

Non-parametric Statistics: Notes 3

K -sample methods

Gregory Matthews ¹

¹Department of Mathematics and Statistics
Loyola University Chicago

Spring 2016

Outline

K -sample methods

Kruskal-Wallis

Multiple Comparisons

- ▶ In the last section, we compared two groups; Now we generalize to K groups.
- ▶ In general, we first test to see if there is ANY difference between the groups.
- ▶ If ANY difference is found, we then perform multiple comparisons to see which groups differ significantly.

F -test review

- ▶ Let's review the F -test from a one-way ANOVA analysis.
- ▶ What are we testing in ANOVA?

$H_0 : \mu_1 = \mu_2 = \dots = \mu_j$ vs $H_1 : \mu_i \neq \mu_j$ for some i, j

- ▶ The null hypothesis here is that the means of all the groups are equal.
- ▶ The alternative is that there is at least a pair of groups with different means.

These should look familiar!

$$SST(or SSR) = \sum_{i=1}^k n_i (\bar{X}_i - \bar{X})^2$$

$$SSE = \sum_{i=1}^k (n_i - 1) S_i^2$$

Then

$$MST = \frac{SST}{k - 1}$$

and

$$MSE = \frac{SSE}{n - k}$$

and finally

$$F = \frac{MST}{MSE} \stackrel{H_0}{\sim} F_{k-1, n-k}$$

- ▶ Q: But what assumption do we need to make the last equation true?
- ▶ A: We need the assumption of normality! (among others. Like what?)
- ▶ Q: What if we don't assume normality here?
- ▶ A: We can do a non-parametric test.
- ▶ Q: How should we do that with K -samples?
- ▶ A: Permutation test!

Assumptions:

1. The observations are independent.
2. For each group, $j \in \{1, \dots, k\}$ the n_j observations are a random sample from a continuous distribution with distribution function F_j
3. The distribution functions F_1, \dots, F_k are connected through the relationship

$$F_j(t) = F(t - \tau_j),$$

where $j = 1, \dots, k$ and F is a distribution function for a continuous distribution with unknown median θ and τ_j is the unknown treatment effect for the j -th population. (Note: Under the third assumption here, the k underlying populations can differ by at most a shift in location. They cannot differ by scales parameters and must belong to the same general family.)

Permutation F -test

- ▶ Under the null hypothesis, each observation is equally likely to come from each group.
- ▶ So we randomly permute the observations between the different groups in the same numbers as the actual observations (i.e. If group 1 had 5 observation and group 2 has 7 actual observations, each permutation needs to have 5 observations in group 1 and 7 observations in group 2.)
- ▶ Once the observations have been permuted, we calculate the F -statistic and record it.
- ▶ Repeat this process many times, calculating and collecting the F -statistic each time.

Hypothesis

$$H_0 : \tau_1 = \tau_2 = \dots = \tau_k = 0$$

$$H_1 : \tau_1, \tau_2, \dots, \tau_k \text{ not all equal } 0.$$

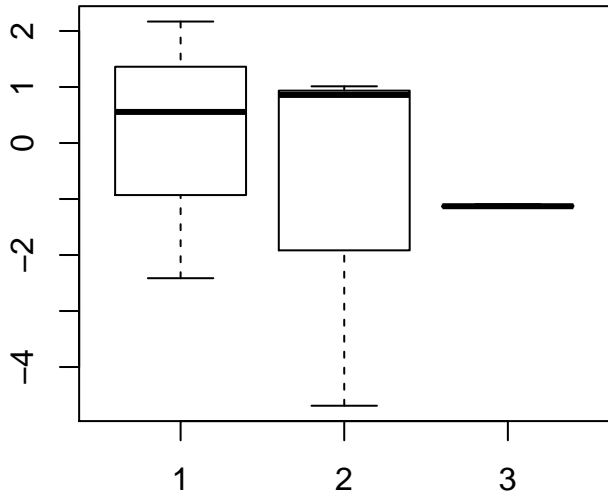
Permutation F -test

- ▶ This collection of F -statistics creates the null distribution.
- ▶ Therefore, we can calculate a p -value based on the fraction of these F statistics that are greater than or equal to the F -statistic that we actually observed.
- ▶ (Note: this is always a one-sided, upper-tailed test.)

Example 1

```
set.seed(1234)
dat<-data.frame(score=c(rnorm(3,0,2),rnorm(3,0,2),rnorm(3,0,2)),group=factor(c(rep(1,3),rep(2,3),rep(3,3))),
tapply(dat$score,dat$group,mean)
```

```
##           1           2           3
## 0.1032031 -0.9403447 -1.1238825
```



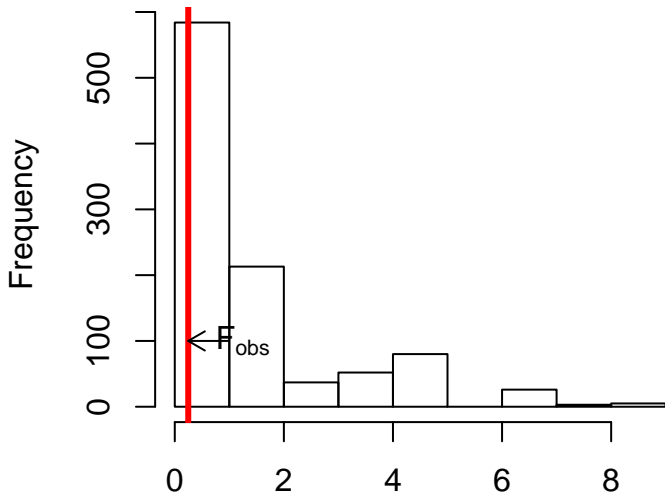
```
lmOut <- lm(score~group,data=dat)
anovaOut<-anova(lmOut)
Fobs<-anovaOut[1,4]
pval<-anovaOut[1,5]
#p-value if ANOVA assumptions are valid
pval

## [1] 0.7887257
```

```
#randomly permute the scores
n <- length(dat$score)
nsim<-1000
FstatVecNull<-rep(NA,nsim)
for (i in 1:nsim){print(i)
datPermute <- dat
datPermute$score <- dat$score[sample(1:n,n)]
lmOut <- lm(score~group,data=datPermute)
anovaOut <- anova(lmOut)
FstatVecNull[i]<-anovaOut[1,4]
}
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
## [1] 11
## [1] 12
## [1] 13
## [1] 14
## [1] 15
## [1] 16
## [1] 17
## [1] 18
## [1] 19
## [1] 20
## [1] 21
## [1] 22
## [1] 23
## [1] 24
## [1] 25
## [1] 26
```

