

```
##      as.zoo.data.frame zoo
```

```
library("tseries")
```

```
library("ggplot2")
```

```
library("reshape2")
```

```
library("tidyverse")
```

```
## -- Attaching core tidyverse packages ----- tidyverse 2.0.0 --
```

```
## v dplyr      1.1.4      v readr      2.1.5
```

```
## v forcats    1.0.0      v stringr    1.5.1
```

```
## v lubridate  1.9.3      v tibble     3.2.1
```

```
## v purrr      1.0.2      v tidyr      1.3.1
```

```
## -- Conflicts ----- tidyverse_conflicts() --
```

```
## x dplyr::filter() masks stats::filter()
```

```
## x dplyr::first()  masks xts::first()
```

```
## x dplyr::lag()    masks stats::lag()
```

```
## x dplyr::last()   masks xts::last()
```

```
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library("tidyquant")
```

```
## -- Attaching core tidyquant packages ----- tidyquant 1.0.9 --
```

```
## v PerformanceAnalytics 2.0.4
```

```
## -- Conflicts ----- tidyquant_conflicts() --
```

```
## x zoo::as.Date()          masks base::as.Date()
```

```
## x zoo::as.Date.numeric()  masks base::as.Date.numeric()
```

```
## x dplyr::filter()         masks stats::filter()
```

```
## x dplyr::first()          masks xts::first()
```

```

## x dplyr::lag()           masks stats::lag()
## x dplyr::last()          masks xts::last()
## x PerformanceAnalytics::legend() masks graphics::legend()
## x quantmod::summary()    masks base::summary()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors

library("car")

## Loading required package: carData
##
## Attaching package: 'car'
##
## The following object is masked from 'package:dplyr':
##
##     recode
##
## The following object is masked from 'package:purrr':
##
##     some

library("ggfortify")
library("glmnet")

## Loading required package: Matrix
##
## Attaching package: 'Matrix'
##
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
##
## Loaded glmnet 4.1-8

```

1. Load data into R session

1.1 Daily Stock Price Data from Yahoo

Apply quantmod(sub-package TTR) function: getYahoodata

Returns historical data for any symbol at the website

<http://finance.yahoo.com>

1.1.1 Set start and end date for collection in YYYYMMDD (numeric) format

```
date.start<-"2000-01-01"  
date.end<-"2024-09-20"
```

1.1.2 Input list of ETF symbols

```
options(stringsAsFactors=FALSE)  
list.etf0.labels0<-read.table(file="spymod_symbols_labels.csv",sep=",", header=FALSE)  
list.etf0<-as.character(list.etf0.labels0[,1])
```

```
for (etf0 in list.etf0){  
  #etf0<-"XLB"  
  getSymbols(etf0, from=date.start, to=date.end)  
  print(etf0)  
}
```

```
## [1] "SPY"  
## [1] "MDY"  
## [1] "QQQ"  
## [1] "XLB"  
## [1] "XLV"  
## [1] "XLP"  
## [1] "XLY"  
## [1] "XLE"  
## [1] "XLF"  
## [1] "XLI"  
## [1] "XLK"  
## [1] "XLU"  
## [1] "DIA"
```

1.1.3 Construct matrix of closing prices

```
etf0.pmat0<-matrix(NA, nrow=NROW(SPY),ncol=length(list.etf0))  
dimnames(etf0.pmat0)<-list(dimnames(SPY)[[1]], list.etf0)  
# Create zoo objects of just Adjusted prices  
for (j.etf0 in c(1:length(list.etf0))){  
  #j.etf0<-3  
  j.etf0.name0<-list.etf0[j.etf0]  
  obj0.0<-get(list.etf0[j.etf0])[,6]  
  dimnames(obj0.0)[[2]]<-j.etf0.name0  
  obj0.0.name0<-paste(j.etf0.name0, ".0", sep="")  
  print(obj0.0.name0)
```

```

assign(obj0.0.name0, obj0.0)
}

```

```

## [1] "SPY.0"
## [1] "MDY.0"
## [1] "QQQ.0"
## [1] "XLB.0"
## [1] "XLV.0"
## [1] "XLP.0"
## [1] "XLY.0"
## [1] "XLE.0"
## [1] "XLF.0"
## [1] "XLI.0"
## [1] "XLK.0"
## [1] "XLU.0"
## [1] "DIA.0"

```

```

list.etf0.0<-paste(list.etf0, ".0", sep="")

```

```

pmat0.0<-eval(parse(text=paste("merge(",
                                paste(list.etf0.0, collapse=", "),
                                ")", collapse="")))

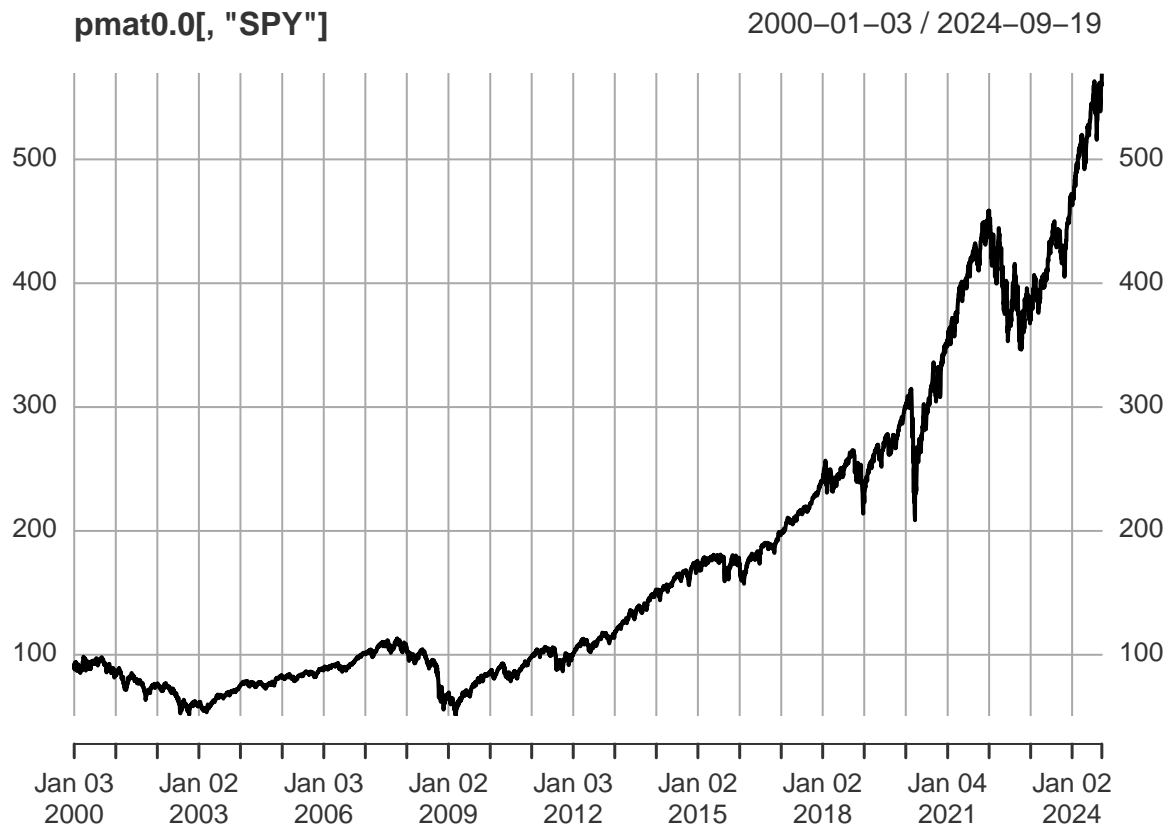
```

1.1.4 Plot time series of daily SPY prices

```

plot(pmat0.0[, "SPY"])

```



```
chartSeries(pmat0.0[, "XLF"])
```



1.2 Weekly Stock Price Data

1.2.1 Create matrix of weekly closing prices

```
pmat0.0.1.weekly<-to.weekly(pmat0.0[,1])
pmat0.0.weekly.coredata<-matrix(NA, nrow=nrow(pmat0.0.1.weekly), ncol=ncol(pmat0.0))

for (jcol0 in c(1:ncol(pmat0.0))) {
  pmat0.0.weekly.coredata[,jcol0]<- to.weekly(pmat0.0[,jcol0])[,4]
}

dim(pmat0.0.weekly.coredata)

## [1] 1290 13

length(time(pmat0.0.1.weekly))

## [1] 1290

as.numeric(time(pmat0.0.1.weekly)[1:5])

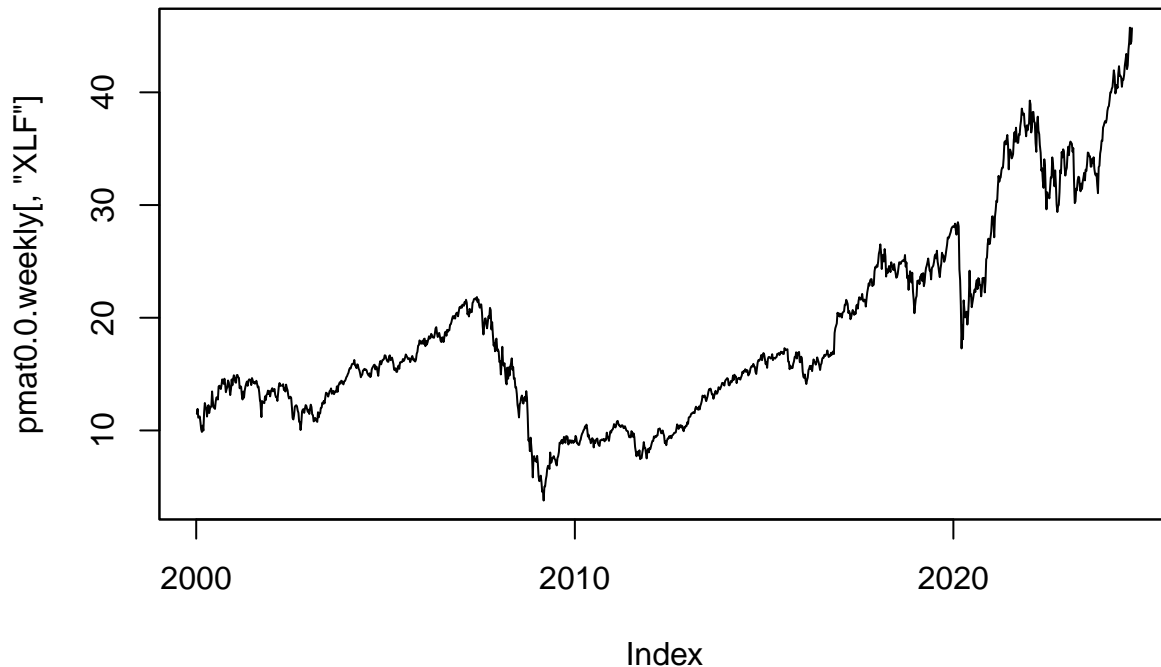
## [1] 10963 10970 10977 10984 10991

pmat0.0.weekly<-zoo(pmat0.0.weekly.coredata, order.by=time(pmat0.0.1.weekly))
```

```
dimnames(pmat0.0.weekly)[[2]]<-dimnames(pmat0.0)[[2]]
```

1.2.2 Create matrix of weekly log returns

```
plot(pmat0.0.weekly[, "XLF"])
```



```
head(pmat0.0.weekly)
```

```
##          SPY      MDY      QQQ      XLB      XLV      XLP      XLY
## 2000-01-07 93.20473 60.77374 76.63247 15.80842 20.71106 13.61024 22.95817
## 2000-01-14 93.98407 61.94606 79.50620 15.29557 22.30756 13.48810 22.64126
## 2000-01-21 92.36543 62.27433 81.95421 14.31485 22.19897 13.02571 21.70228
## 2000-01-28 86.88986 59.03872 73.22660 13.51409 21.02602 12.73779 20.28206
## 2000-02-04 91.18636 61.12545 82.72583 13.48710 21.99260 12.68544 21.08020
## 2000-02-11 88.68837 61.19579 85.04080 12.50639 21.30840 12.04856 20.27032
##          XLE      XLF      XLI      XLK      XLU      DIA
## 2000-01-07 14.88109 11.47027 18.86840 38.36603 11.60620 67.18424
## 2000-01-14 14.87281 11.91383 18.62753 39.77654 11.38846 68.34763
## 2000-01-21 15.41937 11.12786 18.19596 40.34074 11.77116 65.66431
## 2000-01-28 14.14408 11.17456 17.34287 36.81445 11.15752 62.51754
## 2000-02-04 13.94533 11.24460 17.41311 40.71688 11.44784 63.88175
## 2000-02-11 13.63065 10.70765 16.35930 40.90495 11.07835 60.82583
```

```
rmat0.0.weekly<-na.omit(diff(log(pmat0.0.weekly)))
```

Table frequency of return horizon in days. Note that September 11, 2001 corresponds to 11.

```
table(diff(time(rmat0.0.weekly)))
```

```
##
##    3    6    7    8   11
##    1   39 1209   38    1
```

```
cor(rmat0.0.weekly)
```

```
##          SPY          MDY          QQQ          XLB          XLV          XLP          XLY
## SPY 1.0000000 0.9263243 0.8406520 0.8169416 0.8048858 0.7133531 0.8851436
## MDY 0.9263243 1.0000000 0.7857584 0.8350611 0.7038067 0.6111278 0.8456268
## QQQ 0.8406520 0.7857584 1.0000000 0.5875577 0.6467431 0.4414164 0.7333355
## XLB 0.8169416 0.8350611 0.5875577 1.0000000 0.6174499 0.5939059 0.7566495
## XLV 0.8048858 0.7038067 0.6467431 0.6174499 1.0000000 0.6477538 0.6718103
## XLP 0.7133531 0.6111278 0.4414164 0.5939059 0.6477538 1.0000000 0.6295340
## XLY 0.8851436 0.8456268 0.7333355 0.7566495 0.6718103 0.6295340 1.0000000
## XLE 0.6713568 0.7040840 0.4252871 0.7034906 0.4825471 0.4657258 0.5259730
## XLF 0.8404447 0.8232554 0.5853946 0.7296197 0.6391774 0.5685929 0.7733551
## XLI 0.9091361 0.9002694 0.7081745 0.8512077 0.7054342 0.6662992 0.8287770
## XLK 0.8542516 0.7790898 0.9640271 0.6072690 0.6348142 0.4588329 0.7419045
## XLU 0.6166184 0.5736053 0.3873535 0.5073341 0.5512262 0.6180849 0.4970912
## DIA 0.9556731 0.8819195 0.7305151 0.8294259 0.7928105 0.7558891 0.8572590
##          XLE          XLF          XLI          XLK          XLU          DIA
## SPY 0.6713568 0.8404447 0.9091361 0.8542516 0.6166184 0.9556731
## MDY 0.7040840 0.8232554 0.9002694 0.7790898 0.5736053 0.8819195
## QQQ 0.4252871 0.5853946 0.7081745 0.9640271 0.3873535 0.7305151
## XLB 0.7034906 0.7296197 0.8512077 0.6072690 0.5073341 0.8294259
## XLV 0.4825471 0.6391774 0.7054342 0.6348142 0.5512262 0.7928105
## XLP 0.4657258 0.5685929 0.6662992 0.4588329 0.6180849 0.7558891
## XLY 0.5259730 0.7733551 0.8287770 0.7419045 0.4970912 0.8572590
## XLE 1.0000000 0.5858358 0.6812257 0.4344606 0.4907897 0.6729488
## XLF 0.5858358 1.0000000 0.8241748 0.5965308 0.4832331 0.8299173
## XLI 0.6812257 0.8241748 1.0000000 0.7224324 0.5653708 0.9232702
## XLK 0.4344606 0.5965308 0.7224324 1.0000000 0.4106987 0.7606382
## XLU 0.4907897 0.4832331 0.5653708 0.4106987 1.0000000 0.6176373
## DIA 0.6729488 0.8299173 0.9232702 0.7606382 0.6176373 1.0000000
```

```
dim(rmat0.0.weekly)
```

```
## [1] 1289    13
```

