STAT 3480 Lab7

Shaoran Sun

March 31, 2016

Pairwise Comparisons

1

We would have to consider 6 comparisons.

In general, for K different groups, we will have to perform $\frac{(K-1)K}{2}$ pairwise comparisons.

Multiple Testing Problem

2

H_0: The mean of the normal distribution is 10.

H_a: The mean of the normal distribution is NOT 10.

After performing a t-test on the data points, we have p-value equals to 0.6293, which is greater than 0.05. Hence, we fail to reject the null hyopthesis at 0.05 significance level, and conclude that there is not enough eveidence to show that the mean of the normal distribution is NOT 10. In another words, the mean is indeed 10.

3

After running the code 20 times, there is one time that we can reject the null hypothesis at 0.05 significance level. This is acceptable, since with 20 tests and 0.05 significance level, we would expect to incorrectly reject the null hypothesis in 1 of these 20 tests.

4

There are 51 times, which is 0.051 chance that we reject the null hypothesis that $\mu = 10$.

Bonferroni Correction

5

We would need a significance level of $\frac{0.05}{6} = 0.00833$.

6

We would multiply the originial p-values by 6.

6

After performing a permutation test, using a difference in means, to determine whether there is a differece in run times between *G and PG* movies. We have the original p-value of 0.049, and the Bonferroni-adjusted p-value of **0.294**.

Based on the Bonferroni-adjusted p-value of 0.294, we fail to reject the null hypothesis at 0.05 significance level, and conclude that there is not enough evidence to show that there were differences in run times among the rating groups of *G and PG*.

7

After performing a permutation test, using a difference in means, to determine whether there is a differece in run times between *G and PG-13* movies. We have the original p-value of 0.043, and the Bonferroni-adjusted p-value of **0.258**.

Based on the Bonferroni-adjusted p-value of 0.258, we fail to reject the null hypothesis at 0.05 significance level, and conclude that there is not enough evidence to show that there were differences in run times among the rating groups of *G and PG-13*.

8

After performing a permutation test, using a difference in means, to determine whether there is a differece in run times between *G and R* movies. We have the original p-value of 0.007, and the Bonferroni-adjusted p-value of **0.042**.

Based on the Bonferroni-adjusted p-value of 0.042, we reject the null hypothesis at 0.05 significance level, and conclude that there were differences in run times among the rating groups of *G and R*.

9

After performing a permutation test, using a difference in means, to determine whether there is a differece in run times between *PG and PG-13* movies. We have the original p-value of 0.006, and the Bonferroni-adjusted p-value of **0.036**.

Based on the Bonferroni-adjusted p-value of 0.036, we reject the null hypothesis at 0.05 significance level, and conclude that there were differences in run times among the rating groups of *PG and PG-13*.

10

After performing a permutation test, using a difference in means, to determine whether there is a differece in run times between *PG and R* movies. We have the original p-value of 0.001, and the Bonferroni-adjusted p-value of **0.006**.

Based on the Bonferroni-adjusted p-value of 0.006, we reject the null hypothesis at 0.05 significance level, and conclude that there were differences in run times among the rating groups of *PG and R*.

11

After performing a permutation test, using a difference in means, to determine whether there is a differece in run times between *PG-13 and R* movies. We have the original p-value of 1.018, and the Bonferroni-adjusted p-value of **6.108**.

Based on the Bonferroni-adjusted p-value of 6.108, we fail to reject the null hypothesis at 0.05 significance level, and conclude there is not enough evidence to show that there were differences in run times among the rating groups of *PG-13 and R*.

12

From the results of #6 through #11, (G and PG), (G and PG-13), and (PG-13 and R) do NOT show significant evidence of differences in run times.

However, (G and R), (PG and PG-13), and (PG and R) do show significant evidence of differences in run times within each pair.

Lab Summary

1

We first performed an overall test for differences between movie ratings and scores on *rottentomatoes.com*, using Kruskal-Wallis test, to test H_0: the scores of different ratings are identical populations, against H_a: the scores of different ratings are NOT identical populations.

We have a p-value of 0.528, which is greater than 0.05. Hence we fail to reject the null hypothesis at 0.05 siginificance level, and conclude that the scores of different ratings are identical populations. In another words, *rating and scores* are *not* correlated.

