Lab 1: How is T-test Unreliable?

Trang Nguyen, Dennis Liu

3/9/2017

Objective

This lab series, which consists of 3 labs, aims at demonstrating some bad practices of experiment design and statistical analysis, and how these bad practices can give misleading conslusions. More particularly, we will look at the impact of **data cleaning**, **lurking variables** and **sample size** on statistical conclusions students can draw for the same question. After the lab, students should be able to understand at least 3 things:

* **Outliers**: Initial data expoloration to identify outliers is a very important step. These outliers might change your statistical conclusion radically. Therefore, you need to identify outliers and justify whehter to omit or keep those data points.

* **Lurking Variables:** Though it's impossible to put all lurking variables into your analysis, it's important to design an experiment in a way that some lurking variables can be minimized, and to conduct a statistical analysis with the concern that some lurking variables might impact the conclusion.

* **SampleSize:** It is a misconception that all statistical analyses with big data sample give you the right conclusion. A well-designed experiment with 20 subjects might be better than a badly-designed experiment with 200 subjects.

Outline:

- **Lab 1**: How is T-test Unreliable?
  This lab will show the impact of data cleaning on statistical conclusions and how t-tests can give dramatically different conclusions for the same question.

- **Lab 2**: ANOVA

- **Lab 3**: CART analysis

Background:

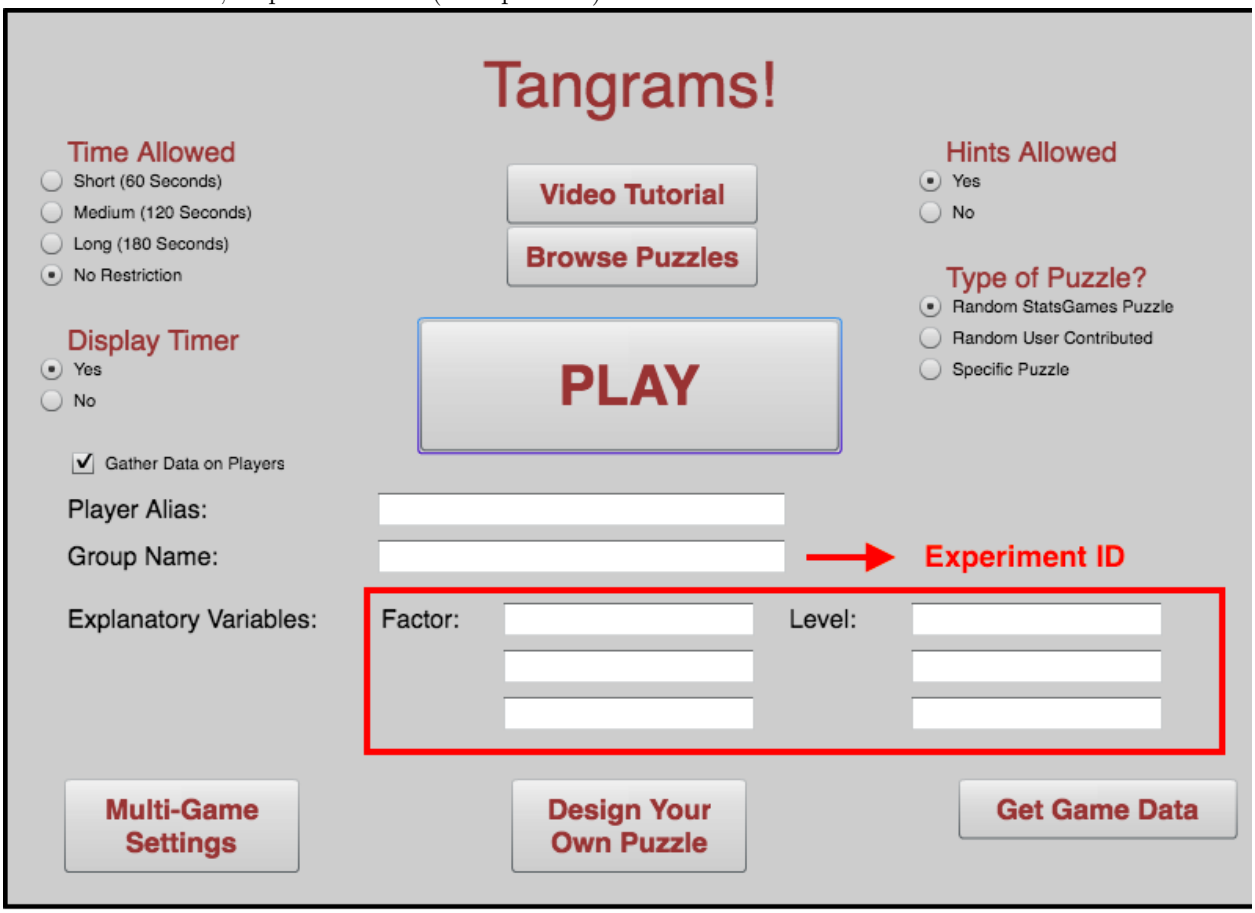
Previous Research on the Impact of Data Cleaning and Model Assumption Checking

A bigger picture of the reliability of a statistical test is the reproducibility of a study. To be more clear, statistical reproducibility refers to transparent raw data processing from collecting to cleaning (Peng, 2009). More than 70% of researchers have tried and failed to reproduce another scientist's experiments, and more than half have failed to reproduce their own experiments (Nature, 2016). There have been many cases in which non-reproducible research make it harder to verify the results and cause dangerous consequences to the society.

