IS5 in R: Paired Samples and Blocks (Chapter 18)

Nicholas Horton (nhorton@amherst.edu)

December 19, 2020

Introduction and background

This document is intended to help describe how to undertake analyses introduced as examples in the Fifth Edition of *Intro Stats* (2018) by De Veaux, Velleman, and Bock. This file as well as the associated R Markdown reproducible analysis source file used to create it can be found at http://nhorton.people.amherst.edu/is5.

This work leverages initiatives undertaken by Project MOSAIC (http://www.mosaic-web.org), an NSF-funded effort to improve the teaching of statistics, calculus, science and computing in the undergraduate curriculum. In particular, we utilize the `mosaic` package, which was written to simplify the use of R for introductory statistics courses. A short summary of the R needed to teach introductory statistics can be found in the mosaic package vignettes (https://cran.r-project.org/web/packages/mosaic). A paper describing the mosaic approach was published in the *R Journal*: https://journal.r-project.org/archive/2017/RJ-2017-024.

Chapter 18: Paired Samples and Blocks

```
library(mosaic)
library(readr)
library(janitor)
library(tidyr) # for the gather() function
Dexterity <- read_csv("http://nhorton.people.amherst.edu/is5/data/Dexterity.csv") %>%
  janitor::clean_names()
```

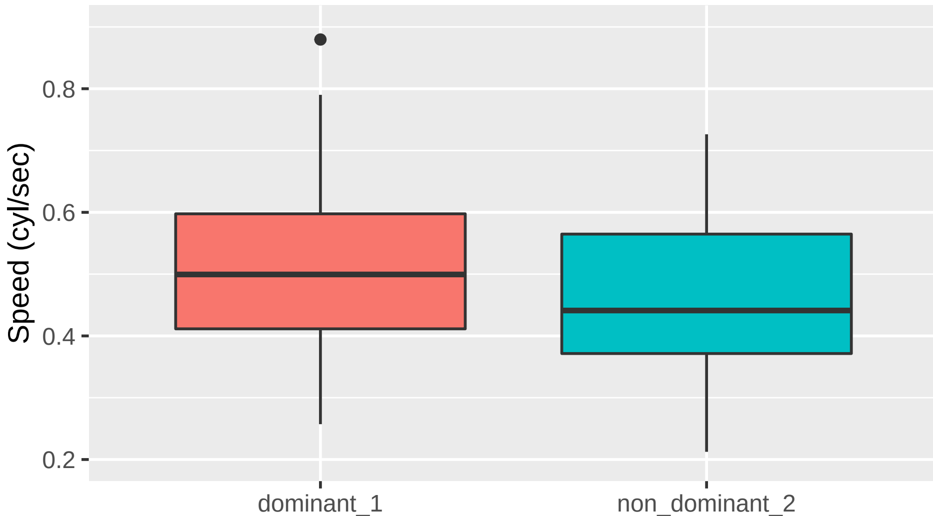
By default, `read_csv()` prints the variable names. These messages have been suppressed using the `message=FALSE` code chunk option to save space and improve readability. Here we use the `clean_names()` function from the `janitor` package to sanitize the names of the columns (which would otherwise contain special characters or whitespace).

```
Dexterity %>%
  select(age_months, dominant_1, non_dominant_2, gender) %>%
  head(n = 7)
```

```
## # A tibble: 7 x 4
##   age_months dominant_1 non_dominant_2 gender
##        <dbl>      <dbl>          <dbl> <chr>
## 1        117      0.353          0.216 male
## 2        101      0.257          0.343 male
## 3        135      0.537          0.497 male
## 4        119      0.444          0.496 male
## 5        124      0.483          0.388 female
## 6        127      0.524          0.422 female
## 7        101      0.455          0.381 male
```

Section 18.1: Paired Data

```
# Figure 18.1
Dexterity %>%
  select(dominant_1, non_dominant_2) %>%
  gather(key = hand_type, value = speed, dominant_1, non_dominant_2) %>%
  gf_boxplot(speed ~ hand_type, fill = ~hand_type) %>%
  gf_labs(x = "", y = "Speed (cyl/sec)") +
  ylim(.2, .9) +
  guides(fill = FALSE)
```