Secondly, we perform pairwise tests of each rating pair using the Wilcoxon rank-sum test and Bonferroni-adjusted p-values between movie ratings and scores on *rottentomatoes.com*, to test H_0: the *pairwise* scores of different ratings are identical populations, against H_a: the *pairwise* scores of different ratings are NOT identical populations.

We have Bonferroni-adjusted p-values of 1.020, 0.925, 0.962, 5.448, 5.190, and 5.260, which are all greater than 0.05. Hence we fail to reject the null hypothesis at 0.05 siginificance level, and conclude the *same* conclusions that the scores of different ratings are identical populations. In another words, *rating and scores* are *not* correlated.

2

We first performed an overall test for differences between *movie ratings and box office gross*, using Kruskal-Wallis test, to test H_0: the box office gross of different ratings are identical populations, against H_a: the box office gross of different ratings are NOT identical populations.

We have a p-value of 0.0778, which is greater than 0.05. Hence we fail to reject the null hypothesis at 0.05 siginificance level, and conclude that the box office gross of different ratings are identical populations. In another words, *box office gross and scores* are *not* correlated.

Secondly, we perform pairwise tests of each rating pair using the Wilcoxon rank-sum test and Bonferroni-adjusted p-values between movie ratings and box office gross, to test H_0: the *pairwise* box office gross of different ratings are identical populations, against H_a: the *pairwise* box office gross of different ratings are NOT identical populations.

We have Bonferroni-adjusted p-values of 3.853, 2.254, 0.905, 1.693, 0.216, and 0.462, which are all greater than 0.05. Hence we fail to reject the null hypothesis at 0.05 siginificance level, and conclude the *same* conclusions that the box office gross of different ratings are identical populations. In another words, *box office gross and scores* are *not* correlated.

Appendix

2

```
data = rnorm(100, 10, 5)
t.test(data, mu = 10, alternative = "two.sided")
```

```
##
##  One Sample t-test
##
## data:  data
## t = -0.43197, df = 99, p-value = 0.6667
## alternative hypothesis: true mean is not equal to 10
## 95 percent confidence interval:
##   8.893614 10.710785
## sample estimates:
## mean of x
##    9.8022
```

3

```
replicate(
  20,
  {
     data = rnorm(100, 10, 5)
t.test(data, mu = 10, alternative = "two.sided")
  }
)
```

```
##             [,1]                [,2]                [,3]
## statistic   -1.38616            -0.8345986          0.08574029
## parameter   99                  99                  99
## p.value     0.1688121           0.4059532           0.931846
## conf.int    Numeric,2           Numeric,2           Numeric,2
## estimate    9.318297            9.606335            10.04354
## null.value  10                  10                  10
## alternative "two.sided"         "two.sided"         "two.sided"
## method      "One Sample t-test" "One Sample t-test" "One Sample t-test"
## data.name   "data"              "data"              "data"
##             [,4]                [,5]                [,6]
## statistic   -0.2624929          1.344138            1.199562
## parameter   99                  99                  99
## p.value     0.7934869           0.1819759           0.2331725
## conf.int    Numeric,2           Numeric,2           Numeric,2
## estimate    9.861177            10.61375            10.62572
## null.value  10                  10                  10
## alternative "two.sided"         "two.sided"         "two.sided"
## method      "One Sample t-test" "One Sample t-test" "One Sample t-test"
## data.name   "data"              "data"              "data"
```

```
##             [,7]                [,8]                [,9]
## statistic   -0.5713014          -1.251158           1.598659
## parameter   99                  99                  99
## p.value     0.5690901           0.2138256           0.1130831
## conf.int    Numeric,2           Numeric,2           Numeric,2
## estimate    9.707808            9.364613            10.7977
## null.value  10                  10                  10
## alternative "two.sided"         "two.sided"         "two.sided"
## method      "One Sample t-test" "One Sample t-test" "One Sample t-test"
## data.name   "data"              "data"              "data"
##             [,10]               [,11]               [,12]
## statistic   -0.1082122          1.344504            -1.082318
## parameter   99                  99                  99
## p.value     0.9140464           0.1818581           0.2817406
## conf.int    Numeric,2           Numeric,2           Numeric,2
## estimate    9.948901            10.6249             9.450237
## null.value  10                  10                  10
## alternative "two.sided"         "two.sided"         "two.sided"
## method      "One Sample t-test" "One Sample t-test" "One Sample t-test"
## data.name   "data"              "data"              "data"
##             [,13]               [,14]               [,15]
## statistic   1.256872            -1.760665           -0.6074658
## parameter   99                  99                  99
## p.value     0.2117572           0.0813831           0.5449327
## conf.int    Numeric,2           Numeric,2           Numeric,2
## estimate    10.6916             9.183341            9.722455
## null.value  10                  10                  10
## alternative "two.sided"         "two.sided"         "two.sided"
## method      "One Sample t-test" "One Sample t-test" "One Sample t-test"
## data.name   "data"              "data"              "data"
##             [,16]               [,17]               [,18]
## statistic   0.09771875          0.2717496           0.05857755
## parameter   99                  99                  99
## p.value     0.9223532           0.7863805           0.9534067
## conf.int    Numeric,2           Numeric,2           Numeric,2
## estimate    10.04932            10.1193             10.03106
## null.value  10                  10                  10
## alternative "two.sided"         "two.sided"         "two.sided"
## method      "One Sample t-test" "One Sample t-test" "One Sample t-test"
## data.name   "data"              "data"              "data"
##             [,19]               [,20]
## statistic   -1.417944           -2.630814
## parameter   99                  99
## p.value     0.1593471           0.009879744
## conf.int    Numeric,2           Numeric,2
## estimate    9.355798            8.736466
## null.value  10                  10
## alternative "two.sided"         "two.sided"
## method      "One Sample t-test" "One Sample t-test"
## data.name   "data"              "data"
```

