Gene Expression Analysis and Visualization
410.671
HW #1

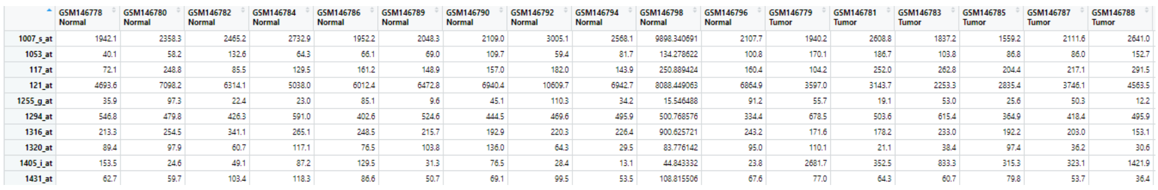
For this homework, we will be working with a study from Gene Expression Omnibus (GEO) with the accession GDS2880. This is an Affymetrix microarray experiment (HGU133A array). The data researchers were investigating patient matched normal and stage 1 or stage 2 clear cell renal cell carcinoma (cRCC) tumors to provide insight into the molecular pathogenesis of cRCC. We will be conducting outlier analysis using various methods to identify aberrant samples, followed by missing value imputation to assess the accuracy of two different algorithms.

1.) Download and load the renal cell carcinoma data file into R. Make sure that the row names are in the correct location (Affymetrix fragment names). Look at the dimensions and verify that you have 22 arrays and 22,283 probesets. (2pts.)

```
> str(carcinoma)
'data.frame':   22283 obs. of  22 variables:
```

2.) Label the header columns of your data frame maintaining the GSM ID, but adding the Normal/Tumor identity. (2pts.)

```
colnames(carcinoma) <- paste(colnames(carcinoma),annotation01$`N/T`,sep = '\n')
```

