Problem Set 2

Ian Sims

January 23, 2019

Load packages:

```
library(ggplot2)
library(lattice)
library(gapminder)
library(GGally)
library(NHANES)
library(broom)
library(BSDA)
library(coin)
library(reshape2)
library(gtools)
library(dplyr)
library(data.table)
```

Problem #1

The data in Table 1 are simulated exam scores. Suppose the exam was given in the semester after the course content was revised, and the previous median exam score was 70.

79 74 88 80 80 66 65 86 84 80 78 72 71 74 86 96 77 81 76 80 76 75 78 87 87 74 85 84 76 77 76 74 85 74 76 77 76 74 81 76

(a) We would like to know whether or not the median score has increased. Answer the question by applying the binomial test.

```
#Let p be the probablity that an observation is greater than 70
#Ho: p = 0.5, m = 70
#H1: p > 0.5, m > 70
dd = c(79,74,88,80,80,66,65,86,84,80,78,72,71,74,86,96,77,81,76,80,76,75,78,8
7,87,74,85,84,76,77,76,74,85,74,76,77,76,74,81,76)
pvalue = 1 - pbinom(length(dd[dd > 70]) - 1, length(dd), 0.5)
pvalue
```

```
## [1] 7.466952e-10
```

In this case the p-value is close to 0.0, which would lead us to reject the null hypothesis that the median is 70. There is strong evidence to suggest that the median is greater than 70.

(b) Make a 90% confidence interval for the median.

```r
dd = c(79,74,88,80,80,66,65,86,84,80,78,72,71,74,86,96,77,81,76,80,76,75,78,8
7,87,74,85,84,76,77,76,74,85,74,76,77,76,74,81,76)
k_rng = c(1:length(dd))
min_diff = 1
#loop through all values of k
for (k in k_rng){
  prob = 1 - 2 * pbinom(k - 1, length(dd), 0.5)
  curr_diff = prob - 0.9
  #find the minimum non-negative difference for .9 confidence
  if((curr_diff < min_diff) & curr_diff > 0){
    min_diff = curr_diff
    k_val = k
    prob_val = prob
  }
}
k_val
```

```
## [1] 15
```

```r
prob_val
```

```
## [1] 0.9193095
```

```r
c(sort(dd)[k_val], sort(dd, decreasing=TRUE)[k_val])
```

```
## [1] 76 80
```

The 90% (91.9%) confidence interval is (76, 80).

(c) Make a 90% confidence interval for F(80), the probability that a score is less than or equal to 80.

```r
dd = c(79,74,88,80,80,66,65,86,84,80,78,72,71,74,86,96,77,81,76,80,76,75,78,8
7,87,74,85,84,76,77,76,74,85,74,76,77,76,74,81,76)
binom.test(sum(dd <= 80), length(dd), conf.level = 0.90)
```

```
##
##  Exact binomial test
##
## data:  sum(dd <= 80) and length(dd)
## number of successes = 28, number of trials = 40, p-value = 0.01659
## alternative hypothesis: true probability of success is not equal to 0.5
## 90 percent confidence interval:
##  0.5597203 0.8168787
## sample estimates:
## probability of success
##                    0.7
```

Assuming the sample is representative we're 90% confident that between 56% and 82% of the scores are smaller than 80.

Problem #2

The data in Table 2 are the yearly rainfall totals in Scranton, PA, for the years 1951-1984.

21.3 28.8 17.6 23.0 27.2 28.5 32.8 28.2 25.9 22.5 27.2 33.1 28.7 24.8 24.3 27.1 30.6 26.8 18.9 36.3 28.0 17.9 25.0 27.5 27.7 32.1 28.0 30.9 20.0 20.2 33.5 26.4 30.9 33.2

(a) Make a 95% confidence interval for the median.

```
dd = c(21.3, 28.8, 17.6, 23.0, 27.2, 28.5, 32.8, 28.2, 25.9, 22.5, 27.2, 33.1
, 28.7, 24.8, 24.3, 27.1, 30.6, 26.8, 18.9, 36.3, 28.0, 17.9, 25.0, 27.5, 27.
7, 32.1, 28.0, 30.9, 20.0, 20.2, 33.5, 26.4, 30.9, 33.2)
k_rng = c(1:length(dd))
min_diff = 1
#loop through all values of k
for (k in k_rng){
  prob = 1 - 2 * pbinom(k - 1, length(dd), 0.5)
  curr_diff = prob - 0.95
  #find the minimum non-negative difference for .95 confidence
  if((curr_diff < min_diff) & curr_diff > 0){
    min_diff = curr_diff
    k_val = k
    prob_val = prob
  }
}
k_val
```

```
## [1] 11
```

```
prob_val
```

```
## [1] 0.9756935
```

```
c(sort(dd)[k_val], sort(dd, decreasing=TRUE)[k_val])
```

```
## [1] 25.0 28.7
```

