Module 7. Nonparameter Hypothesis testing.

Overview:

This module introduces nonparametric hypothesis testing methods. In this module you will be able to conduct nonparametric tests for two sample comparison and 2 by 2 tables. Particularly, many of these tests are permutation tests. In this module you will also learn how to derive and program general permutation tests. Some diagnostic tests for normality and outliers are introduced. As in other modules, you have opportunities to practice as well.
This module introduce nonparametric hypothesis test. We first introduce two-sample nonparametric (Wilcoxon) tests, which corresponds to the two-sample parametric t-tests. The concepts of nonparametric test and exact test are introduced with these examples.

We also introduce permutation tests which are exact nonparametric tests. The Wilcoxon tests are all examples of the permutation tests. We teach how to program these tests in R.

We will also teach nonparametric tests of independence on 2 by 2 tables. Finally, we teach some diagnostic tests to check the parametric assumptions.

By the end of this module, you should know the difference between parametric and nonparametric tests. You should know how to use R to carry out several standard nonparametric tests. And know how to program permutation tests.

Learning Objectives

Nonparametric test

1. Mathematically, distinguish conditions and models between **parametric** and **nonparametric** tests
2. Conduct nonparametric tests, including Wilcoxon rank test, chi-squared test, Fisher's test
3. **Program permutation tests** in R
4. Use the normality test to **check normal assumption** (qq-plot)

Readings:

Seefeld & Linder’s book pages 256-262.
Krijnen’s book pages 59-69.

Lesson 1: Wilcoxon Sign Test

Objectives

By the end of this lesson you will have had opportunity to:

- Conduct the Wilcoxon Sign Test
- Learn the concepts of parametric versus nonparametric tests, asymptotic versus exact test.
- Program a permutation test version of the Wilcoxon Sign Test.

Overview

We have learned about hypothesis tests in the previous module. The main mathematical calculation in statistical hypothesis testing is to quantify the evidence (p-value). We have seen several basic tests including z-test, t-test and binomial test. The p-values of these tests are calculated under a null hypothesis using **parametric** probability models: normal distribution, binomial distribution, etc. The **nonparametric** tests will not need those parametric assumptions, but only that the data is a random sample from a probability distribution.

Wilcoxon Sign Test (paired test)
Recall that we can use the paired t-test to compare μ_X and μ_Y when we have a random sample $X_1, \ldots, X_n$ from $N(\mu_X, \sigma_X)$ and a random sample $Y_1, \ldots, Y_n$ from $N(\mu_Y, \sigma_Y)$. In that case, we test H_0: $\mu_D = \mu_X - \mu_Y = 0$ versus H_A: $\mu_D \neq 0$ with pairwise differences $D_1 = X_1 - Y_1, \ldots, D_n = X_n - Y_n$. The application of the t-test uses the normal assumptions so that the Ds follow the $N(\mu_D = \mu_X - \mu_Y, \sigma_D)$ distribution.

For a nonparametric test, we will *not* make the normal assumption. Then $D_1, \ldots, D_n$ is random sample from a common distribution which may not be normal. The Wilcoxon sign test can be used to test H_0: $m_D = 0$ versus H_A: $m_D \neq 0$ where m_D denotes the population median of the Ds. Notice that $m_D = 0$ is equivalent to $\mu_D = 0$ for normal distribution.

Let W be the number of pairs for which $D_i = X_i - Y_i > 0$. Then the null hypothesis H_0: $m_D = 0$ is equivalent to H_0: $P(X-Y > 0) = 0.5$. Hence, we can use the binomial test for the proportion of pairs with positive differences $(X_i - Y_i > 0)$.

This is called the **Wilcoxon sign test**, since it tests if the proportion of positive signs is 0.5 for the pairwise differences.

