Non-parametric Statistics: Notes 3
K-sample methods

Gregory Matthews [1]

[1]Department of Mathematics and Statistics
Loyola University Chicago

Spring 2016

Outline

- 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 is ANOVA?

$H_0 : \mu_1 = \mu_2 = ... = \mu_j$ 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!

$$SST(orSSR) = \Sigma_{i=1}^{k} n_i(\bar{X}_i - \bar{X})^2$$

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

Then

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

and

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

and finally

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

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

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

where $j = 1, ..., k$ 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 obervations 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

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

$$H_1 : \tau_1, \tau_2, ..., \tau_k notallequal 0.$$

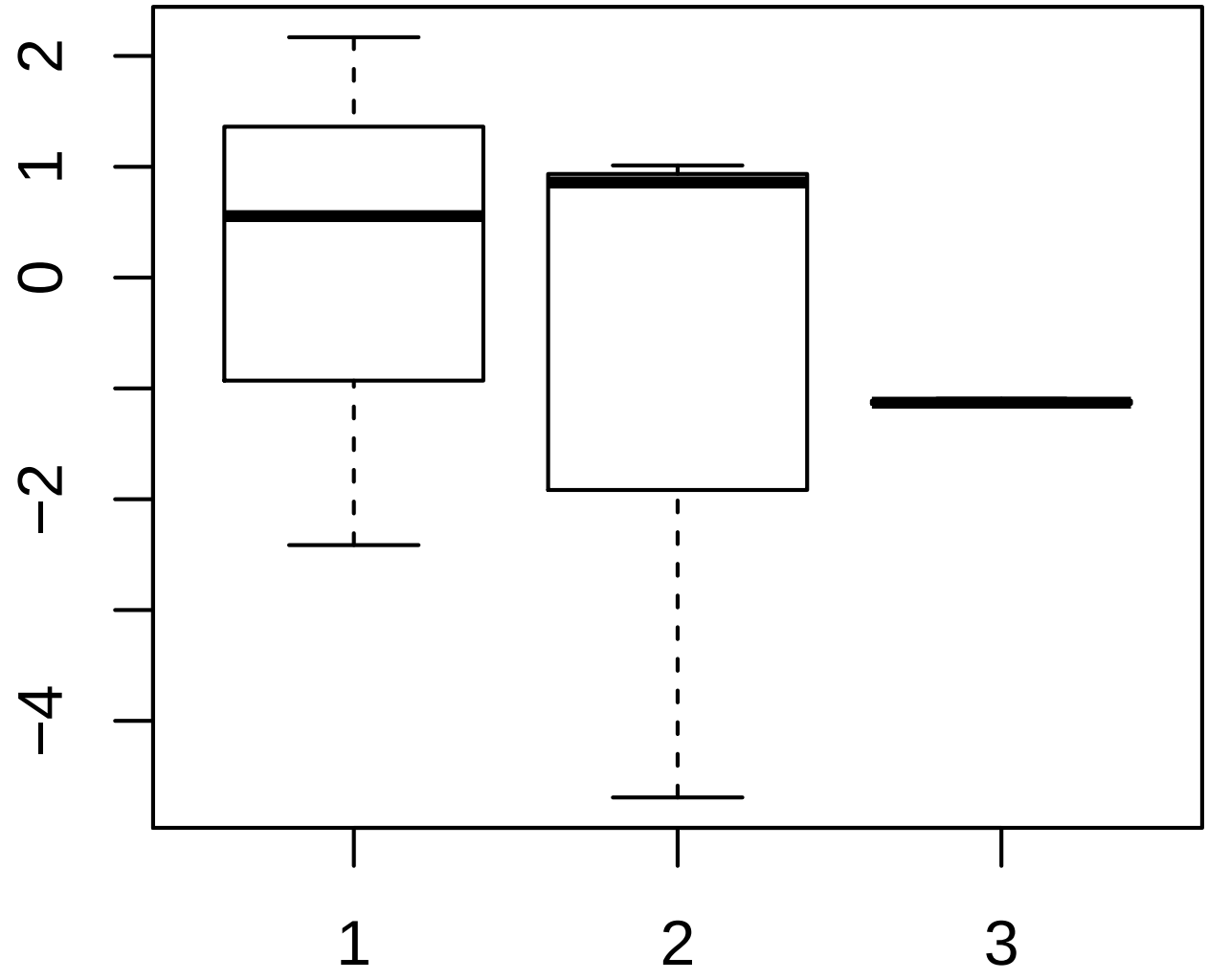
Permutation F-test

- 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)
```

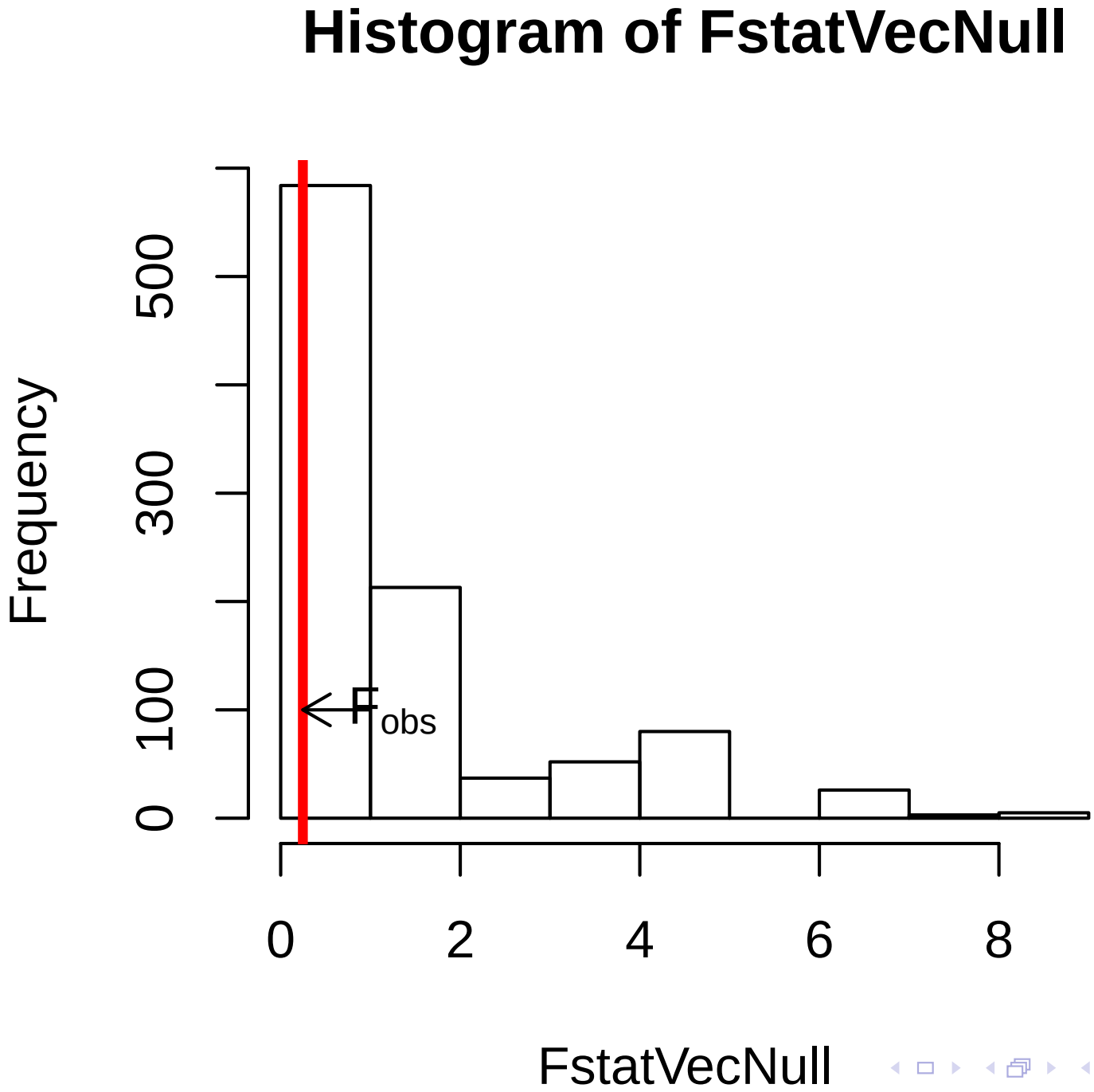
```
##          1          2          3
##  0.1032031 -0.9403447 -1.1238825
```


2
1
0
−2
−4
1
2
3

