ETF Casestudy

Peter Kempthorne

2024-09-23

Contents

0. Load Libraries

```r
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
```

```r
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
```

```r
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
```

```r
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
```

```r
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="spyders_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)
```

```r
  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"
```

```r
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

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

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

1.2 Weekly Stock Price Data

1.2.1 Create matrix of weekly closing prices

```r
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
```

```r
length(time(pmat0.0.1.weekly))
```

```
## [1] 1290
```

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

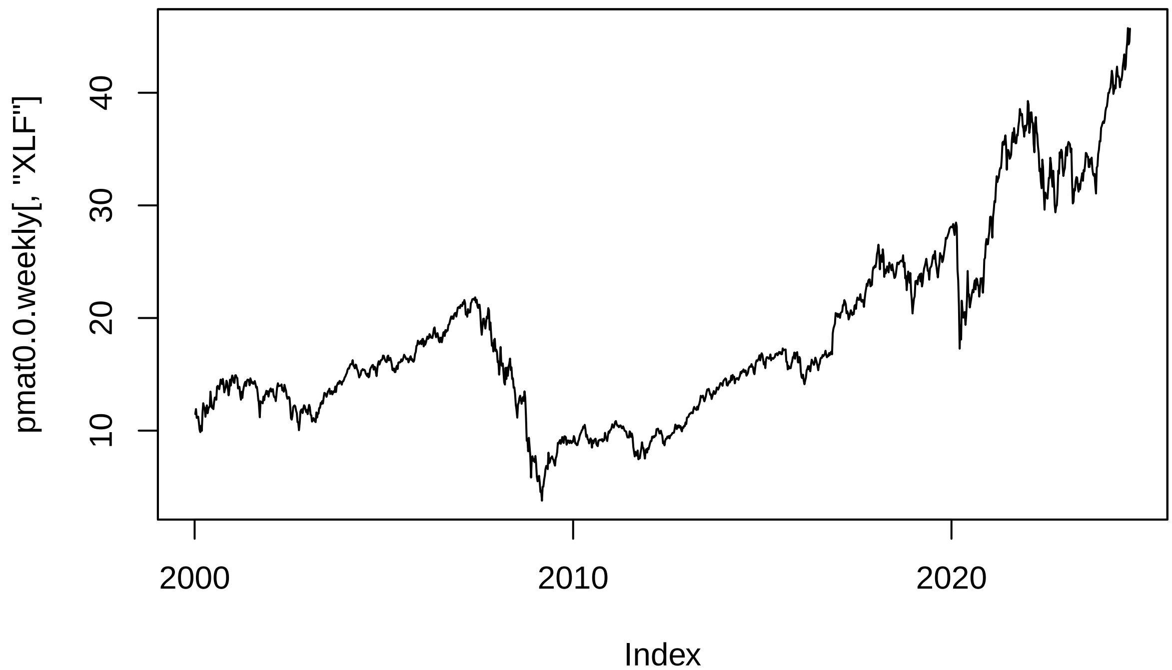
```
## [1] 10963 10970 10977 10984 10991
```

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

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

1.2.2 Create matrix of weekly log returns

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




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

```
##                 SPY      MDY      QQQ      XLB      XLV      XLP      XLY
## 2000-01-07 93.20474 60.77375 76.63248 15.80843 20.71107 13.61024 22.95816
## 2000-01-14 93.98414 61.94609 79.50619 15.29558 22.30757 13.48810 22.64126
## 2000-01-21 92.36544 62.27434 81.95415 14.31486 22.19897 13.02570 21.70226
## 2000-01-28 86.88984 59.03870 73.22663 13.51409 21.02603 12.73780 20.28206
## 2000-02-04 91.18630 61.12546 82.72588 13.48710 21.99261 12.68545 21.08020
## 2000-02-11 88.68842 61.19578 85.04080 12.50639 21.30840 12.04856 20.27033
##                 XLE      XLF      XLI      XLK      XLU      DIA
## 2000-01-07 14.88110 11.47026 18.86840 38.36602 11.60620 67.18431
## 2000-01-14 14.87281 11.91383 18.62753 39.77654 11.38846 68.34761
## 2000-01-21 15.41936 11.12786 18.19596 40.34076 11.77115 65.66428
## 2000-01-28 14.14408 11.17456 17.34287 36.81446 11.15752 62.51753
## 2000-02-04 13.94533 11.24459 17.41313 40.71688 11.44784 63.88174
## 2000-02-11 13.63065 10.70765 16.35930 40.90496 11.07835 60.82587
```

```r
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.

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

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

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

```
##           SPY       MDY       QQQ       XLB       XLV       XLP       XLY
## SPY 1.0000000 0.9263244 0.8406530 0.8169412 0.8048856 0.7133526 0.8851434
## MDY 0.9263244 1.0000000 0.7857591 0.8350609 0.7038070 0.6111273 0.8456266
## QQQ 0.8406530 0.7857591 1.0000000 0.5875579 0.6467435 0.4414167 0.7333358
## XLB 0.8169412 0.8350609 0.5875579 1.0000000 0.6174494 0.5939057 0.7566491
## XLV 0.8048856 0.7038070 0.6467435 0.6174494 1.0000000 0.6477534 0.6718096
## XLP 0.7133526 0.6111273 0.4414167 0.5939057 0.6477534 1.0000000 0.6295337
## XLY 0.8851434 0.8456266 0.7333358 0.7566491 0.6718096 0.6295337 1.0000000
## XLE 0.6713572 0.7040841 0.4252876 0.7034908 0.4825468 0.4657262 0.5259733
## XLF 0.8404444 0.8232555 0.5853957 0.7296194 0.6391777 0.5685931 0.7733551
## XLI 0.9091362 0.9002693 0.7081750 0.8512075 0.7054340 0.6662999 0.8287766
## XLK 0.8542522 0.7790903 0.9640273 0.6072689 0.6348147 0.4588322 0.7419041
## XLU 0.6166184 0.5736058 0.3873537 0.5073337 0.5512258 0.6180840 0.4970907
## DIA 0.9556731 0.8819201 0.7305154 0.8294260 0.7928100 0.7558884 0.8572597
##           XLE       XLF       XLI       XLK       XLU       DIA
## SPY 0.6713572 0.8404444 0.9091362 0.8542522 0.6166184 0.9556731
## MDY 0.7040841 0.8232555 0.9002693 0.7790903 0.5736058 0.8819201
## QQQ 0.4252876 0.5853957 0.7081750 0.9640273 0.3873537 0.7305154
## XLB 0.7034908 0.7296194 0.8512075 0.6072689 0.5073337 0.8294260
## XLV 0.4825468 0.6391777 0.7054340 0.6348147 0.5512258 0.7928100
## XLP 0.4657262 0.5685931 0.6662999 0.4588322 0.6180840 0.7558884
## XLY 0.5259733 0.7733551 0.8287766 0.7419041 0.4970907 0.8572597
## XLE 1.0000000 0.5858364 0.6812261 0.4344610 0.4907898 0.6729492
## XLF 0.5858364 1.0000000 0.8241749 0.5965313 0.4832339 0.8299180
## XLI 0.6812261 0.8241749 1.0000000 0.7224326 0.5653708 0.9232702
## XLK 0.4344610 0.5965313 0.7224326 1.0000000 0.4106982 0.7606385
## XLU 0.4907898 0.4832339 0.5653708 0.4106982 1.0000000 0.6176370
## DIA 0.6729492 0.8299180 0.9232702 0.7606385 0.6176370 1.0000000
```

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

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

1.2.3 Create separate objects for sector etfs and index etfs

```r
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  0.008327424  0.019106545  0.03681378  0.01716685
## 2000-01-21 -0.017373142  0.005284925  0.03032506 -0.04005145
## 2000-01-28 -0.061111787 -0.053356329 -0.11260074 -0.04910812
## 2000-02-04  0.048263591  0.034735339  0.12197327  0.02158654
## 2000-02-11 -0.027775413  0.001149762  0.02759872 -0.04901837
```

```
## 2000-02-18 -0.024636530 -0.013889528 -0.01449766 -0.02053414
```

```
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          XLY           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
##                     XLF         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 (XLP)

```
y0.name="XLP (Consumer Staples ETF)"
y0.symbol="XLP"
y0<-rmat0.0.weekly.sectoretfs.period0$XLP
```

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.    -0.116194529 -0.157189170 -0.202701342 -0.119356994 -0.187241821
## 1st Qu. -0.007267913 -0.007898118 -0.010483530 -0.010700447 -0.007719788
## Median   0.003452617  0.004176716  0.003299196  0.004865223  0.003425232
## Mean     0.001926583  0.002634949  0.002100222  0.003474893  0.002332767
## 3rd Qu.  0.012286899  0.014691387  0.015611677  0.019312326  0.013328769
## Max.     0.065588245  0.114145893  0.172891253  0.091102251  0.119922619
```

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

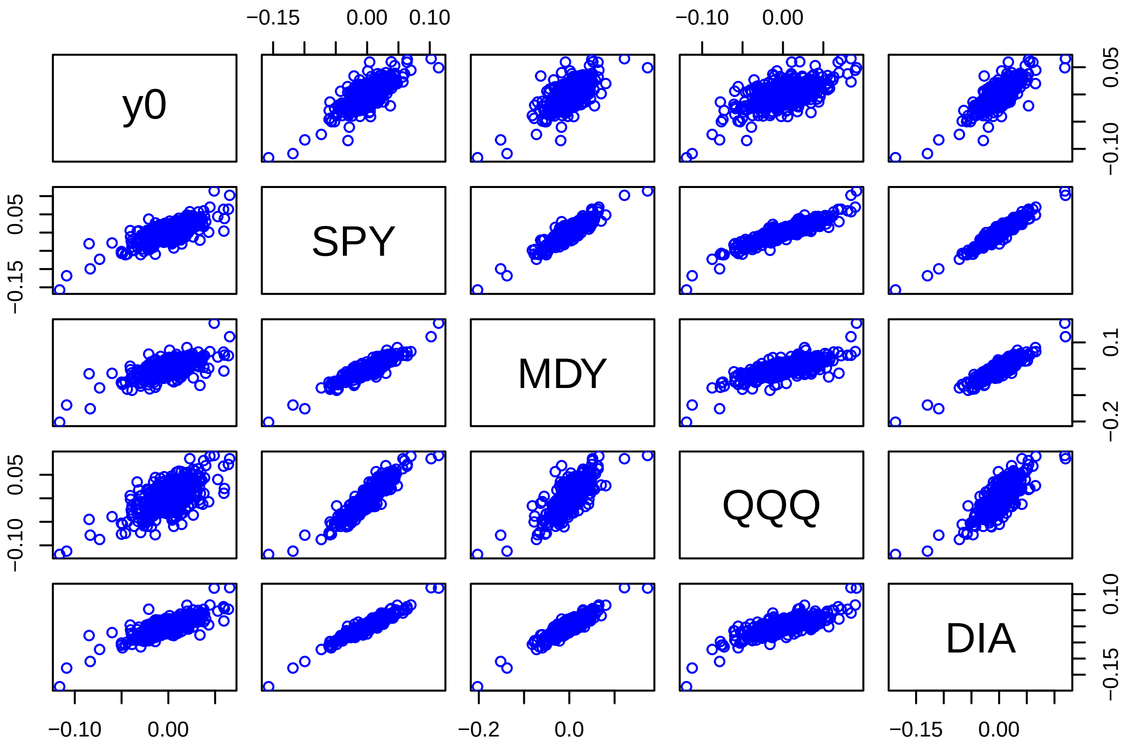
```
##          mean       vol
## y0  0.1001823 0.1370785
## SPY 0.1370173 0.1607707
## MDY 0.1092116 0.1957695
## QQQ 0.1806944 0.1893826
## DIA 0.1213039 0.1628826
```

2.4.2 Pairwise correlations and pairs plots

```
cor(df0)
```

```
##            y0       SPY       MDY       QQQ       DIA
## y0  1.0000000 0.7526003 0.6552977 0.6243527 0.7768283
## SPY 0.7526003 1.0000000 0.9175267 0.9181598 0.9540900
## MDY 0.6552977 0.9175267 1.0000000 0.7689490 0.9092160
## QQQ 0.6243527 0.9181598 0.7689490 1.0000000 0.7977508
## DIA 0.7768283 0.9540900 0.9092160 0.7977508 1.0000000
```