Demonstration: The Wilcoxon Sign Test

In this example, we want to show that two genes (CD33 and CCND3 Cyclin D3) express differently, using the Golub data set. To use the sign test, we first compute the pairwise difference of the two genes (in the 808th row and 1042th row). Then apply the binomial test.

```
> data(golub, package='multtest')
> diff<-golub[808,]-golub[1042,]
> binom.test(x=sum(diff>0), n=length(diff), p=0.5, alternative="two.sided")

                Exact binomial test

data:    sum(diff > 0) and length(diff)
number of successes = 4, number of trials = 38, p-value = 6.039e-07
alternative hypothesis: true probability of success is not equal to 0.5
95 percent confidence interval:
  0.02943452 0.24804938
sample estimates:
probability of success
                     0.1052632
```

We can see that, for only 4 out of 38 pairs, the expression of CD33 gene is bigger than the expression of the CCND3 Cyclin D3 gene. This results in a p-value = 6.039e-07. Hence, the null hypothesis is rejected: the two genes do express differently.

Notice that, instead of the binomial test, we can also use the proportion test (z-test) `> prop.test(x=sum(diff>0), n=length(diff), p=0.5, alternative="two.sided")`.

This will result also in a very small p-value = 2.546e-06 and rejection of the null hypothesis.

Parametric Versus Nonparametric Test

A **parametric** test makes assumptions of certain parametric distributions, particularly the normal distribution. For example, we often assume that the data is a random sample $D_1 = X_1 - Y_1, \ldots, D_n = X_n - Y_n$ from the $N(mean = \mu, sd = \sigma)$ distribution. Then we can derive the paired t-test from the fact that, under the null hypothesis of H_0: μ = 0,

$$\frac{\bar{D} - 0}{s / \sqrt{n}} \sim t_{df = n-1}.$$

The ***nonparametric*** test does not make those parametric assumptions. For example, the Wilcoxon sign test only assumed that $D_1, \ldots, D_n$ is a random sample from a common distribution, which has zero median under the null hypothesis.

A common mistake by novice statistics user is in thinking that "nonparametric" means "no assumption". This is not the case—nonparametric tests still need assumptions, particularly the random sample assumption. That is, the observations are independently identically distributed (i.i.d.). It does make less (normality) assumptions, and therefore is safer to use in practice. We call them a **robust test** since they are also valid for non-normal data (thus robust to the normality assumption).

Asymptotic Test Versus Exact Test

Asymptotic test is based on the asymptotic approximated distribution of test statistic. That is, the distribution is approximately true for large sample size. For example, without the normality assumption, the sample mean is asymptotically normal distributed by the Central Limit Theorem. Therefore the z-test and t-tests for means and proportions are asymptotically correct.

The **exact** test is based on exact distribution of the test statistic. For example, for the proportion test of H_0: $p = p_0$ in the previous module, the number of the type in the sample (X) follows a Binomial(size=n, prob=p) distribution. The binomial test for the proportion is an exact test. In contrast, the z-test for proportion is an asymptotic test, since it is based on X following an approximately normal distribution for large sample size n.

Is a t-test a Parametric or Nonparametric Test?

Recall that, in the previous module, we provided two justification for the t-test. First, for a random sample D_1, …, D_n from $N(mean = \mu, sd = \sigma)$ we have the exact distribution $\frac{\bar{D} - \mu}{s/\sqrt{n}} \sim t_{df = n-1}$. Therefore, the t-test *is an exact parametric test*.

The second justification was provided through the Central Limit Theorem without normality assumption. Assuming a random sample D_1, …, D_n from a distribution with $mean = \mu$ and $sd = \sigma$, then for large sample size n, $\frac{\bar{D} - \mu}{s/\sqrt{n}}$ follows $t_{df = n-1}$ distribution approximately. Since there is no parametric assumption in this derivation, the t-test is also *an asymptotic nonparametric test.*

In contrast, the Wilcoxon Sign test is *an exact nonparametric test*, based on the fact that W = number of ($D_i > 0$) follows a binomial distribution.