The 95% (97.6%) confindence interval for the median is (25.0, 28.7)

(b) The confidence interval procedure assumes that the observations are independent and identically distributed. Do you think this is a reasonable assumption for the rainfall data? If not, what could cause this assumption to be invalid?

In order to be i.i.d. each random variable must be independent with the same probablity distribution. Given that this is data tracked over time, this assumption may not hold. For example if there is a trend in the rainfall (e.g. average rainfall is increasing over time) then the individual random variables would not be indepedent, as knowing the prior year rainfall would have a bearing on the probablity of the current year rainfall. If the time periods were provided with the data one could check to see if there was an apparent trend

to test the independence assumptions. Given the current indications of changing global weather patterns the independence assumption may not be reasonable.

Problem #3

McCusker et al. (2006) studied the amount of caffeine in Starbucks decaffeinated coffee. They collected samples of espresso and brewed coffee. The measured caffeine levels (in mg per serving) were:

. Espresso: 15.8, 3.3, 4.1, 3.0, 12.7, 3.2 . Brewed: 12.0, 12.5, 13.0, 13.4, 13.4, 13.0

We wish to do a two-tailed nonparametric test of the null hypothesis that the caffeine distribution is the same in decaf espresso and decaf brewed coffee. What is the P-value if the test statistic is:

(a) the difference in sample means?

```
espr = c(15.8, 3.3, 4.1, 3.0, 12.7, 3.2)
brew = c(12.0, 12.5, 13.0, 13.4, 13.4, 13.0)

x_name <- "group"
y_name <- "caff"

caff.df <- melt(data.frame(espr,brew))

## No id variables; using all as measure variables

colnames(caff.df) <- c(x_name, y_name)

oneway_test(caff ~ group, data = caff.df, dist = "exact")

##
##  Exact Two-Sample Fisher-Pitman Permutation Test
##
## data:  caff by group (espr, brew)
## Z = -2.0617, p-value = 0.05195
## alternative hypothesis: true mu is not equal to 0
```

The P-value for the difference in sample means is approximatley .052. Therfore there is some evidence against the null hypothesis that the caffeine distribution is the same in decaf espresso and decaf brewed coffee.

(b) the Wilcoxon rank-sum statistic?

```
caff = c(15.8, 3.3, 4.1, 3.0, 12.7, 3.2, 12.0, 12.5, 13.0, 13.4, 13.4, 13.0)
group = c("espr", "espr", "espr", "espr", "espr", "espr", "brew", "brew", "br
ew", "brew", "brew", "brew")

caff.df <- data.frame(caff, group)

#order by caff amount
```

```
caff.df = caff.df[order(caff.df$caff),]

caff.groups = caff.df$group
ranks = 1:12
caff.groups = factor(caff.groups)

wilcox_test(ranks ~ caff.groups, dist = "exact")

##
##  Exact Wilcoxon-Mann-Whitney Test
##
## data:  ranks by caff.groups (brew, espr)
## Z = 1.6013, p-value = 0.132
## alternative hypothesis: true mu is not equal to 0
```

The P-value for the Wilcoxon rank-sum statistic is 0.132.