1.2.3 Create separate objects for sector etfs and index etfs

```
rmat0.0.weekly.sectoretfs<-rmat0.0.weekly[,c(4:12)]
list.sectoretfs<-dimnames(rmat0.0.weekly.sectoretfs)[[2]]
list.sectoretfs.labels<-list.etf0.labels0[4:12,2]
list.sectoretfs.labels.0<-paste(list.sectoretfs.labels,"(",list.sectoretfs,")",sep="")

rmat0.0.weekly.indexetfs<-rmat0.0.weekly[,c(1:3, 13)]

list.indexetfs<-dimnames(rmat0.0.weekly.indexetfs)[[2]]

head(rmat0.0.weekly.indexetfs)
```

```
##          SPY          MDY          QQQ          DIA
## 2000-01-14 0.008326775 0.019106116 0.03681408 0.01716810
## 2000-01-21 -0.017372494 0.005285418 0.03032571 -0.04005133
## 2000-01-28 -0.061111441 -0.053355880 -0.11260191 -0.04910840
## 2000-02-04 0.048263996 0.034734888 0.12197314 0.02158665
## 2000-02-11 -0.027776598 0.001150073 0.02759928 -0.04901918
```



```
## 2000-02-18 -0.024636191 -0.013889208 -0.01449785 -0.02053326
list.sectoretfs<-dimnames(rmat0.0.weekly.sectoretfs)[[2]]
list.sectoretfs.labels<-list.etf0.labels0[4:12,2]
list.sectoretfs.labels.0<-paste(list.sectoretfs.labels,"(",list.sectoretfs,")",sep="")

rmat0.0.weekly.indexetfs<-rmat0.0.weekly[,c(1:3, 13)]

list.indexetfs<-dimnames(rmat0.0.weekly.indexetfs)[[2]]
```

1.3 Save data objects for casestudy in R data file

```
list.obj.tosave0<-c(
  "pmat0.0.weekly",
  "rmat0.0.weekly.sectoretfs",
  "rmat0.0.weekly.indexetfs",
  "list.sectoretfs",
  "list.sectoretfs.labels",
  "list.sectoretfs.labels.0",
  "list.indexetfs")

save(file="casestudy_1_0_etfs.RData", list=list.obj.tosave0)
```

2. Analysis of returns for specific period (2013-2024)

2.1 Subset data objects for specific period

```
period0.startyear0<-"2013"
period0.endyear0<-"2024"
period0.startdate0<-as.Date(paste(period0.startyear0,"-01-01",sep=""))
period0.enddate0<-as.Date(paste(period0.endyear0,"-12-31",sep=""))
period0.label0<-paste("Period: ", period0.startyear0, "-" , period0.endyear0,sep="")

rmat0.0.weekly.sectoretfs.period0<-window(rmat0.0.weekly.sectoretfs,
                                         start = period0.startdate0,
                                         end=period0.enddate0)

rmat0.0.weekly.indexetfs.period0<-window(rmat0.0.weekly.indexetfs,
                                         start = period0.startdate0,
                                         end=period0.enddate0)

tail(rmat0.0.weekly.sectoretfs.period0)
```

##		XLB	XLV	XLP	XLV	XLE
##	2024-08-16	0.02244277	0.019085932	0.017036621	0.049188915	0.0117857438
##	2024-08-23	0.02356905	0.016736262	0.016384650	0.025553898	-0.0008846825
##	2024-08-30	0.01657698	0.011194744	0.008153447	-0.004000112	0.0097980571
##	2024-09-06	-0.04768237	-0.020891006	0.005800569	-0.025548052	-0.0594680988
##	2024-09-13	0.03067392	0.014253292	0.011263029	0.053790186	-0.0048951823
##	2024-09-19	0.02187566	-0.001538103	-0.015368279	0.023921903	0.0386184091
##		XLV	XLI	XLK	XLU	
##	2024-08-16	0.031800263	0.02141377	0.07389253	0.011212570	
##	2024-08-23	0.014966262	0.01802974	0.01149650	0.013079003	
##	2024-08-30	0.029058301	0.01672209	-0.01634106	0.011469413	
##	2024-09-06	-0.032214287	-0.04329417	-0.07739868	-0.004993404	
##	2024-09-13	0.004954907	0.03650103	0.07789781	0.034059585	
##	2024-09-19	0.026165580	0.02682535	0.01432260	-0.006642900	

```
tail(rmat0.0.weekly.indexetfs.period0)
```

##		SPY	MDY	QQQ	DIA
##	2024-08-16	0.039221464	0.026451262	0.053219670	0.02971464
##	2024-08-23	0.014009032	0.028103817	0.010408147	0.01243647
##	2024-08-30	0.002753516	-0.001537826	-0.007801207	0.01057932
##	2024-09-06	-0.042251072	-0.049590953	-0.059652666	-0.02861231
##	2024-09-13	0.039284067	0.031976940	0.057698098	0.02555637
##	2024-09-19	0.015834531	0.029501289	0.016731358	0.01547228

2.2 Define y0 equal to weekly returns of specific sector etf (XLB)

```
y0.name="XLB (Materials ETF)"
y0.symbol="XLB"
y0<-rmat0.0.weekly.sectoretfs.period0$XLB
```

The following R code can be copied above to change the choice of sector etf

```
if (FALSE) {
# 1.1 Change to XLK Technology:
```

```

# Define y0 equal to weekly returns of sector etf XLK
y0.name="XLK (Technology ETF)"
y0.symbol="XLK"
y0<-rmat0.0.weekly.sectoretfs.period0$XLK

# 1.2 Change to XLI Industrials:
# Define y0 equal to weekly returns of sector etf XLI
y0.name="XLI (Industrials ETF)"
y0.symbol="XLI"
y0<-rmat0.0.weekly.sectoretfs.period0$XLI

}

```

2.3 Define df0 equal to data frame with y0 and index etfs

```
df0<-data.frame(cbind(y0,rmat0.0.weekly.indexetfs.period0))
```

2.4 Print out summary statistics for weekly returns

2.4.1 Annual means, volatilities

```
apply(df0,2,summary)
```

```

##              y0              SPY              MDY              QQQ              DIA
## Min.      -0.140575013 -0.157188876 -0.202701201 -0.119356913 -0.187242297
## 1st Qu.   -0.011895963 -0.007898187 -0.010483289 -0.010700617 -0.007719642
## Median    0.003261342  0.004176944  0.003299164  0.004865104  0.003425429
## Mean      0.001927505  0.002634948  0.002100222  0.003474893  0.002332767
## 3rd Qu.   0.016121934  0.014691272  0.015611675  0.019312547  0.013328994
## Max.      0.187938181  0.114145841  0.172891121  0.091101987  0.119923264

```

```

mean.annual<-apply(df0,2,mean)*52
vol.annual<-sqrt(52*apply(df0,2,var))
data.frame(mean=mean.annual, vol=vol.annual)

```

```

##          mean          vol
## y0  0.1002303 0.1963206
## SPY 0.1370173 0.1607707
## MDY 0.1092116 0.1957695
## QQQ 0.1806944 0.1893825
## DIA 0.1213039 0.1628827

```

2.4.2 Pairwise correlations and pairs plots

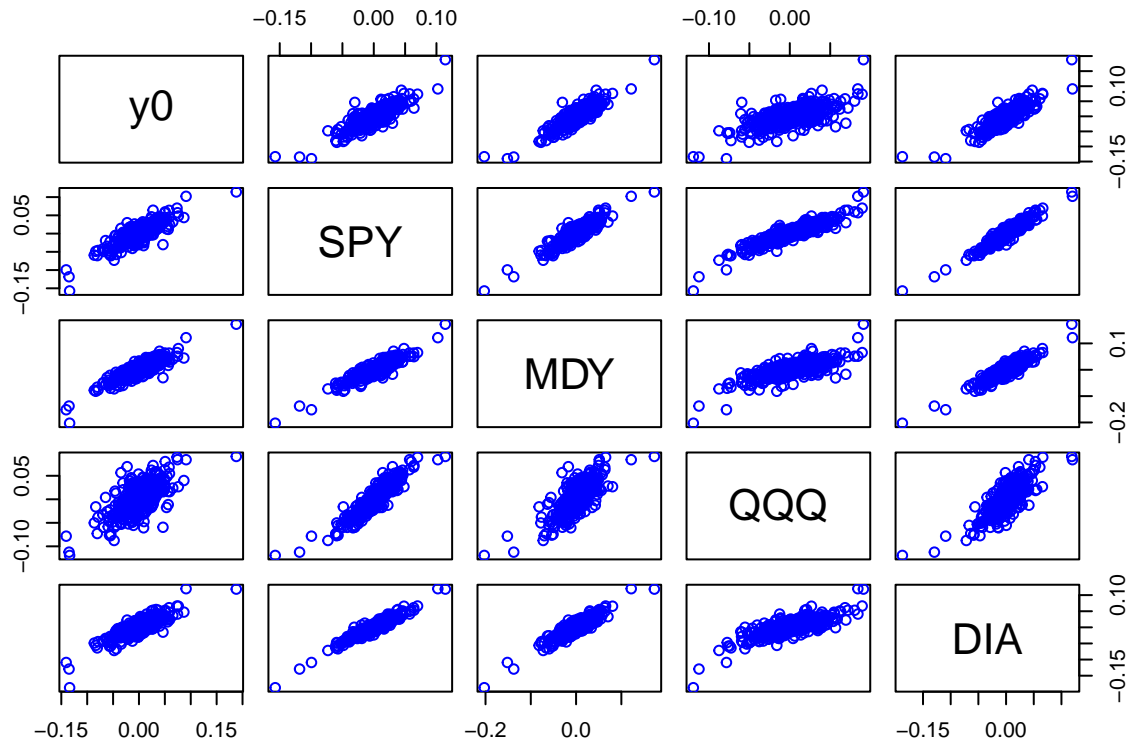
```
cor(df0)
```

```

##              y0              SPY              MDY              QQQ              DIA
## y0  1.0000000 0.8449814 0.8825664 0.6783256 0.8551028
## SPY 0.8449814 1.0000000 0.9175269 0.9181595 0.9540902
## MDY 0.8825664 0.9175269 1.0000000 0.7689487 0.9092158
## QQQ 0.6783256 0.9181595 0.7689487 1.0000000 0.7977512
## DIA 0.8551028 0.9540902 0.9092158 0.7977512 1.0000000

```

```
pairs(df0,col='blue')
```



2.5 Compute/plot cumulative returns

```
# Convert df0 to cumulative return series  
# All starting at 1.
```

```
df0cumret<-data.frame(exp(apply(log(1+df0),2,cumsum)))  
names(df0cumret)
```

```
## [1] "y0" "SPY" "MDY" "QQQ" "DIA"
```

```
names(df0cumret)[1]<-y0.symbol
```

```
# Rename y0 to actual symbol for plotting
```

```
# Add Date variable (after computing cumulative returns)
```

```
df0cumret$Date<-time(rmat0.0.weekly.indexetfs.period0)
```

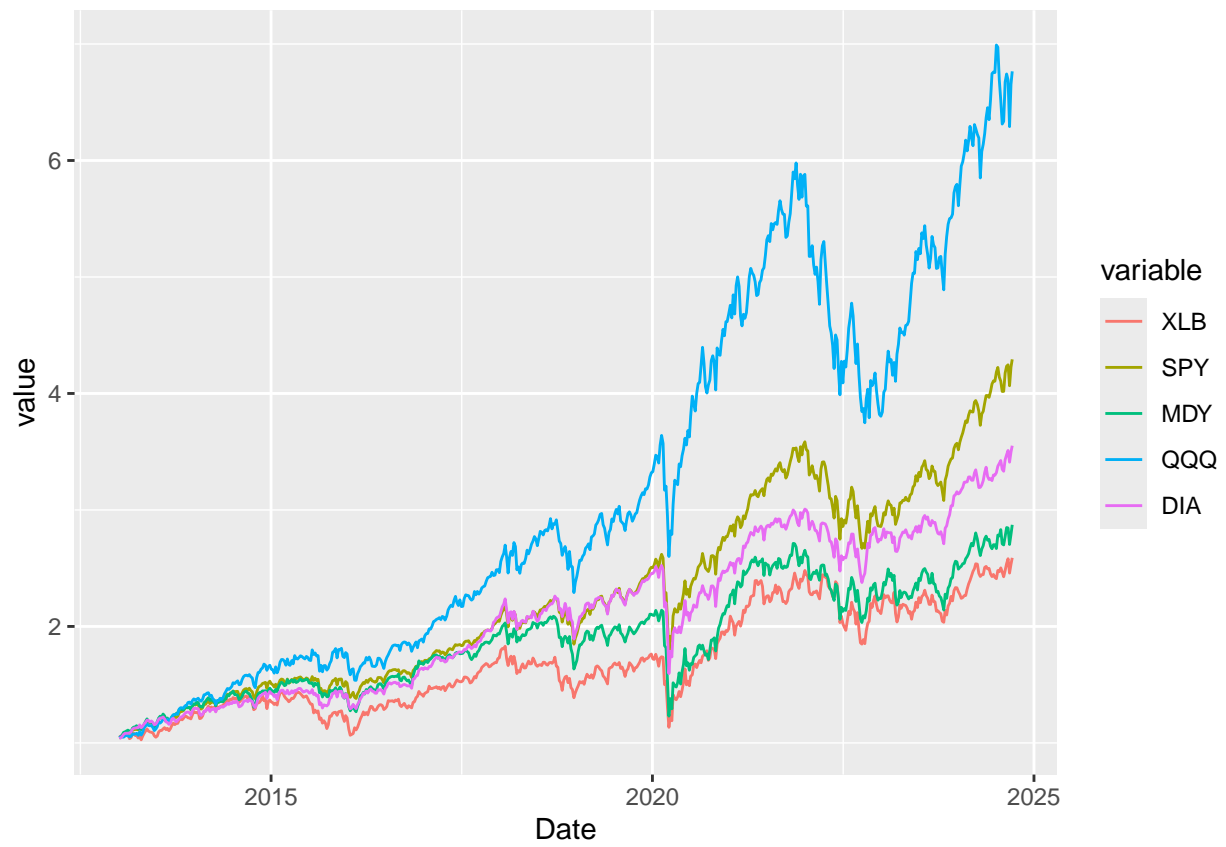
```
# Apply melt function from reshape2 package for using ggplot
```

```
meltdf0cumret <- melt(df0cumret,id="Date")
```

```
#names(meltdf)
```

```
#meltdf$variable
```

```
gg0<-ggplot(meltdf0cumret,aes(x=Date,y=value,colour=variable,group=variable)) +  
  geom_line()  
print(gg0)
```