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

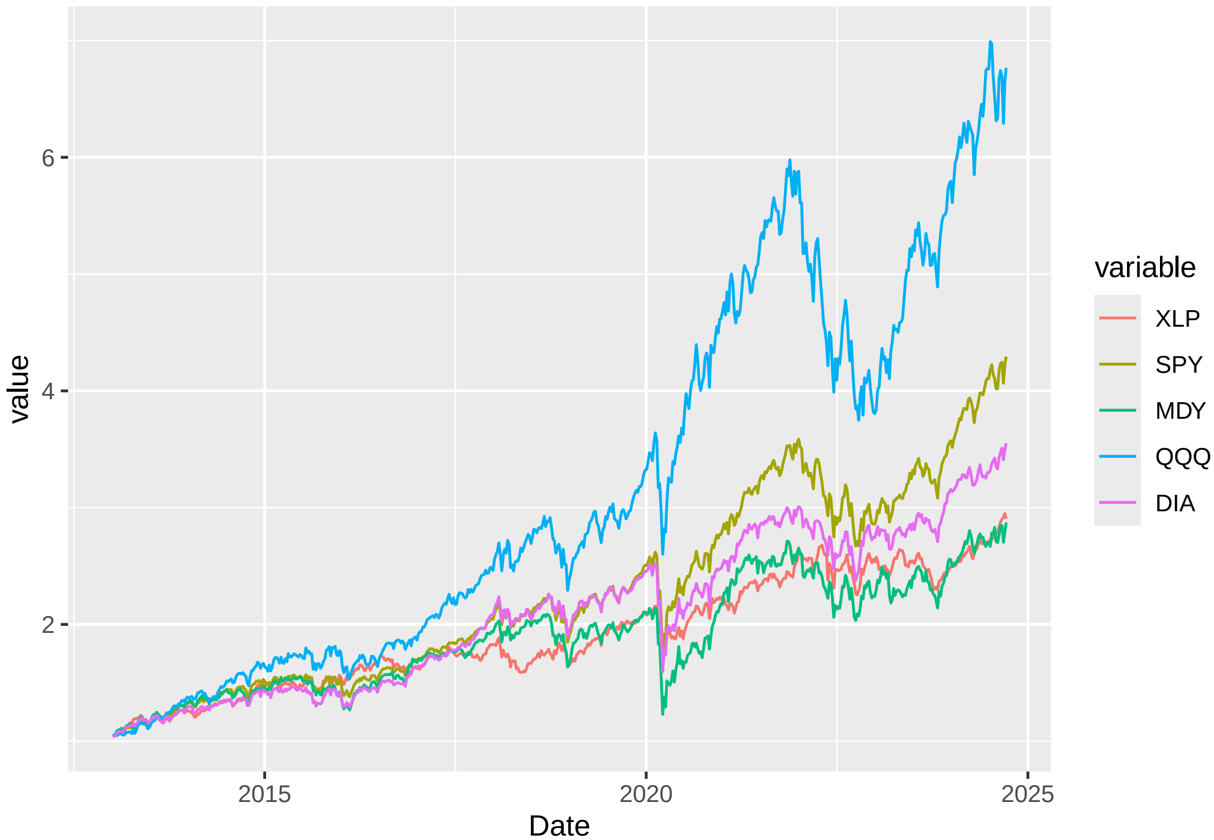


2.5 Compute/plot cumulative returns

```r
# Convert df0 to cumulative return series
# Allstarting at 1.

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

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

```r
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)
```



6
4
2
value
2015
2020
2025
Date
variable
XLP
SPY
MDY
QQQ
DIA

3. Regression of Sector ETF on Index ETFs

3.1 Fit regression of y0 on index etf returns

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

fit.summary
```

```
##
## Call:
## lm(formula = y0 ~ ., data = df0)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.063859 -0.006274  0.000033  0.007041  0.048673
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.0002070  0.0004662   0.444    0.657
## SPY          1.0069903  0.1632673   6.168 1.27e-09 ***
## MDY         -0.3683498  0.0483853  -7.613 1.03e-13 ***
## QQQ         -0.3142920  0.0640313  -4.908 1.18e-06 ***
## DIA          0.3995081  0.0908524   4.397 1.29e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01141 on 607 degrees of freedom
## Multiple R-squared:  0.6418, Adjusted R-squared:  0.6394
## F-statistic: 271.9 on 4 and 607 DF,  p-value: < 2.2e-16
```

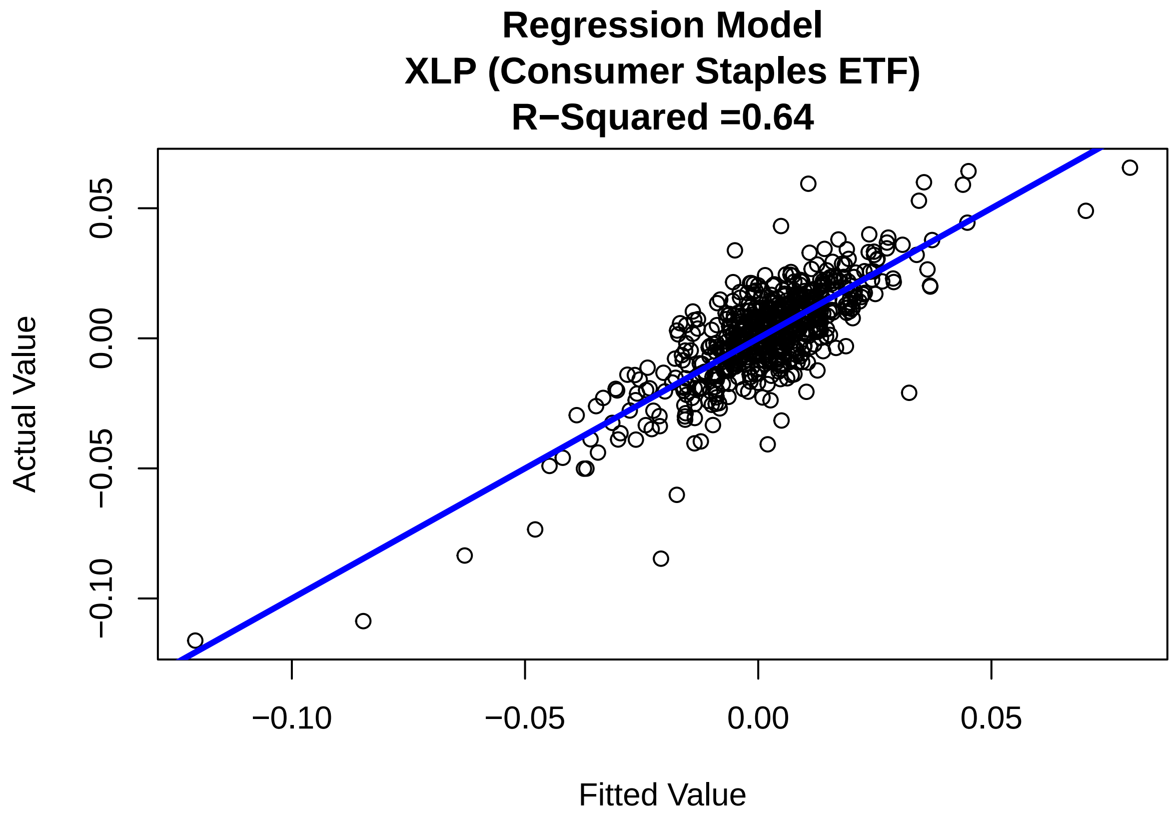
```r
#   Note list components of output from summary()
names(fit.summary)
```

```
##  [1] "call"          "terms"         "residuals"     "coefficients"
##  [5] "aliased"       "sigma"         "df"            "r.squared"
##  [9] "adj.r.squared" "fstatistic"    "cov.unscaled"
```

```r
#   print method for fit.summary is condensed
```

3.2 Plot Actual versus Fitted Values

```r
 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)
```



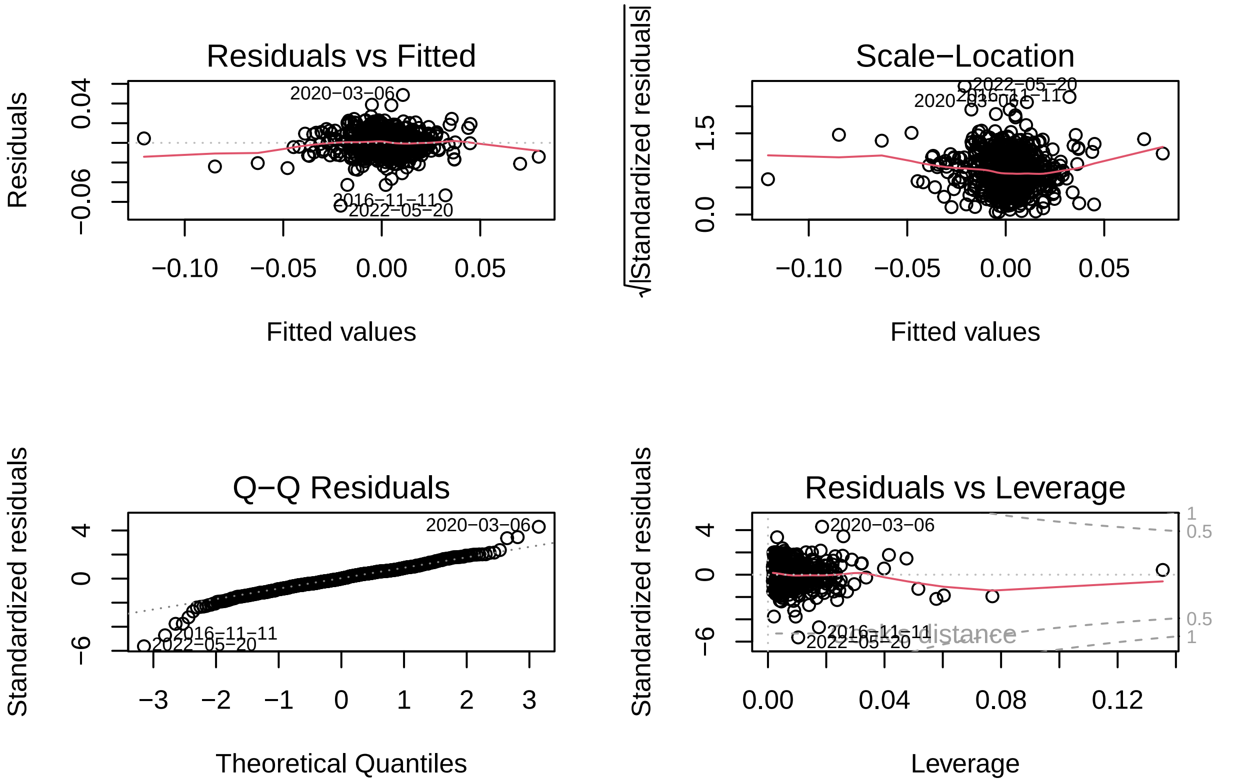
Regression Model
XLP (Consumer Staples ETF)
R−Squared =0.64
Actual Value
Fitted Value
0.05
0.00
−0.05
−0.10
−0.10
−0.05
0.00
0.05

4. Use R package car to analyse residuals/influence

```r
library(car)
```

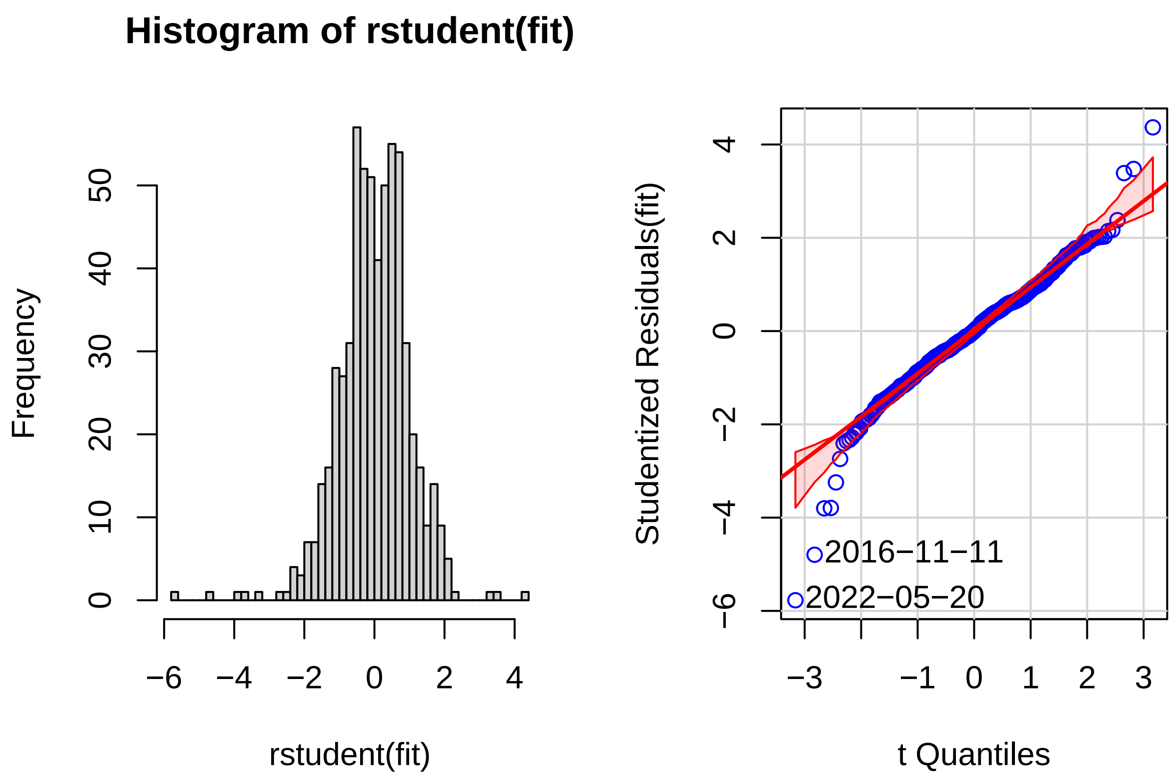
4.1 plot() method gives 4-panel plots of residuals and Leverage

```r
  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() -

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

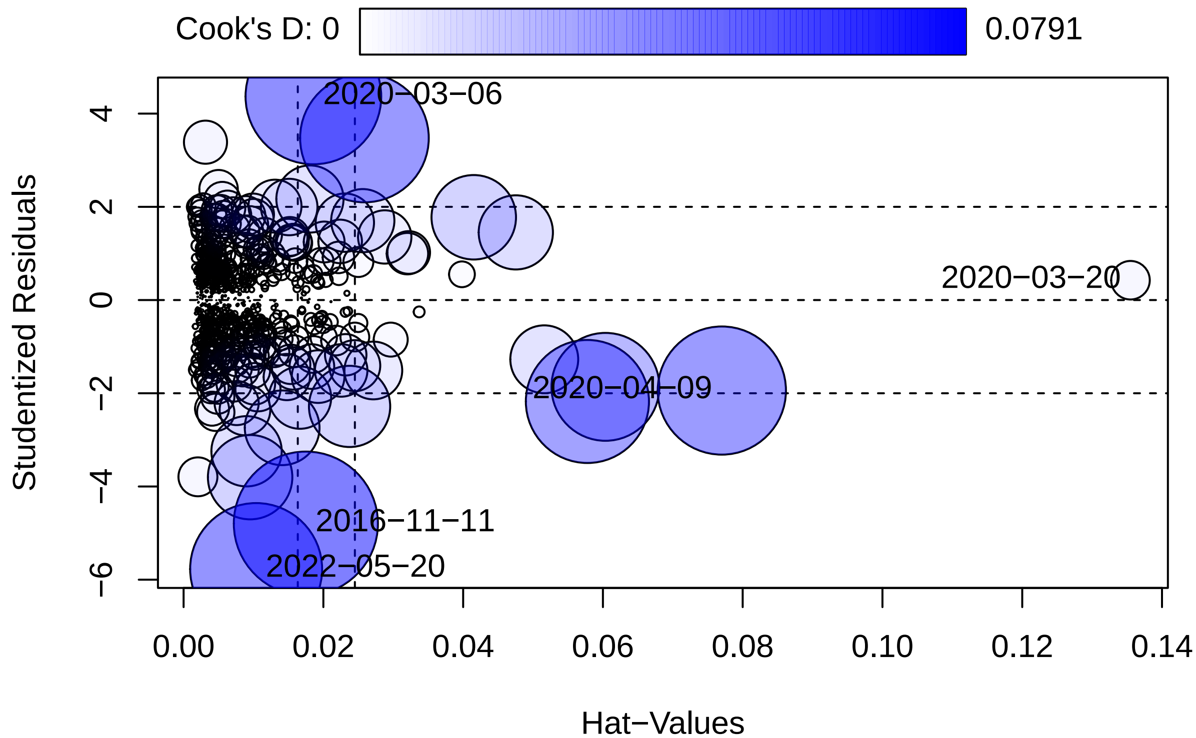