Problem #4

Suppose we observe two samples:

x = (1, 2, 3, 5) y = (4, 6, 9)

We decide to do a permutation test where the test statistic is the difference between the maximum x-value and the maximum y-value

(a) What is the permutation distribution of this test statistic?

```
xy = c(1, 2, 3, 5, 4, 6, 9)

#Gets all possible permutations of order
perm = permutations(n=length(xy), r=length(xy), v=xy)

perm.unique = perm[0,]

#Captures only the unique permutations for the two distinct vectors
vec1.strt = 1
vec1.end = 4
vec2.strt = 5
vec2.end = 7
for(perm.row in 1:nrow(perm)) {
  if(perm.row == 1) {
    perm.unique = rbind(perm.unique, perm[perm.row,])
  } else {
    flag = FALSE
    #remove duplicate permutations
    for(perm.unique.row in 1:nrow(perm.unique)) {
      if ((all(sort(perm[perm.row, vec1.strt:vec1.end]) == sort(perm.unique[p
erm.unique.row, vec1.strt:vec1.end]))) && +
          (all(sort(perm[perm.row, vec2.strt:vec2.end]) == sort(perm.unique[p
```

```
erm.unique.row, vec2.strt:vec2.end])))) {
        flag = FALSE
        break
      } else {
        flag = TRUE
      }
    }
    if(flag) {
      perm.unique = rbind(perm.unique, perm[perm.row,])
    }
  }
}

perm.unique
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6] [,7]
##  [1,]    1    2    3    4    5    6    9
##  [2,]    1    2    3    5    4    6    9
##  [3,]    1    2    3    6    4    5    9
##  [4,]    1    2    3    9    4    5    6
##  [5,]    1    2    4    5    3    6    9
##  [6,]    1    2    4    6    3    5    9
##  [7,]    1    2    4    9    3    5    6
##  [8,]    1    2    5    6    3    4    9
##  [9,]    1    2    5    9    3    4    6
## [10,]    1    2    6    9    3    4    5
## [11,]    1    3    4    5    2    6    9
## [12,]    1    3    4    6    2    5    9
## [13,]    1    3    4    9    2    5    6
## [14,]    1    3    5    6    2    4    9
## [15,]    1    3    5    9    2    4    6
## [16,]    1    3    6    9    2    4    5
## [17,]    1    4    5    6    2    3    9
## [18,]    1    4    5    9    2    3    6
## [19,]    1    4    6    9    2    3    5
## [20,]    1    5    6    9    2    3    4
## [21,]    2    3    4    5    1    6    9
## [22,]    2    3    4    6    1    5    9
## [23,]    2    3    4    9    1    5    6
## [24,]    2    3    5    6    1    4    9
## [25,]    2    3    5    9    1    4    6
## [26,]    2    3    6    9    1    4    5
## [27,]    2    4    5    6    1    3    9
## [28,]    2    4    5    9    1    3    6
## [29,]    2    4    6    9    1    3    5
## [30,]    2    5    6    9    1    3    4
## [31,]    3    4    5    6    1    2    9
## [32,]    3    4    5    9    1    2    6
## [33,]    3    4    6    9    1    2    5
```

```
## [34,]    3    5    6    9    1    2    4
## [35,]    4    5    6    9    1    2    3

#captures all the max differences for the unique permutations
for(perm.unique.row in 1:nrow(perm.unique)) {
  if(perm.unique.row == 1) {
    perm.unique.max.diff = c(max(perm.unique[perm.unique.row, vec1.strt:vec1.
end]) - max(perm.unique[perm.unique.row, vec2.strt:vec2.end]))
  } else {
    perm.unique.max.diff = c(perm.unique.max.diff, max(perm.unique[perm.uniqu
e.row, vec1.strt:vec1.end]) - max(perm.unique[perm.unique.row, vec2.strt:vec2
.end]))
  }
}

perm.unique.max.diff

##  [1] -5 -4 -3  3 -4 -3  3 -3  3  4 -4 -3  3 -3  3  4 -3  3  4  5 -4 -3  3
## [24] -3  3  4 -3  3  4  5 -3  3  4  5  6
```

Above shows all the unique permutations (where order isn't considered) of the two vectors. The first four columns represent vector x and the last 3 columns vector y. The bottom row shows all the possible values for the difference in the maximums for each unique permutation.

(b) What is the P-value of a two-tailed test using this statistic?

```
x = c(1, 2, 3, 5)
y = c(4, 6, 9)

test.stat = max(x) - max(y)
test.stat

## [1] -4

sum(abs(perm.unique.max.diff) >= abs(test.stat))/length(perm.unique.max.diff)

## [1] 0.4285714
```

The above code uses the test statistic permutation distribution calculated in the previous problem to calculate the P-value. The P-value for the test statistic, the difference in the maximum values, is approximatley .429.