Since statistical reproducibility is a big concept, this lab will approach this topic from a specific aspect which is practical and important for student researchers: *what questions a student researcher should him/herself when drawing a statistical conclusion?*. Cummiskey, Kuiper, and Sturdivant (2012) pointed out that though many statistics courses mention model assumptions (for example, normality for *t-test*) and data cleaning, students rarely realize the impact of violated model assumptions or outliers on the conclusions they draw from statistical tests.

Data Used: The Tangrams Game and Lab

Tangrams is an ancient Chinese puzzle where players arrange geometrically shaped pieces into a particular design by flipping, rotating, and moving them. The online Tangrams game allows students the opportunity to design many versions of the original game. You can go to the Tangrams website and leave all the variables blank when you are simply trying out the game. However, if you want to find your completion time in the database of results, a specific course (Group Name) will be needed.



In this activity, we will use Tangram database to look at some experiments (identified by **Group Name**) and to look at the collection of all experiments (about *50,000* observations).

1. Getting Data

You can import the raw dataset *tangrams_data.csv* into RStudio. This dataset is a collection of all experiments using **Tangrams Game and Lab** in the past. Each row is a game trial. Experiments can be identified by **Group Name**.

```
tangram_data <- read.csv("tangrams_data.csv", stringsAsFactors = F)
```

One alternative to using a local static csv file is to stream data live from the **Tangrams Game and Lab** website by using the library **RCurl**. However, for the sake of this lab, students will use the csv file imported as menioned above.

2. Re-formating and Cleaning Data

Step 1. Re-formatting Data

This raw dataset is not ready to be put in any statiscal analysis yet. Therefore, we need to reformat the data into a more accessible form. For the future convinence, we change the type of the column into factor and some into numeric.

```
tangram_data$RequestedTime <- as.numeric(tangram_data$RequestedTime)
tangram_data$NumClicks <- as.numeric(tangram_data$NumClicks)
tangram_data$TimeUsed <- as.numeric(tangram_data$TimeUsed)
tangram_data$NumShapes <- as.numeric(tangram_data$NumShapes)
```

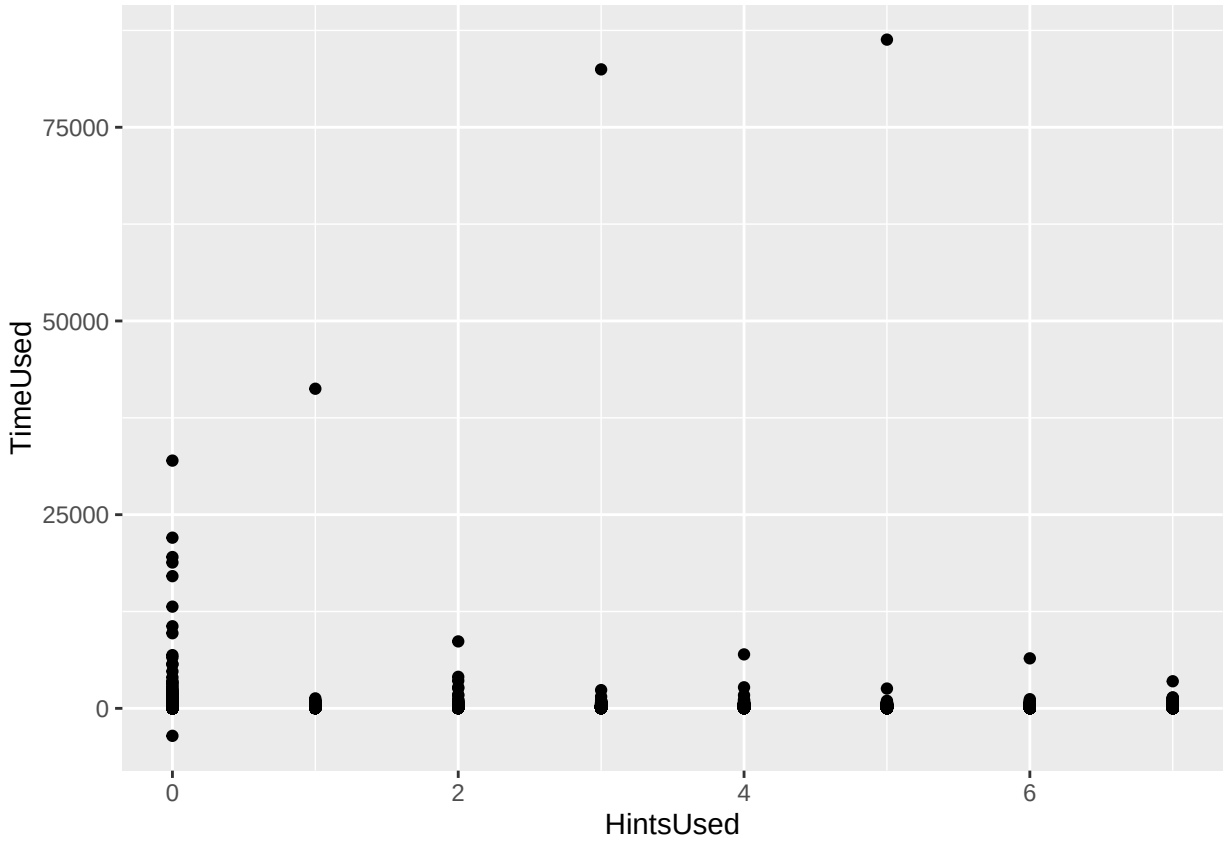
As mentioned above, we plan to conduct analyses on multiple experiments identified by *Group Name*. However, there are some typos and cap inconsistency in the dataset, which is very typical of self-input data.

```
tangram_data$GroupName <- tolower(tangram_data$GroupName)
```