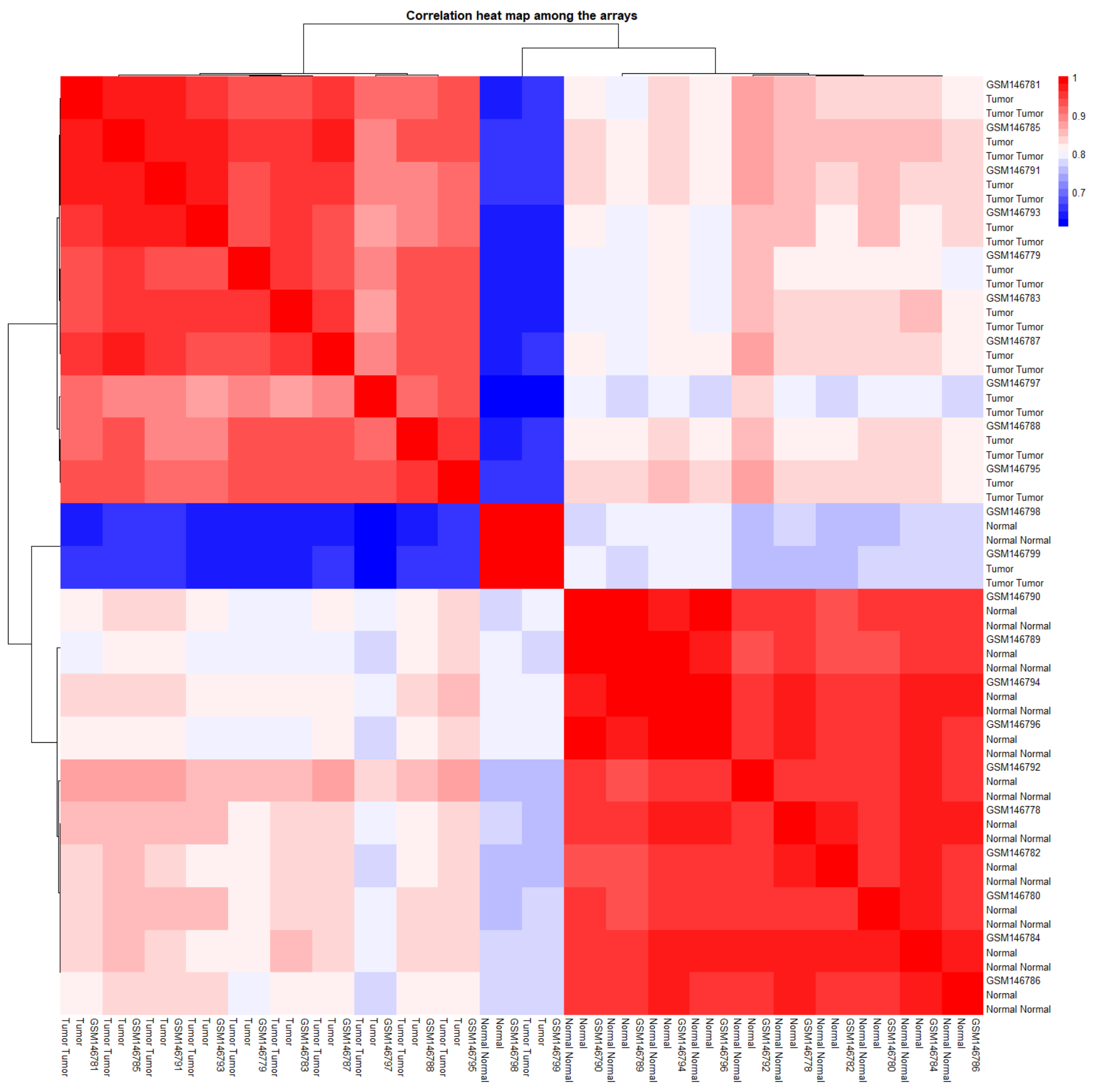


	GSM146778 Normal	GSM146780 Normal	GSM146782 Normal	GSM146784 Normal	GSM146786 Normal	GSM146789 Normal	GSM146790 Normal	GSM146792 Normal	GSM146794 Normal	GSM146798 Normal	GSM146796 Normal	GSM146779 Tumor	GSM146781 Tumor	GSM146783 Tumor	GSM146785 Tumor	GSM146787 Tumor	GSM146788 Tumor
1007_s_at	1942.1	2358.3	2465.2	2732.9	1952.2	2048.3	2109.0	3005.1	2568.1	9898.340691	2107.7	1940.2	2608.8	1837.2	1559.2	2111.6	2641.0
1053_at	40.1	58.2	132.6	64.3	66.1	69.0	109.7	59.4	81.7	134.278622	100.8	170.1	186.7	103.8	86.8	86.0	152.7
117_at	72.1	248.8	85.5	129.5	161.2	148.9	157.0	182.0	143.9	250.889424	160.4	104.2	252.0	262.8	204.4	217.1	291.5
121_at	4693.6	7098.2	6314.1	5038.0	6012.4	6472.8	6940.4	10609.7	6942.7	8088.449063	6864.9	3597.0	3143.7	2253.3	2835.4	3746.1	4563.5
1255_g_at	35.9	97.3	22.4	23.0	85.1	9.6	45.1	110.3	34.2	15.546488	91.2	55.7	19.1	53.0	25.6	50.3	12.2
1294_at	546.8	479.8	426.3	591.0	402.6	524.6	444.5	469.6	495.9	500.768576	334.4	678.5	503.6	615.4	364.9	418.4	495.9
1316_at	213.3	254.5	341.1	265.1	248.5	215.7	192.9	220.3	226.4	900.625721	243.2	171.6	178.2	233.0	192.2	203.0	153.1
1320_at	89.4	97.9	60.7	117.1	76.5	103.8	136.0	64.3	29.5	83.776142	95.0	110.1	21.1	38.4	97.4	36.2	30.6
1405_i_at	153.5	24.6	49.1	87.2	129.5	31.3	76.5	28.4	13.1	44.843332	23.8	2681.7	352.5	833.3	315.3	323.1	1421.9
1431_at	62.7	59.7	103.4	118.3	86.6	50.7	69.1	99.5	53.5	108.815506	67.6	77.0	64.3	60.7	79.8	53.7	36.4

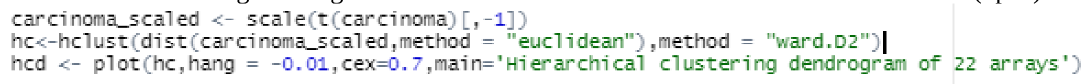
3.) Identify any outlier samples using the following visual plots:

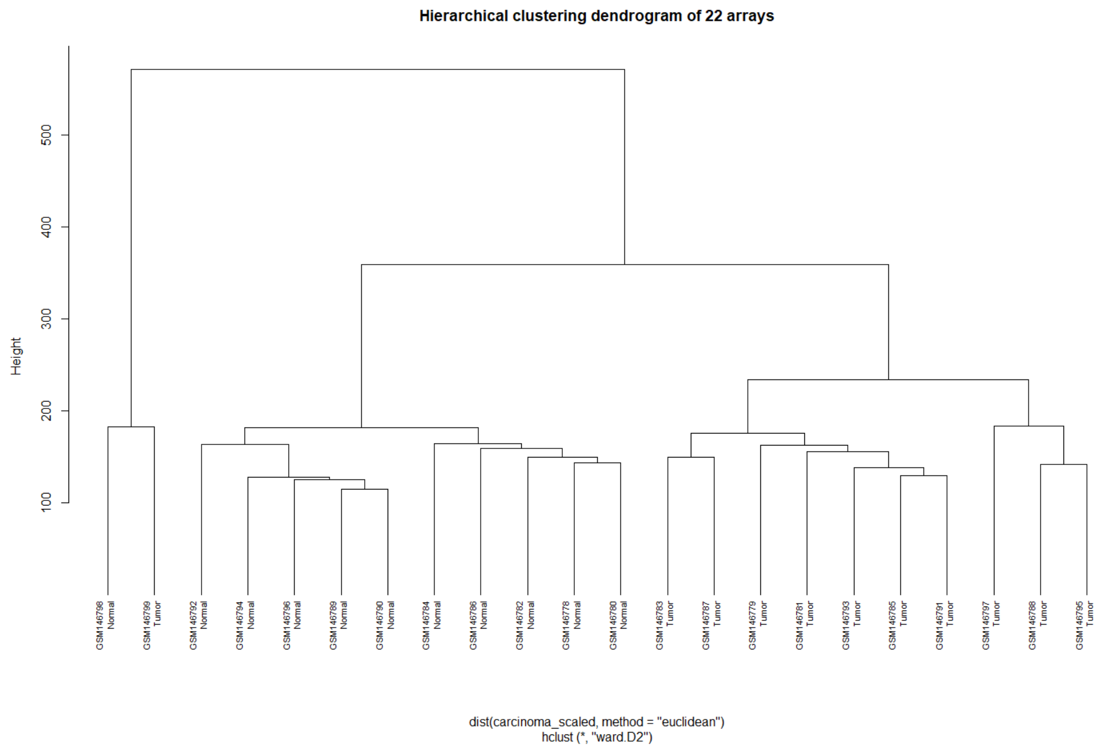
4.) Correlation plot (heat map) (2pts.)

```
pearsonmatrix <- cor(carcinoma,method = 'pearson',use = 'pairwise.complete.obs')
col <- colorRampPalette(c('blue','white','red'))(20)
p <- pheatmap(pearsonmatrix,col=col,clustering_distance_rows='correlation',clustering_distance_cols='correlation',border=F,
              main = 'Correlation heat map among the arrays')
```