Example 18.1: Identifying Paired Data We begin by creating the data set on page 586.

```
WorkWeek <- rbind(
  data.frame(name = "Jeff", fiveday = 2798, fourday = 2914),
  data.frame(name = "Betty", fiveday = 7724, fourday = 6112),
  data.frame(name = "Roger", fiveday = 7505, fourday = 6177),
  data.frame(name = "Tom", fiveday = 838, fourday = 1102),
  data.frame(name = "Aimee", fiveday = 4592, fourday = 3281),
  data.frame(name = "Greg", fiveday = 8107, fourday = 4997),
  data.frame(name = "Larry G.", fiveday = 1228, fourday = 1695),
  data.frame(name = "Tad", fiveday = 8718, fourday = 6606),
  data.frame(name = "Larry M.", fiveday = 1097, fourday = 1063),
  data.frame(name = "Leslie", fiveday = 8089, fourday = 6392),
  data.frame(name = "Lee", fiveday = 3807, fourday = 3362)
)
WorkWeek
```

```
##          name fiveday fourday
## 1        Jeff    2798    2914
## 2       Betty    7724    6112
## 3       Roger    7505    6177
## 4         Tom     838    1102
## 5       Aimee    4592    3281
## 6        Greg    8107    4997
## 7    Larry G.    1228    1695
## 8         Tad    8718    6606
## 9    Larry M.    1097    1063
```

```
## 10   Leslie    8089    6392
## 11      Lee    3807    3362
```

Looking at pairwise differences in `Dexterity`.

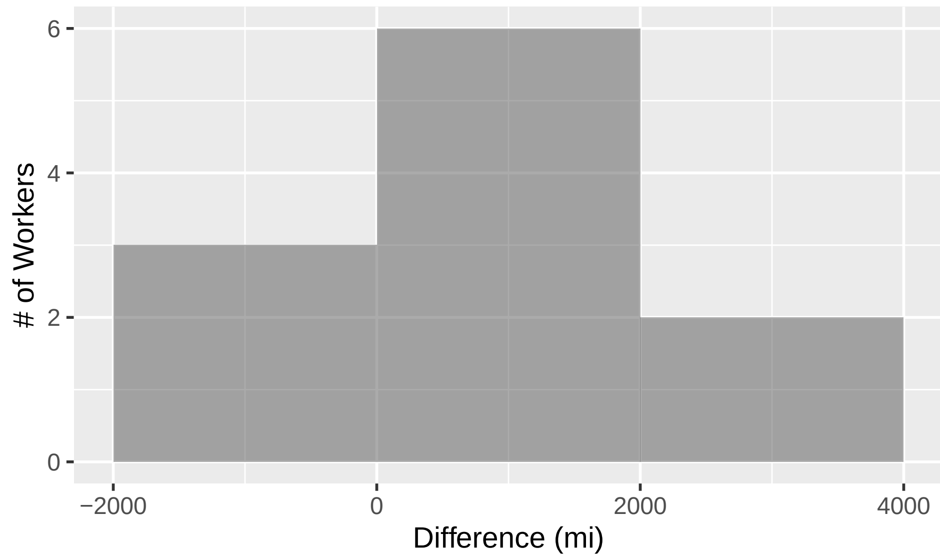
```
Dexterity %>%
  select(dominant_1, non_dominant_2) %>%
  mutate(difference = dominant_1 - non_dominant_2) %>%
  head(n = 18)
```

```
## # A tibble: 18 x 3
##    dominant_1 non_dominant_2 difference
##         <dbl>          <dbl>      <dbl>
##  1      0.353          0.216    0.137
##  2      0.257          0.343   -0.0863
##  3      0.537          0.497    0.0392
##  4      0.444          0.496   -0.0524
##  5      0.483          0.388    0.0947
##  6      0.524          0.422    0.102
##  7      0.455          0.381    0.0742
##  8      0.394          0.403   -0.00904
##  9      0.451          0.328    0.124
## 10      0.527          0.271    0.256
## 11      0.565          0.415    0.149
## 12      0.653          0.298    0.355
## 13      0.421          0.337    0.0833
## 14      0.320          0.233    0.0872
## 15      0.344          0.241    0.102
## 16      0.428          0.612   -0.184
## 17      0.556          0.521    0.0357
## 18      0.465          0.411    0.0543
```

Section 18.2: The Paired *t*-Test

Example 18.2: Checking Assumptions and Conditions We can display the distribution of the differences (see page 588).

```
WorkWeek <- WorkWeek %>%
  mutate(difference = fiveday - fourday)
gf_histogram(~difference, data = WorkWeek, binwidth = 2000, center = 1000) %>%
  gf_labs(x = "Difference (mi)", y = "# of Workers")
```



Example 18.3: Doing a Paired *t*-Test It is straightforward to carry out the paired t-test.

```
t.test(~difference, data = WorkWeek)
```

```
##
##  One Sample t-test
##
## data:  difference
## t = 2.858, df = 10, p-value = 0.01701
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##   216.4276 1747.5724
## sample estimates:
## mean of x
##       982
```

or

```
nwork <- nrow(WorkWeek)
nwork # number of pairs
```

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

```
dwork <- mean(~difference, data = WorkWeek)
dwork # mean of differences
```

```
## [1] 982
```

```
swork <- sd(~difference, data = WorkWeek)
swork # SD of differences
```

```
## [1] 1139.568
```