Step 2. Cleaning Data

You can see that this dataset is quite big and messy. Therefore, we need some initial data exploration to identify some biaes and outliers. The following plot uses **ggplot** to visualize the *TimeUsed* variable relative to *HintUsed*. You can use this algorithm and replace the variable names (x and y) to look at other aspects of the dataset.

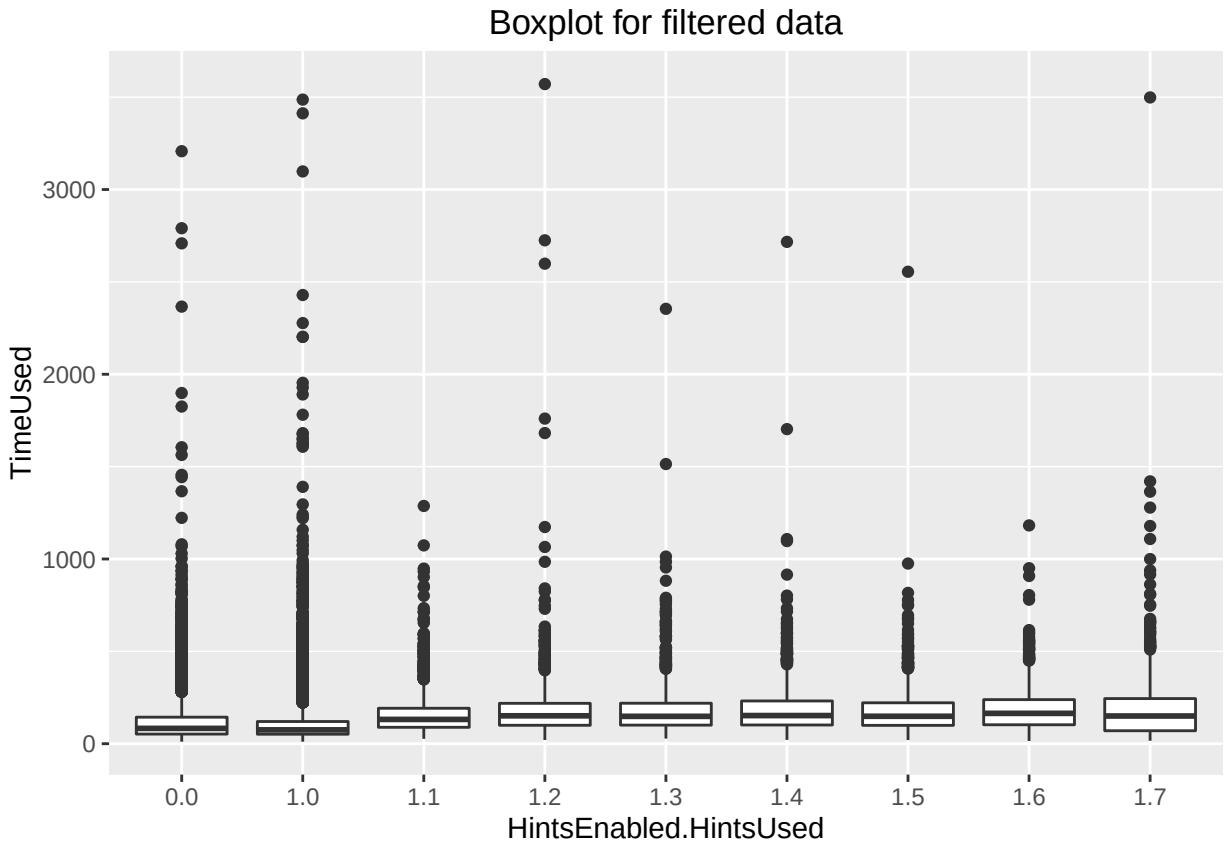
```
ggplot(data=tangram_data) + geom_point(mapping= aes(x=HintsUsed, y=TimeUsed))
```



Question 1: Do you see any extreme outliers? What might be the reasons behind those outliers? How will they impact your later conclusion from the data?