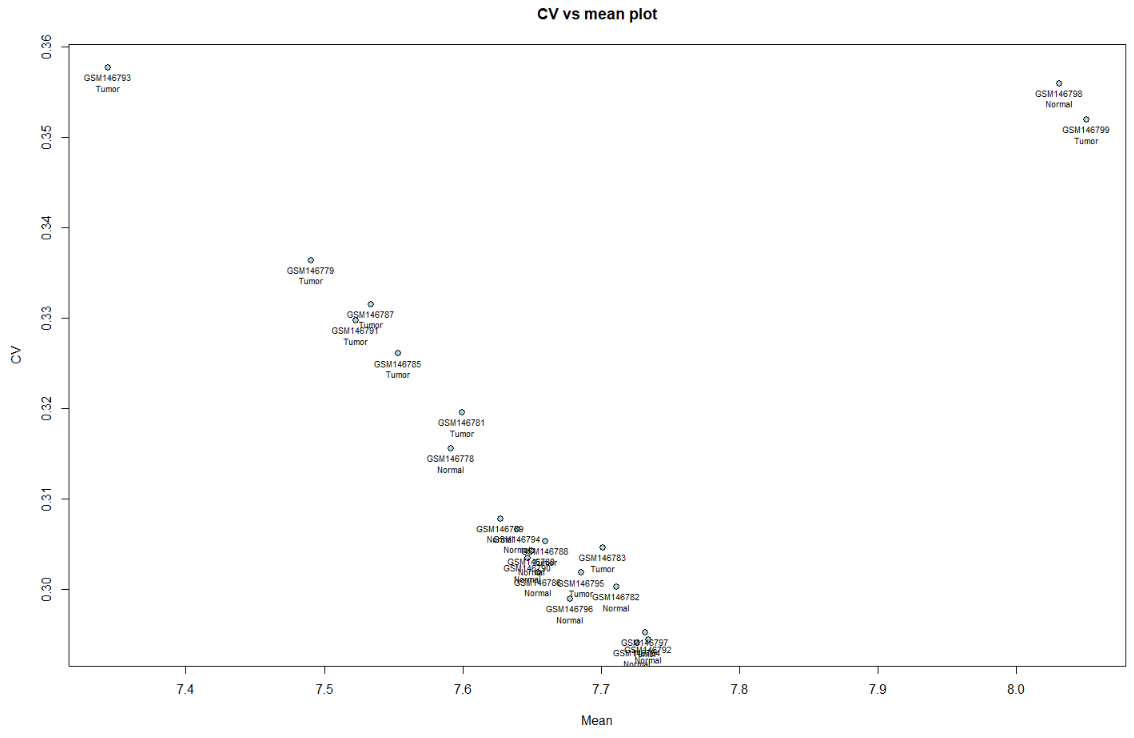
Hierarchical clustering dendrogram (2pts.)


```
carcinoma_scaled <- scale(t(carcinoma)[,-1])
hc<-hclust(dist(carcinoma_scaled,method = "euclidean"),method = "ward.D2")
hcd <- plot(hc,hang = -0.01,cex=0.7,main='Hierarchical clustering dendrogram of 22 arrays')
```