```
## 2016-11-11 2022-05-20
##        202        490
```

```
#
```

4.3 influencePlot() output

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

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

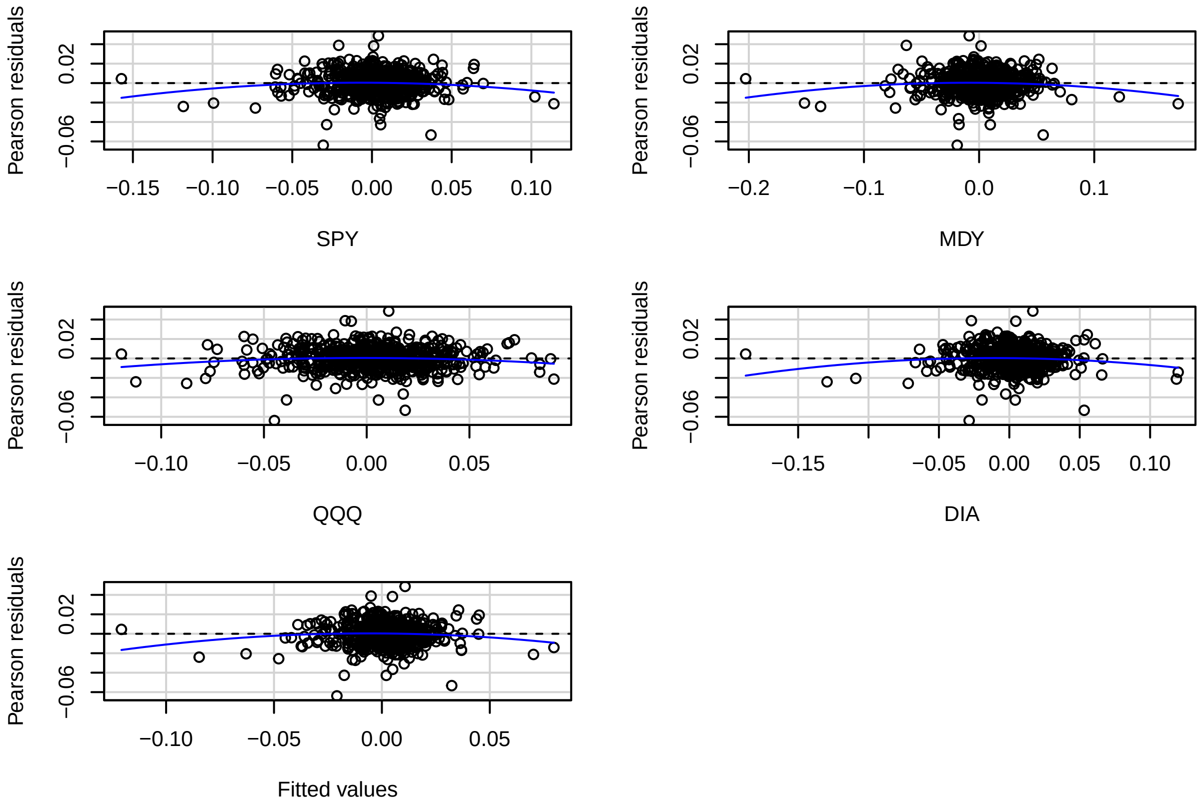



```
##                StudRes        Hat       CookD
## 2016-11-11 -4.7964026 0.01749576 0.079066753
## 2020-03-06  4.3678362 0.01857384 0.070123119
## 2020-03-20  0.4249459 0.13549138 0.005667955
## 2020-04-09 -1.9418203 0.07701466 0.062639579
## 2022-05-20 -5.7714057 0.01037971 0.066341866
```

4.4 residualPlots()

Visually test for curvature in linear regression terms.

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



Print out test results

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

```
##            Test stat Pr(>|Test stat|)
## SPY          -2.2498          0.02482 *
## MDY          -2.2815          0.02287 *
## QQQ          -2.0270          0.04310 *
## DIA          -2.3455          0.01932 *
## Tukey test   -2.2004          0.02778 *
## ---
## 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
## -0.063859 -0.006274  0.000033  0.007041  0.048673
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.0002070  0.0004662   0.444    0.657
## SPY          1.0069903  0.1632673   6.168 1.27e-09 ***
## MDY         -0.3683498  0.0483853  -7.613 1.03e-13 ***
## QQQ         -0.3142920  0.0640313  -4.908 1.18e-06 ***
## DIA          0.3995081  0.0908524   4.397 1.29e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01141 on 607 degrees of freedom
## Multiple R-squared:  0.6418, Adjusted R-squared:  0.6394
## F-statistic: 271.9 on 4 and 607 DF,  p-value: < 2.2e-16
```

```
fit$coefficients
```

```
##   (Intercept)           SPY           MDY           QQQ           DIA
##  0.0002070038  1.0069902798 -0.3683498406 -0.3142920210  0.3995081430
```

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

```
##            (Intercept)         SPY         MDY          QQQ           DIA
## 2013-01-04           1 0.044280965 0.049746267  0.043715341  3.793252e-02
## 2013-01-11           1 0.004771257 0.001196815  0.009410744  4.910796e-03
## 2013-01-18           1 0.008530791 0.015020782 -0.002828828  1.118619e-02
## 2013-01-25           1 0.012861058 0.022542269 -0.001044416  1.841951e-02
## 2013-02-01           1 0.006567324 0.004198834  0.009802910  8.047782e-03
## 2013-02-08           1 0.003695823 0.007057868  0.004865621 -7.152686e-05
```

```
#
```

5.2 Repeat regression with Z-scores of index etfs

```r
# 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
## -0.063859 -0.006274  0.000033  0.007041  0.048673
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.0019266  0.0004614   4.175 3.41e-05 ***
## SPY          0.0224507  0.0036400   6.168 1.27e-09 ***
## MDY         -0.0100001  0.0013136  -7.613 1.03e-13 ***
## QQQ         -0.0082541  0.0016816  -4.908 1.18e-06 ***
## DIA          0.0090240  0.0020522   4.397 1.29e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01141 on 607 degrees of freedom
## Multiple R-squared:  0.6418, Adjusted R-squared:  0.6394
## F-statistic: 271.9 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 0.9175267 0.9181598 0.9540900
## MDY 0.9175267 1.0000000 0.7689490 0.9092160
## QQQ 0.9181598 0.7689490 1.0000000 0.7977508
## DIA 0.9540900 0.9092160 0.7977508 1.0000000
```

```
var(x)
```

```
##           SPY       MDY       QQQ       DIA
## SPY 1.0000000 0.9175267 0.9181598 0.9540900
## MDY 0.9175267 1.0000000 0.7689490 0.9092160
## QQQ 0.9181598 0.7689490 1.0000000 0.7977508
## DIA 0.9540900 0.9092160 0.7977508 1.0000000
```

6.2 Compute PCA of x

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

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

```
## Importance of components:
##                           Comp.1     Comp.2     Comp.3      Comp.4
## Standard deviation     1.9053417 0.51045028 0.30208710 0.106400323
## Proportion of Variance 0.9090671 0.06524648 0.02285149 0.002834889
## Cumulative Proportion  0.9090671 0.97431362 0.99716511 1.000000000
```

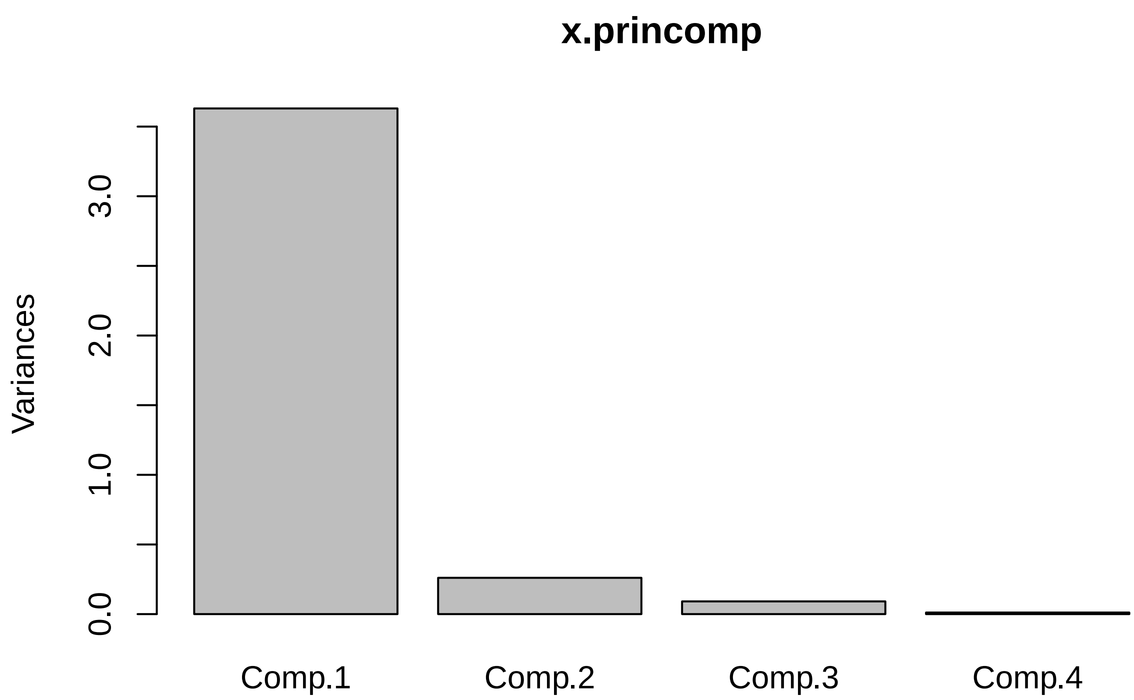
Note available output from princomp() is a named list:

```
names(x.princomp)
```

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

6.3 The Screeplot: Barplot of Principal Component Variances

```
screeplot(x.princomp)
```



6.4 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
```

