Lab: Week 9

36-350 – Statistical Computing

Week 9 – Fall 2020

Name: Kimberly Zhang

Andrew ID: kyz

You must submit **your own** lab as a PDF file on Gradescope.

Question 1

(5 points)

Notes 9A (2)

Compute $f(x_1, x_2)$ for the point $x_1 = 1$ and $x_2 = 1$ given the bivariate normal with mean $\mu = \{2, 2\}$ and $\sigma = \{2, 2\}$ and $\rho = 0$. *Do this using the formula on slide 2 of Notes 9A, and not via the use of `dmvnorm()`.*

```
mu = c(2,2)
sigma.1 = 2
sigma.2 = 2
rho.12  = 0
Sigma = matrix(c(sigma.1^2,rho.12*sigma.1*sigma.2,rho.12*sigma.1*sigma.2,
                 sigma.2^2),nrow=2)
x= c(1, 1)
1/sqrt((2*pi)^2*det(Sigma))*exp(-0.5*t(x-mu)%*%solve(Sigma)%*%(x-mu))
```

```
##           [,1]
## [1,] 0.0309875
```

Question 2

(5 points)

Notes 9A (3)

Verify your answer for Q1 using `dmvnorm()`.

```
if ( require(emdbook) == FALSE ) {
  install.packages("emdbook",repos="https://cloud.r-project.org")
  library(emdbook)
}
```

```
## Loading required package: emdbook
```

```
dmvnorm(x, mu, Sigma)
```

```
## [1] 0.0309875
```

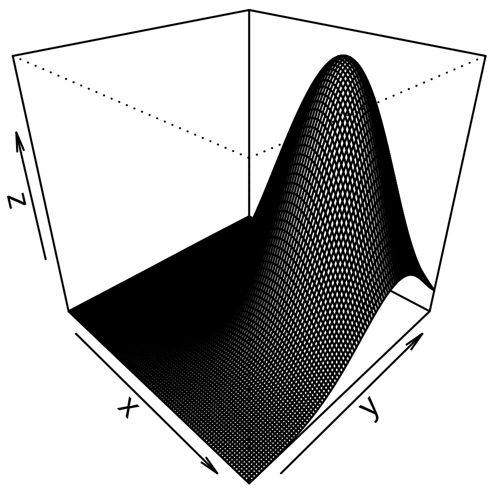
Question 3

(5 points)

Notes 9A (3)

Use `persp()` to display your bivariate normal from Q1 and Q2. The inputs are `x`, `y`, and `z`, where `x` and `y` are vectors indicating the input values to `dmvnorm()` (think: sequences where the steps are small enough so that the displayed bivariate surface is smooth), and `z` is a matrix of output values. The simplest approach here is to use nested `for` loops to compute each value for `z`. In your call to `persp()`, set the arguments `theta` and `phi` to 45 and 30, respectively.

```r
x = seq(-5,5,by=0.1)
y = seq(-5,5,by=0.1)
z = matrix(rep(NA, length(x)*length(y)), nrow= length(x))
for (i in 1:length(x)){
  for (j in 1:length(y)){
    z[i, j]=dmvnorm(c(x[i], y[j]), mu, Sigma)
  }
}
persp(x, y, z, theta = 45, phi= 30)
```



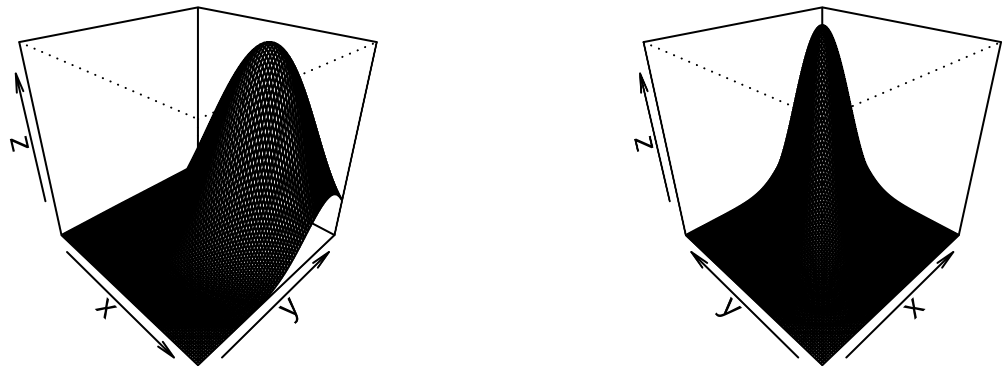
Question 4

(5 points)

Notes 9A (3)

Repeat Q3, except change the bivariate normal to have correlation coefficient $\rho = 0.5$. Show two perspective plots, one with (`theta`,`phi`) = (45,30), and one with values (-45,30). You should be able to see the effects of the non-zero correlation coefficient as you toggle between the two plots.

```
rho.12  = 0.5
Sigma = matrix(c(sigma.1^2,rho.12*sigma.1*sigma.2,rho.12*sigma.1*sigma.2,
                 sigma.2^2),nrow=2)
z = matrix(rep(NA, length(x)*length(y)), nrow= length(x))
for (i in 1:length(x)){
  for (j in 1:length(y)){
    z[i, j]=dmvnorm(c(x[i], y[j]), mu, Sigma)
  }
}
par(mfrow=c(1,2))
persp(x, y, z, theta = 45, phi= 30)
persp(x, y, z, theta = -45, phi= 30)
```



