Non-parametric Statistics: Notes 4
Paired Comparisons and Block Designs

Gregory Matthews [1]

[1]Department of Mathematics and Statistics
Loyola University Chicago

Spring 2016

Outline

Paired Comparisons and Blocked Designs

- So far we have seen 1, 2, and K sample **independent** tests.
- We will now focus on paired comparison tests.
- Question: Why are these useful? Reduction in variability.

- Setup: We have two groups, but each element of group A matches with an element of group B.
- We are interested in test $H_0 : \mu_A = \mu_B$ vs $H_0 : \mu_A \neq \mu_B$ (or $H_0 : \mu_A < \mu_B$ or $H_0 : \mu_A > \mu_B$).
- Parametrically what would we do?
- **Two-sample dependent t-test**
- Take the differences of each pair of observations and then perform a 1 sample t-test on the differences ($H_0 : \mu_D = 0$) where $\mu_D = \mu_A - \mu_B$.

- How do we do this non-parametrically?
- You might guess “Permutation test”. And you’d be correct.
- This one is a little bit different. How do we do a permutation test in this setting?

- Under the null hypothesis the the medians of both groups are equal, each observation pair has an equal chance of occurring in either group.
- Therefore, we can randomly permute each pair of observations, take the differences, and compute the test statistic to approximate the null distribution of the test statistics.
- This is the same as permuting the signs of the differences of the $|D_i|$.
- Or more formally....

1. Compute the differences, D_i's, for the n pairs of data, and computer the mean of the differences, $\bar{D}_{obs}$
2. For the n possible pairs, obtain the 2^n possible assignments of plus minus signs to the $|D_i|$'s.
3. For each of the 2^n possibilities in step 2, compute $\bar{D}$, the mean of the differences.
4. The upper-tail p-value is:

$$P_{uppertail} = \frac{\#\bar{D} \geq \bar{D}_{obs}}{2^n}$$

```r
groupA<-c(17.7, 11.3, 11.0,  8.3, 10.2)
groupB<-c(12.3,  4.0, 11.1,  3.8,  5.5)
diffs<-groupB-groupA
diffs
```

```
## [1] -5.4 -7.3  0.1 -4.5 -4.7
```

```r
Dobs<-mean(diffs)
Dobs
```

```
## [1] -4.36
```

```
signs<-expand.grid(c(-1,1),c(-1,1),c(-1,1),c(-1,1),c(-1,1))
signs
```

```
##    Var1 Var2 Var3 Var4 Var5
## 1    -1   -1   -1   -1   -1
## 2     1   -1   -1   -1   -1
## 3    -1    1   -1   -1   -1
## 4     1    1   -1   -1   -1
## 5    -1   -1    1   -1   -1
## 6     1   -1    1   -1   -1
## 7    -1    1    1   -1   -1
## 8     1    1    1   -1   -1
## 9    -1   -1   -1    1   -1
## 10    1   -1   -1    1   -1
## 11   -1    1   -1    1   -1
## 12    1    1   -1    1   -1
## 13   -1   -1    1    1   -1
## 14    1   -1    1    1   -1
## 15   -1    1    1    1   -1
## 16    1    1    1    1   -1
## 17   -1   -1   -1   -1    1
## 18    1   -1   -1   -1    1
## 19   -1    1   -1   -1    1
## 20    1    1   -1   -1    1
## 21   -1   -1    1   -1    1
## 22    1   -1    1   -1    1
## 23   -1    1    1   -1    1
## 24    1    1    1   -1    1
## 25   -1   -1   -1    1    1
## 26    1   -1   -1    1    1
## 27   -1    1   -1    1    1
## 28    1    1   -1    1    1
## 29   -1   -1    1    1    1
## 30    1   -1    1    1    1
## 31   -1    1    1    1    1
## 32    1    1    1    1    1
```

```
signs<-expand.grid(rep(list(c(-1,1)),5))
signs
```

```
##     Var1 Var2 Var3 Var4 Var5
## 1     -1   -1   -1   -1   -1
## 2      1   -1   -1   -1   -1
## 3     -1    1   -1   -1   -1
## 4      1    1   -1   -1   -1
## 5     -1   -1    1   -1   -1
## 6      1   -1    1   -1   -1
## 7     -1    1    1   -1   -1
## 8      1    1    1   -1   -1
## 9     -1   -1   -1    1   -1
## 10     1   -1   -1    1   -1
## 11    -1    1   -1    1   -1
## 12     1    1   -1    1   -1
## 13    -1   -1    1    1   -1
## 14     1   -1    1    1   -1
## 15    -1    1    1    1   -1
```

```r
n<-5
#Compute all of the permutation means
perms<-as.matrix(signs)%*%as.matrix(diffs)/n
sort(perms)
```

```
##  [1] -4.40 -4.36 -2.60 -2.56 -2.52 -2.48 -2.24 -2.20 -1.48 -1.44 -0.72
## [12] -0.68 -0.44 -0.40 -0.36 -0.32  0.32  0.36  0.40  0.44  0.68  0.72
## [23]  1.44  1.48  2.20  2.24  2.48  2.52  2.56  2.60  4.36  4.40
```

```r
Dobs
```

```
## [1] -4.36
```

```r
#One-sided p-value
2/(2^5)
```

```
## [1] 0.0625
```

```r
#Two-sided p-value
#Why times 2??? Permutation
#distribution is symmetric here!
2*(2/(2^5))
```

```
## [1] 0.125
```

