Final Exam - Intro. to Computational Statistics

Unless otherwise specified, assume all ?? (p-value) thresholds to be 0.05, and all tests to be two-sided if that is an option. All calculations may be done with R or by hand unless otherwise specified. Please show and explain your work as much as possible, using latex for displaying all math. Note that all problems are worth 3 points except problem 1, which is worth 7 points, and problems 7(a), 8(a), and 9(a), which are each worth 5 points. Good luck!

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.

Exact solution:

```
prob<-function(m,sum)
{

  if(m==1)
  {
    if((sum>=0)&(sum<=6))
    {return(1/6)}
    if((sum<0)|(sum>6))
    {return(0)}

  }
  return(1/6*(prob(m-1,sum-6)+prob(m-1,sum-5)+prob(m-1,sum-4)+prob(m-1,sum-3)+prob(m-1,su
m-2)+
                prob(m-1,sum-1)));


}

P=prob(5,15)+prob(5,16)+prob(5,17)+prob(5,18)+prob(5,19)+prob(5,20)
P
```

```
## [1] 0.6265432
```

Numerical Solution:

```
n=10000
s<-0;
for (i in 1:n)
{
   x1<-sample(1:6, 1)
   x2<-sample(1:6, 1)
   x3<-sample(1:6, 1)
   x4<-sample(1:6, 1)
   x5<-sample(1:6, 1)

   s[i]<-x1+x2+x3+x4+x5;
}
P<-sum((s>14)&(s<21))/length(s)
P
```

```
## [1] 0.5532
```

The results are very close and by increasing n, numerical results would converge to the exact solution.

---

2. Create a simulated dataset of 100 observations, where x is a random normal variable with mean 0 and standard deviation 1, and y = 0.1 + 2 ??? x + e, where epsilon is also a random normal error with mean 0 and sd 1. (One reminder: remember that in creating simulated data with, say, 100 observations, you need to use rnorm(100) for epsilon, not rnorm(1), to ensure that each observation gets a different error.)

```
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.

we consider x and y as two independent variables we want to see if they have the same minimum.

```
df<-data.frame(x=as.numeric(x),y=as.numeric(y))
#df<-as.numeric(df)
t.test(df$y,df$x)
```

```
##
##  Welch Two Sample t-test
##
## data:  df$y and df$x
## t = 0.066175, df = 136.23, p-value = 0.9473
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.4785708  0.5117088
## sample estimates:
##   mean of x   mean of y
## -0.04350342 -0.06007238
```

with p-value=0.564 the hypothesis that y and x has the same average is NOT rejected.

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

mean_x=(x1+x2+x3+x4+x5)/5 sd_x=

```
n<-5
x1<-x[1:n]
y1<-y[1:n]
mean(x1)
```

```
## [1] 0.4746663
```

```
sd(x1)
```

```
## [1] 0.8232769
```

```
se_x1<-sd(x1)/sqrt(n)
se_x1
```

```
## [1] 0.3681806
```

```
mean(y1)
```

```
## [1] -0.1065127
```

```
sd(y1)
```

```
## [1] 2.605636
```

```
se_y<-sd(y1)/sqrt(n)
se_y
```

```
## [1] 1.165276
```

```
se_diff<-sqrt(se_x1^2+se_y^2)
se_diff
```

```
## [1] 1.222057
```

```
df_eq<- ((se_diff)^2)/((se_x1)^4/(n-1)+(se_y)^4/(n-1))
T_statistics=((mean(x1)-mean(y1))/se_diff)
T_statistics
```

```
## [1] 0.4755742
```

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

```
## [1] 2.776445
```

since the t_statistics lies withing the thresholds, the Null hypothesis(True Mean(y)=True Mean(x)) is NOT rejected.

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

```
t.test(y,mu=0)
```

```
##
##  One Sample t-test
##
## data:  y
## t = -0.18995, df = 99, p-value = 0.8497
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  -0.4979394  0.4109326
## sample estimates:
##   mean of x
## -0.04350342
```

So according to the p-value mean(y)=0 is Not rejected.

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

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

```
## [1] -0.1065127
```

```
sd(y1)
```

```
## [1] 2.605636
```

```
se_y<-sd(y1)/sqrt(n)
T_statistics<-mean(y1)/se_y
T_statistics
```

```
## [1] -0.09140556
```

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

