R Lab #10c - Nonparametric alternative to ANOVA

Contents

Introduction

The ANOVA is the appropriate parametric test when comparing the means of three or more groups. This test has the assumptions that each population has a normal distribution for the variable, and that all populations have the same variance for the variable. Thus, we follow our flow chart that first checks each group for normality (Shapiro-Wilk), and then checks for equal variances (Levene's test). Following this flow chart, if violations are found then a natural log transformation can be applied, and if violations remain then a nonparametric alternative should be used. In this lab we learn how to run the nonparametric **Kruskal-Wallis test** and the corresponding post-hoc **Dunn's test**.

Gene expression and mental illness

In this lab we will use the data presented in question #7 at the end of chapter 15. In this study, researchers were interested in the association between expression of a gene involved in the production and activity of myelin (proteolipid protein 1; *PLP1*) and mental illnesses (bipolar disorder and schizophrenia).

Import and explore data

First, we import the data (file: chap15q07DisordersAndGeneExpression.csv) and look at the structure/summary of the data frame.

```
PLP1data <- read.csv("chap15q07DisordersAndGeneExpression.csv",
    stringsAsFactors = T)
str(PLP1data)
## 'data.frame':    45 obs. of  2 variables:
##  $ group         : Factor w/ 3 levels "bipolar","control",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ PLP1Expression: num  -0.02 -0.27 -0.11 0.09 0.25 -0.02 0.48 -0.24 0.06 0.07 ...
```

```
summary(PLP1data)
##      group    PLP1Expression
##  bipolar:15   Min.   :-0.430
##  control:15   1st Qu.:-0.320
##  schizo :15   Median :-0.220
##               Mean   :-0.154
##               3rd Qu.:-0.020
##               Max.   : 0.480
```

We see that there are two columns in the data frame: one categorical factor that designates group and one numerical that has the gene expression values (in normalized units).

Re-order factor levels

Notice that our factor levels, in alphabetical order, are "bipolor", "control", and "schizo". When we eventually generate a graph it would be better to follow convention and put the control group first (i.e., all the way to the left in the graph). So let's manually re-order the factor levels:

```
PLP1data$group <- factor(PLP1data$group, levels = c("control",
    "bipolar", "schizo"))
```

Assumptions of ANOVA

An analysis of variance has some underlying assumptions:

- Measurements from each group represent a random sample from its corresponding population
- Variable is normally distributed in each of the k populations
- Variance is the same in all k populations

Assumption of normality in each population

Following our flow chart (Which_Test_three-or-more-samples.pdf), we first check for normality in each group with shapiro.test(). The easiest way to do this is to use tapply() to apply the function to all groups in one command:

```
tapply(PLP1data$PLP1Expression, PLP1data$group, shapiro.test)
## $control
##
##  Shapiro-Wilk normality test
##
## data:  X[[i]]
## W = 0.95131, p-value = 0.5453
##
##
## $bipolar
##
##  Shapiro-Wilk normality test
##
## data:  X[[i]]
## W = 0.84052, p-value = 0.01279
##
##
## $schizo
##
##  Shapiro-Wilk normality test
```

```
##
## data:  X[[i]]
## W = 0.89717, p-value = 0.08617
```

From the output notice that normality is rejected for the bipolar group with a P-value of 0.01.

Apply natural log transformation

ANOVA is robust to departures from normality, and some people might ignore this violation. However, following our flow chart for the course, the next steps to apply a natural log data transformation and re-check for normality. Apply the transformation by establishing a new variable with the function log().

```
PLP1data$PLP1ExpressionLN <- log(PLP1data$PLP1Expression)
## Warning in log(PLP1data$PLP1Expression): NaNs produced
```