- What should you do if the number of permutation is too large?
- Randomly sample permutations.

```r
set.seed(1234)
groupA<-round(rchisq(30,10))
groupB<-round(rchisq(30,10))
#Over a billion permutations
diffs<-groupB-groupA
diffs
```

```
##  [1]  10  -3  -6  -1  -4   5  -4  -1  -2  15   3   9   (
## [18]   3   0  -2   6   1   1  -6  -8  -3  -7   6 -10   3
```

```r
Dobs<-mean(diffs)
Dobs
```

```
## [1] -0.06666667
```

```r
nsim<-1000
permVecNull<-rep(NA,nsim)
for (i in 1:nsim){
n<-length(diffs)
permVecNull[i]<-mean(sample(c(-1,1),n,replace=TRUE)*abs(diffs))
}
```

```r
#upper p-value
(upper<-sum(permVecNull>=abs(Dobs))/nsim)

## [1] 0.516

#lower p-value
(lower<-sum(permVecNull<=-abs(Dobs))/nsim)

## [1] 0.468

#two-sided
upper+lower

## [1] 0.984

#I think we should not reject the null
```

Wilcoxon Sign-Rank test

- We can do this exact same test with the ranks rather than the raw values.
- This test is called the Wilcoxon rank-sum test.

Wilcoxon Sign-Rank test

- Description of test

Wilcoxon Sign-Rank test

```
groupA<-c(17.7, 11.3, 11.0,  8.3, 10.2)
groupB<-c(12.3,  4.0, 11.1,  3.8,  5.5)
diffs<-groupB-groupA
diffs

## [1] -5.4 -7.3  0.1 -4.5 -4.7

sign(diffs)*rank(abs(diffs))

## [1] -4 -5  1 -2 -3

Dobs<-sum(sign(diffs)*rank(abs(diffs)))
Dobs

## [1] -13
```

Wilcoxon Sign-Rank test

```
#Lets calculate the exact p-value
signs<-expand.grid(rep(list(c(-1,1)),5))
t(t(signs)*c(1:5))
```

```
##       Var1 Var2 Var3 Var4 Var5
##  [1,]   -1   -2   -3   -4   -5
##  [2,]    1   -2   -3   -4   -5
##  [3,]   -1    2   -3   -4   -5
##  [4,]    1    2   -3   -4   -5
##  [5,]   -1   -2    3   -4   -5
##  [6,]    1   -2    3   -4   -5
##  [7,]   -1    2    3   -4   -5
##  [8,]    1    2    3   -4   -5
##  [9,]   -1   -2   -3    4   -5
## [10,]    1   -2   -3    4   -5
## [11,]   -1    2   -3    4   -5
## [12,]    1    2   -3    4   -5
## [13,]   -1   -2    3    4   -5
```

Wilcoxon Sign-Rank test

```
(rankSumPerms<-apply(t(t(signs)*c(1:5)),1,sum))

##  [1] -15 -13 -11  -9  -9  -7  -5  -3  -7  -5  -3  -1  -1
## [18]  -3  -1   1   1   3   5   7   3   5   7   9   9  11
```

Wilcoxon Sign-Rank test

```
# Calculate the two-sided p-value
(sum(rankSumPerms <= Dobs) + sum(rankSumPerms >= abs(Dobs)))/length(rankSumPerms)
```

```
## [1] 0.125
```