As in the graph, we can see that there are multiple outliners which are biasing our data. Therefore, we want to eliminate those which are more than 3600 seconds(one hour). We also notice that the range of the data is relatively big. We want to take a log transformation on the data. The new dataset is tangram.

```
tangram <- tangram_data[tangram_data$TimeUsed < 3600,]
tangram <- tangram[tangram$TimeUsed > 0,]
tangram$TimeUsed_log <- log(tangram$TimeUsed)
ggplot(data = tangram, aes(x=interaction(HintsEnabled,HintsUsed,lex.order = TRUE), y=TimeUsed)) + geom_
```



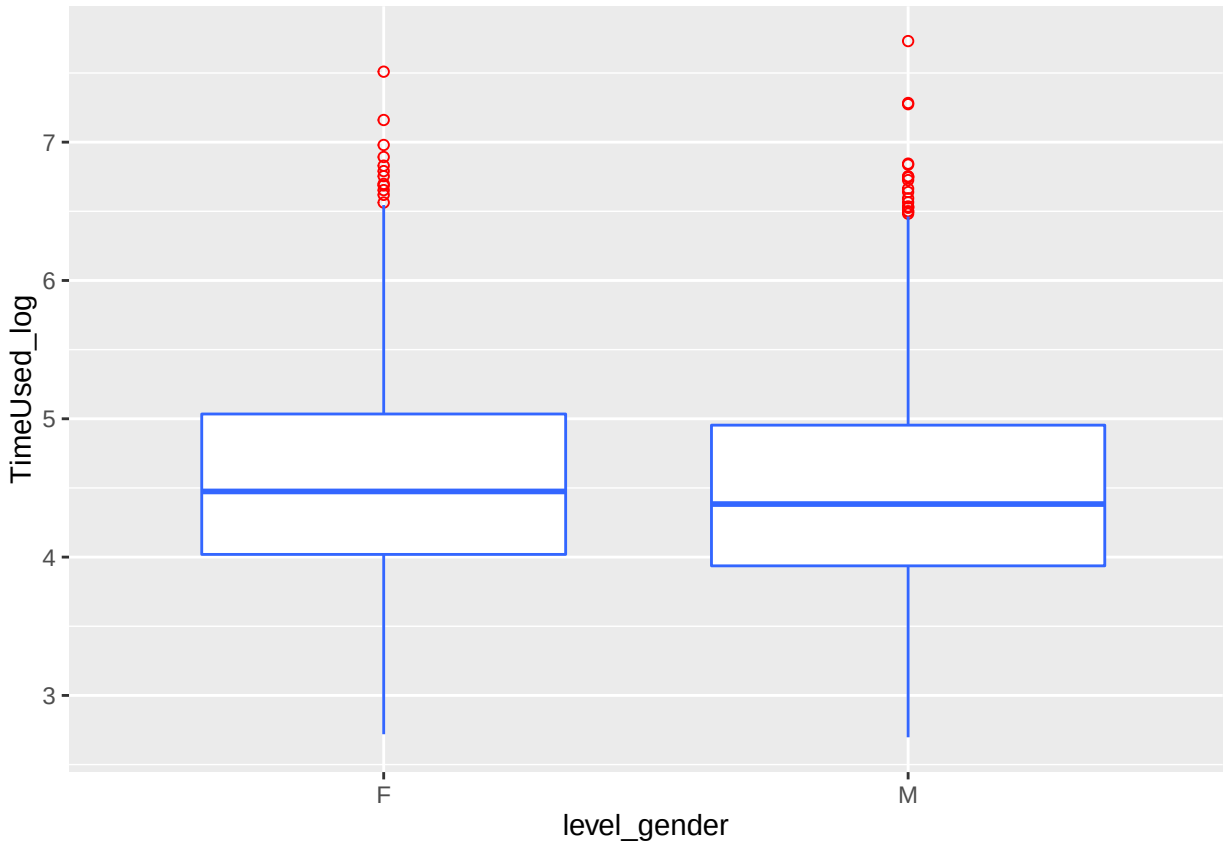
We do notice that there are multiple useful information in the customized factor and level section. The most interesting one is the gender information. We want to extract data with gender information from the larger dataset.

```
tangram <- as.data.frame(tangram)
### Factor to be filtered: GENDER
### A set of criteria for regular expressions/ key patterns
gen = c("^gen","^sex")
gen_male = c("^m","^h")
gen_female = c("^f", "^mu")

### ismatch
### @input: cond: a vector of key patterns (e.g. gen)
###         x, y, z: 3 columns to look for key
### @return: 0 if no match
###          index of the factor (1, 2, 3() if there is match
ismatch <- function(cond, x, y, z) {
```

```
  x <- grep(paste(cond, collapse = "|"), c(x, y, z), ignore.case = T, value = FALSE)
  return (ifelse(length(x), as.numeric(x), as.numeric(0)))
}

### level_gen_fun
### @input: x is the col index (given by factor_gender)
###         y is the row index
### @return: -1 if no gender factor indicated
###          -2 if gender factor indicated but level_gen key patterns not matched (require future manual
###           M/F
### @note: this function is specific to gender only. Needs to think about how to generalize it.
level_gen_fun <- function(x, y) {
  return (ifelse(x == 0, -1,
                 ifelse(grepl(paste(gen_female, collapse = "|"),tangram[y,2*x+3], ignore.case = T),
                        "F",
                        ifelse(grepl(paste(gen_male, collapse = "|"),tangram[y,2*x+3], ignore.case = T)
                               "M",-2))))
}
tangram <- as.data.table(tangram)
tangram[, factor_gender := ismatch(gen, Factor1, Factor2, Factor3), by = 1:nrow(tangram)]
tangram <- as.data.frame(tangram)
tangram$level_gender <- mapply(level_gen_fun, tangram$factor_gender, 1:nrow(tangram), SIMPLIFY = TRUE)
gen_tangram <- tangram[tangram$level_gender == "F"|tangram$level_gender == "M",]
```

