Introduction to the to t-test: data exploration

Nathan Brouwer | brouwern@gmail.com | @lobrowr

September 29, 2016

Introduction

In this lab exericse we will use data from Chapter 12 of Whitlock & Shulter's *Analysis of Biological Data*. The data are from an observational study (Levin et al 2002) where the survival of native Chinook Salmon was monitered in 6 streams in the Columbia River Basin in Idaho where the the eastern species brook trout had been introduced for sport fishing. Brook trout where hypothesized to reduce survival rates of the native Chinook salmon. Sixe Site where brook trout where absent were used as controls. In each stream, juvenile Chinook where captured, PIT tagged, and released. PIT tags are similar to the microchips put in pets to ID them. As the fish migrated downstream, their PIT tags were detected by receivers. The number of PIT tags detected divided by the total number released indicates the survival rate. The original study used several years of data; Whitlock and Shulter present just one, which we will examine in detail.

References

Levin et al. 2002. Non–indigenous brook trout and the demise of Pacific salmon: a forgotten threat? Proceedings of the Royal Society B. http://rspb.royalsocietypublishing.org/content/269/1501/1663.short

Preliminaries

Load packages

```
# For plots
library(ggplot2)

# Great ggplot helper functions
library(ggpubr)

#data manipulation
library(dplyr)
```

Set the number of digits to display

R's default is to display numbers to a high level of precisions: 7 digits. We can see this using this command

```
options()$digits
```

```
## [1] 7
```

We can set R to automatically round things off like this

```
options(digits = 4)
```

Working directory

Set working the directory if needed. We'll be loading data "by hand" so this probabl isn't necessary

Load Brook Trout Data

These data are from Example 12.4 “So long; thanks to all the fish” in Whitlock and Shulter 2nd ed.

Load data by hand

We’ll 1st load three piecies of info into vectors then put them together into a dataframe with 3 coluns

1) Make A vector to hold the status of the stream - whether brook trout are present or absent

```
brook.trout.PRES.ABS <- c("present","present","present","present","present","present",
                          "absent", "absent", "absent", "absent", "absent", "absent")
```

Advanced technique

This is **OPTIONAL**: A nice trick is this, using the rep() command. Note the c() that surrounds BOTH of the rep() commands.

```
  brook.trout.PRES.ABS <- c(rep("present",6,),rep("absent",6))
```

2) The number of salmon released

```
salmon.released <- c(820,960,700,545,769,1001,467,959,1029,27,998,936)
```

3) The number of salmon that survived

```
salmon.surv <- c(166,136,153,103,173,188,180,178,326,7,120,135)
```

4) Make a **dataframe** from the 3 vectors

```
salmon <- data.frame(brook.trout.PRES.ABS,
                     salmon.released,
                     salmon.surv)
```

THe resulting dataframe as 12 rows and 3 columns

```
dim(salmon)
```

```
## [1] 12  3
```

OPTIONA: Load via .csv file

If the data are in a .csv file in the working directory they could be loaded as. This does NOT need to be done if you have already loaded it by hand as above.

```
  salmon <- read.csv("Lab7_data_brook_trout.csv")
```

Examine the brook trout data

Each line of data is a sperate stream that either has introduced brook trout present or not present.

```
# size of dataframe
dim(salmon)

#top of data
head(salmon)

#bottom of data
tail(salmon)
```

```r
#summary of data
summary(salmon)
```

Data setup

- The **response variable** in this study is the **percent (%)** of chinook salmon that survived over the course of the study.

- We therefore need to calcualte this value using R’s basic math function for divison.

Calcualte survival rates

1st, calcualte the fraction of fish that survived in each stream

- Note the use of the “$” operator to access the columns
- We acces the survival column using “salmon*salmon.surv”anddividethisbythenumberreleasedin”salmon*salmon.released”

```r
percent.surv <-salmon$salmon.surv/salmon$salmon.released
```

2nd, check to make sure that the data make sense.