```r
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
```

```r
#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]
}
```

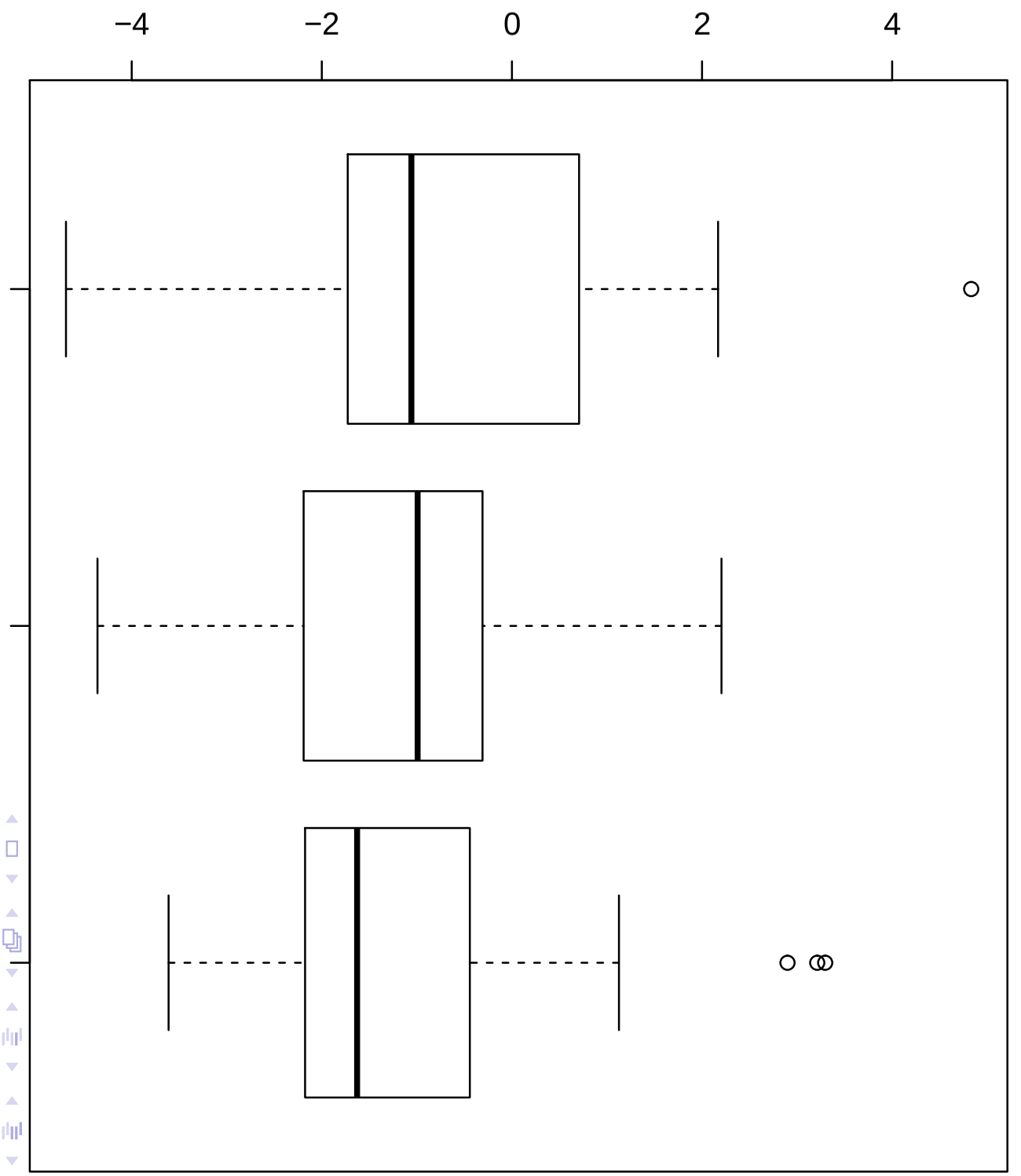
```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
## [1] 11
## [1] 12
## [1] 13
## [1] 14
## [1] 15
## [1] 16
## [1] 17
## [1] 18
## [1] 19
## [1] 20
## [1] 21
## [1] 22
## [1] 23
## [1] 24
## [1] 25
```


Histogram of FstatVecNull
Frequency
0
100
300
500
F_{obs}
0
2
4
6
8
FstatVecNull