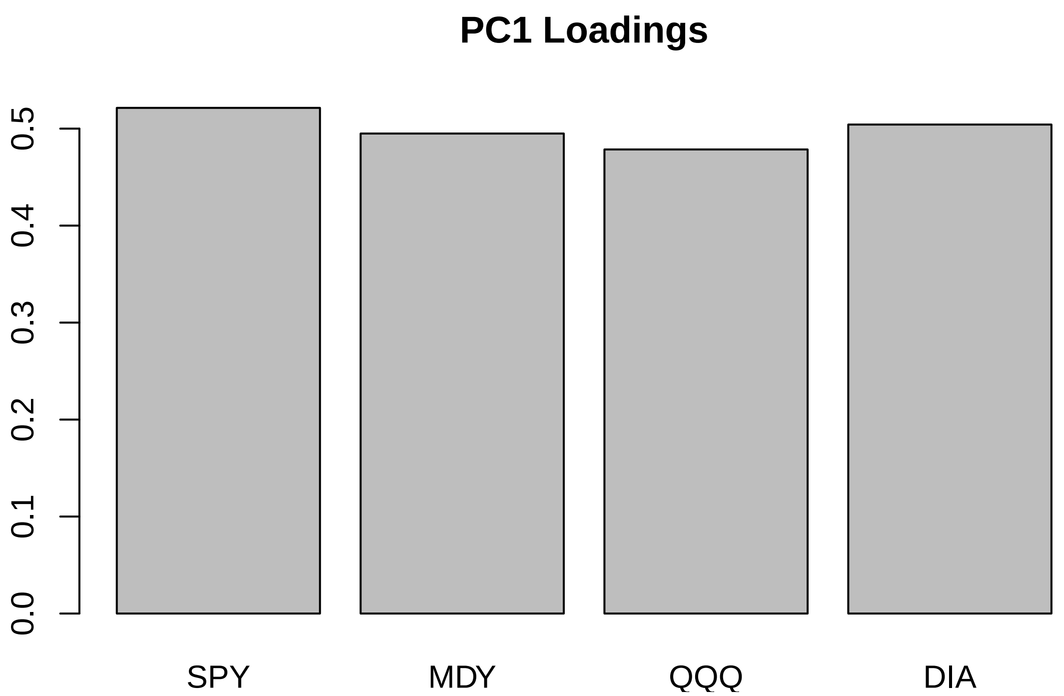
```
# Get all values with no cutoffs
print(x.princomp$loadings,cutoff=0.)
```

```
##
## 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
```

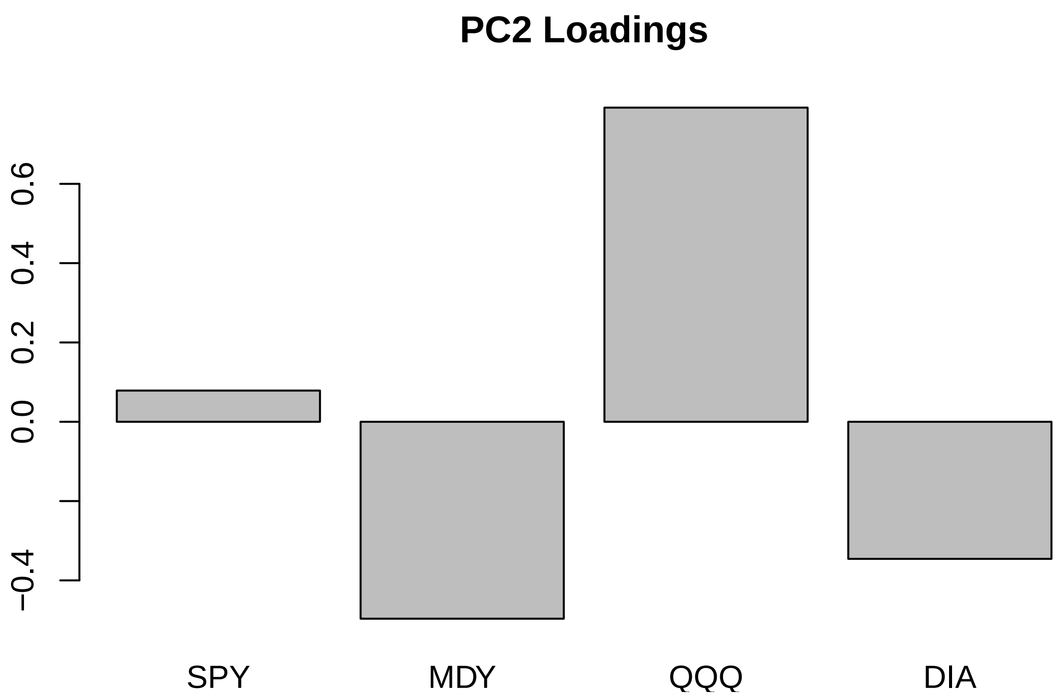
```
## Cumulative Var   0.25   0.50   0.75   1.00
```

Plotting the loadings of each Principal Component variable

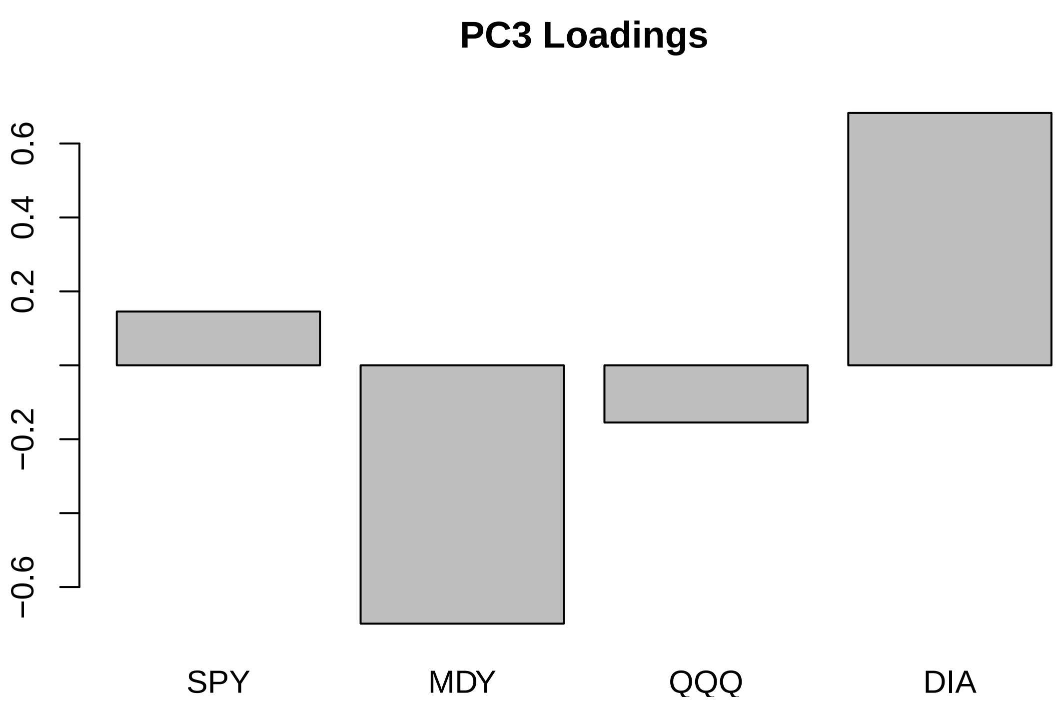
```r
# Plot loadings of PC1-PC4

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



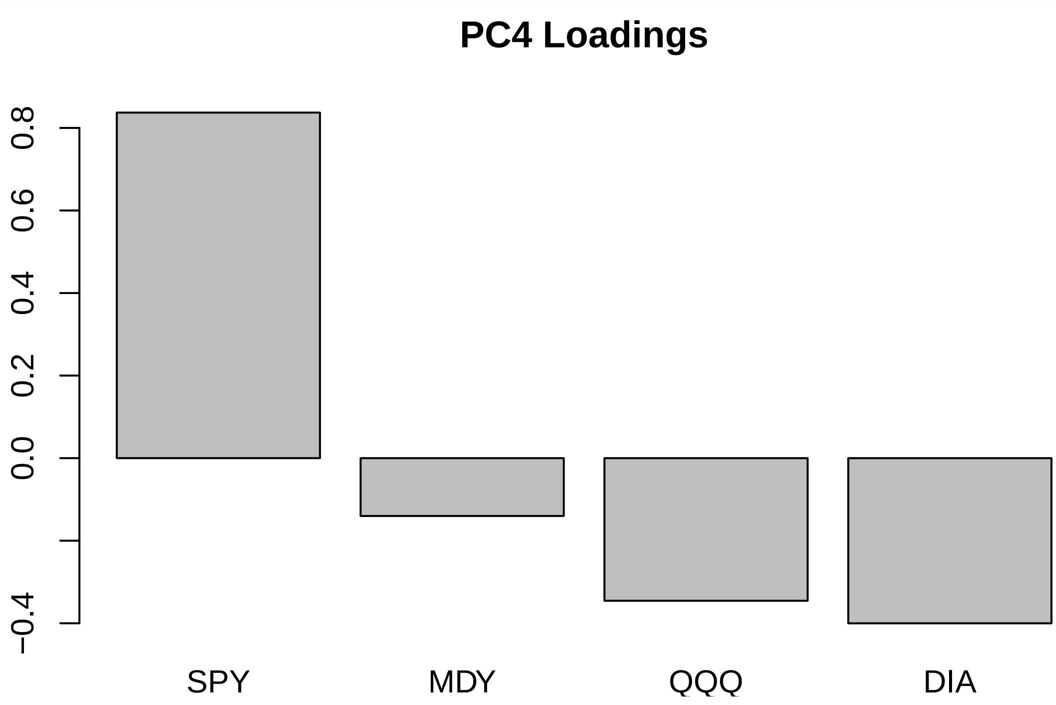

```r
barplot(x.princomp$loadings[,2], main="PC2 Loadings")
```




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




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

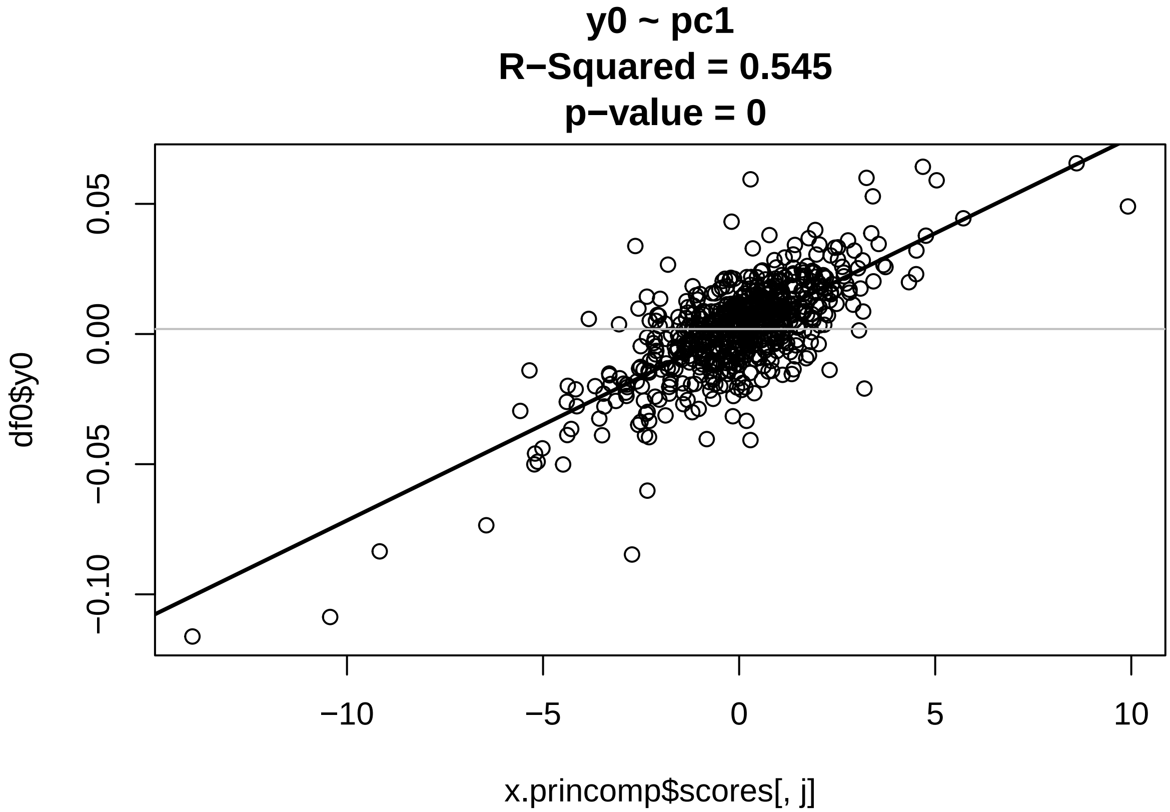


7. Regressions on PC variables

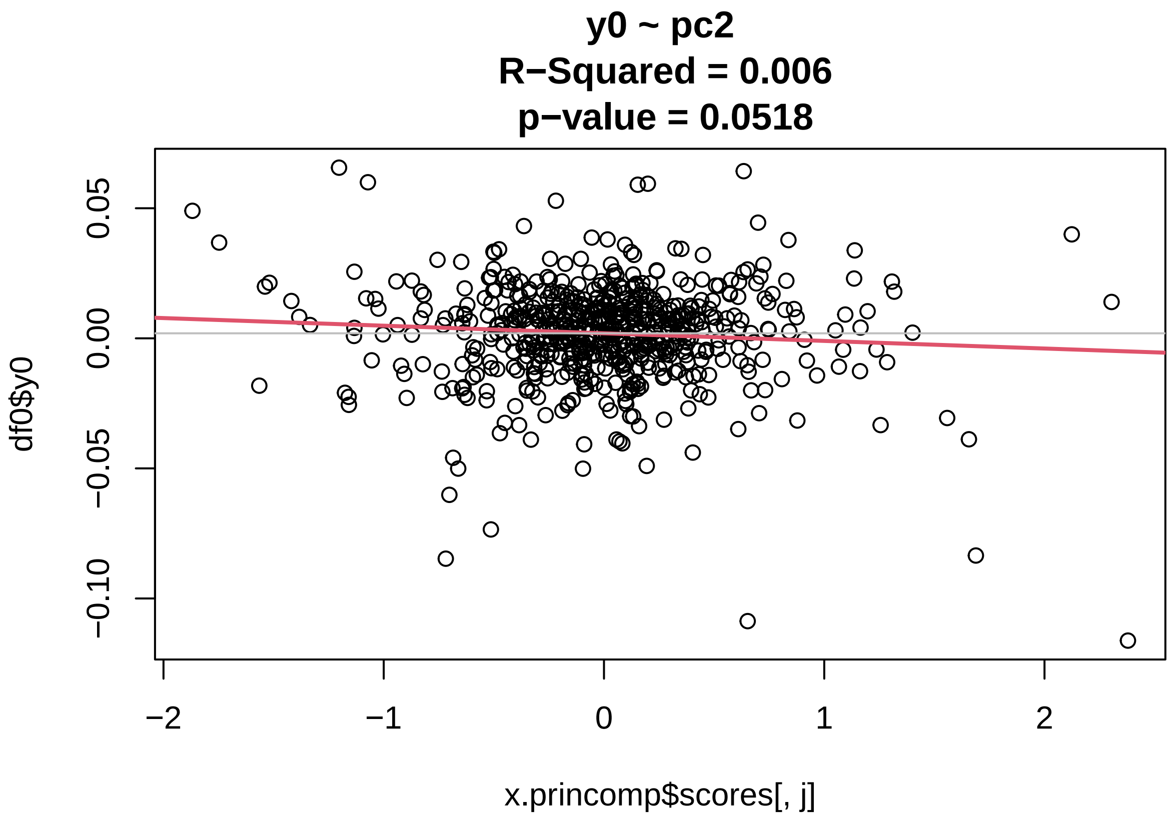
7.1 Univariate regressions

