```
# Appendix

getwd()

#QUESION 1

  # Read in data to data frame. Header row confirm, first col names confirm
    assign1 <- read.table("assignOct1.txt", header=TRUE, row.names=1)

  # Display first rows of df - check read in
    head(assign1)

  # display df dimensions
    dim(assign1)

  # categorical variables

    table(assign1$Gender)      # Gender
    table(assign1$Response)    # Response
    table(assign1$Histology)   # Histology
    table(assign1$Differentiation)  # Differentiation
    table(assign1$ProteinA)  # Protein A
    table(assign1$PositiveNodes)    # PositiveNodes
    table(assign1$Event)       # Events

    table(assign1$Gender,assign1$Event)      # Gender (+ events '1' vs non-events '0')
    table(assign1$Response,assign1$Event)    # Response (+ events)
    table(assign1$Histology,assign1$Event)   # Histology (+ events)
    table(assign1$Differentiation,assign1$Event)  # Differentiation (+ events)
    table(assign1$ProteinA,assign1$Event)          # Protein A (+ events)
    table(assign1$PositiveNodes,assign1$Event)     # PositiveNodes   (+ events)
          summary(assign1$PositiveNodes) #PosNodes NAs

        # Vector to store number of positive nodes
          num_nodes <- c(1, 2, 3, 4, 5, 6, 7, 8, 9, "10 or greater", "NA")

        # Quant of positive nodes
          node_counts <- table(ifelse(is.na(assign1$PositiveNodes), "NA",
ifelse(assign1$PositiveNodes >= 10, "10 or greater", assign1$PositiveNodes)))

        # Create table + print
          node_table <- data.frame(Number_of_nodes = num_nodes, Count = node_counts)

          print(node_table)

  # continuous variables
    summary(assign1$Length)      #Length Median, Range, NAs
    summary(assign1$Width)       #Width
    summary(assign1$Vol)         #Volume
    summary(assign1$Age)         #Age

  # Median survival
    library(survival)
    fit <- survfit(Surv(assign1$Survival, assign1$Event) ~ 1)
    print(fit)


# QUESTION 2

  # part a : Protein A levels linked to age or sex

    # protein a v sex
      proteinA.vs.Sex<- table(assign1$Gender,assign1$ProteinA)
      proteinA.vs.Sex
      # table
        chisq.test(proteinA.vs.Sex)
          ## output = (p-value) 0.8932 therefore non-significant
        fisher.test(proteinA.vs.Sex)
          ## output = (p-value) 0.8176 therefore non-significant
```

```
      # protein a v age
         lm.fit <- lm(assign1$Age ~ assign1$ProteinA, data = assign1)
      # display results
         summary(lm.fit)
          ## output = Multiple R-squared:  0.00142 (0.14% of age variation explained by
Protein A level), p-value: 0.7466 (no significant relationship level & age)


  # part b : univariate non-parametric method

    # load package
      library(survival)

    # create survfit object & Kaplan-Meier curves for Protein A levels
      KM.Test <- survfit(Surv(assign1$Survival,
assign1$Event)~assign1$ProteinA,data=assign1)

    # perform a log-rank test
      survdiff(Surv(assign1$Survival, assign1$Event)~assign1$ProteinA,data=assign1)
      ## log-rank output: Chisq= 3.9  on 1 degrees of freedom, p= 0.05

    # create plot of the Kaplan-Meier
      plot(KM.Test, main="Protein A levels as a Predictor of Overall Survivability",
col.main="black",
           xlab="Time (Days)", ylab="Overall Survival Probability",
           col.lab="blue", cex.lab=0.9,col=c("red","blue"), lty = 2:3)
      legend(2500, 1.0, title="Legend",c("Low","High"),
             lty = 2:3,col=c("red","blue"),cex=0.7)
      # add legend to plot
        legend(2300, .82, c("p-value: 5.29e-13"), cex=0.8,box.col="white")

# QUESTION 3

    #Stat test
      # Read in data
        assign2 <- read.table("assignOct2.txt", header=TRUE, row.names=1)
        #check
          head(assign2)
        # display df dimensions
          dim(assign2)
      # sort by Sample ID
        sort1<-assign1[sort(row.names(assign1)),]
        sort2<-assign2[sort(row.names(assign2)),]
      # Wilcoxon rank sum test for each gene in assign2
        p_values <- numeric(length = ncol(assign2))
        for (i in seq_along(p_values)) {
          p_values[i] <- wilcox.test(sort2[,i] ~ sort1[,4])$p.value
        }
      # create a data frame to store results