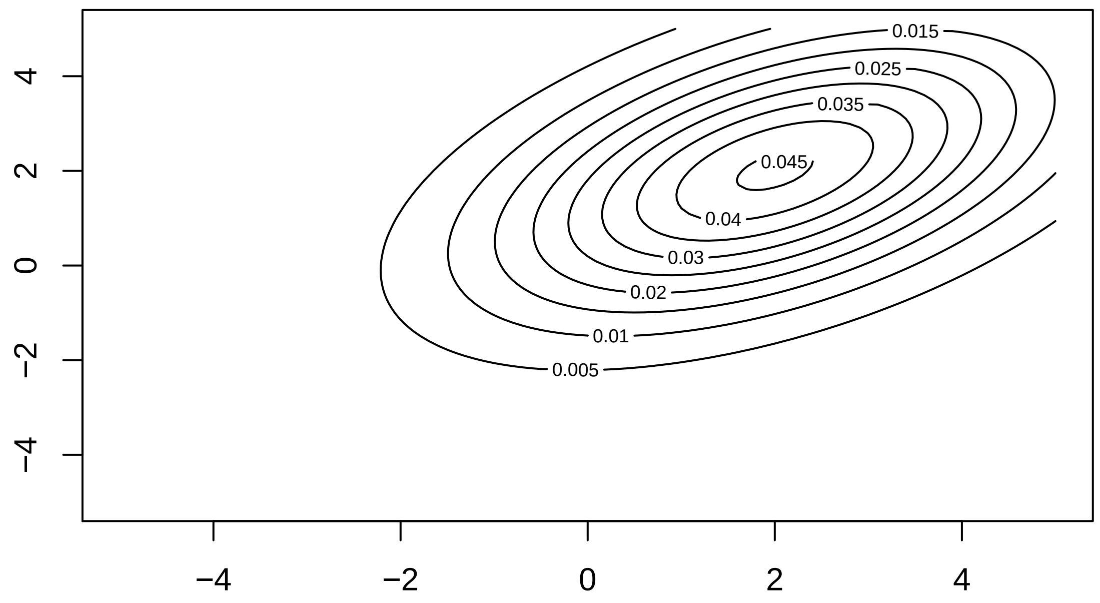
Question 5

(5 points)

Notes 9A (3)

Use `contour()` to create a contour plot of the multivariate normal from Q4. Because $\rho > 0$, you should see that the contours are “tilted” so that they are lower on the left and higher on the right.

```
contour(x, y, z)
```



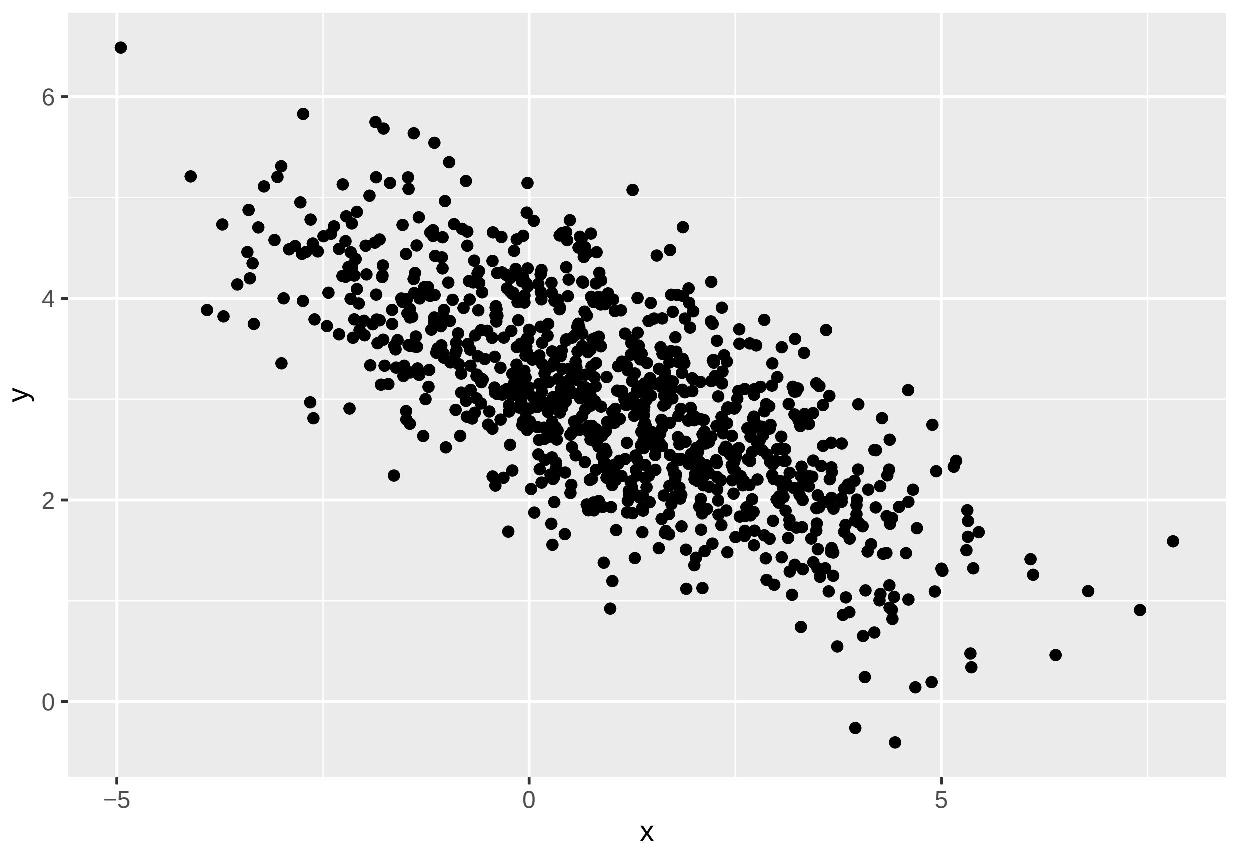
Question 6

(5 points)

