Homework 1

Ryan Yancey

19 June 2021

For this homework, we will be working with a study from Gene Expression Omnibus (GEO) with the accession GDS2880 (source). 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.

ABSTRACT Although renal cell carcinoma (RCC) is the sixth-leading cause of cancer death, the molecular events leading to disease onset and progression are not well understood. Genomic profiling of clear cell RCC (cRCC) patients indicated that loss of a negative regulator of the Wnt pathway, secreted frizzled-related protein 1 (sFRP1), occurred in the majority of more than 100 patients tested. To our knowledge, this is the first report of loss of sFRP1 expression in patients diagnosed with cRCC; this loss occurs in early stage cRCC, suggesting that it may be an important early event in renal carcinogenesis. Genomic profiling of patient matched normal and cRCC tissues identified Wnt regulated genes to be aberrantly increased in cRCC tissues suggesting sFRP1 suppresses Wnt signaling in cRCC. In order to test the hypothesis that sFRP1 acts as a tumor suppressor in cRCC, we have stably expressed sFRP1 in cRCC cells. sFRP1 expression in cRCC cells resulted in decreased growth in cell culture, inhibition of anchorage-independent growth, and decreased tumor volume in a nude mouse model. Together these data suggest an important role for sFRP1 as a tumor suppressor in cRCC (source).

---

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 have 22 arrays and 22,283 probesets.

The first thing we need to do is decompress the downloaded files. In our directory, they're listed as `renal_cell_carcinoma.zip` and `renal_carcinoma_annotation.zip`.

```
# see what files are in directory
dir()
```

```
## [1] "avg-corr.tiff"                 "gedav-hw-1.pdf"
## [3] "gedav-hw1.Rproj"               "renal_carcinoma_annotation.zip"
## [5] "renal_cell_carcinoma.zip"      "ryancey3-gedav-hw1.Rmd"
```

```
# unzip both compressed files
system(command = "unzip -o renal_cell_carcinoma.zip")
system(command = "unzip -o renal_carcinoma_annotation.zip")
```

Then, we can load the resulting files into our environment.

```
dir()
```

```
## [1] "avg-corr.tiff"                   "gedav-hw-1.pdf"
## [3] "gedav-hw1.Rproj"                 "renal_carcinoma_annotation.txt"
## [5] "renal_carcinoma_annotation.zip" "renal_cell_carcinoma.txt"
## [7] "renal_cell_carcinoma.zip"        "ryancey3-gedav-hw1.Rmd"
```

```
gumz <-
  read.table(file = "renal_cell_carcinoma.txt",
             header = TRUE,
             row.names = 1)
meta <- read.table(file = "renal_carcinoma_annotation.txt")

dim(gumz)
```

```
## [1] 22283    22
```

There are 22283 probesets (rows) and 22 arrays (columns).

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

First, we need to verify the `meta` samples and the `gumz` samples are listed in the same order.

```
# verify the samples occur in the same order
colnames(gumz) == meta[, 1]
```

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

Since they already occur in the same order, we simply can iterate over the first and ninth columns in the meta dataset, paste the two together, and update the column names accordingly.

```
# update column names
colnames(gumz) <- paste(meta[, 1], meta[, 9], sep = "_")

# view updated column names
colnames(gumz)
```

```
##  [1] "GSM146778_Normal" "GSM146780_Normal" "GSM146782_Normal" "GSM146784_Normal"
##  [5] "GSM146786_Normal" "GSM146789_Normal" "GSM146790_Normal" "GSM146792_Normal"
##  [9] "GSM146794_Normal" "GSM146798_Normal" "GSM146796_Normal" "GSM146779_Tumor"
## [13] "GSM146781_Tumor"  "GSM146783_Tumor"  "GSM146785_Tumor"  "GSM146787_Tumor"
## [17] "GSM146788_Tumor"  "GSM146791_Tumor"  "GSM146799_Tumor"  "GSM146793_Tumor"
## [21] "GSM146795_Tumor"  "GSM146797_Tumor"
```