        results <- data.frame(Gene = colnames(assign2), P_value = p_values)
      # sort the results by p-value
        results_sorted <- results[order(results$P_value),]
      # view for excel
        View(results_sorted)

    #boxplots
      #standard individual
        # create a vector of colors for the two groups
          colors <- c("blue", "red")
        # create the boxplot with customized colors
          boxplot(assign2$GeneA1 ~ assign1$ProteinA, data=assignOct2,
                  xlab="Protein A", ylab="Gene A1 expression",
                  main="Boxplot of GeneA1 expression by Protein A level",
                  col=colors)

      #combined

        # load package
```

```
          library(ggplot2)
        # create a data frame with the relevant columns
          df <- data.frame(
            GeneA = rep(paste0("GeneA", 1:14), each = length(assign1$ProteinA)),
            ProteinA = rep(assign1$ProteinA, times = 14),
            Expression = c(assign2$GeneA1, assign2$GeneA2, assign2$GeneA3,
                           assign2$GeneA4, assign2$GeneA5, assign2$GeneA6,
                           assign2$GeneA7, assign2$GeneA8, assign2$GeneA9,
                           assign2$GeneA10, assign2$GeneA11, assign2$GeneA12,
                           assign2$GeneA13, assign2$GeneA14)
          )
        # create the boxplot using ggplot2
          ggplot(df, aes(x = ProteinA, y = Expression, fill = ProteinA)) +
            geom_boxplot(size = 0.5) +
            scale_fill_manual(values = c("blue", "red")) +
            labs(x = "Protein A", y = "Gene expression",
                 title = "Boxplot of Gene expression by Protein A level") +
            facet_wrap(~ GeneA, ncol = 4)


# QUESTION 4

      #part a

        #cor matrix data
          # extract gene expression columns
            genes<- assign2[, 2:14]
          # create correlation matrix
            cor_matrix <- cor(genes)
          # identify genes with high correlation coefficients
            high_corr_genes <- which(cor_matrix > 0.8 & cor_matrix < 1, arr.ind=TRUE)
          # remove duplicate gene pairs
            high_corr_genes <- high_corr_genes[!duplicated(t(apply(high_corr_genes, 1,
sort))),]
          # print the pairs of highly correlated genes and their correlation
coefficients
            cat("Pairs of highly correlated genes (correlation coefficient > 0.8):\n")
            for (i in 1:nrow(high_corr_genes)) {
              gene1 <- colnames(genes)[high_corr_genes[i,1]]
              gene2 <- colnames(genes)[high_corr_genes[i,2]]
              corr <- cor_matrix[high_corr_genes[i,1], high_corr_genes[i,2]]
              cat(gene1, "and", gene2, "with correlation coefficient", round(corr, 2),
"\n")
               }

        #heatmap visual
          # extract gene expression columns
            genes<- assign2[, 2:14]
          # create correlation matrix
            cor_matrix <- cor(genes)
          # remove duplicates
            cor_matrix[lower.tri(cor_matrix)] <- NA
          # create heatmap
            heatmap(cor_matrix, Rowv=NA, Colv=NA, col = cm.colors(256), scale="none")
            # legend
              legend("topleft",
                     legend = c("Low","Medium", "High "),
                     fill = c("white","cyan", "magenta"),
                     title = "Correlation",
                     cex = 0.8)

          #dendrogram visual
            # extract gene expression columns
              genes <- assign2[, 2:14]
            # create correlation matrix
              cor_matrix <- cor(genes)
            # create distance matrix
              dist_matrix <- 1 - cor_matrix
            # hierarchical clustering
```

```
              hc <- hclust(as.dist(dist_matrix), method="ward.D2")
            # plot dendrogram
              plot(hc, main="Dendrogram of Gene Expression Data", xlab="Genes", sub="",
ylab="Distance")


        #part b
            # load package
              library(pwr)
            # calculate effect size (d) using the means and standard deviations of
three groups
            n <- pwr.t.test(n = NULL,
                            d = abs(diff(c(mean(assign2$GeneA1[assign2$GeneA1<0]),