Histogram of FstatVecNull



FstatVecNull

```
pvalPerm<-sum(Fobs<FstatVecNull)/nsim
```

```
#permutation p-value
```

```
pvalPerm
```

```
## [1] 0.832
```

```
#p-value from F distribution
```

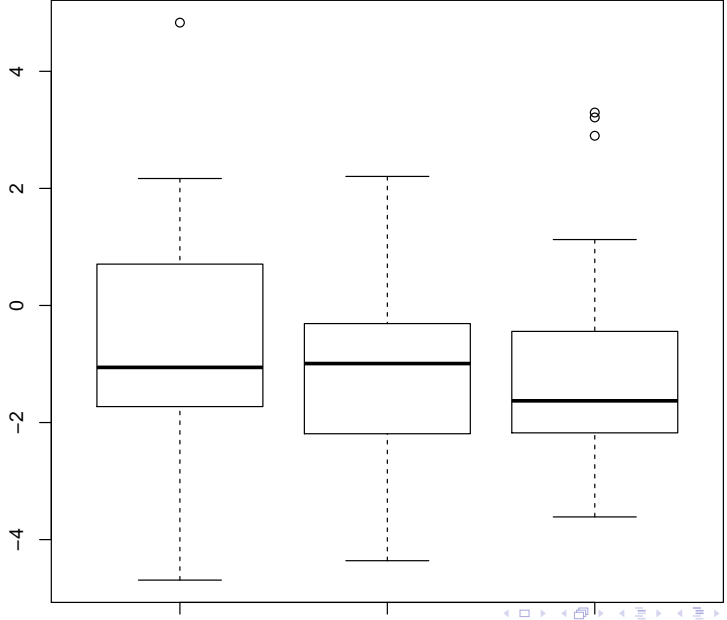
```
pval
```

```
## [1] 0.7887257
```


Example 2

```
set.seed(1234)
dat<-data.frame(score=c(rnorm(20,0,2),rnorm(20,0,2),rnorm(20,0,2)),group=factor(c(rep(1,20),rep(2,20),rep(3,20))),
tapply(dat$score,dat$group,mean)

##           1           2           3
## -0.5013281 -1.1541399 -0.8886618
```

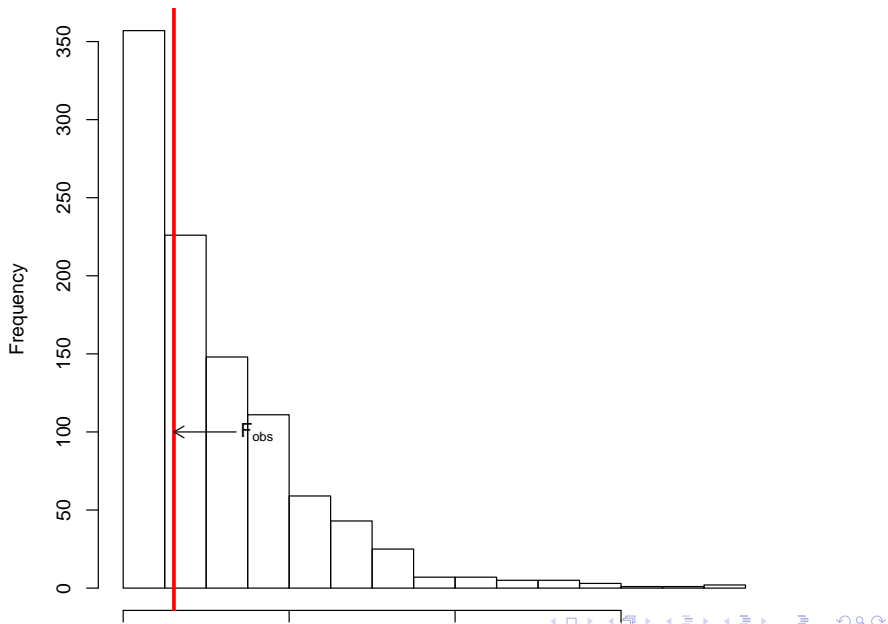


```
lmOut <- lm(score~group,data=dat)
anovaOut<-anova(lmOut)
Fobs<-anovaOut[1,4]
pval<-anovaOut[1,5]
#F test p-value
pval

## [1] 0.5459521
```

```
#randomly permute the scores
n <- length(dat$score)
nsim<-1000
FstatVecNull<-rep(NA,nsim)
for (i in 1:nsim){
  datPermute <- dat
  datPermute$score <- dat$score[sample(1:n,n)]
  lmOut <- lm(score~group,data=datPermute)
  anovaOut <- anova(lmOut)
  FstatVecNull[i]<-anovaOut[1,4]
}
```

Histogram of FstatVecNull



```
pvalPerm<-sum(Fobs<FstatVecNull)/nsim
```

```
#permutation p-value
```

```
pvalPerm
```

```
## [1] 0.575
```

```
#p-value F-test
```

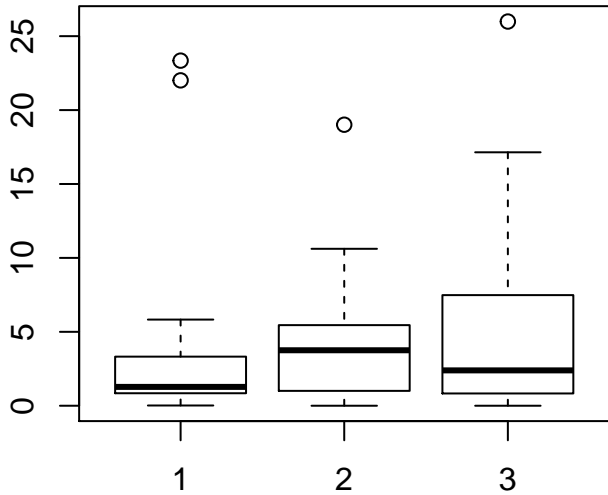
```
pval
```

```
## [1] 0.5459521
```

Example 3

```
set.seed(1234)
dat<-data.frame(score=c(rnorm(25,0,2)^2,rnorm(25,0,2)^2,rnorm(25,0,2)^2),group=factor(c(rep(1,25),rep(2,25),rep(3,25))))
tapply(dat$score,dat$group,mean)
```

```
##          1          2          3
## 3.508842 4.274302 5.451073
```