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

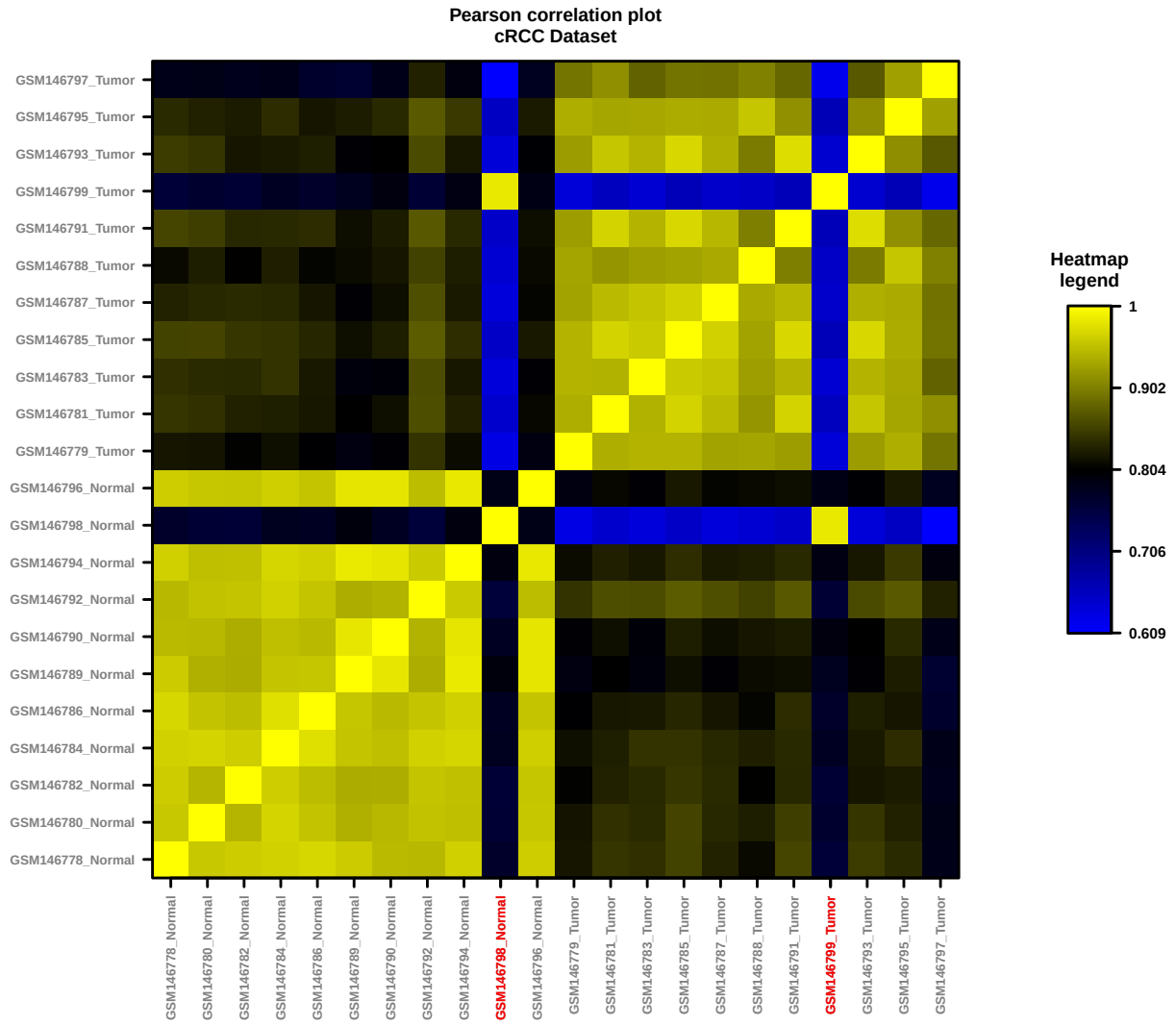
- Correlation plot (heatmap)
- Heirarchical clustering dendrogram
- CV vs. mean plot
- Average correlation plot.

For all plots, make sure you label the points appropriately, title plots, and label axes. You 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.

Correlation plot (heatmap)

```
# correlation matrix
cor.gumz <- cor(gumz)
# set custom layout of plots (heatmap and legend)
layout(matrix(c(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 2, 2, 3), 4, 4,
              byrow = FALSE), widths = c(4, 1))
# set margins for graphs
par(oma = c(4, 5, 1, 2.5))
# main plot
cols <- colorRampPalette(c("blue", "black", "yellow"))(1000)
image(cor.gumz,
      axes = FALSE,
      col = cols,
      main = "Pearson correlation plot\ncRCC Dataset")
box(lwd = 2)
axis(
  2,
  font = 2,
  at = seq(0, 1, length = ncol(cor.gumz)),
  labels = colnames(cor.gumz),
  las = 2,
  lwd = 2,
  col.axis = "gray50",
  cex.axis = 0.9
)
axis(
  1,
  font = 2,
  at = seq(0, 1, length = ncol(cor.gumz)),
  labels = colnames(cor.gumz),
  las = 2,
  lwd = 2,
  col.axis = "gray50",
  cex.axis = 0.9
)
# overlay red text to highlight potential outliers
axis(
  1,
  font = 2,
  at = seq(0, 1, length = ncol(cor.gumz))[c(10, 19)],
  labels = colnames(cor.gumz)[c(10, 19)],
  las = 2,
  lwd = 2,
  col.axis = "red",
  cex.axis = 0.9
)
# legend plot
leg <- seq(min(cor.gumz), max(cor.gumz), length = 500)
image(t(as.matrix(leg)),
      col = cols,
      axes = FALSE,
      main = "Heatmap\nlegend")
```