Notes 9A (4)

Sample 1000 data from a bivariate normal with $\mu = \{1,3\}$, $\sigma = \{2,1\}$, and $\rho = -0.75$, and plot your sampled points using `ggplot`.

```r
suppressMessages(library(tidyverse))
suppressMessages(library(MASS))
mu = c(1,3)
sigma.1 = 2
sigma.2 = 1
rho.12  = -0.75
Sigma = matrix(c(sigma.1^2,rho.12*sigma.1*sigma.2,rho.12*sigma.1*sigma.2,sigma.2^2),nrow=2)
set.seed(101)
data = mvrnorm(1000,mu,Sigma)
df = data.frame(x=data[,1],y=data[,2])
ggplot(data=df,mapping=aes(x=x,y=y)) + geom_point()
```



Question 7

(5 points)

Notes 9A (5)

Display both the population covariance matrix and the sample covariance matrix (given your sample from Q6). The take-away point here is that the elements of the sample covariance matrix are random variables! (Whose values are hopefully close to those of the population matrix.)

```
Sigma
```

```
##      [,1] [,2]
## [1,]  4.0 -1.5
## [2,] -1.5  1.0
```

```
cov(df)
```

```
##           x          y
## x  3.687473 -1.3590332
## y -1.359033  0.9743536
```

You are given the following multivariate normal:

```
mu = c(1,2,6,2,-4)
sigma = c(1,2,1,0.5,2)
rho = diag(rep(1,5))
rho[1,2] = rho[2,1] = 0.4
```

```r
rho[1,3] = rho[3,1] = -0.3
rho[1,4] = rho[4,1] = -0.7
rho[3,5] = rho[5,3] = 0.2
rho[4,5] = rho[5,4] = 0.5
Sigma = rho * (sigma %o% sigma)
```

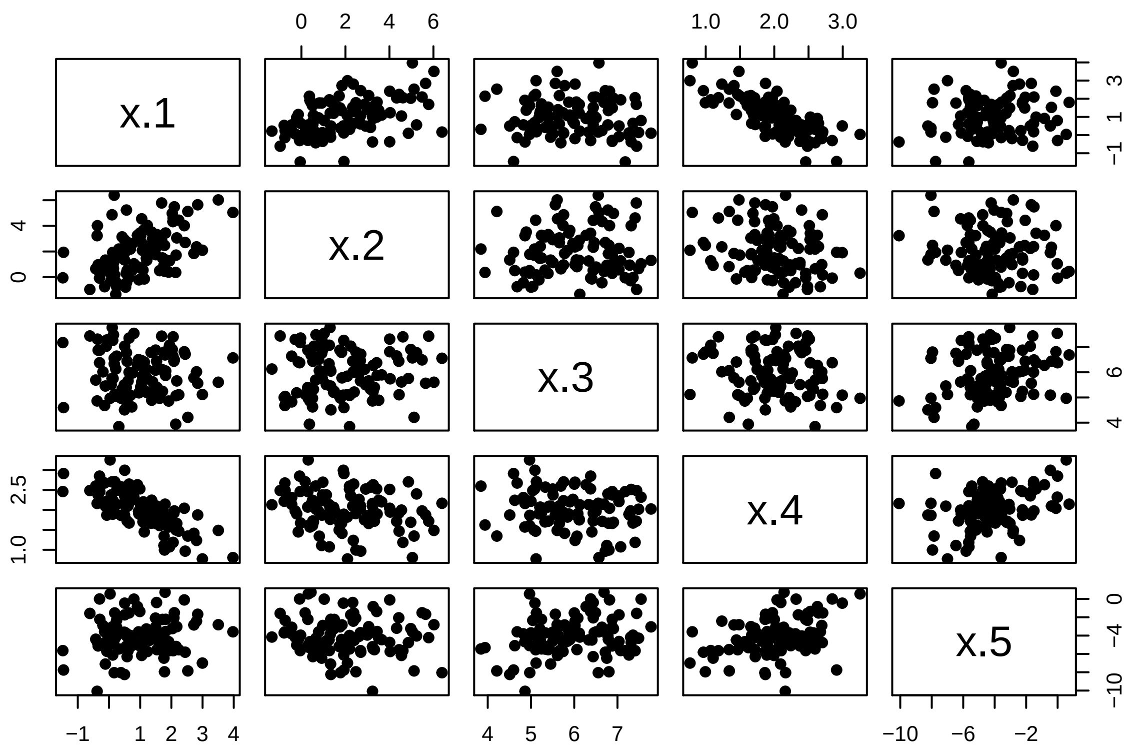
Question 8

(5 points)

Notes 9A (4)

Sample 100 data from this distribution. Display your sampled data via the `pairs()` function, which generates a pairwise grid of scatterplots. Change the labels on the `pairs()` plot to "x.1", "x.2", . . . , "x.5". Also, apply the argument `pch=19`.

```r
set.seed(101)
data = mvrnorm(100,mu,Sigma)
pairs(data, labels = c("x.1", "x.2", "x.3", "x.4", "x.5"), pch=19)
```