Now, there are two new columns in our dataset: factor_gender and level_gender. factor_gender indicates that whether this specific data entry contains gender information. level_gender indicates the specific gender information. In the level gender column, -1 means there is no gender information at all. -2 means that gender information is indicated, however, it does not match. F means Female and M means Male. We create a new dataset which contains all valid gender information. (gen_tangram)

```
ggplot(gen_tangram,aes(level_gender,TimeUsed_log)) +
  geom_boxplot(varwidth = TRUE,fill = "white", colour = "#3366FF",
               outlier.colour = "red", outlier.shape = 1)
```



You can refer to an additional guidline *factor-filter.R* for more detailed instructions regarding creating new factor columns like STEM, age, and major.

3. Analyzing Data

In this activity, we will use ** T-test ** in big and small datasets and in various ways to answer the same question: **"Does a player's gender influence his or her completion time of a game?"**

Recall that from now on, we will use **gen_tangram** to conduct our analysis.

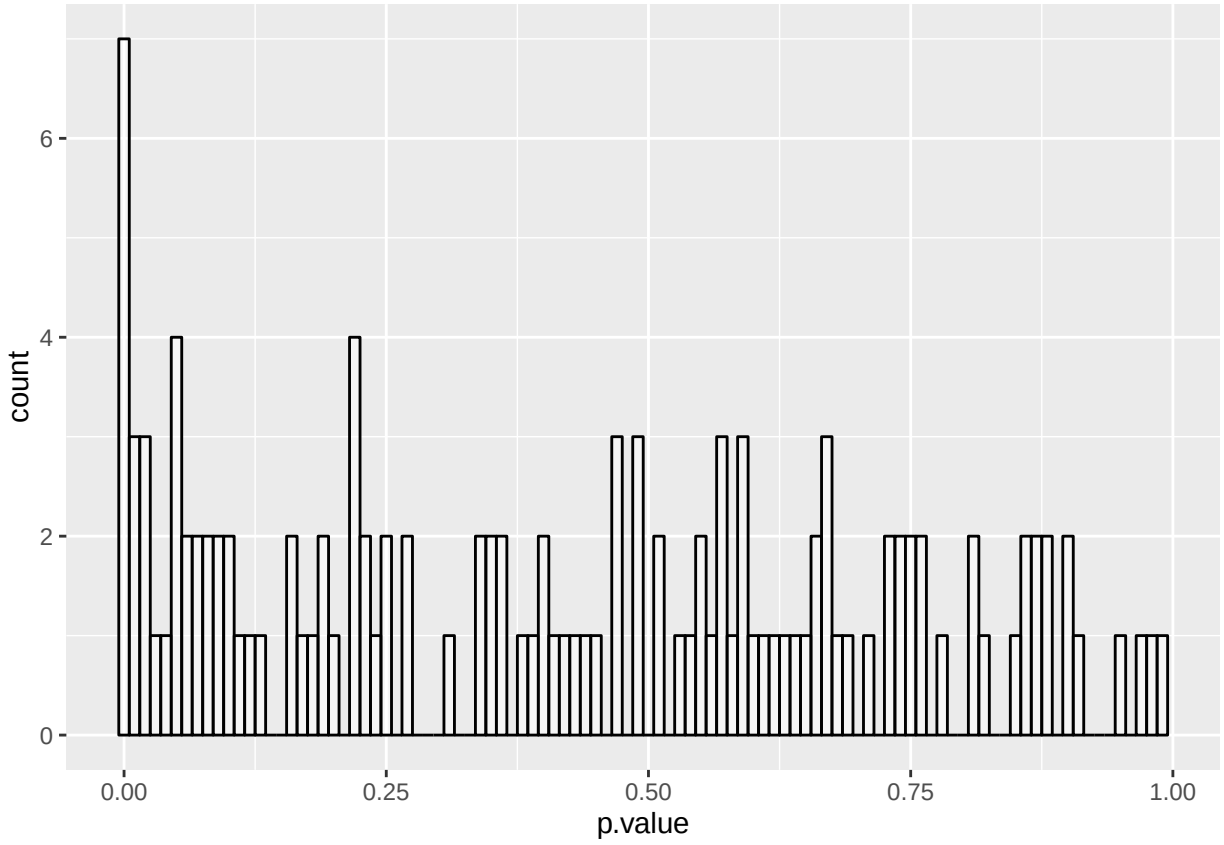
We start with the basic **T-test** on each experiment with ID (*Group Name*) to see whether the difference in *Time Used (log)* between *Female* and *Male* group is significant at $\alpha = 0.1$.