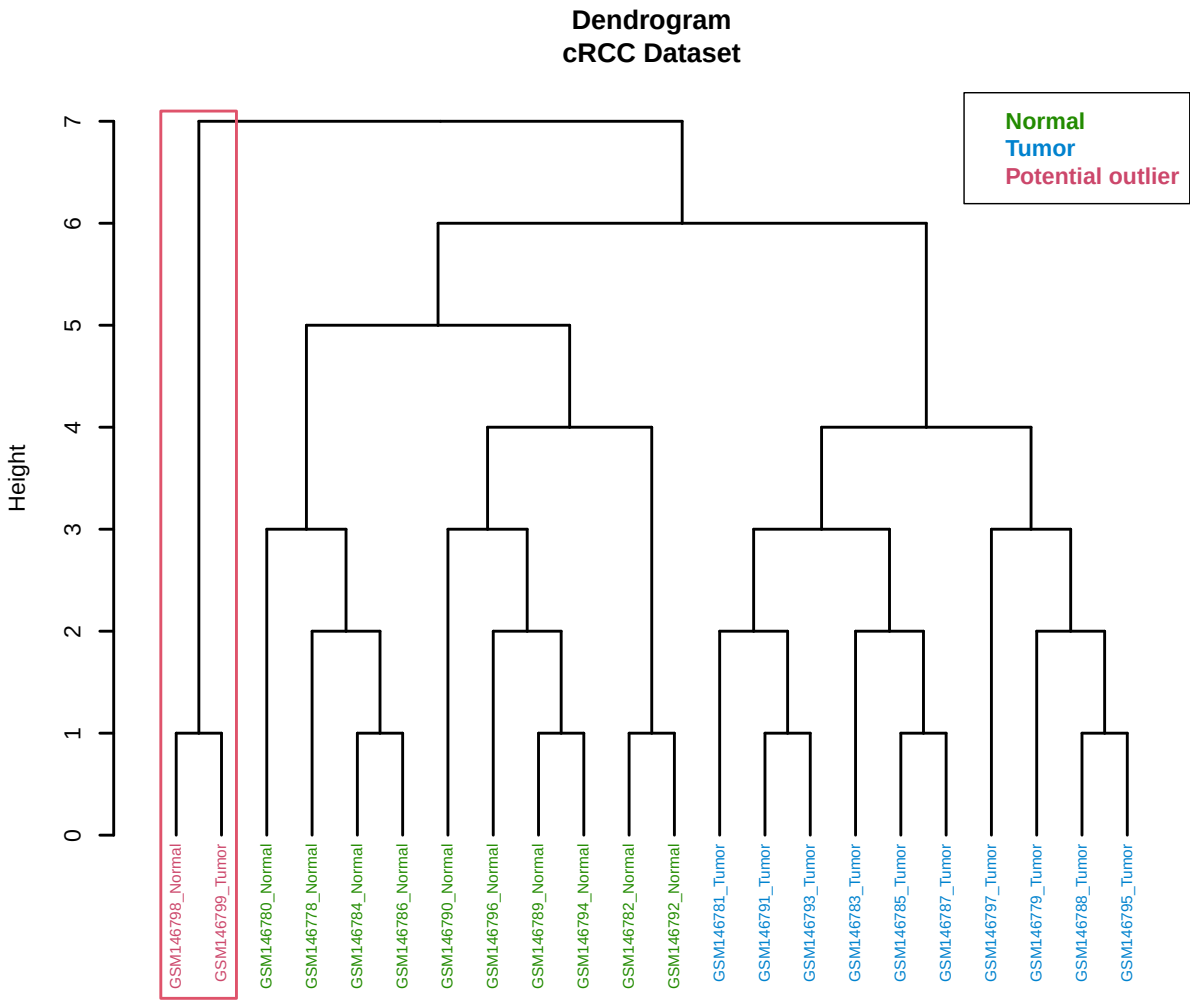
```
box(lwd = 2)
axis(
  4,
  font = 2,
  at = seq(0, 1, length = 5),
  labels = round(seq(min(cor.gumz), max(cor.gumz), length = 5), 3),
  las = 2,
  lwd = 2
)
```



It appears that `"GSM146798_Normal"` and `"GSM146799_Tumor"` (highlighted labels in red above) may be outliers to the rest of the samples.