```
## [1] 2.776445
```

since the t_statistics lies withing the thresholds, the Null hypothesis(True Mean(y)=0) is NOT rejected.

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?

$$T_{statistics} = \frac{\sqrt{n} \times (\bar{x} - \mu)}{sd}$$

$$Thresholds(p_{value} = 0.95) = \pm T_{distribution}(0.975, n-1)$$

By increseing n we increase T_{statistics} to exceed the thresholds to reject the Null-Hypothesis. Knowing that increasing n would change thresholds very slightly, we may approximate n as follows and then check if our assumption works.In fact we need to do itteration over n: n_{old}=5

$$n = (\frac{T_{distribution}(0.995, n_{old}-1) \times sd}{\bar{x} - \mu})^2$$

```
T<-qt(0.995,4)
print(T)
```

```
## [1] 4.604095
```

```
mean(y1)
```

```
## [1] -0.1065127
```

```
sd(y1)
```

```
## [1] 2.605636
```

$$n = (\frac{4.604 \times 1.355}{1.750459})^2 = 12.70$$

So n=13 would be a good choice. Now we substitute df=n-1=12 into the T_distribution.

$$n = (\frac{T_{distribution}(0.995, n-1) \times sd}{\bar{x} - \mu})^2$$

```
T<-qt(0.995,12)
print(T)
```

```
## [1] 3.05454
```

```
mean(y1)
```

```
## [1] -0.1065127
```

```
sd(y1)
```

```
## [1] 2.605636
```

$$n = (\frac{3.05454 \times 1.355476}{1.750459})^2 = 5.6$$

so it did not converge at all. That means even if we increase n to 10^6 still we are unable to reject the Null Hypothesis (True Mean(y)=0).

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.)

as calculated in e there would be no convergance to define n. as we increase the n we are just increasing the random numbers around zero and there would be no specific tendecy or direction of the data to be bossted.

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.

```
datafilename="http://personality-project.org/r/datasets/R.appendix1.data"
data.ex1=read.table(datafilename,header=T)
head(data.ex1)
```

```
##   Dosage Alertness
## 1      a        30
## 2      a        38
## 3      a        35
## 4      a        41
## 5      a        27
## 6      a        24
```

```
aov.ex1 = aov(Alertness~Dosage,data=data.ex1)
summary(aov.ex1)
```

```
##             Df Sum Sq Mean Sq F value  Pr(>F)
## Dosage       2  426.2  213.12   8.789 0.00298 **
## Residuals   15  363.8   24.25
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##############

set.seed(123)
x<-rnorm(n=100,mean=0,sd=1)

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)
df<-data.frame(c=as.numeric(c),x=as.numeric(x))
aov.ex1 = aov(x~c,data=df)
aov.ex1
```

```
## Call:
##    aov(formula = x ~ c, data = df)
##
## Terms:
##                         c Residuals
## Sum of Squares  61.70104  20.78901
## Deg. of Freedom        1        98
##
## Residual standard error: 0.4605787
## Estimated effects may be unbalanced
```

```
summary(aov.ex1)
```

```
##             Df Sum Sq Mean Sq F value Pr(>F)
## c            1  61.70   61.70   290.9 <2e-16 ***
## Residuals   98  20.79    0.21
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

So the TRUE mean for these 3 groups are significantly different.

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

```
c<-as.numeric(c)
x1<-x[c==1]
x2<-x[c==2]
x3<-x[c==3]

n1<-length(x1)
mean1<-mean(x1[1:5])
sd1<-sd(x1[1:5])

n2<-length(x2)
mean2<-mean(x2[1:5])
sd2<-sd(x2[1:5])

n3<-length(x3)
mean3<-mean(x3[1:5])
sd3<-sd(x3[1:5])

N<-length(x)
G=3;
meanx<-mean(x)

avbg<-(n1*(mean1-meanx)^2+n2*(mean2-meanx)^2+n3*(mean3-meanx)^2)/(G-1)
avwg<-((n1-1)*sd1^2+(n2-1)*sd2^2+(n3-1)*sd3^2)/(N-G)
df1<-G-1;
df2<-N-G
F_stat<-avbg/avwg
F_stat
```

```
## [1] 239.5437
```

```
threshold<-qf(p=0.95,df1,df2,lower.tail = F)
threshold
```

```
## [1] 0.05132043
```

As it is seen, the F-statistics which is way more than the thresholds, thus the Null hypothesis is rejected.