> Question 2: Use the following code to conduct t-test on each experiment with ID in the dataset and describe the distribution of p-value?

```
tangram_pvalue <- as.data.table(gen_tangram)
tangram_pvalue[, `:=`( SampleSize = .N) , by = GroupName]
tangram_pvalue <- tangram_pvalue[tangram_pvalue$SampleSize >= 5,]
tangram_pvalue[, `:=`(log_timeused_mean = mean(TimeUsed_log), timeused_mean = mean(TimeUsed)), by = c("
tangram_pvalue[, `:=`(n = .N, sd = sd(TimeUsed_log)), by = c("GroupName", "level_gender")]

ttestFun <- function(dat) {
  if (sum(dat$level_gender == "F") > 1 && sum(dat$level_gender == "M") > 1) {
    the_fit <- t.test(TimeUsed_log ~ level_gender, data = dat)
    #setNames(the_fit$p.value, "p.value")}
    c("p.value" = the_fit$p.value, "samplesize" = mean(dat$SampleSize))}
```

```
  else {
    c("p.value" = -1, "samplesize" = mean(dat$SampleSize))
  }
}
alpha <- 0.1
gender_pval_dist <- ddply(tangram_pvalue, ~ GroupName, ttestFun)
gender_pval_dist$significant <-(gender_pval_dist$p.value < alpha)*1
gender_pval_dist$significant <- as.factor(gender_pval_dist$significant)
```

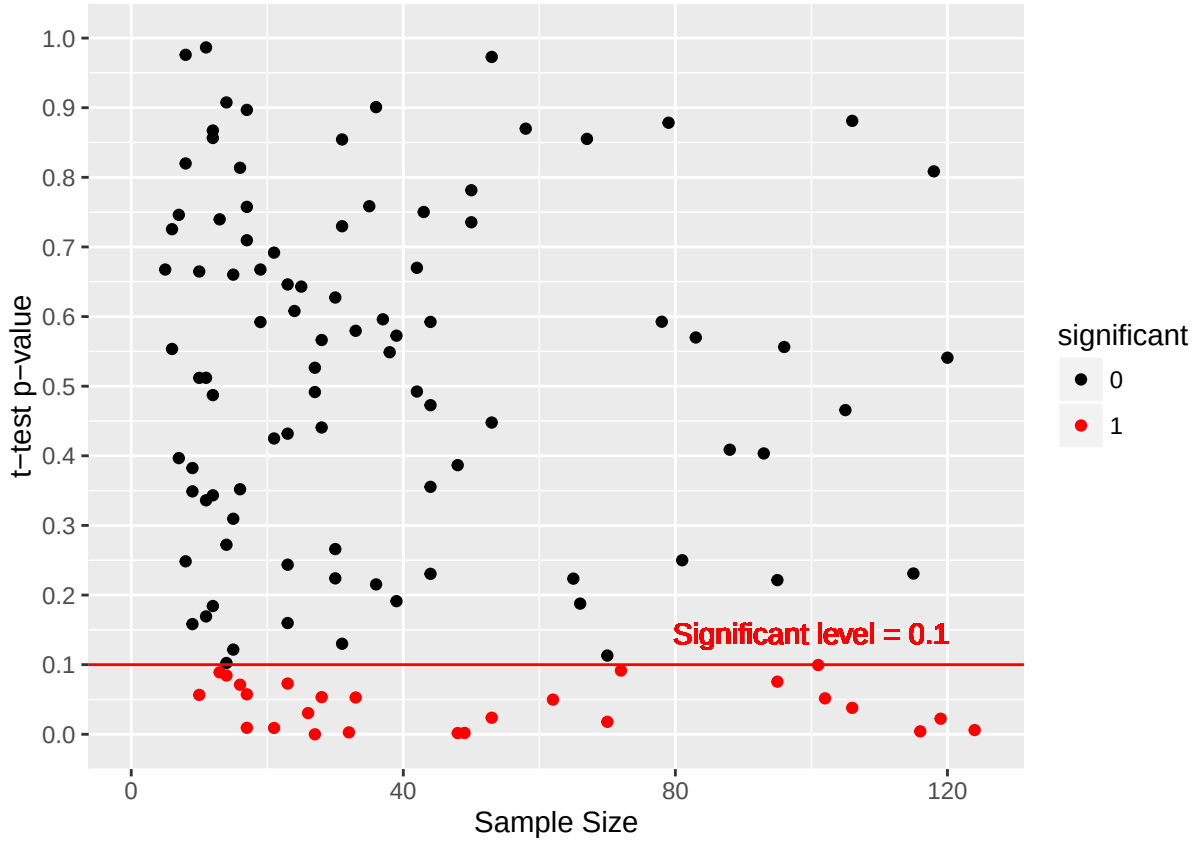
```
ggplot(gender_pval_dist[gender_pval_dist$p.value != -1,], aes(p.value)) +
  geom_histogram(binwidth = 0.01, position = "identity", alpha = 0.5, colour = "black", fill = "white")
```