Dendrogram

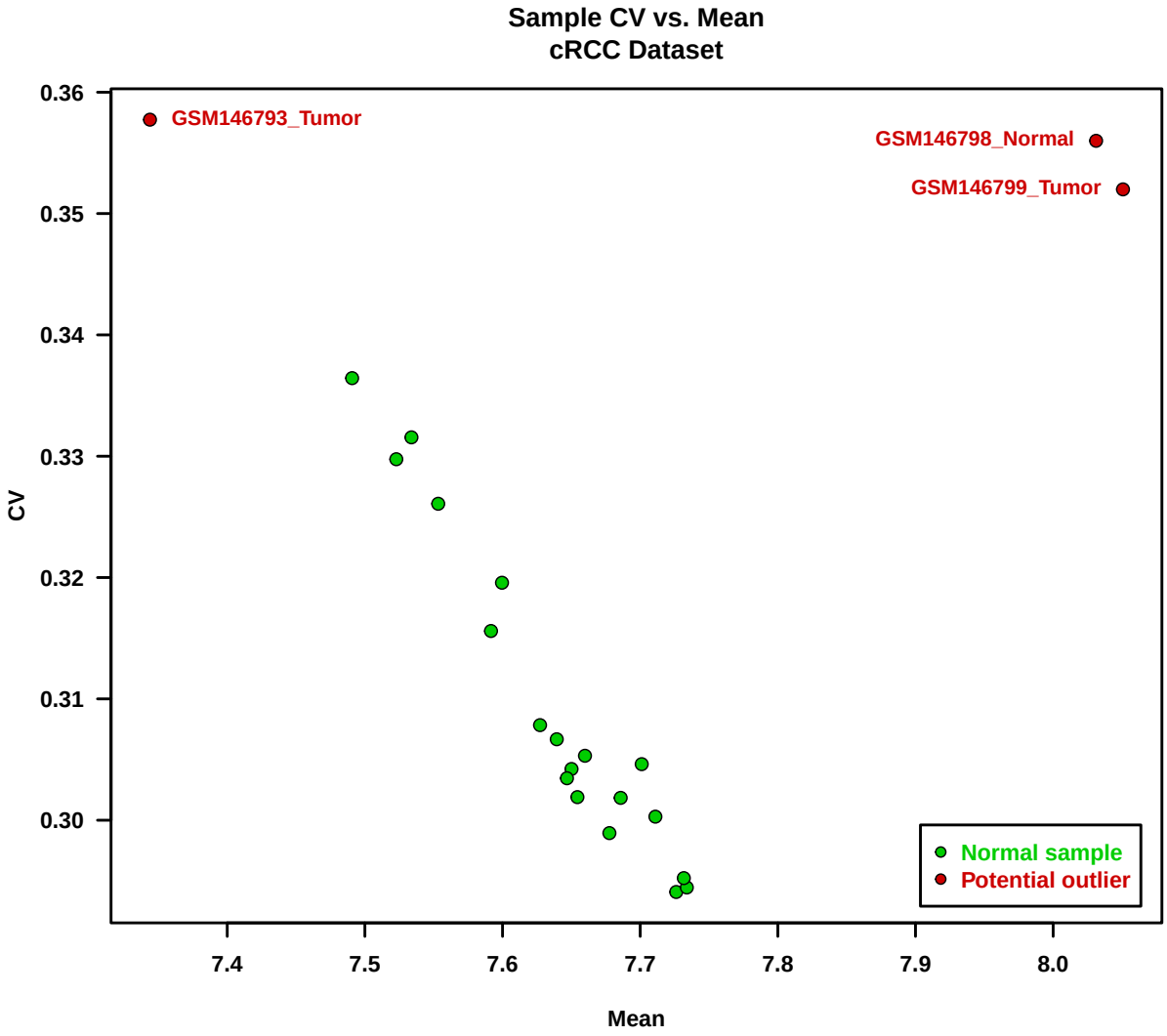
```
# install.packages("dendextend")
suppressPackageStartupMessages(library("dendextend"))
# layout(1)
par(oma = c(3, 0, 0, 0))
# dendrogram visualization stuff
dend <- hclust(dist(t(gumz), method = "euclidean")) %>%
  as.dendrogram() %>%
  color_labels(k = 3) %>%
  rank_branches() %>%
  set("branches_lwd", 2)
labels_cex(dend) <- 0.7
# plot the dendrogram
plot(dend,
     main = "Dendrogram\ncRCC Dataset",
     axes = FALSE,
     ylab = "Height")
axis(2, lwd = 2)
legend(
  "topright",
  legend = c("Normal", "Tumor", "Potential outlier"),
  text.col = c("#228B00", "#0082CE", "#CC476B"),
  text.font = 2
)
# add rectangle to highlight potential outliers
rect.dendrogram(
  dend,
  k = 3,
  which = 1,
  lower_rect = -1.6,
  upper_rect = 2.1,
  lwd = 2
)
```



`GSM146798_Normal` and `GSM146799_Tumor` still appear as outliers (red text and rectangle).