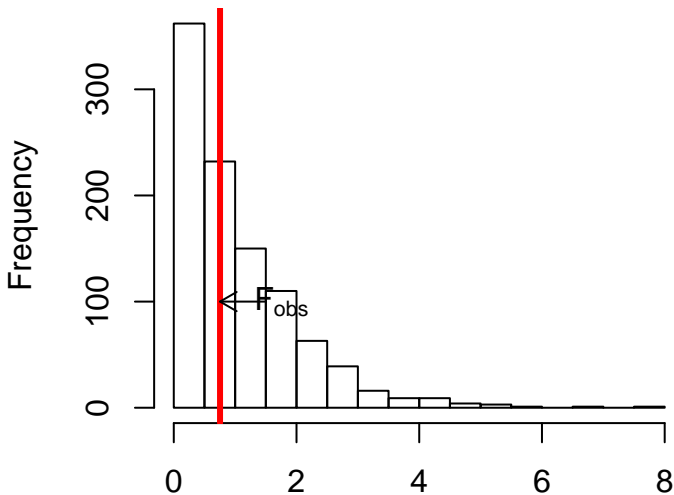



```
lmOut <- lm(score~group,data=dat)
anovaOut<-anova(lmOut)
Fobs<-anovaOut[1,4]
pval<-anovaOut[1,5]
pval

## [1] 0.4742715
```

```
#randomly permute the scores
n <- length(dat$score)
nsim<-1000
FstatVecNull<-rep(NA,nsim)
for (i in 1:nsim){
  datPermute <- dat
  datPermute$score <- dat$score[sample(1:n,n)]
  lmOut <- lm(score~group,data=datPermute)
  anovaOut <- anova(lmOut)
  FstatVecNull[i]<-anovaOut[1,4]
}
```

Histogram of FstatVecNull



FstatVecNull

```
pvalPerm<-sum(Fobs<FstatVecNull)/nsim
```

```
#permutation p-value
```

```
pvalPerm
```

```
## [1] 0.513
```

```
#F test p-value
```

```
pval
```

```
## [1] 0.4742715
```

- ▶ As in two sample testing where we moved from permutation tests to the Wilcoxon-Rank sum test, we will move from k -sample permutation tests to the Kruskal-Wallis test.
- ▶ Kruskal-Wallis is essentially a k -sample permutation test based on the ranks rather than the raw data.

The Kruskal-Wallis test statistic is:

$$H = \frac{12}{N(N+1)} \sum_{j=1}^k n_j \left(\bar{R}_{.j} - \frac{N+1}{2} \right)^2$$

where N is the total number of observations, k is the number of groups, $\bar{R}_{.j}$ is the mean of the ranks of the j -th group, and n_j is the number of observations in the j -th group.

The term $\frac{12}{N(N+1)}$ is a scaling factor and allows us to calculate a p-value by using a χ^2 distribution with $k-1$ degrees of freedom.

```
## Hollander & Wolfe (1973), 116.  
## Mucociliary efficiency from the rate of removal of dust in normal  
## subjects, subjects with obstructive airway disease, and subjects  
## with asbestosis.  
# normal subjects  
x <- c(2.9, 3.0, 2.5, 2.6, 3.2)  
# with obstructive airway disease  
y <- c(3.8, 2.7, 4.0, 2.4)  
# with asbestosis  
z <- c(2.8, 3.4, 3.7, 2.2, 2.0)
```



```
rate <- c(x, y, z)
g <- factor(rep(1:3, c(5, 4, 5)),
            labels = c("Normal subjects",
                        "Subjects with obstructive airway disease",
                        "Subjects with asbestosis"))
dat<-data.frame(rate=rate,group=g)
dat$rank<-rank(dat$rate)
```

```
Rbar<-tapply(dat$rank,dat$group,mean)
n<-tapply(dat$rank,dat$group,length)
N<-sum(n)
kwStat<-12/(N*(N+1))*sum(n*(Rbar-(N+1)/2)^2)
kwStat

## [1] 0.7714286

#Chi square approximation.
(pval<-1-pchisq(kwStat,2))

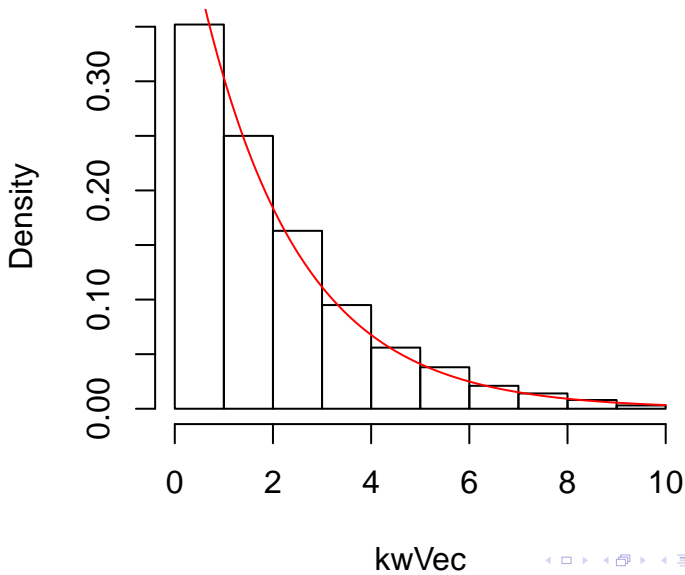
## [1] 0.6799648
```

```

#randomly permute the scores
N <- length(dat$rate)
n<-tapply(dat$ranks,dat$group,length)
nsim<-1000
kwVec<-rep(NA,nsim)
for (i in 1:nsim){
  datPermute <- dat
  datPermute$ranks <- dat$ranks[sample(1:N,N)]
  Rbar<-tapply(datPermute$ranks,datPermute$group,mean)
  kwVec[i]<-12/(N*(N+1))*sum(n*(Rbar-(N+1)/2)^2)
}

```

Histogram of kwVec



```
pvalKW<-sum(kwStat<kwVec)/nsim
```

```
#permutation p-value
```

```
pvalKW
```

```
## [1] 0.698
```

```
#chi-square approximation
```

```
pval
```

```
## [1] 0.6799648
```

```
kruskal.test(list(x,y,z))
```

```
##
```

```
## Kruskal-Wallis rank sum test
```

```
##
```

```
## data: list(x, y, z)
```

```
## Kruskal-Wallis chi-squared = 0.7714, df = 2, p-value = 0.68
```

When to do what?