```r
# 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=""))
}
```

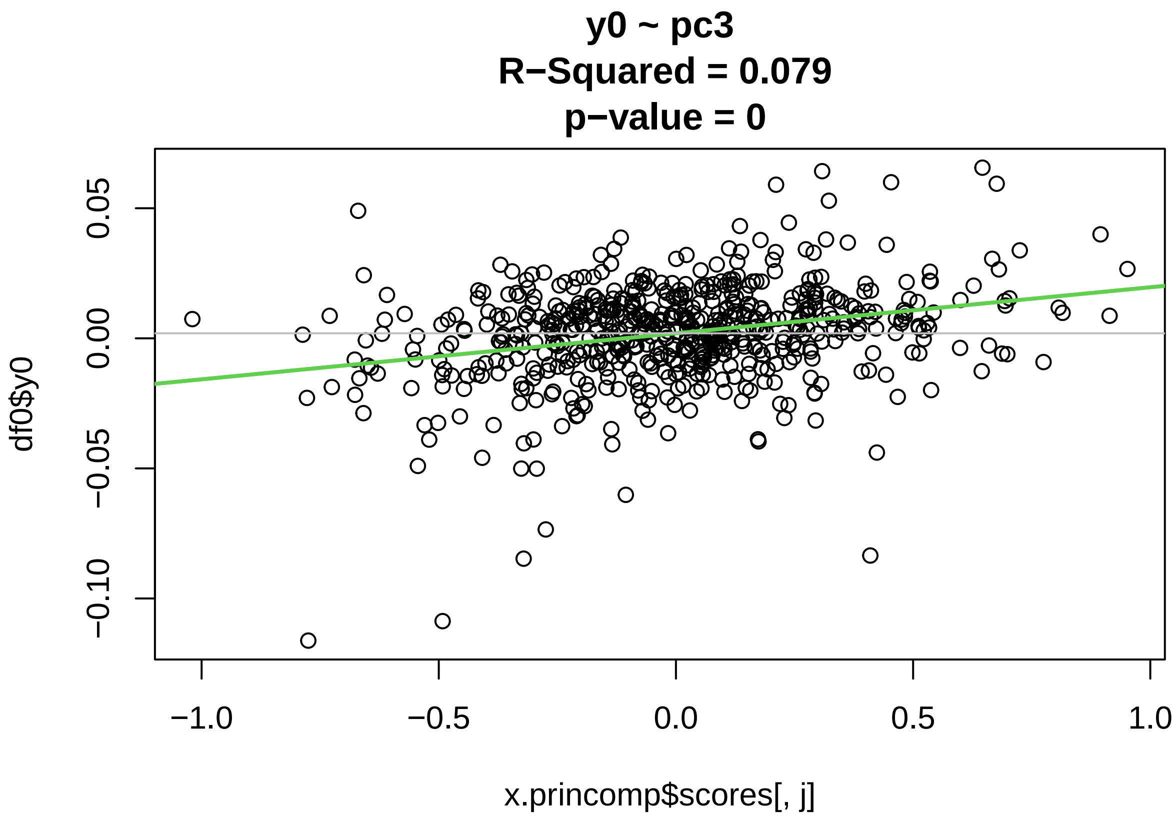
```
## Residual Standard Error=0.0128
## R-Square=0.5445
## F-statistic (df=1, 610)=729.3205
## p-value=0
##
##           Estimate Std.Err t-value Pr(>|t|)
## Intercept   0.0019   5e-04  3.7121    2e-04
## X           0.0074   3e-04 27.0059    0e+00
```



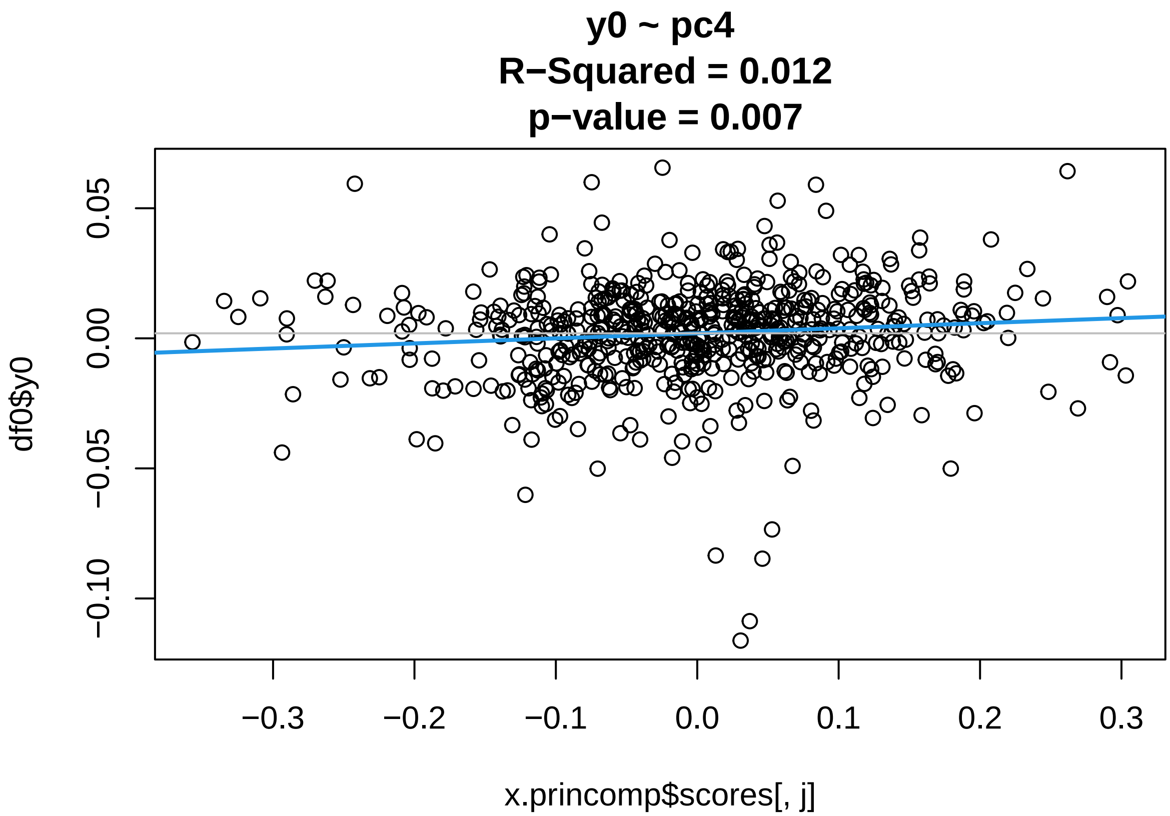

```
## Residual Standard Error=0.019
## R-Square=0.0062
## F-statistic (df=1, 610)=3.797
## p-value=0.0518
##
##            Estimate Std.Err t-value Pr(>|t|)
## Intercept    0.0019  0.0008  2.5130   0.0122
## X           -0.0029  0.0015 -1.9486   0.0518
```




```
## Residual Standard Error=0.0183
## R-Square=0.0792
## F-statistic (df=1, 610)=52.4679
## p-value=0
##
##           Estimate Std.Err t-value Pr(>|t|)
## Intercept   0.0019  0.0007  2.6107   0.0093
## X           0.0177  0.0024  7.2435   0.0000
```




```
## Residual Standard Error=0.0189
## R-Square=0.0119
## F-statistic (df=1, 610)=7.319
## p-value=0.007
##
##           Estimate Std.Err t-value Pr(>|t|)
## Intercept   0.0019  0.0008  2.5202    0.012
## X           0.0194  0.0072  2.7054    0.007
```



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
## -0.063859 -0.006274  0.000033  0.007041  0.048673
##
## Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)                0.0019266  0.0004614   4.175 3.41e-05 ***
## x.princomp$scoresComp.1  0.0073563  0.0002422  30.377  < 2e-16 ***
## x.princomp$scoresComp.2 -0.0029266  0.0009039  -3.238  0.00127 **
## x.princomp$scoresComp.3  0.0176948  0.0015274  11.585  < 2e-16 ***
## x.princomp$scoresComp.4  0.0194375  0.0043366   4.482 8.83e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01141 on 607 degrees of freedom
## Multiple R-squared:  0.6418, Adjusted R-squared:  0.6394
## F-statistic: 271.9 on 4 and 607 DF,  p-value: < 2.2e-16
```

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

7.3 Use PCA regression to recompute regression parameters

```r
#       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
```

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

```
##  (Intercept)          SPY          MDY          QQQ          DIA
##  0.001926583  0.022450726 -0.010000091 -0.008254136  0.009023990
```