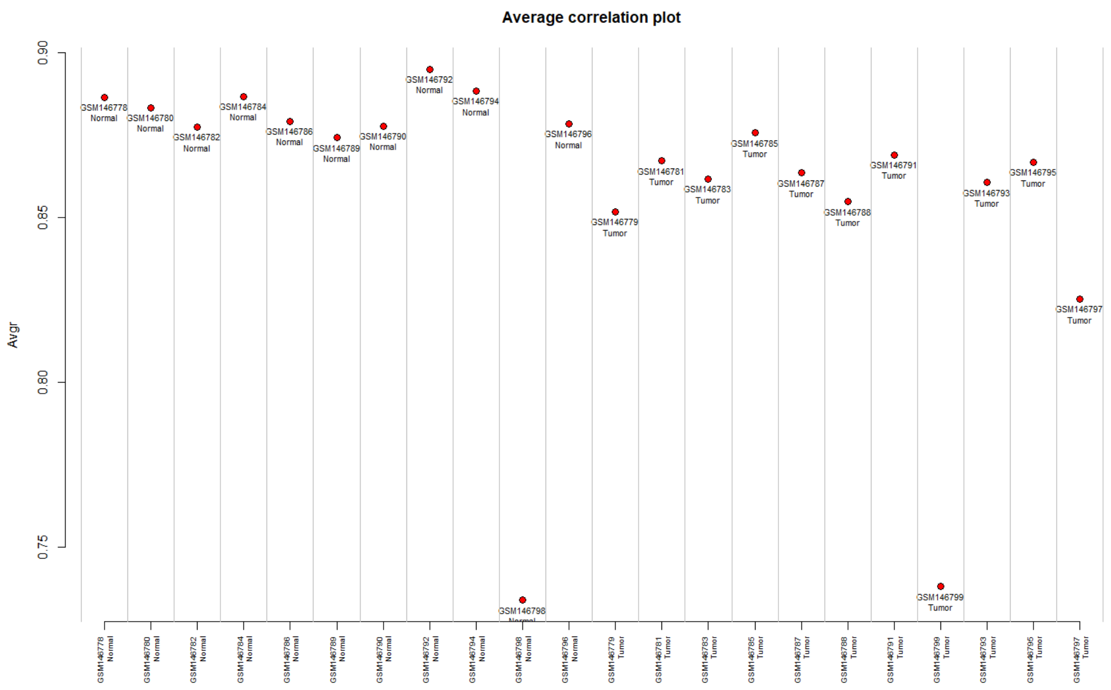
CV vs. mean plot (2pts.)

```
car.mean <- apply(log2(carcinoma),2,mean)
car.sd <- sqrt(apply(log2(carcinoma),2,var))
car.cv <- car.sd/car.mean
plot(car.mean,car.cv,main="CV vs mean plot",xlab="Mean",ylab="CV",col='blue',cex=1.5,type="n")
points(car.mean,car.cv,bg="lightblue",col=1,pch=21,cex=1.1)
text(car.mean,car.cv,label=dimnames(carcinoma)[[2]],pos=1,cex=0.7)
```



Average correlation plot (2pts.)

```
car.avg <- apply(pearsonmatrix,1,mean)
plot(c(1,length(car.avg)),range(car.avg),type="n",xlab="",ylab="Avgr",main="Average correlation plot",axes=F)
points(car.avg,bg="red",col=1,pch=21,cex=1.2)
text(car.avg,label=dimnames(carcinoma)[[2]],pos=1,cex=0.7)
axis(1,at=c(1:length(car.avg)),labels=dimnames(carcinoma)[[2]],las=2,cex.lab=0.4,cex.axis=0.6)
axis(2)
abline(v=seq(0.5,62.5,1),col="grey")
```



For all plots, make sure you label the points appropriately, title plots, and label axes. You will also need to provide a legend for the correlation plot. You can use the gplots for a color gradient, or just use the default colors.

6.) Install and load the impute library. (1pt.)

```
BiocManager::install('impute')
library(impute)
```

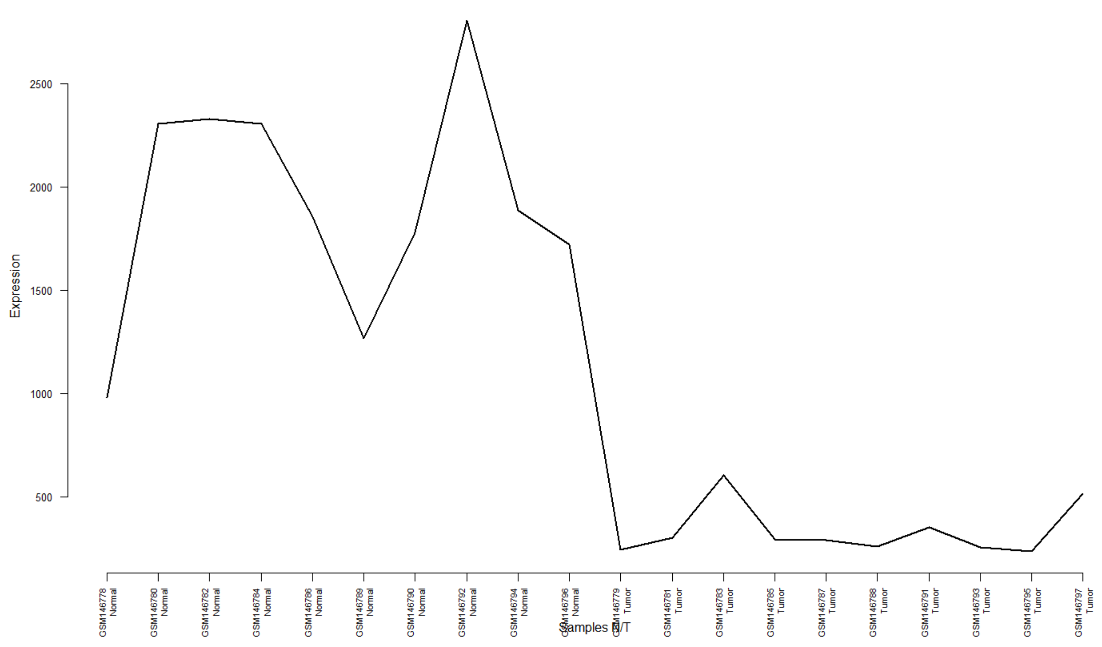
7.) Remove the outlier samples you identified in the first part of this assignment. (2pts.)