- ▶ Kruskal-Wallis preferred
 - ▶ Potential outliers in the data
 - ▶ Data has heavy tails
 - ▶ Data distribution is significantly skewed
- ▶ ANOVA F-test preferred
 - ▶ Data distribution is approximately normal
 - ▶ Data distribution is light tailed and symmetric

- ▶ Alright. So what happens if we reject the null hypothesis?
- ▶ Then we need to do multiple comparisons.
- ▶ Rejecting the null hypothesis means that some groups are different. Let's answer the question of WHICH ones are different.

- ▶ One way to do this is by performing multiple comparisons.
- ▶ To do this we compare all pairs of treatments to look for pair-wise differences.
- ▶ This can be done by using the Wilcoxon rank-sum test.
- ▶ How many pairs are there?
- ▶ If there are k groups then there are $\frac{k(k-1)}{2}$ pairs.

- ▶ HOWEVER, we need to make some adjustments because we are performing multiple tests.
- ▶ WHY? What is the PROBLEM?
- ▶ If we do a large number of tests all at $\alpha = 0.05$ level, we are bound to reject some of the null hypotheses totally by chance.
- ▶ Therefore, we want to control the family-wise (or experiment-wise) error rate.

- ▶ Rather than controlling the probability of making a type I error in any given test at level α
- ▶ We want to control the probability of making ANY type I error at level α .

```

#A quick experiment
set.seed(1234)
#repeat experiment 10000 times under the null
test<-cbind(rbinom(10000,1,0.05),rbinom(10000,1,0.05),rbinom(10000,1,0.05))
#Control each test at 0.05
apply(test,2,mean)

## [1] 0.0477 0.0481 0.0509

table(apply(test,1,sum))

##
##      0      1      2      3
## 8593 1349      56      2

#Probability of making ANY type 1 Error
mean(apply(test,1,sum)>0)

## [1] 0.1407

```

- ▶ So how do we control family-wise error rate?
 - ▶ Bonferroni Correction
 - ▶ Fisher's Protect Least Significant Difference (LSD)
 - ▶ Tukey's Honest Significant Difference (HSD) Procedure

Bonferroni

- ▶ Perform each test at $\alpha' = \frac{\alpha}{\frac{k(k-1)}{2}}$
- ▶ If $k=4$, there are 6 pairwise experiments so $\alpha' = \frac{0.05}{6} = 0.0083$

LSD

- ▶ With normality assumption:

$$|\bar{X}_i - \bar{X}_j| \geq t\left(\frac{\alpha}{2}, df = N - k\right) \sqrt{MSE\left(\frac{1}{n_i} + \frac{1}{n_j}\right)}$$

- ▶ Non-parametric version:

$$|\bar{R}_i - \bar{R}_j| \geq z\left(\frac{\alpha}{2}, df = N - k\right) \sqrt{\frac{N(N+1)}{12} \left(\frac{1}{n_i} + \frac{1}{n_j}\right)}$$

- ▶ This procedure is “protected” in the sense that we only perform this test if we reject the global test.