When to Use a Nonparametric Test

Since a nonparametric test makes less assumptions, we should *always* prefer the nonparametric test to be safe. A parametric test can be used only if we have strong reason to believe in the parametric assumption. The nonparametric test is often more computationally intensive and hard to derive theoretically. Therefore, parametric tests are used when a nonparametric test is not available. It is less common nowadays with advanced nonparametric procedures implemented in statistical software such as R.

Since a t-test is also an asymptotic nonparametric test, it can still be safely applied when sample size n is big. When sample size n is small, the nonparametric exact tests such as the Wilcoxon Sign test are preferred.

Illustration of the Sign Test as a Permutation Test

Example: We illustrate the permutation test on the comparison of CD33 and CCND3 Cyclin D3 in the previous example. For illustration purpose, we will only take the data on the first 3 patients so that the permutation number $2^3 = 8$ is small enough. The three pairwise differences are

```
> data(golub, package='multtest')
> diff<-golub[808, 1:3]-golub[1042,1:3]
> diff
[1] -2.68169 -2.90944 -2.43442
```

To get all permutations, we use the permutations() function in the package “gtools”. Each permutation is represented by three numbers of -1 and 1, with -1 indicating that pair is switched and 1 indicating no switching. The following R script finds all permutations and calculate the test statistic W =number of ($D_i > 0$) on the permutated data.

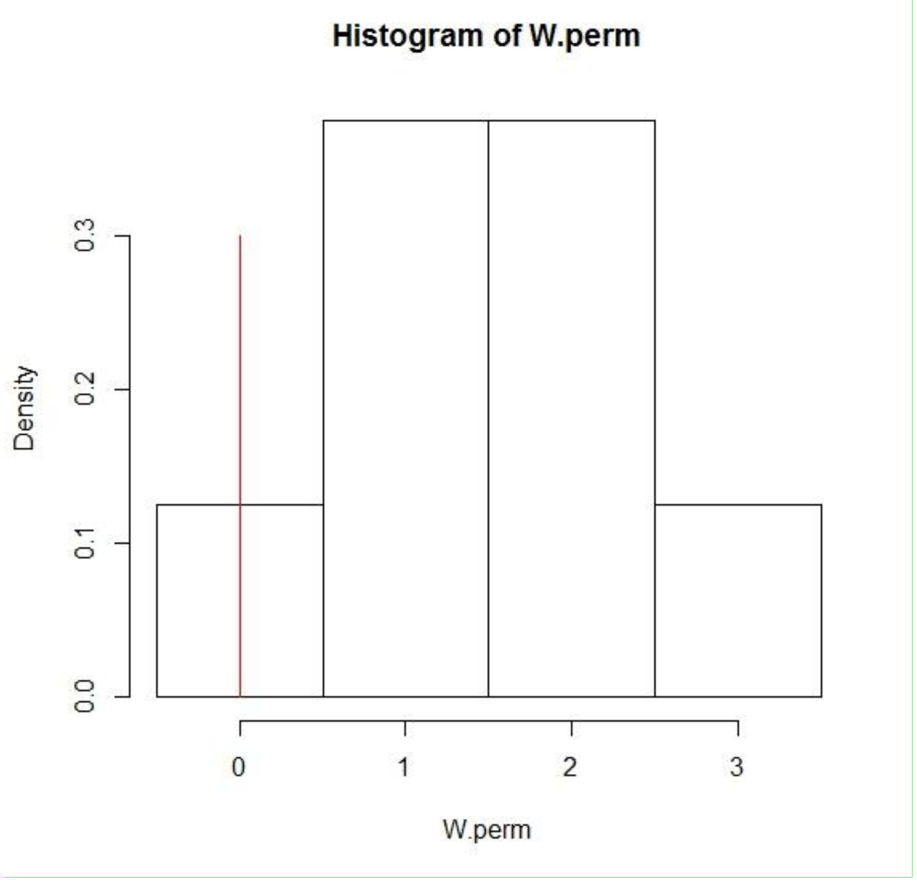
```
install.packages('gtools')
library(gtools)
# List all possible permutations of switching in signs.
signs = permutations(n=2,r=3,v=c(-1,1), repeats.allowed=T)
n.perm=2^3
data.perm=signs*NA
W.perm = rep(NA, n.perm)
for(i in 1:n.perm) {
   data.perm[i,] = signs[i,]*diff              #The permuted pairwise
differences
   W.perm[i] = sum(data.perm[i,]>0)    #The test statistic on permuted
data
}
# Display the permuted data and test statistic
cbind(data.perm, W.perm)
```