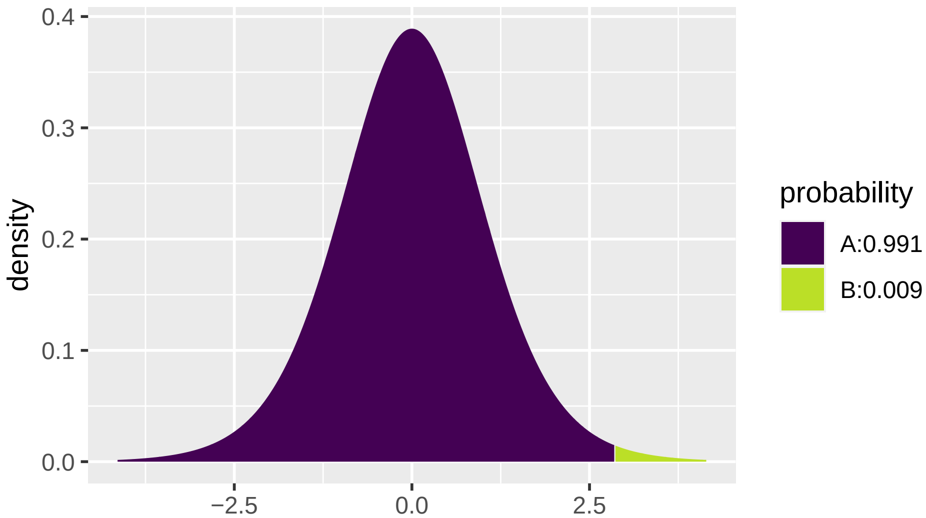
```
sework <- swork / (nwork^.5)
sework # SE of differences
```

```
## [1] 343.5928
```

```
twork <- (dwork - 0) / sework
twork # t stat
```

```
## [1] 2.858034
```

```
2 * xpt(twork, df = nwork - 1, lower.tail = FALSE)
```




```
## [1] 0.01701413
```

The `xpt()` function finds the p-value and plots it on a graph to visualize it. Here, the visualization shows a one-sided test, but in the book, it is two sided.

Section 18.3: Confidence Intervals for Matched Pairs

We begin by reading the data.

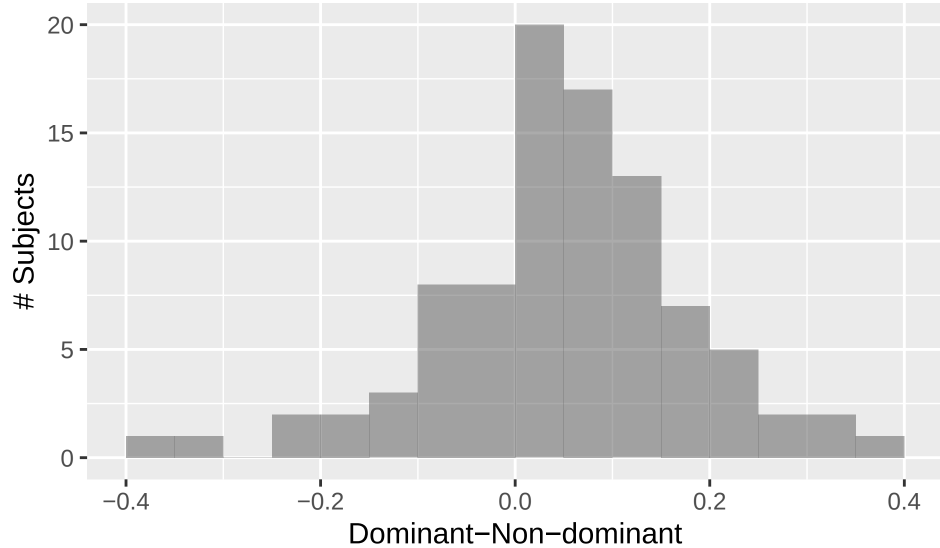
```
Couples <- read_csv("http://nhorton.people.amherst.edu/is5/data/Couples.csv") %>%
  filter(wAge != "*") %>%
  mutate(wAge = as.numeric(wAge))
# table on page 592
Couples %>%
  select(wAge, hAge) %>%
  mutate(difference = hAge - wAge) %>%
  head(n = 7)
```

```
## # A tibble: 7 x 3
##    wAge  hAge difference
##   <dbl> <dbl>      <dbl>
## 1    43    49          6
## 2    28    25         -3
## 3    30    40         10
## 4    57    52         -5
## 5    52    58          6
## 6    27    32          5
## 7    52    43         -9
```