3. Regression of Sector ETF on Index ETFs

3.1 Fit regression of y0 on index etf returns

```
fit<-lm(y0 ~., data=df0); fit.summary<-summary(fit)

fit.summary

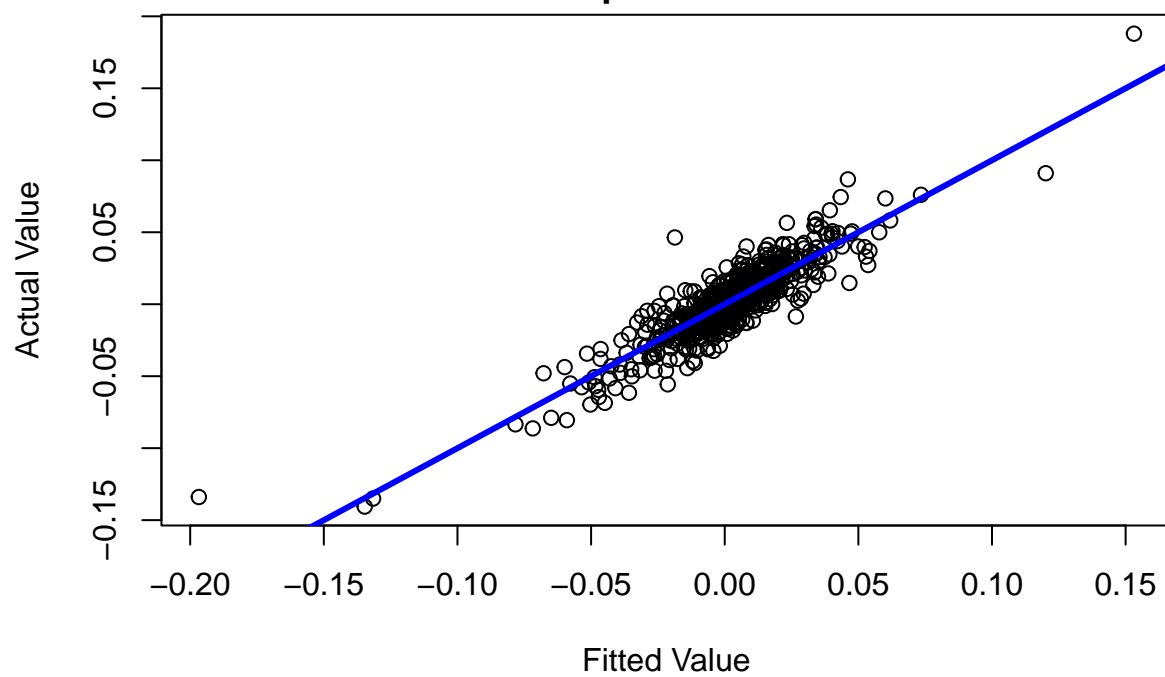
##
## Call:
## lm(formula = y0 ~ ., data = df0)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.035159 -0.007275 -0.000405  0.007083  0.065036
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.0001203  0.0004922  -0.244   0.807
## SPY          0.8491202  0.1723861   4.926 1.09e-06 ***
## MDY          0.5002258  0.0510879   9.791 < 2e-16 ***
## QQQ         -0.3842899  0.0676074  -5.684 2.04e-08 ***
## DIA          0.0408147  0.0959266   0.425   0.671
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01205 on 607 degrees of freedom
## Multiple R-squared:  0.8053, Adjusted R-squared:  0.804
## F-statistic: 627.7 on 4 and 607 DF,  p-value: < 2.2e-16
# Note list components of output from summary()
names(fit.summary)

## [1] "call"          "terms"          "residuals"      "coefficients"
## [5] "aliased"        "sigma"          "df"             "r.squared"
## [9] "adj.r.squared" "fstatistic"     "cov.unscaled"
# print method for fit.summary is condensed
```

3.2 Plot Actual versus Fitted Values

```
plot(fit$fitted.values, df0$y0,
     xlab="Fitted Value",
     ylab = "Actual Value",
     main=paste(c("Regression Model","\n", y0.name,
                  "\nR-Squared =",
                  as.character(round(fit.summary$r.squared,digits=2))),
             collapse=""),
     abline(a=0,b=1,col="blue",lwd=3))
```

Regression Model
XLB (Materials ETF)
R-Squared =0.81

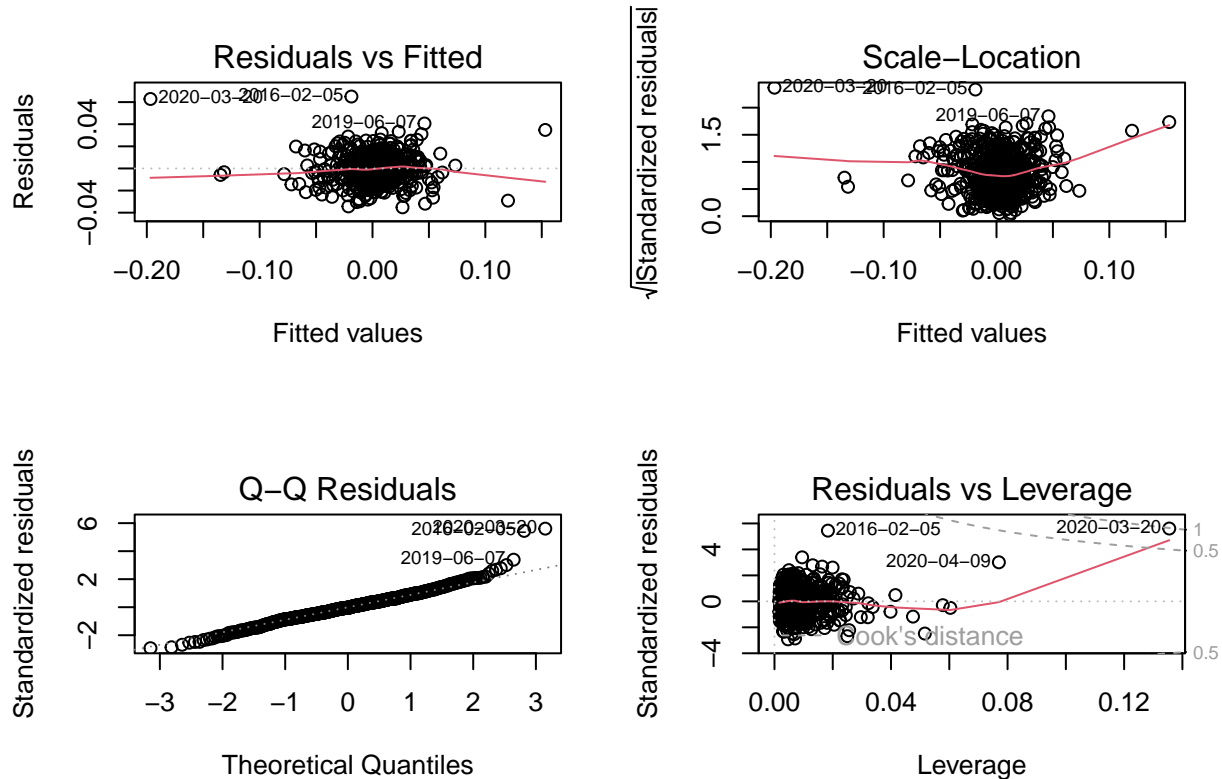


4. Use R package car to analyse residuals/influence

```
library(car)
```

4.1 plot() method gives 4-panel plots of residuals and Leverage

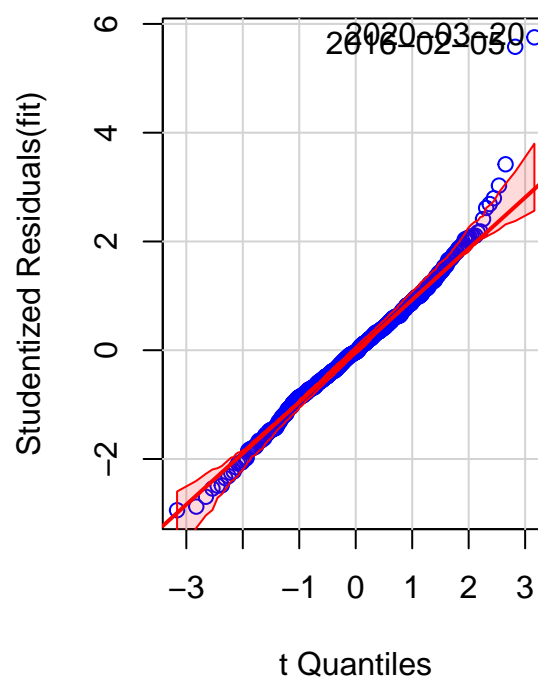
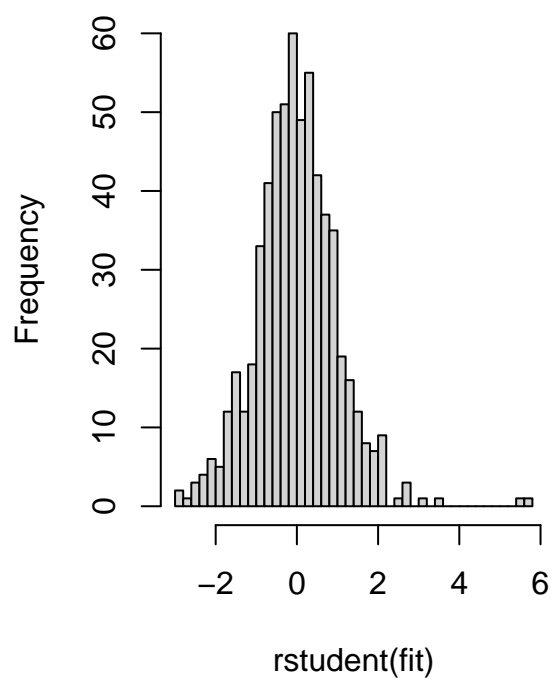
```
oldpar=par(no.readonly=TRUE)
layout(matrix(c(1,2,3,4),2,2)) # optional 4 graphs/page
plot(fit) ; par(oldpar,no.readonly=TRUE)
```



4.2 studres() Studentized residuals and qqPlot() -

```
# plot histogram and t-dist QQ plot
par(mfcol=c(1,2));
hist(rstudent(fit),breaks=50);
qqPlot(fit,col="blue",col.lines="red")
```


Histogram of rstudent(fit)



```
## 2016-02-05 2020-03-20
```

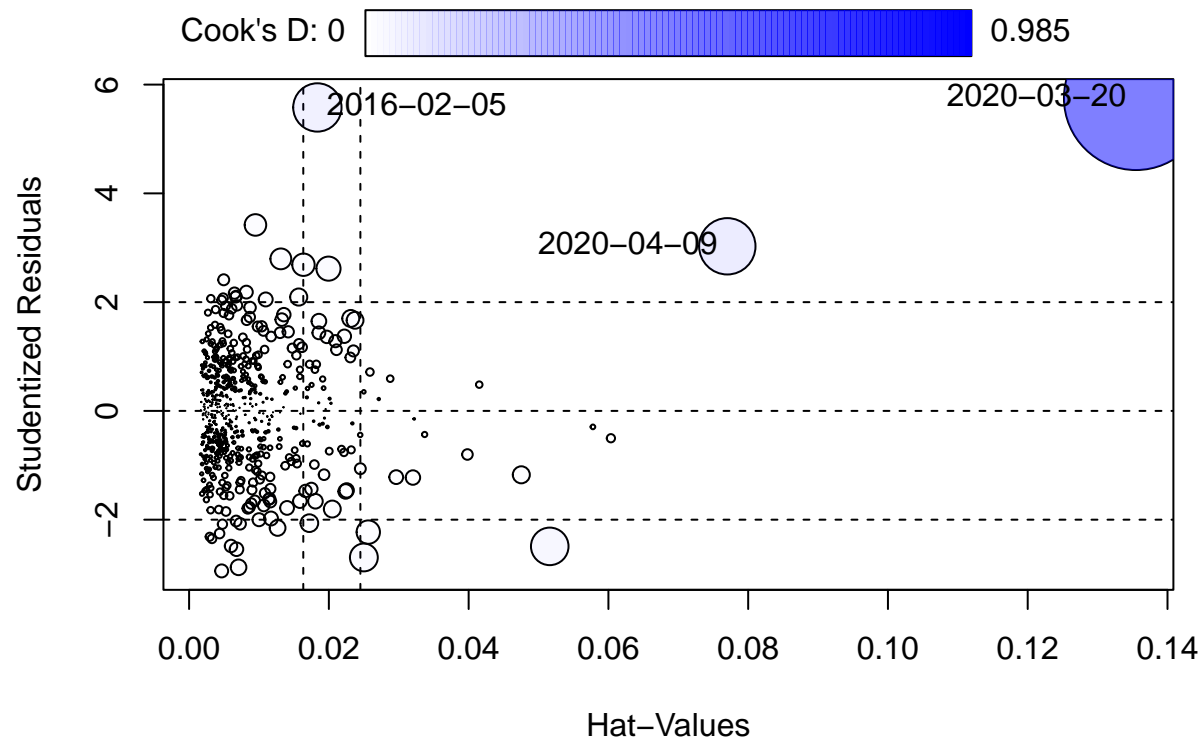
```
##          162          377
```

```
#
```

4.3 influencePlot() output

```
par(mfcol=c(1,1))
```

```
influencePlot(lm(y0~., data=df0))
```

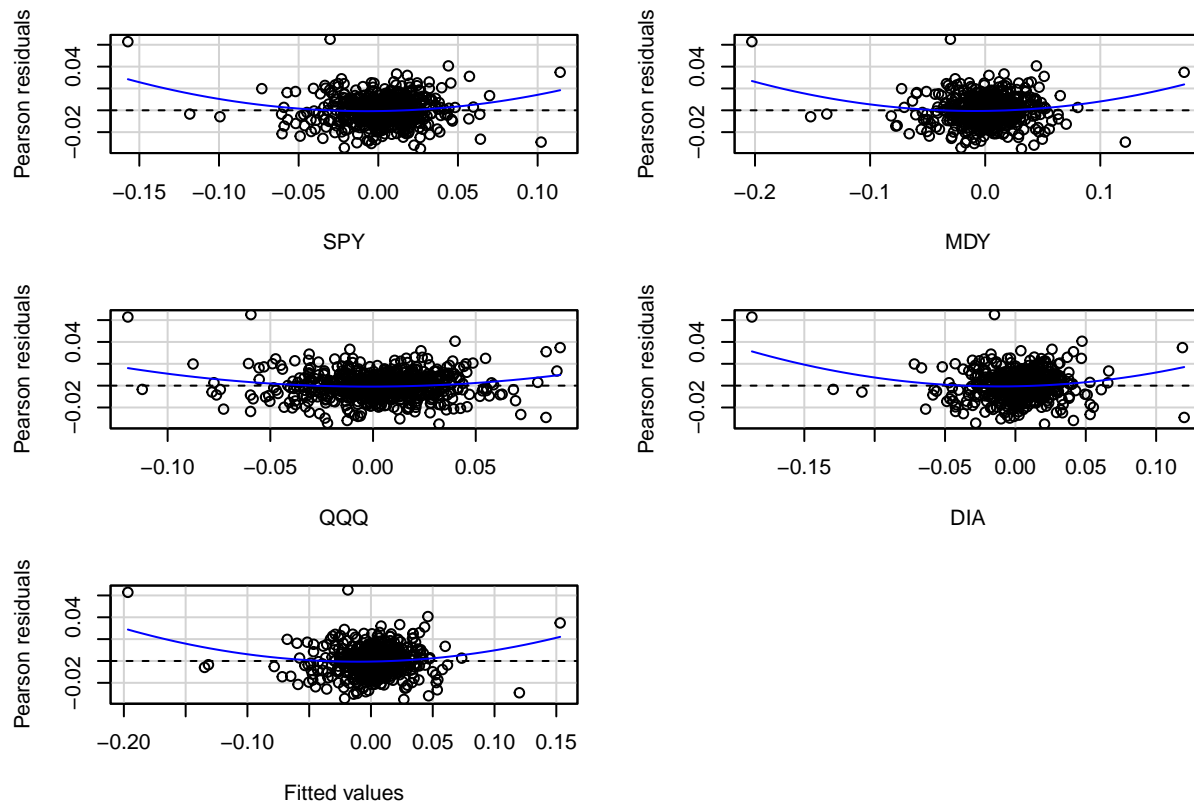


##		StudRes	Hat	CookD
##	2016-02-05	5.579939	0.0183555	0.1109322
##	2020-03-20	5.753398	0.1354922	0.9854690
##	2020-04-09	3.026982	0.0770149	0.1508789

4.4 residualPlots()

Visually test for curvature in linear regression terms.

```
residualPlots(lm(y0~., data=df0), tests=FALSE)
```



Print out test results

```
# Test results (for curvature in plots)
residualPlots(lm(y0~., data=df0), tests=TRUE, plot=FALSE)

##           Test stat Pr(>|Test stat|)
## SPY          4.0447    5.916e-05 ***
## MDY          3.8808    0.0001155 ***
## QQQ          3.5042    0.0004917 ***
## DIA          3.9770    7.823e-05 ***
## Tukey test    3.8962    9.770e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

5. Compare Regression Using Z-scores of Predictors

5.1 Reproduce Linear Regression on Predictors