Wilcoxon Sign-Rank test

```
wilcox.test(groupA, groupB, paired = T, exact = TRUE)
```

```
##
##  Wilcoxon signed rank test
##
## data:  groupA and groupB
## V = 14, p-value = 0.125
## alternative hypothesis: true location shift is not equal to 0
```

Wilcoxon Sign-Rank test

```
wilcox.test(groupA, groupB, paired = T, exact = FALSE)
```

```
##
##  Wilcoxon signed rank test with continuity correction
##
## data:  groupA and groupB
## V = 14, p-value = 0.1056
## alternative hypothesis: true location shift is not equal to 0
```

```
# The option exact=FALSE uses a normal approximation Normal approximation
# should only be used when n is larger than approximately 10
```

Blocking

- Blocking is a technique that is used when the experimental units to which the treatments are to be applied at not homogenous, or when the conditions under which the experiment is to be conducted cannot be held constant throughout.
- I think of this as a generalizated of the paired two sample tests. But we can have k treatment groups instead of just 2 and instead of dependent pairs we can have k related elements within b blocks.

	Blocks				
Treaments	1	2	...	b	Means
1	X_{11}	X_{12}	...	X_{1b}	$\bar{X}_{1.}$
2	X_{21}	X_{22}	...	X_{2b}	$\bar{X}_{2.}$
...	...	...	...	...	...
k	X_{k1}	X_{k2}	...	X_{kb}	$\bar{X}_{k.}$
Means	$\bar{X}_{.1}$	$\bar{X}_{.2}$	...	$\bar{X}_{.b}$	$\bar{X}$

- How do we do this parametrically?
- Assume the following model: $X_{ij} = \mu + \tau_i + block_j + \epsilon_{ij}$
- where μ is the overall effect, τ_i are the treatment effects, $block_j$ are the block effects, and ϵ_{ij} are iid random variables with median 0.
- We want to test $H_0 : \tau_1 = \tau_2 = \cdots = \tau_k$ vs H_a: Not all τ_i are the same.
- When $\epsilon_{ij} \sim N(0, \sigma^2)$ then we use the following F statistic:

$$F = \frac{MS_{trt}}{MSE} \overset{H_0}{\sim} F_{k-1,(k-1)(b-1)}$$

- Non-parametrically: We test the same hypothesis, we just relax the assumption of normality.
- Assume the following model: $X_{ij} = \mu + \tau_i + block_j + \epsilon_{ij}$
- Here we have the same assumptions and the non-parametric ANOVA set-up EXCEPT that observations are NOT assumed independent within a block!
- This changes how we have to do our permutations to approximate the null hypothesis.

Permutation F-test for Randomized Complete Block Design

1. Compute the F-Statistic, F_{obs}, for the original data.
2. Permute the observations *within* each of the blocks, doing so for all of the blocks. (Do NOT permute across blocks. (WHY?)) There are $(k!)^b$ possibilities.
3. For each of the possiblities in step 2, compute the F statistic for the randomized complete block design.
4. Since the values of F indicate a difference among treatments, obtain the upper-tail p-values as

$$P_{uppertail} = \frac{number\ of\ F's\ \geq F_{obs}}{(k!)^b}$$

```
dat[1:10,]

##    yield trt block
## 1    120   1     1
## 2    208   1     2
## 3    199   1     3
## 4    194   1     4
## 5    177   1     5
## 6    195   1     6
## 7    207   2     1
## 8    188   2     2
## 9    181   2     3
## 10   164   2     4

dat<-dat[order(dat$block),]
```

```r
#Usual F-test
lmOut<-lm(yield~trt+block,data=dat)
anova(lmOut)

## Analysis of Variance Table
##
## Response: yield
##           Df Sum Sq Mean Sq F value  Pr(>F)
## trt        3 5408.3 1802.78  3.1211 0.05752 .
## block      5 2816.8  563.37  0.9753 0.46401
## Residuals 15 8664.2  577.61
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1

str(anova(lmOut))

## Classes 'anova' and 'data.frame': 3 obs. of  5 variables
##  $ Df     : int  3 5 15
##  $ Sum Sq : num  5408 2817 8664
##  $ Mean Sq: num  1803 563 578
```

```r
str(anova(lmOut))

## Classes 'anova' and 'data.frame': 3 obs. of  5 variables
##  $ Df     : int  3 5 15
##  $ Sum Sq : num  5408 2817 8664
##  $ Mean Sq: num  1803 563 578
##  $ F value: num  3.121 0.975 NA
##  $ Pr(>F) : num  0.0575 0.464 NA
##  - attr(*, "heading")= chr  "Analysis of Variance Table\

#We can pull out the piece that we want
Fobs<-anova(lmOut)$'F value'[1]
```

