Final_Romera_Guillermo

1. You roll five six-sided dice. Write a script in R to calculate the probability of getting between 15 and 20 (inclusive) as the total amount of your roll (ie, the sum when you add up what is showing on all five dice). Exact solutions are preferable but approximate solutions are OK as long as they are precise.

```
n=10000
s<-0;
for (i in 1:n)
{
   d1<-sample(1:6, 1)
   d2<-sample(1:6, 1)
   d3<-sample(1:6, 1)
   d4<-sample(1:6, 1)
   d5<-sample(1:6, 1)

   s[i]<-d1+d2+d3+d4+d5;
}
prob<-sum((s>=15)&(s<=20))/length(s)
prob
```

```
## [1] 0.5544
```

The estimated solution varies between 0.54 +- 0.56, being the exact solution 0.557, I hope my result is precise enough. I understand if I roll the dices more and more the solution will get more accurate, but computational time allows me only to raise the rolls up to 10000.

2. Create a simulated dataset of 100 observations, y =0.1+2×x+ϵ, where independent variable x and error term ϵ are random normal variables with mean $\mu = 0$ and standard deviation $\sigma = 1$, i.e. both x and ϵ are drawn from standard normal distribution: N(μ =0, σ^2 =1). Remember that when creating simulated data with, say, 100 observations, you need to use rnorm(100) for ϵ, not rnorm(1), to ensure that each observation gets a different error.

```
set.seed(1)
x<-rnorm(n=100,mean=0,sd=1)
e<-rnorm(n=100,mean=0,sd=1)
y<-0.1+(2*x)+e
```

(a) Perform a t-test for whether the mean of Y equals the mean of X using R.

```
alpha<-0.05
df1<-data.frame(x=as.numeric(x),y=as.numeric(y))
test1<- t.test(df1$x-df1$y, conf.level = 1-alpha)
test1
```

```
##
##  One Sample t-test
##
## data:  df1$x - df1$y
```

```
## t = -1.3035, df = 99, p-value = 0.1954
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  -0.43150226  0.08934368
## sample estimates:
##  mean of x
## -0.1710793
```

We fail to reject the Null Hypothesis since the P-value is higher than it's alpha value $\alpha = 0.05$

(b) Now perform this test by hand using just the first 5 observations. Please write out all your steps in latex.

$$mean_x = \frac{X1 + X2 + X3 + X4 + X5}{5}$$

$$mean_x = \frac{0.646349423}{5}$$

$$mean_x = 0.1292698846$$

```
n<-5
x1<- x1<-x[1:n]
mean1<-mean(x1)
mean1
```

```
## [1] 0.1292699
```

```
x[1]
```

```
## [1] -0.6264538
```

```
x1 <- ( x[1] - mean1 ) ^ 2
x2 <- ( x[2]- mean1 ) ^ 2
x3 <- ( x[3]- mean1 ) ^ 2
x4 <- (x[4] - mean1 ) ^ 2
x5 <- (x[5] - mean1 ) ^ 2
sx2 <- ( x1 + x2 + x3 + x4 + x5  ) / 4
sx1 <- sqrt ( sx2 )
sx1
```

```
## [1] 0.9610394
```

$$sd^2 = \frac{1}{n-1}\sum_{i}^{n}(x_1 - \overline{x})^2$$

$$sd^2 = \frac{1}{5-1}\sum_{i}^{n}(x_1 - 0.1292699)^2$$

$$sd = 0.9610394$$

```
Se_x <- ( sx1 ) / ( sqrt ( n ) )
Se_x
```

```
## [1] 0.4297899
```

$$se_x = \frac{sd}{\sqrt{n}}$$

$$se_x = \frac{0.9610394}{\sqrt{5}}$$

$$se_x = 0.4297899$$

$$mean_y = \frac{Y1 + Y2 + Y3 + Y4 + Y5}{5}$$

$$mean_y = \frac{-0.1930294}{5}$$

$$mean_y = -0.03860587$$

```
n<-5
y1<-y[1:n]
mean2<-mean(y1)
mean2
```

```
## [1] -0.03860587
```

```
y[1]+y[2]+y[3]+y[4]+y[5]
```

```
## [1] -0.1930294
```

```
y1 <- ( y[1] - mean2 ) ^ 2
y2 <- ( y[2]- mean2 ) ^ 2
y3 <- ( y[3]- mean2 ) ^ 2
y4 <- (y[4] - mean2 ) ^ 2
y5 <- (y[5] - mean2 ) ^ 2
sy3 <- ( y1 + y2 + y3 + y4 + y5  ) / 4
sy2 <- sqrt ( sy3 )
sy2
```

```
## [1] 2.316324
```

$$sd^2 = \frac{1}{n-1}\sum_{i}^{n}(x_1 - \overline{x})^2$$

$$sd^2 = \frac{1}{5-1}\sum_{i}^{n}(x_1 - -0.1930294)^2$$

$$sd = 2.316324$$

```
Se_y <- ( sy2 ) / ( sqrt ( n ) )
Se_y
```

```
## [1] 1.035892
```

$$se_y = \frac{sd}{\sqrt{n}}$$

$$se_y = \frac{2.316324}{\sqrt{5}}$$

$$se_y = 1.035892$$

```
Sedif <- sqrt (Se_x^2 + Se_y^2)
Sedif
```

```
## [1] 1.121513
```

$$Se_{diff} = \sqrt{Se_x^2 + Se_y^2}$$

$$Se_{diff} = \sqrt{1.257791 * 2}$$

$$Se_{diff} = 1.121513$$

$$Tstatistic = \frac{mean_x - mean_y}{Se_{diff}}$$

$$Tstatistic = \frac{0.129269 - (-0.038605)}{1.121513}$$

```
Tstat <- (( mean1 - mean2) / 1.121513)
Tstat
```

```
## [1] 0.1496869
```

$$Degree\ of\ Freedom = \frac{se_{ab}^2}{se_a^4/(n_a - 1) + se_b^4/(n_b - 1)}$$

```
df<- ((Sedif)^2)/((Se_x)^4/(n-1)+(Se_y)^4/(n-1))
df
```

```
## [1] 4.243545
```

```
thresholds<-qt(0.025, 0.975,df=df)
thresholds
```

```
## [1] -1.189986
```

We still fail to reject the Null Hypothesis since thee T statistic falls well within threshold.