- The data are a percentage, so each datum should be between 0 and 1.

- We can look at the data using **summary()**
- Note that we can use summary on a single vector (percent.surv) or on a dataframe (salmon)

```r
summary(percent.surv)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.120   0.175   0.196   0.215   0.234   0.385
```

We could also use range(), min(), and max(). to get this info. We could also make a histogram using hist().

3rd, add the percentages to the original dataframe

- We’ll call the new column “percent.surv” also.

- We make a new column by typing “salmon$percent.surv”
- “<- percent.surv” adds the “percent.surv” vector to this new column in the dataframe

```r
salmon$percent.surv <- percent.surv
```

Note: this isn’t actually a percent, but a fraction. To get a percent we’d multiple it by 100.

4th, check the calculated data

```r
#Look at the whole datafame
salmon
```

```
##    brook.trout.PRES.ABS salmon.released salmon.surv percent.surv
## 1               present             820         166       0.2024
## 2               present             960         136       0.1417
## 3               present             700         153       0.2186
## 4               present             545         103       0.1890
## 5                present             769         173        0.2250
## 6                present            1001         188        0.1878
## 7                 absent             467         180        0.3854
## 8                 absent             959         178        0.1856
## 9                 absent            1029         326        0.3168
## 10                absent              27           7        0.2593
## 11                absent             998         120        0.1202
## 12                absent             936         135        0.1442
```

```
#summary() of whole dataframe
summary(salmon)
```

```
##  brook.trout.PRES.ABS salmon.released  salmon.surv   percent.surv
##  absent :6            Min.   :  27    Min.   :  7   Min.   :0.120
##  present:6            1st Qu.: 661    1st Qu.:131   1st Qu.:0.175
##                       Median : 878    Median :160   Median :0.196
##                       Mean   : 768    Mean   :155   Mean   :0.215
##                       3rd Qu.: 970    3rd Qu.:178   3rd Qu.:0.234
##                       Max.   :1029    Max.   :326   Max.   :0.385
```

Data exploration

- We should explore the data using boxplots and histograms.

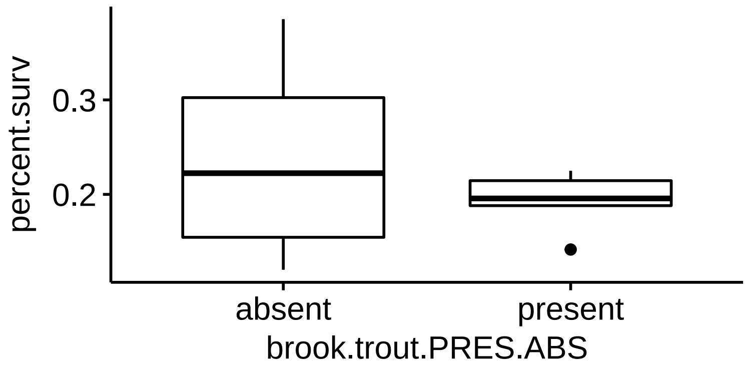
- The **response variable** is the numeric variable "percent.surv" (again, actually a fracation, sorry…)
- The **predictor variable* is a** categorical variable** the indicates whether introduced brook trout are present or absent.

Boxplots

Basic boxplot

- We can make a boxplot using ggpubr
- Note that in ggpubr, the names of variables occur in quotes "…"

```
ggboxplot(data = salmon,
          y = "percent.surv",
          x = "brook.trout.PRES.ABS")
```

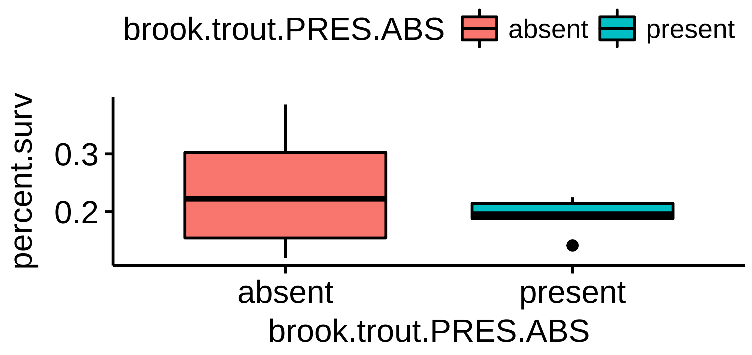


Median survival is a bit lower when brook trout are present, but there is not much difference.