```r
pvalPerm<-sum(Fobs<FstatVecNull)/nsim
#permutation p-value
pvalPerm

## [1] 0.832

#p-value from F distribution
pval

## [1] 0.7887257
```

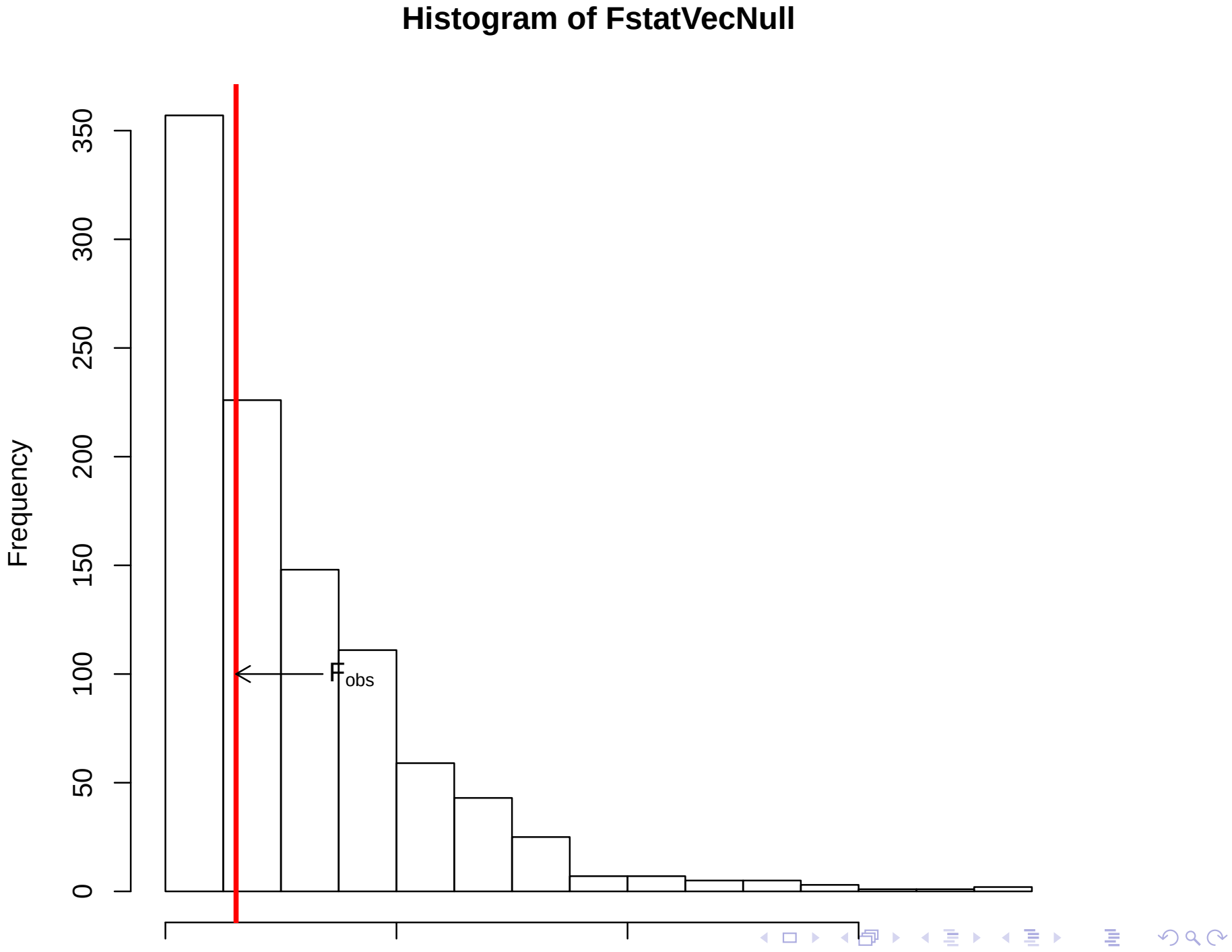
Example 2

```
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(
tapply(dat$score,dat$group,mean)