Problem #5

I want to see if I like a typical Taylor Swift song or a typical Katy Perry song better. I take a random sample of six Taylor Swift songs and a random sample of six Katy Perry songs, and rank them in order of preference:

1. Swift: "Red" 2. Swift: "22" 3. Perry: "California Gurls" 4. Swift: "Teardrops on My Guitar" 5. Swift: "Treacherous" 6. Swift: "White Horse" 7. Swift: "Innocent" 8. Perry:

"Peacock" 9. Perry: "This Moment" 10. Perry: "Miss You More" 11. Perry: "If You Can Afford Me" 12. Perry: "Lost"

Find the Wilcoxon rank-sum statistic and a two-tailed P-value.

```
artists = c("Swift", "Swift", "Perry", "Swift", "Swift", "Swift", "Swift", "P
erry", "Perry", "Perry", "Perry", "Perry")
ranks = 1:12

artists = factor(artists)

wilcox_test(ranks ~ artists, dist = "exact")

##
##  Exact Wilcoxon-Mann-Whitney Test
##
## data:  ranks by artists (Perry, Swift)
## Z = 2.2418, p-value = 0.02597
## alternative hypothesis: true mu is not equal to 0
```

The P-value for the Wilcoxon rank-sum statistic is approximatley .026.

Problem #6

The file women.txt contains data on the height (in centimeters) and race of a sample of 423 American adult women.

(a) Use R to perform a nonparametric test of your choice of the null hypothesis that white women and black women have the same distribution of heights.

```
dd <- read.table("women.txt", sep = " " , header = T,
                      na.strings ="", stringsAsFactors= T)
dd$Height = as.numeric(as.character(dd$Height))

## Warning: NAs introduced by coercion

dd = dd[which(dd$Race =='Black' | dd$Race =='White'), ]

#order by Heights
dd = dd[order(dd$Height),]

dd = na.omit(dd)

dd.race = dd$Race
ranks = 1:length(dd.race)
dd.race = factor(dd.race)

wilcox_test(ranks ~ dd.race, dist = "exact")

##
##  Exact Wilcoxon-Mann-Whitney Test
```

```
##
## data:  ranks by dd.race (Black, White)
## Z = 0.4103, p-value = 0.683
## alternative hypothesis: true mu is not equal to 0
```

The P-value for the Wilcoxon rank-sum statistic is 0.683. This is evidence that the null hypothesis should not be rejected, that white women and black women have the same distribution of heights.

(b) Find a 95% confidence interval for the difference between a typical white woman's height and a typical black woman's height. Carefully state any assumptions you make.

```
dd <- read.table("women.txt", sep = " " , header = T,
                      na.strings ="", stringsAsFactors= T)
dd$Height = as.numeric(as.character(dd$Height))

## Warning: NAs introduced by coercion

dd = dd[which(dd$Race =='Black' | dd$Race =='White'), ]

#order by Heights
dd = dd[order(dd$Height),]

dd = na.omit(dd)

dd.race = dd$Race
ranks = 1:length(dd.race)
dd.race = factor(dd.race)

wilcox_test(ranks ~ dd.race, dist = "exact", conf.int = TRUE)

##
##  Exact Wilcoxon-Mann-Whitney Test
##
## data:  ranks by dd.race (Black, White)
## Z = 0.4103, p-value = 0.683
## alternative hypothesis: true mu is not equal to 0
## 95 percent confidence interval:
##  -25  39
## sample estimates:
## difference in location
##                      7
```