Fancier boxplot

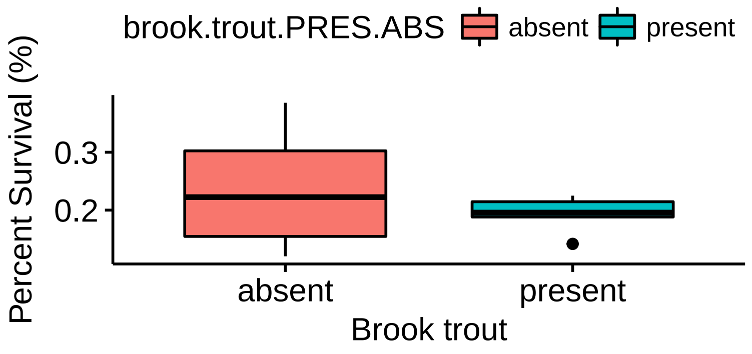
- We can color-code the plot using fill = "brook.trout.PRES.ABS"
- Again, note the quotes around "brook.trout.PRES.ABS" in both places it occurs

```
ggboxplot(data = salmon,
          y = "percent.surv",
          x = "brook.trout.PRES.ABS",
          fill = "brook.trout.PRES.ABS")#
```



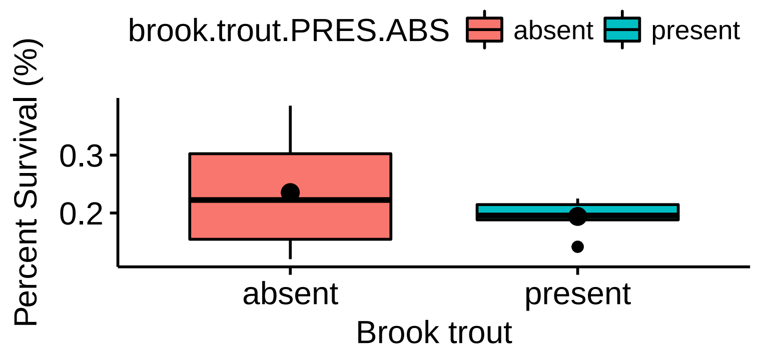
We can change our axes lables using xlab = and ylab =

```
ggboxplot(data = salmon,
          y = "percent.surv",
          x = "brook.trout.PRES.ABS",
          fill = "brook.trout.PRES.ABS",
          xlab = "Brook trout",          #
          ylab = "Percent Survival (%)")#
```



- Boxplot indicate medians using a thick line. We can indicate where the mean is using "add = mean"
- The mean shows up as a dot.

```
ggboxplot(data = salmon,
          y = "percent.surv",
          x = "brook.trout.PRES.ABS",
          fill = "brook.trout.PRES.ABS",
          xlab = "Brook trout",
          ylab = "Percent Survival (%)",
          add = "mean")#
```