Step-By-Step Example: A Paired *t*-Interval We replicate the example from page 593.

```
DexData <- Dexterity %>%
  select(dominant_1, non_dominant_2) %>%
  mutate(difference = dominant_1 - non_dominant_2) %>%
  filter(dominant_1 < 1)
# For some reason, the book has removed one observation where dominant_1 = 1,
```

```r
# but has kept the count of children at 93 instead of 92
gf_histogram(~difference, data = DexData, binwidth = .05, center = .025) %>%
  gf_labs(x = "Dominant-Non-dominant", y = "# Subjects")
```



Here we display the calculations using the `t.test()` function and then by hand.

```r
df_stats(~difference, data = DexData)
```

```
##     response        min           Q1    median        Q3       max       mean
## 1 difference -0.3859649 -0.001236833 0.0525483 0.1240941 0.3550096 0.05148209
##          sd  n missing
## 1 0.1298746 92       0
```

```r
t.test(~difference, data = DexData)
```

```
##
##  One Sample t-test
##
## data:  difference
## t = 3.8021, df = 91, p-value = 0.0002592
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  0.02458583 0.07837834
## sample estimates:
##  mean of x
## 0.05148209
```

```r
ndex <- nrow(DexData) + 1 # the book kept n at 93 for some reason
ndex # number of pairs (children)
```

```
## [1] 93
```

```r
ddex <- mean(~difference, data = DexData)
ddex # mean difference
```

```
## [1] 0.05148209
```

```r
sdex <- sd(~difference, data = DexData)
sdex # standard deviation of the differences
```

```
## [1] 0.1298746
```

```r
sedex <- sdex / (ndex^.5)
sedex # standard error of the differences
```

```
## [1] 0.01346736
```

```r
df <- ndex - 1
df # this matches the book, but it should be 91
```

```
## [1] 92
```

```r
tstats <- qt(p = c(.025, .975), df = df)
tstats
```

```
## [1] -1.986086  1.986086
```

```r
medex <- tstats * sedex
medex # margin of error of the differences
```

```
## [1] -0.02674735  0.02674735
```

```r
ddex + medex
```

```
## [1] 0.02473474 0.07822943
```

```r
# Or, if you don't want to go through all those calculations:
t.test(~difference, data = DexData, df = df)
```

```
##
##  One Sample t-test
##
## data:  difference
## t = 3.8021, df = 91, p-value = 0.0002592
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  0.02458583 0.07837834
## sample estimates:
##  mean of x
## 0.05148209
```

Effect Size

Example 18.4: Looking at Effect Size with a Paired *t* Confidence Interval We can verify the calculations from the example.

```r
tstats <- qt(p = c(.025, .975), df = nwork - 1)
tstats
```

```
## [1] -2.228139  2.228139
```

```r
me <- tstats * sework
me # margin of error
```

```
## [1] -765.5724  765.5724
```

```r
dwork + me # confidence interval
```

```
## [1]  216.4276 1747.5724
```

Section 18.4: Blocking

What's Independent?

Random Matters: A Bootstrapped Paired Data Confidence Interval and Hypothesis Test Our usual approach to bootstrapping works here.

```
set.seed(2345)
numsim <- 5000

# What does do() do?
mean(~difference, data = resample(DexData)) # One mean of a random resample
```

```
## [1] 0.04017783
```

```
mean(~difference, data = resample(DexData)) # Another mean of a random resample
```

```
## [1] 0.06400291
```

```
do(2) * mean(~difference, data = resample(DexData)) # Calculates two means
```

```
##         mean
## 1 0.06821305
## 2 0.03010977
```

```
# We need numsim means
DexBoots <- do(numsim) * mean(~difference, data = resample(DexData))
```