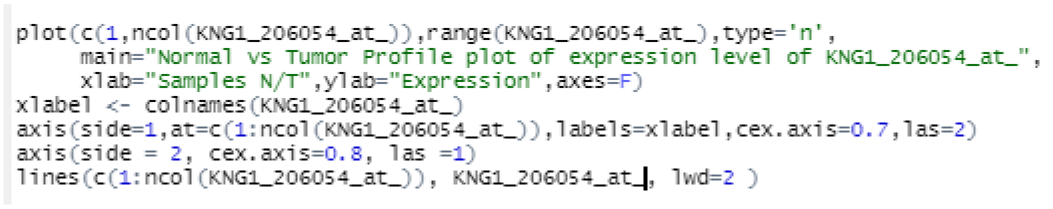
8.) Now we are going to use a couple of transcripts that were determined in this study to be indicative of normal renal function. The genes we will assess are kininogen 1 (KNG1) and aquaporin 2 (AQP2). Using either NetAffx or Gene Cards websites (or other resources, if you like), extract the probesets for these two genes. Hint: KNG1 has two while AQP2 has one. Then plot a profile plot (expression intensity vs. samples) for each probeset for these two genes. You may have to convert the data frame row to a vector to plot it. Do the plots of these genes seem to indicate normal renal function? Explain. (6pts.)

```
carcinoma_normal_tumor <- carcinoma
KNG1_206054_at_ <-carcinoma_normal_tumor[grep(pattern="206054",row.names(carcinoma_normal_tumor)),]
KNG1_217512_at <-carcinoma_normal_tumor[grep(pattern="217512",row.names(carcinoma_normal_tumor)),]
AQP2_206672_at <- carcinoma_normal_tumor[grep(pattern="206672",row.names(carcinoma_normal_tumor)),]
selected_normal <- rbind(KNG1_206054_at_,KNG1_217512_at,AQP2_206672_at)
```

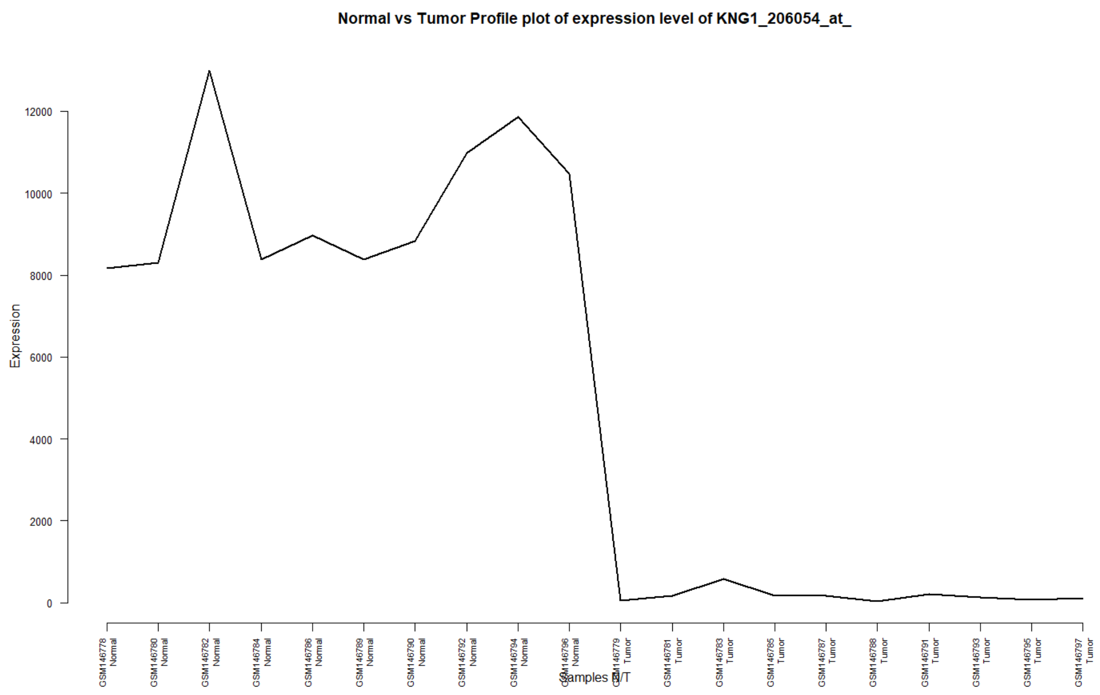
```
plot(c(1,ncol(AQP2_206672_at)),range(AQP2_206672_at),type='n',
     main="Normal vs Tumor Profile plot of expression level of AQP2_206672_at",
     xlab="Samples N/T",ylab="Expression",axes=F)
xlabel <- colnames(AQP2_206672_at)
axis(side=1,at=c(1:ncol(AQP2_206672_at)),labels=xlabel,cex.axis=0.7,las=2)
axis(side = 2, cex.axis=0.8, las =1)
lines(c(1:ncol(AQP2_206672_at)), AQP2_206672_at, lwd=2 )
```