OPTIONAL: Classic Boxplot

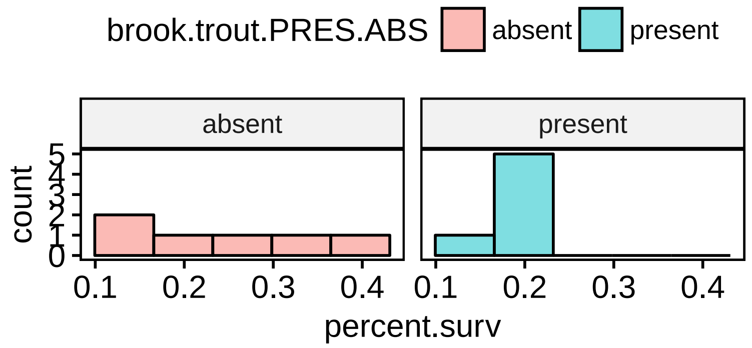
For reference, this is the old-school, pre-ggplot way of doing this. This is **just shown for comparison**.

```
  boxplot(percent.surv~brook.trout.PRES.ABS,
          data = salmon)
```

Histograms

This dataset is a bit small so a histogram looks a bit awkward unless you change the deafults. The code below makes a pretty good one. I use "facet.by = . . . " to split the graph into 2 panels. Note that I set "bins = 5." What happens when you remove this?

```
gghistogram(data = salmon,
            x = "percent.surv",
            fill = "brook.trout.PRES.ABS",
            bins = 5,
            facet.by = "brook.trout.PRES.ABS")
```



Calculating mean and standard deviation

We are going to using the dplyr package to extract information from our data.

Reminder: you dataframe

If we just highlight the word "salmon" and run it we get our whole dataframe

```
salmon
```

```
##    brook.trout.PRES.ABS salmon.released salmon.surv percent.surv
## 1               present             820         166       0.2024
## 2               present             960         136       0.1417
## 3               present             700         153       0.2186
## 4               present             545         103       0.1890
## 5               present             769         173       0.2250
## 6               present            1001         188       0.1878
## 7                absent             467         180       0.3854
## 8                absent             959         178       0.1856
## 9                absent            1029         326       0.3168
## 10               absent              27           7       0.2593
## 11               absent             998         120       0.1202
## 12               absent             936         135       0.1442
```

dplyr's summarize function

- dplyr has a function called summarize that can calcualte means and other thigns for us.
- we can tell it to summarize a column from the salmon dataframe using a "pipe", indicated by "%>%".

- Within the summarize() command we tell it how to summarize the data, mean(), and what column to summarize, percent.surv

```
salmon %>%
    summarize(mean(percent.surv))
```

```
##   mean(percent.surv)
## 1             0.2147
```

2 summaries at the same time

- We can go further and cacaluate the mean and the SD.
- Remember the comma between the mean() and sd() line
- The mean is the the 1st column, and the SD is in the 2nd column.

- (If you can't see the SD, made your console window larger)

```
salmon %>%
    summarize(mean(percent.surv),
              sd(percent.surv))
```

```
##   mean(percent.surv) sd(percent.surv)
## 1             0.2147          0.07585
```

OPTIONAL We can copare theese results to these two other ways of doing thigns

```
    #mean on a single column
    mean(salmon$percent.surv)
```

```
## [1] 0.2147
```

```
    #summary on a single column
    summary(salmon$percent.surv)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.120    0.175    0.196    0.215    0.234    0.385
```

dplyr's group_by() function

dplyr has a function called group_by() that can seperate out data based on a categorical variable. On its own it doesn't do any thing fancy, but in general we can tell it to group things from a dataframe like this. (Don't run this code - its output isn't really useful)

```
salmon %>%
  group_by(brook.trout.PRES.ABS)
```

Again, Note that the "%>%" is called a "pipe" that tells that group() function to work on the slamon dataframe

Using group_by() with summarize()

- We can "pipe" things together in a series of steps.