Example (Continued)
Continuing with the R script, we can see the all possible permutations and the corresponding test statistic

```
                                                    W.perm
[1,]    2.68169    2.90944    2.43442          3
[2,]    2.68169    2.90944 -2.43442          2
[3,]    2.68169 -2.90944    2.43442          2
[4,]    2.68169 -2.90944 -2.43442          1
[5,] -2.68169    2.90944    2.43442          2
[6,] -2.68169    2.90944 -2.43442          1
[7,] -2.68169 -2.90944    2.43442          1
[8,] -2.68169 -2.90944 -2.43442          0
```

The distribution of test statistic W is derived from the permutations, shown in last column. This is in fact the binomial(n=3, p=0.5) distribution. We can display it using a histogram together with observed test statistic

```
W.obs<-sum(diff>0)
hist(W.perm, freq=F, breaks=(-0.5:3.5))
points(x=W.obs,0.3, col=2,type="h")
```



Example (Continued)
With the distribution of W derived from the permutations, the p-value of the one-sided test can then be calculated as following.

```
> W.obs<-sum(diff>0)
> mean(W.perm<= W.obs)    #p-value
[1] 0.125
```

The p-value = 0.125 is exactly the same from the binomial test which can be checked using the following command.

```
binom.test(x=sum(diff>0), n=length(diff), p=0.5, alternative="less")$p.value
[1] 0.125
```

The above example illustrates the permutation test on a small data set. For large data set, the total number of permutations is large even for the modern computers. In those cases, a *randomized* test using a *subset of possible permutations* suffice. We illustrate the randomized permutation test next.

Randomized Permutation Test

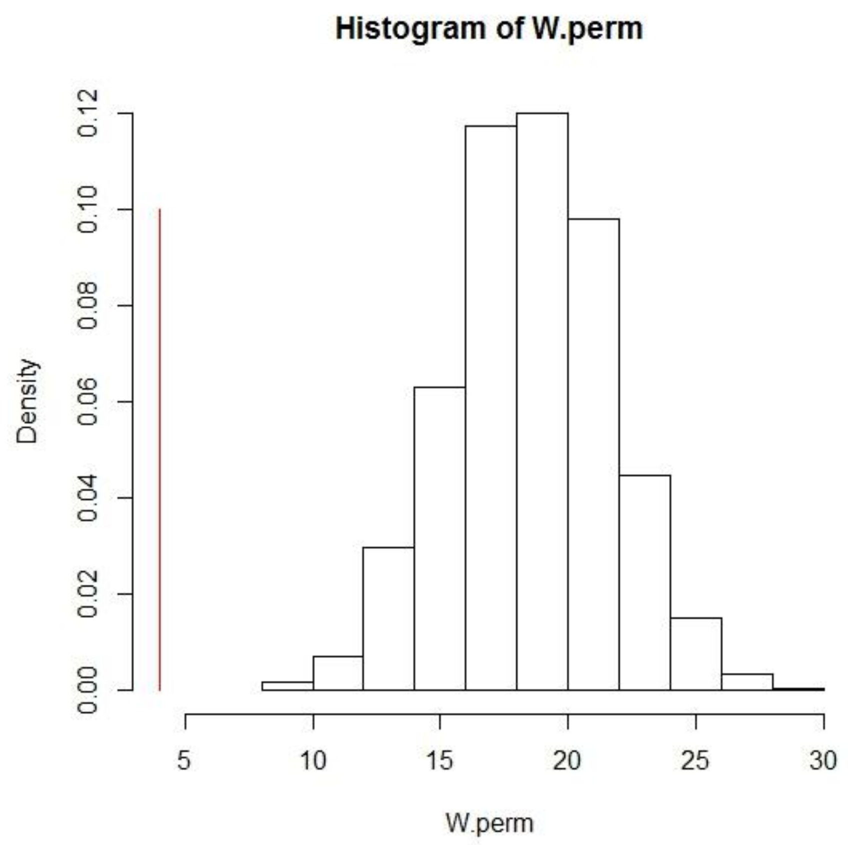
Continuing our example, the pairwise differences between CD33 and CCND3 Cyclin D3 in the Golub data set are as the following.

```
> diff<-golub[808, ]-golub[1042,]
> diff
  [1] -2.68169 -2.90944 -2.43442 -2.75066 -2.00513 -3.21110 -3.43983
-2.34605
  [9] -3.27988 -2.55257 -3.42235 -1.91283 -3.75751 -2.38213 -3.98888
-2.66194
[17] -1.78467 -2.15033 -2.65620 -2.91305 -1.21818 -3.10350 -3.44933
-3.46401
[25] -1.96458 -3.42172 -3.06536 -1.27546 -0.94200   0.27379   0.23235
-0.25035
[33] -1.21663   0.97802   1.50963 -0.24903 -0.63255 -1.52779
```