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


−4
−2
0
2
4

```r
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
```

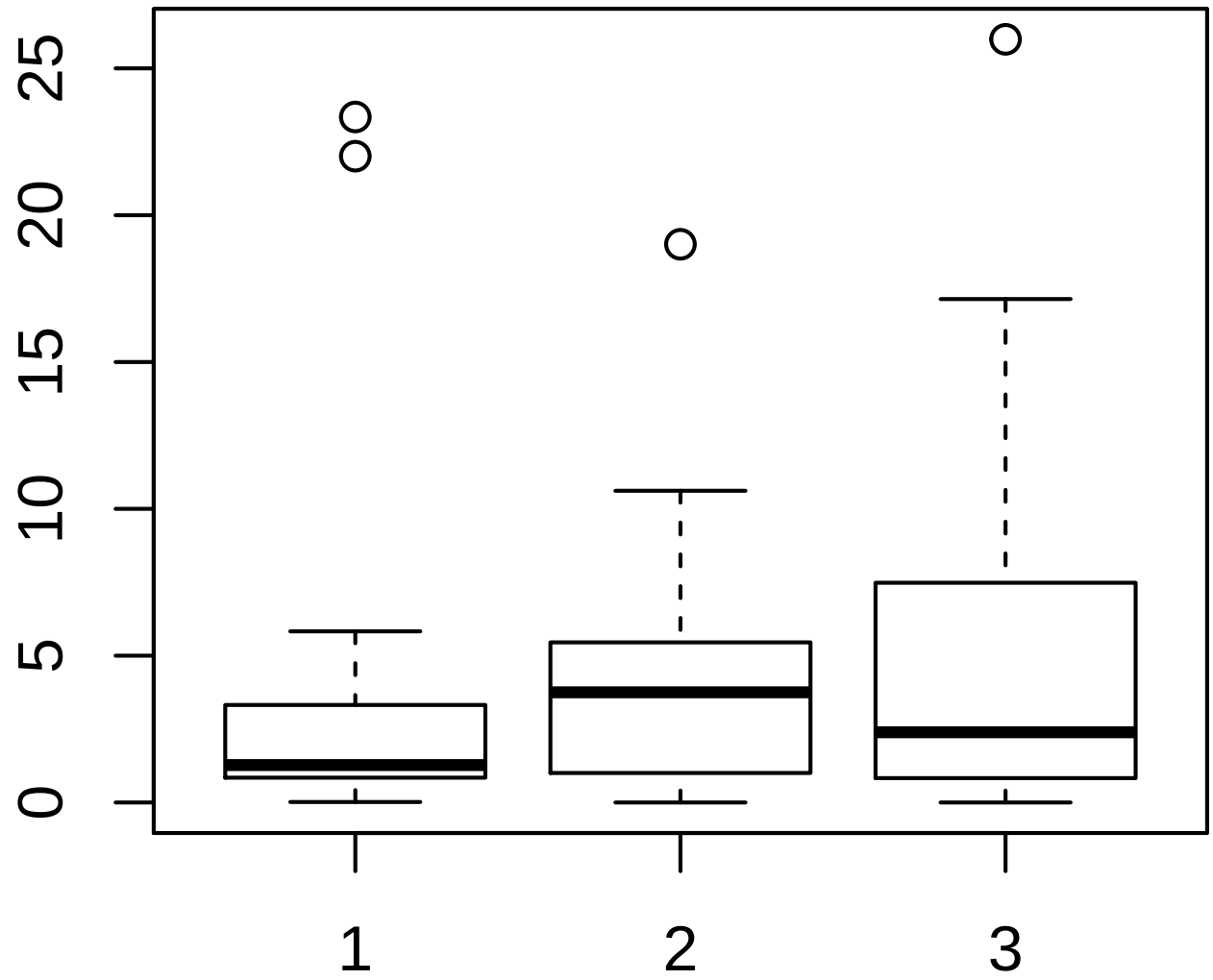
```r
#randomly permute the scores
n <- length(dat$score)
nsim<-1000
FstatVecNull<-rep(NA,nsim)
for (i in 1:nsim){
datPermute <- dat
datPermute$score <- dat$score[sample(1:n,n)]
lmOut <- lm(score~group,data=datPermute)
anovaOut <- anova(lmOut)
FstatVecNull[i]<-anovaOut[1,4]
}
```


Histogram of FstatVecNull
Frequency
0
50
100
150
200
250
300
350
F
obs

```r
pvalPerm<-sum(Fobs<FstatVecNull)/nsim
#permutation p-value
pvalPerm

## [1] 0.575

#p-value F-test
pval

## [1] 0.5459521
```