```
# df0: data frame with security and indexes
# y0
# Regress y0 on indexes

# Add arguments to get x matrix and y
fit<-lm(y0 ~., data=df0,x=TRUE,y=TRUE);
fit.summary<-summary(fit)
fit.summary

##
## Call:
## lm(formula = y0 ~ ., data = df0, x = TRUE, y = TRUE)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.035159 -0.007275 -0.000405  0.007083  0.065036
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.0001203  0.0004922  -0.244   0.807
## SPY          0.8491202  0.1723861   4.926 1.09e-06 ***
## MDY          0.5002258  0.0510879   9.791 < 2e-16 ***
## QQQ         -0.3842899  0.0676074  -5.684 2.04e-08 ***
## DIA          0.0408147  0.0959266   0.425   0.671
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01205 on 607 degrees of freedom
## Multiple R-squared:  0.8053, Adjusted R-squared:  0.804
## F-statistic: 627.7 on 4 and 607 DF,  p-value: < 2.2e-16

fit$coefficients

##      (Intercept)          SPY          MDY          QQQ          DIA
## -0.0001203132  0.8491202453  0.5002258206 -0.3842899053  0.0408146633

# Output argument x has matrix of predictor variables
head(fit$x)

##      (Intercept)          SPY          MDY          QQQ          DIA
## 2013-01-04      1 0.044280959 0.049746180 0.043715347 3.793304e-02
## 2013-01-11      1 0.004771449 0.001197091 0.009411122 4.911085e-03
## 2013-01-18      1 0.008530408 0.015020600 -0.002829205 1.118540e-02
## 2013-01-25      1 0.012860996 0.022542625 -0.001044164 1.842000e-02
## 2013-02-01      1 0.006567076 0.004198745 0.009802534 8.048130e-03
## 2013-02-08      1 0.003696383 0.007057780 0.004865312 -7.208728e-05
#
```

5.2 Repeat regression with Z-scores of index etfs

```
# The R function scale outputs z scores of data vectors
df0.zscore<-data.frame(cbind(y0,apply(rmat0.0.weekly.indexetfs.period0,2,scale)))
fit<-lm(y0 ~., data=df0.zscore, x=TRUE,y=TRUE); fit.summary<-summary(fit)
fit.summary
```

```
##
## Call:
## lm(formula = y0 ~ ., data = df0.zscore, x = TRUE, y = TRUE)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.035159 -0.007275 -0.000405  0.007083  0.065036
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.0019275  0.0004872   3.956 8.51e-05 ***
## SPY          0.0189310  0.0038433   4.926 1.09e-06 ***
## MDY          0.0135803  0.0013870   9.791 < 2e-16 ***
## QQQ         -0.0100925  0.0017755  -5.684 2.04e-08 ***
## DIA          0.0009219  0.0021668   0.425  0.671
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01205 on 607 degrees of freedom
## Multiple R-squared:  0.8053, Adjusted R-squared:  0.804
## F-statistic: 627.7 on 4 and 607 DF,  p-value: < 2.2e-16
```

Note the two regressions have identical results for t-stats, R-Square.

6. PCA of Explanatory Variables

6.1 Extract scaled x matrix (except intercept)

```
x=fit$x[,-1]
cor(x)

##           SPY           MDY           QQQ           DIA
## SPY 1.0000000 0.9175269 0.9181595 0.9540902
## MDY 0.9175269 1.0000000 0.7689487 0.9092158
## QQQ 0.9181595 0.7689487 1.0000000 0.7977512
## DIA 0.9540902 0.9092158 0.7977512 1.0000000

var(x)

##           SPY           MDY           QQQ           DIA
## SPY 1.0000000 0.9175269 0.9181595 0.9540902
## MDY 0.9175269 1.0000000 0.7689487 0.9092158
## QQQ 0.9181595 0.7689487 1.0000000 0.7977512
## DIA 0.9540902 0.9092158 0.7977512 1.0000000
```

6.2 Compute PCA of x

The R function `princomp()` is used below. Note that other functions performing PCA are available, e.g. `prcomp()`.

```
#library(ggplot2)
#library(ggfortify)
x.primcomp<-princomp(x)
summary(x.primcomp)

## Importance of components:
##           Comp.1      Comp.2      Comp.3      Comp.4
## Standard deviation  1.9053417 0.51045017 0.30208725 0.1064005
## Proportion of Variance 0.9090671 0.06524646 0.02285152 0.0028349
## Cumulative Proportion 0.9090671 0.97431358 0.99716510 1.0000000
```

Note available output from `princomp()` is a named list:

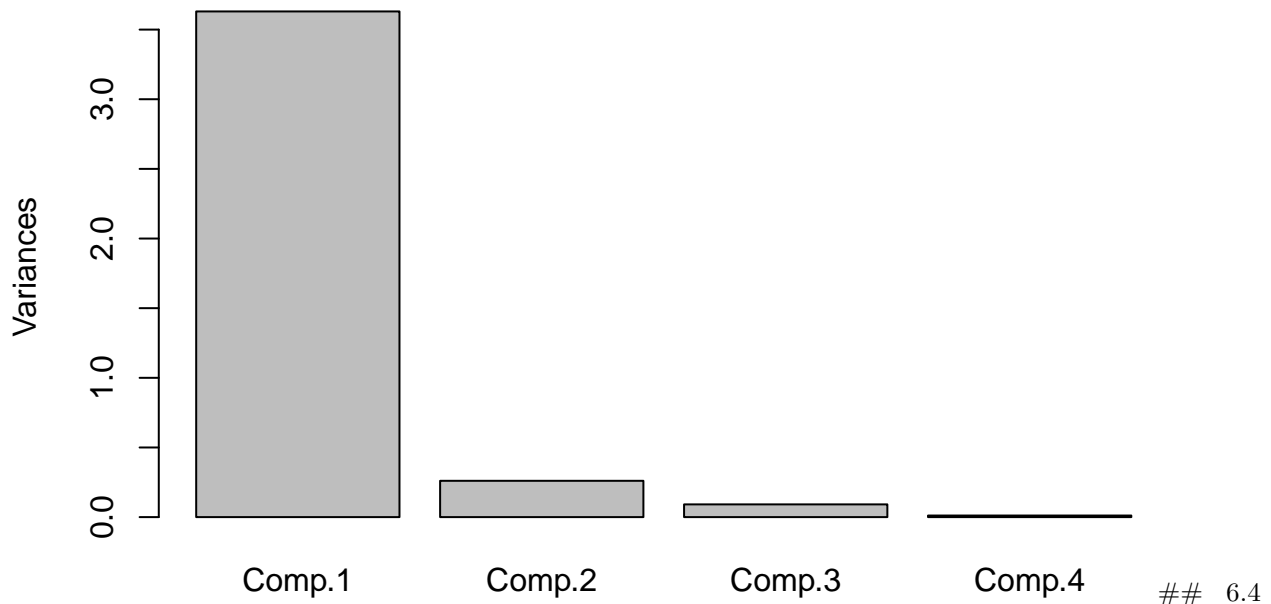
```
names(x.primcomp)

## [1] "sdev"      "loadings" "center"   "scale"    "n.obs"    "scores"   "call"
```

6.3 The Screeplot: Barplot of Principal Component Variances

```
screeplot(x.primcomp)
```

x.princomp



Loadings of Principal Components

Printing out the loadings matrix (all values and with cutoff):

```
# Print out loadings
```

```
x.princomp$loadings
```

```
##
```

```
## Loadings:
```

```
##      Comp.1 Comp.2 Comp.3 Comp.4
```

```
## SPY  0.521      0.145  0.837
```

```
## MDY  0.495 -0.497 -0.699 -0.140
```

```
## QQQ  0.479  0.792 -0.155 -0.346
```

```
## DIA  0.504 -0.346  0.683 -0.400
```

```
##
```

```
##              Comp.1 Comp.2 Comp.3 Comp.4
```

```
## SS loadings      1.00  1.00  1.00  1.00
```

```
## Proportion Var   0.25  0.25  0.25  0.25
```

```
## Cumulative Var   0.25  0.50  0.75  1.00
```

```
# Get all values with no cutoffs
```

```
print(x.princomp$loadings,cutoff=0.)
```

```
##
```

```
## Loadings:
```

```
##      Comp.1 Comp.2 Comp.3 Comp.4
```

```
## SPY  0.521  0.079  0.145  0.837
```

```
## MDY  0.495 -0.497 -0.699 -0.140
```

```
## QQQ  0.479  0.792 -0.155 -0.346
```

```
## DIA  0.504 -0.346  0.683 -0.400
```

```
##
```

```
##              Comp.1 Comp.2 Comp.3 Comp.4
```

```
## SS loadings      1.00  1.00  1.00  1.00
```

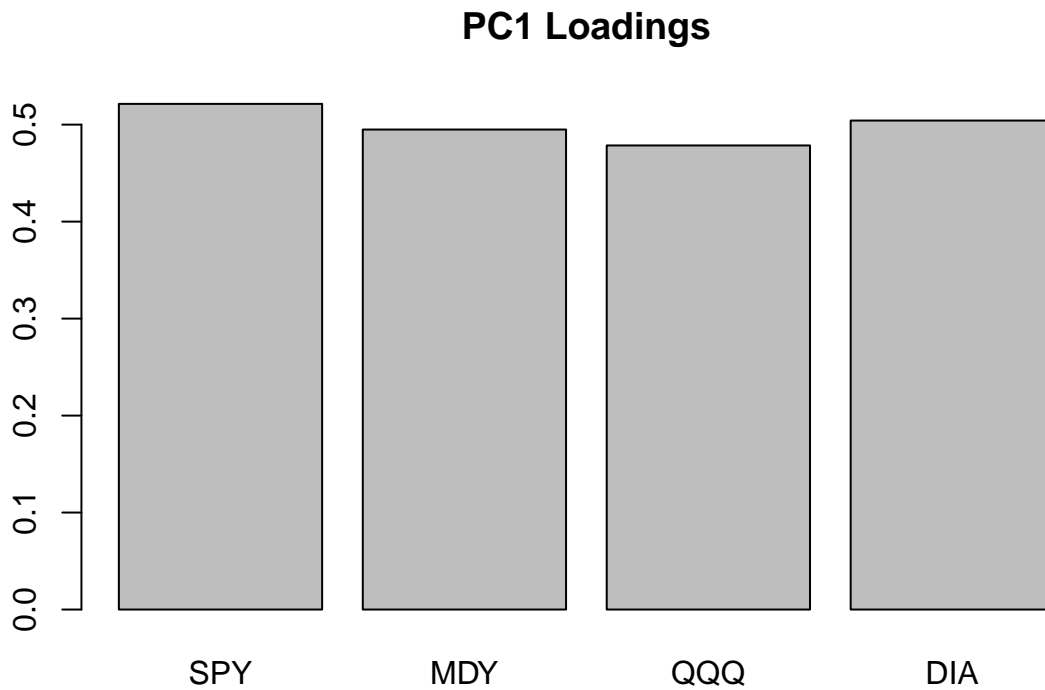
```
## Proportion Var   0.25  0.25  0.25  0.25
```

```
## Cumulative Var    0.25    0.50    0.75    1.00
```

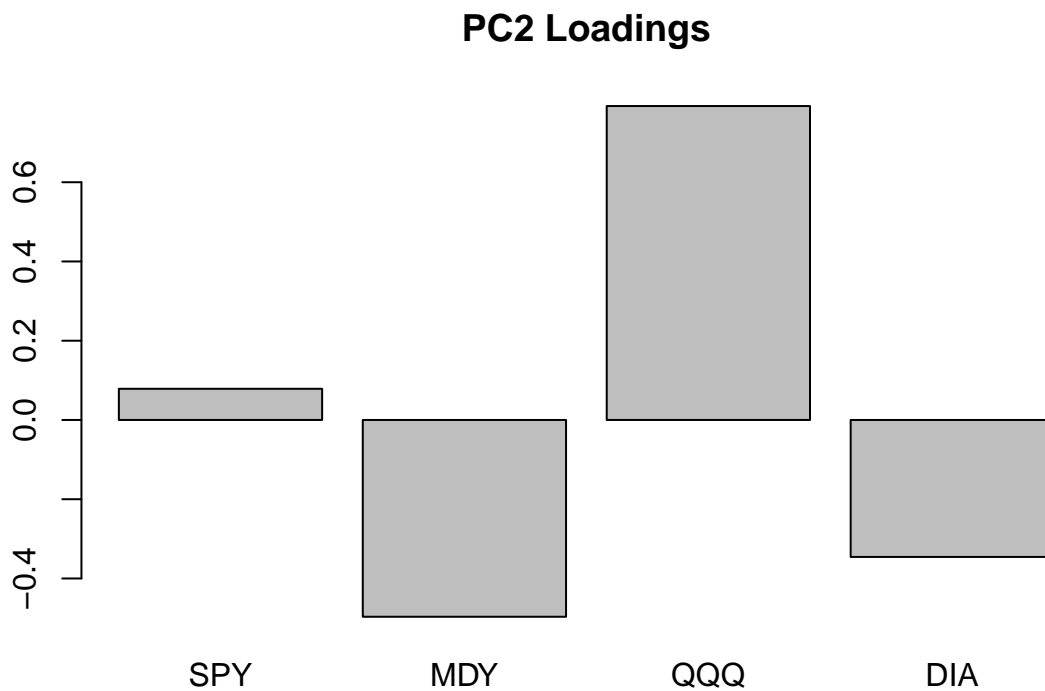
Plotting the loadings of each Principal Component variable

```
# Plot loadings of PC1-PC4
```

```
barplot(x.princomp$loadings[,1], main="PC1 Loadings")
```

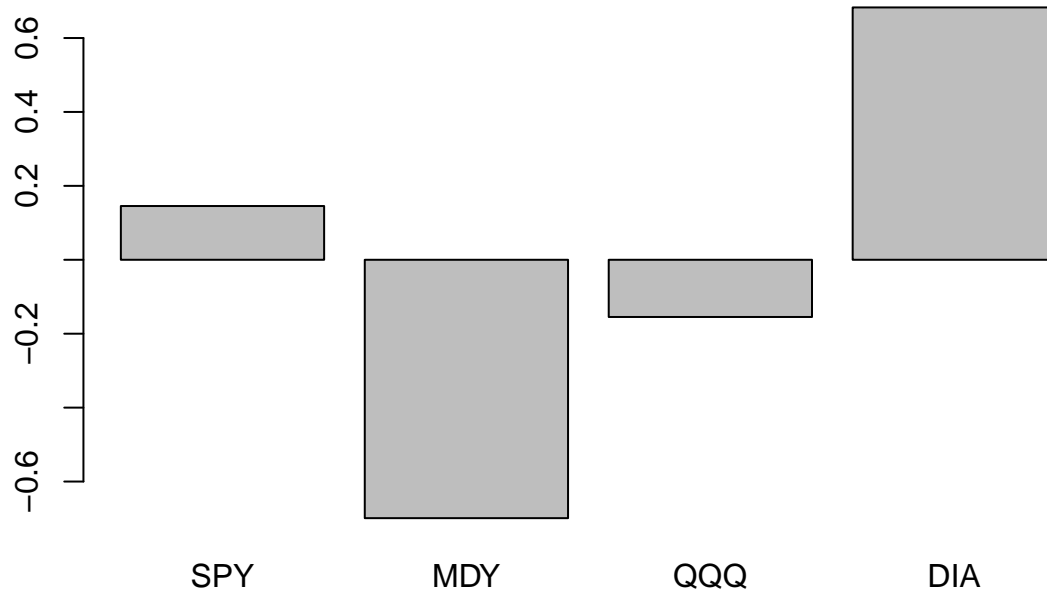