(c) Using R, test whether the mean of Y is significantly different from 0.

```
 t.test(y, alternative="two.sided", mu=0)
```

```
##
##  One Sample t-test
##
## data:  y
## t = 1.3758, df = 99, p-value = 0.172
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  -0.1238183  0.6837516
## sample estimates:
## mean of x
## 0.2799667
```

Again we fail to reject the Null hypothesis since the p-value is still higher than it's alpha value (0.05)

(d) Again using the first five obsevations, test by hand whether the mean of Y is different from 0.

$$mean_y = \frac{Y1 + Y2 + Y3 + Y4 + Y5}{5}$$
$$mean_y = \frac{-0.1930294}{5}$$
$$mean_y = -0.03860587$$

```
n=5
y1<-y[1:n]
mean3<- mean(y1)
```

$$sd^2 = \frac{1}{n-1}\sum_{i}^{n}(x_1 - \overline{x})^2$$

$$sd^2 = \frac{1}{5-1}\sum_{i}^{n}(x_1 - -0.1930294)^2$$

$$sd = 2.316324$$

```
y1 <- ( y[1] - mean3 ) ^ 2
y2 <- ( y[2]- mean3 ) ^ 2
y3 <- ( y[3]- mean3 ) ^ 2
y4 <- (y[4] - mean3 ) ^ 2
y5 <- (y[5] - mean3 ) ^ 2
sy3 <- ( y1 + y2 + y3 + y4 + y5  ) / 4
sy2 <- sqrt ( sy3 )
sy2
```

```
## [1] 2.316324
```

```
Se_y <- ( sy2 ) / ( sqrt ( n ) )
Se_y
```

```
## [1] 1.035892
```

$$se_y = \frac{sd}{\sqrt{n}}$$
$$se_y = \frac{2.316324}{\sqrt{5}}$$
$$se_y = 1.035892$$

$$Tstatistic = \frac{mean_y}{Se_y}$$

```
Tstatistics<-mean3/Se_y
Tstatistics
```

```
## [1] -0.03726825
```

```
thresholds<-qt(0.025,0.975,df=n-1)
thresholds
```

```
## [1] -1.205035
```

Still we can't reject that the Null hypothesis is rejected due to the T-statistic being within the threshold.

(e) Assuming the mean and sd of Y that you calculate from the first five observations would not change, what is the minimum total number of observations you would need to be able to conclude that the mean of Y is different from 0 at the p = 0.01 confidence level?

(f) Verify (d) (approximately) by increasing the simulated data to the n you calculated in (e) that would be necessary. If the test of Y =0 is still not significant, explain why. (Go back to using the original 100-observation dataset for g and h.)

(g) Create a categorical (factor) variable c, where c =1 if x < -1, c =3 if x > 1, and c =2 otherwise. Use R to perform an F test for whether the mean of y differs across these three groups.

```
c=0
for (i in 1:length(x))
{
  if(x[i]<(-1))
  {c[i]=1}
  if(x[i]>1)
  {c[i]=3}
  if((x[i]>=-1)&(x[i]<=1))
  {c[i]=2}
}
c<-as.numeric(c)
df1<-data.frame(c=as.numeric(c),x=as.numeric(x),y=as.numeric(y))
anova = aov(y~c,data=df1)
anova
```

```
## Call:
##    aov(formula = y ~ c, data = df1)
##
## Terms:
##                         c Residuals
## Sum of Squares  214.2642  195.7105
## Deg. of Freedom        1        98
##
## Residual standard error: 1.413169
## Estimated effects may be unbalanced
```

```
summary(anova)
```

```
##             Df Sum Sq Mean Sq F value Pr(>F)
## c            1  214.3   214.3   107.3 <2e-16 ***
## Residuals   98  195.7     2.0
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Based on te ANOVA test it does seems like the means of these three groups are indeed different.

(h) Using the first three observations for each group, calculate the same F test by hand

```
n1<-3
n2<-3
n3<-3
N<-9
G<-3
y1<-head(y[c==1],n1)
y2<-head(y[c==2],n2)
y3<-head(y[c==3],n3)

y <- c(y1, y2, y3)
ytotal <- mean(y)
ymean1 <- mean(y1)
ymean2 <- mean(y2)
ymean3 <- mean(y3)

sdy1 <- sd(y1)
sdy2 <- sd(y2)
sdy3 <- sd(y3)
```

$$BV = \frac{n_1(\overline{y_1} - \overline{y})^2 + ... + (n_g(\overline{y_g} - \overline{y})^2}{G-1}$$

```
bv <- (n1*(ymean1-ytotal)^2 + n2*(ymean2-ytotal)^2 + n3*(ymean3-ytotal)^2)/(G-1)
```

$$WV = \frac{(\overline{n_1} - \overline{1})S_1^2 + ... + (n_g - 1)S_G^2}{N-G}$$

```
wv <- ((n1-1)*sdy1^2 + (n2-1)*sdy2^2 + (n3-1)*sdy3^2)/(N-G)

ftst <- (bv)/(wv)
ftst
```

```
## [1] 25.20071
```

```
dgf1<- G-1
dgf2<- N-G

threshold<-qf(0.95,dgf1,dgf2,lower.tail = F)
threshold
```

```
## [1] 0.0517343
```

As the Threshold and the F statistic shows at least one of the means within the group has a different mean.

3. Now generate y =0.1+0.2 $*$ x - 0.5 $*$ x2 +ϵ with 100 observations, with x and ϵ drawn from N(0,1).

```
set.seed(1)
x <- rnorm(100,0,1)
```

```
y <- 0.1 + 0.2*x -0.5*x^2+ rnorm(100,0,1)
dat <- data.frame(x=x,y=y)
```

(a) Regress y on x and x2 and report the results. If x or x2 are not statistically significant, suggest why.

```
mod <- lm(y ~ x + I(x^2))
signif(coef(summary(mod)),3)
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    0.157     0.1180    1.33 1.86e-01
## x              0.217     0.1080    2.01 4.71e-02
## I(x^2)        -0.619     0.0848   -7.30 7.93e-11
```

Well based on previous experience from our homework the issue might be that the sample size is not as big as one would like making the results a little bit unpredictable.

(b) Based on the known coefficients that we used to create y, how much y changes when we change x from 1 to 2?