- 1st, we declare out dataframe, then we use group_by() to split it aprt
- 2nd we use summarize(. . .) to carry out a summary function
- Note that the pipe command %>% occurs twice

```
salmon %>%
  group_by(brook.trout.PRES.ABS) %>%
    summarize(mean(percent.surv))
```

```
## # A tibble: 2 x 2
##   brook.trout.PRES.ABS `mean(percent.surv)`
##                 <fctr>                <dbl>
## 1               absent               0.2353
## 2              present               0.1941
```

We can go further and cacaluate the mean and the SD for both groups. The mean is the the middle column, and the SD is in the far right column. (If you can't see the SD, made your console window larger)

```
salmon %>%
  group_by(brook.trout.PRES.ABS) %>%
    summarize(mean(percent.surv),
              sd(percent.surv))
```

```
## # A tibble: 2 x 3
##   brook.trout.PRES.ABS `mean(percent.surv)` `sd(percent.surv)`
##                 <fctr>                <dbl>              <dbl>
## 1               absent               0.2353            0.10369
## 2              present               0.1941            0.02979
```

We can save our output like by assigning it to an R object like this

```
my.summary <- salmon %>%
  group_by(brook.trout.PRES.ABS) %>%
    summarize(mean(percent.surv),
              sd(percent.surv))
```

And then look at it

```
my.summary
```

```
## # A tibble: 2 x 3
##   brook.trout.PRES.ABS `mean(percent.surv)` `sd(percent.surv)`
##                 <fctr>                <dbl>              <dbl>
## 1               absent               0.2353            0.10369
## 2              present               0.1941            0.02979
```

Adding labels

- The output of dply is a bit "verbose"
- We can shorten it by adding our own shorter lables within the summarize() function
- This is similar to how data.frame() works

```
my.summary <- salmon %>%
  group_by(brook.trout.PRES.ABS) %>%
    summarize(mean = mean(percent.surv),
              sd = sd(percent.surv))
```

Getting the standard error (SE)

- We need the SD and the sample size to get the standard error (SE)
- The sample size can be determined using the length() function

Getting the sample size

```
my.summary <- salmon %>%
  group_by(brook.trout.PRES.ABS) %>%
    summarize(mean = mean(percent.surv),
              sd = sd(percent.surv),
              N = length(percent.surv))
```

Calculating the SE

```
my.summary$SE <- my.summary$sd/sqrt(my.summary$N)
```

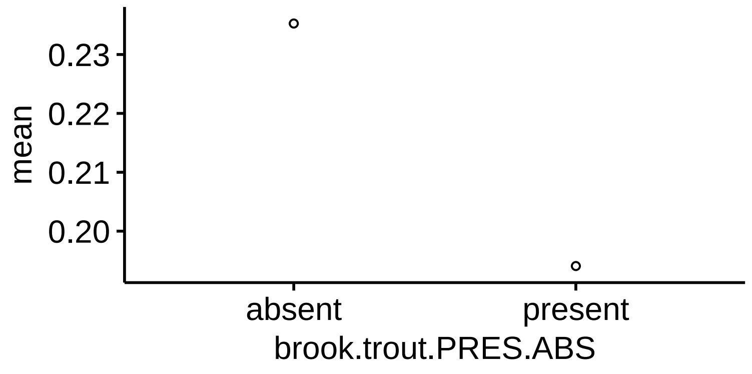
Plotting the mean and error bars

Boxplot and histograms are good for exploring the oveall distribution of the data. We typically want to plot the means with error bars when we are doing hypothesis testing. Note that while the SD tells us about how variable the data are, the standard error tells us about how precise our estiamtes are, and therefore how confident we can be that two means are different from each other (or whether we can be pretty sure they are similar).

Plotting means

First, we can plot the means like this.

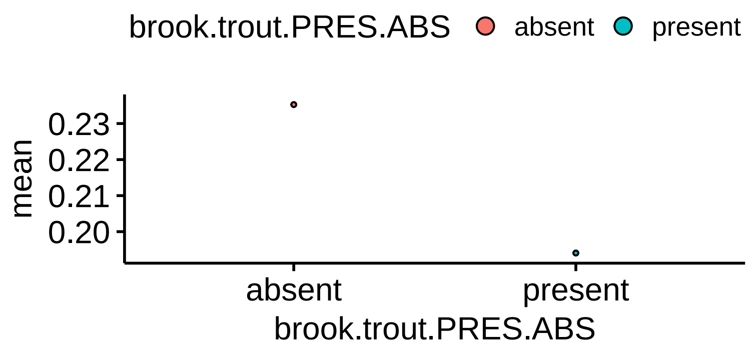
```
ggdotplot(data = my.summary,
          y = "mean",
          x = "brook.trout.PRES.ABS")
```



IGNORE THE ERROR "`stat_bindot() using bins = 30. Pick better value with binwidth`."