Normal vs Tumor Profile plot of expression level of AQP2_206672_at

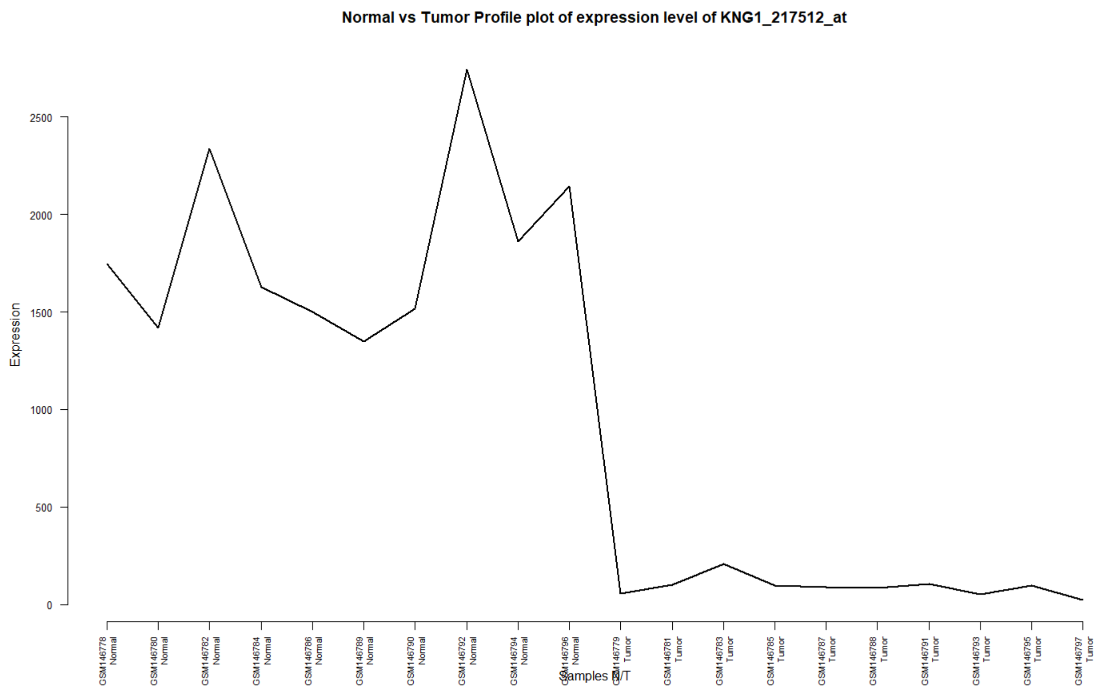




```
plot(c(1,ncol(KNG1_206054_at_)),range(KNG1_206054_at_),type='n',
     main="Normal vs Tumor Profile plot of expression level of KNG1_206054_at_",
     xlab="Samples N/T",ylab="Expression",axes=F)
xlabel <- colnames(KNG1_206054_at_)
axis(side=1,at=c(1:ncol(KNG1_206054_at_)),labels=xlabel,cex.axis=0.7,las=2)
axis(side = 2, cex.axis=0.8, las =1)
lines(c(1:ncol(KNG1_206054_at_)), KNG1_206054_at_, lwd=2 )
```

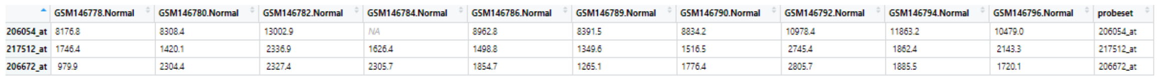


```
plot(c(1,ncol(KNG1_217512_at)),range(KNG1_217512_at),type='n',
     main="Normal vs Tumor Profile plot of expression level of KNG1_217512_at"
     xlab="Samples N/T",ylab="Expression",axes=F)
xlabel <- colnames(KNG1_217512_at)
axis(side=1,at=c(1:ncol(KNG1_217512_at)),labels=xlabel,cex.axis=0.7,las=2)
axis(side = 2, cex.axis=0.8, las =1)
lines(c(1:ncol(KNG1_217512_at)), KNG1_217512_at, lwd=2 )
```



