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

# Brian Weinstein - bmw2148
# STAT W4201 001
# Homework 3
# 2016-02-17

# set working directory
setwd("~/Documents/advanced-data-analysis/homework_03")

# load packages
library(dplyr)
library(Sleuth3) # Data sets from Ramsey and Schafer's "Statistical Sleuth
(3rd ed)"
library(ggplot2); theme_set(theme_bw())
library(data.table)



# Problem 1: Ramsey 4.30
#######################################################################

# load data
spfData <- Sleuth3::ex0430

# create spfEstimate column
spfData <- spfData %>%
  mutate(spfEstimate=Sunscreen/PreTreatment)

ggplot(spfData, aes(x="", y=spfEstimate)) +
  geom_boxplot() +
  labs(y="SPF Estimate (Sunscreen / PreTreatment)", x="")
ggsave(filename="writeup/1.png", width=6.125, height=3.5, units="in")

# Define a fn to calcuate a conf interval for a given mean, std error, df, and
conf level
MakeConfidenceIntervalForMean <- function(mean, se, df, confLevel){

  sigLevel <- 1 - confLevel

  error <- qt(p=1-(sigLevel/2), df=df) * se

  lowerBound <- mean - error
  upperBound <- mean + error

  confidenceInterval <- c(lowerBound, upperBound)

  return(confidenceInterval)

}

# define variables for this problem
mean_value <- mean(spfData$spfEstimate); mean_value
```

```
std_err <- sd(spfData$spfEstimate)/sqrt(nrow(spfData)); std_err
df_value <- nrow(spfData) - 1; df_value

# 95% and 90% confidence intervals for (mu_2 - mu_1)
MakeConfidenceIntervalForMean(mean=mean_value, se=std_err, df=df_value,
confLevel=0.95)

rm(list = ls()) # clear working environment



# Problem 2: Ramsey 4.32
########################################################################

# load data and compute treatment differences
marData <- Sleuth3::ex0432 %>% mutate(diff=Marijuana-Placebo)
marData

# plot diffs
ggplot(marData, aes(x="", y=diff)) +
  geom_boxplot() +
  labs(y="Difference in the number of retching episodes\n(Marijuana -
Placebo)", x="")
ggsave(filename="writeup/2.png", width=6.125, height=3.5, units="in")

# perform sign test on hypothesis that Marijuana-Placebo < 0

numObs <- sum(marData$diff != 0) # number of nonzero observations
numPosDiffs <- sum(marData$diff > 0) # compute the number of positive
differences

zStat <- (numPosDiffs - (numObs/2)) / sqrt(numObs/4) # compute the Z statistic
zStat

# compute an estimated one-sided p-value
pnorm(-1 * abs(zStat), mean=0, sd=1)

# Compute a 95% CI and find an estimate for the additive treatment effect
# test several hypothesized values for the treatment effect
# and find the smallest and largest deltas that lead to a
# two-sided pvalue >= 0.05. Those are the endpoints of a
# 95% CI, with the midpoint as the estimate for delta.

# initialize an empty list
deltaPvalList <- list()

# get 2 sided pvalues for various hypothetical deltas
for(i in 1:50){

  delta <- i
  marDataNew <- marData %>%
    mutate(MarijuanaNew=Marijuana+delta,
           diffNew=MarijuanaNew-Placebo)

  numObs <- sum(marDataNew$diffNew != 0)
```

```
  numPosDiffs <- sum(marDataNew$diffNew > 0)
  zStat <- (numPosDiffs - (numObs/2)) / sqrt(numObs/4)
  pval <- 2 * pnorm(q=(-1 * abs(zStat)), mean=0, sd=1) # 2-sided p-value

  deltaPvalList[[i]] <- data.frame(delta=delta, pval=pval)

}

deltaPvalList <- rbindlist(deltaPvalList) %>% as.data.frame()

# find the min and max deltas that lead to a two-sided p-value >= 0.05
confInt <- deltaPvalList %>% filter(pval >= 0.05) %>% select(delta) %>%
range()
confInt
mean(confInt)

rm(list = ls()) # clear working environment



# Problem 3: Ramsey 5.19
########################################################################

# input data
cavityData <- data.frame(n=c(127, 44, 24, 41, 18, 16, 11, 7, 6),
                         logMean=c(7.347, 7.368, 7.418, 7.487, 7.563, 7.568,
8.214, 8.272, 8.297),
                         logSampleSd=c(0.4979, 0.4235, 0.3955, 0.3183, 0.3111,
0.4649, 0.2963, 0.3242, 0.5842))

# Part a ### ### ### ### ### ### ### ### ### ### ### ### ### ### ### ### ###
###

# pooled estimate of variance
pooledVar <- cavityData %>%
  mutate(numeratorTerm=((n-1)*logSampleSd^2),
         denominatorTerm=(n-1)) %>%
  summarize(numerator=sum(numeratorTerm),
            denomintor=sum(denominatorTerm))
pooledVar <- pooledVar$numerator / pooledVar$denomintor
pooledVar

# Part b ### ### ### ### ### ### ### ### ### ### ### ### ### ### ### ### ###
###

# sum of squares within
ssw <- cavityData %>%
  mutate(term=((n-1)*logSampleSd^2)) %>%
  summarize(sumOfSquaresWithin=sum(term)) %>%
  as.numeric()
ssw