Sample CV vs. mean plot

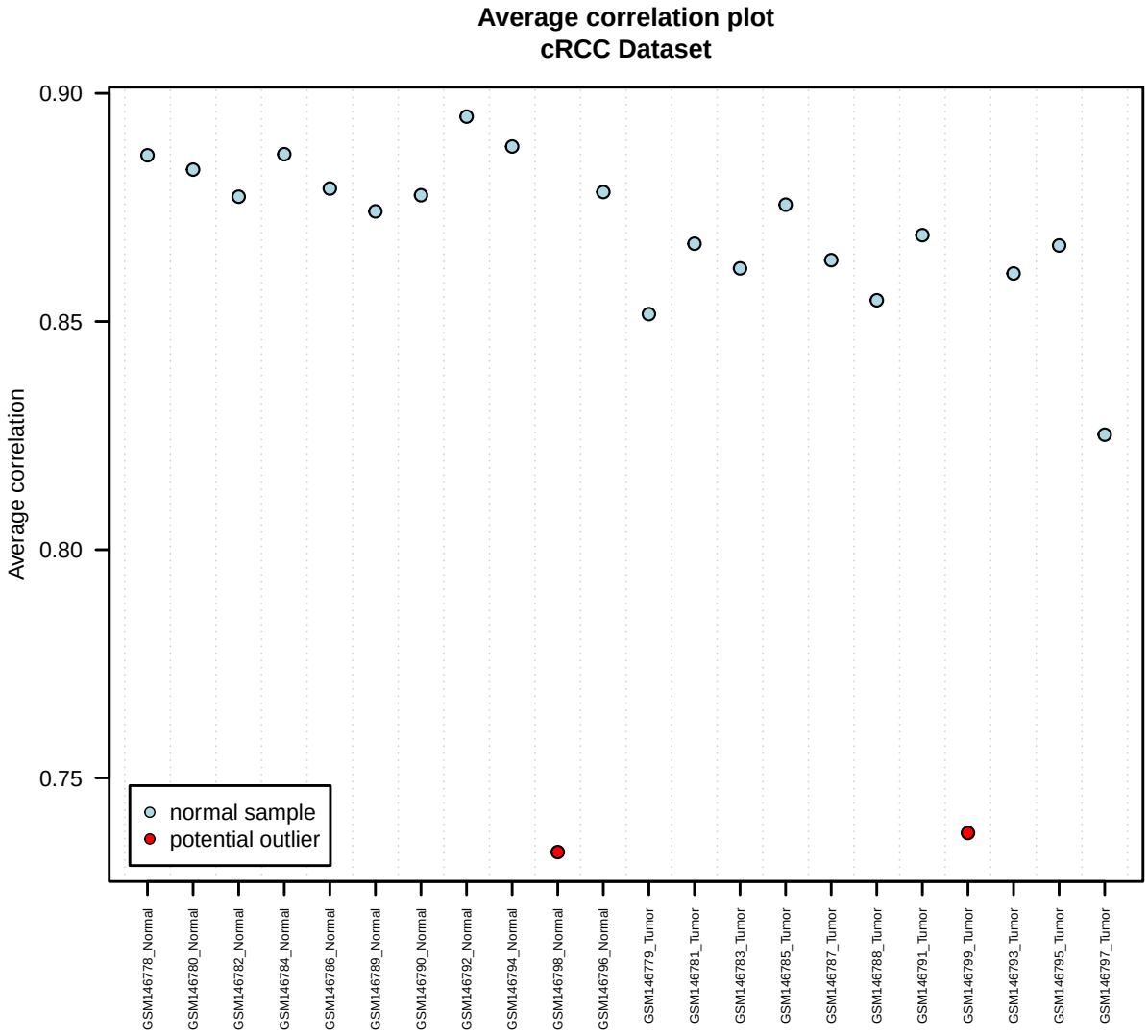
```
# calculate sample cv and mean for each sample
sample_mean <- apply(log2(gumz), 2, mean)
sample_sd <- apply(log2(gumz), 2, sd)
sample_cv <- sample_sd / sample_mean
# main plot
plot(
  y = sample_cv,
  x = sample_mean,
  ylab = "CV",
  xlab = "Mean",
  main = "Sample CV vs. Mean\ncRCC Dataset",
  type = "n",
  font.lab = 2,
  axes = FALSE
)
box(lwd = 2)
axis(1, lwd = 2, font.axis = 2)
axis(2,
     lwd = 2,
     font.axis = 2,
     las = 2)
# conditionally highlight text
text(
  y = sample_cv,
  x = sample_mean,
  offset = 0.8,
  labels = names(sample_cv),
  pos = ifelse(sample_mean < 7.4, 4, 2),
  col = ifelse(sample_cv > 0.34, "red3", "transparent"),
  cex = 0.9,
  font = 2
)
# conditionally highlight points
points(
  y = sample_cv,
  x = sample_mean,
  pch = 21,
  cex = 1.25,
  bg = ifelse(sample_cv > 0.34, "red3", "green3")
)
legend(
  "bottomright",
  legend = c("Normal sample", "Potential outlier"),
  text.col = c("green3", "red3"),
  pch = 21,
  pt.bg = c("green3", "red3"), bg = "white",
  inset = 0.02,
  box.lwd = 2,
  text.font = 2,
  title.col = "black"
)
```



`GSM146798_Normal` and `GSM146799_Tumor` are most likely candidate outliers, and it appears that `GSM146793_Tumor` might be an outlier as well, though it did not appear on the other graphs.

Average correlation plot

```
avg.corr.gumz <- apply(cor.gumz, 1, mean)
par(oma = c(1, 0, 0, 0))
# main plot
plot(
  c(1, length(avg.corr.gumz)),
  range(avg.corr.gumz),
  type = "n",
  xlab = "",
  ylab = "Average correlation",
  axes = FALSE,
  main = "Average correlation plot\ncRCC Dataset"
)
abline(
  v = seq(from = 0.5, to = 22.5, by = 1),
  col = "lightgray",
  lty = 3,
  lwd = 1.25
)
box(lwd = 2)
# conditionally plot and color points
points(
  x = avg.corr.gumz,
  pch = 21,
  bg = ifelse(avg.corr.gumz < 0.75, "red", "lightblue"),
  lwd = 1.5,
  cex = 1.25
)
axis(2, lwd = 2, las = 2)
axis(
  1,
  at = c(1:22),
  labels = colnames(gumz),
  lwd = 2,
  las = 2,
  cex.axis = 0.6
)
# legend
legend(
  "bottomleft",
  legend = c("normal sample", "potential outlier"),
  box.lwd = 2,
  pch = 21,
  pt.bg = c("lightblue", "red"), bg = "white",
  inset = 0.02
)
```



This last plot appears to confirm that `GSM146798_Normal` and `GSM146799_Tumor` are outlier samples and should be removed from the data set.