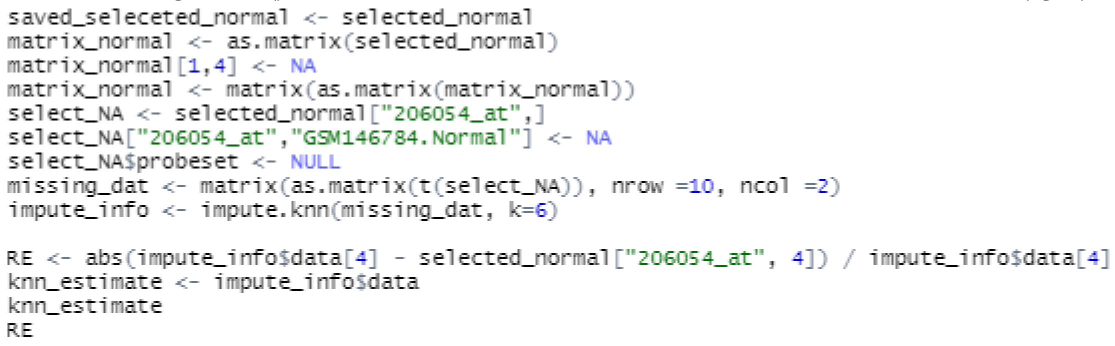
Based on the comparison of normal sample and tumor samples among the 3 probesets and 2 genes, we can see the suffifencet variantion between the normal and tumor, the KNG1 and AQP2 are all downregulated in the tumor tissue. So we can say that the result shows the data can represent the normal renal function.

9.) We want to assess the accuracy of missing value imputation. So assign the KNG1 probeset (206054_at) an NA value, only for array GSM146784. Be sure to first save the original value before replacing it with an NA. Also cast the data frame to a matrix to run this function. (2pts.)



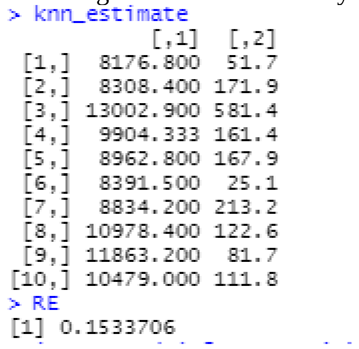
	GSM146778.Normal	GSM146780.Normal	GSM146782.Normal	GSM146784.Normal	GSM146786.Normal	GSM146789.Normal	GSM146790.Normal	GSM146792.Normal	GSM146794.Normal	GSM146796.Normal	probeset
206054_at	8176.8	8308.4	13002.9	NA	8962.8	8391.5	8834.2	10978.4	11863.2	10479.0	206054_at
217512_at	1746.4	1420.1	2336.9	1626.4	1498.8	1349.6	1516.5	2745.4	1862.4	2143.3	217512_at
206672_at	979.9	2304.4	2327.4	2305.7	1854.7	1265.1	1776.4	2805.7	1885.5	1720.1	206672_at

10.) Now estimate the missing values in the array using 6 nearest neighbors and Euclidean distance with the impute.knn() function. (2pts.)



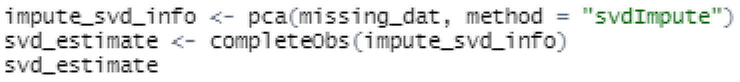
```
saved_seleceted_normal <- selected_normal
matrix_normal <- as.matrix(selected_normal)
matrix_normal[1,4] <- NA
matrix_normal <- matrix(as.matrix(matrix_normal))
select_NA <- selected_normal["206054_at",]
select_NA["206054_at","GSM146784.Normal"] <- NA
select_NA$probeset <- NULL
missing_dat <- matrix(as.matrix(t(select_NA)), nrow =10, ncol =2)
impute_info <- impute.knn(missing_dat, k=6)

RE <- abs(impute_info$data[4] - selected_normal["206054_at", 4]) / impute_info$data[4]
knn_estimate <- impute_info$data
knn_estimate
RE
```

11.) Look at the value that was imputed for your gene and calculate the relative error of this value using the actual value that you saved. (2pts.)

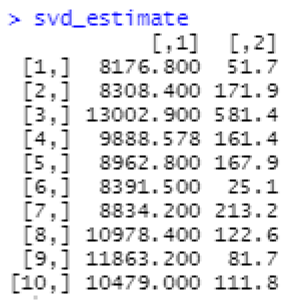


```
> knn_estimate
           [,1]  [,2]
 [1,]  8176.800  51.7
 [2,]  8308.400 171.9
 [3,] 13002.900 581.4
 [4,]  9904.333 161.4
 [5,]  8962.800 167.9
 [6,]  8391.500  25.1
 [7,]  8834.200 213.2
 [8,] 10978.400 122.6
 [9,] 11863.200  81.7
[10,] 10479.000 111.8
> RE
[1] 0.1533706
```

12.) Now impute the missing values using the SVD imputation method. This is in the pcaMethods package and the function is called pca ()with method svdImpute and set nPcs=9. To retrive the output matrix, see the help file. (2pts.)