```
y1<- 0.1 + 0.2*1 - 0.5*1 ^ 2
y2<- 0.1 + 0.2*2 - 0.5*2 ^ 2
y1 - y2
```

```
## [1] 1.3
```

(c) Based on the coefficients estimated from 3(a), how much y changes when you change x from -0.5 to -0.7?

```
y1<- sum (coef (mod) * c( 1, -0.5, -0.5 ^ 2))
y2<- sum (coef (mod) * c( 1, -0.7, -0.7 ^ 2))
y2 - y1
```

```
## [1] 0.1051089
```

4. Now generate x2 as a random normal variable with a mean of -1 and a sd of 1. Create a new dataset where y =0.1+0.2 $*$x - 0.5 $*$ x $*$ x2 + ϵ.

(a) Based on the known coefficients, what is the effect of increasing x2 from 0 to 1 with x held at its mean?

```
set.seed(1)
n<-100
x <- rnorm(100,0,1)
x2<- rnorm(100,-1,1)
e<-rnorm(n=100,mean=0,sd=1)
y <-0.1 + 0.2*x  - 0.5* x * x2 +e
df<-data.frame(x,x2,y)


y1 <- 0.1 + 0.2* mean(x) - 0.5* mean(x) * 0 + e
```

```r
y2 <- 0.1 + 0.2* mean(x)  - 0.5* mean(x) * 1 + e

eff<-  abs(mean(y2) - mean(y1))
eff
```

```
## [1] 0.05444368
```

(b) Regress y on x, x2, and their interaction. Based on the regression-estimated coefficients, what is the effect on y of shifting x from -0.5 to -0.7 with x2 held at 1?

```r
reg1<- lm(y~x + x2 + x * x2)
summary(reg1)
```

```
##
## Call:
## lm(formula = y ~ x + x2 + x * x2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.92554 -0.43139  0.00249  0.65651  2.60188
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.10285    0.15470   0.665    0.508
## x           -0.07321    0.21598  -0.339    0.735
## x2          -0.02822    0.10970  -0.257    0.798
## x:x2        -0.73968    0.14847  -4.982 2.78e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.035 on 96 degrees of freedom
## Multiple R-squared:  0.4476, Adjusted R-squared:  0.4304
## F-statistic: 25.93 on 3 and 96 DF,  p-value: 2.262e-12
```

```r
y1<- sum (coef (reg1)* c(1, -0.5, 1, -0.5 ))
y2<- sum (coef (reg1)* c(1, -0.7, 1, -0.7 ))

y2 - y1
```

```
## [1] 0.1625782
```

(c) Regress the current y on x alone. Using the R2 from this regression and the R2 from (b), perform by hand an F test of the complete model (4b) against the reduced, bivariate model. What does this test tell you?

```r
reg2<-lm(y~x,data = df)
summary(reg2)
```

```
##
## Call:
## lm(formula = y ~ x, data = df)
##
## Residuals:
```

```
##     Min      1Q  Median      3Q     Max
## -2.9227 -0.7076  0.0501  0.6996  3.3161
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.1174     0.1162   1.011    0.315
## x             0.8352     0.1291   6.470 3.87e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.154 on 98 degrees of freedom
## Multiple R-squared:  0.2993, Adjusted R-squared:  0.2922
## F-statistic: 41.86 on 1 and 98 DF,  p-value: 3.873e-09
```

```
n <- length(x)
r2reg1<- summary(reg1)$r.squared
r2reg2<- summary(reg2)$r.squared
```

$$TSS = \sum_i (y_i - \overline{y})^2$$

```
TSS<- sum((y-mean(y))^2)
```

$$R^2(TSS) = TSS - SSE$$
$$R^2 = 1 - \frac{SSE}{TSS}$$
$$-R^2 + 1 = \frac{SSE}{TSS}$$
$$SSE = TSS * (1 - R^2)$$

```
SSEreg1<- TSS*(1-r2reg1)
SSEreg2<- TSS*(1-r2reg2)

#### Degrees of Freedom ####
dfreg1<- n - 4
dfreg2<- n - 2
```

$$F = \frac{SSE_1 - SSE_2/(df_1 - df_2)}{SSE_2/df_2}$$

```
#### F-test ####

ftest<- ((SSEreg2 - SSEreg1) / (dfreg2 - dfreg1)) / (SSEreg1/dfreg1)
ftest
```

```
## [1] 12.88734
```

```
#### P-Value ####

p<- 1 - pf(ftest, df1=dfreg2-dfreg1, df2=dfreg1)
p
```

```
## [1] 1.102254e-05
```

```r
anova(reg2, reg1)
```

```
## Analysis of Variance Table
##
## Model 1: y ~ x
## Model 2: y ~ x + x2 + x * x2
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1     98 130.42
## 2     96 102.81  2    27.604 12.887 1.102e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

It seems as if the F-test renders better result with the complete model rather than with the reduced model alone.

5. Create a dataset: $y = 0.1 + 0.2 * x\text{-}0.5 * x * x2 + \epsilon$, with $n = 100$ observations; where x and ϵ are drawn from N(0,1) and x2 from N(-1,1). Generate a binary variable y2 which is 1 if $y > 0$ and 0 otherwise.

(a) Perform a logistic regression of y2 on x, x2, and their interaction, and interpret the results.

```r
n<- 100
set.seed(1)
x<- rnorm(n, mean=0, sd=1)
x2<- rnorm(n, mean=-1, sd=1)
e<- rnorm(n, mean=0, sd=1)

y<- 0.1 + 0.2*x - 0.5*x*x2 + e
y2<- as.factor(ifelse(y>0, 1, 0))

mod<- glm(y2 ~ x + x2 + x*x2, family=binomial)

summary(mod)
```

```
##
## Call:
## glm(formula = y2 ~ x + x2 + x * x2, family = binomial)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9230  -1.0427   0.3383   0.9791   1.7524
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.01077    0.31023   0.035   0.9723
## x           -0.03099    0.48461  -0.064   0.9490
## x2          -0.17171    0.24183  -0.710   0.4777
## x:x2        -1.03817    0.42302  -2.454   0.0141 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
```

```
##
##     Null deviance: 137.63  on 99  degrees of freedom
## Residual deviance: 113.96  on 96  degrees of freedom
## AIC: 121.96
##
## Number of Fisher Scoring iterations: 5
```

Seeing the results, and running it multiple times, although I have decided to Set.seed to make my answer more consistent, seems like the only significant coefficient is the $X * X2$