4

```
pval = rep(NA, 1000)
for(i in 1:1000) {
data = rnorm(100, 10, 5)
pval[i] = t.test(data, mu = 10, alternative = "two.sided")$p.value
}
sum(pval < .05)
```

```
## [1] 49
```

```
sum(pval < .05)/1000
```

```
## [1] 0.049
```

6

```
moviesall <- read.delim("~/Desktop/lab7/moviesall.txt")
attach(moviesall)
movies.pair = moviesall[rating=="G" | rating=="PG",]
detach(moviesall)
attach(movies.pair)
teststat.obs = mean(runtime[rating == "G"]) - mean(runtime[rating == "PG"])
teststat.obs
```

```
## [1] -12.58333
```

```
m = length(runtime[rating == "G"])
n = length(runtime[rating == "PG"])
teststat = rep(NA, 1000)
for(i in 1:1000) {
### randomly "shuffle" the elements of the runtime vector
runtimeSHUFFLE = sample(runtime)
### assign the first m to the first group
### and the next n to the other group
G = runtimeSHUFFLE[1:m]
PG = runtimeSHUFFLE[(m+1):(m+n)]
### compute the test stat for the shuffled data
teststat[i] = mean(G) - mean(PG)
}
### calculate the approximate p-value
sum(teststat <= teststat.obs)/1000 + sum(teststat >= -teststat.obs)/1000
```

```
## [1] 0.047
```

```
teststat.obs
```

```
## [1] -12.58333
```

```r
((sum(teststat <= teststat.obs)/1000 + sum(teststat >= -teststat.obs)/1000))*6
```

```
## [1] 0.282
```

7

```r
moviesall <- read.delim("~/Desktop/lab7/moviesall.txt")
attach(moviesall)
```

```
## The following objects are masked from movies.pair:
##
##     genre, gross, rating, runtime, score
```

```r
movies.pair = moviesall[rating=="G" | rating=="PG-13",]
detach(moviesall)
attach(movies.pair)
```

```
## The following objects are masked from movies.pair (pos = 3):
##
##     genre, gross, rating, runtime, score
```

```r
teststat.obs = mean(runtime[rating == "G"]) - mean(runtime[rating == "PG-13"])
teststat.obs
```

```
## [1] -27.94231
```

```r
m = length(runtime[rating == "G"])
n = length(runtime[rating == "PG-13"])
teststat = rep(NA, 1000)
for(i in 1:1000) {
### randomly "shuffle" the elements of the runtime vector
runtimeSHUFFLE = sample(runtime)
### assign the first m to the first group
### and the next n to the other group
G = runtimeSHUFFLE[1:m]
PG13 = runtimeSHUFFLE[(m+1):(m+n)]
### compute the test stat for the shuffled data
teststat[i] = mean(G) - mean(PG13)
}
### calculate the approximate p-value
sum(teststat <= teststat.obs)/1000 + sum(teststat >= -teststat.obs)/1000
```

```
## [1] 0.047
```

```r
teststat.obs
```

```
## [1] -27.94231
```

```r
((sum(teststat <= teststat.obs)/1000 + sum(teststat >= -teststat.obs)/1000))*6
```

```
## [1] 0.282
```