Question 3: We also want to see how p-value changes relative to the experiment's sample size. Use the following code to draw a scatterplot of p-value vs. sample size. How many experiments show a signficant p-value? Does big sample size guarantee a stable and reliable p-value?

```
ggplot(gender_pval_dist[gender_pval_dist$p.value != -1,], aes(x = samplesize, y = p.value)) +
  geom_point(aes(colour = significant)) +
  scale_color_manual(values=c("black", "red")) +
  geom_hline(aes(yintercept = alpha), color = "red") +
  geom_text(aes(100,0.1,label = "Significant level = 0.1", vjust = -1), color = "red") +
  scale_x_continuous(name="Sample Size", limits=c(0, 125)) +
  scale_y_continuous(name="t-test p-value", limits=c(0, 1), breaks = seq(0,1, by = 0.1))
```

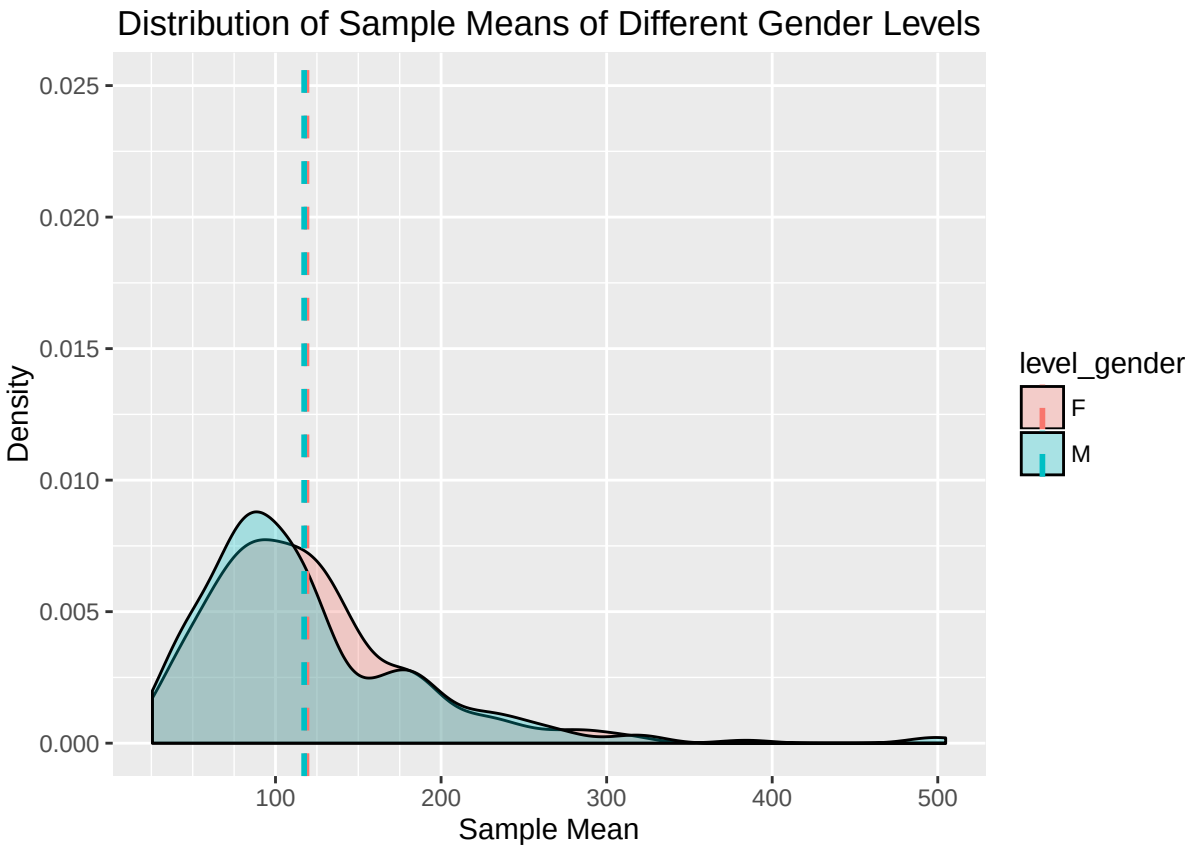
```
## Warning: Removed 8 rows containing missing values (geom_point).
```