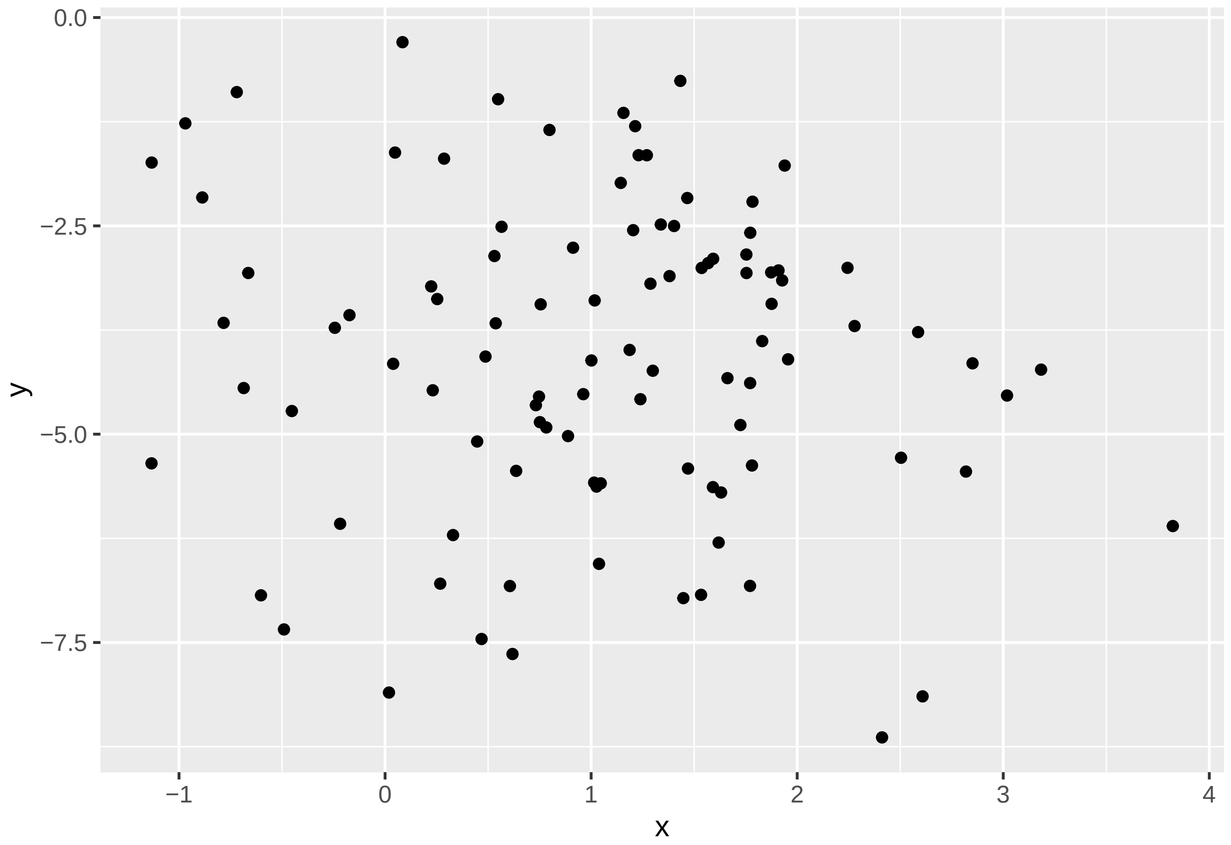
Question 9

(5 points)

Notes 9A (4,6)

Sample 100 data from the marginal distribution $f(x_1, x_5)$. Plot your result with either a base `R` or `ggplot` plotting function.

```r
set.seed(101)
p= c(1,5)
mu.marginal = mu[p]
Sigma.marginal= Sigma[p, p]
data = mvrnorm(100,mu.marginal,Sigma.marginal)
df = data.frame(x=data[,1],y=data[,2])
ggplot(data=df,mapping=aes(x=x,y=y)) + geom_point()
```



Question 10

(5 points)

Notes 9A (7-8)

Compute the mean and the covariance matrix for $f(x_2, x_3|x_1 = 1, x_4 = 1, x_5 = 1)$. Display your results.

```r
k = c(2,3)
d.minus.k = c(1,4,5)
x.1 = 1 # condition: x_3 = 2
x.4 = 1 # condition: x_4 = 3
x.5 = 1
Sigma.kk = Sigma[k,k]
Sigma.kd = Sigma[k,d.minus.k]
Sigma.dk = Sigma[d.minus.k,k]
Sigma.dd = Sigma[d.minus.k,d.minus.k]
mu.cond = mu[k] +
  Sigma.kd %*% solve(Sigma.dd) %*% matrix(c(x.1,x.4,x.5)-mu[d.minus.k],nrow=3)
```

```
mu.cond
```

```
##        [,1]
## [1,] -5.000
## [2,] 10.375
```

```
Sigma.cond = Sigma.kk - Sigma.kd %*% solve(Sigma.dd) %*% Sigma.dk
Sigma.cond
```

```
##           [,1]      [,2]
## [1,] 2.1538462 0.9076923
## [2,] 0.9076923 0.5003846
```