```
barplot(x.princomp$loadings[,2], main="PC2 Loadings")
```



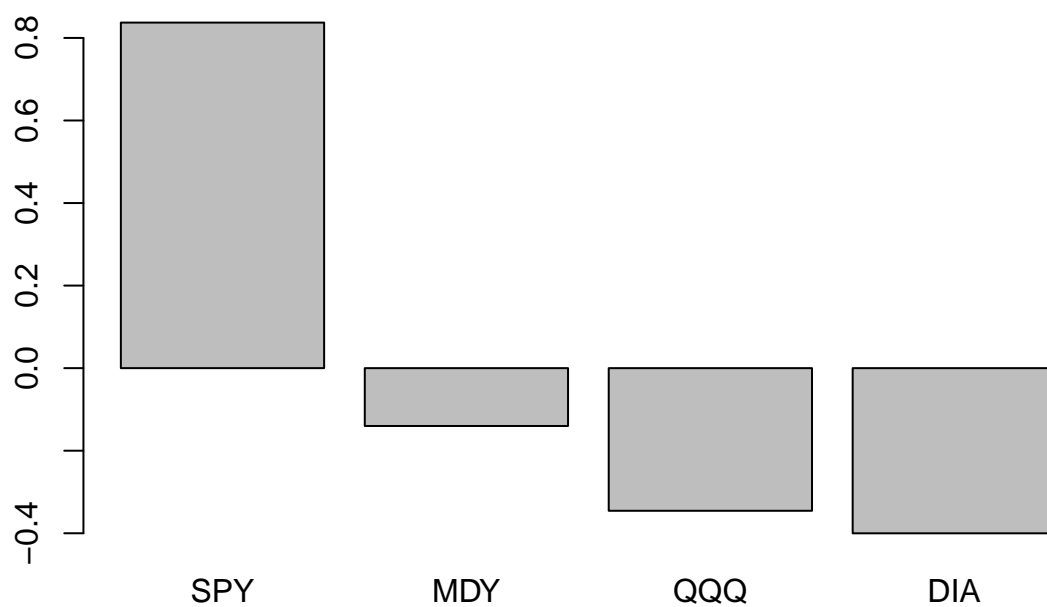
```
barplot(x.princomp$loadings[,3], main="PC3 Loadings")
```


PC3 Loadings



```
barplot(x.princomp$loadings[,4], main="PC4 Loadings")
```

PC4 Loadings



7. Regressions on PC variables

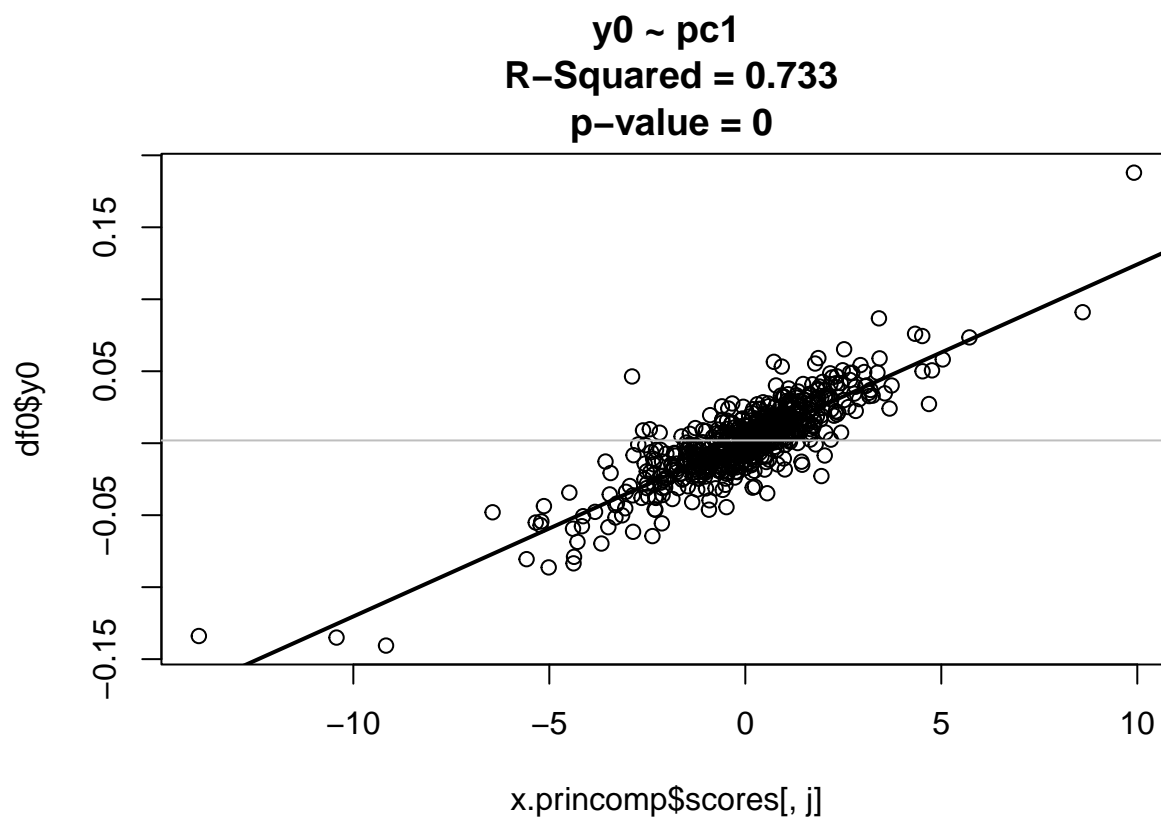
7.1 Univariate regressions

```
# Loop over all PC Variables
for (j in 1:NCOL(x.princomp$scores)){
  lm.j<-lm(df0$y0 ~ x.princomp$scores[,j])
  lm.j.summary<-summary(lm.j)

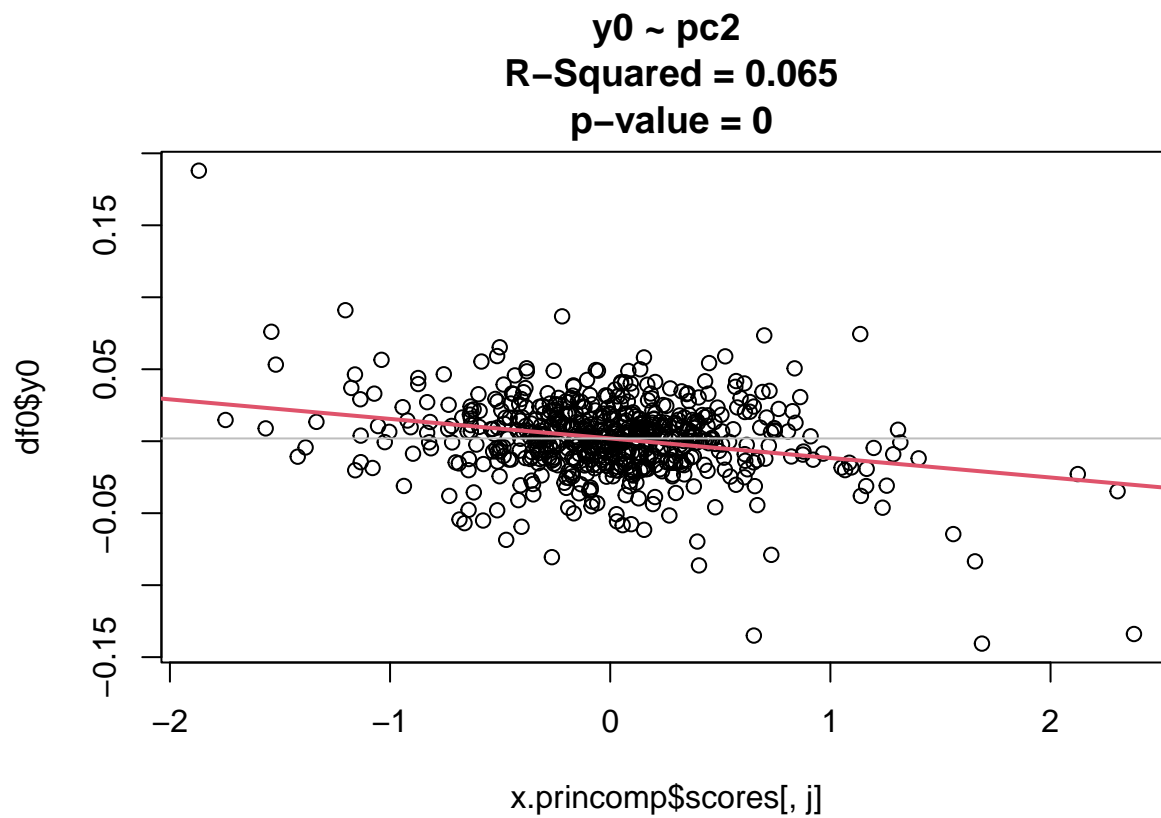
  # Use lsfit() to compute p-value s
  lsfit.j<-lsfit(x=x.princomp$scores[,j], y=y0)
  lsfit.j.print<-ls.print(lsfit.j)

  plot(x.princomp$scores[,j], df0$y0,main=c(" \n \n "))
  abline(lm.j, col=j,lwd=2)
  abline(h=mean(df0$y0),col='gray')
  title(main=paste(c("y0 ~ pc",as.character(j),
    "\n R-Squared = ", as.character(round(lm.j.summary$r.squared,digits=3)),
    "\n p-value = ", as.character(lsfit.j.print$summary[1,6])),
    collapse=""))
}
```

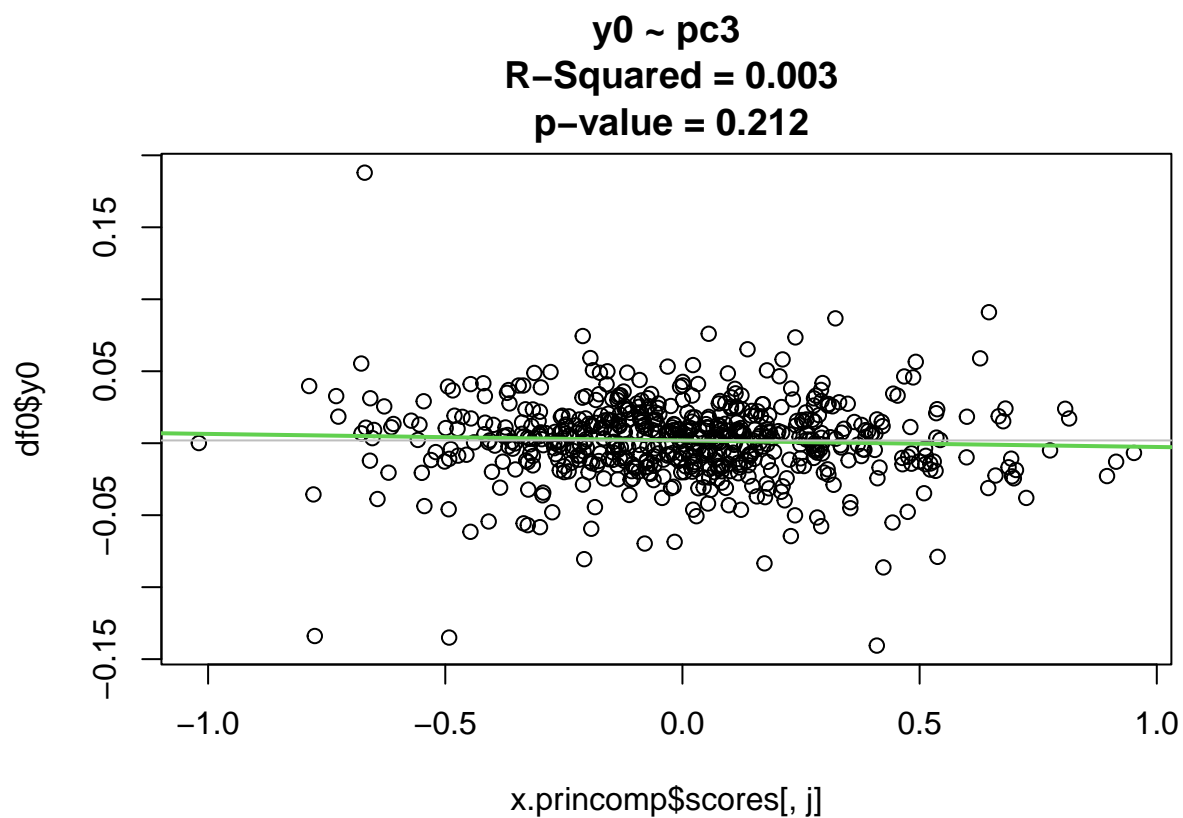
```
## Residual Standard Error=0.0141
## R-Square=0.7335
## F-statistic (df=1, 610)=1678.537
## p-value=0
##
##           Estimate Std.Err t-value Pr(>|t|)
## Intercept    0.0019   6e-04   3.3897    7e-04
## X             0.0122   3e-04  40.9700    0e+00
```



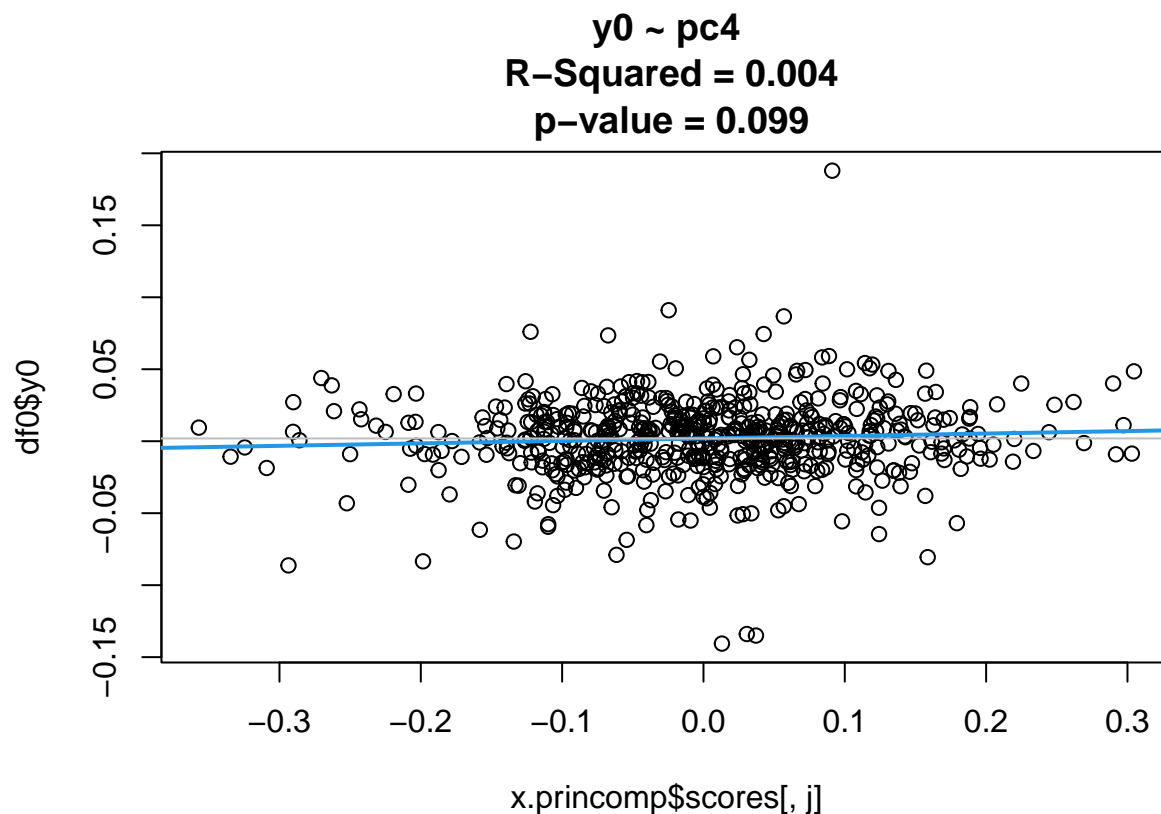
```
## Residual Standard Error=0.0263
## R-Square=0.0648
## F-statistic (df=1, 610)=42.2976
## p-value=0
##
##      Estimate Std.Err t-value Pr(>|t|)
## Intercept  0.0019  0.0011  1.8097  0.0708
## X          -0.0136  0.0021 -6.5037  0.0000
```



```
## Residual Standard Error=0.0272
## R-Square=0.0026
## F-statistic (df=1, 610)=1.5609
## p-value=0.212
##
##           Estimate Std.Err t-value Pr(>|t|)
## Intercept   0.0019  0.0011  1.7523  0.0802
## X           -0.0045  0.0036 -1.2494  0.2120
```



```
## Residual Standard Error=0.0272
## R-Square=0.0045
## F-statistic (df=1, 610)=2.7295
## p-value=0.099
##
##           Estimate Std.Err t-value Pr(>|t|)
## Intercept  0.0019  0.0011  1.7540  0.0799
## X          0.0171  0.0103  1.6521  0.0990
```



7.2 Regression model on all PC variables