Thus, there are 2^{38} total number of possible permutations, which is too many to go through even with the advanced computing power. To approximate the distribution of test statistic, we use a random sample of n.perm=2000 permutations. The modified code is

```
n.perm=2000
n=length(diff)
W.perm = rep(NA, n.perm)
for(i in 1:n.perm) {
   data.perm = sample(c(-1,1), n, replace=T)*diff      #The permuted
pairwise differences
   W.perm[i] = sum(data.perm>0)    #The test statistic on permuted data
}
W.obs<-sum(diff>0) #The observed test statistic
hist(W.perm, freq=F, xlim=c(0,30))
points(x=W.obs,0.1, col=2,type="h")
```

If we run the R script, we can see the display of results distribution together with observed test statistic



The observed test statistic is extreme so that the p-value equals zero.

```
> mean(W.perm<= W.obs)    #p-value
[1] 0
```

In contrast, we can conduct the Wilcoxon sign test with R command

```
binom.test(x=sum(diff>0), n=length(diff), p=0.5, alternative="less")
```

This gives a more exact p-value = 3.019e-07. However, for all practical purposes, there is no need to have p-value accurate in many decimal spaces. The randomized permutation sign test gives the same conclusion as the exact Wilcoxon sign test.

Summary of Permutation Tests

The permutation test has two steps:

(1) Create permutated data sets using symmetry under the null hypothesis.
(2) Calculate the test statistic on each permutated data sets.

These permutated test statistic values give the null distribution of the test static. Using this distribution we find the p-value for the observed test statistic.

In step (1), we iterate through all possible permutations if the total number is small. If the total number of permutations is huge, use $m \approx 2000$ random permutation suffices for testing purpose. This results in a *randomized* test.

The permutation tests are one type of Monte Carlo procedures very useful for statistical inference. Most Monte Carlo simulations of previous modules generate random samples from a known theoretical distribution. The permutation tests, on the other hand, resamples (permutations) from the observed data sets. This is a standard method to create nonparametric procedures similar to bootstrap procedures covered in previous module.

Summary

In this lesson, we learned the Wilcoxon sign test which is an exact nonparametric paired test. We clarified the concepts of nonparametric test and exact test, and when should they be used.

We also showed how the Wilcoxon sign test is in fact a permutation test. The permutation test can be used for many other test statistics. Such tests are precise but computationally intensive. This type of tests is becoming popular in bioinformatics applications because of the precision. We next cover some more two-sample Wilcoxon tests that are also permutation tests.