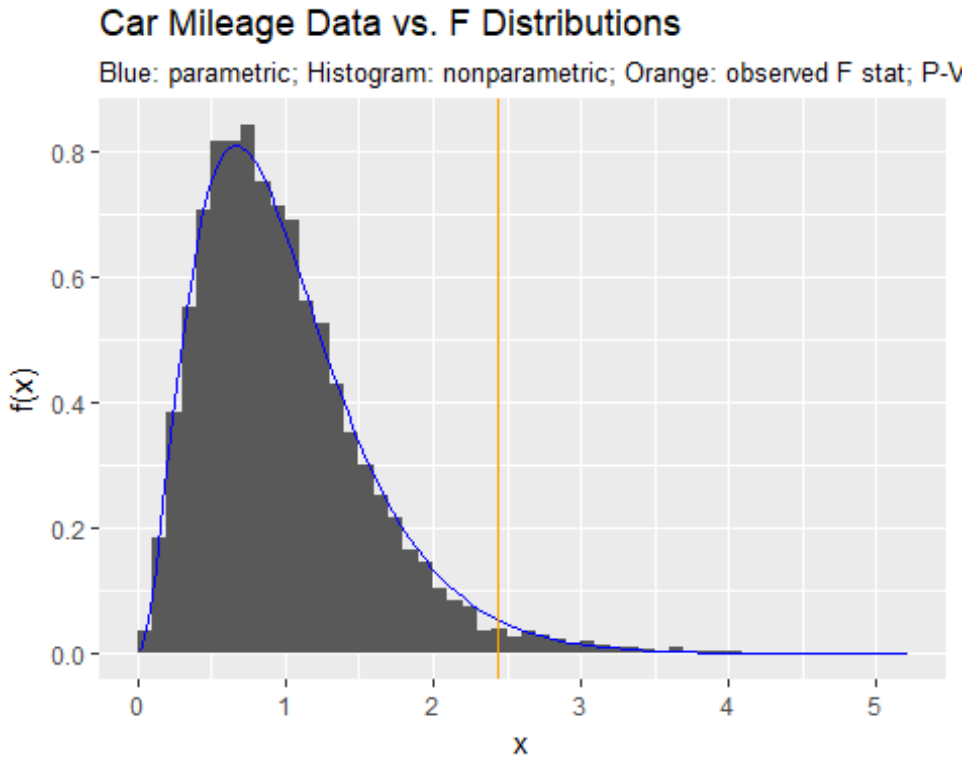
The 95% confidence interval for the difference between a typical white woman's height and a typical black woman's height is [-25 cm, 39 cm]. The Hodges-Lehmann estimate assumes that the observations for white and black women's heights are both IID and the observations between the groups are independent as well. There is also an assumption that there is indeed a shift between the groups. For example, if one distribution is very spread out and the other isn't, this estimate would not be very good.

Problem #7

The file cars2014.txt on Canvas contains data on the fuel economy of 795 model year 2014 cars. (The data is adapted from the 'fueleconomy' package. Station wagons are included while vans, trucks, SUVs, and limousines are excluded.) The numerical response we will study is cty, which gives the car's miles per gallon when driving in the city. The two explanatory variables we'll look at are trans, which gives the car's transmission (either Automatic or Manual), and class, which gives the type of car. The possible classes are minicompact, subcompact, compact, midsize, large, station wagon, and two-seater. (For the purpose of this homework, consider the data here as sampled from some theoretical superpopulation of cars.) Test the hypothesis that the distribution of city miles per gallon is the same for all classes of cars, using a nonparametric test of your choice. State what test you're doing and give R code, the test statistic, a P-value, and a substantive conclusion.

```
dd = read.table("cars2014.txt", sep = " " , header = T,
                    na.strings ="", stringsAsFactors= T)
dd = select(dd, class, cty)

#Parametric case
anova(lm(cty ~ class, data = dd))

## Analysis of Variance Table
##
## Response: cty
##            Df Sum Sq Mean Sq F value  Pr(>F)
## class       6   2269  378.10   2.432 0.02457 *
## Residuals 788 122507  155.47
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

anova(lm(cty ~ class, data = dd))$F

## [1] 2.432046       NA

observed.F = anova(lm(cty ~ class, data = dd))$F[1]

#Non-Parametric case
permutationF = function(data) {
  perm.data = data.frame(group = data[, 1], numbers = sample(data[, 2]))
  perm.F = anova(lm(numbers ~ group, data = perm.data))$F[1]
  return(perm.F)
}

perm.dist = replicate(10000, permutationF(dd))
pval = mean(perm.dist >= observed.F)

gg = ggplot(data.frame(perm.dist), aes(x = perm.dist)) +
  geom_histogram(aes(y = ..density..), breaks = seq(0, 5, 0.1)) +
```

```
  stat_function(fun = "df", args = list(df1=6, df2=788), col = "blue") +
  geom_vline(xintercept = observed.F, col = "orange")
gg + ggtitle("Car Mileage Data vs. F Distributions") +
  xlab("x") + ylab("f(x)") + labs(subtitle = paste0("Blue: parametric; Histog
ram: nonparametric; Orange: observed F stat; P-Value nonparametric: ", pval))
```



For this analysis I used the nonparametric permutation F-test. This test uses the F-statistic as the test statistic. The P-Value produced by the chosen 10,000 permutations is .0255. The null hypothesis for this test is that the mean city mpg across all the different classes of vehicles is the same. Given the low P-Value (.0255) there is strong evidence that the null hypothesis should be rejected, meaning there is strong evidence that the mean city mpg across the different classes of vehicles are not the same.

