Lab 11
Tests for binomial data

Tests for binomial data are as powerful and useful as test for normally distributed data. Undeservedly, they are not as widely used as ANOVA & T-tests. Binomial data is data with two classes: male/female, yes/no, alive/dead, susceptible/resistant, present/absent, pass/fail, etc. You can see that there are many potential applications for these tests.

Sometimes it may be an advantage to convert hopelessly skewed data to binomial data rather than trying to analyze it with non-parametric tests. For example, an ecology dataset with frequencies of plant species on sample plots or experimental plots can easily be converted to presence/absence data.

For this analysis, you first have to build a so-called contingency table. You need to collapse your rows of observations by treatments and report counts of presence and absence. Fortunately, there is a very easy way to do this in R:

Enter the first table below in Excel and import it as a CSV. Then do the following conversion with the contingency table command:

Treatment	Presence
A	Yes
A	No
B	No
B	Yes
B	No

```
dat2=table(dat1)
```

Treatment	Yes	No
A	1	1
B	1	2

11.1 Sample dataset

Here is the example dataset that we covered in class:

Species	Survived	Dead
A	8	12
B	16	4
C	24	16

Import the data as a CSV into R:

```
surv=read.csv("survival.csv")
head(surv)
```

We generate matrices for the analysis that just contain numbers. Therefore, each row represents a different treatment even though we lose the first column. We also generate a second dataset that just contains species A and B:

```
chitab=as.matrix(surv[,2:3])
xtab=as.matrix(surv[1:2,2:3])
```

11.2 Chi-square test

This test applies if you have more than two treatments and is equivalent to an ANOVA. We ask: are there significant differences among treatments (species)? The two tests below are identical:

```
chisq.test(chitab)
prop.test(chitab)
```

You can also apply this test if you have only two treatments, but it is not as powerful as a z-test in detecting significant differences. Below, we do a two-tailed test and a one-tailed test, equivalent to a T-test:

```
prop.test(xtab)
prop.test(xtab, alternative=c("less"))
```

You further carry out tests for one sample. Here we are asking: "is the survival rate of species B significantly larger than 50%?" Note that I entered the survival data as a vector instead of importing it.

```
binom.test(c(16,4), 0.5, alternative=c("greater"))
```

11.3 Z-test for proportions

The two and one-sample tests for proportions should preferably be done with a Z-test rather than a Chi-square test. This test is much more powerful. You have to install the R package "Corpora".

For the single-sample z-test where we asked: is the survival rate of species B (80%) significantly larger than 50%? the R code is as the following:

```
library(corpora)
z.score.pval(16,20,0.5, alternative=c("greater"))
1-z.score.pval(16,20,0.5, alternative=c("greater"))
```

11.4 Fisher's exact test

In case there are many zeros in your data, Fisher's exact test is a better choice (Chi-square test may produce a warning).

```
dat1=data.frame(Group=c("A"   ,"B","C"),
                Yes=c(34,77,30)     ,
                No=c(34,0,35))
dat2

tab=as.matrix(dat1[,2:3])
tab

fisher.test(tab, alternative = 'two.sided') # there is a significant
difference, so we can follow up with pairwise tests
```

For pairwise comparisons, we can subset our dataset to create contrasts:

```
t1 = fisher.test(tab[1:2,], alternative = 'two.sided') # pairwise test

t2 =fisher.test(tab[c(1,3),], alternative = 'two.sided') # pairwise test

t3 = fisher.test(tab[2:3,], alternative = 'two.sided') # pairwise test

#get the p-values
t1$p.value
t2$p.value
t3$p.value
```

11.5 Adjustments for multiple inference

The p-values must be adjusted as per usual.

```
p.adjust(c(t1$p.value, t2$p.value, t3$p.value), method=c("holm"), n=3)
```

You have the choice of several adjustment methods, here in an example for three p-values in an experiment with 3 pairwise comparisons. “Holm’s adjustment”, also called “Sequential Bonferroni adjustment” is a good method:

```
p.adjust(c(0.001,0.032,0.231), method=c("bonferroni"), n=3)
p.adjust(c(0.001,0.032,0.231), method=c("holm"), n=3)
```