Lesson 2: More Wilcoxon Tests.

Objectives

By the end of this lesson you will have had the opportunity to:

- Select and conduct Wilcoxon signed-ranks test and Wilcoxon rank-sum test
- Program permutation tests in R

Overview

We will now look at additional Wilconxon texts. The Wilcoxon sign test generally has less power than the t-test because the magnitudes of the pairwise differences are ignored; only the signs are used. In this lesson, we teach the rank-based Wilcoxon tests for paired and unpaired data.

Two-sample Paired Wilcoxon Signed-ranks Test

Suppose we have a random sample of the pairwise differences $D_1 = X_1 - Y_1$, …, $D_n = X_n - Y_n$. We wish to test the H_0: P(X-Y > 0)= 0.5 versus H_A: P(X-Y > 0) $\neq$ 0.5. This is equivalent to test H_0: $m_D = 0$ versus H_A: $m_D \neq 0$ where m_D denote the population median of the Ds.

Compared to the previous Wilcoxon sign test, the Wilcoxon signed-ranks test also takes the magnitude of the differences into account by summing over the ranks of the absolute differences. Let $R_i = rank(|D_i|)$. The test statistic is the rank-sum of those positive differences $W = \sum_{i:D_i>0} R_i$. Under the null hypothesis, the expected value of W is $\frac{n(n+1)}{4}$, half the sum of all ranks. The null hypothesis is rejected if the observed test statistic is far from its expected value.

The Wilcoxon signed-ranks test is implemented in R with the wilcox.test(). The distribution of the test statistic $W = \sum_{i:D_i>0} R_i$ is derived using a normal approximate for large sample size n, while the exact distribution is used for small sample size.

Demonstration: The Paired Wilcoxon Signed-ranks Test

In this example, we want to show that two genes (CD33 and CCND3 Cyclin D3) express differently, using the Golub data set. We apply the signed-rank test on these two genes (in the 808th row and 1042th row).

```
> data(golub, package='multtest')
> wilcox.test (x= golub[808,], y= golub[1042,], paired=T,
alternative="two.sided")
                Wilcoxon signed rank test
data:    golub[808, ] and golub[1042, ]
V = 23, p-value = 4.657e-09
alternative hypothesis: true location shift is not equal to 0
```

Since p-value = 4.657e-09 is very small, we reject the null hypothesis. We conclude that the two genes do express differently.

The Wilcoxon signed-rank test can also be calculated as a permutation test. The R code is modified from the previous code for the sign test.

```
> diff<-golub[808, ]-golub[1042,]
> W.obs<-sum(rank(abs(diff))*(diff>0))    #Observed statistic
> n.perm=2000
> n=length(diff)
> W.perm = rep(NA, n.perm)
> for(i in 1:n.perm) {
+       data.perm = sample(c(-1,1), n, replace=T)*diff    #Permuted data
+       W.perm[i] = sum(rank(abs(data.perm))*(data.perm>0)) #Permuted
statistic
+ }
> W.obs #display observed statistic
[1] 23
> 2*min(mean(W.perm<=W.obs), mean(W.perm>=W.obs))    #p-value
[1] 0
```

Two-Sample Unpaired Wilcoxon Rank-sum Test

When we have two random samples $X_1, \ldots, X_n$ and $Y_1, \ldots, Y_m$ of difference sizes n and m, paired tests cannot be used. We wish to test the H_0: P(X-Y > 0) = 0.5 versus H_A: P(X-Y > 0) $\neq$ 0.5.

Under the null hypothesis, $X_1, \ldots, X_n$ and $Y_1, \ldots, Y_m$ all comes from the same distribution. We rank the data $X_1, \ldots, X_n$ and $Y_1, \ldots, Y_m$ together, and the rank numbers of the Xs are summed to form the test statistic W. Under null hypothesis, the expected value of W is $\frac{n(n+m+1)}{2}$. The null hypothesis is rejected if the observed test statistic is far from its expected value.