```
pcregfit<-lm(df0$y0 ~ x.princomp$scores)
summary(pcregfit)
```

```
##
## Call:
## lm(formula = df0$y0 ~ x.princomp$scores)
##
## Residuals:
```

	Min	1Q	Median	3Q	Max
	-0.035159	-0.007275	-0.000405	0.007083	0.065036

```
##
## Coefficients:
```

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	0.0019275	0.0004872	3.956	8.51e-05 ***
x.princomp\$scoresComp.1	0.0122271	0.0002557	47.819	< 2e-16 ***
x.princomp\$scoresComp.2	-0.0135703	0.0009544	-14.218	< 2e-16 ***
x.princomp\$scoresComp.3	-0.0045493	0.0016127	-2.821	0.004946 **
x.princomp\$scoresComp.4	0.0170636	0.0045788	3.727	0.000212 ***

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01205 on 607 degrees of freedom
## Multiple R-squared:  0.8053, Adjusted R-squared:  0.804
## F-statistic: 627.7 on 4 and 607 DF, p-value: < 2.2e-16
```

```
pcregfit.summary<-summary(pcregfit)
```

7.3 Use PCA regression to recompute regression parameters

```
#      on original variables
pcbetavec<-as.numeric(pcregfit$coefficients[-1])
x.princomp$loadings

##
## Loadings:
##      Comp.1 Comp.2 Comp.3 Comp.4
## SPY  0.521      0.145  0.837
## MDY  0.495 -0.497 -0.699 -0.140
## QQQ  0.479  0.792 -0.155 -0.346
## DIA  0.504 -0.346  0.683 -0.400
##
##              Comp.1 Comp.2 Comp.3 Comp.4
## SS loadings      1.00  1.00  1.00  1.00
## Proportion Var   0.25  0.25  0.25  0.25
## Cumulative Var   0.25  0.50  0.75  1.00

betaFromPCA<-x.princomp$loadings %*% as.matrix(pcbetavec)
fit$coefficients

##      (Intercept)          SPY          MDY          QQQ          DIA
## 0.0019275050 0.0189310332 0.0135803017 -0.0100924647 0.0009219118
```

7.4 Compute regression parameter based on only first 3 pc vars

```
pcbetavec

## [1] 0.012227077 -0.013570331 -0.004549317 0.017063552

pcbetavec123<-0*pcbetavec
pcbetavec123[1:3]<-pcbetavec[1:3]

betaFromPC123<-x.princomp$loadings %*% as.matrix(pcbetavec123)
```

7.5 Compute regression parameter based on significant pc vars

The code below computes the regression parameter (original scale) using only the statistically significant pc variables, as determined by those that have $abs\ t\ stat > \sqrt{2}$.

```
ind.tokeep<-as.numeric(
  abs(pcregfit.summary$coefficients[-1,3])>sqrt(2))

pcbetavectokeep<-pcbetavec*ind.tokeep

betaFromPCtokeep<-x.princomp$loadings %*% as.matrix(pcbetavectokeep)
```

7.6 Create table of betas from these three fits

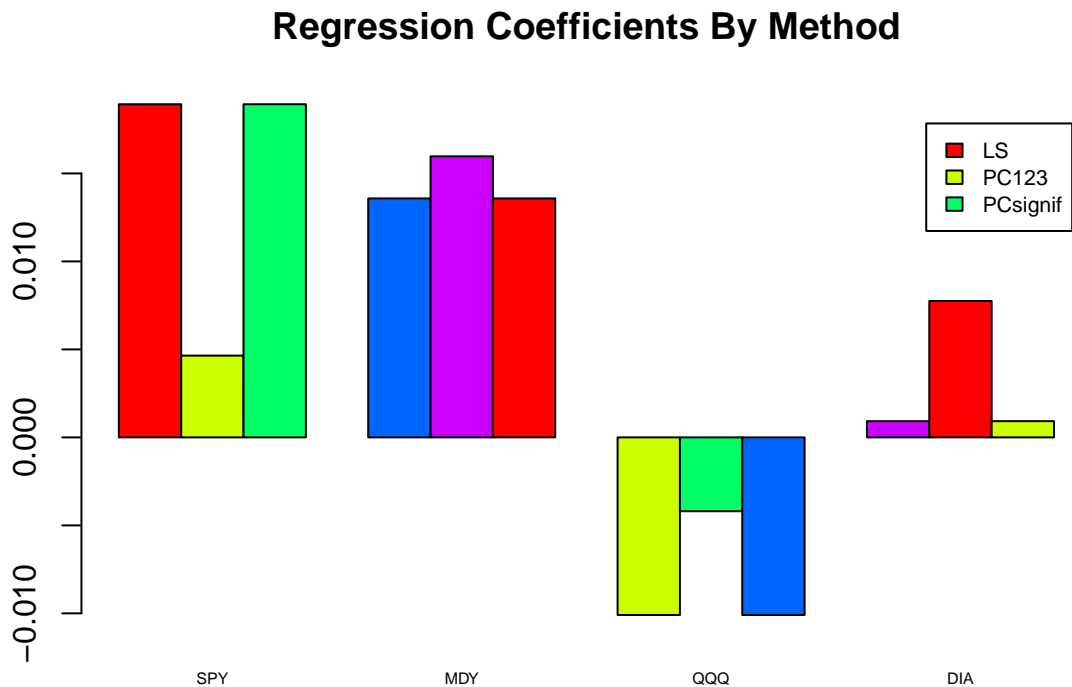
```
tab.betas<-cbind(betaFromPCA, betaFromPC123, betaFromPCtokeep)
dimnames(tab.betas)[[2]]<-c("LS", "PC123", "PCsignif")
```

```
print(tab.betas)
```

```
##           LS           PC123           PCsignif
## SPY  0.0189310332  0.004646279  0.0189310332
## MDY  0.0135803017  0.015971749  0.0135803017
## QQQ -0.0100924647 -0.004195881 -0.0100924647
## DIA  0.0009219118  0.007749934  0.0009219118
```

```
# 5. Compare coefficients from fits including LASSO and Ridge
```

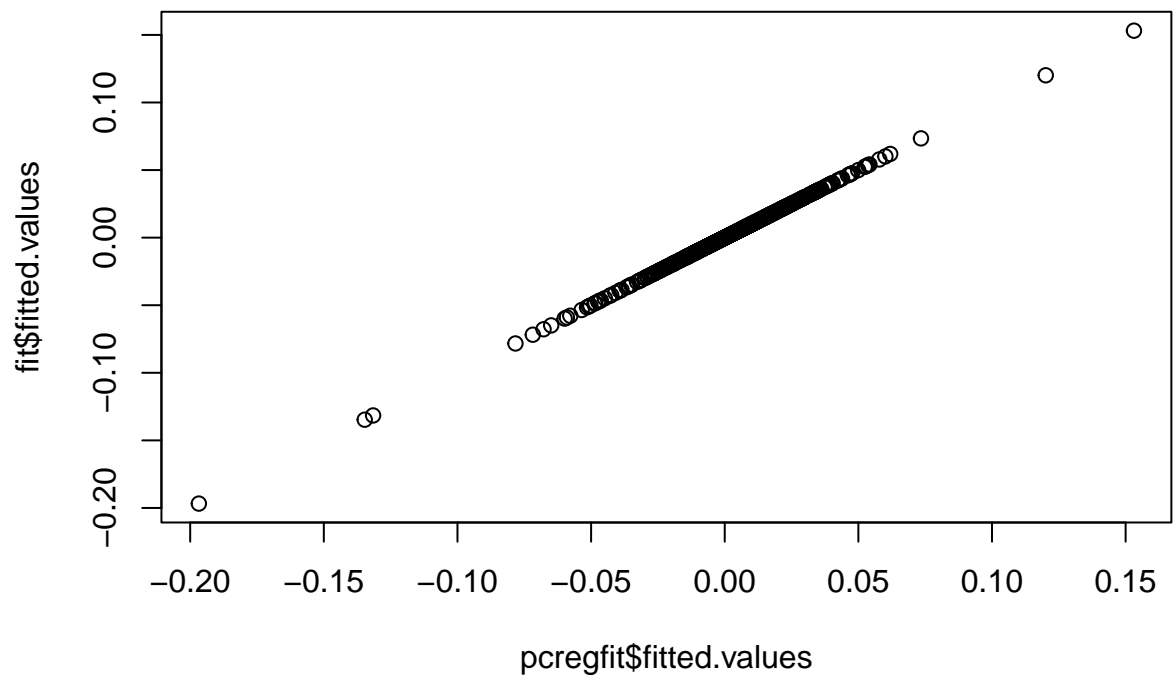
```
barplot(t(as.matrix(tab.betas)), beside=TRUE,
        col=rainbow(5), cex.names=.5,
        legend=TRUE, args.legend=list(cex=.7))
title(main="Regression Coefficients By Method")
```



7.7 Demonstrate equality of LS and PCA Regression Fitted Values

```
# pcregfit equals LS fit
```

```
par(mfcol=c(1,1))
plot(pcregfit$fitted.values, fit$fitted.values)
```

8. Ridge and Lasso Regression Fits

8.1 Load glmnet package for ridge/lasso regressions

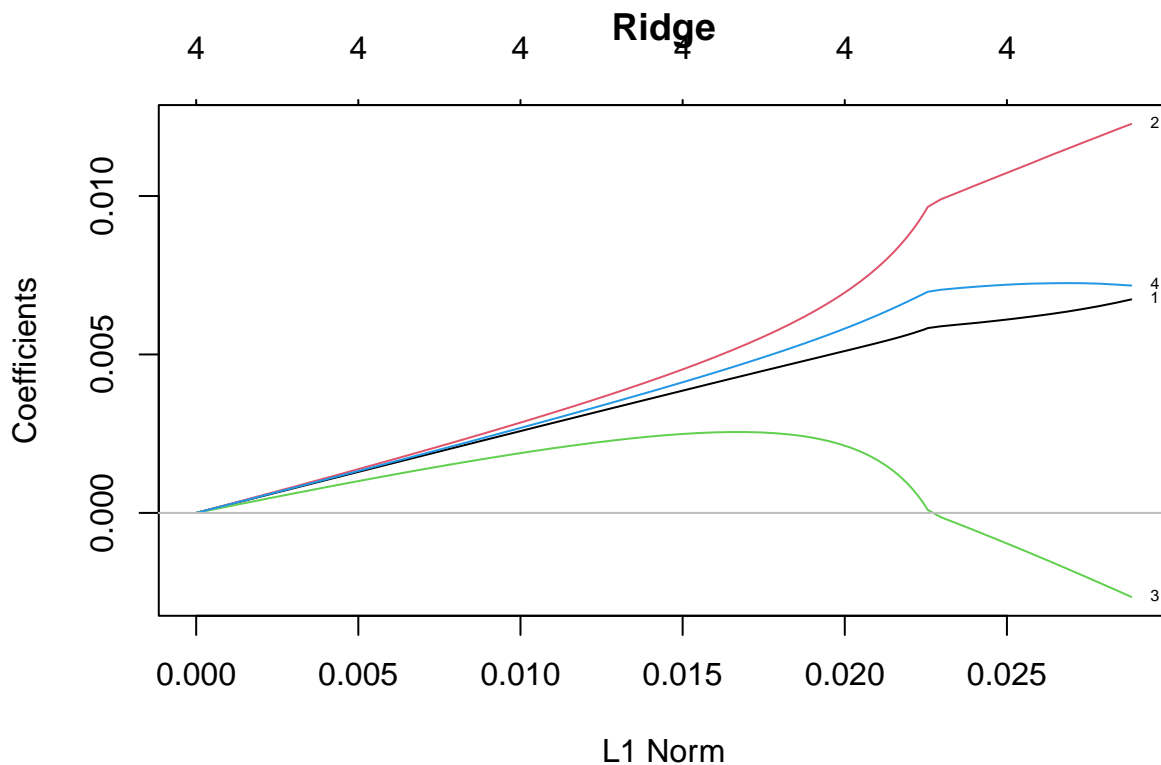
```
library(glmnet)
```

8.2 Define y vector and x matrix for ridge/lasso fits

```
y=fit$y  
x=fit$x[, -1]
```

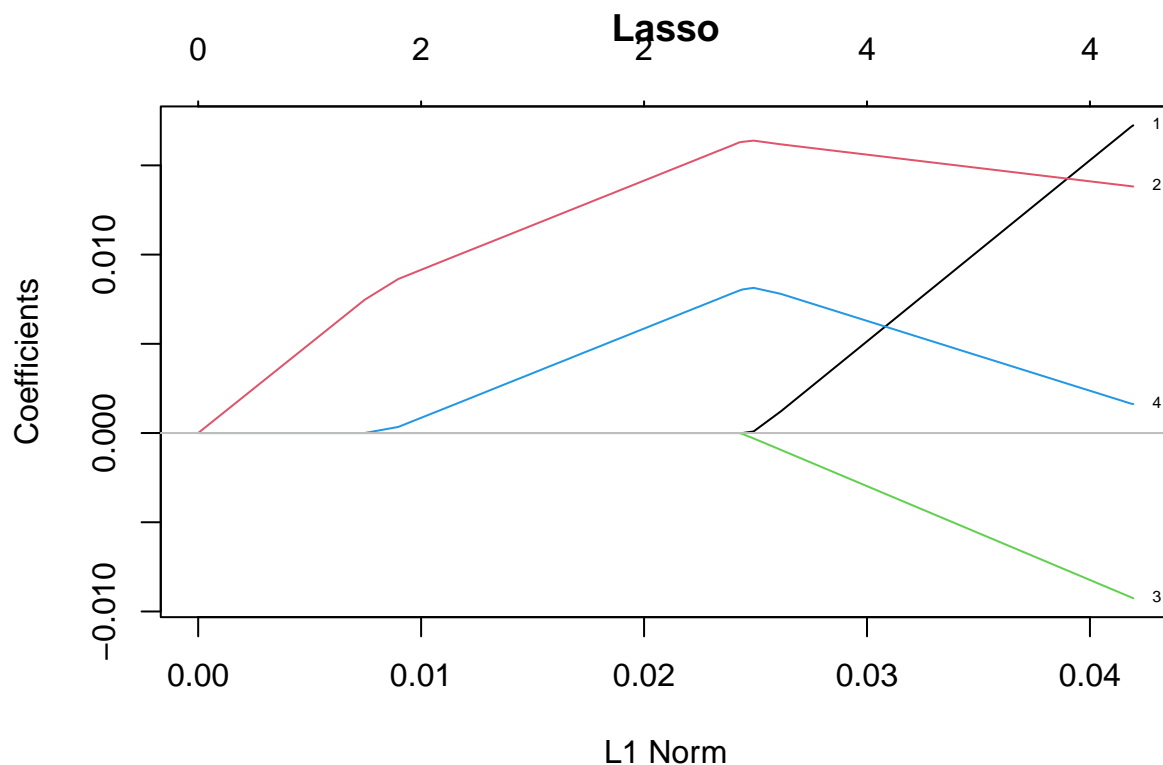
8.3 Plot coefficient trace of ridge regression model

```
y.glmnet.ridge<-glmnet(x,y, alpha=0)  
# alpha=0 for Ridge  
plot(y.glmnet.ridge, label=TRUE, main="Ridge")  
abline(h=0,col='gray')
```



8.4 Plot coefficient trace of lasso regression model —

```
y.glmnet.lasso<-glmnet(x,y, alpha=1)  
# alpha=1 for LASSO  
plot(y.glmnet.lasso, label=TRUE,main="Lasso")  
abline(h=0,col='gray')
```