5) Install and load the impute library

```
# install.packages("impute", dependencies = TRUE)
suppressPackageStartupMessages(library("impute"))
```

6) Remove the outlier samples you identified in the first part of this assignment

The outlier samples are `GSM146798_Normal` and `GSM146799_Tumor`, so we will remove those.

```
# remove by creating a logical vector of colnames matching the outliers
col_outliers <- colnames(gumz) %in% c("GSM146798_Normal", "GSM146799_Tumor")

# remove outliers
gumz.trimmed <- gumz[, !col_outliers]

# verify the columns have been trimmed
c("GSM146798_Normal", "GSM146799_Tumor") %in% colnames(gumz.trimmed)
```

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

7) 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.

We can map the PROBEID to the alias using the `hgu133a.db` SQLite annotation of the HGU133A Affymetrix microarray probeset. The source for this method is a part of the reference manual for the `hgu133a.db` Bioconductor package (source).

```
# BiocManager::install("hgu133a.db", ask = FALSE)
suppressPackageStartupMessages(library("hgu133a.db"))

# map ALIAS (KNG1 & AQP2) to PROBEID
probes <- hgu133aALIAS2PROBE
mapped_probes <- as.list(probes[mappedkeys(probes)])

# save probes as value
(KNG1_probes <- mapped_probes[["KNG1"]])
```

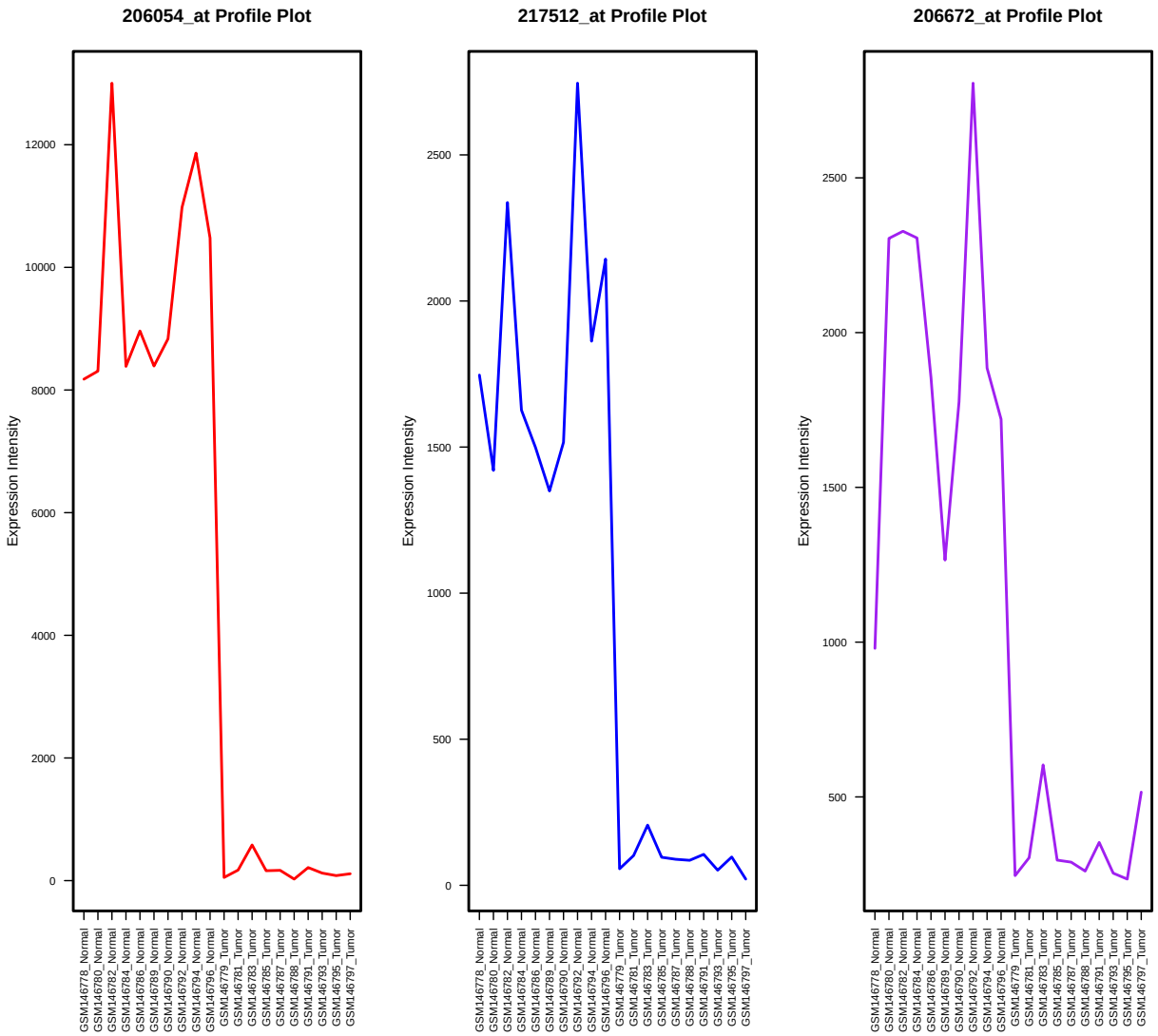
```
## [1] "206054_at" "217512_at"
```

```
(AQP2_probes <- mapped_probes[["AQP2"]])
```

```
## [1] "206672_at"
```