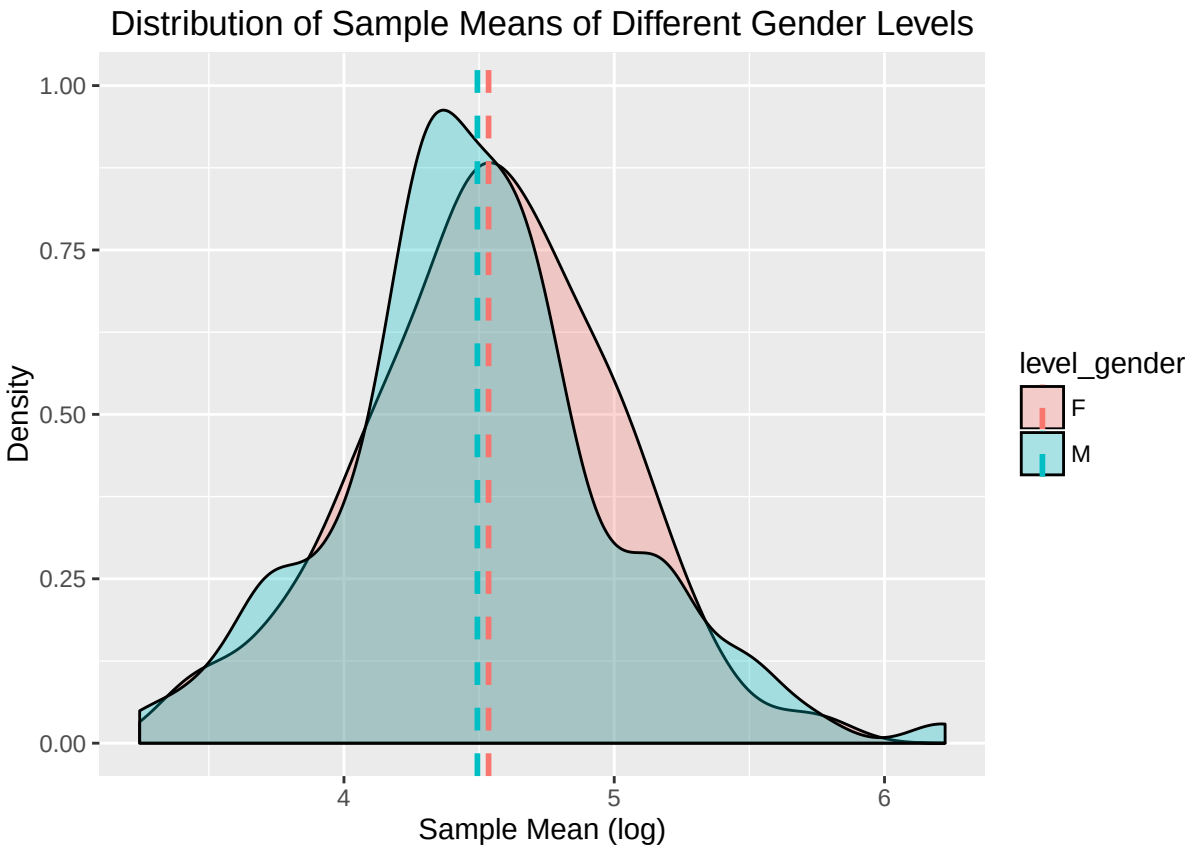
Question 4: Now let's look at the sampling distribution of Female and Male group. Use the following code to visualize the sampling distributuion of Time Used and Time Used (log) in two groups. Do you think there is a significant difference? Explain some concerns in drawing the conclusion.

```
gender_samplingdis <- as.data.table(gen_tangram)
gender_samplingdis[,`:=`(SampleSize = .N), by = GroupName]
gender_samplingdis[, `:=`(log_timeused_mean = mean(TimeUsed_log), timeused_mean = mean(TimeUsed)), by =
gender_samplingdis[, `:=`(n = .N, sd = sd(TimeUsed_log)), by = c("GroupName", "level_gender")]
gender_samplingdis <- gender_samplingdis[,c("GroupName", "level_gender", "SampleSize", "timeused_mean",
gender_samplingdis <- gender_samplingdis[!duplicated(gender_samplingdis),]

### Aggregate data for vertical lines
gender_samplingdis_vline <- gender_samplingdis %>%
  group_by(level_gender) %>%
  summarise(log_timeused_mean = mean(log_timeused_mean), timeused_mean = mean(timeused_mean))
```

```
ggplot(gender_samplingdis, aes(timeused_mean, fill = level_gender)) +
  geom_density(alpha = 0.3) +
  geom_vline(data=gender_samplingdis_vline, aes(xintercept=timeused_mean,  colour=level_gender),
             linetype="dashed", size=1) +
  labs(title = "Distribution of Sample Means of Different Gender Levels") +
  labs(x = "Sample Mean", y = "Density") +
  ylim(0.00, 0.025) +
  scale_x_continuous(minor_breaks = seq(0, 200, by = 25))
```




```
ggplot(gender_samplingdis, aes(log_timeused_mean, fill = level_gender)) +
  geom_density(alpha = 0.3) +
  geom_vline(data=gender_samplingdis_vline, aes(xintercept=log_timeused_mean,  colour=level_gender),
             linetype="dashed", size=1) +
  labs(title = "Distribution of Sample Means of Different Gender Levels") +
  labs(x = "Sample Mean (log)", y = "Density") +
  ylim(0.00, 1.00)
```



Take-away Lesson

This lab demonstrates some questions you need to ask yourself when conducting t-test and interpreting its results. In several cases, especially when you analyze data without a good understanding of the design of experiments, T-test can lead to some unreliable conclusions. Fortunately, there are some good alternatives to t-test. One of the most common options is *ANOVA*. The second lab (*Lab 02: ANOVA*) will use *ANOVA* to analyze the same data and demonstrate how this technique can give a more thorough analysis of our data.