Tukey's HSD

- ▶ Tukey asked the question, what is the largest pairwise difference that we expect to observe by chance.
- ▶ Declare a difference to be significant if it is larger than the largest difference that we expect to see by chance.


```

#Table 3.3.1 from book
#Percentage of Clay in Soil Samples
clay<-data.frame(pct=c(26.5,15.0,18.2,19.5,23.1,17.3,16.5,15.8,14.1,30.2,25.1,17.4,19.2,21.4,26.0,21.6,35.0,28.7,22.5,20.1,24.3,27.8,29.9,25.5,22.2,26.8,23.9,21.1,25.2,28.4,24.6,27.1,29.3,26.2,23.5,27.6,24.8,28.1,25.7,22.9,26.5,23.2,27.3,24.1,28.6,25.4,22.7,26.9,23.8,27.5,24.5,28.3,25.6,22.8,27.0,23.7,27.2,24.2,28.5,25.3,22.6,27.4,23.6,27.7,24.4,28.2,25.5,22.9,27.1,23.5,27.8,24.7,28.0,25.2,22.7,27.3,23.4,27.9,24.9,28.7,25.1,22.5,27.5,23.3,28.1,25.0,22.4,27.6,23.1,28.4,25.4,22.3,27.7,23.0,28.6,25.6,22.2,27.9,23.2,28.9,25.8,22.1,28.1,23.4,29.1,26.0,22.0,28.3,23.7,29.3,26.2,21.9,28.5,24.0,21.7,28.7,24.3,21.5,28.9,24.6,21.3,29.1,24.9,21.1,29.3,25.2,20.9,29.5,25.5,20.7,29.7,25.8,20.5,29.9,26.1,20.3,30.1,26.4,20.1,30.3,26.7,20.0,30.5,27.0,20.0,30.7,27.3,20.0,30.9,27.6,20.0,31.1,27.9,20.0,31.3,28.2,20.0,31.5,28.5,20.0,31.7,28.8,20.0,31.9,29.1,20.0,32.1,29.4,20.0,32.3,29.7,20.0,32.5,30.0,20.0,32.7,30.3,20.0,32.9,30.6,20.0,33.1,30.9,20.0,33.3,31.2,20.0,33.5,31.5,20.0,33.7,31.8,20.0,33.9,32.1,20.0,34.1,32.4,20.0,34.3,32.7,20.0,34.5,33.0,20.0,34.7,33.3,20.0,34.9,33.6,20.0,35.1,33.9,20.0,35.3,34.2,20.0,35.5,34.5,20.0,35.7,34.8,20.0,35.9,35.1,20.0,36.1,35.4,20.0,36.3,35.7,20.0,36.5,36.0,20.0,36.7,36.3,20.0,36.9,36.6,20.0,37.1,36.9,20.0,37.3,37.2,20.0,37.5,37.5,20.0,37.7,37.8,20.0,37.9,38.1,20.0,38.1,38.4,20.0,38.3,38.7,20.0,38.5,39.0,20.0,38.7,39.3,20.0,38.9,39.6,20.0,39.1,39.9,20.0,39.3,40.2,20.0,39.5,40.5,20.0,39.7,40.8,20.0,39.9,41.1,20.0,40.1,41.4,20.0,40.3,41.7,20.0,40.5,42.0,20.0,40.7,42.3,20.0,40.9,42.6,20.0,41.1,42.9,20.0,41.3,43.2,20.0,41.5,43.5,20.0,41.7,43.8,20.0,41.9,44.1,20.0,42.1,44.4,20.0,42.3,44.7,20.0,42.5,45.0,20.0,42.7,45.3,20.0,42.9,45.6,20.0,43.1,45.9,20.0,43.3,46.2,20.0,43.5,46.5,20.0,43.7,46.8,20.0,43.9,47.1,20.0,44.1,47.4,20.0,44.3,47.7,20.0,44.5,48.0,20.0,44.7,48.3,20.0,44.9,48.6,20.0,45.1,48.9,20.0,45.3,49.2,20.0,45.5,49.5,20.0,45.7,49.8,20.0,45.9,50.1,20.0,46.1,50.4,20.0,46.3,50.7,20.0,46.5,51.0,20.0,46.7,51.3,20.0,46.9,51.6,20.0,47.1,51.9,20.0,47.3,52.2,20.0,47.5,52.5,20.0,47.7,52.8,20.0,47.9,53.1,20.0,48.1,53.4,20.0,48.3,53.7,20.0,48.5,54.0,20.0,48.7,54.3,20.0,48.9,54.6,20.0,49.1,54.9,20.0,49.3,55.2,20.0,49.5,55.5,20.0,49.7,55.8,20.0,49.9,56.1,20.0,50.1,56.4,20.0,50.3,56.7,20.0,50.5,57.0,20.0,50.7,57.3,20.0,50.9,57.6,20.0,51.1,57.9,20.0,51.3,58.2,20.0,51.5,58.5,20.0,51.7,58.8,20.0,51.9,59.1,20.0,52.1,59.4,20.0,52.3,59.7,20.0,52.5,60.0,20.0,52.7,60.3,20.0,52.9,60.6,20.0,53.1,60.9,20.0,53.3,61.2,20.0,53.5,61.5,20.0,53.7,61.8,20.0,53.9,62.1,20.0,54.1,62.4,20.0,54.3,62.7,20.0,54.5,63.0,20.0,54.7,63.3,20.0,54.9,63.6,20.0,55.1,63.9,20.0,55.3,64.2,20.0,55.5,64.5,20.0,55.7,64.8,20.0,55.9,65.1,20.0,56.1,65.4,20.0,56.3,65.7,20.0,56.5,66.0,20.0,56.7,66.3,20.0,56.9,66.6,20.0,57.1,66.9,20.0,57.3,67.2,20.0,57.5,67.5,20.0,57.7,67.8,20.0,57.9,68.1,20.0,58.1,68.4,20.0,58.3,68.7,20.0,58.5,69.0,20.0,58.7,69.3,20.0,58.9,69.6,20.0,59.1,69.9,20.0,59.3,70.2,20.0,59.5,70.5,20.0,59.7,70.8,20.0,59.9,71.1,20.0,60.1,71.4,20.0,60.3,71.7,20.0,60.5,72.0,20.0,60.7,72.3,20.0,60.9,72.6,20.0,61.1,72.9,20.0,61.3,73.2,20.0,61.5,73.5,20.0,61.7,73.8,20.0,61.9,74.1,20.0,62.1,74.4,20.0,62.3,74.7,20.0,62.5,75.0,20.0,62.7,75.3,20.0,62.9,75.6,20.0,63.1,75.9,20.0,63.3,76.2,20.0,63.5,76.5,20.0,63.7,76.8,20.0,63.9,77.1,20.0,64.1,77.4,20.0,64.3,77.7,20.0,64.5,78.0,20.0,64.7,78.3,20.0,64.9,78.6,20.0,65.1,78.9,20.0,65.3,79.2,20.0,65.5,79.5,20.0,65.7,79.8,20.0,65.9,80.1,20.0,66.1,80.4,20.0,66.3,80.7,20.0,66.5,81.0,20.0,66.7,81.3,20.0,66.9,81.6,20.0,67.1,81.9,20.0,67.3,82.2,20.0,67.5,82.5,20.0,67.7,82.8,20.0,67.9,83.1,20.0,68.1,83.4,20.0,68.3,83.7,20.0,68.5,84.0,20.0,68.7,84.3,20.0,68.9,84.6,20.0,69.1,84.9,20.0,69.3,85.2,20.0,69.5,85.5,20.0,69.7,85.8,20.0,69.9,86.1,20.0,70.1,86.4,20.0,70.3,86.7,20.0,70.5,87.0,20.0,70.7,87.3,20.0,70.9,87.6,20.0,71.1,87.9,20.0,71.3,88.2,20.0,71.5,88.5,20.0,71.7,88.8,20.0,71.9,89.1,20.0,72.1,89.4,20.0,72.3,89.7,20.0,72.5,90.0,20.0,72.7,90.3,20.0,72.9,90.6,20.0,73.1,90.9,20.0,73.3,91.2,20.0,73.5,91.5,20.0,73.7,91.8,20.0,73.9,92.1,20.0,74.1,92.4,20.0,74.3,92.7,20.0,74.5,93.0,20.0,74.7,93.3,20.0,74.9,93.6,20.0,75.1,93.9,20.0,75.3,94.2,20.0,75.5,94.5,20.0,75.7,94.8,20.0,75.9,95.1,20.0,76.1,95.4,20.0,76.3,95.7,20.0,76.5,96.0,20.0,76.7,96.3,20.0,76.9,96.6,20.0,77.1,96.9,20.0,77.3,97.2,20.0,77.5,97.5,20.0,77.7,97.8,20.0,77.9,98.1,20.0,78.1,98.4,20.0,78.3,98.7,20.0,78.5,99.0,20.0,78.7,99.3,20.0,78.9,99.6,20.0,79.1,100.0,20.0,79.3,100.3,20.0,79.5,100.6,20