(b) What is the effect of increasing x2 from 0 to 1 with x held at its mean on the probability that y2 is 1?

```
invlogit<- function(x) exp(x) / (1+exp(x))

invlogit(sum(coef(mod)*c(1, mean(x), 1, 1*mean(x))))-
  invlogit(sum(coef(mod)*c(1, mean(x), 0, 0*mean(x))))
```

```
## [1] -0.07074798
```

6. Generate a dataset with 300 observations and three variables: f, x1, and x2. f should be a factor with three levels, where level 1 corresponds to observations 1-100, level 2 to 101-200, and level 3 to 201-300. Create x1 and x2 such that the first 100 observations have a mean of 1 for x1 and 1 for x2, each with a standard deviation of 2; the second 100 observations have a mean of 0 for x1 and 1 for x2, both with a standard deviation of 1; and the third 100 observations have a mean of 1 for x1 and 0 for x2, both with a standard deviation of 0.5. NOTE: play a little - try smaller and larger values for number of observations and standard deviations.

(a) Using the k-means algorithm, peform a cluster analysis of these data using a k of 3 (use only x1 and x2 in your calculations; use f only to verify your results). Comparing your clusters with f, how many datapoints are correctly classified into the correct cluster? How similar are the centroids from your analysis to the true centers?

```
library(flexclust)
```

```
## Warning: package 'flexclust' was built under R version 3.3.3
```

```
## Loading required package: grid
```

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

```
## Loading required package: modeltools
```

```
## Loading required package: stats4
```

```
library(cluster)
set.seed(1)
x1<-c(rnorm(100,1,2),rnorm(100,0,1),rnorm(100,1,0.5))
x2<-c(rnorm(100,1,2),rnorm(100,1,1),rnorm(100,0,0.5))
f<-as.factor(c(rep(1,100),rep(2,100),rep(3,100)))
df<-data.frame(x1,x2,f)
truec <- data.frame(x1=c(1,0,1), x2=c(1,1,0), row.names=c("x1","x2","x3"))
plane1 <- NULL
k<-3
```

```
km <- kmeans(df[c("x1","x2")], k, nstart=9)
a2 <- dist2(truec, km$centers)

for (i in 1:nrow(a2))
  {
  plane1 <- c(plane1, names(which(a2[i,] == min(a2[i,]))))
}

plane1 <- as.factor(sub("x","", plane1))
plane2 <- match(km$cluster, plane1)
table1 <- table(plane2, df$f)
class1 <- apply(table1, 2, sum)
```

```
class1
```

```
##   1   2   3
##  59  76 100
```

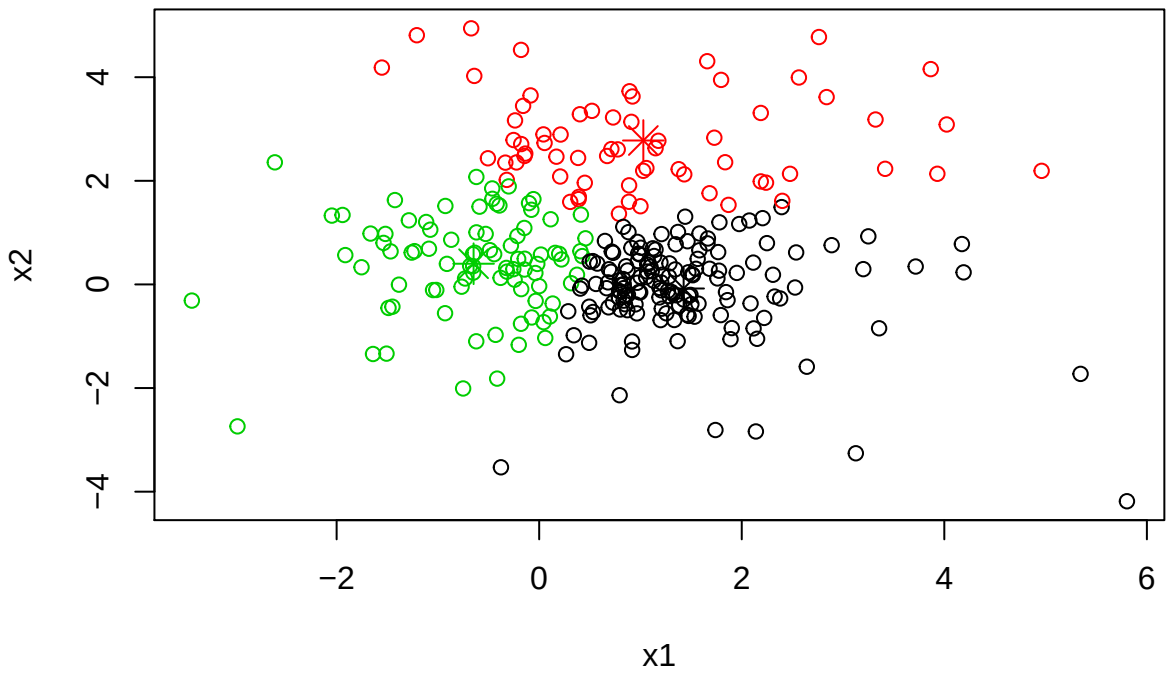
This table show how many of those points are correctly clasified in the correct cluster. In total we have 139 points correctly clasified, not counting the f as those have little to no standard deviation thus classifing all points correctly (most of the time)

```
a2
```

```
##            1        2         3
## x1 1.1596320 1.778037 1.7537237
## x2 1.7880150 2.053790 0.8832164
## x3 0.4332959 2.777955 1.6954948
```

This table shows the closest distance of the points generated by Kmeans and the true centers generate for the problem. For us to know the closest distance we need to chose the lower values in the table which would be x1 1.1596320, x2 0.8832164 and x3 0.4332959

```
plot(df[c("x1", "x2")], col=km$cluster)
points(km$centers[,c("x1", "x2")], col=1:3, pch=8, cex=2)
```



This plot shows the clusters and their centers.

