Comparisons

Andrew Roth

2024-05-21

Introduction

The next step in our R journey is to look at how to do comparison of groups. We have already seen one example using the `t.test` function. Here we will go into a bit more depth. This will be a relatively short tutorial, since performing the tests is straightforward. The hard part will be wrangling the data which we have already covered.

Setup

First let's load the libraries we will need.

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)
```

We will use the diabetes data again. Let's load it up.

```
my_data <- read.csv("/home/andrew/Desktop/path/Diabetes_Full.csv")
head(my_data)
```

```
##   Random.Blood.Glucose.mg.dL Random.Blood.Glucose.Binary
## 1                        151                         Low
## 2                         75                         Low
## 3                        141                         Low
## 4                        206                        High
## 5                        135                         Low
## 6                         97                         Low
##   Random.Blood.Glucose.Ordinal Age Sex  BMI  BP Total.Cholesterol   LDL HDL TCH
## 1                       Medium  59   2 32.1 101               157  93.2  38   4
## 2                          Low  48   1 21.6  87               183 103.2  70   3
## 3                          Low  72   2 30.5  93               156  93.6  41   4
## 4                         High  24   1 25.3  84               198 131.4  40   5
## 5                          Low  50   1 23.0 101               192 125.4  52   4
## 6                          Low  23   1 22.6  89               139  64.8  61   2
##      LTG Fasting.Glucose
```

```
## 1 4.8598                87
## 2 3.8918                69
## 3 4.6728                85
## 4 4.8903                89
## 5 4.2905                80
## 6 4.1897                68
```

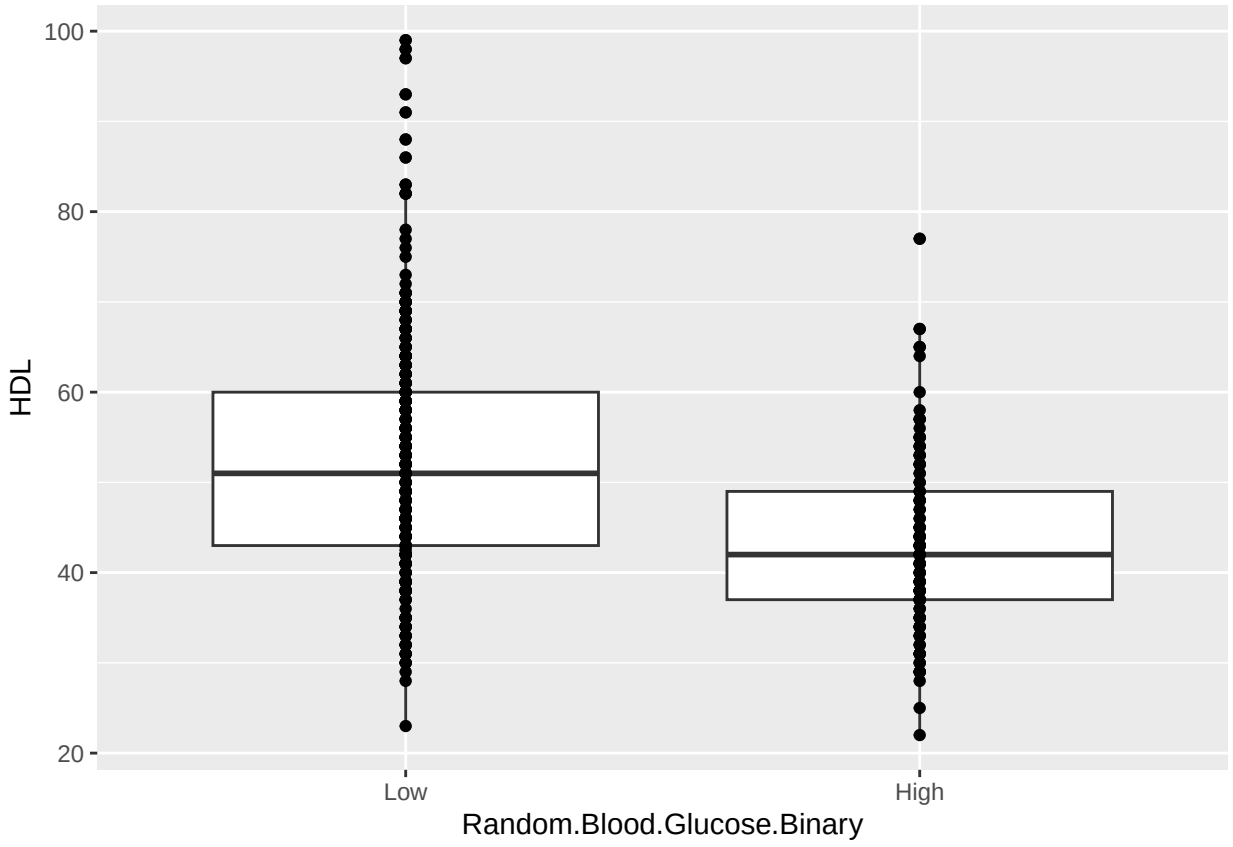
Let's do the data cleanup to get our factors setup.