---

3. Generate a new 100-observation dataset as before, except now y = 0.1 + 0.2 ??? x + e

a. Regress y on x using R, and report the results.

```
set.seed(1)
x <- rnorm(100,0,1)
y <- 0.1 + 0.2*x + rnorm(100,0,1)
dat <- data.frame(x=x,y=y)
#To estimate our model - ie, to regress Y on X - we simply run:
biv_model <- lm(y~x,data=dat)
summary(biv_model)
```

```
##
## Call:
## lm(formula = y ~ x, data = dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.8768 -0.6138 -0.1395  0.5394  2.3462
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.06231    0.09699   0.642   0.5221
## x            0.19894    0.10773   1.847   0.0678 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9628 on 98 degrees of freedom
## Multiple R-squared:  0.03363,    Adjusted R-squared:  0.02377
## F-statistic:  3.41 on 1 and 98 DF,  p-value: 0.06781
```

b. Discuss the coefficient on x and its standard error, and present the 95% CI.

So the calculated x coefficient is 0.19894 with std.error=0.10773

```
qt(0.975,98)
```

```
## [1] 1.984467
```

B1-se*qt|t|)*
(Intercept) 0.15672 0.11766 1.332 0.1860
x 0.21716 0.10798 2.011 0.0471
I(x^2) -0.61892 0.08477 -7.302 7.93e-11 *** ## — ## Signif. codes: 0 ‘***’ 0.001 ’’ 0.01 ” 0.05 ‘.’ 0.1 ‘’ 1 ## ## Residual standard error: 0.958 on 97 degrees of freedom ## Multiple R-squared: 0.3602, Adjusted R-squared: 0.347 ## F-statistic: 27.31 on 2 and 97 DF, p-value: 3.912e-10 ```

b. Based on the known coefficients that we used to create y, what is the effect on y of increasing x by 1 unit from 1 to 2?

```
x1<-1
x2<-2
x<-x1
y1 = 0.1 + 0.2 * x - 0.5 * x^2
x<-x2
y2 = 0.1 + 0.2 * x - 0.5 * x^2

print(y2-y1)
```

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

c. Based on the coefficients estimated from 4(a), what is the effect on y of changing x from -0.5 to -0.7?

```
beta<-as.numeric(biv_model$coefficients)
x<-(-0.5)
y1<-beta[1]+beta[2]*x+beta[3]*x^2

x<-(-0.7)
y2<-beta[1]+beta[2]*x+beta[3]*x^2
print(y2-y1)
```

```
## [1] -0.1919733
```

5. 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 +20.5 ??? x ??? x2 + e.

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

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

```
a<-mean(x)
b1<-0
b2<-1
y1 <-0.1 + 0.2*a-0.5*a*b1
y2 <-0.1 + 0.2*a-0.5*a*b2
y1
```

```
## [1] 0.1005825
```

```
y2
```

```
## [1] 0.09912623
```

```
print(y2-y1)
```

```
## [1] -0.001456281
```

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?

```
reg1<-lm(y~x+x2+x*x2,data = df)
summary(reg1)
```

```
## Warning in summary.lm(reg1): essentially perfect fit: summary may be
## unreliable
```

```
##
## Call:
## lm(formula = y ~ x + x2 + x * x2, data = df)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -3.124e-16 -4.877e-17 -6.240e-18  4.045e-17  8.898e-16
##
## Coefficients:
##               Estimate Std. Error    t value Pr(>|t|)
## (Intercept)  1.000e-01  2.242e-17  4.460e+15   <2e-16 ***
## x            2.000e-01  1.916e-17  1.044e+16   <2e-16 ***
## x2           3.138e-17  1.771e-17  1.772e+00   0.0795 .
## x:x2        -5.000e-01  1.544e-17 -3.238e+16   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.39e-16 on 96 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 1.055e+33 on 3 and 96 DF,  p-value: < 2.2e-16
```

The coefficients in the regresion model are 100% equal to the true value and consequently, R^2=1

```
beta<-as.numeric(reg1$coefficients)
yp<-beta[1]+beta[2]*x+beta[3]*x1+beta[4]*x*x1
b<-1
a1<-(-0.5)
a2<-(-0.7)
y1<-beta[1]+beta[2]*a1+beta[3]*b+beta[4]*a1*b
y2<-beta[1]+beta[2]*a2+beta[3]*b+beta[4]*a2*b
print(y2-y1)
```

```
## [1] 0.06
```

c. Regress the current y on x alone. Using the R2 from this regression and the R2 from 5(b), perform by hand an F test of the complete model (5b) against the reduced, bivariate model. What does this test tell you?