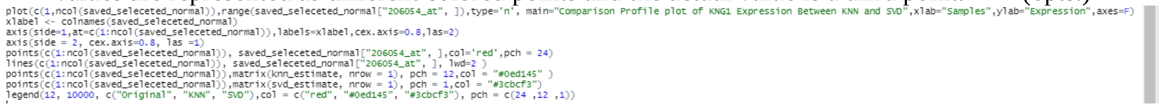


```
impute_svd_info <- pca(missing_dat, method = "svdImpute")
svd_estimate <- completeObs(impute_svd_info)
svd_estimate
```

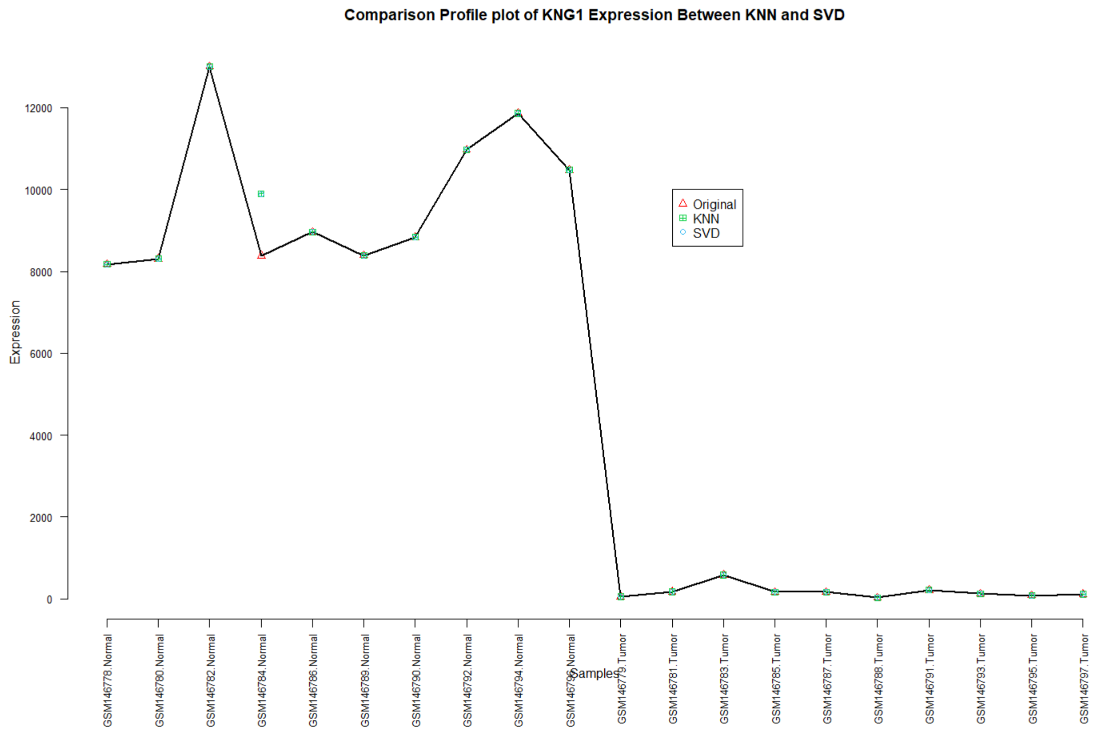


```
> svd_estimate
          [,1]  [,2]
 [1,]  8176.800  51.7
 [2,]  8308.400 171.9
 [3,] 13002.900 581.4
 [4,]  9888.578 161.4
 [5,]  8962.800 167.9
 [6,]  8391.500  25.1
 [7,]  8834.200 213.2
 [8,] 10978.400 122.6
 [9,] 11863.200  81.7
[10,] 10479.000 111.8
```

13.) Finally, plot a gene profile plot of the probeset for this gene, where the two different imputed values are represented as different colored points and the actual value is a third point. (6pts.)



```
plot(c(1,ncol(saved_seleceted_normal)),range(saved_seleceted_normal["206054_at", ]),type='n', main="Comparison Profile plot of KNG1 Expression Between KNN and SVD",xlab="Samples",ylab="Expression",axes=F)
xlabel <- colnames(saved_seleceted_normal)
axis(side=1,at=c(1:ncol(saved_seleceted_normal)),labels=xlabel,cex.axis=0.8,las=2)
axis(side = 2, cex.axis=0.8, las =1)
points(c(1:ncol(saved_seleceted_normal)), saved_seleceted_normal["206054_at", ],col='red',pch = 24)
lines(c(1:ncol(saved_seleceted_normal)), saved_seleceted_normal["206054_at", ], lwd=2 )
points(c(1:ncol(saved_seleceted_normal)),matrix(knn_estimate, nrow = 1), pch = 12,col = "#0ed145" )
points(c(1:ncol(saved_seleceted_normal)),matrix(svd_estimate, nrow = 1), pch = 1,col = "#3cbcf3")
legend(12, 10000, c("Original", "KNN", "SVD"),col = c("red", "#0ed145", "#3cbcf3"), pch = c(24 ,12 ,1))
```



Generate the code and plots for each. Turn in the visuals, code, and an explanation of the questions asked. Paste all information into a PDF doc.