Notice that this produces a warning that NaNs are produced. The textbook describes that the expression data are in "normalized units," and if you peak at the data frame you see that the expression data includes negative values. Since you can't calculate the natural log of a negative number these negative expression values are translated to NaN.

```
PLP1data
##      group PLP1Expression PLP1ExpressionLN
## 1  control          -0.02              NaN
## 2  control          -0.27              NaN
## 3  control          -0.11              NaN
## 4  control           0.09       -2.4079456
## 5  control           0.25       -1.3862944
## 6  control          -0.02              NaN
## 7  control           0.48       -0.7339692
## 8  control          -0.24              NaN
## 9  control           0.06       -2.8134107
## 10 control           0.07       -2.6592600
## 11 control          -0.30              NaN
## 12 control          -0.18              NaN
## 13 control           0.04       -3.2188758
## 14 control          -0.16              NaN
## 15 control           0.25       -1.3862944
## 16  schizo          -0.10              NaN
## 17  schizo          -0.31              NaN
## 18  schizo          -0.05              NaN
## 19  schizo           0.11       -2.2072749
## 20  schizo          -0.38              NaN
## 21  schizo           0.23       -1.4696760
## 22  schizo          -0.23              NaN
## 23  schizo          -0.28              NaN
## 24  schizo          -0.36              NaN
## 25  schizo          -0.22              NaN
## 26  schizo          -0.40              NaN
## 27  schizo          -0.19              NaN
## 28  schizo          -0.34              NaN
## 29  schizo          -0.29              NaN
## 30  schizo          -0.12              NaN
## 31 bipolar          -0.34              NaN
## 32 bipolar          -0.39              NaN
## 33 bipolar          -0.22              NaN
```

```
## 34 bipolar          -0.32              NaN
## 35 bipolar          -0.32              NaN
## 36 bipolar          -0.05              NaN
## 37 bipolar          -0.43              NaN
## 38 bipolar          -0.33              NaN
## 39 bipolar          -0.41              NaN
## 40 bipolar          -0.36              NaN
## 41 bipolar          -0.25              NaN
## 42 bipolar          -0.29              NaN
## 43 bipolar           0.06       -2.8134107
## 44 bipolar          -0.30              NaN
## 45 bipolar           0.01       -4.6051702
```

Since we can't do the transformation we are left with the original data and violation of normality in at least one group. Thus, we need to proceed to the nonparametric alternative Kruskal-Wallis test.

Kruskal-Wallis test

This test does not rely on the assumption of normality. The null hypothesis is that all groups have the same distribution of ranks. It is run with the function kruskal.test().

```
kruskal.test(PLP1data$PLP1Expression ~ PLP1data$group)
##
##  Kruskal-Wallis rank sum test
##
## data:  PLP1data$PLP1Expression by PLP1data$group
## Kruskal-Wallis chi-squared = 13.1, df = 2, p-value = 0.00143
```

Note that the P-value is significant (0.00143). This means that the null hypothesis of equal distributions among groups is rejected. However, we do not yet know which group(s) is/are different from others.

Post-hoc Dunn's test

After a significant ANOVA we run the TukeyHSD() to analyze all possible pairwise comparisons among groups while controlling for running multiple tests. Since we did a nonparametric Kruskal-Wallis test we need to apply a nonparametric post-hoc alternative. In this class we will use the Dunn's test. It runs all pairwise comparisons using a method similar to the Mann-Whitney U-test, but controls for running multiple tests by adjusting the P-values. There are multiple correction options available, but we will use one developed by Bonferroni.

First, we need to install a new package (FSA) to get the function to run the Dunn's test. **Run: install.packages("FSA")**

Now import the functions of the package.

```
library(FSA)
```

Now the function dunnTest() is available to us. It has a syntax in which you provide the numerical variable to be analyzed, the factor variable used to define groups, and the multiple test correction method:

```
dunnTest(PLP1data$PLP1Expression, PLP1data$group, method = "bonferroni")
##          Comparison         Z      P.unadj       P.adj
## 1 bipolar - control -3.538960 0.0004017061 0.001205118
## 2  bipolar - schizo -1.112443 0.2659475734 0.797842720
## 3  control - schizo  2.426517 0.0152445334 0.045733600
```