Now that we know which probes refer to KNG1 and AQP2, we can reference them in our data frame and plot gene profiles of each probe to observe their expression across all samples.

```
# graph parameters
par(
  lwd = 2,
  font = 2,
  las = 1,
  mfrow = c(1, 3),
  oma = c(2, 0, 0, 0),
  cex.axis = 0.75
)
# create profile_plot function -- reduces redundancy
profile_plot <- function(gene, color) {
  plot(
    x = range(1:20),
    y = range(gumz.trimmed[gene,]),
    type = "n",
    axes = FALSE,
    ylab = "Expression Intensity",
    xlab = ""
  )
  lines(x = c(1:20),
        y = gumz.trimmed[gene, ],
        col = color)
  title(main = paste(gene, "Profile Plot"))
  axis(
    1,
    at = c(1:20),
    labels = colnames(gumz.trimmed[gene, ]),
    las = 2
  )
  axis(2)
  box()
}

#### PLOT 1 of 3 ####
profile_plot(KNG1_probes[1], "red")
#### PLOT 2 of 3 ####
profile_plot(KNG1_probes[2], "blue")
#### PLOT 3 of 3 ####
profile_plot(AQP2_probes[1], "purple")
```



Yes, it appears these genes are indicative of normal renal function. In all of the tumor samples, these genes are nearly unexpressed, whereas they are expressed quite highly in normal tissue.

8) 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.

```
# store the original value
KNG1_probe1 <- gumz.trimmed["206054_at",]

# use grep because unsure of "tumor" or "normal" suffix
(GSM146784_206054_at <- gumz.trimmed["206054_at", grep("GSM146784", colnames(KNG1_probe1))])
```

```
## [1] 8385.3
```

```
# replace the stored value with NA
gumz.trimmed["206054_at", grep("GSM146784", colnames(KNG1_probe1))] <- NA

# verify the new value is NA
gumz.trimmed["206054_at", grep("GSM146784", colnames(KNG1_probe1))]
```

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

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

```
# impute using KNN method
estimate_knn <- impute.knn(as.matrix(gumz.trimmed), k = 6)

# convert to data frame and save as new table
gumz.trimmed.knn.imputed <- as.data.frame(estimate_knn[["data"]])
```

10) 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.

```
# look for imputed value
(GSM146784_206054_at.knn.imputed <-
   gumz.trimmed.knn.imputed["206054_at", grep("GSM146784", colnames(KNG1_probe1))])
```

```
## [1] 7559.533
```

```
# calculate relative error
(relative_error_knn <-
    abs(GSM146784_206054_at.knn.imputed - GSM146784_206054_at) / abs(GSM146784_206054_at))
```

```
## [1] 0.09847789
```

11) 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 retrieve the output matrix, see the help file.

```
# BiocManager::install("pcaMethods", ask = FALSE)
suppressPackageStartupMessages(library("pcaMethods"))

# impute using svd
estimate_svd <- pca(gumz.trimmed, method = "svdImpute", nPcs = 9)

# retrieve output matrix
gumz.trimmed.svd.imputed <- as.data.frame(completeObs(estimate_svd))

# look for imputed value
(GSM146784_206054_at.svd.imputed <-
    gumz.trimmed.svd.imputed["206054_at", grep("GSM146784", colnames(KNG1_probe1))])
```

```
## [1] 10418
```

```
# calculate relative error
(relative_error_svd <-
    abs(GSM146784_206054_at.svd.imputed - GSM146784_206054_at) / abs(GSM146784_206054_at))
```