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

```
##
## Call:
## lm(formula = y ~ x, data = df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.75770 -0.16325 -0.05562  0.07010  2.04874
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.15453    0.04546   3.399 0.000978 ***
## x            0.62961    0.04476  14.066  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4546 on 98 degrees of freedom
## Multiple R-squared:  0.6688, Adjusted R-squared:  0.6654
## F-statistic: 197.9 on 1 and 98 DF,  p-value: < 2.2e-16
```

```
r2_r<-0.6688        # r squared reduced

r2_c<-1     # r squared complete regression
df1<-2   #number of additional variables
df2<-(100-3-1)
F_stat<- ((r2_c^2-r2_r^2)/df1)/((1-r2_c^2)/df2)
F_stat
```

```
## [1] Inf
```

```
pf(F_stat,2,(100-6-1),lower.tail=F)
```

```
## [1] 0
```

This very small p-value tells us that the complete model is infact highly boost the model and is much better.

6. Generate a new variable y2 using the data from (5) 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.

```
y1<-ifelse(y>0,1,0)
df<-cbind(df,y1)
reg1<-glm(y1~x+x2+x*x2,data=df,family="binomial")
```

```
## Warning: glm.fit: algorithm did not converge
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(reg1)
```

```
##
## Call:
## glm(formula = y1 ~ x + x2 + x * x2, family = "binomial", data = df)
##
## Deviance Residuals:
##        Min          1Q      Median          3Q         Max
## -6.157e-04  -2.000e-08   2.000e-08   2.000e-08   6.568e-04
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)    277.51   32267.09   0.009    0.993
## x              618.78   47587.20   0.013    0.990
## x2              13.52   26399.55   0.001    1.000
## x:x2         -1259.15  115000.31  -0.011    0.991
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1.2949e+02  on 99  degrees of freedom
## Residual deviance: 8.2017e-07  on 96  degrees of freedom
## AIC: 8
##
## Number of Fisher Scoring iterations: 25
```

Unfortunately the result is not converging.

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?

---

7. 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.

```
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))
y<-as.factor(c(rep(1,100),rep(2,100),rep(3,100)))
df<-data.frame(x1,x2,y)
head(df)
```

```
##           x1         x2 y
## 1  1.0563435 -1.8925749 1
## 2  0.2865932  1.6317115 1
## 3  2.7052528  0.3145050 1
## 4  2.0267305 -2.8627062 1
## 5  3.0364060  1.4856420 1
## 6 -1.0429582  0.2744641 1
```

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

```
set.seed(1)
cat <- as.factor(floor(runif(300,1,4)))
df <- cbind(df,cat)
for(i in 1:100)  #100 iteration
{
  # 2(a): get centroids of two groups
  centroids <- aggregate(df[,1:2],by=list(cat=df$cat),FUN=mean)

  # 2(b): calculate distances of each point to centroid 1 (d1) and centroid 2 (d2)
  d1 <- sqrt( (df[,1]-centroids[1,2])^2 + (df[,2]-centroids[1,3])^2 )
  d2 <- sqrt(  (df[,1]-centroids[2,2])^2 + (df[,2]-centroids[2,3])^2 )
  d3 <- sqrt(  (df[,1]-centroids[3,2])^2 + (df[,2]-centroids[3,3])^2 )
  # then reassign the category variable depending on which centroid is closer
  for (j in 1:length(df[,1]))
  {
          if ((d1[j]<d2[j])&(d1[j]<d3[j]))
        {
          df$cat[j] <- 1
        }
        if ((d2[j]<d1[j])&(d2[j]<d3[j]))
        {
          df$cat[j]<- 2
        }
         if ((d3[j]<d1[j])&(d3[j]<d2[j]))
        {
          df$cat[j]<- 3
        }
  }
}
sum(df$y==df$cat)
```

```
## [1] 44
```

```
print(centroids)
```

```
##   cat         x1         x2
## 1   1  1.1701052 -0.1185835
## 2   2  2.0284513  2.8271152
## 3   3 -0.7244402  1.3719655
```

```
true_centrid<-rbind(c(1,1),c(0,1),c(1,0))
true_centrid
```

```
##      [,1] [,2]
## [1,]    1    1
## [2,]    0    1
## [3,]    1    0
```

so 176 out of 300 observations are classified correctly. So the centroids calculated by the algorithm is not very accuarate but some how close to the true centroids. ——————————-

8. Generate a dataset of 200 observations, this time with 90 independent variables, each of mean 0 and sd
9. Create y such that: y = 2x1 + … + 2x30 ??? x31 ??? … ??? x60 + 0 * x61 + … + 0 * x90 + e where e is a random normal variable with mean 0 and sd 10. (Ie, 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.)

```
df<-rnorm(200,0,1)
y<-0;
for (i in 1:89)
{

    df<-cbind(df,rnorm(200,0,1))
}
for (i in 1:200)
{

    y[i]<-2*sum(df[i,1:30])-1*sum(df[i,31:60])
}

df<-as.data.frame(cbind(df,y))
```