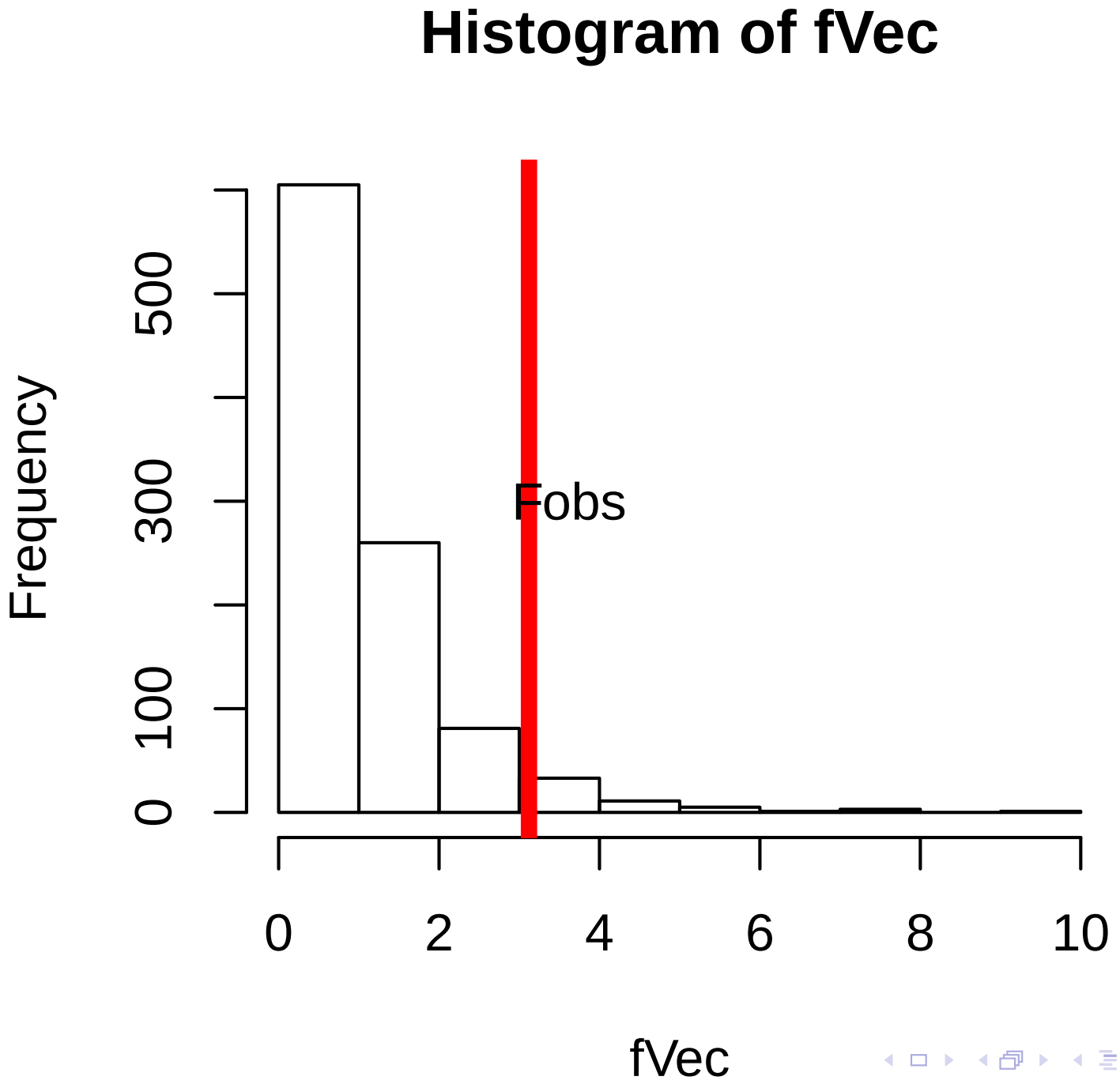
```r
datPermute<-dat
tapply(datPermute$yield,datPermute$block,function(x){sample(x,length(x),replace=FALSE)})
```

```
## $`1`
## [1] 207 120 128 122
##
## $`2`
## [1] 128 208 188 137
##
## $`3`
## [1] 181 199 177 160
##
## $`4`
## [1] 194 164 142 177
##
## $`5`
## [1] 160 155 157 177
##
## $`6`
## [1] 195 179 138 175
```

```r
unlist(tapply(datPermute$yield,datPermute$block,function(x){sample(x,length(x),replace=FALSE)}))
```

```
##  11  12  13  14  21  22  23  24  31  32  33  34  41  42  43  44  51  52
## 122 128 120 207 208 128 137 188 160 199 177 181 142 164 194 177 177 160
##  53  54  61  62  63  64
## 157 155 138 195 179 175
```

```r
set.seed(1234)
nsim<-1000
fVec<-rep(NA,nsim)
for (i in 1:nsim){
datPermute<-dat
#Shuffle WITHIN blocks
datPermute$yield<-unlist(tapply(datPermute$yield,datPermute$block,
function(x){sample(x,length(x),replace=FALSE)}))
#fit the linear model
lmOut<-lm(yield~trt+block,data=datPermute)
#Compute the F statistic for the treatments.
fVec[i]<-anova(lmOut)$'F value'[1]
}
```