8

```r
moviesall <- read.delim("~/Desktop/lab7/moviesall.txt")
attach(moviesall)
```

```
## The following objects are masked from movies.pair (pos = 3):
##
##     genre, gross, rating, runtime, score

## The following objects are masked from movies.pair (pos = 4):
##
##     genre, gross, rating, runtime, score
```

```r
movies.pair = moviesall[rating=="G" | rating=="R",]
detach(moviesall)
attach(movies.pair)
```

```
## The following objects are masked from movies.pair (pos = 3):
##
##     genre, gross, rating, runtime, score
##
## The following objects are masked from movies.pair (pos = 4):
##
##     genre, gross, rating, runtime, score
```

```r
teststat.obs = mean(runtime[rating == "G"]) - mean(runtime[rating == "R"])
teststat.obs
```

```
## [1] -27.85
```

```r
m = length(runtime[rating == "G"])
n = length(runtime[rating == "R"])
teststat = rep(NA, 1000)
for(i in 1:1000) {
### randomly "shuffle" the elements of the runtime vector
runtimeSHUFFLE = sample(runtime)
### assign the first m to the first group
### and the next n to the other group
G = runtimeSHUFFLE[1:m]
R = runtimeSHUFFLE[(m+1):(m+n)]
### compute the test stat for the shuffled data
teststat[i] = mean(G) - mean(R)
}
### calculate the approximate p-value
sum(teststat <= teststat.obs)/1000 + sum(teststat >= -teststat.obs)/1000
```

```
## [1] 0.003
```

```r
teststat.obs
```

```
## [1] -27.85
```

```r
((sum(teststat <= teststat.obs)/1000 + sum(teststat >= -teststat.obs)/1000))*6
```

```
## [1] 0.018
```

9

```r
moviesall <- read.delim("~/Desktop/lab7/moviesall.txt")
attach(moviesall)
```

```
## The following objects are masked from movies.pair (pos = 3):
##
##     genre, gross, rating, runtime, score

## The following objects are masked from movies.pair (pos = 4):
##
##     genre, gross, rating, runtime, score

## The following objects are masked from movies.pair (pos = 5):
##
##     genre, gross, rating, runtime, score
```

```r
movies.pair = moviesall[rating=="PG" | rating=="PG-13",]
detach(moviesall)
attach(movies.pair)
```

```
## The following objects are masked from movies.pair (pos = 3):
##
##     genre, gross, rating, runtime, score

## The following objects are masked from movies.pair (pos = 4):
##
##     genre, gross, rating, runtime, score

## The following objects are masked from movies.pair (pos = 5):
##
##     genre, gross, rating, runtime, score
```

```r
teststat.obs = mean(runtime[rating == "PG"]) - mean(runtime[rating == "PG-13"])
teststat.obs
```

```
## [1] -15.35897
```

```r
m = length(runtime[rating == "PG"])
n = length(runtime[rating == "PG-13"])
teststat = rep(NA, 1000)
for(i in 1:1000) {
### randomly "shuffle" the elements of the runtime vector
runtimeSHUFFLE = sample(runtime)
### assign the first m to the first group
### and the next n to the other group
PG = runtimeSHUFFLE[1:m]
PG13 = runtimeSHUFFLE[(m+1):(m+n)]
### compute the test stat for the shuffled data
teststat[i] = mean(PG) - mean(PG13)
}
### calculate the approximate p-value
sum(teststat <= teststat.obs)/1000 + sum(teststat >= -teststat.obs)/1000
```

```
## [1] 0.008
```

```r
teststat.obs
```

```
## [1] -15.35897
```

```r
((sum(teststat <= teststat.obs)/1000 + sum(teststat >= -teststat.obs)/1000))*6
```

```
## [1] 0.048
```