7.4 Compute regression parameter based on only first 3 pc vars

```r
pcbetavec
```

```
## [1]  0.007356254 -0.002926641  0.017694763  0.019437483
```

```r
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).

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

pcbetavectokeep<-pcbetavec*ind.tokeep

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

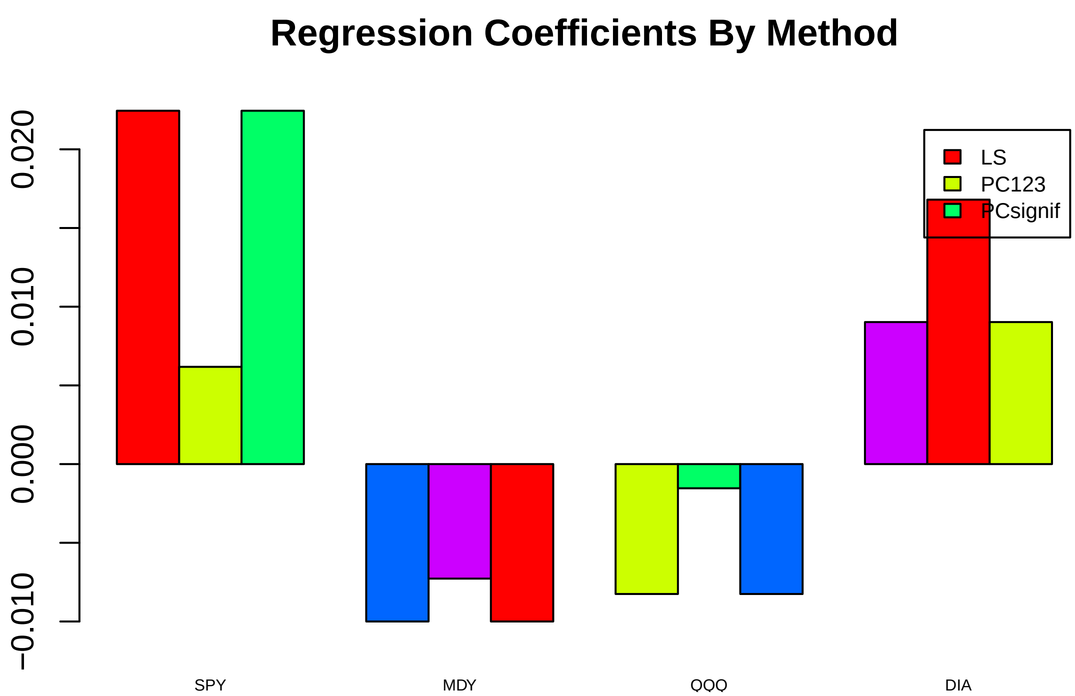
7.6 Create table of betas from these three fits

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

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

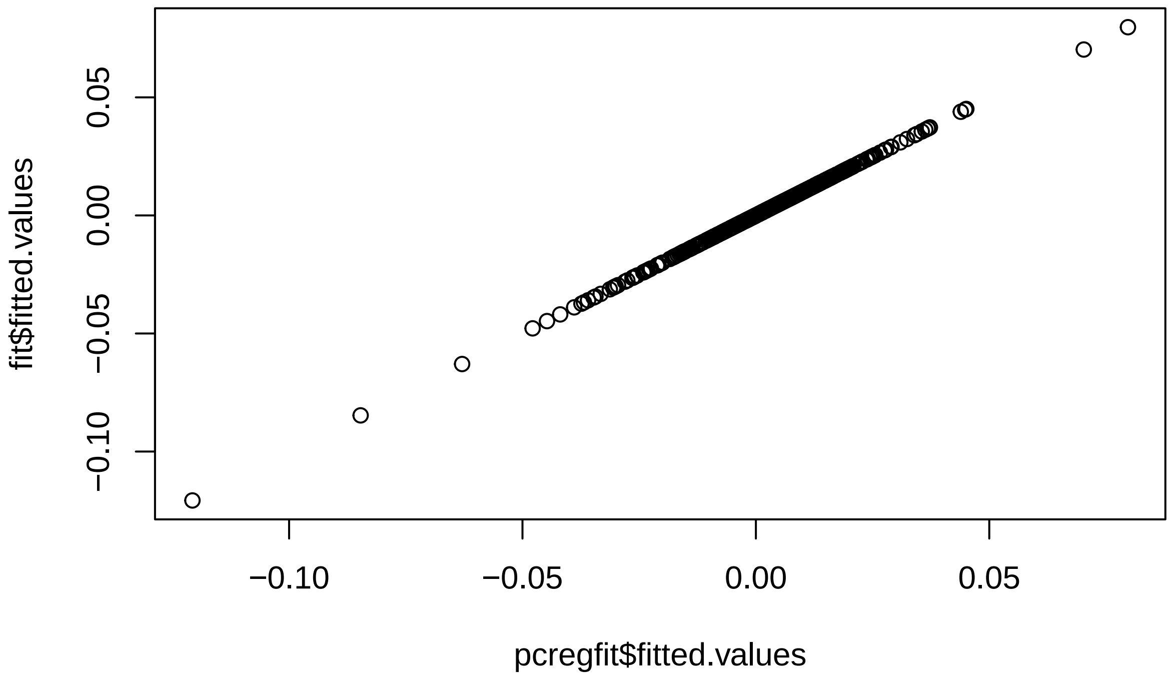
```
##                  LS        PC123     PCsignif
## SPY  0.022450726  0.006178644  0.022450726
## MDY -0.010000091 -0.007275983 -0.010000091
## QQQ -0.008254136 -0.001537180 -0.008254136
## DIA  0.009023990  0.016801960  0.009023990
```

```r
# 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

```r
# pcregfit equals LS fit

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



fit$fitted.values
0.05
0.00
−0.05
−0.10
−0.10
−0.05
0.00
0.05
pcregfit$fitted.values

8. Ridge and Lasso Regression Fits

8.1 Load glmnet package for ridge/lasso regressions

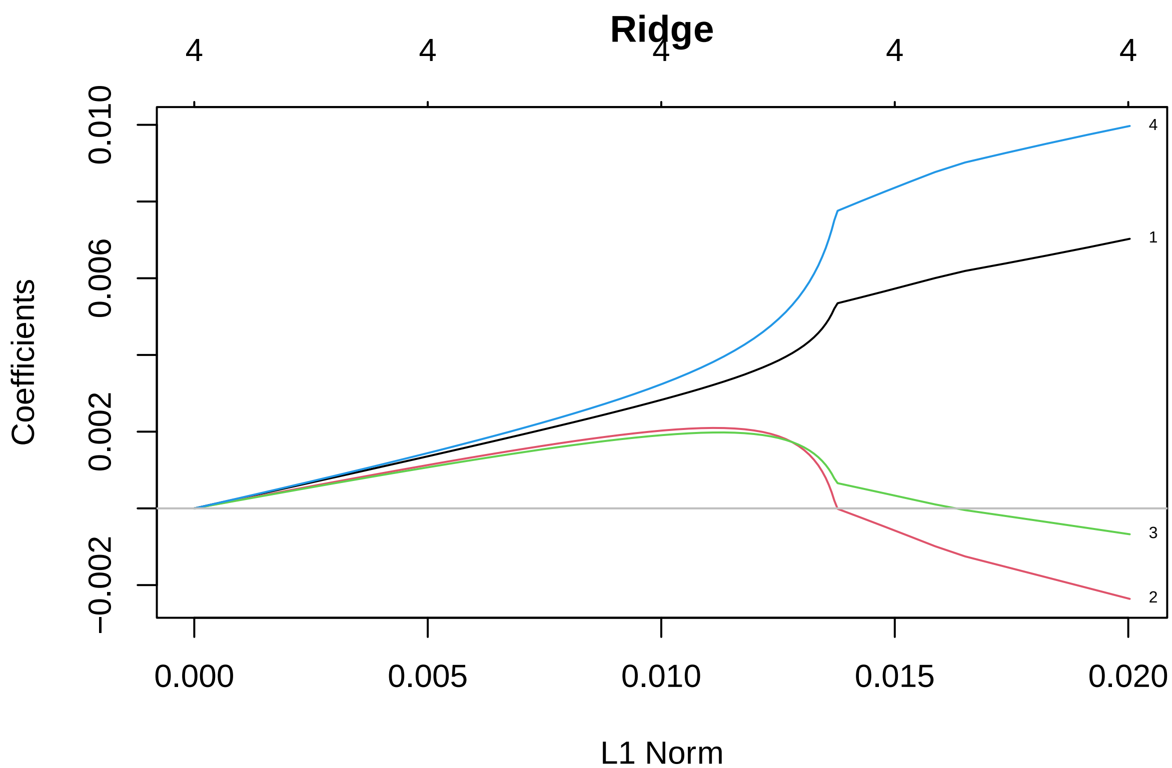
```
library(glmnet)
```

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

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

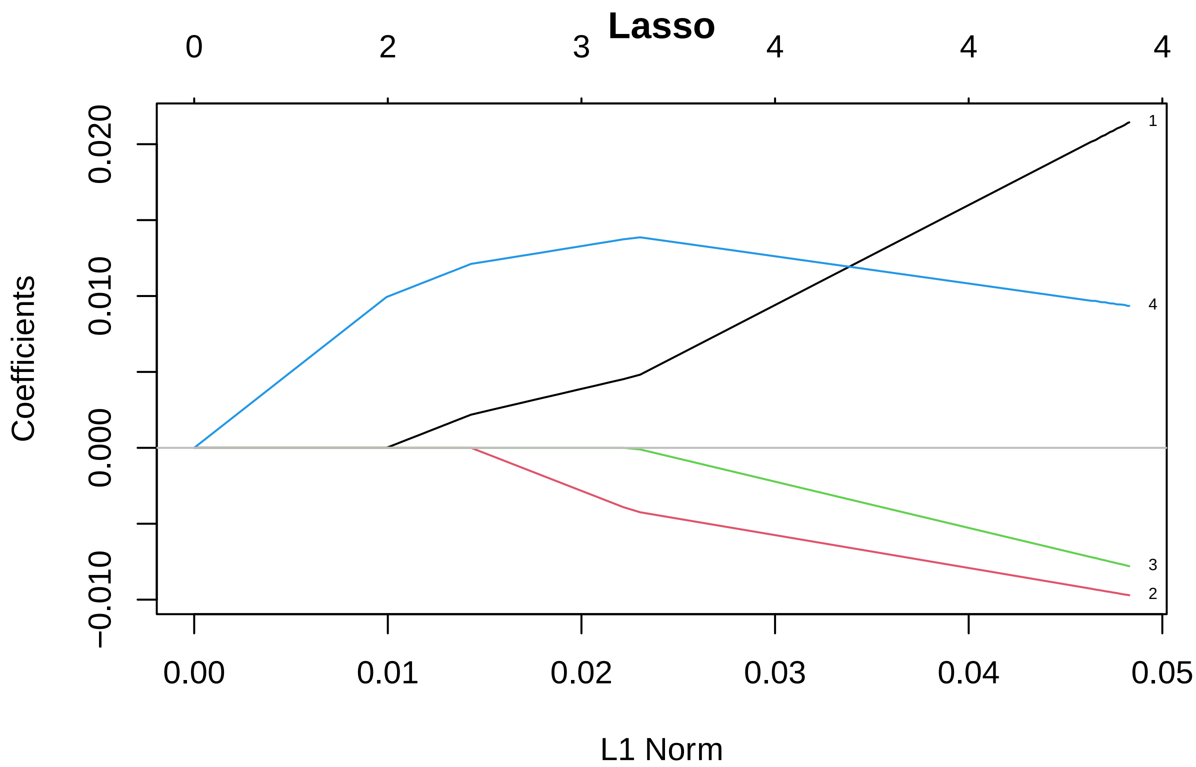
8.3 Plot coefficient trace of ridge regression model

```
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)
```

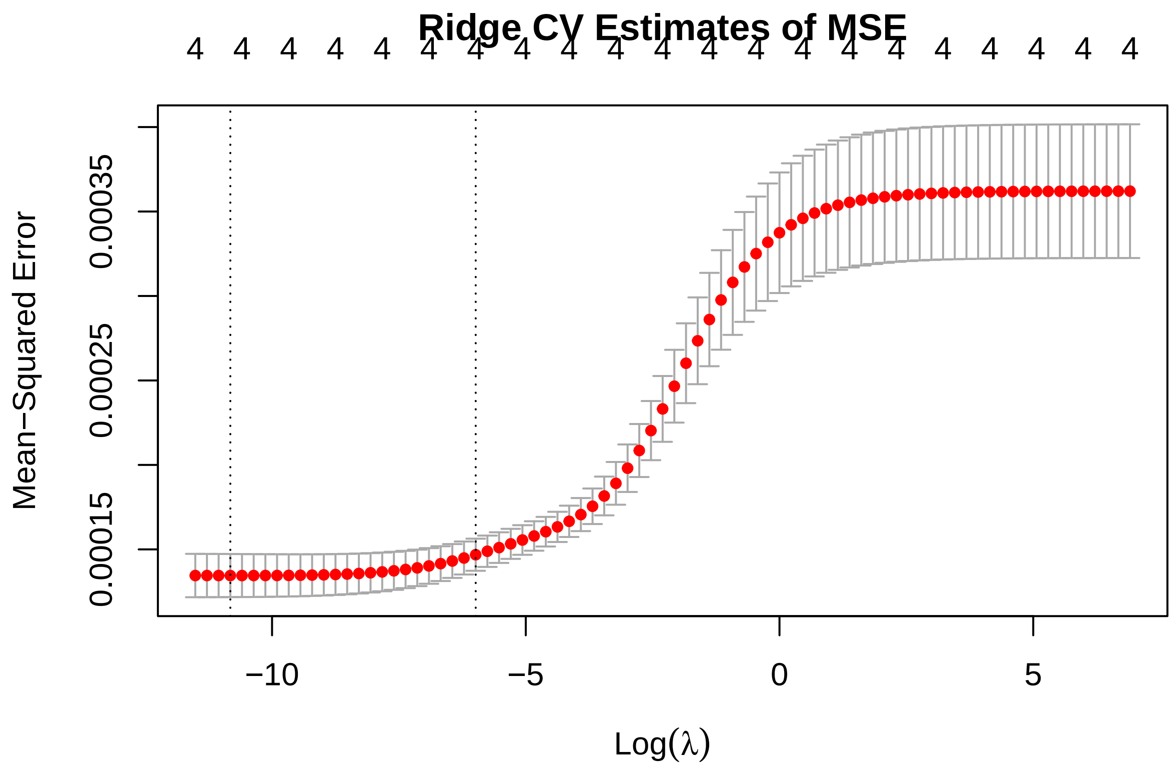
```
##           Length Class     Mode
## a0        100    -none-    numeric
## beta      400    dgCMatrix S4
## df        100    -none-    numeric
## dim         2    -none-    numeric
## lambda    100    -none-    numeric
## dev.ratio 100    -none-    numeric
## nulldev     1    -none-    numeric
## 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
```

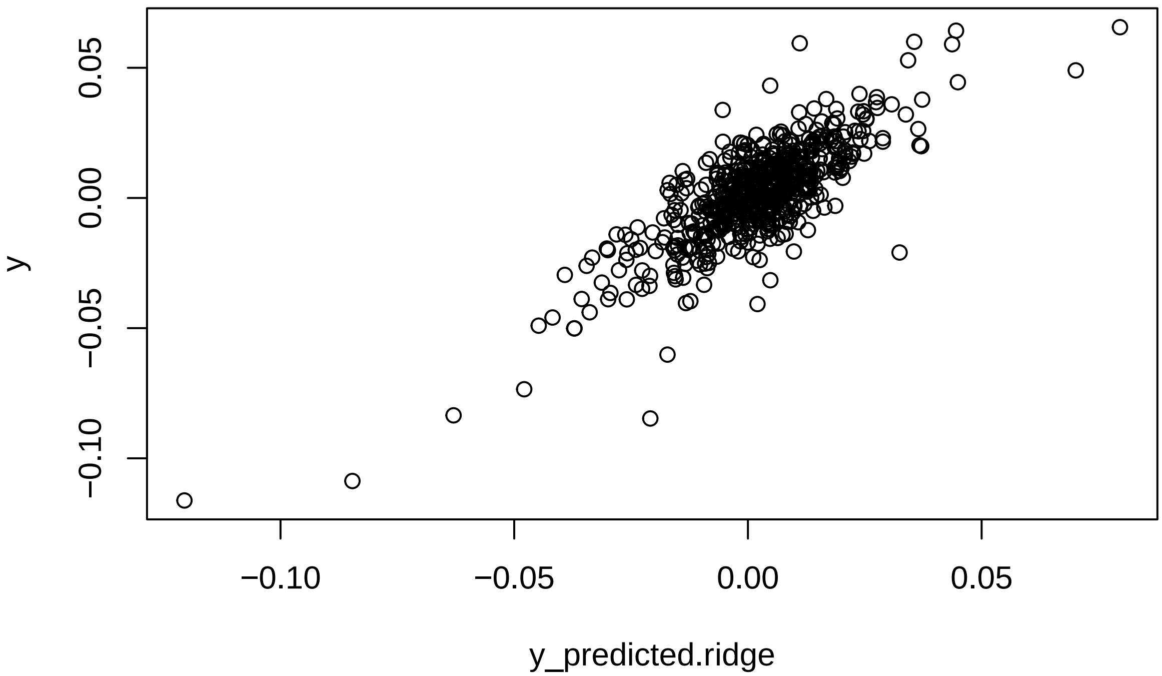
```
y.cv.glmnet.ridge<-cv.glmnet(x,y,alpha=0, lambda=lambdas)
plot(y.cv.glmnet.ridge, main="Ridge CV Estimates of MSE")
```