Problem #8

For the cars2014.txt, determine which differences between classes are significant at an overall significance level of ?? = 0.05, using the nonparametric rank-based version of Tukey's HSD.

```
dd = read.table("cars2014.txt", sep = " " , header = T,
                      na.strings ="", stringsAsFactors= T)
dd = select(dd, class, cty)

Ranks = rank(dd$cty)
dd.ranks = data.frame(dd, Ranks)
dd.rank.means = aggregate(Ranks ~ class, mean, data = dd.ranks)
```

```r
q = qtukey(0.95, nmeans = 7, df = 100000)

#Get unique combinations of classes
class.combn.temp = expand.grid(classX = dd.rank.means$class, classY = dd.rank
.means$class)
for (i in 1:nrow(class.combn.temp))
{
    class.combn.temp[i, ] = sort(class.combn.temp[i, ])
}
class.combn.temp = class.combn.temp[!duplicated(class.combn.temp),]

class.combn = class.combn.temp[0,]
for(row in 1:nrow(class.combn.temp)) {
  if (class.combn.temp[row,]$classX != class.combn.temp[row,]$classY) {
    class.combn = rbind(class.combn, class.combn.temp[row,])
  }
}

#For each combination of classes fill in the needed values
N = length(dd$class)
class.combn$rank.mean.x = NA
class.combn$rank.mean.y = NA
class.combn$q = q
class.combn$N = N
class.combn$n.x = NA
class.combn$n.y = NA
class.combn$reject.null = NA
for(row in 1:nrow(class.combn)) {
  class.combn[row,]$n.x = length(dd[which(dd$class==class.combn[row,]$classX)
, ]$class)
  class.combn[row,]$n.y = length(dd[which(dd$class==class.combn[row,]$classY)
, ]$class)
  class.combn[row,]$rank.mean.x = dd.rank.means[which(dd.rank.means$class==cl
ass.combn[row,]$classX), ]$Ranks
  class.combn[row,]$rank.mean.y = dd.rank.means[which(dd.rank.means$class==cl
ass.combn[row,]$classY), ]$Ranks
}

#Calculate minimun rejection difference
class.combn$rank.mean.abs.diff = abs(class.combn$rank.mean.x - class.combn$ra
nk.mean.y)
class.combn$min.reject.diff = q * sqrt(((N*(N + 1))/24) * ((1 / class.combn$n
.x) + (1 / class.combn$n.y)))

#Determine if the given difference is larger than the minimum
for(row in 1:nrow(class.combn)) {
  if(class.combn[row,]$rank.mean.abs.diff >=  class.combn[row,]$min.reject.di
ff) {
```

```
    class.combn[row,]$reject.null = TRUE
  } else {
    class.combn[row,]$reject.null = FALSE
  }
}

#subset to only rejected class combinations
class.combn.reject = class.combn[which(class.combn$reject.null==TRUE), ]
class.combn.reject[, c(1, 2, 10, 11)]
```

```
##            classX           classY rank.mean.abs.diff min.reject.diff
## 2    Compact Cars       Large Cars           235.7842        82.04351
## 4    Compact Cars Minicompact Cars           119.0363       105.33790
## 6    Compact Cars  Subcompact Cars           110.3340        83.41586
## 7    Compact Cars      Two Seaters           186.0870        86.54334
## 10     Large Cars     Midsize Cars           194.7157        81.25671
## 11     Large Cars Minicompact Cars           116.7479       115.17804
## 12     Large Cars   Station Wagons           206.5517       126.13729
## 13     Large Cars  Subcompact Cars           125.4502        95.54115
## 21   Midsize Cars      Two Seaters           145.0185        85.79781
## 35 Station Wagons      Two Seaters           156.8545       129.10937
```

Of the 21 unique combinations of vehicles classes, 10 had a difference in the absolute value of their ranked mean difference that was greater than the minimum difference for rejection at 95% confidence.