Histogram of fVec
Frequency
0
100
300
500
Fobs
0
2
4
6
8
10
fVec

```r
#Compute the p-value
sum(fVec >= Fobs)/nsim

## [1] 0.05
```

- So what happens is we fail to reject the null hypothesis? (Nothing. We stop.)
- If we DO reject the null hypothesis, the next logical question is: "which pairs of groups are different?"
- However, we need to account for the problem of multiple comparisons.

```
pairwise.wilcox.test(dat$yield,dat$trt,p.adjust="bonf",pai
```

```
## Warning in wilcox.test.default(xi, xj, paired =
paired, ...):  cannot compute exact p-value with ties
## Warning in wilcox.test.default(xi, xj, paired =
paired, ...):  cannot compute exact p-value with ties
```

```
##
##  Pairwise comparisons using Wilcoxon signed rank test
##
## data:  dat$yield and dat$trt
##
##   1    2    3
## 2 1.00 -    -
## 3 0.35 1.00 -
## 4 0.38 0.94 1.00
##
## P value adjustment method: bonferroni
```

```r
nsim<-1000
maxDiffVec<-rep(NA,nsim)
for (i in 1:nsim){
datPermute<-dat
#Shuffle WITHIN blocks
datPermute$yield<-unlist(tapply(datPermute$yield,datPermute
$block,
function(x){sample(x,length(x),replace=FALSE)}))
#fit the linear model
diffs<-tapply(datPermute$yield,datPermute$trt,mean)
#Compute the max absolute difference
maxDiffVec[i]<-abs(max(diffs)-min(diffs))
}
```

```r
#nintieth percentile of the max diff
quantile(maxDiffVec,0.9)

##  90%
## 36.5

diffsObs<-tapply(dat$yield,dat$trt,mean)
#Matrix of observed differences.
abs(outer(diffsObs,diffsObs,"-"))

##           1         2         3         4
## 1  0.000000  3.833333 30.333333 33.166667
## 2  3.833333  0.000000 26.500000 29.333333
## 3 30.333333 26.500000  0.000000  2.833333
## 4 33.166667 29.333333  2.833333  0.000000

#Nothing is pairwise significant?  Is this ok?
```

- So we just did a permutation test for a randomized complete block design?
- Can you guess what's next?
- If you guessed "Repeat the same thing with ranks" then you are correct!
- This test is called the Friedman test.

```r
#We could go through how to do this the long way, but....
#Friedman test in R
friedman.test(dat$yield,dat$trt,dat$block)

##
##  Friedman rank sum test
##
## data:  dat$yield, dat$trt and dat$block
## Friedman chi-squared = 5.6, df = 3, p-value = 0.1328
```

Parametric Test	Non-Parametric	With ranks
1 sample t-test	sign test	
2 sample Independent t-test	permutation	Wilcoxon Rank- Sum
2 sample Dependent t-test	permutation	Wilcoxon Sign-Rank
ANOVA	permutation	Kruskal-Wallis
RCBD	permutation	Friedman