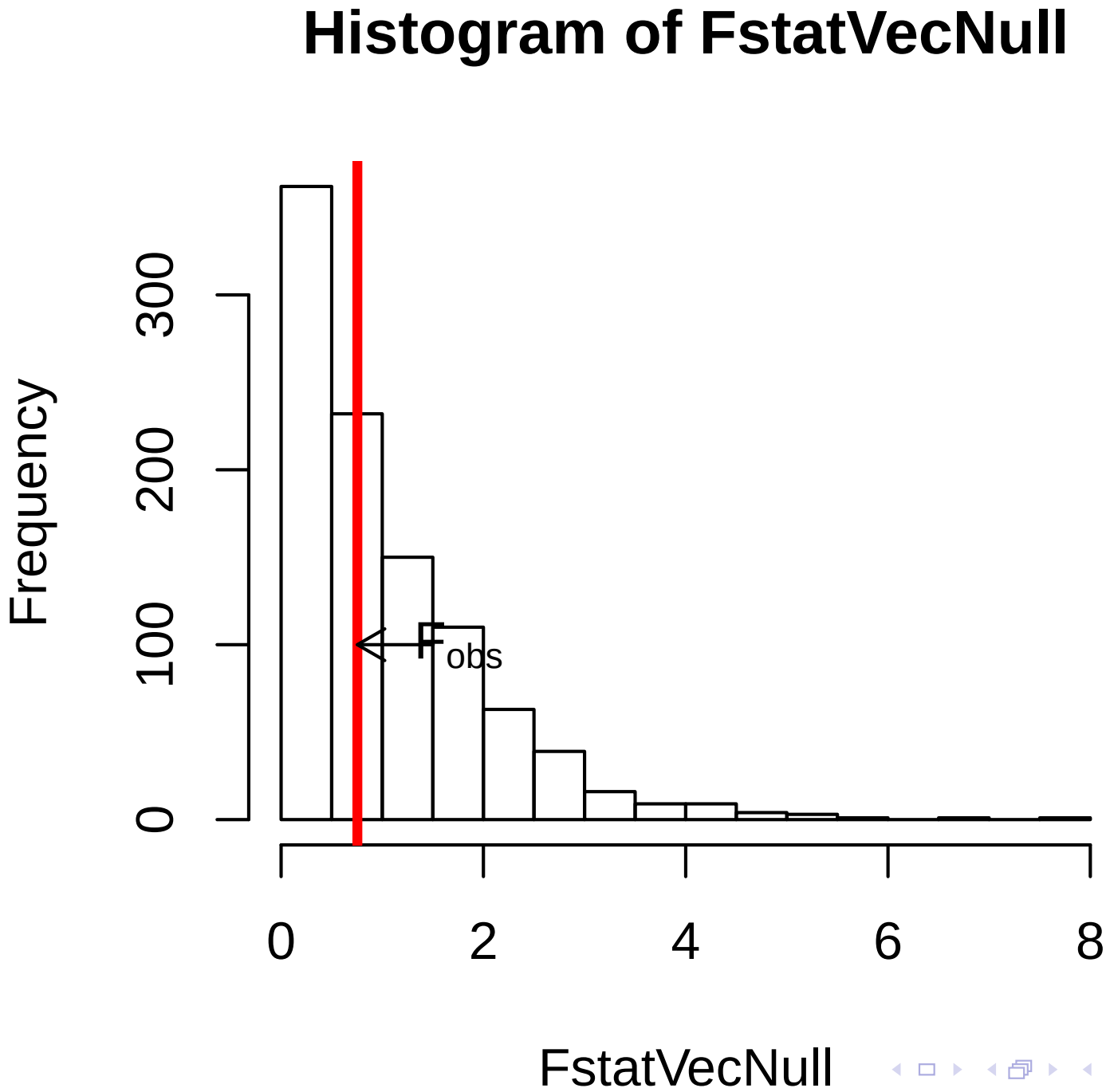
Example 3

```
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
tapply(dat$score,dat$group,mean)


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


0
5
10
15
20
25
1
2
3

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

## [1] 0.4742715
```

```r
#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
Frequency
0
100
200
300
F_{obs}
0
2
4
6
8
FstatVecNull

```r
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 essential a k-sample permutation test based on the ranks rather than the raw data.

The Kruskal-Wallis test statistic is:

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

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.

```r
## 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 di
                      "Subjects with asbestosis"))
dat<-data.frame(rate=rate,group=g)
dat$ranks<-rank(dat$rate)
```