8.5 Apply Cross Validation to choose ridge parameters

```
lambdas=10^seq(3.,-5, by=-.1)
summary(y.glmnet.ridge)
```

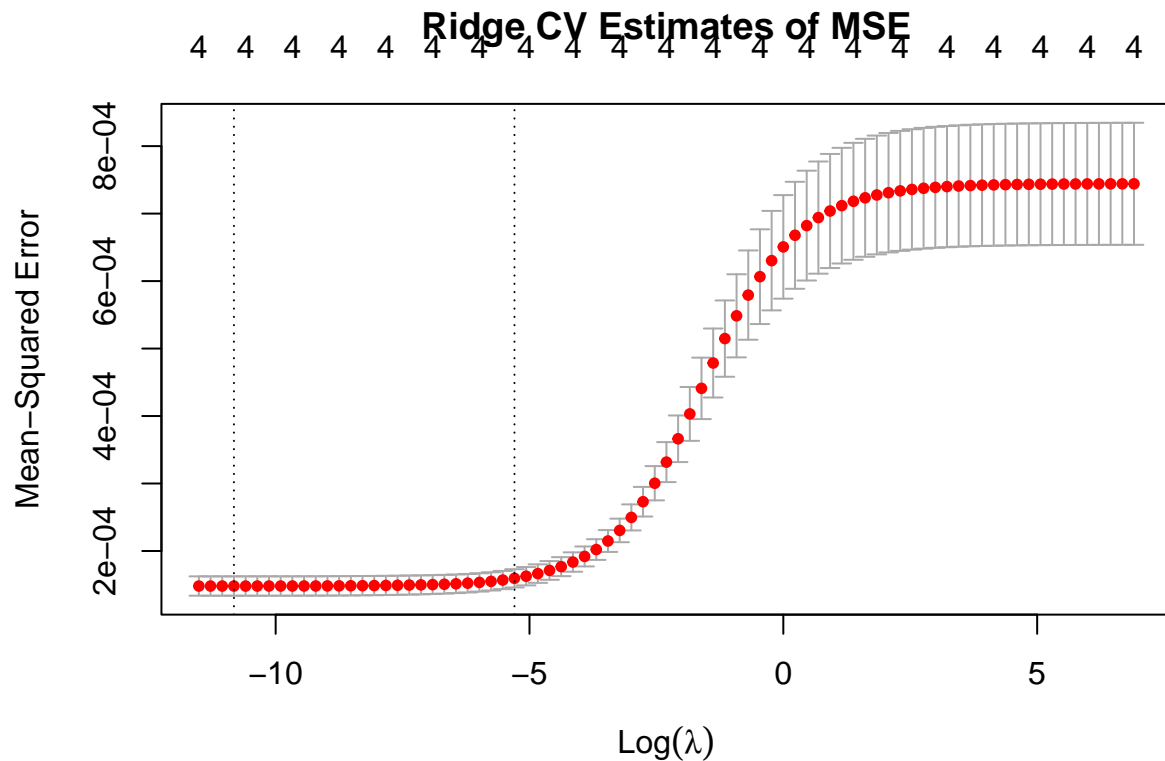
```
##          Length Class      Mode
## a0         100   -none-    numeric
## beta        400 dgCMatrix S4
## df          100   -none-    numeric
## dim           2   -none-    numeric
## lambda       100   -none-    numeric
## dev.ratio   100   -none-    numeric
## nulldev       1   -none-    numeric
## npasses       1   -none-    numeric
## jerr          1   -none-    numeric
## offset        1   -none-   logical
## call          4   -none-     call
## nobs          1   -none-    numeric
```

```
#
# Cross-validation estimates of prediction error
# ridge case (alpha=0)
y.glmnet.ridge<-glmnet(x,y,alpha=0, lambda=lambdas)
summary(y.glmnet.ridge)
```

```
##          Length Class      Mode
## a0          81   -none-    numeric
## beta        324 dgCMatrix S4
## df          81   -none-    numeric
## dim           2   -none-    numeric
```

```
## lambda      81    -none-    numeric
## dev.ratio   81    -none-    numeric
## nulldev     1    -none-    numeric
## npasses     1    -none-    numeric
## jerr        1    -none-    numeric
## offset      1    -none-    logical
## call        5    -none-    call
## nobs        1    -none-    numeric
```

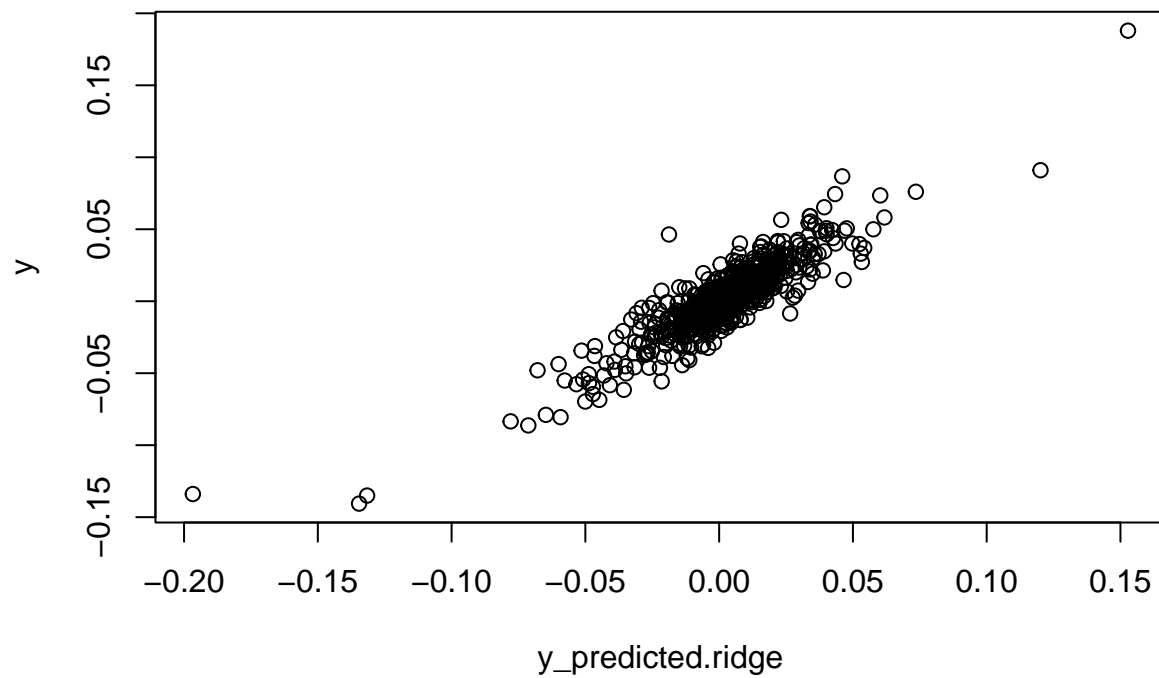
```
y.cv.glmnet.ridge<-cv.glmnet(x,y,alpha=0, lambda=lambdas)
plot(y.cv.glmnet.ridge, main="Ridge CV Estimates of MSE")
```



```
optlambda.ridge<-y.cv.glmnet.ridge$lambda.min
glmnet.ridgefit<-y.cv.glmnet.ridge$glmnet.fit
y_predicted.ridge<- predict(glmnet.ridgefit,s=optlambda.ridge,newx=x)
```

Plot Observed vs Fitted for Ridge Regression

```
plot(y_predicted.ridge, y)
```

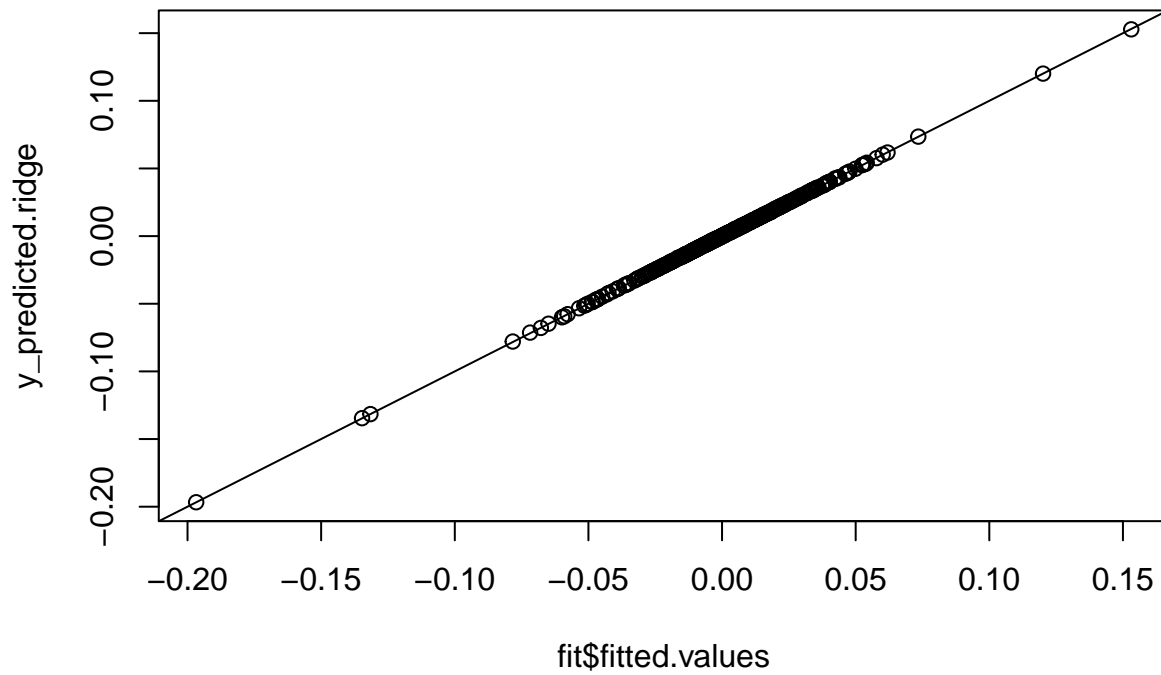


```
cor(y_predicted.ridge,y)^2
```

```
##           [,1]
## s1 0.8052741
```

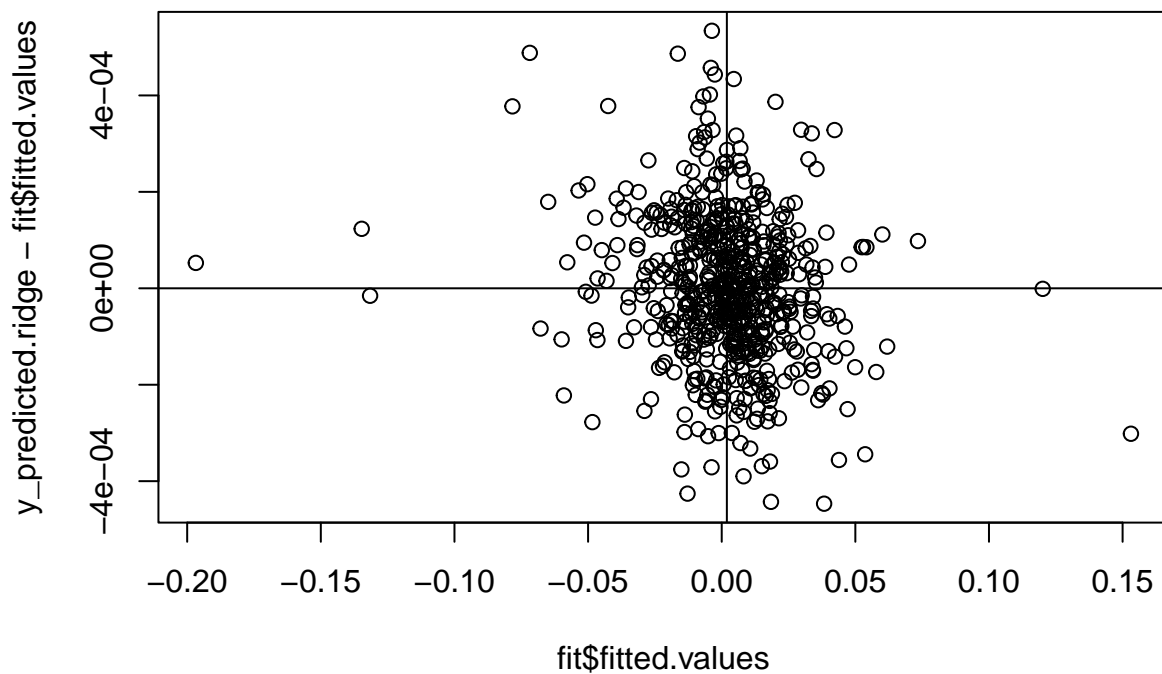
```
# Compare ridge fitted to ls fitted
plot(fit$fitted.values, y_predicted.ridge,
     main="Ridge Regression\n(Shrinks to Mean)")
abline(a=0,b=1)
```

Ridge Regression (Shrinks to Mean)



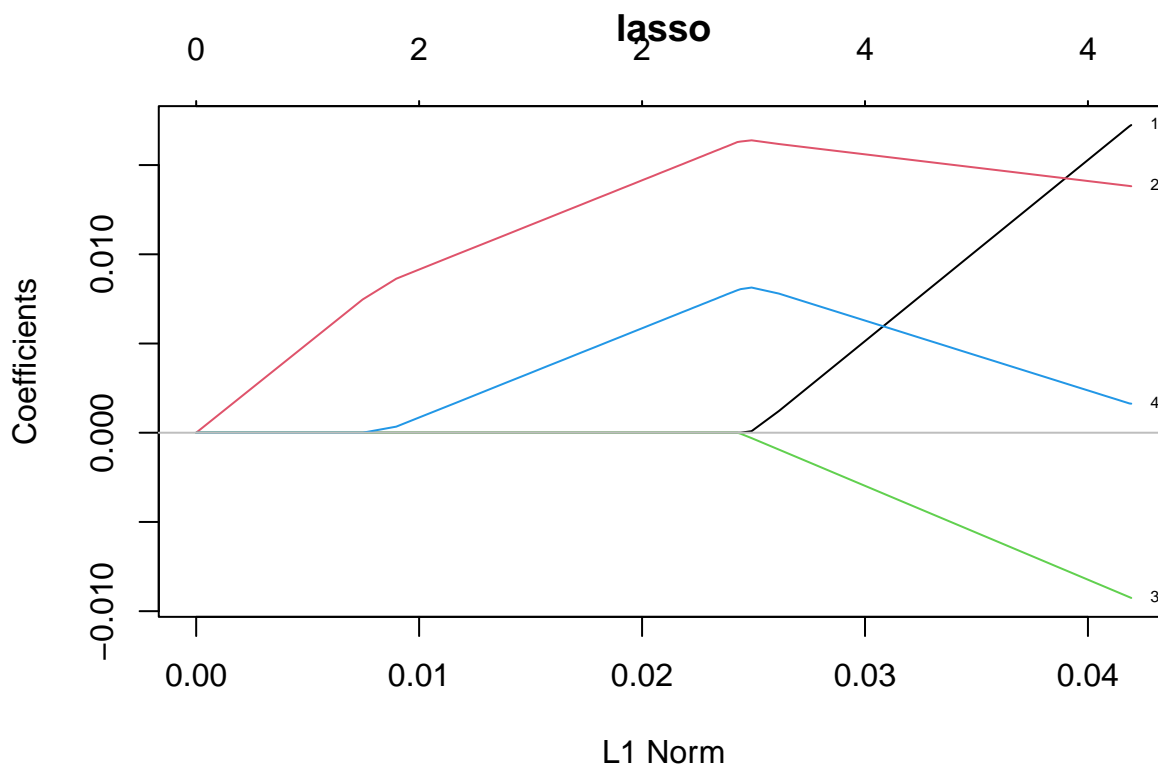
```
plot(fit$fitted.values, y_predicted.ridge ~ fit$fitted.values,
     main="Ridge Regression \n(Shrinks to Mean)")
abline(h=0)
abline(v=mean(fit$fitted.values))
```

Ridge Regression (Shrinks to Mean)



8.6 Apply Cross Validation to choose lasso parameters

```
y.glmnet.lasso<-glmnet(x,y, alpha=1)
# alpha=1 for lasso
plot(y.glmnet.lasso, label=TRUE, main="lasso")
abline(h=0,col='gray')
```



```
lambdas=10^seq(3.,-5, by=-.1)
summary(y.glmnet.lasso)
```