10

```r
moviesall <- read.delim("~/Desktop/lab7/moviesall.txt")
attach(moviesall)
```

```
## The following objects are masked from movies.pair (pos = 3):
##
##     genre, gross, rating, runtime, score

## The following objects are masked from movies.pair (pos = 4):
##
##     genre, gross, rating, runtime, score

## The following objects are masked from movies.pair (pos = 5):
##
##     genre, gross, rating, runtime, score

## The following objects are masked from movies.pair (pos = 6):
##
##     genre, gross, rating, runtime, score
```

```r
movies.pair = moviesall[rating=="PG" | rating=="R",]
detach(moviesall)
attach(movies.pair)
```

```
## The following objects are masked from movies.pair (pos = 3):
##
##     genre, gross, rating, runtime, score

## The following objects are masked from movies.pair (pos = 4):
##
##     genre, gross, rating, runtime, score

## The following objects are masked from movies.pair (pos = 5):
##
##     genre, gross, rating, runtime, score

## The following objects are masked from movies.pair (pos = 6):
##
##     genre, gross, rating, runtime, score
```

```r
teststat.obs = mean(runtime[rating == "PG"]) - mean(runtime[rating == "R"])
teststat.obs
```

```
## [1] -15.26667
```

```r
m = length(runtime[rating == "PG"])
n = length(runtime[rating == "R"])
teststat = rep(NA, 1000)
for(i in 1:1000) {
### randomly "shuffle" the elements of the runtime vector
runtimeSHUFFLE = sample(runtime)
### assign the first m to the first group
### and the next n to the other group
PG = runtimeSHUFFLE[1:m]
R = runtimeSHUFFLE[(m+1):(m+n)]
### compute the test stat for the shuffled data
teststat[i] = mean(PG) - mean(R)
}
### calculate the approximate p-value
sum(teststat <= teststat.obs)/1000 + sum(teststat >= -teststat.obs)/1000
```

```
## [1] 0.003
```

```r
teststat.obs
```

```
## [1] -15.26667
```

```r
((sum(teststat <= teststat.obs)/1000 + sum(teststat >= -teststat.obs)/1000))*6
```

```
## [1] 0.018
```

11

```r
moviesall <- read.delim("~/Desktop/lab7/moviesall.txt")
attach(moviesall)
```

```
## The following objects are masked from movies.pair (pos = 3):
##
##     genre, gross, rating, runtime, score

## The following objects are masked from movies.pair (pos = 4):
##
##     genre, gross, rating, runtime, score

## The following objects are masked from movies.pair (pos = 5):
##
##     genre, gross, rating, runtime, score

## The following objects are masked from movies.pair (pos = 6):
##
##     genre, gross, rating, runtime, score

## The following objects are masked from movies.pair (pos = 7):
##
##     genre, gross, rating, runtime, score
```

```r
movies.pair = moviesall[rating=="PG-13" | rating=="R",]
detach(moviesall)
attach(movies.pair)
```

```
## The following objects are masked from movies.pair (pos = 3):
##
##     genre, gross, rating, runtime, score

## The following objects are masked from movies.pair (pos = 4):
##
##     genre, gross, rating, runtime, score

## The following objects are masked from movies.pair (pos = 5):
##
##     genre, gross, rating, runtime, score

## The following objects are masked from movies.pair (pos = 6):
##
##     genre, gross, rating, runtime, score

## The following objects are masked from movies.pair (pos = 7):
##
##     genre, gross, rating, runtime, score
```

```r
teststat.obs = mean(runtime[rating == "PG-13"]) - mean(runtime[rating == "R"])
teststat.obs
```

```
## [1] 0.09230769
```

```r
m = length(runtime[rating == "PG-13"])
n = length(runtime[rating == "R"])
teststat = rep(NA, 1000)
for(i in 1:1000) {
### randomly "shuffle" the elements of the runtime vector
runtimeSHUFFLE = sample(runtime)
### assign the first m to the first group
### and the next n to the other group
PG13 = runtimeSHUFFLE[1:m]
R = runtimeSHUFFLE[(m+1):(m+n)]
### compute the test stat for the shuffled data
teststat[i] = mean(PG13) - mean(R)
}
### calculate the approximate p-value
sum(teststat <= teststat.obs)/1000 + sum(teststat >= -teststat.obs)/1000
```

```
## [1] 1.018
```

```r
teststat.obs
```

```
## [1] 0.09230769
```

```r
((sum(teststat <= teststat.obs)/1000 + sum(teststat >= -teststat.obs)/1000))*6
```

```
## [1] 6.108
```