.0,79.7,100.9,20.0,79.9,101.2,20.0,80.1,101.5,20.0,80.3,101.8,20.0,80.5,102.1,20.0,80.7,102.4,20.0,80.9,102.7,20.0,81.1,103.0,20.0,81.3,103.3,20.0,81.5,103.6,20.0,81.7,103.9,20.0,81.9,104.2,20.0,82.1,104.5,20.0,82.3,104.8,20.0,82.5,105.1,20.0,82.7,105.4,20.0,82.9,105.7,20.0,83.1,106.0,20.0,83.3,106.3,20.0,83.5,106.6,20.0,83.7,106.9,20.0,83.9,107.2,20.0,84.1,107.5,20.0,84.3,107.8,20.0,84.5,108.1,20.0,84.7,108.4,20.0,84.9,108.7,20.0,85.1,109.0,20.0,85.3,109.3,20.0,85.5,109.6,20.0,85.7,110.0,20.0,85.9,110.3,20.0,86.1,110.6,20.0,86.3,110.9,20.0,86.5,111.2,20.0,86.7,111.5,20.0,86.9,111.8,20.0,87.1,112.1,20.0,87.3,112.4,20.0,87.5,112.7,20.0,87.7,113.0,20.0,87.9,113.3,20.0,88.1,113.6,20.0,88.3,113.9,20.0,88.5,114.2,20.0,88.7,114.5,20.0,88.9,114.8,20.0,89.1,115.1,20.0,89.3,115.4,20.0,89.5,115.7,20.0,89.7,116.0,20.0,89.9,116.3,20.0,90.1,116.6,20.0,90.3,116.9,20.0,90.5,117.2,20.0,90.7,117.5,20.0,90.9,117.8,20.0,91.1,118.1,20.0,91.3,118.4,20.0,91.5,118.7,20.0,91.7,119.0,20.0,91.9,119.3,20.0,92.1,119.6,20.0,92.3,120.0,20.0,92.5,120.3,20.0,92.7,120.6,20.0,92.9,120.9,20.0,93.1,121.2,20.0,93.3,121.5,20.0,93.5,121.8,20.0,93.7,122.1,20.0,93.9,122.4,20.0,94.1,122.7,20.0,94.3,123.0,20.0,94.5,123.3,20.0,94.7,123.6,20.0,94.9,123.9,20.0,95.1,124.2,20.0,95.3,124.5,20.0,95.5,124.8,20.0,95.7,125.1,20.0,95.9,125.4,20.0,96.1,125.7,20.0,96.3,126.0,20.0,96.5,126.3,20.0,96.7,126.6,20.0,96.9,126.9,20.0,97.1,127.2,20.0,97.3,127.5,20.0,97.5,127.8,20.0,97.7,128.1,20.0,97.9,128.4,20.0,98.1,128.7,20.0,98.3,129.0,20.0,98.5,129.3,20.0,98.7,129.6,20.0,98.9,130.0,20.0,99.1,130.3,20.0,99.3,130.6,20.0,99.5,130.9,20.0,99.7,131.2,20.0,99.9,131.5,20.0,100.1,131.8,20.0,100.3,132.1,20.0,100.5,132.4,20.0,100.7,132.7,20.0,100.9,133.0,20.0,101.1,133.3,20.0,101.3,133.6,20.0,101.5,133.9,20.0,101.7,134.2,20.0,101.9,134.5,20.0,102.1,134.8,20.0,102.3,135.1,20.0,102.5,135.4,20.0,102.7,135.7,20.0,102.9,136.0,20.0,103.1,136.3,20.0,103.3,136.6,20.0,103.5,136.9,20.0,103.7,137.2,20.0,103.9,137.5,20.0,104.1,137.8,20.0,104.3,138.1,20.0,104.5,138.4,20.0,104.7,138.7,20.0,104.9,139.0,20.0,105.1,139.3,20.0,105.3,139.6,20.0,105.5,140.0,20.0,105.7,140.3,20.0,105.9,140.6,20.0,106.1,140.9,20.0,106.3,141.2,20.0,106.5,141.5,20.0,106.7,141.8,20.0,106.9,142.1,20.0,107.1,142.4,20.0,107.3,142.7,20.0,107.5,143.0,20.0,107.7,143.3,20.0,107.9,143.6,20.0,108.1,143.9,20.0,108.3,144.2,20.0,108.5,144.5,20.0,108.7,144.8,20.0,108.9,145.1,20.0,109.1,145.4,20.0,109.3,145.7,20.0,109.5,146.0,20.0,109.7,146.3,20.0,109.9,146.6,20.0,110.1,146.9,20.0,110.3,147.2,20.0,110.5,147.5,20.0,110.7,147.8,20.0,110.9,148.1,20.0,111.1,148.4,20.0,111.3,148.7,20.0,111.5,149.0,20.0,111.7,149.3,20.0,111.9,149.6,20.0,112.1,150.0,20.0,112.3,150.3,20.0,112.5,150.6,20.0,112.7,150.9,20.0,112.9,151.2,20.0,113.1,151.5,20.0,113.3,151.8,20.0,113.5,152.1,20.0,113.7,152.4,20.0,113.9,152.7,20.0,114.1,153.0,20.0,114.3,153.3,20.0,114.5,153.6,20.0,114.7,153.9,20.0,114.9,154.2,20.0,115.1,154.5,20.0,115.3,154.8,20.0,115.5,155.1,20.0,115.7,155.4,20.0,115.9,155.7,20.0,116.1,156.0,20.0,116.3,156.3,20.0,116.5,156.6,20.0,116.7,156.9,20.0,116.9,157.2,20.0,117.1,157.5,20.0,117.3,157.8,20.0,117.5,158.1,20.0,117.7,158.4,20.0,117.9,158.7,20.0,118.1,159.0,20.0,118.3,159.3,20.0,118.5,159.6,20.0,118.7,160.0,20.0,118.9,160.3,20.0,119.1,160.6,20.0,119.3,160.9,20.0,119.5,161.2,20.0,119.7,161.5,20.0,119.9,161.8,20.0,120.1,162.1,20.0,120.3,162.4,20.0,120.5,162.7,20.0,120.7,163.0,20.0,120.9,163.3,20.0,121.1,163.6,20.0,121.3,163.9,20.0,121.5,164.2,20.0,121.7,164.5,20.0,121.9,164.8,20.0,122.1,165.1,20.0,122.3,165.4,20.0,122.5,165.7,20.0,122.7,166.0,20.0,122.9,166.3,20.0,123.1,166.6,20.0,123.3,166.9,20.0,123.5,167.2,20.0,123.7,167.5,20.0,123.9,167.8,20.0,124.1,168.1,20.0,124.3,168.4,20.0,124.5,168.7,20.0,124.7,169.0,20.0,124.9,169.3,20.0,125.1,169.6,20.0,125.3,170.0,20.0,125.5,170.3,20.0,125.7,170.6,20.0,125.9,170.9,20.0,126.1,171.2,20.0,126.3,171.5,20.0,126.5,171.8,20.0,126.7,172.1,20.0,126.9,172.4,20.0,127.1,172.7,20.0,127.3,173.0,20.0,127.5,173.3,20.0,127.7,173.6,20.0,127.9,173.9,20.0,128.1,174.2,20.0,128.3,174.5,20.0,128.5,174.8,20.0,128.7,175.1,20.0,128.9,175.4,20.0,129.1,175.7,20.0,129.3,176.0,20.0,129.5,176.3,20.0,129.7,176.6,20.0,129.9,176.9,20.0,130.1,177.2,20.0,130.3,177.5,20.0,130.5,177.8,20.0,130.7,178.1,20.0,130.9,178.4,20.0,131.1,178.7,20.0,131.3,179.0,20.0,131.5,179.3,20.0,131.7,179.6,20.0,131.9,180.0,20.0,132.1,180.3,20.0,132.3,180.6,20.0,132.5,180.9,20.0,132.7,181.2,20.0,132.9,181.5,20.0,133.1,181.8,20.0,133.3,182.1,20.0,133.5,182.4,20.0,133.7,182.7,20.0,133.9,183.0,20.0,134.1,183.3,20.0,134.3,183.6,20.0,134.5,183.9,20.0,134.7,184.2,20.0,134.9,184.5,20.0,135.1,184.8,20.0,135.3,185.1,20.0,135.5,185.4,20.0,135.7,185.7,20.0,135.9,186.0,20.0,136.1,186.3,20.0,136.3,186.6,20.0,