```
my_data$Random.Blood.Glucose.Binary <- factor(
  my_data$Random.Blood.Glucose.Binary, levels=c("Low", "High")
  )
my_data$Random.Blood.Glucose.Ordinal <- factor(
  my_data$Random.Blood.Glucose.Ordinal, levels=c("Low", "Medium", "High")
  )
my_data$Sex <- as.character(my_data$Sex)
my_data <- mutate(my_data, Sex=recode(Sex, "1"="male", "2"="female"))
my_data$Sex <- factor(my_data$Sex)
```

Two group comparison

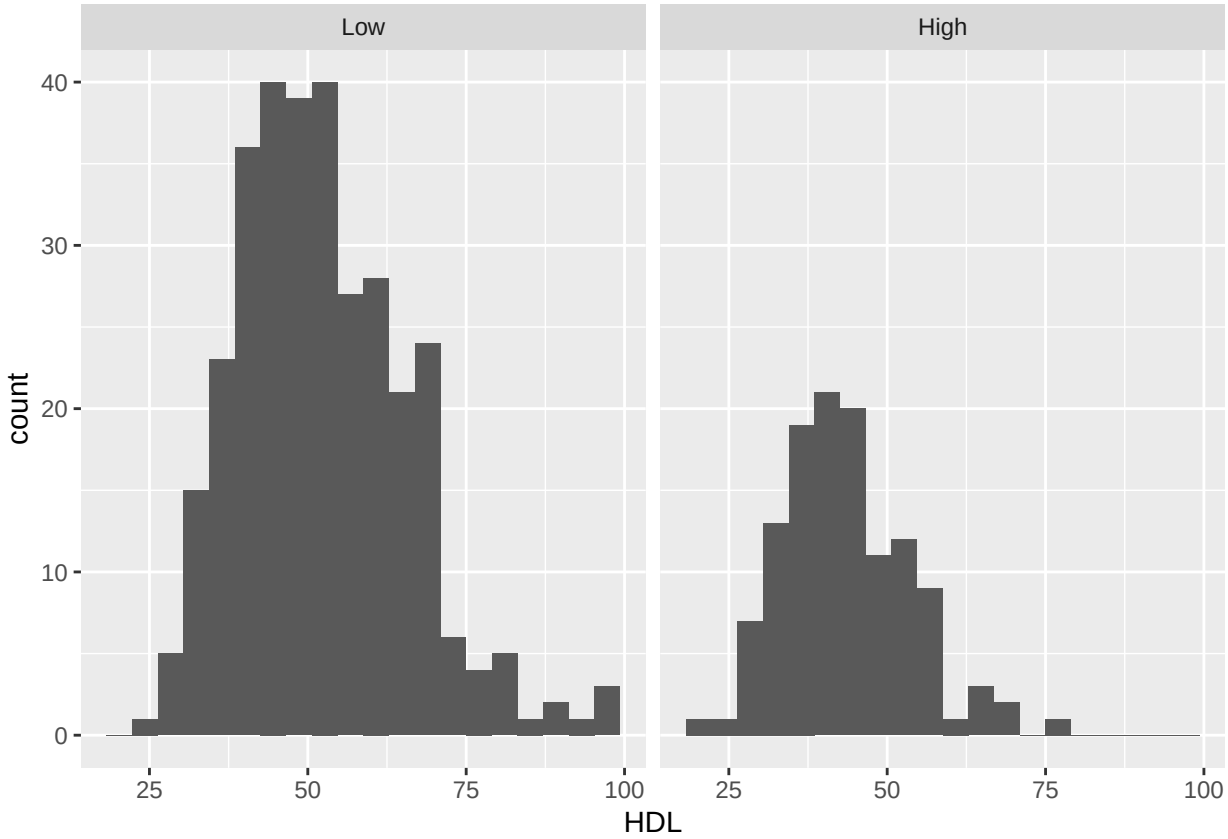
Let's suppose we want to divide our data into low/hi glucose groups and see if there are any statistical differences. Let's consider HDL first and do an exploratory plot.