The Wilcoxon rank-sum test is also implemented in R with the wilcox.test().

Demonstration: The Unpaired Wilcoxon Rank-sum Test

Suppose we want to show that the gene CCND3 Cyclin D3 (1042th row) expresses differently in the ALL and AML groups. We apply the rank-sum test in the following

```
> gol.fac <- factor(golub.cl,levels=0:1, labels= c("ALL","AML"))
> wilcox.test (golub[1042,] ~gol.fac, paired=F, alternative="two.sided")
          Wilcoxon rank sum test
data:   golub[1042, ] by gol.fac
W = 284, p-value = 6.15e-07
alternative hypothesis: true location shift is not equal to 0
```

Since the p-value = 6.15e-07 is very small, we reject the null hypothesis. We conclude that the two genes do express differently.

The Wilcoxon rank-sum test can also be calculated as a permutation test. A permutation in this case randomly selects n data points for the X group from the n+m observations: $X_1, \ldots, X_n$ and $Y_1, \ldots, Y_m$.

```
> gol.fac <- factor(golub.cl,levels=0:1, labels= c("ALL","AML"))
> data<-golub[1042,]
> n<-sum(gol.fac=="ALL")
> m<-sum(gol.fac=="AML")
> W.obs<-abs(sum(rank(data)*(gol.fac=="ALL"))-n*(n+m+1)/2)
#Observed statistic
> n.perm=2000
> W.perm = rep(NA, n.perm)
> for(i in 1:n.perm) {
+       data.perm = sample(data, n+m, replace=F)     #permute data
+       W.perm[i] =
abs(sum(rank(data.perm)*(gol.fac=="ALL"))-n*(n+m+1)/2)    #Permuted
statistic
+ }
> 2*min(mean(W.perm<=W.obs), mean(W.perm>=W.obs))    #p-value
[1] 0
```

R Commands for Wilcoxon Rank Tests

When the data in two groups are saved in vectors x and y, the paired Wilcoxon signed-rank test is done in R by:

```
wilcox.test (x, y, paired=T)
```

The unpaired Wilcoxon rank-sum test is done in R by:

```
wilcox.test (x, y, paired=F)
```

Also, in many data sets the x and y are saved in one vector `z<-c(x,y)` , while a `group` indicator is used to indicate the group members. In those cases, the unpaired Wilcoxon rank-sum test is done by

```
wilcox.test (z ~ group)
```

You can get more details using the help function:

```
help(wilcox.test)
```

Coding Two-sample Permutation Tests in R

The permutation test can be used with any test statistic, not just the rank-sum statistic. To get the distribution of the test statistic, we just calculate the test statistic on each permutation of the data. Hence the codes for permutation tests of different statistics are very similar.

For example, we can also do a permutation test for the statistic $T = |\bar{X} - \bar{Y}|$. We can quickly modify the R code to do this permutation test on the gene CCND3 Cyclin D3 (1042th row) expression values in the ALL and AML groups.

```
gol.fac <- factor(golub.cl,levels=0:1, labels= c("ALL","AML"))
data<-golub[1042,]
n<-length(data)
T.obs<-abs(mean(data[gol.fac=="ALL"])- mean(data[gol.fac=="AML"]))
#Observed statistic
n.perm=2000
T.perm = rep(NA, n.perm)
for(i in 1:n.perm) {
        data.perm = sample(data, n, replace=F)      #permute data
        T.perm[i] = abs(mean(data.perm[gol.fac=="ALL"])-
mean(data.perm[gol.fac=="AML"]))    #Permuted statistic
}
mean(T.perm>=T.obs)    #p-value
```

Notice the only changes are on the lines to calculate the observed statistic and permuted statistic.