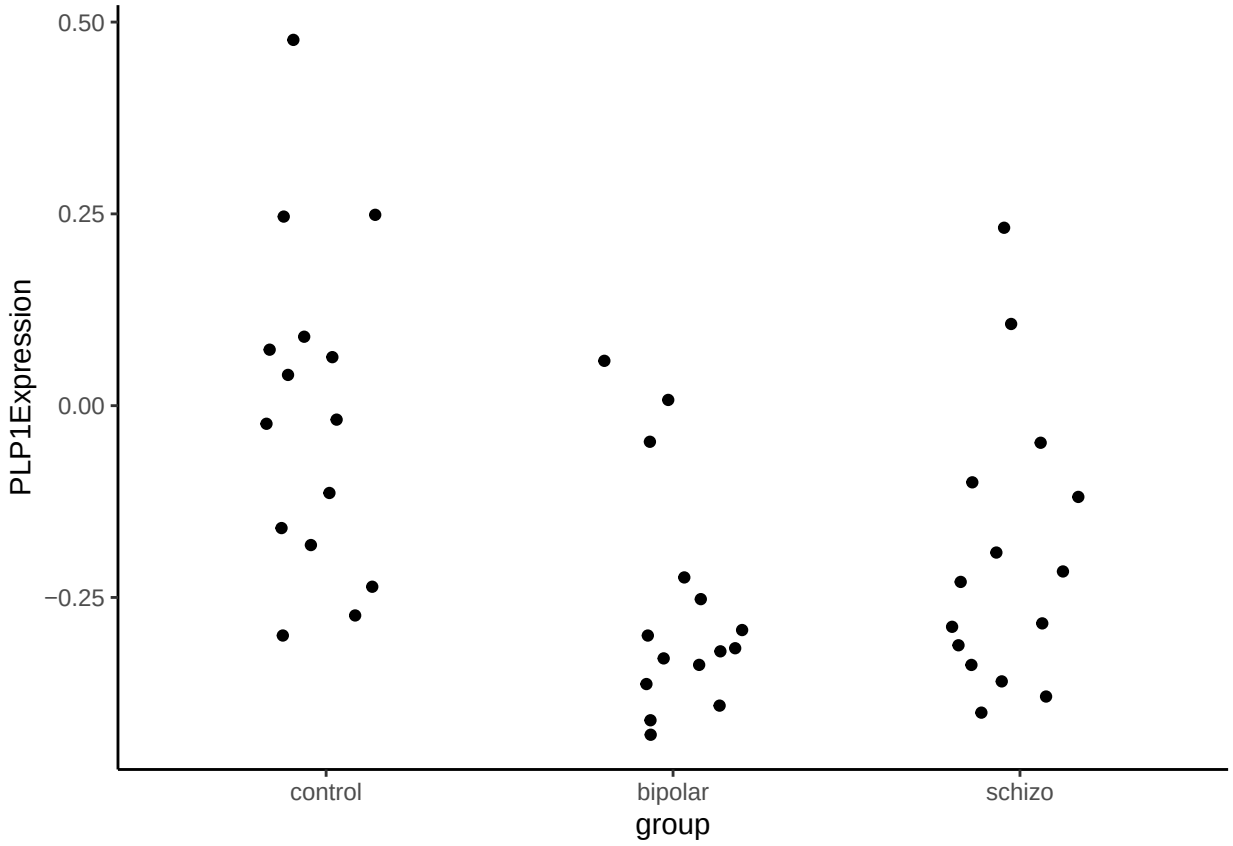
The output table is similar to what is produced with TukeyHSD(). Look carefully at the groups compared in each line, however, because it outputs comparisons in alphabetical order. That is, it starts with comparisons involving "bipolar" even though we changed the order to start with "control".

From the output, we see that the control group is significantly different from the bipolar and schizo groups, and the latter two are not significantly different from each other.