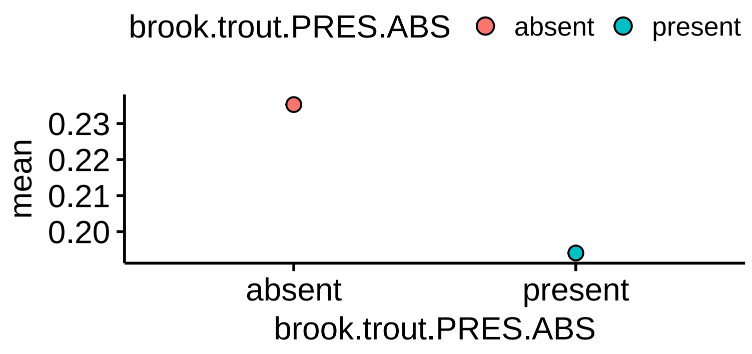
ANd give them color by adding "fill ="

```
ggdotplot(data = my.summary,
          y = "mean",
          x = "brook.trout.PRES.ABS",
           fill = "brook.trout.PRES.ABS")
```



We can make things a bit bigger like this Even better than just color is color AND symbol, using shape = . . .

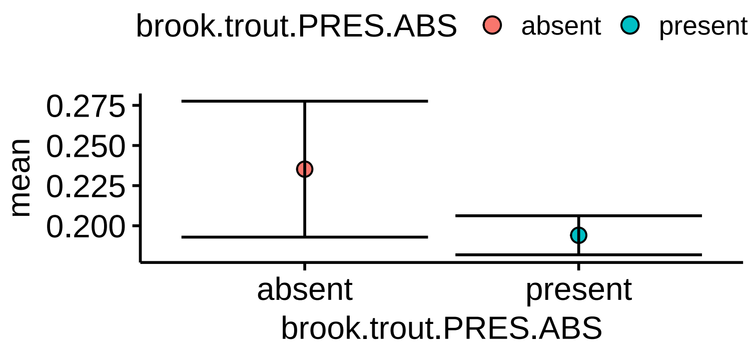
```
ggdotplot(data = my.summary,
          y = "mean",
          x = "brook.trout.PRES.ABS",
          fill = "brook.trout.PRES.ABS",
          size = 3)
```



Plotting error bars w/ geom_errorbar()

We can combine the ggpubr command "ggdotplot()" and the regular ggplot2 command geom_errorbar() to plot the means and error bars.

```r
ggdotplot(data = my.summary,
          y = "mean",
          x = "brook.trout.PRES.ABS",
          fill = "brook.trout.PRES.ABS",
          size = 3) +
  geom_errorbar(aes(ymax = mean + SE,
                    ymin = mean - SE))
```



The really wide bars extension on the error bars are goofy, so we can get ride of them by using "width = 0" within geom_errorbar().

```r
ggdotplot(data = my.summary,
          y = "mean",
          x = "brook.trout.PRES.ABS",
          fill = "brook.trout.PRES.ABS",
          size = 3) +
  geom_errorbar(aes(ymax = mean + SE,
                    ymin = mean - SE),
                width = 0)
```

- And we can make the error bars wide by using "size =" within geom_errorbar()

```r
ggdotplot(data = my.summary,
          y = "mean",
          x = "brook.trout.PRES.ABS",
          fill = "brook.trout.PRES.ABS",
          size = 3) +
  geom_errorbar(aes(ymax = mean + SE,
                    ymin = mean - SE),
                width = 0,
                size = 2)
```