1

```r
moviesall <- read.delim("~/Desktop/lab7/moviesall.txt")
attach(moviesall)
```

```
## The following objects are masked from movies.pair (pos = 3):
##
##     genre, gross, rating, runtime, score

## The following objects are masked from movies.pair (pos = 4):
##
##     genre, gross, rating, runtime, score

## The following objects are masked from movies.pair (pos = 5):
##
##     genre, gross, rating, runtime, score
```

```
## The following objects are masked from movies.pair (pos = 6):
##
##     genre, gross, rating, runtime, score

## The following objects are masked from movies.pair (pos = 7):
##
##     genre, gross, rating, runtime, score

## The following objects are masked from movies.pair (pos = 8):
##
##     genre, gross, rating, runtime, score
```

```
kruskal.test(score ~ rating, data = moviesall)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  score by rating
## Kruskal-Wallis chi-squared = 2.2204, df = 3, p-value = 0.5279
```

```
wilcox.test(score[rating == "G"], score[rating == "PG"], correct=T)$p.value*6
```

```
## Warning in wilcox.test.default(score[rating == "G"], score[rating ==
## "PG"], : cannot compute exact p-value with ties

## [1] 1.019695
```

```
wilcox.test(score[rating == "G"], score[rating == "PG-13"], correct=T)$p.value*6
```

```
## [1] 0.9246197
```

```
wilcox.test(score[rating == "G"], score[rating == "R"], correct=T)$p.value*6
```

```
## [1] 0.9623499
```

```
wilcox.test(score[rating == "PG"], score[rating == "PG-13"], correct=T)$p.value*6
```

```
## [1] 5.447783
```

```
wilcox.test(score[rating == "PG"], score[rating == "R"], correct=T)$p.value*6
```

```
## [1] 5.189631
```

```
wilcox.test(score[rating == "PG-13"], score[rating == "R"], correct=T)$p.value*6
```

```
## [1] 5.260191
```

2

```r
moviesall <- read.delim("~/Desktop/lab7/moviesall.txt")
attach(moviesall)
```

```
## The following objects are masked from moviesall (pos = 3):
##
##     genre, gross, rating, runtime, score

## The following objects are masked from movies.pair (pos = 4):
##
##     genre, gross, rating, runtime, score

## The following objects are masked from movies.pair (pos = 5):
##
##     genre, gross, rating, runtime, score

## The following objects are masked from movies.pair (pos = 6):
##
##     genre, gross, rating, runtime, score

## The following objects are masked from movies.pair (pos = 7):
##
##     genre, gross, rating, runtime, score

## The following objects are masked from movies.pair (pos = 8):
##
##     genre, gross, rating, runtime, score

## The following objects are masked from movies.pair (pos = 9):
##
##     genre, gross, rating, runtime, score
```

```r
kruskal.test(gross ~ rating, data = moviesall)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  gross by rating
## Kruskal-Wallis chi-squared = 6.8215, df = 3, p-value = 0.07781
```

```r
wilcox.test(gross[rating == "G"], gross[rating == "PG"], correct=T)$p.value*6
```

```
## [1] 3.853281
```

```r
wilcox.test(gross[rating == "G"], gross[rating == "PG-13"], correct=T)$p.value*6
```

```
## [1] 2.254099
```

```r
wilcox.test(gross[rating == "G"], gross[rating == "R"], correct=T)$p.value*6
```

```
## [1] 0.9047211
```

```r
wilcox.test(gross[rating == "PG"], gross[rating == "PG-13"], correct=T)$p.value*6
```

```
## [1] 1.692653
```

```r
wilcox.test(gross[rating == "PG"], gross[rating == "R"], correct=T)$p.value*6
```

```
## [1] 0.2155884
```

```r
wilcox.test(gross[rating == "PG-13"], gross[rating == "R"], correct=T)$p.value*6
```

```
## [1] 0.4616502
```