(b) Perform a principal component analysis (PCA) of this data using your preferred function. Using the scree plot, how many principal components do you think you should include? Speculate about how these results relate to those you got with the cluster analysis.

```
pca<- prcomp(df[,1:2])

pcaA1<- pca$center+ pca$rotation[, 1]*pca$sdev[1]
pcaA2<- pca$center+ pca$rotation[,2]*pca$sdev[1]
sort(pcaA1)
```

```
##        x1        x2
## 0.3878057 2.0898233
```

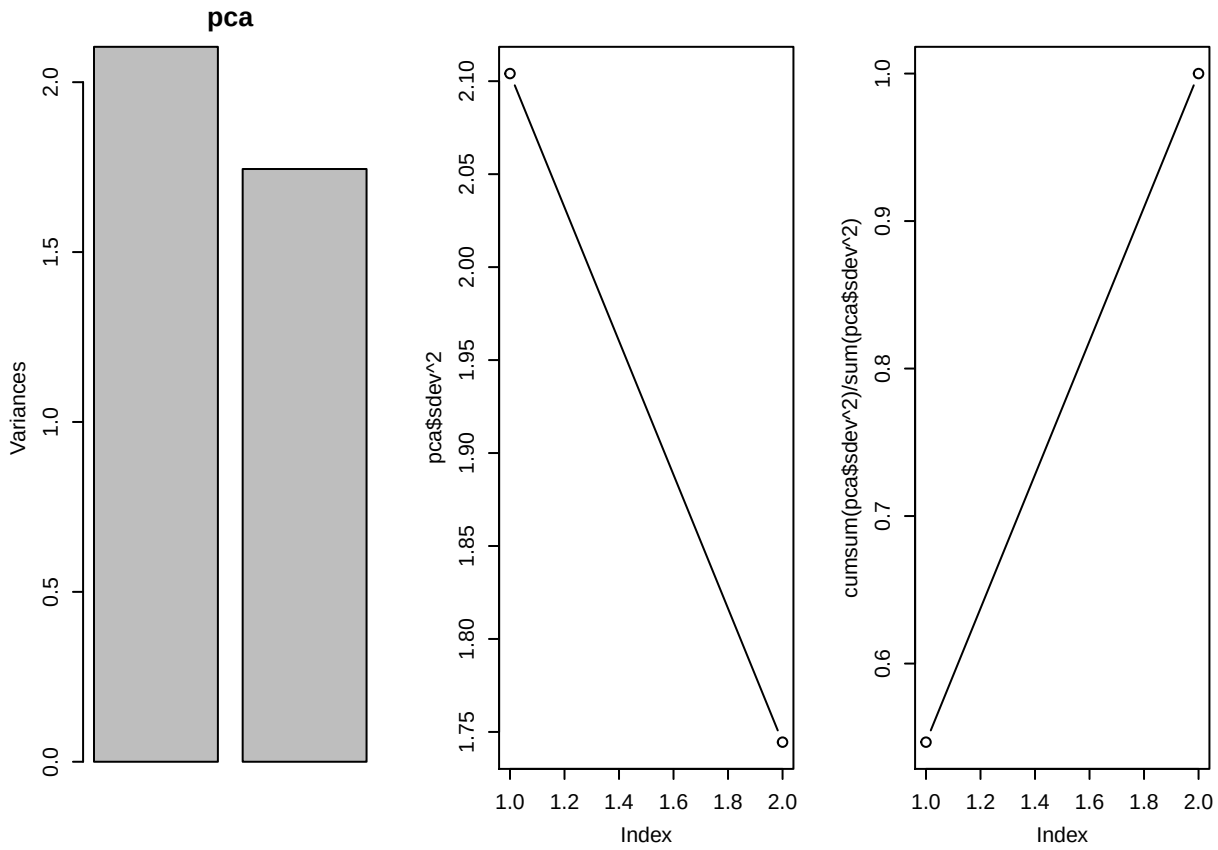
```
sort(pcaA2)
```

```
##         x1         x2
## -0.6776189  0.3368078
```

```
par(mfrow=c(1, 3), mar = c(4, 4, 3, 2), mgp = c(2.6, 1, 0), mex = .8)
screeplot(pca)

plot(pca$sdev^2, type = "b")

plot(cumsum(pca$sdev^2)/sum(pca$sdev^2),type="b")
```



Well, looking at the graph it seems like there's no "elbow" in here to make any reduction, I don't really think this would admit any reduction of any kind and all factors should be retained.

7. Generate a dataset of 200 observations, this time with 90 independent variables, each drawn from N(0,1). Create y such that: y =2x1 +...+2x30-x31-...-x60 +0∗x61 +...+0∗x90 +ϵ where ϵ is drawn from N(μ =0,σ =25). (i.e., the first 30 x's have a coefficient of 2; the next 30 have a coefficient of -1; and the last 30 have a coefficient of 0.)

```
set.seed(1)
obs<- 200
vrbles<- 90
xmat<- matrix( rnorm( obs * vrbles, mean=0, sd=1), ncol=vrbles)
coeff<- c(0, rep(c(2,-1,0), each=30))
e<- rnorm (obs, sd= 5)
y<- cbind(1, xmat) %*% coeff + e
```