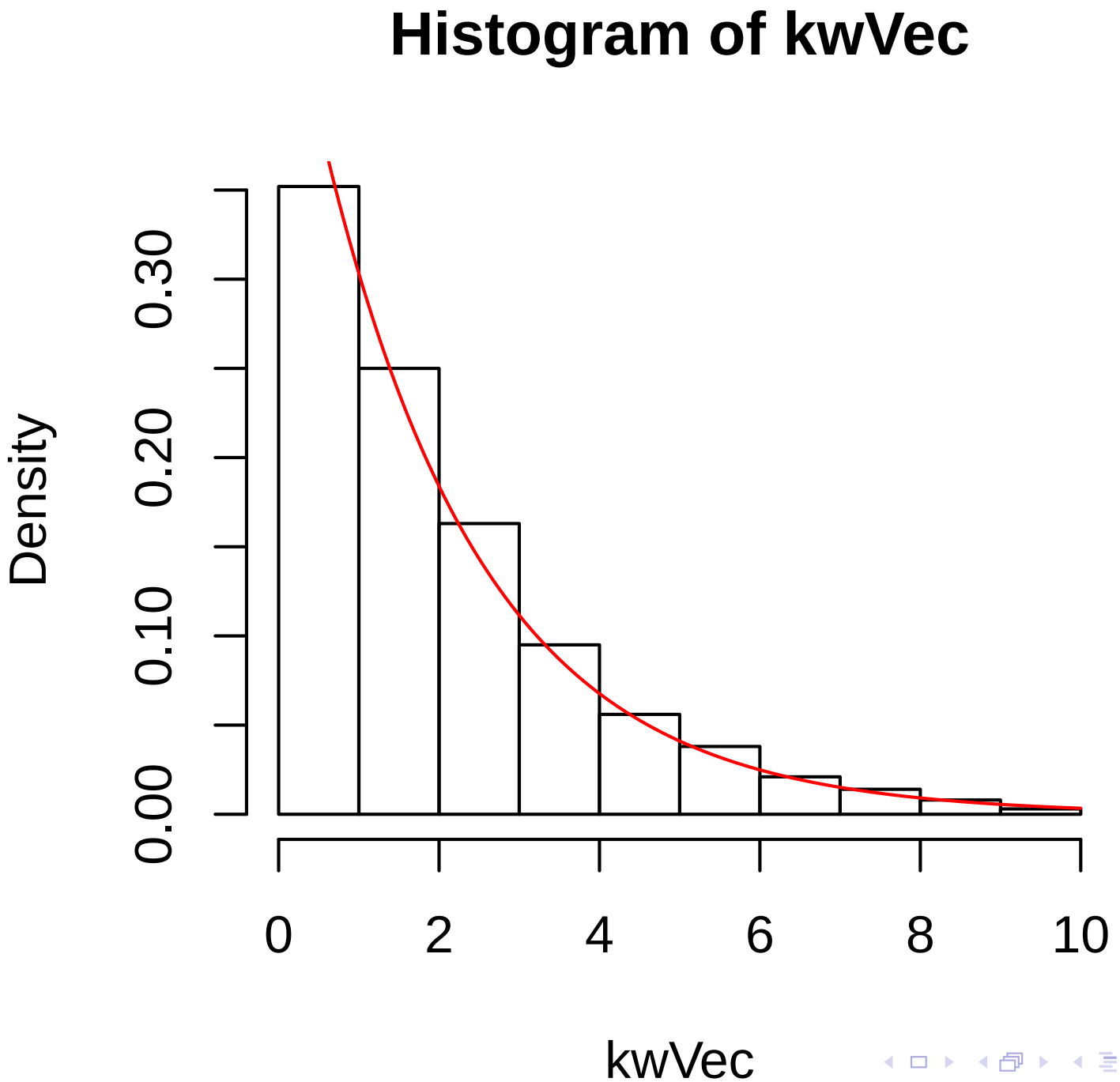
```r
Rbar<-tapply(dat$ranks,dat$group,mean)
n<-tapply(dat$ranks,dat$group,length)
N<-sum(n)
kwStat<-12/(N*(N+1))*sum(n*(Rbar-(N+1)/2)^2)
kwStat
```

```
## [1] 0.7714286
```

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

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

```r
#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
Density
0.00
0.10
0.20
0.30
0
2
4
6
8
10
kwVec

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

#permutation p-value
pvalKW

## [1] 0.698

#chi-square approximation
pval

## [1] 0.6799648
```

```r
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.

```r
#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
```

```r
table(apply(test,1,sum))
```

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

```r
#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 Signifcant 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:
  $|\bar{X}_i - \bar{X}_j| \geq t(\frac{\alpha}{2}, df = N - k)\sqrt{MSE(\frac{1}{n_i} + \frac{1}{n_j})}$
- Non-parametric version:
  $|\bar{R}_i - \bar{R}_j| \geq z(\frac{\alpha}{2}, df = N - k)\sqrt{\frac{N(N+1)}{12}(\frac{1}{n_i} + \frac{1}{n_j})}$
- This procedure is "protected" in the sense that we only perform this test if we rejct 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.

```r
#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
clay$ranks<-rank(clay$pct)
tapply(clay$ranks,clay$group,mean)
```

```
##        1        2        3        4
##  8.50000  8.00000 14.16667 19.33333
```

```r
#We reject the null hypothesis!
kruskal.test(pct~group,data=clay)
```

```
##
##  Kruskal-Wallis rank sum test
##
## data:  pct by group
## Kruskal-Wallis chi-squared = 10.2867, df = 3, p-value = 0.01628
```

```r
#Bonferroni cut-off
k<-4
alpha<-0.05
#cut-off
alpha/(k*(k-1)/2)

## [1] 0.008333333
```