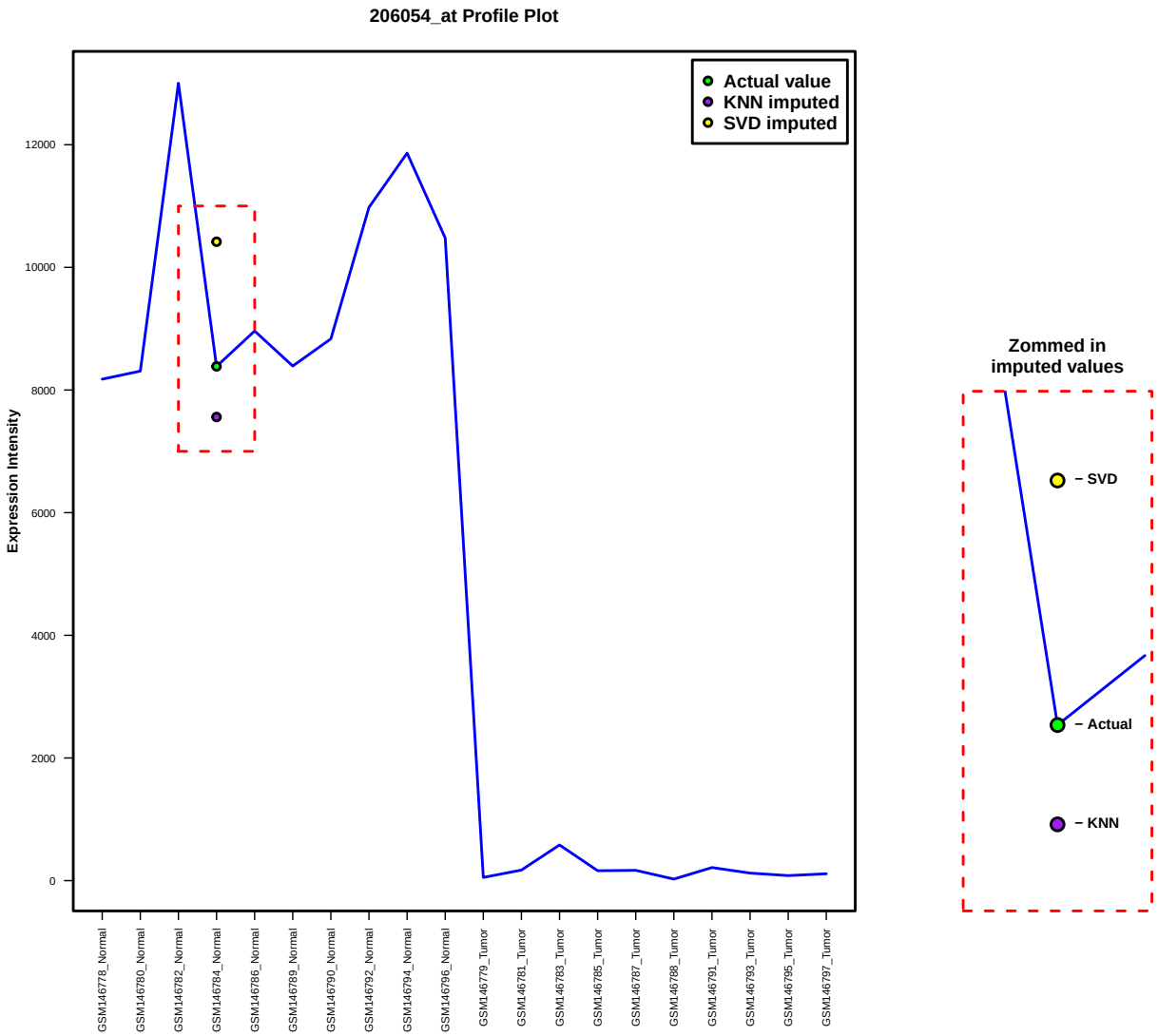
```
## [1] 0.2424125
```

12) 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.

```
# return the original value
gumz.trimmed["206054_at", grep("GSM146784", colnames(gumz.trimmed))] <- GSM146784_206054_at

# save that gene
KNG1_probe1 <- gumz.trimmed["206054_at",]

# concatenate the three values (2 imputed, 1 original)
values <-
  c(GSM146784_206054_at.knn.imputed,
    GSM146784_206054_at.svd.imputed,
    GSM146784_206054_at)
```

```r
# graph parameters
par(
  lwd = 2,
  font = 2,
  las = 1,
  oma = c(2, 0, 0, 0),
  cex.axis = 0.75
)
layout(matrix(
  c(1, 1, 1, 0, 1, 1, 1, 2, 1, 1, 1, 2),
  nrow = 3,
  ncol = 4,
  byrow = TRUE
))

# plot gene profile
plot(
  x = range(1:20),
  y = range(gumz.trimmed["206054_at",]),
  type = "n",
  axes = FALSE,
  ylab = "Expression Intensity",
  xlab = "",
  font.lab = 2
)
lines(x = c(1:20), y = gumz.trimmed["206054_at",], col = "blue")
title(main = "206054_at Profile Plot")
axis(
  1,
  at = c(1:20),
  labels = colnames(gumz.trimmed["206054_at",]),
  las = 2
)
axis(2)
box()
points(
  x = rep(grep("GSM146784", colnames(gumz.trimmed)), 3),
  y = values,
  bg = c("purple", "yellow", "green"),
  pch = 21,
  cex = 1.2
)
legend(
  "topright",
  legend = c("Actual value", "KNN imputed", "SVD imputed"),
  pch = 21,
  pt.bg = c("green", "purple", "yellow"),
  inset = 0.01,
  cex = 1.2
)

# rectangle to display zoomed in region
rect(3, 7000, 5, 11000, border = "red", lty = 2)
```

```
# plot zoomed in region
plot(
  x = range(3:5),
  y = range(7000:11000),
  type = "n",
  axes = FALSE,
  ylab = "", xlab = "",
  main = "Zommed in\nimputed values"
)
lines(x = c(3:5), y = gumz.trimmed["206054_at", c(3:5)], col = "blue", lwd = 2)
box(col = "red", lty = 2)
points(
  x = rep(grep("GSM146784", colnames(gumz.trimmed)), length(values)),
  y = values,
  bg = c("purple", "yellow", "green"),
  pch = 21,
  cex = 2
)
text(
  x = rep(grep("GSM146784", colnames(gumz.trimmed)), length(values)),
  y = values,
  labels = c("- KNN", "- SVD", "- Actual"),
  pos = 4,
  offset = 1
)
```



It appears that the KNN imputation is closest to the original point, and the SVD imputation appears to overestimate what the original point was. This is also apparent with the two relative errors. KNN relative error $\approx 9.85\%$ whereas the SVD relative error $\approx 24.24\%$, or nearly 2.5X higher.