Graphing results

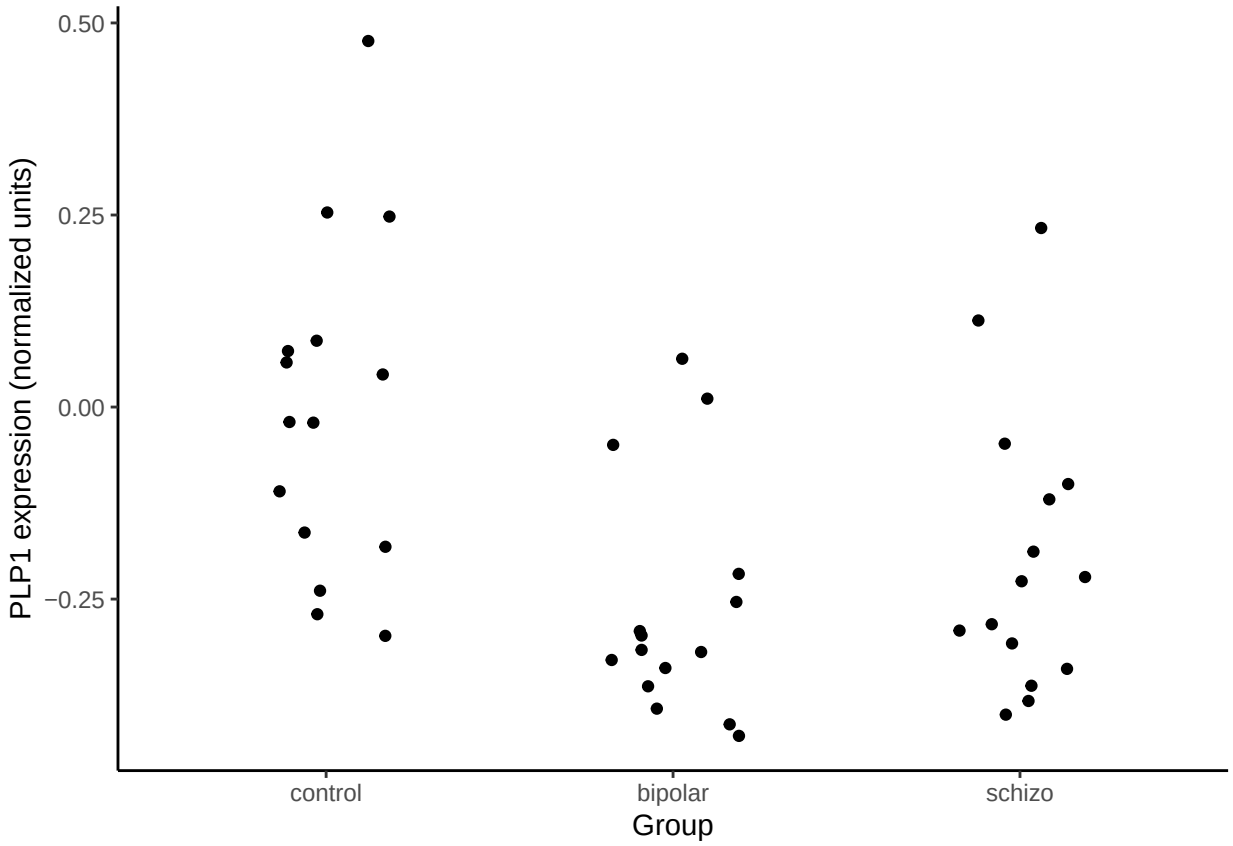
Now let's produce a graph to display the data and results of the statistical tests. We'll start with a simple jitter plot to get a first look at the data:

```
library(ggplot2)
ggplot(PLP1data, aes(x = group, y = PLP1Expression)) + geom_jitter(width = 0.2) +
    theme_classic()
```