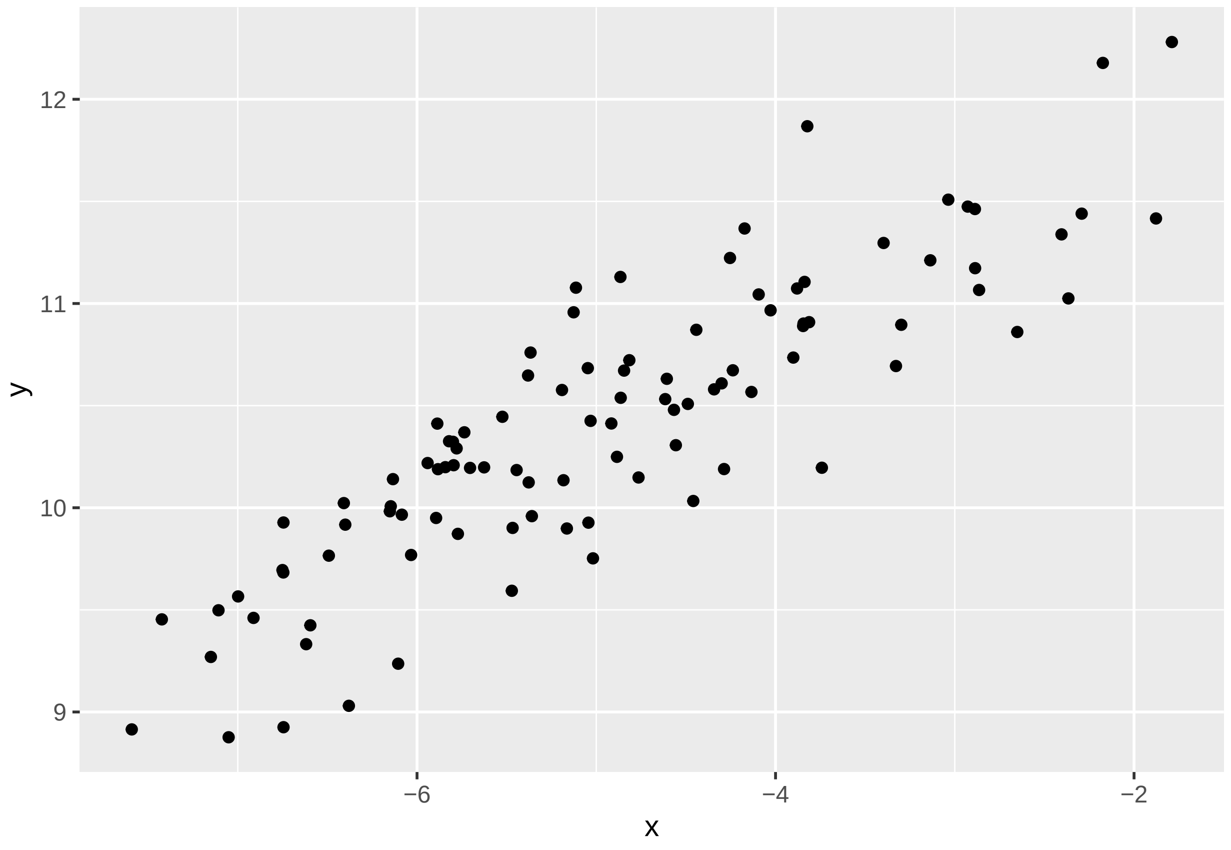
Question 11

(5 points)

Notes 9A (4,6)

Sample 100 data from the conditional distribution $f(x_2, x_3|x_1 = 1, x_4 = 1, x_5 = 1)$. Plot your result with either a base `R` or `ggplot` plotting function.

```
set.seed(101)
data = mvrnorm(100,mu.cond,Sigma.cond)
df = data.frame(x=data[,1],y=data[,2])
ggplot(data=df,mapping=aes(x=x,y=y)) + geom_point()
```



In the following code chunk, we input four morphological measurements for each of 3419 galaxies. The data frame is `df`.

```r
load(url("http://www.stat.cmu.edu/~pfreeman/Lab_09_PCA.Rdata"))
names(df)
```

```
## [1] "G"    "M20"  "C"    "size"
```