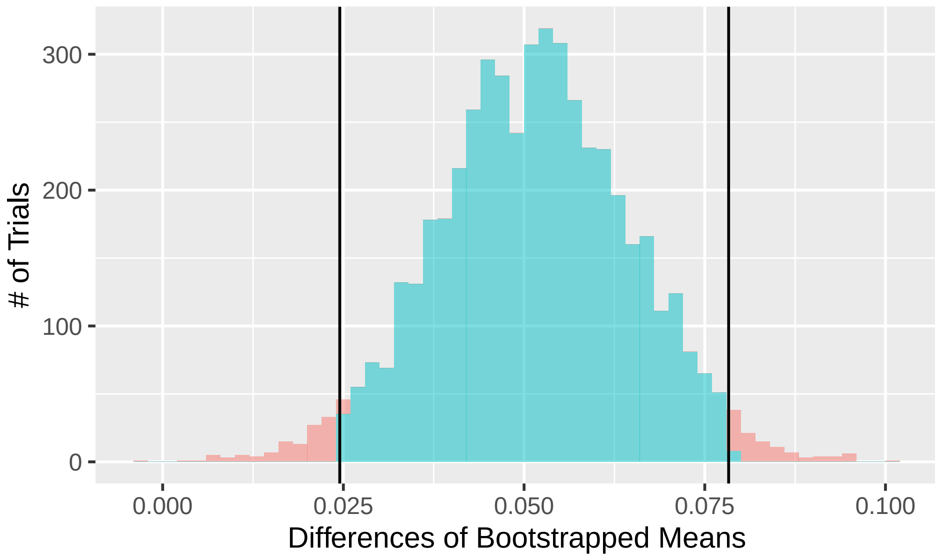
For more information about `resample()`, refer to the resample vignette in mosaic.

```
qdata(~mean, p = c(.025, .975), data = DexBoots)
```

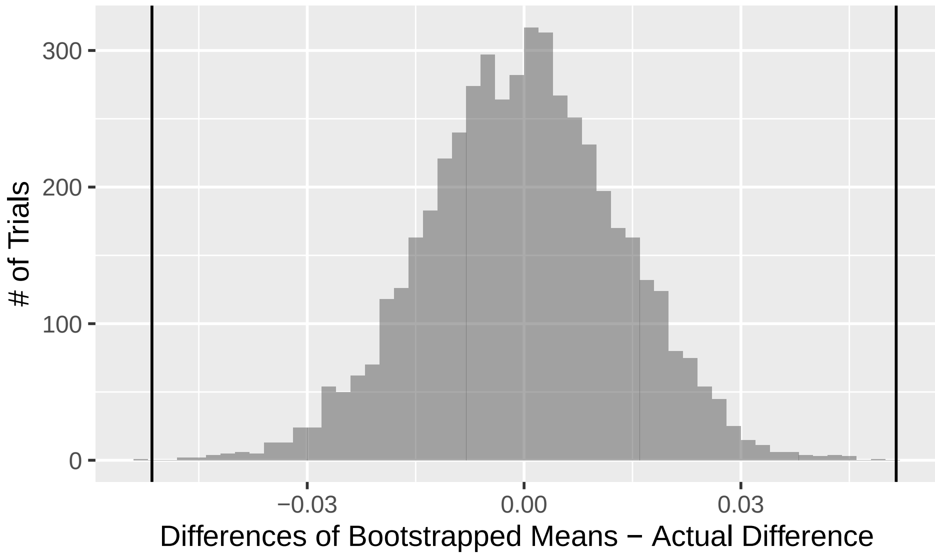
```
##       2.5%      97.5%
## 0.02445067 0.07745861
```

```
DexBoots <- DexBoots %>%
  mutate(interval = ifelse(mean > .0245 & mean < .0783, "Within 95% Confidence",
    "Outside 95% Confidence"
  ))

# Figure 18.4, page 597
gf_histogram(~mean, fill = ~interval, data = DexBoots, binwidth = .002, center = .001) %>%
  gf_vline(xintercept = .0245) %>%
  gf_vline(xintercept = .0783) %>%
  gf_labs(x = "Differences of Bootstrapped Means", y = "# of Trials") +
  guides(fill = FALSE)
```




```
# Figure 18.5
gf_histogram(~ (mean - ddex), data = DexBoots, binwidth = .002, center = .001) %>%
  gf_vline(xintercept = ddex) %>%
  gf_vline(xintercept = -ddex) %>%
  gf_labs(x = "Differences of Bootstrapped Means - Actual Difference", y = "# of Trials")
```




```
df_stats(~ (mean - ddex), data = DexBoots)
```

```
##          response         min         Q1        median          Q3        max
## 1 I(mean - ddex) -0.05376577 -0.009114 -1.635027e-05 0.008940328 0.04903609
##            mean         sd    n missing
## 1 -0.0001587614 0.01349299 5000       0
```

With `df_stats()`, we can see that our minimum is within the interval, but our maximum isn't.

```
DexBoots %>%
  filter((mean - ddex) > ddex)
```

```
## [1] mean     interval
## <0 rows> (or 0-length row.names)
```

Like the book, there is one instance (out of 5,000), so we estimate the P-value as 1/5,000 (the book says 50,000, which is incorrect), or .0002.