136.5,186.9,20.0,136.7,187.2,20.0,136.9,187.5,20.0,137.1,187.8,20.0,137.3,188.1,20.0,137.5,188.4,20.0,137.7,188.7,20.0,137.9,189.0,20.0,138.1,189.3,20.0,138.3,189.6,20.0,138.5,190.0,20.0,138.7,190.3,20.0,138.9,190.6,20.0,139.1,190.9,20.0,139.3,191.2,20.0,139.5,191.5,20.0,139.7,191.8,20.0,139.9,192.1,20.0,140.1,192.4,20.0,140.3,192.7,20.0,140.5,193.0,20.0,140.7,193.3,20.0,140.9,193.6,20.0,141.1,193.9,20.0,141.3,194.2,20.0,141.5,194.5,20.0,141.7,194.8,20.0,141.9,195.1,20.0,142.1,195.4,20.0,142.3,195.7,20.0,142.5,196.0,20.0,142.7,196.3,20.0,142.9,196.6,20.0,143.1,196.9,20.0,143.3,197.2,20.0,143.5,197.5,20.0,143.7,197.8,20.0,143.9,198.1,20.0,144.1,198.4,20.0,144.3,198.7,20.0,144.5,199.0,20.0,144.7,199.3,20.0,144.9,199.6,20.0,145.1,200.0,20.0,145.3,200.3,20.0,145.5,200.6,20.0,145.7,200.9,20.0,145.9,201.2,20.0,146.1,201.5,20.0,146.3,201.8,20.0,146.5,202.1,20.0,146.7,202.4,20.0,146.9,202.7,20.0,147.1,203.0,20.0,147.3,203.3,20.0,147.5,203.6,20.0,147.7,203.9,20.0,147.9,204.2,20.0,148.1,204.5,20.0,148.3,204.8,20.0,148.5,205.1,20.0,148.7,205.4,20.0,148.9,205.7,20.0,149.1,206.0,20.0,149.3,206.3,20.0,149.5,206.6,20.0,149.7,206.9,20.0,149.9,207.2,20.0,150.1,207.5,20.0,150.3,207.8,20.0,150.5,208.1,20.0,150.7,208.4,20.0,150.9,208.7,20.0,151.1,209.0,20.0,151.3,209.3,20.0,151.5,209.6,20.0,151.7,210.0,20.0,151.9,210.3,20.0,152.1,210.6,20.0,152.3,210.9,20.0,152.5,211.2,20.0,152.7,211.5,20.0,152.9,211.8,20.0,153.1,212.1,20.0,153.3,212.4,20.0,153.5,212.7,20.0,153.7,213.0,20.0,153.9,213.3,20.0,154.1,213.6,20.0,154.3,213.9,20.0,154.5,214.2,20.0,154.7,214.5,20.0,154.9,214.8,20.0,155.1,215.1,20.0,155.3,215.4,20.0,155.5,215.7,20.0,155.7,216.0,20.0,155.9,216.3,20.0,156.1,216.6,20.0,156.3,216.9,20.0,156.5,217.2,20.0,156.7,217.5,20.0,156.9,217.8,20.0,157.1,218.1,20.0,157.3,218.4,20.0,157.5,218.7,20.0,157.7,219.0,20.0,157.9,219.3,20.0,158.1,219.6,20.0,158.3,220.0,20.0,158.5,220.3,20.0,158.7,220.6,20.0,158.9,220.9,20.0,159.1,221.2,20.0,159.3,221.5,20.0,159.5,221.8,20.0,159.7,222.1,20.0,159.9,222.4,20.0,160.1,222.7,20.0,160.3,223.0,20.0,160.5,223.3,20.0,160.7,223.6,20.0,160.9,223.9,20.0,161.1,224.2,20.0,161.3,224.5,20.0,161.5,224.8,20.0,161.7,225.1,20.0,161.9,225.4,20.0,162.1,225.7,20.0,162.3,226.0,20.0,162.5,226.3,20.0,162.7,226.6,20.0,162.9,226.9,20.0,163.1,227.2,20.0,163.3,227.5,20.0,163.5,227.8,20.0,163.7,228.1,20.0,163.9,228.4,20.0,164.1,228.7,20.0,164.3,229.0,20.0,164.5,229.3,20.0,164.7,229.6,20.0,164.9,230.0,20.0,165.1,230.3,20.0,165.3,230.6,20.0,165.5,230.9,20.0,165.7,231.2,20.0,165.9,231.5,20.0,166.1,231.8,20.0,166.3,232.1,20.0,166.5,232.4,20.0,166.7,232.7,20.0,166.9,233.0,20.0,167.1,233.3,20.0,167.3,233.6,20.0,167.5,233.9,20.0,167.7,234.2,20.0,167.9,234.5,20.0,168.1,234.8,20.0,168.3,235.1,20.0,168.5,235.4,20.0,168.7,235.7,20.0,168.9,236.0,20.0,169.1,236.3,20.0,169.3,236.6,20.0,169.5,236.9,20.0,169.7,237.2,20.0,169.9,237.5,20.0,170.1,237.8,20.0,170.3,238.1,20.0,170.5,238.4,20.0,170.7,238.7,20.0,170.9,239.0,20.0,171.1,239.3,20.0,171.3,239.6,20.0,171.5,240.0,20.0,171.7,240.3,20.0,171.9,240.6,20.0,172.1,240.9,20.0,172.3,241.2,20.0,172.5,241.5,20.0,172.7,241.8,20.0,172.9,242.1,20.0,173.1,242.4,20.0,173.3,242.7,20.0,173.5,243.0,20.0,173.7,243.3,20.0,173.9,243.6,20.0,1
```

```
#Bonferroni cut-off
```

```
k<-4
```

```
alpha<-0.05
```

```
#cut-off
```

```
alpha/(k*(k-1)/2)
```

```
## [1] 0.008333333
```

```
wilcox.test(clay$pct[clay$group==1],clay$pct[clay$group==2])$p.value
```

```
## [1] 0.6991342
```

```
wilcox.test(clay$pct[clay$group==1],clay$pct[clay$group==3])$p.value
```

```
## [1] 0.1320346
```

```
wilcox.test(clay$pct[clay$group==1],clay$pct[clay$group==4])$p.value
```

```
## [1] 0.004329004
```

```
wilcox.test(clay$pct[clay$group==2],clay$pct[clay$group==3])$p.value
```

```
## [1] 0.1320346
```

```
wilcox.test(clay$pct[clay$group==2],clay$pct[clay$group==4])$p.value
```

```
## [1] 0.02597403
```

```
wilcox.test(clay$pct[clay$group==3],clay$pct[clay$group==4])$p.value
```

```
## [1] 0.1320346
```

```
#Do multiple comparisons.  
pairwise.wilcox.test(clay$pct,clay$group,p.adj="bonf")
```

```
##  
## Pairwise comparisons using Wilcoxon rank sum test  
##  
## data: clay$pct and clay$group  
##  
##      1      2      3  
## 2 1.000 -      -  
## 3 0.792 0.792 -  
## 4 0.026 0.156 0.792  
##  
## P value adjustment method: bonferroni
```

```
pairwise.wilcox.test(clay$pct,clay$group)
```

```
##  
## Pairwise comparisons using Wilcoxon rank sum test  
##  
## data: clay$pct and clay$group  
##  
##      1      2      3  
## 2 0.699 -      -  
## 3 0.528 0.528 -  
## 4 0.026 0.130 0.528  
##  
## P value adjustment method: holm
```

```
(meanRanks<-as.vector(tapply(clay$rank,clay$group,mean)))
```

```
## [1] 8.50000 8.00000 14.16667 19.33333
```