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 lambda and approximately the best value of alpha.

```
insample <- df[1:100,]
outsample <- df[101:200,]
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 3.2.1
```

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

```
## Warning: package 'foreach' was built under R version 3.2.1
```

```
## Loaded glmnet 2.0-2
```

```
trainx<-as.matrix(insample[,1:90])
trainy<-insample[,91]
testx<-as.matrix(outsample[,1:90])
testy<-outsample[,91]

lambdalevels <- 10^seq(7,-2,length=100)


###############
reg=cv.glmnet(trainx,trainy,alpha=1,lambda=lambdalevels)

bestlambda <- reg$lambda.min
mse.min <- reg$cvm[reg$lambda == reg$lambda.min]
print(mse.min)
```

```
## [1] 2.602191
```

```
reg=cv.glmnet(trainx,trainy,alpha=0.8,lambda=lambdalevels)
mse.min <- reg$cvm[reg$lambda == reg$lambda.min]
print(mse.min)
```

```
## [1] 2.982033
```

```
#37.011

reg=cv.glmnet(trainx,trainy,alpha=0.6,lambda=lambdalevels)
mse.min <- reg$cvm[reg$lambda == reg$lambda.min]
print(mse.min)
```

```
## [1] 0.8829161
```

```
reg=cv.glmnet(trainx,trainy,alpha=0.4,lambda=lambdalevels)
mse.min <- reg$cvm[reg$lambda == reg$lambda.min]
print(mse.min)
```

```
## [1] 0.6643207
```

```
reg=cv.glmnet(trainx,trainy,alpha=0.2,lambda=lambdalevels)
mse.min <- reg$cvm[reg$lambda == reg$lambda.min]
print(mse.min)
```

```
## [1] 3.080613
```

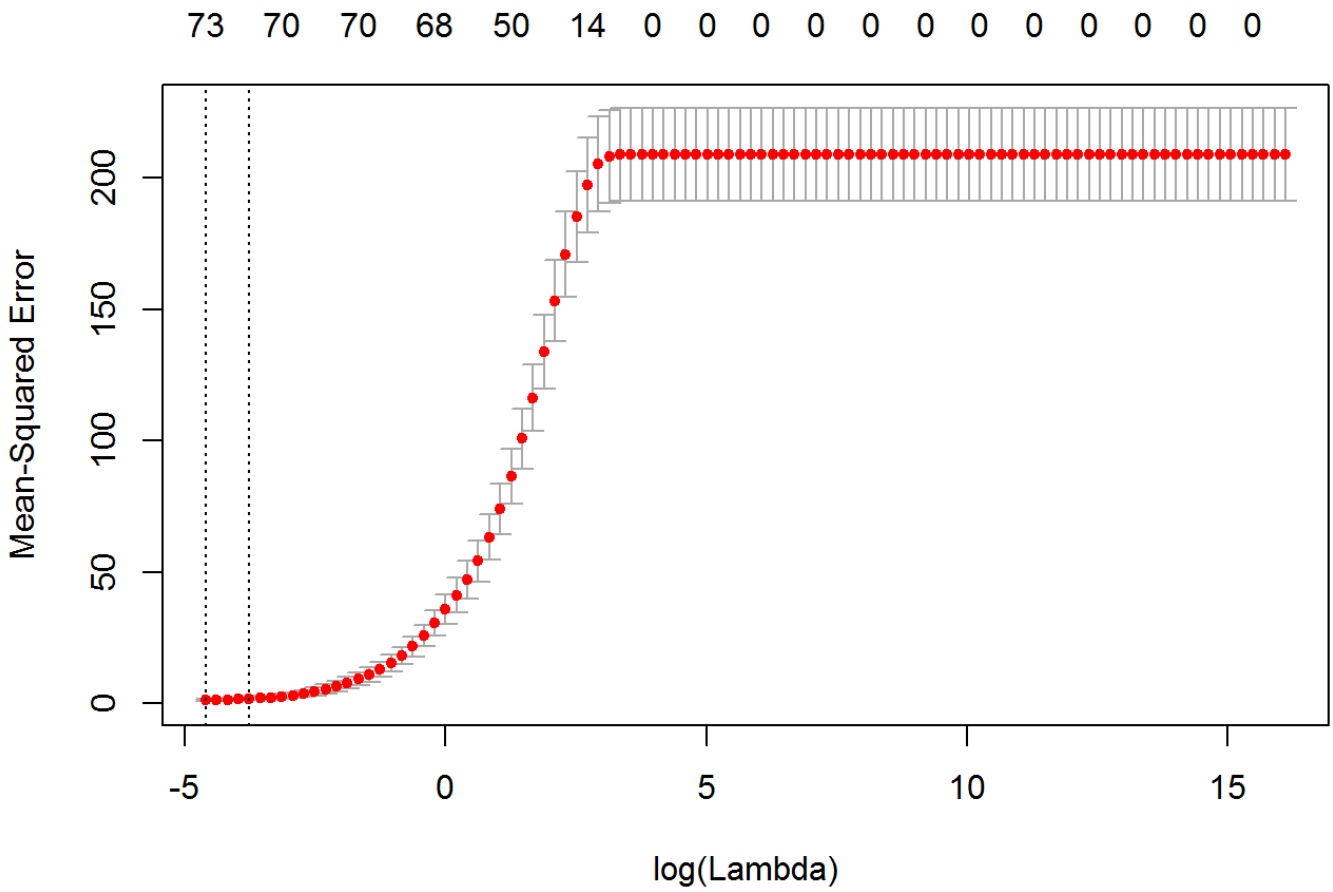
```
reg=cv.glmnet(trainx,trainy,alpha=0,lambda=lambdalevels)
mse.min <- reg$cvm[reg$lambda == reg$lambda.min]
print(mse.min)
```

```
## [1] 3.823864
```