# total sum of squares
sst <- (sum(cavityData$n) - 1) * (0.4962)^2
sst
```

```
# sum of squares between
ssb <- sst - ssw
ssb

# compute mean squares
msb <- ssb / 8 ; msb
msw <- ssw / 285 ; msw

# compute the F-statistic
fstat <- msb/msw ; fstat

# p-value of F-statistic
fstat
pf(q=fstat, df1=8, df2=285, lower.tail=FALSE)

# Part c ### ### ### ### ### ### ### ### ### ### ### ### ### ### ### ### ###
###

# compute the overall mean using a weight average of the group means
overallLogMean <- sum(cavityData$n * cavityData$logMean) / sum(cavityData$n)
overallLogMean

# recomopute the sum of squares between
sum((cavityData$n * (cavityData$logMean)^2 )) - (sum(cavityData$n) *
overallLogMean^2)

rm(list = ls()[ls() != "cavityData"]) # clear working environment, excluding
cavityData

# Part d ### ### ### ### ### ### ### ### ### ### ### ### ### ### ### ### ###
###

# compute group means, pooled SD, and RSS for Group A and Group B
cavityDataTwoGroups <- cavityData %>%
  mutate(group=c("A", "A", "A", "A", "A", "A", "B", "B", "B"),
         tempVal_mean=n*logMean,
         tempVal_nmin1=(n-1),
         tempVal_var=((n-1)*logSampleSd^2)) %>%
  group_by(group) %>%
  summarise(n=sum(n),
            tempVal_mean=sum(tempVal_mean),
            tempVal_nmin1=sum(tempVal_nmin1),
            tempVal_var=sum(tempVal_var)) %>%
  mutate(mean=tempVal_mean/n,
         pooled_sd=sqrt(tempVal_var/tempVal_nmin1)) %>%
  select(group, n, mean, pooled_sd) %>%
  mutate(rss=((n-1)*(pooled_sd)^2)) %>%
  as.data.frame()
cavityDataTwoGroups

# sum of squares within for the two-group model
ssw <- sum(cavityDataTwoGroups$rss)
ssw
```

```
# total sum of squares
sst <- (sum(cavityData$n) - 1) * (0.4962)^2
sst

# sum of squares between
ssb <- sst - ssw
ssb

# compute mean squares
msb <- ssb / 1 ; msb
msw <- ssw / 292 ; msw

# compute the F-statistic
fstat <- msb/msw ; fstat

# p-value of F-statistic
fstat
pf(q=fstat, df1=8, df2=285, lower.tail=FALSE)

rm(list = ls()) # clear working environment



# Problem 4
########################################################################

# load data
sparrowData <- Sleuth3::ex0221

rm(list = ls()) # clear working environment



# Problem 5
########################################################################

# no code needed



# Problem 6
########################################################################

# Define a function to compute the power of a test
ComputePower <- function(mu, sigma, n1, n2){

  tStat <- (0-mu) / (sigma * sqrt((1/n1) + (1/n2)))
  power <- 1 - pt(q=(qt(p=0.95, df=(n1+n2-2)) - abs(tStat)),
                  df=(n1+n2-2)) ; power

}

ComputePower(mu=0, sigma=1, n1=10, n2=10) # should be 0.05

# define list of mu values to test
```

```
muvals <- c(0.1, 0.5, 1, 2)

# initialize an empty list to store power
powerList <- list()

# test each mu value at sample sizes 10 and 20
for(i in 1:length(muvals)){

  mu_value <- muvals[i]
  tempPowerList <- rbind(data.frame(mu_value=mu_value,
                                    sample_size=10,
                                    power=ComputePower(mu=mu_value, sigma=1,
n1=10, n2=10)),
                         data.frame(mu_value=mu_value,
                                    sample_size=20,
                                    power=ComputePower(mu=mu_value, sigma=1,
n1=20, n2=20)))

  powerList[[i]] <- tempPowerList
  rm(tempPowerList)

}

# put results into dataframe
powerList <- rbindlist(powerList) %>%
  mutate(sample_size=as.factor(sample_size))

# plot
ggplot(powerList, aes(x=mu_value, y=power, color=sample_size)) +
  geom_line(aes(linetype=sample_size), size=1) +
  geom_point(aes(shape=sample_size), size=2.5) +
  xlim(0,2) + ylim(0,1)
ggsave(filename="writeup/6.png", width=7, height=3.5, units="in")

rm(list = ls()) # clear working environment



# Problem 7
#######################################################################

set.seed(1)
sigma <- 1

# initialize an empty list to store results
pvalList <- list()

for(i in 1:20000){

  # generate random samples
  nx <- 10
  x.vals <- rnorm(n=nx, mean=0, sd=sigma)
  ny <- 10
  y.vals <- rnorm(n=10, mean=0, sd=sigma)
```

```
  # calculate difference in means
  diffMean <- mean(x.vals) - mean(y.vals)

  # calculate the pooled sd
  sp <- sqrt( ((nx-1)*(sd(x.vals)^2) + (ny-1)*(sd(y.vals)^2)) / (nx + ny -2) )

  # calculate the t statistic and 2-sided p value
  tStat <- diffMean / (sp * sqrt((1/nx) + (1/ny)))
  pval <- 2 * pt(q=-1 * abs(tStat), df=(nx + ny - 2)) # 2-sided pvalue

  pvalList[[i]] <- data.frame(pval)

}

pvalList <- rbindlist(pvalList)

# plot a histogram of 2-sided pvalues
ggplot(pvalList, aes(x=pval)) +
  geom_histogram(binwidth=0.01) +
  labs(title=paste0("Distribution of n=", nrow(pvalList), " Two-Sided p-
Values"))
ggsave(filename="writeup/7.png", width=7, height=3.5, units="in")

rm(list = ls()) # clear working environment
```