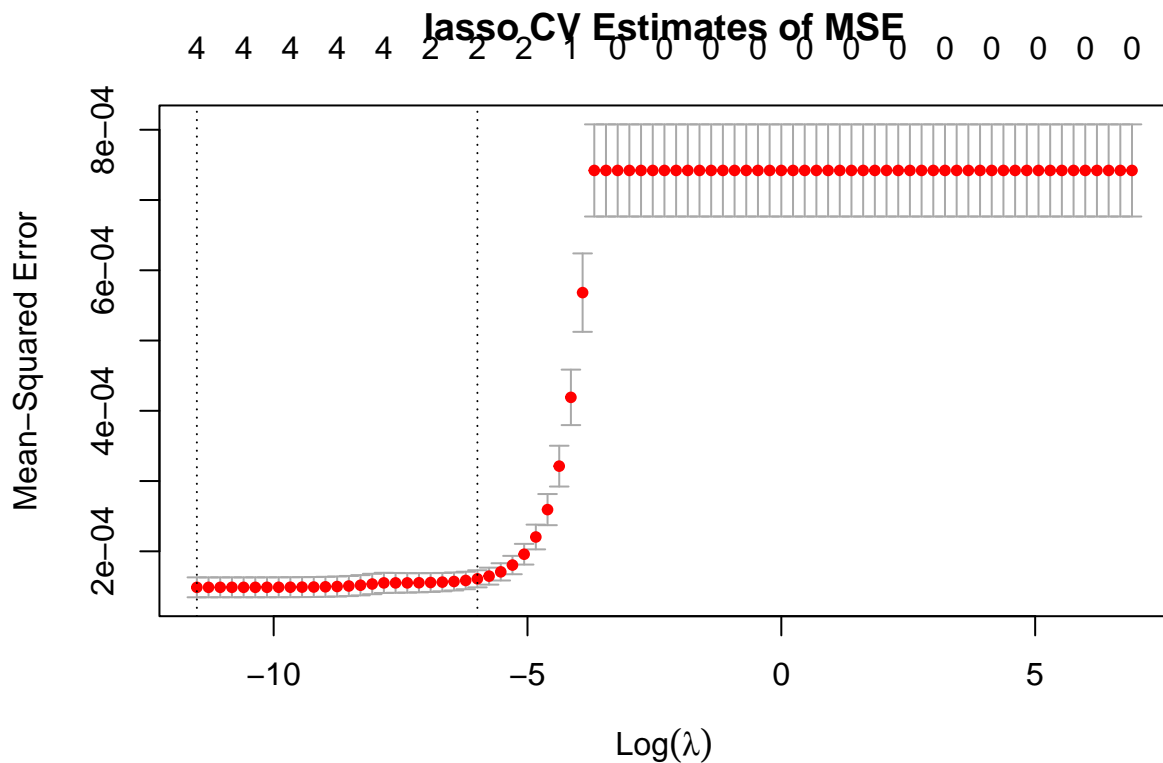
```
##          Length Class      Mode
## a0         74    -none-   numeric
## beta       296  dgCMatrix S4
## df         74    -none-   numeric
## dim         2    -none-   numeric
## lambda      74    -none-   numeric
## dev.ratio   74    -none-   numeric
## nulldev     1    -none-   numeric
## npasses     1    -none-   numeric
## jerr        1    -none-   numeric
## offset      1    -none-   logical
## call        4    -none-   call
## nobs        1    -none-   numeric
```

```
#
# Cross-validation estimates of prediction error
# lasso case (alpha=1)
y.glmnet.lasso<-glmnet(x,y,alpha=1, lambda=lambdas)
```

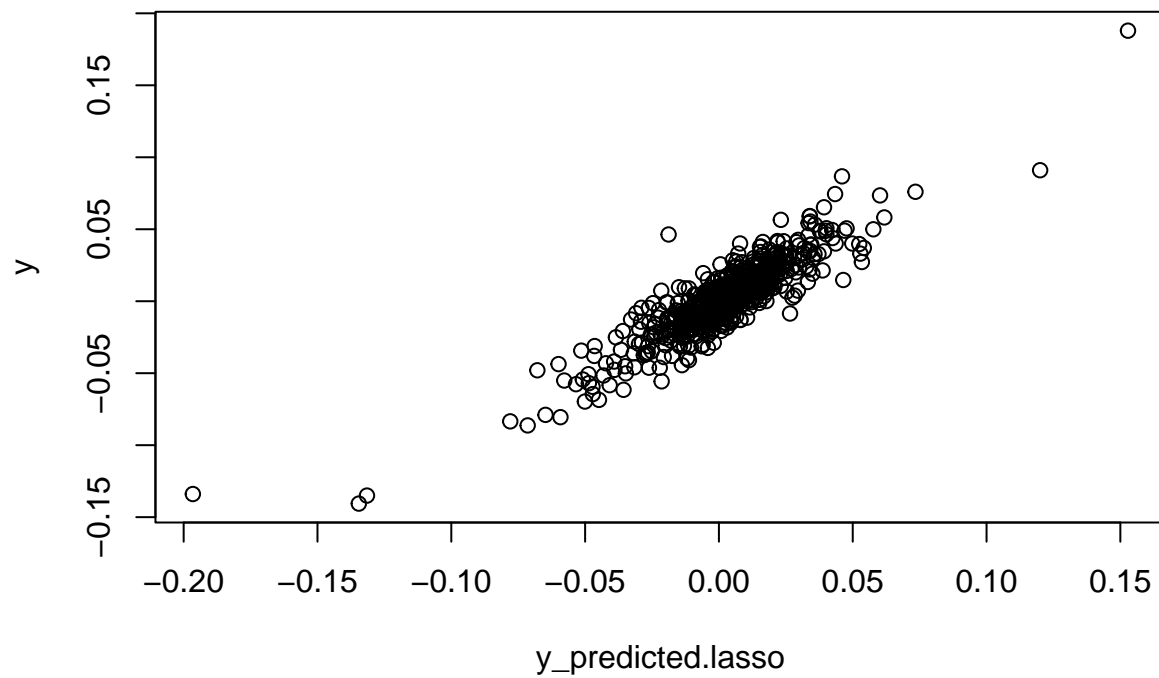
```
summary(y.glmnet.lasso)
```

```
##          Length Class      Mode
## a0         81    -none-   numeric
## beta       324   dgCMatrix S4
## df          81    -none-   numeric
## dim         2    -none-   numeric
## lambda      81    -none-   numeric
## dev.ratio   81    -none-   numeric
## nulldev     1    -none-   numeric
## npasses     1    -none-   numeric
## jerr        1    -none-   numeric
## offset      1    -none-   logical
## call        5    -none-   call
## nobs        1    -none-   numeric
```

```
y.cv.glmnet.lasso<-cv.glmnet(x,y,alpha=1, lambda=lambdas)
plot(y.cv.glmnet.lasso, main="lasso CV Estimates of MSE")
```



```
optlambda.lasso<-y.cv.glmnet.lasso$lambda.min
glmnet.lassofit<-y.cv.glmnet.lasso$glmnet.fit
y_predicted.lasso<- predict(glmnet.lassofit,s=optlambda.lasso,newx=x)
#
plot(y_predicted.lasso, y)
```

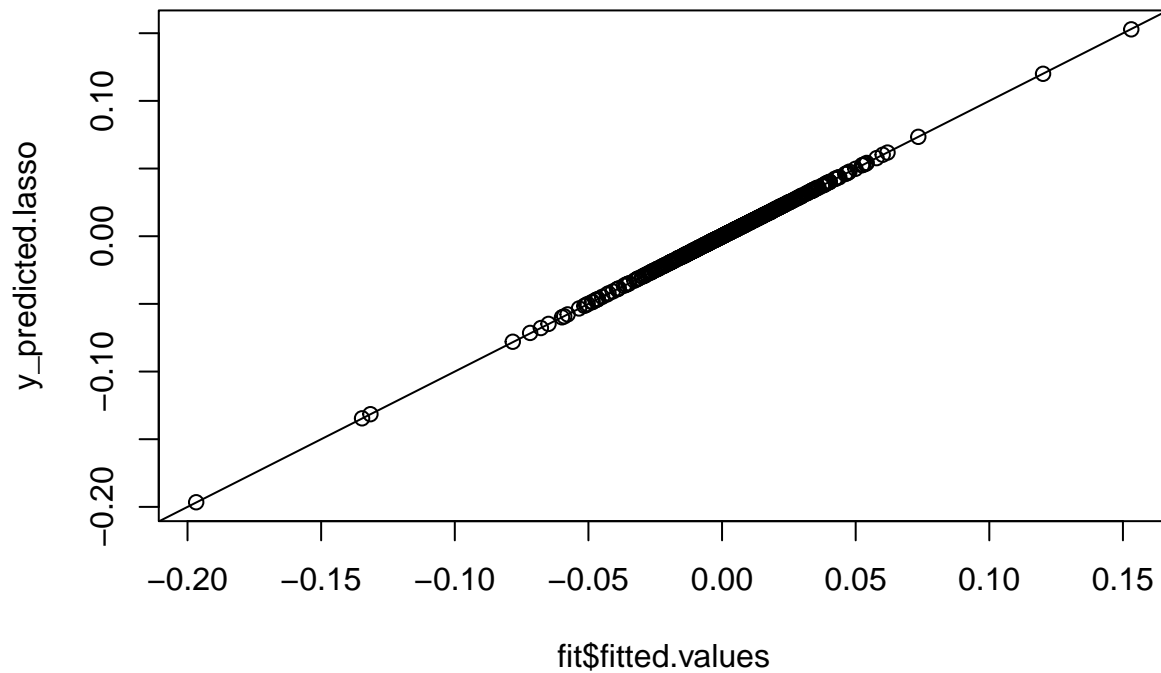



```
cor(y_predicted.lasso,y)^2
```

```
##           [,1]
## s1 0.8052873
```

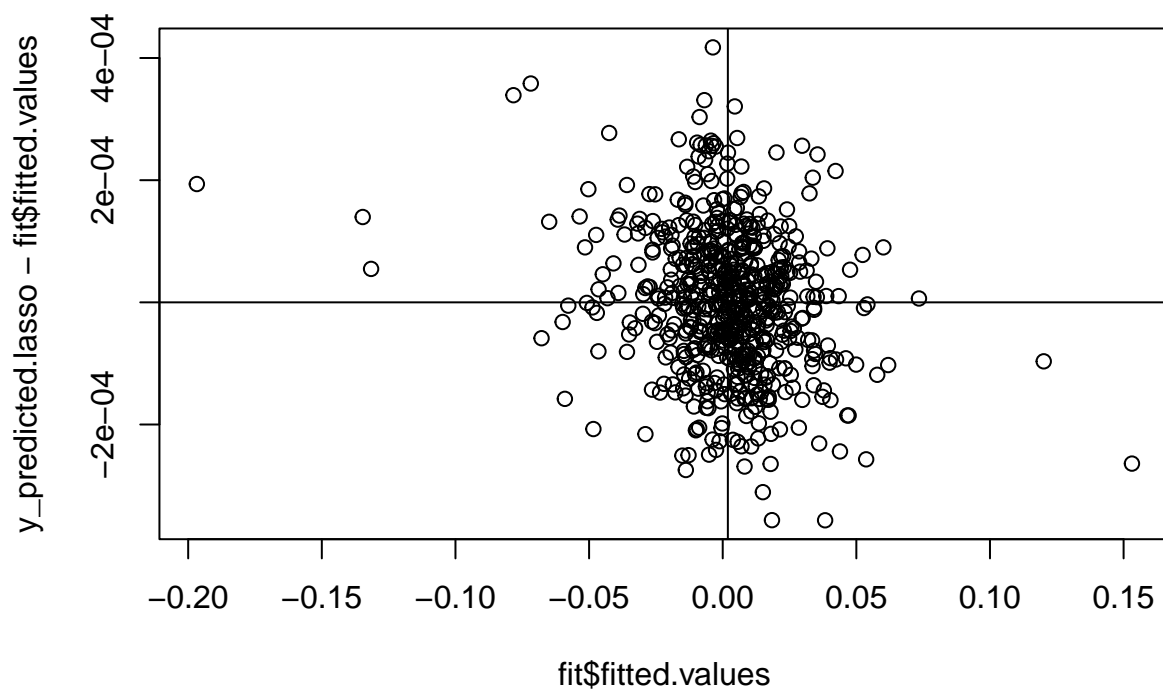
```
# Compare lasso fitted to ls fitted
plot(fit$fitted.values, y_predicted.lasso,
     main="lasso Regression \n(Shrinks to Mean)")
abline(a=0,b=1)
```

lasso Regression (Shrinks to Mean)



```
plot(fit$fitted.values, y_predicted.lasso ~ fit$fitted.values,  
     main="lasso Regression \n(Shrinks to Mean)")  
abline(h=0, v=mean(fit$fitted.values))
```

lasso Regression (Shrinks to Mean)



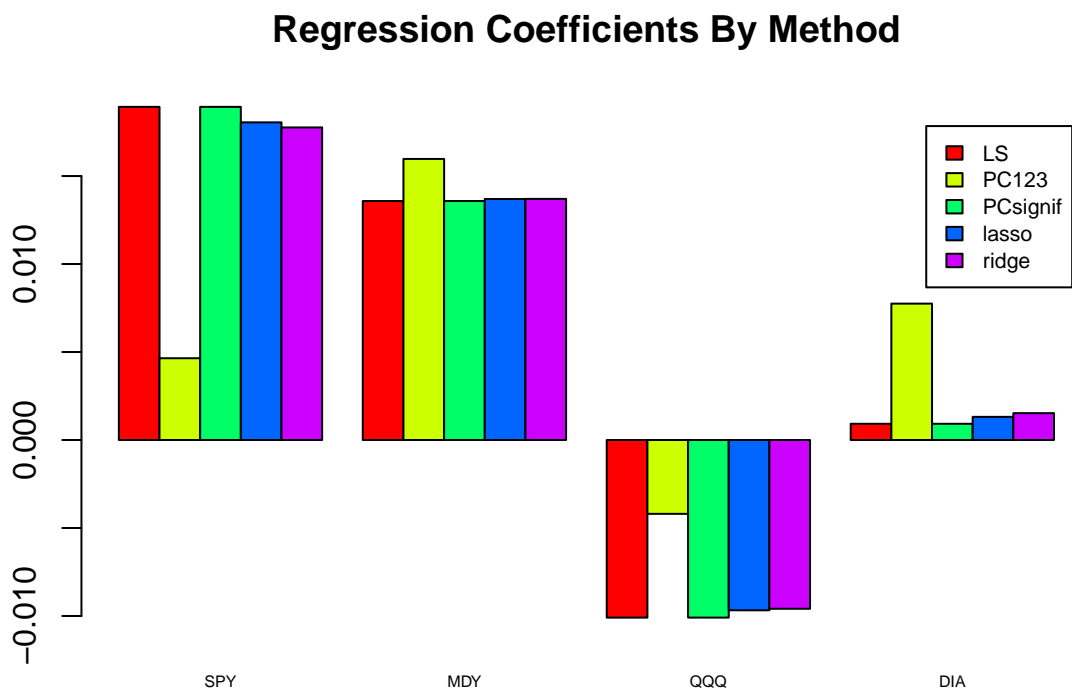
```
## 8.7 Compare coefficients from fits including LASSO and Ridge ----
```

```
tab.betas<-cbind(betaFromPCA, betaFromPC123, betaFromPCtokeep)
dimnames(tab.betas)[[2]]<-c("LS", "PC123", "PCsignif")
print(tab.betas)
```

```
##           LS           PC123           PCsignif
## SPY  0.0189310332  0.004646279  0.0189310332
## MDY  0.0135803017  0.015971749  0.0135803017
## QQQ -0.0100924647 -0.004195881 -0.0100924647
## DIA  0.0009219118  0.007749934  0.0009219118
```

```
tab.betas2<-data.frame(cbind(tab.betas,
                             lasso=coef(y.cv.glmnet.lasso,s="lambda.min")[-1],
                             ridge=coef(y.cv.glmnet.ridge,s="lambda.min")[-1]))
```

```
barplot(t(as.matrix(tab.betas2)), beside=TRUE,
        col=rainbow(5), cex.names=.5,
        legend=TRUE, args.legend=list(cex=.7))
title(main="Regression Coefficients By Method")
```



MIT OpenCourseWare
<https://ocw.mit.edu>

18.642 Topics in Mathematics with Applications in Finance
Fall 2024

For information about citing these materials or our Terms of Use, visit: <https://ocw.mit.edu/terms>.