                                           mean(assign2$GeneA2[assign2$GeneA2<0]),

mean(assign2$GeneA9[assign2$GeneA9<0])))/sd(c(assign2$GeneA1[assign2$GeneA1<0],

assign2$GeneA2[assign2$GeneA2<0],

assign2$GeneA9[assign2$GeneA9<0]))))),
            # min sample size for 80% power at 5% sig lvl
              sig.level = 0.05,
              power = 0.8,
              type = "one.sample")
            # print to the nearest integer
              cat("Minimum sample size required for 80% power:", ceiling(n))



# QUESTION 5

      # part a

          # read in
             df.5a <- assign2
            # remove missing values
              df.5a <- na.omit(assign2)
            # standardise
              df.5a <- scale(df.5a)
            # check
              head(df.5a)

          # dendrogram
            # compute dissimilarity matrix using Euclidean distance
              dist_matrix <- dist(df.5a, method = "euclidean")
            # Hierarchical clustering using Ward d2
              cluster_model <- hclust(dist_matrix, method = "ward.D2" )
            # plot dendrogram
              plot(cluster_model, cex = 0.6, hang = -1)

          # estimate  optimal number of clusters
            # gap statistic method
              library("factoextra")
              set.seed(123)
              fviz_nbclust(df.5a, kmeans, nstart = 25,  method = "gap_stat", nboot =
500)+
                labs(subtitle = "Gap statistic method")
                ## output 2
            # silhouette method
              fviz_nbclust(df.5a, FUN = hcut, method = "silhouette")+
                labs(subtitle = "Silhouette method")
                ## output 2

           # allocate clusters
              # cut tree into 2 clusters
              cluster_labels <- cutree(cluster_model, k = 2)
              # members of each cluster
                table(cluster_labels)
              # plot dendrogram
```

```
          plot(cluster_model, cex = 0.6, hang = -1)
        # draw cluster border
          rect.hclust(cluster_model, k = 2, border = 2:5)

      # cluster scatter plot
        fviz_cluster(list(data = df.5a, cluster_labels = sub_grp))



# part b

      # contingency table for cluster labels and differentiation
        cont_table_dif <- table(cluster_labels, assign1$Differentiation)
        # perform chi-squared test
          chi_dif <- chisq.test(cont_table_dif)
          # results
             chi_dif
        # fishers
          fisher_dif <- fisher.test(cont_table_dif)
           # results
              fisher_dif


      # contingency table for cluster labels and histology
        cont_table_his <- table(cluster_labels, assign1$Histology)
        # perform chi-squared test
          chi_his <- chisq.test(cont_table_his)
          # view the results
            chi_his
        # fishers
          fisher_his <- fisher.test(cont_table_his)
          # results
            fisher_his

# part c

            # Kaplan-Meier survival analysis
              library(survival)

            # create survival object
              surv_obj <- Surv(time = assign1$Survival, event = assign1$Event)

            # create dataframe with  cluster labels & survival object
              cluster_surv <- data.frame(cluster = cluster_labels, surv_obj)

            # plot Kaplan-Meier survival curves for each cluster
              ggsurvplot(survfit(surv_obj ~ cluster, data = cluster_surv),
                         pval = TRUE,
                         legend.title = "Legend",
                         xlab = "Time (days)",
                         ylab = "Survival probability")

            # log-rank test to compare the survival curves between the clusters
              survdiff(surv_obj ~ cluster, data = cluster_surv)
                ## output: Chisq= 0.4  on 1 degrees of freedom, p= 0.6


            # cox
              # perform Cox proportional hazards model
                cox_model <- coxph(surv_obj ~ cluster_surv$cluster)
              # print summary of Cox proportional hazards model
                summary(cox_model)
                ## output: p = 0.552, concordanc= 0.534, wald + liklihood +
score all = 0.6
```