(a) Perform an elastic net regression of y on all the x variables using just the first 100 observations. Use 10-fold cross-validation to find the best value of λ and approximately the best value of α.

```
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 3.3.3
```

```
## Loading required package: Matrix
```

```
## Loading required package: foreach
```

```
## Loaded glmnet 2.0-5
```

```
set.seed(1)
train <- 1:100
test <- setdiff(1:obs, train)
inx  <- xmat[train,]
iny <- y[train]
outx <- xmat[test,]
outy <- y[test]


lambdalevels <- 10^seq(4,-3,length=100)

par(mfrow=c(2,2))

cv.lasso.mod_1=cv.glmnet(inx,iny,alpha=0,lambda=lambdalevels)
bestlambda <- cv.lasso.mod_1$lambda.min
bestlambda
```

```
## [1] 1.097499
```

```
yhat.l <- predict(cv.lasso.mod_1$glmnet.fit, s=cv.lasso.mod_1$lambda.min, newx=inx)
mse.las1 <- sum((iny - yhat.l)^2)/nrow(inx)
mse.las1
```

```
## [1] 4.115227
```

```
cv.lasso.mod_2=cv.glmnet(inx,iny,alpha=0.125,lambda=lambdalevels)
bestlambda <- cv.lasso.mod_2$lambda.min
bestlambda
```

```
## [1] 0.4132012
```

```
yhat.l <- predict(cv.lasso.mod_2$glmnet.fit, s=cv.lasso.mod_2$lambda.min, newx=inx)
mse.las2 <- sum((iny - yhat.l)^2)/nrow(inx)
mse.las2
```

```
## [1] 3.265083
```

```
cv.lasso.mod_3=cv.glmnet(inx,iny,alpha=0.25,lambda=lambdalevels)
bestlambda <- cv.lasso.mod_3$lambda.min
bestlambda
```

```
## [1] 0.1830738
```

```
yhat.l <- predict(cv.lasso.mod_3$glmnet.fit, s=cv.lasso.mod_3$lambda.min, newx=inx)
mse.las3 <- sum((iny - yhat.l)^2)/nrow(inx)
mse.las3
```

```
## [1] 2.384537
```

```
cv.lasso.mod_4=cv.glmnet(inx,iny,alpha=0.40,lambda=lambdalevels)
bestlambda <- cv.lasso.mod_4$lambda.min
bestlambda
```

```
## [1] 0.09545485
```

```
yhat.l <- predict(cv.lasso.mod_4$glmnet.fit, s=cv.lasso.mod_4$lambda.min, newx=inx)
mse.las4 <- sum((iny - yhat.l)^2)/nrow(inx)
```

```
mse.las4
```

```
## [1] 1.835094
```

```
cv.lasso.mod_5=cv.glmnet(inx,iny,alpha=0.60,lambda=lambdalevels)
bestlambda <- cv.lasso.mod_5$lambda.min
bestlambda
```

```
## [1] 0.1321941
```

```
yhat.l <- predict(cv.lasso.mod_5$glmnet.fit, s=cv.lasso.mod_5$lambda.min, newx=inx)
mse.las5 <- sum((iny - yhat.l)^2)/nrow(inx)
mse.las5
```

```
## [1] 3.096559
```

```
cv.lasso.mod_6=cv.glmnet(inx,iny,alpha=0.80,lambda=lambdalevels)
bestlambda <- cv.lasso.mod_6$lambda.min
bestlambda
```

```
## [1] 0.05857021
```

```
yhat.l <- predict(cv.lasso.mod_6$glmnet.fit, s=cv.lasso.mod_6$lambda.min, newx=inx)
mse.las6 <- sum((iny - yhat.l)^2)/nrow(inx)
mse.las6
```

```
## [1] 1.970529
```

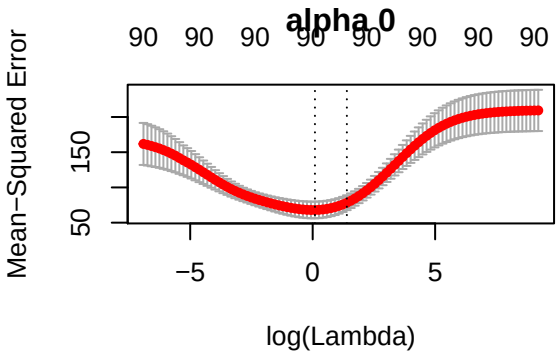
```
cv.lasso.mod_7=cv.glmnet(inx,iny,alpha=1,lambda=lambdalevels)
bestlambda <- cv.lasso.mod_7$lambda.min
bestlambda
```

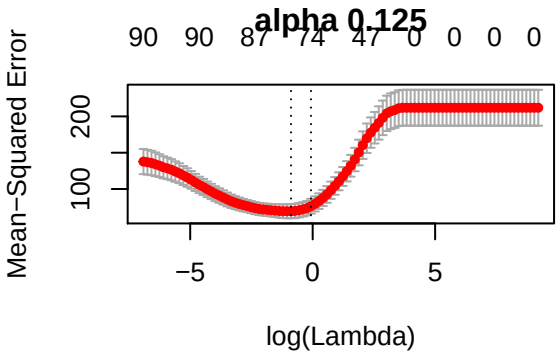
```
## [1] 0.04977024
```

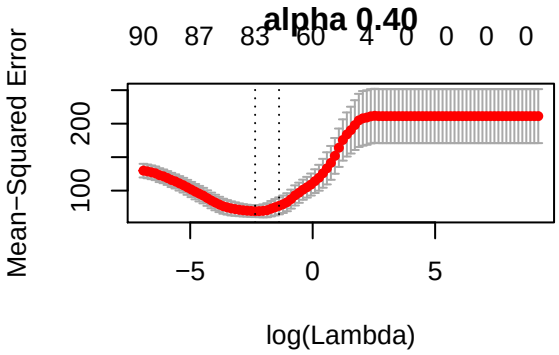
```
yhat.l <- predict(cv.lasso.mod_7$glmnet.fit, s=cv.lasso.mod_7$lambda.min, newx=inx)
mse.las7 <- sum((iny - yhat.l)^2)/nrow(inx)
mse.las7
```

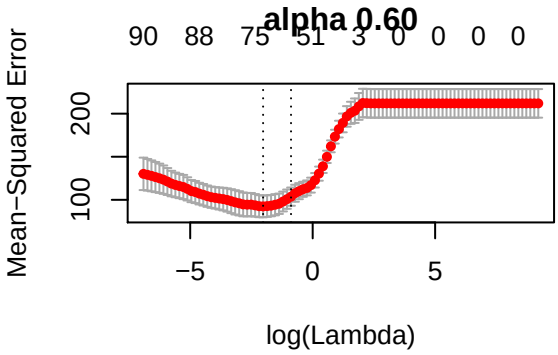
```
## [1] 2.014119
```

```
plot(cv.lasso.mod_1, main="alpha 0")
plot(cv.lasso.mod_2, main="alpha 0.125")
plot(cv.lasso.mod_4, main="alpha 0.40")
plot(cv.lasso.mod_5,main="alpha 0.60")
```

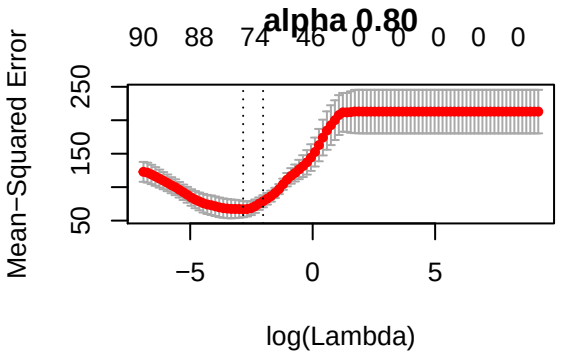


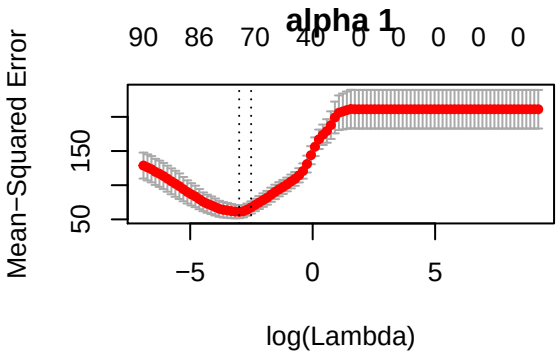







```
plot(cv.lasso.mod_6,main="alpha 0.80")
plot(cv.lasso.mod_7,main="alpha 1")
```