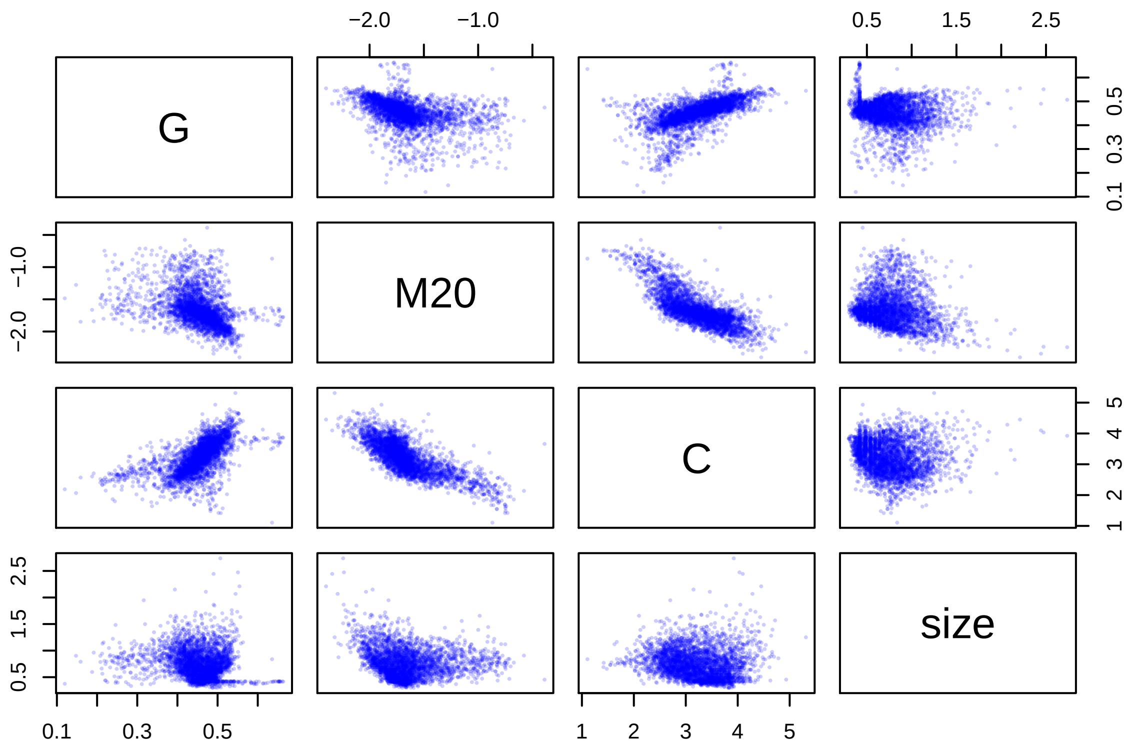
Question 12

(5 points)

Not in Notes

As you did in Q8, create a `pairs()` plot for the data frame `df`. Here, there is no need to change the labels, although you should still apply the argument `pch=19`...and since there are many points, also apply the argument `cex=0.2`, which shrinks the sizes of the points. Last, apply the argument `col=rgb(0,0,1,alpha=0.2)`: this changes the color of the points to blue (the "(0,0,1)" part of the function call), and makes them mostly transparent (the "alpha=0.2" part of the function call...a value of 1 is opaque). Two things should jump out at you: (1) the data are not distributed as a multivariate normal, and (2) there are definite correlations between at least some pairs of variables.

```r
pairs(df, pch=19, cex= 0.2, col=rgb(0,0,1,alpha=0.2))
```



Question 13

(5 points)

Not in Notes

Use the `scale()` function to scale the data frame `df`. By default, `scale()` will subtract the sample mean of each column's data from the data values, then divide by the sample standard deviation. (In 36-225 talk, this is "standardizing" the data.) Save the scaled data frame as `df.scaled`. Confirm that the data are scaled by using `colMeans()` to check the sample means, and by using `apply()` with an appropriate function to check the sample standard deviations. (You may observe that the means are not exactly zero. This is normal.)

```
df.scaled = scale(df)
colMeans(df.scaled)
```

```
##             G           M20             C          size
##  2.377008e-16 -2.196032e-16 -3.047423e-17 -4.943277e-17
```

```
apply(df.scaled, 2, sd)
```

```
##    G  M20    C size
##    1    1    1    1
```

Question 14

(5 points)

Notes 9B (4)

Compute the eigenvectors and eigenvalues for the covariance matrix of `df.scaled`. Save these as `v` and `lambda`, respectively. Use `dim()` to display the dimensionality of the matrix `v` and `length()` to show the length of the vector `lambda`. Also check, using `all()` with a relational operator, that all the values of lambda are greater than zero. Finally, display `v` and the square root of `lambda`.

```
A = cov(df.scaled)
eA = eigen(A)
lambda = eA$values
v = eA$vectors
dim(v)
```

```
## [1] 4 4
```

```
length(lambda)
```

```
## [1] 4
```

```
all(lambda > 0)
```

```
## [1] TRUE
```

```
v
```

```
##             [,1]        [,2]       [,3]       [,4]
## [1,]  0.51461474 -0.21735992  0.8020834  0.2111600
## [2,] -0.57408998 -0.31096317  0.4453620 -0.6126788
## [3,]  0.63449159 -0.02505601 -0.2188305 -0.7408818
## [4,] -0.05488889  0.92489392  0.3323072 -0.1764379
```

```
sqrt(lambda)
```

```
## [1] 1.4800888 1.0405683 0.7494448 0.4060633
```