So the best results (lowest Mean square error(MSE)) comes with alpha=0.2 .

```
reg=cv.glmnet(trainx,trainy,alpha=0.2,lambda=lambdalevels)

plot(reg)
```




```
bestlambda <- reg$lambda.min
print(bestlambda)
```

```
## [1] 0.01
```

```
mse.min <- reg$cvm[reg$lambda == reg$lambda.min]
print(mse.min)
```

```
## [1] 1.312098
```

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.

```
head(predict(reg, type="coefficients",s=bestlambda))
```

```
## 6 x 1 sparse Matrix of class "dgCMatrix"
##                        1
## (Intercept) -0.003903261
## df           1.991568209
## V2           2.005925823
## V3           2.000084572
## V4           2.010886014
## V5           2.011902188
```

we have a great output. all first 30 coefficients are close to 2, the next 30 close to -1 and the last 30are close to 0. Just as what is expected.

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

```
yhat.l <- predict(reg$glmnet.fit, s=reg$lambda.min, newx=testx)
mse.las <- sum((testy - yhat.l)^2)/nrow(testx)
mse.las
```

```
## [1] 0.01915599
```

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.

```
lmout <- lm(trainy~trainx)
summary(lmout)
```

```
## Warning in summary.lm(lmout): essentially perfect fit: summary may be
## unreliable
```

```
##
## Call:
## lm(formula = trainy ~ trainx)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -5.393e-15 -1.257e-15  1.300e-16  1.545e-15  4.678e-15
##
```

```
## Coefficients:
##               Estimate Std. Error    t value Pr(>|t|)
## (Intercept) -8.293e-15  1.808e-15 -4.587e+00  0.00132 **
## trainxdf     2.000e+00  2.144e-15  9.329e+14  < 2e-16 ***
## trainxV2     2.000e+00  3.157e-15  6.336e+14  < 2e-16 ***
## trainxV3     2.000e+00  2.555e-15  7.829e+14  < 2e-16 ***
## trainxV4     2.000e+00  1.513e-15  1.321e+15  < 2e-16 ***
## trainxV5     2.000e+00  2.064e-15  9.690e+14  < 2e-16 ***
## trainxV6     2.000e+00  2.230e-15  8.968e+14  < 2e-16 ***
## trainxV7     2.000e+00  2.181e-15  9.170e+14  < 2e-16 ***
## trainxV8     2.000e+00  3.014e-15  6.637e+14  < 2e-16 ***
## trainxV9     2.000e+00  1.597e-15  1.252e+15  < 2e-16 ***
## trainxV10    2.000e+00  1.904e-15  1.050e+15  < 2e-16 ***
## trainxV11    2.000e+00  2.146e-15  9.318e+14  < 2e-16 ***
## trainxV12    2.000e+00  2.590e-15  7.723e+14  < 2e-16 ***
## trainxV13    2.000e+00  2.320e-15  8.619e+14  < 2e-16 ***
## trainxV14    2.000e+00  2.387e-15  8.380e+14  < 2e-16 ***
## trainxV15    2.000e+00  2.069e-15  9.668e+14  < 2e-16 ***
## trainxV16    2.000e+00  1.954e-15  1.023e+15  < 2e-16 ***
## trainxV17    2.000e+00  2.808e-15  7.121e+14  < 2e-16 ***
## trainxV18    2.000e+00  2.793e-15  7.162e+14  < 2e-16 ***
## trainxV19    2.000e+00  2.707e-15  7.388e+14  < 2e-16 ***
## trainxV20    2.000e+00  2.157e-15  9.270e+14  < 2e-16 ***
## trainxV21    2.000e+00  2.427e-15  8.240e+14  < 2e-16 ***
## trainxV22    2.000e+00  2.885e-15  6.933e+14  < 2e-16 ***
## trainxV23    2.000e+00  2.747e-15  7.282e+14  < 2e-16 ***
## trainxV24    2.000e+00  1.750e-15  1.143e+15  < 2e-16 ***
## trainxV25    2.000e+00  3.160e-15  6.330e+14  < 2e-16 ***
## trainxV26    2.000e+00  3.338e-15  5.991e+14  < 2e-16 ***
## trainxV27    2.000e+00  1.624e-15  1.232e+15  < 2e-16 ***
## trainxV28    2.000e+00  1.443e-15  1.386e+15  < 2e-16 ***
## trainxV29    2.000e+00  2.695e-15  7.420e+14  < 2e-16 ***
## trainxV30    2.000e+00  2.767e-15  7.228e+14  < 2e-16 ***
## trainxV31   -1.000e+00  3.395e-15 -2.945e+14  < 2e-16 ***
## trainxV32   -1.000e+00  2.104e-15 -4.753e+14  < 2e-16 ***
## trainxV33   -1.000e+00  1.976e-15 -5.060e+14  < 2e-16 ***
## trainxV34   -1.000e+00  2.286e-15 -4.374e+14  < 2e-16 ***
## trainxV35   -1.000e+00  2.362e-15 -4.233e+14  < 2e-16 ***