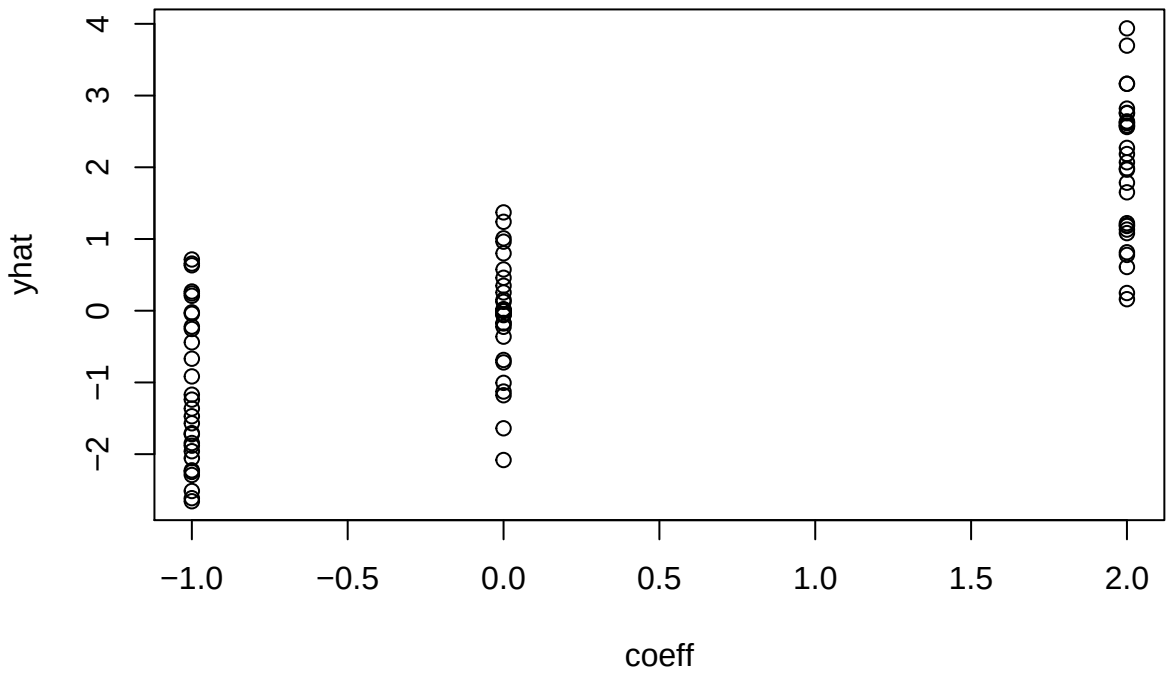




After these results it seems the best value of λ is 0.09545485 with an α of 0.40, yielding a MSE of 1.835094

(b) How accurate are your coefficients from (a)? Summarize your results any way you like, but please don't give us the raw coefficients from 90 variables.

```
yhat <- predict(cv.lasso.mod_4, type="coefficients",s=bestlambda)
plot(coeff,yhat)
```




```
head(predict(cv.lasso.mod_4, type="coefficients",s=bestlambda))
```

```
## 6 x 1 sparse Matrix of class "dgCMatrix"
##                      1
## (Intercept) 0.8002014
## V1          1.7848163
## V2          3.9372061
## V3          2.5797028
## V4          3.1645202
## V5          1.1314035
```

```
tail(predict(cv.lasso.mod_4, type="coefficients",s=bestlambda))
```

```
## 6 x 1 sparse Matrix of class "dgCMatrix"
##               1
## V85 -1.17863205
## V86 -1.00573119
## V87  0.34859494
## V88 -0.06141993
## V89 -0.22452045
## V90  .
```

The results not as accurate as we would like to. We see the first 30 numbers get somewhat close to 2 while the last 30 get scloser to 0. There are some random numbers in between that should not be there, but I guess that might be an side effect of my choice of alpha.

(c) Using the results from (b), predict y for the second 100 observations. How accurate is your prediction?

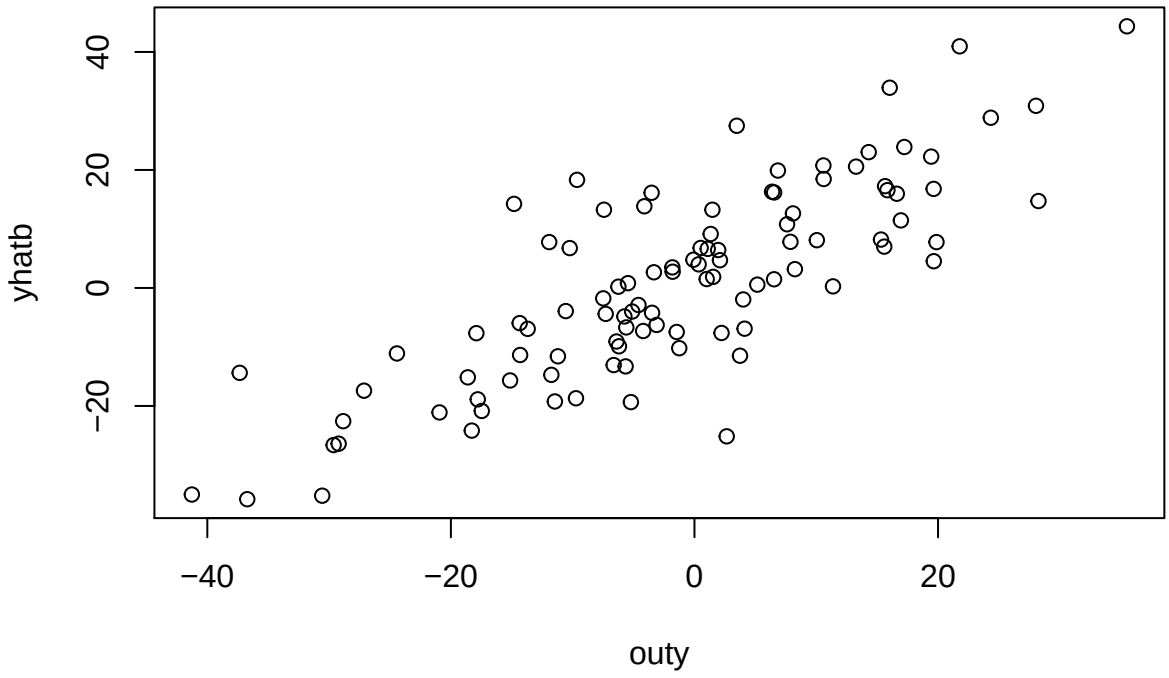
```
yhatb<- predict(cv.lasso.mod_4,  s=bestlambda, newx=outx)
head(yhatb)
```

```
##               1
## [1,]  -7.665306
## [2,]  19.902327
## [3,]  27.492176
## [4,]  -3.917228
## [5,]  -2.887502
## [6,] -10.209208
```

```
mse.b<- mean((outy - yhatb)^2)
mse.b
```

```
## [1] 101.378
```

```
plot(outy,yhatb)
```