```
ggplot(my_data, aes(x=Random.Blood.Glucose.Binary, y=HDL)) + geom_boxplot() + geom_point()
```



It looks like there is a difference between groups. The data also looks normal based on the symmetry of inter-quartile ranges. This might be easier to see with histograms though.

```
ggplot(my_data, aes(x=HDL)) +
  geom_histogram(bins=20) +
  facet_grid(~Random.Blood.Glucose.Binary)
```



This looks a bit dubious with a lot of outlying values. At this point we could do a test of normality. Let's use the Shapiro-Wilk test on each group. We will use `dplyr` and pipes to do this.

```
my_data %>%
  group_by(Random.Blood.Glucose.Binary) %>%
  summarise(shapiro.test(HDL)$p.value)
```

```
## # A tibble: 2 x 2
##   Random.Blood.Glucose.Binary `shapiro.test(HDL)$p.value`
##   <fct>                                             <dbl>
## 1 Low                                       0.000000958
## 2 High                                      0.0172
```

The p-value for both groups is <0.05 suggesting the data is not normal.

Let's ignore this for a second and try a t-test though.

```
t.test(
  my_data[my_data$Random.Blood.Glucose.Binary == "Low", "HDL"],
  my_data[my_data$Random.Blood.Glucose.Binary == "High", "HDL"]
  )
```

```
##
##  Welch Two Sample t-test
##
## data:  my_data[my_data$Random.Blood.Glucose.Binary == "Low", "HDL"] and my_data[my_data$Random.Blood
```

```
## t = 7.729, df = 288.53, p-value = 1.81e-13
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##   6.644705 11.185140
## sample estimates:
## mean of x mean of y
##  52.22897  43.31405
```

So our p-value is <0.001 which suggests a significant difference. But we do not think the data is normally distributed, so it would be more appropriate to use a non-parametric test. Let's try a Wilcoxon test.

```
wilcox.test(
  my_data[my_data$Random.Blood.Glucose.Binary == "Low", "HDL"],
  my_data[my_data$Random.Blood.Glucose.Binary == "High", "HDL"]
  )
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  my_data[my_data$Random.Blood.Glucose.Binary == "Low", "HDL"] and my_data[my_data$Random.Blood
## W = 27448, p-value = 2.006e-11
## alternative hypothesis: true location shift is not equal to 0
```

Our p-value is still <0.001 with the non-parametric, so there appears to be a significant difference. This is a fairly large dataset, so even with the loss of power using a non-parametric test we can still detect an effect.

Let's test some other columns for a difference. I'll do this manually and repeat a lot of code. There is a better way, but let's keep the example simple.

```
p_hdl <- wilcox.test(
  my_data[my_data$Random.Blood.Glucose.Binary == "Low", "HDL"],
  my_data[my_data$Random.Blood.Glucose.Binary == "High", "HDL"]
  )$p.value
p_ldl <- wilcox.test(
  my_data[my_data$Random.Blood.Glucose.Binary == "Low", "LDL"],
  my_data[my_data$Random.Blood.Glucose.Binary == "High", "LDL"]
  )$p.value
p_bmi <- wilcox.test(
  my_data[my_data$Random.Blood.Glucose.Binary == "Low", "BMI"],
  my_data[my_data$Random.Blood.Glucose.Binary == "High", "BMI"]
  )$p.value
p_tch <- wilcox.test(
  my_data[my_data$Random.Blood.Glucose.Binary == "Low", "TCH"],
  my_data[my_data$Random.Blood.Glucose.Binary == "High", "TCH"]
  )$p.value
my_pvals <- c(p_hdl, p_ldl, p_bmi, p_tch)
my_pvals
```

```
## [1] 2.006291e-11 2.177578e-03 1.380127e-24 3.313907e-15
```

All look significant. But wait, we just did multiple tests, so we should multiple test correct. We can use the `p.adjust` function for this. This function requires a vector of p-values as the a mandatory argument which is why I create the `my_pvals` variable. Let's see what happens using the Bonferroni correction.

```
p.adjust(my_pvals, method="bonferroni")
```

```
## [1] 8.025165e-11 8.710310e-03 5.520509e-24 1.325563e-14
```

The p-values have increased but they are all less than <0.05. In fact only the value for LDL is >0.001.

We could try a different adjustment as well. Let's use the Benjamini & Hochberg correction.

```
p.adjust(my_pvals, method="BH")
```

```
## [1] 2.675055e-11 2.177578e-03 5.520509e-24 6.627814e-15
```

The changes are less dramatic than using the Bonferroni corrections. This is typical as the BH correction controls a slightly different type of error and generally has more power for multiple testing.

This is not meant to advocate for any given correction, just to show how to perform them!