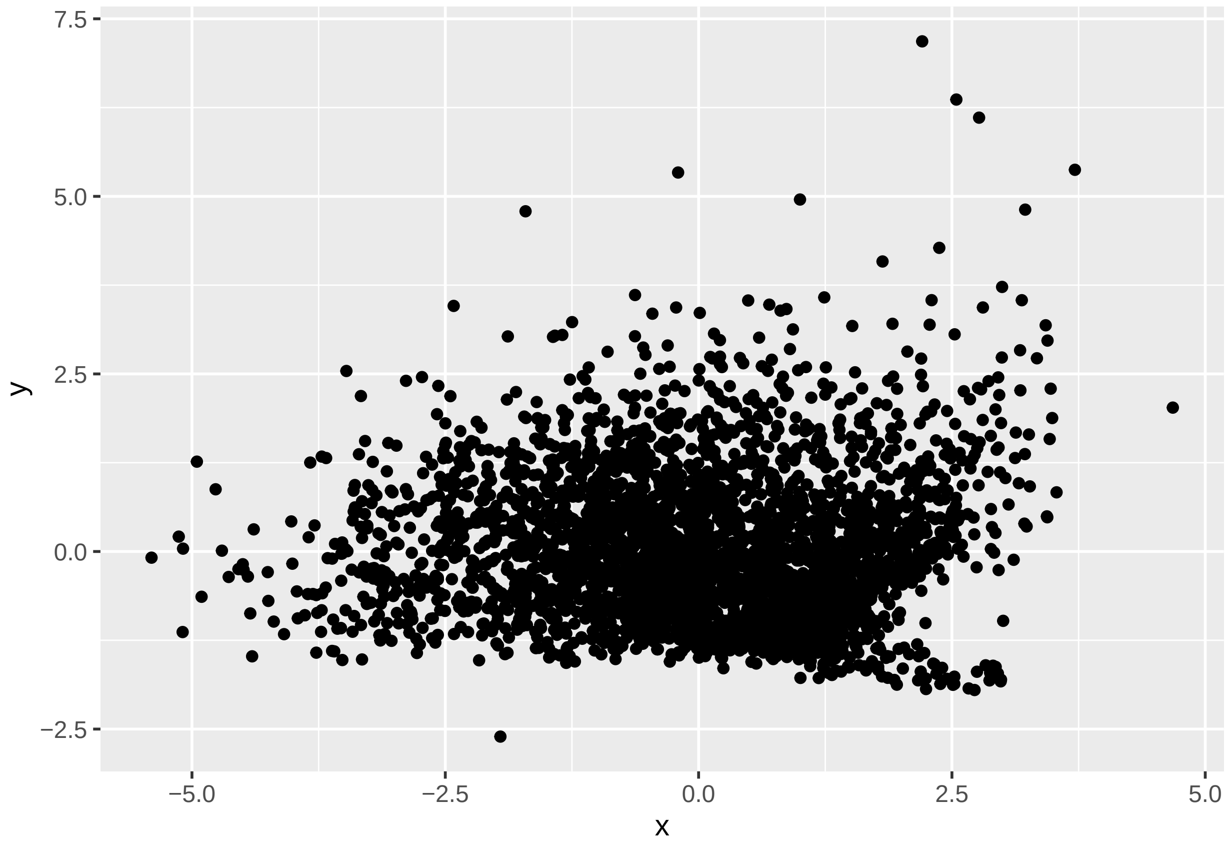
Question 15

(5 points)

Notes 9B (5)

Use matrix multiplication to multiply `df.scaled` and `v`. Save the result as `z`. Use `ggplot` to display a scatterplot for the first two columns of `z`. (You need not overlay lines like those in the plot of Slide 5 of Notes 9B.)

```
z = df.scaled%*%v
data = data.frame(x=z[,1],y=z[,2])
ggplot(data=data,mapping=aes(x=x,y=y)) + geom_point()
```



Congratulations! In Q13-Q15, you recreated principal components analysis by hand. Now let's use `prcomp()`.

Question 16

(5 points)

Notes 9B (9)

Use `prcomp()` to perform PCA on `df` directly. Divide the rotation matrix output by `prcomp()` by `v` from Q14, and the standard deviations (`sdev`) output by `prcomp()` by the square root of `lambda` (also from Q14). If you did everything right, you will see that the first division yields only values of -1 or 1 (due to the arbitrariness of signs of the eigenvectors), and the second division yields only values of 1.

```
p = prcomp(df, scale = TRUE)
p$rotation/v
```

```
##      PC1 PC2 PC3 PC4
## G     -1   1  -1  -1
## M20   -1   1  -1  -1
## C     -1   1  -1  -1
## size  -1   1  -1  -1
```

```
p$sdev/sqrt(lambda)
```

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

Question 17

(5 points)

Notes 9B (9)

Display the rotation matrix output by `prcomp()`. How does each principal component map to the original variables? (Each displayed principal component is a vector; the larger the value for a particular original variable, the more the vector points in that direction.)

```
p$rotation
```

```
##              PC1         PC2        PC3        PC4
## G    -0.51461474 -0.21735992 -0.8020834 -0.2111600
## M20   0.57408998 -0.31096317 -0.4453620  0.6126788
## C    -0.63449159 -0.02505601  0.2188305  0.7408818
## size  0.05488889  0.92489392 -0.3323072  0.1764379
```

```
PC1: variability comes from G and C in opposite direction, and M20 in positive
direction
PC2: size is the primary source of data variability in
direction orthogonal to PC1
PC3: primarily G in the opposite direction
PC4: dominated by C
```