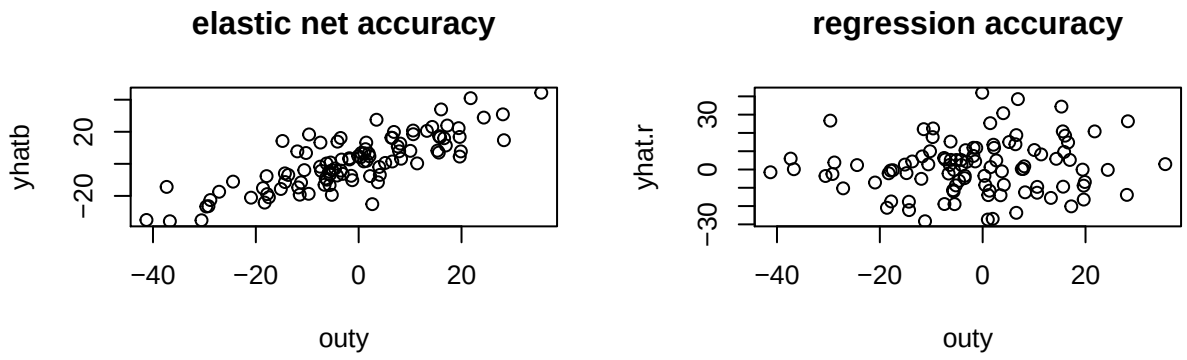
It seems this prediction is failry accurate as thee graph shows, althugh it leave a lot to be desired it could have been worse.

(d) Attempt to compare the predictive accuracy here to the accuracy of a prediction made using regular multiple regression. Explain your results, including if the regular regression failed for any reason.

```
data <- data.frame(cbind(y=iny, inx))
mreg <- lm(y~., data)
yhat.r <- predict(mreg, newx=outx)
mse.reg<- mean((outy - yhat.r)^2)
mse.reg
```

```
## [1] 398.6685
```

```
par(mfrow=c(2,2))
plot(outy,yhatb, main="elastic net accuracy")
plot(outy,yhat.r,main="regression accuracy")
```



As it happened to me in homework 12 I learnt after some research that the elastic net performs better than the regular multiple regression when it comes to prediction. We can also see that the MSE for the multiple regression is higher than for any of the predictions using the elastic net model.

8. Use the data from 7 to generate a new y2 that is 1 if y $>$ 0 and 0 otherwise.

```
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 3.3.3
```

```
y2<- as.factor(ifelse(y>0, 1, 0))
iny<- y2[train]
outy<- y2[test]
```

(a) Using the same process as in 8, estimate an SVM model of y2 on all the x variables for the first 100 variables. Use 10-fold cross-validation to select the best kernel.

```
in1 <- data.frame(y=iny, inx)
cost <- 10^seq(-4,2,1)

tunedL <- tune(svm, y~., data=in1, ranges=list(cost=cost), kernel="linear")
tunedR<- tune(svm, y~., data=in1, ranges=list(cost=cost), kernel="radial")

summary(tunedL)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##   0.1
##
## - best performance: 0.24
##
## - Detailed performance results:
##    cost error dispersion
## 1 1e-04  0.51  0.1100505
## 2 1e-03  0.51  0.1100505
## 3 1e-02  0.38  0.1619328
## 4 1e-01  0.24  0.1577621
## 5 1e+00  0.24  0.1577621
## 6 1e+01  0.24  0.1577621
## 7 1e+02  0.24  0.1577621
```

```
summary(tunedR)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##    10
##
## - best performance: 0.35
##
## - Detailed performance results:
##    cost error dispersion
## 1 1e-04  0.47  0.1059350
## 2 1e-03  0.47  0.1059350
## 3 1e-02  0.47  0.1059350
```

```
## 4 1e-01  0.47  0.1059350
## 5 1e+00  0.36  0.1349897
## 6 1e+01  0.35  0.1269296
## 7 1e+02  0.35  0.1269296
```

It seems like this time the radial Kernel has not performed as well as the lineal kernel having 24% points incorrectly the linea kernel and 35% the radial kernel.

(b) Using the results from (a), predict y2 for the second 100 observations, and report your accuracy.

```
outsample <- data.frame(y=outy, outx)


yhatL <- predict(tunedL$best.model, newdata=outsample)
table(predicted=yhatL,truth=outsample$y)
```

```
##          truth
## predicted  0  1
##         0 34  8
##         1 19 39
```

```
sum(yhatL == outsample$y) / length(outsample$y)
```

```
## [1] 0.73
```

We have an accuracy of 73% for the lineal kernel.

9. BONUS(+10pts) Draw a random chord in an unit circle. What is the probability that this chord has a length greater than$\sqrt{3}$? Hints: (1) The side length of an equilateral triange inscribed in an unit circle is $\sqrt{3}$ (2) There is more than one correct answer.

As a first method select two random points inside the closed disk and draw a chord through them. For providing one correct answer you can earn 5pts. You need to provide at least two distinct correct answers and a graphical illustration of your solution(s) to get a full credit of 10pts.