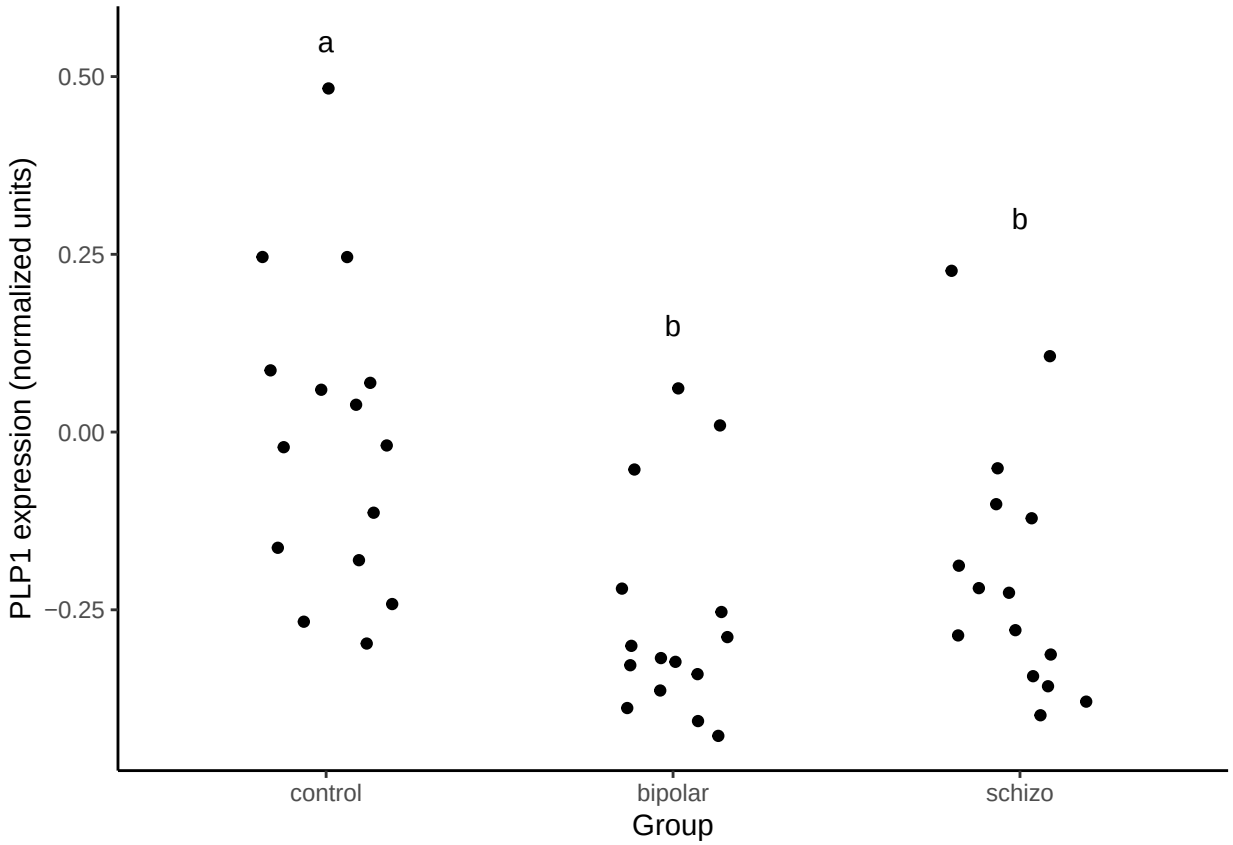
Note that we can improve the axis labels by capitalizing "group" on the x-axis and producing a more readable label with units on the y-axis.

```
ggplot(PLP1data, aes(x = group, y = PLP1Expression)) + geom_jitter(width = 0.2) +
    xlab("Group") + ylab("PLP1 expression (normalized units)") +
    theme_classic()
```



Finally, we'll add letters above groups to specify significantly different groups based on the Dunn's test results. Since the control group is first it's best to assign that group "a". The other two groups are significantly different than the control, but not each other. So they can both be assigned to group "b". I looked at the last plot to estimate a y-coordinate for each group that would be a bit above the maximum value.

```
ggplot(PLP1data, aes(x = group, y = PLP1Expression)) + geom_jitter(width = 0.2) +
    xlab("Group") + ylab("PLP1 expression (normalized units)") +
    annotate(geom = "text", x = 1, y = 0.55, label = "a") + annotate(geom = "text",
    x = 2, y = 0.15, label = "b") + annotate(geom = "text", x = 3,
    y = 0.3, label = "b") + theme_classic()
```



Now we have a nicely formatted graph that displays both the data and the results of our Kruskal-Wallis and Dunn's tests!

R commands summary

- **Kruskal-Wallis test**
  - kruskal.test(y~x)
- **Dunn's test**
  - dunnTest(y~x,method="bonferroni")