```
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

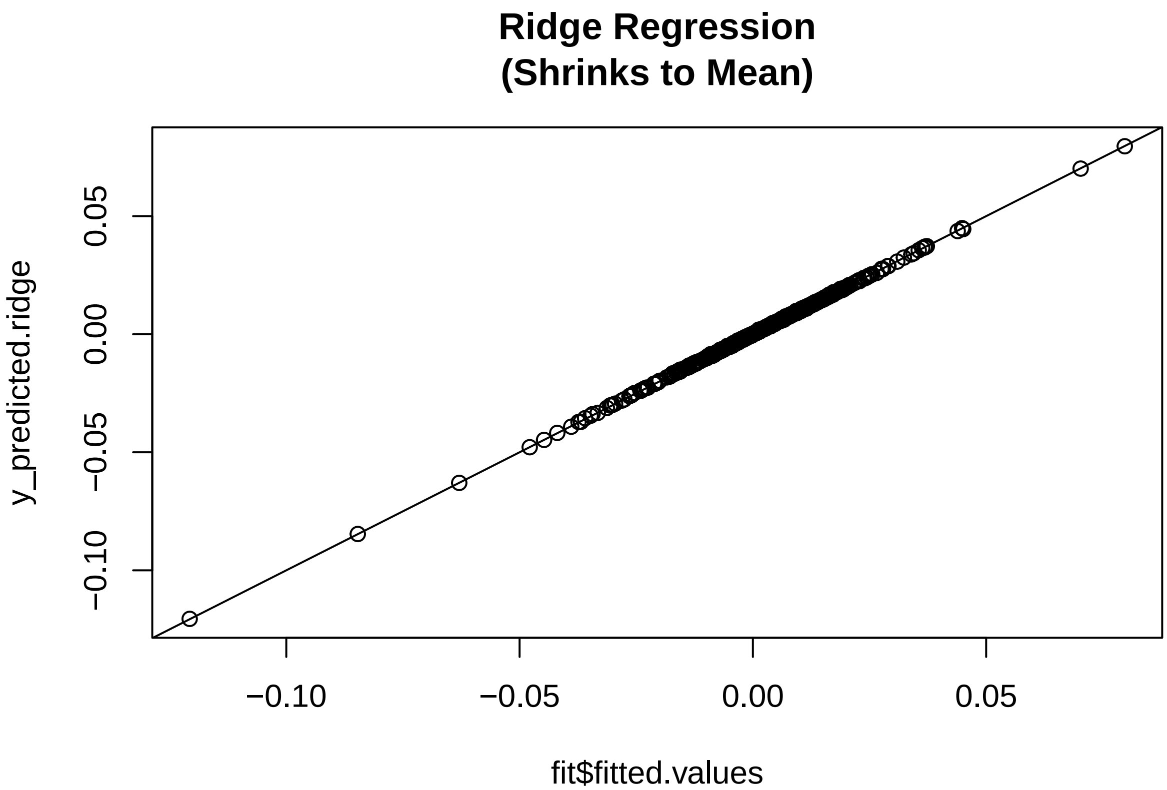
```
plot(y_predicted.ridge, y)
```



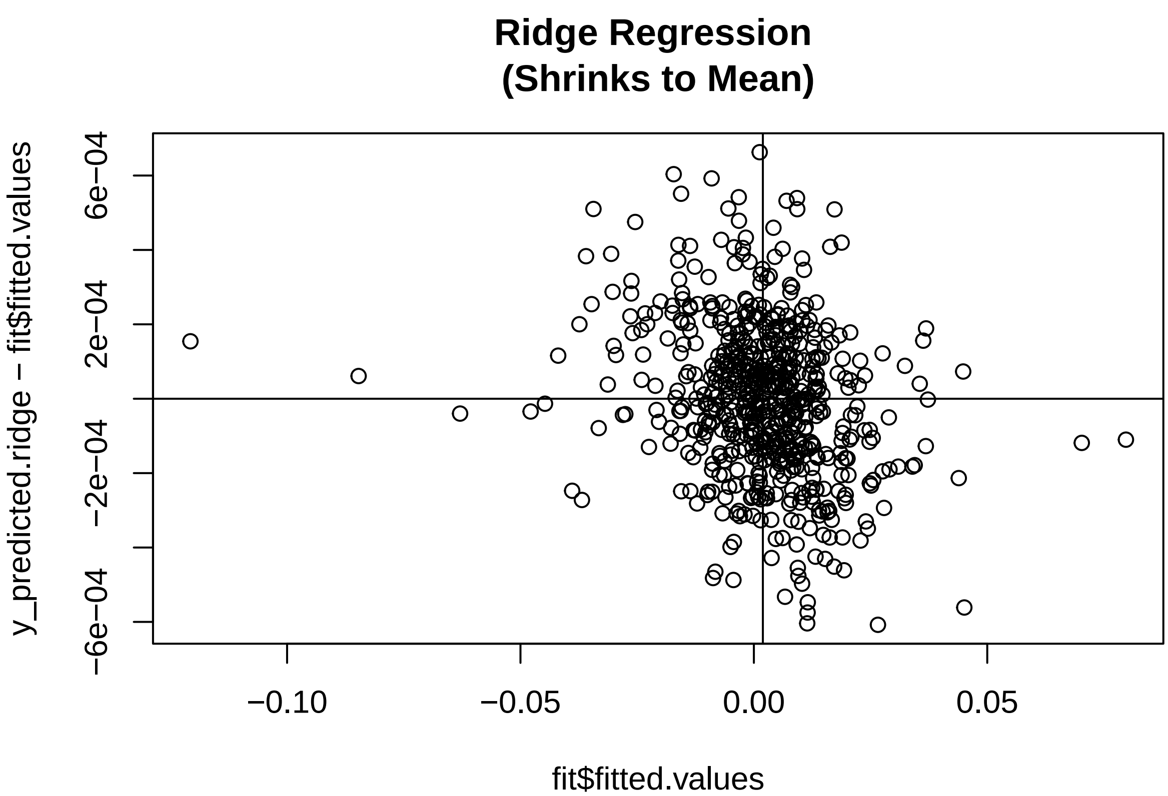

```r
cor(y_predicted.ridge,y)^2
```

```
##         [,1]
## s1 0.6416741
```

```r
# 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)
```

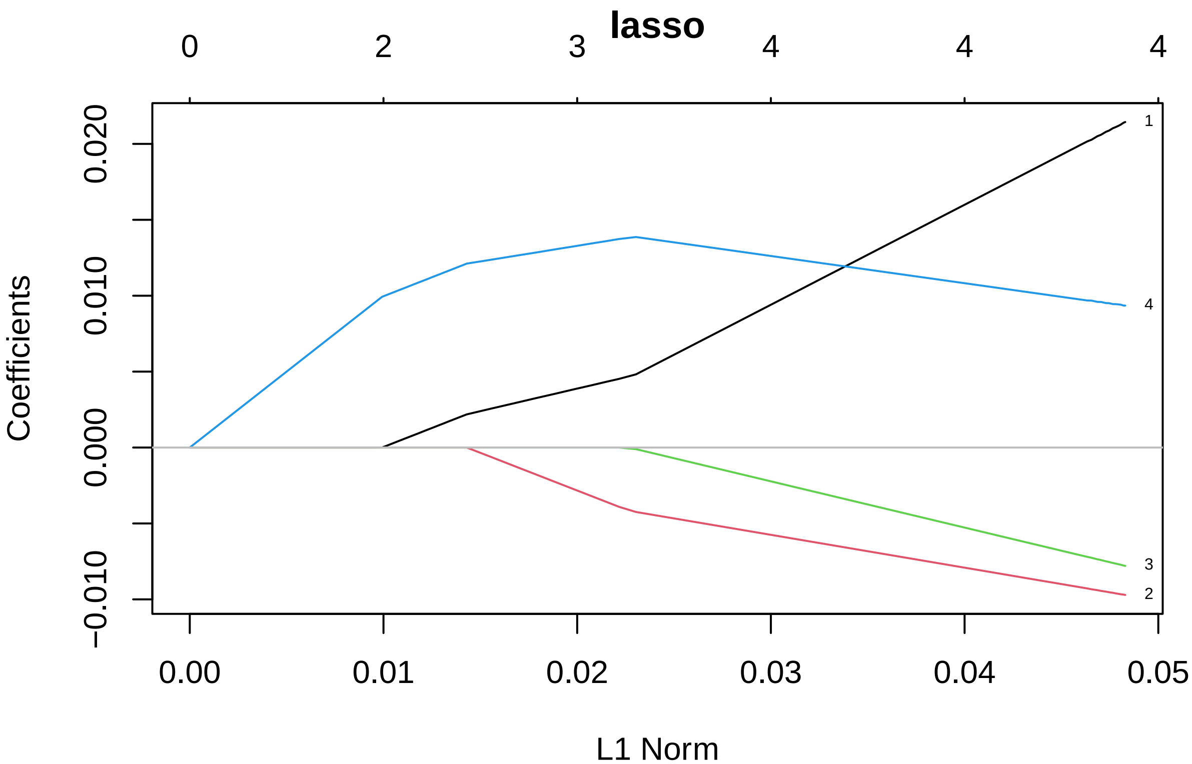



```
plot(fit$fitted.values, y_predicted.ridge -fit$fitted.values,
     main="Ridge Regression \n(Shrinks to Mean)")
abline(h=0)
abline(v=mean(fit$fitted.values))
```