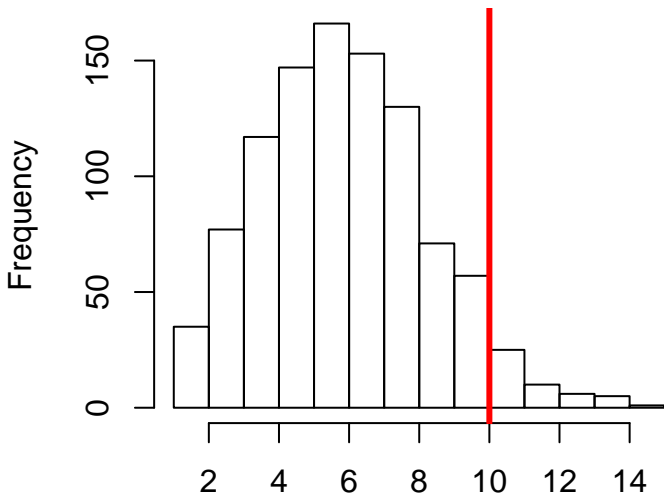
```
abs(outer(meanRanks,meanRanks,"-"))
```

```
##           [,1]      [,2]      [,3]      [,4]
## [1,] 0.000000 0.500000 5.666667 10.833333
## [2,] 0.500000 0.000000 6.166667 11.333333
## [3,] 5.666667 6.166667 0.000000 5.166667
## [4,] 10.833333 11.333333 5.166667 0.000000
```

```
pwDiffs<-(abs(outer(meanRanks,meanRanks,"-")))[upper.tri((a
```

```
nsim<-1000
maxDiff<-rep(NA,nsim)
N<-length(clay$pct)
for (i in 1:nsim){
  clayPermute<-clay
  clayPermute$rank<-clayPermute$rank[sample(1:N,N)]
  meanRanks<-as.vector(tapply(clayPermute$rank,clayPermute$group,mean))
  pwDiffs<-(abs(outer(meanRanks,meanRanks,"-")))[upper.tri((abs(outer(meanRanks,meanRanks,"-"))))]
  maxDiff[i]<-max(pwDiffs)
}
```

Histogram of maxDiff



maxDiff

#10.166667 is the cut-off for significance

```
(meanRanks<-as.vector(tapply(clay$rank,clay$group,mean)))
```

```
## [1] 8.50000 8.00000 14.16667 19.33333
```

```
abs(outer(meanRanks,meanRanks,"-"))
```

```
##           [,1]      [,2]      [,3]      [,4]
## [1,] 0.000000 0.500000 5.666667 10.833333
## [2,] 0.500000 0.000000 6.166667 11.333333
## [3,] 5.666667 6.166667 0.000000 5.166667
## [4,] 10.833333 11.333333 5.166667 0.000000
```