## trainxV36   -1.000e+00  2.135e-15 -4.683e+14  < 2e-16 ***
## trainxV37   -1.000e+00  2.472e-15 -4.045e+14  < 2e-16 ***
## trainxV38   -1.000e+00  2.344e-15 -4.265e+14  < 2e-16 ***
## trainxV39   -1.000e+00  2.787e-15 -3.588e+14  < 2e-16 ***
## trainxV40   -1.000e+00  1.714e-15 -5.833e+14  < 2e-16 ***
## trainxV41   -1.000e+00  1.816e-15 -5.506e+14  < 2e-16 ***
## trainxV42   -1.000e+00  1.957e-15 -5.111e+14  < 2e-16 ***
## trainxV43   -1.000e+00  2.563e-15 -3.902e+14  < 2e-16 ***
## trainxV44   -1.000e+00  2.464e-15 -4.058e+14  < 2e-16 ***
## trainxV45   -1.000e+00  2.118e-15 -4.721e+14  < 2e-16 ***
## trainxV46   -1.000e+00  2.029e-15 -4.927e+14  < 2e-16 ***
## trainxV47   -1.000e+00  2.041e-15 -4.900e+14  < 2e-16 ***
```

```
## trainxV48   -1.000e+00  2.294e-15 -4.359e+14  < 2e-16 ***
## trainxV49   -1.000e+00  2.028e-15 -4.931e+14  < 2e-16 ***
## trainxV50   -1.000e+00  2.013e-15 -4.967e+14  < 2e-16 ***
## trainxV51   -1.000e+00  2.119e-15 -4.720e+14  < 2e-16 ***
## trainxV52   -1.000e+00  2.054e-15 -4.867e+14  < 2e-16 ***
## trainxV53   -1.000e+00  2.698e-15 -3.707e+14  < 2e-16 ***
## trainxV54   -1.000e+00  2.985e-15 -3.350e+14  < 2e-16 ***
## trainxV55   -1.000e+00  1.989e-15 -5.028e+14  < 2e-16 ***
## trainxV56   -1.000e+00  2.142e-15 -4.669e+14  < 2e-16 ***
## trainxV57   -1.000e+00  2.878e-15 -3.475e+14  < 2e-16 ***
## trainxV58   -1.000e+00  2.158e-15 -4.633e+14  < 2e-16 ***
## trainxV59   -1.000e+00  2.026e-15 -4.936e+14  < 2e-16 ***
## trainxV60   -1.000e+00  2.462e-15 -4.061e+14  < 2e-16 ***
## trainxV61   -3.892e-16  2.228e-15 -1.750e-01  0.86518
## trainxV62    1.008e-14  2.887e-15  3.489e+00  0.00684 **
## trainxV63    6.077e-15  2.633e-15  2.308e+00  0.04640 *
## trainxV64    2.669e-15  1.854e-15  1.439e+00  0.18389
## trainxV65    2.971e-15  1.781e-15  1.668e+00  0.12961
## trainxV66    1.650e-15  2.062e-15  8.000e-01  0.44422
## trainxV67    1.192e-15  2.306e-15  5.170e-01  0.61775
## trainxV68    2.526e-15  2.484e-15  1.017e+00  0.33588
## trainxV69   -3.489e-15  2.287e-15 -1.525e+00  0.16149
## trainxV70   -2.262e-15  2.651e-15 -8.530e-01  0.41571
## trainxV71   -3.825e-15  1.796e-15 -2.130e+00  0.06202 .
## trainxV72   -2.796e-15  2.174e-15 -1.286e+00  0.23056
## trainxV73   -1.073e-14  2.962e-15 -3.622e+00  0.00555 **
## trainxV74    4.611e-15  2.228e-15  2.070e+00  0.06842 .
## trainxV75   -2.206e-15  2.609e-15 -8.460e-01  0.41969
## trainxV76   -1.004e-15  1.740e-15 -5.770e-01  0.57811
## trainxV77   -1.869e-15  2.265e-15 -8.250e-01  0.43058
## trainxV78   -3.656e-15  2.292e-15 -1.595e+00  0.14513
## trainxV79    2.079e-17  2.172e-15  1.000e-02  0.99257
## trainxV80   -5.065e-16  2.828e-15 -1.790e-01  0.86181
## trainxV81   -6.672e-15  3.158e-15 -2.113e+00  0.06381 .
## trainxV82    4.775e-16  1.588e-15  3.010e-01  0.77042
## trainxV83    1.977e-15  2.759e-15  7.170e-01  0.49179
## trainxV84    3.238e-15  1.992e-15  1.626e+00  0.13848
## trainxV85   -7.603e-15  2.460e-15 -3.091e+00  0.01292 *
## trainxV86   -5.112e-15  2.020e-15 -2.530e+00  0.03221 *
## trainxV87   -7.865e-17  2.441e-15 -3.200e-02  0.97500
## trainxV88   -2.881e-15  2.228e-15 -1.293e+00  0.22821
## trainxV89    6.044e-15  2.438e-15  2.479e+00  0.03506 *
## trainxV90   -4.430e-15  2.039e-15 -2.173e+00  0.05787 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.851e-15 on 9 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 4.853e+30 on 90 and 9 DF,  p-value: < 2.2e-16
```

```
yhat.r <- cbind(1,testx) %*% lmout$coefficients
# ^^ we predict via matrix multiplication rather than just using predict()
#    because x was made a matrix to make glmnet happy
mse.reg <- sum((testy - yhat.r)^2)/nrow(testx)
mse.reg
```

```
## [1] 1.438141e-27
```