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)
```

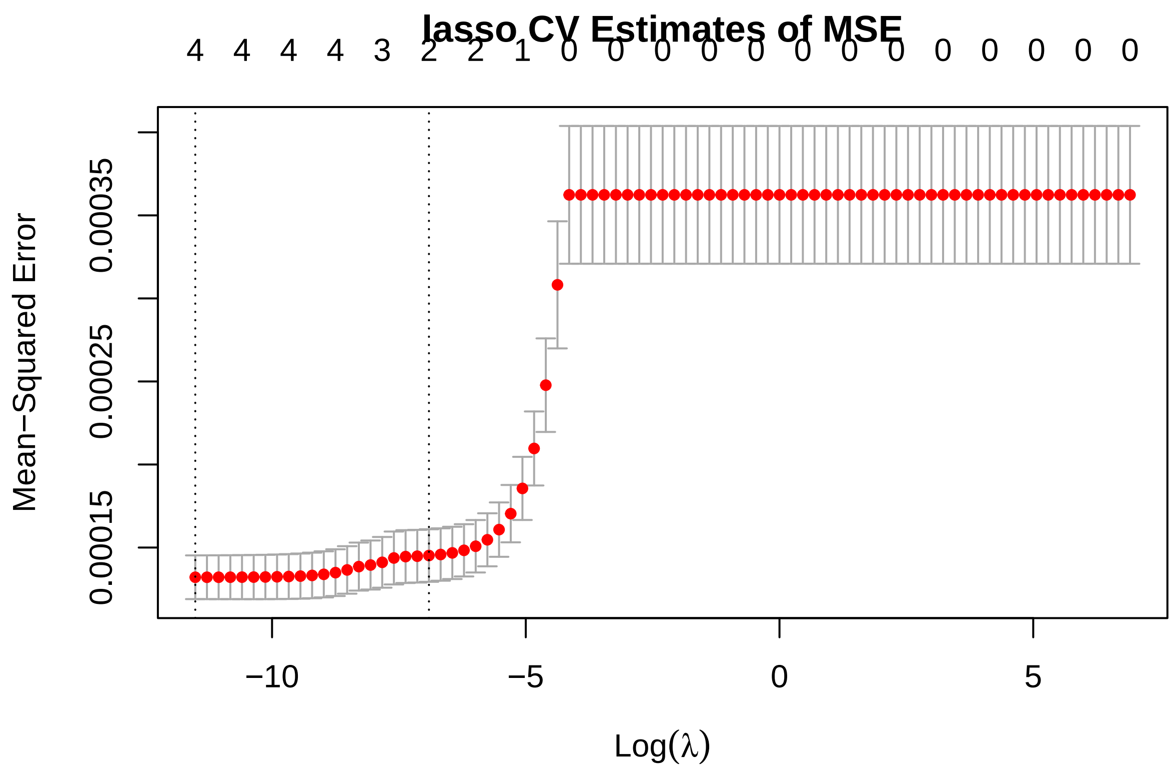
```
##           Length Class     Mode
## a0         79    -none-    numeric
## beta      316    dgCMatrix S4
## df         79    -none-    numeric
## dim         2    -none-    numeric
## lambda     79    -none-    numeric
## dev.ratio  79    -none-    numeric
## nulldev     1    -none-    numeric
## 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)
```

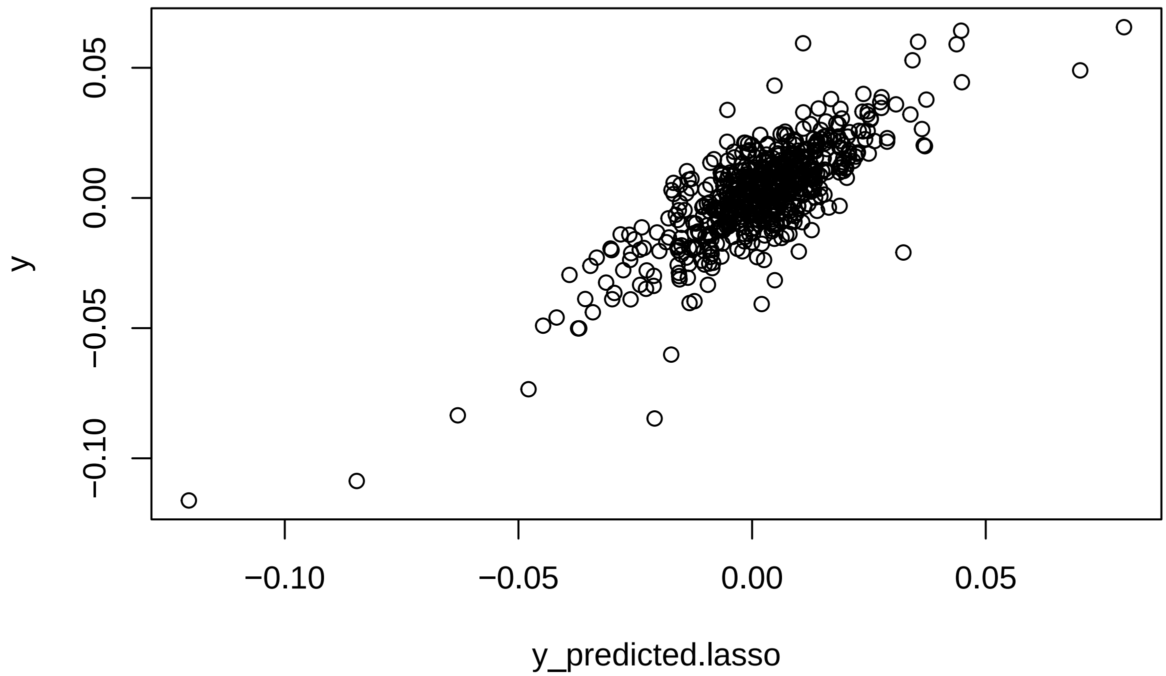
```r
summary(y.glmnet.lasso)
```

```
##           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
```

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



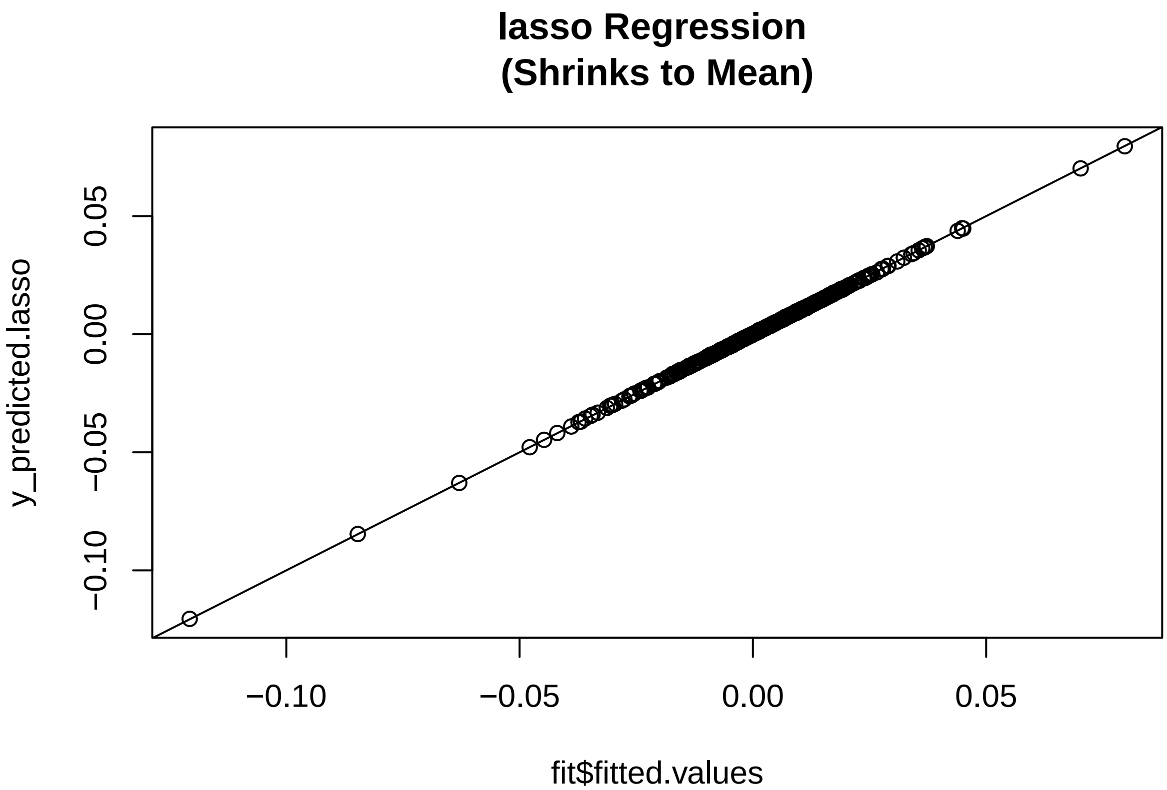

```r
optlambda.lasso<-y.cv.glmnet.lasso$lambda.min
glmnet.lassofit<-y.cv.glmnet.lasso$glmnet.fit
y_predicted.lasso<- predict(glmnet.lassofit,s=optlambda.lasso,newx=x)
#
plot(y_predicted.lasso, y)
```



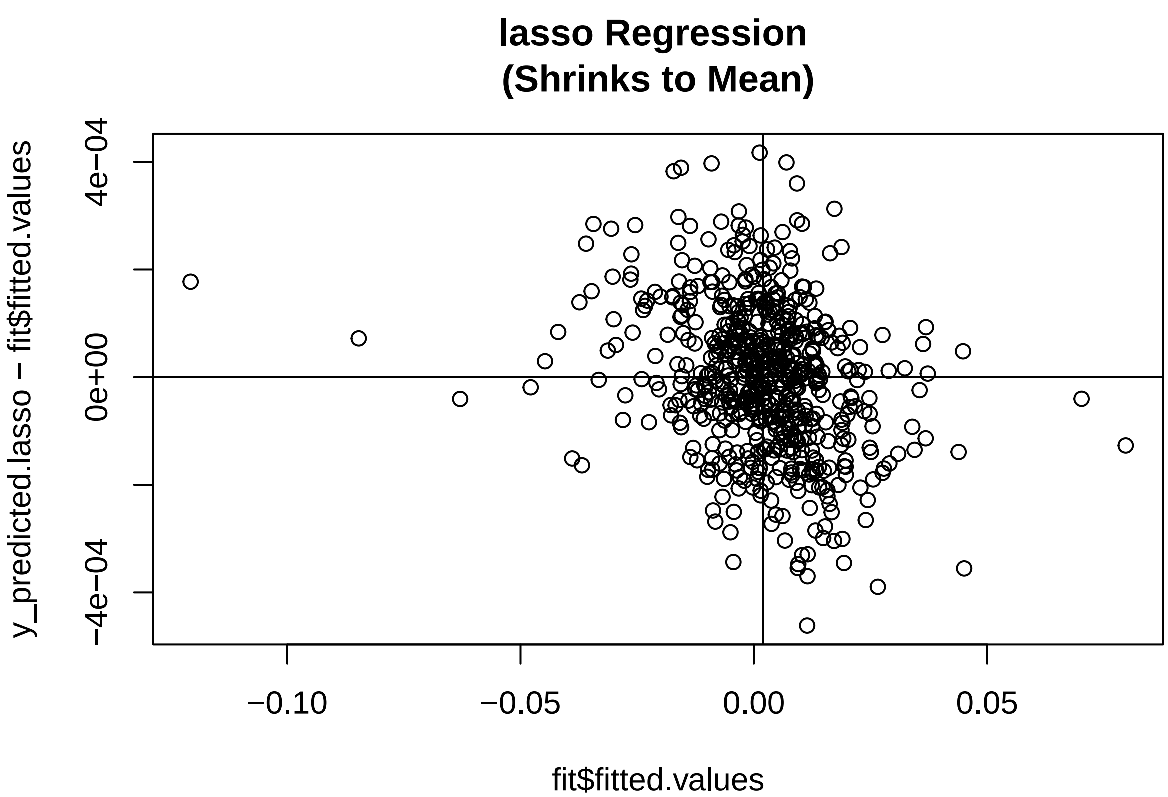

```
cor(y_predicted.lasso,y)^2
```

```
##           [,1]
## s1 0.6417391
```

```
# 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)
```




```
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))
```

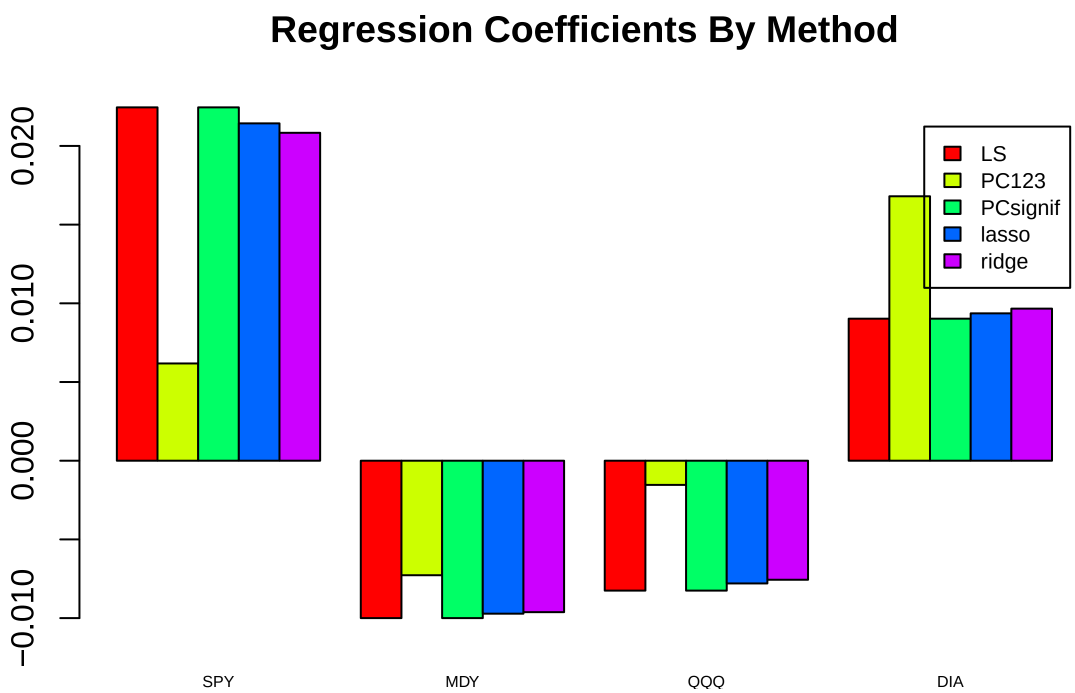


```r
## 8.7 Compare coefficients from fits including LASSO and Ridge ----
```

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

```
##                LS         PC123     PCsignif
## SPY  0.022450726  0.006178644  0.022450726
## MDY -0.010000091 -0.007275983 -0.010000091
## QQQ -0.008254136 -0.001537180 -0.008254136
## DIA  0.009023990  0.016801960  0.009023990
```

```r
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.