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

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

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

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

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

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

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

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

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

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

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

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

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

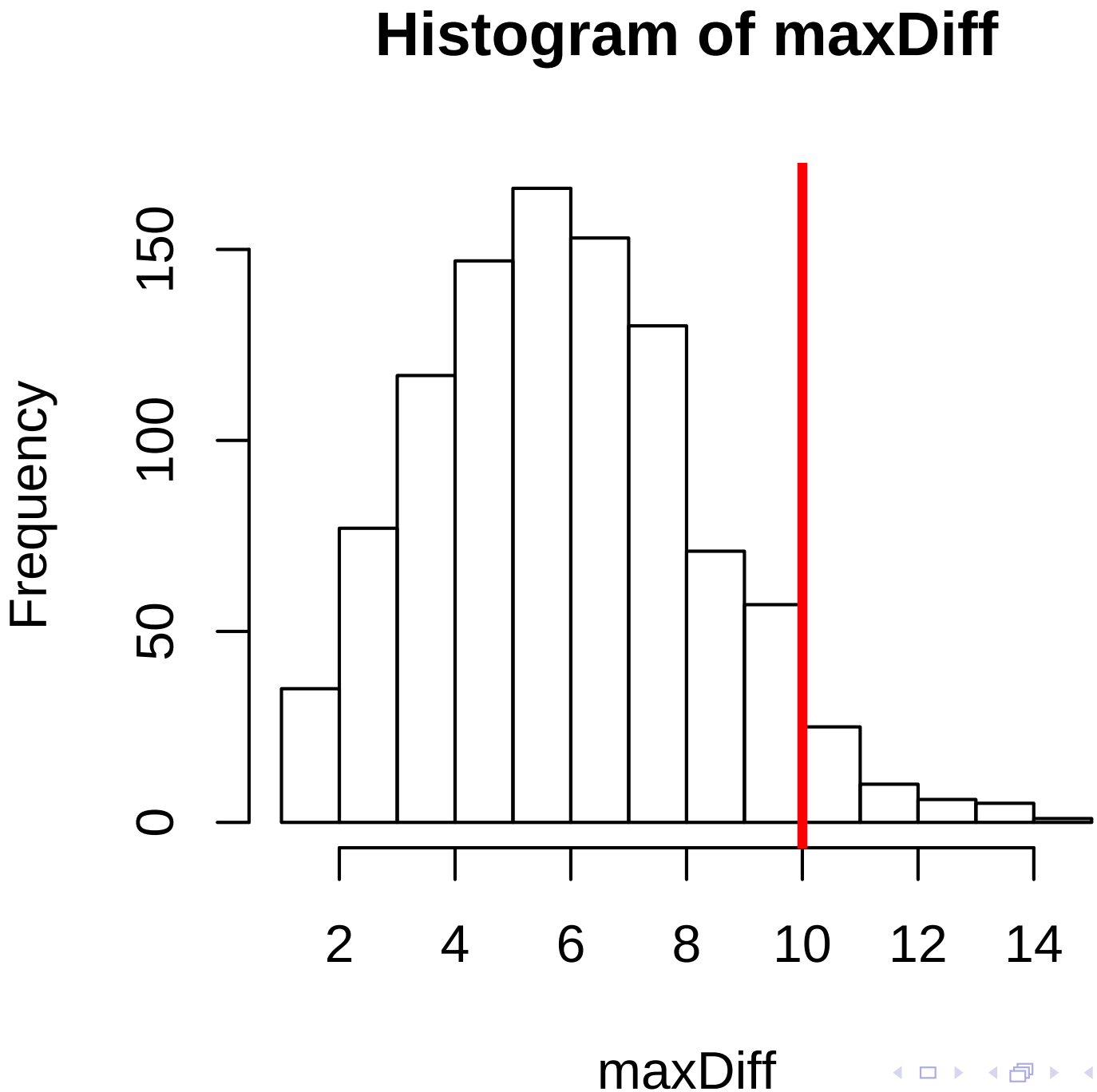
```
##
##  Pairwise comparisons using Wilcoxon rank sum test
##
## data:  clay$pct and clay$group
##
##   1     2     3
## 2 1.000 -     -
## 3 0.792 0.792 -
## 4 0.026 0.156 0.792
##
## P value adjustment method: bonferroni
```

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


Histogram of maxDiff
Frequency
0
50
100
150
2
4
6
8
10
12
14
maxDiff

```r
#10.166667 is the cut-off for significance
(meanRanks<-as.vector(tapply(clay$ranks,clay$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
```