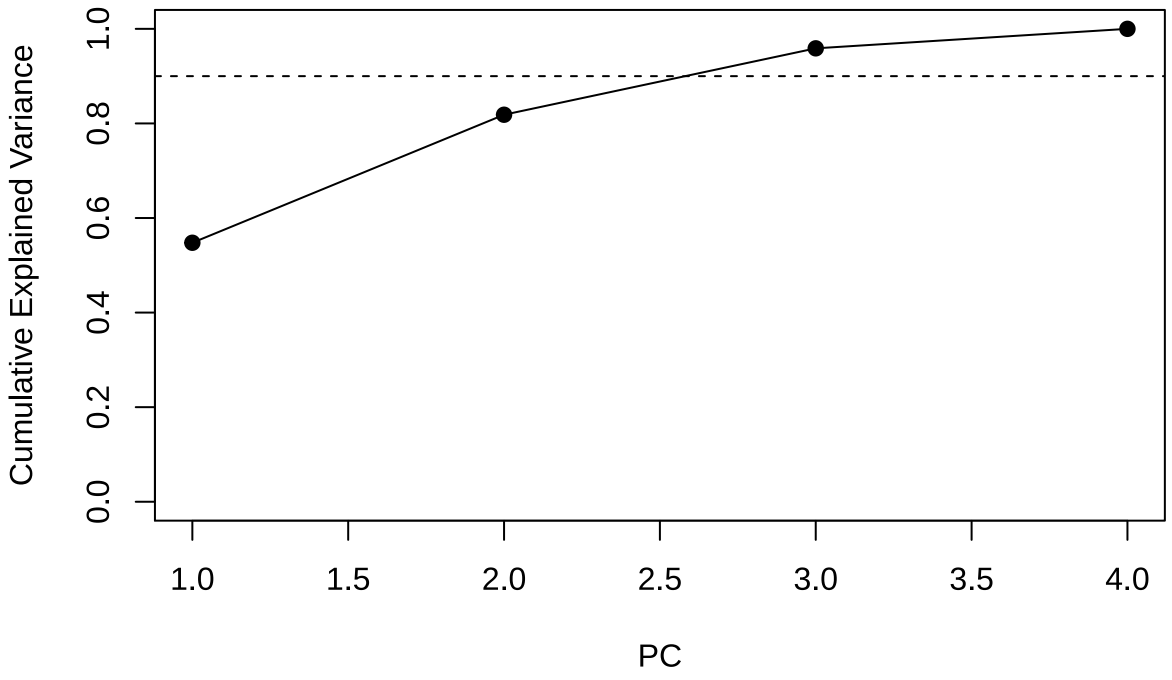
Question 18

(5 points)

Notes 9B (10)

Create a plot showing the proporation of variance explained as a function of the number of principal components. (Use base `R` plotting.) Add a dashed line across the plot for proporation of variance explained equals 0.9. (See `abline()`.) How many principal components would you keep if you were attempting dimension reduction?

```
plot(1:4,cumsum(p$sdev^2)/sum(p$sdev^2),pch=19,xlab="PC",
     ylab="Cumulative Explained Variance",ylim=c(0,1))
lines(1:4,cumsum(p$sdev^2)/sum(p$sdev^2))
abline(h=0.9, lty= "dashed")
```



```
I would keep 3 principal components.
```

Question 19

(5 points)

Not in Notes

Determine which of the elements of the output from `prcomp()` corresponds to the principal component scores, then verify that the data in the score columns are uncorrelated.

```r
cor(p$x)
```

```
##               PC1           PC2           PC3           PC4
## PC1 1.000000e+00  7.539378e-16  3.070020e-15  4.597451e-15
## PC2 7.539378e-16  1.000000e+00 -9.176824e-16 -8.986795e-16
## PC3 3.070020e-15 -9.176824e-16  1.000000e+00 -2.130108e-15
## PC4 4.597451e-15 -8.986795e-16 -2.130108e-15  1.000000e+00
```

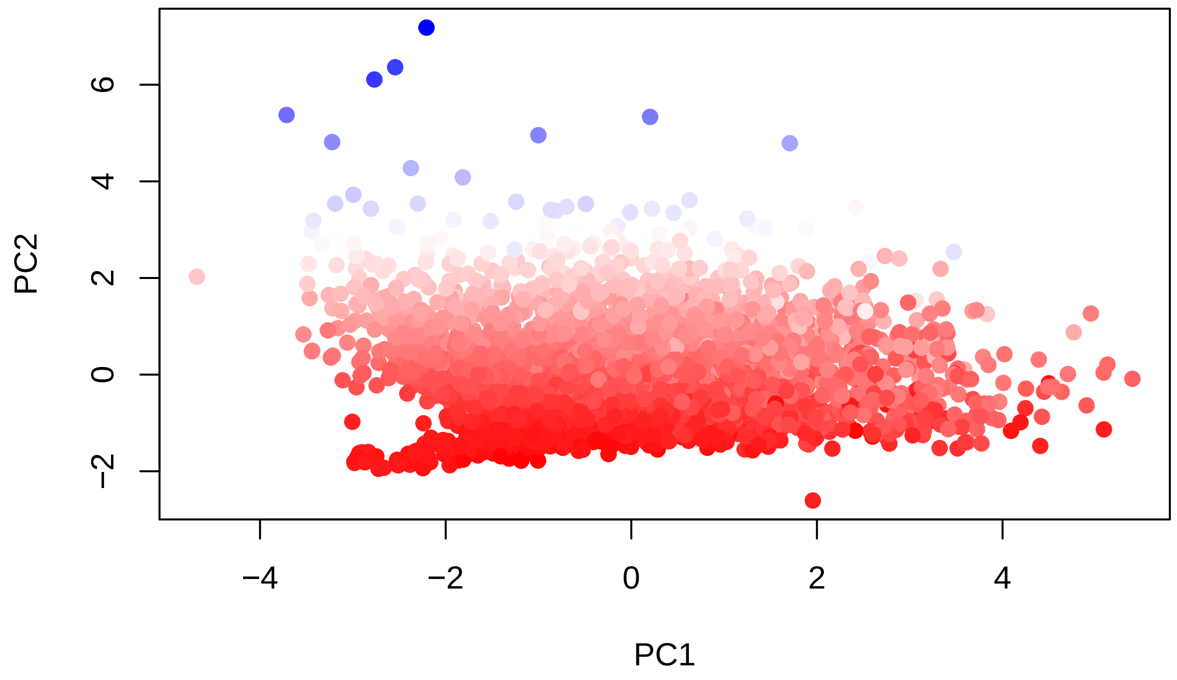
Question 20

(5 points)

Notes 9B (11)

Using base `R` plotting functionality, plot the first two principal component scores, and color the data by the value of `df$size`. Because PC 2 points almost completely along the `size` axis, you should see a definite gradient in color as you go from the bottom to the top of your plot.

```
scores = p$x
gr = .bincode(df$size, seq(min(df$size), max(df$size), len=length(df$size)),
              include.lowest = TRUE)
col = colorRampPalette(c("red", "white", "blue"))(length(df$size))[gr]
plot(scores[,1], scores[,2], col= col, pch= 19, xlab = "PC1", ylab= "PC2")
```


PC2
PC1