what we see here is a perfect fit between regression coefficients and real values in a way that R^2 =1. This is may be because the data size is very well suited for multiple regression. We do not have too much observation (very high n) to cause over fitting. Having very large n (~ 10^6)however, I expect elastic net method more accurate.beacuase of overfitting.

---

9. As in problem 6, use the data from 8 to generate a new y2 that is 1 if y > 0 and 0 otherwise.

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

```
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 3.2.1
```

```
y2<-as.factor(ifelse(y>0,1,0))
df<-cbind(df,y2)
df<-df[,-91]   #removing y
traindf<-df[1:100,]
testdf<-df[101:200,]
costvalues <- 10^seq(-3,2,1)


tuned.svm <- tune(svm,y2~.,data=traindf,ranges=list(cost=costvalues), kernel="linear")
yhat <- predict(tuned.svm$best.model,newdata=testdf)
table(predicted=yhat,truth=testdf$y)
```

```
##          truth
## predicted  0  1
##         0 39 20
##         1 11 30
```

```
sum(yhat==testdf$y)/length(testdf$y)
```

```
## [1] 0.69
```

```
tuned.svm <- tune(svm,y2~.,data=traindf,ranges=list(cost=costvalues), kernel="radial")
yhat <- predict(tuned.svm$best.model,newdata=testdf)
table(predicted=yhat,truth=testdf$y)
```

```
##          truth
## predicted  0  1
##         0 45 23
##         1  5 27
```

```
sum(yhat==testdf$y)/length(testdf$y)
```

```
## [1] 0.72
```

apparently linear kernel (81% accurate) works better than radial kernel(70% accurate) in this dataset.