This permutation test using $T = |\bar{X} - \bar{Y}|$ is an exact nonparametric test. This is different from the two-sample t-test which is an asymptotic nonparametric test also using $T = |\bar{X} - \bar{Y}|$.

Summary of Permutation Tests

The permutation test has two steps:

(1) Create permutated data sets using symmetry under the null hypothesis.
(2) Calculate the test statistic on each permutated data set.

These permutated test statistic values give the null distribution of the test static. Using this distribution we find the p-value for the observed test statistic.

So far, we have seen two different types of permutations in step (1). For paired data, we permutated within each pair (X_i, Y_i) to keep the paired structure in the data. For unpaired data, we permutated the whole data set $(X_1, ..., X_n, Y_1, ..., Y_m)$.

The type of permutation to use is decided by the model assumption: what Xs and Ys are assumed to be from the same distribution under null hypothesis. Other permutation tests can involve different types of permutations in step (1).

step (2) is the same for *all* permutation tests. Changing the test statistic leads to all kinds of tests: rank-sum test, mean test, median test, variance test, etc.

If the null test statistic distribution is theoretically known (e.g., binomial), then the permutation procedure is unnecessary. However, for advanced statistical tests, the theoretical derivation is often not available. In those cases, the permutation test procedure here is very straightforward and leads to very practical tests.

Lesson Summary

We introduced the Wilcoxon (paired) signed-rank test and the Wilcoxon (unpaired) rank-sum test. Both tests can be conducted by the wilcox.test() in R.

We also illustrated how to calculate these two tests directly from the permutation tests. For real data analysis, the ready-made wilcox.test() should be used for appropriate data sets. You should also be able to program in R the permutation test based on other test statistics.

In next lesson, we introduce nonparametric tests for contingency tables.

Lesson 3: Tests on Contingency Tables: Chi-square Test and Fisher’s Exact Test

Objectives

By the end of this lesson you will have had opportunity to:

- Conduct the chi-square test and Fisher’s exact test for 2 by 2 table

Overview

In this lesson we will teach two nonparametric tests of independence on contingency tables. The chi-square test is an asymptotic nonparametric test. The Fisher’s test is an exact nonparametric test.

Chi-Square Test.

We have learned how to test if the proportion of a certain type A in the population equals p_0. This is formulated as testing H_0: $\pi = p_0$ versus H_A: $\pi \neq p_0$, where π is the proportion belonging to type A. This tests if the probability equals a pre-specified value for one category within two possible categories: A and not A (with probabilities π and $1- \pi$ respectively). More generally, there can be more categories and we want to test if their probabilities equal pre-specified values.

Let $(\pi_1, \ldots, \pi_m)$ denote the probabilities for the m possible categories in the population. Then we test

$$H_0: (\pi_1, \ldots, \pi_m) = (p_1, \ldots, p_m) \text{ versus } H_A: (\pi_1, \ldots, \pi_m) \neq (p_1, \ldots, p_m)$$

where $(p_1, \ldots, p_m)$ are pre-specified values. Notice that $p_1, \ldots, p_m$ should add up to 1.

By multiplying the probabilities with the total number of observations, we obtain the expected number of observations ($e_i = np_i$). Now we can compute the statistic

$$q = \sum_{i=1}^{m} \frac{(o_i - e_i)^2}{e_i}$$

where o_i is the *i*-th observed and $e_i = np_i$ the *i*-th expected frequency. Under the null hypothesis, this statistic is asymptotically a chi-squared (χ^2_{m-1}) distributed with m-1 degrees of freedom. Therefore, the p-value is defined as $P(\chi^2_{m-1} \geq q)$. The null hypothesis is rejected when the p-value is less than the significance level α.

The chi-square test is an asymptotic test.

The usual z-test for a proportion is a special case of the chi-square test with m = 2 categories: A and not A. The equivalence occurs due to the fact that the square of the z-statistic (standard normal distributed) follows a χ^2_1 distribution by definition.

Demonstration

Suppose we want to test the hypothesis that the nucleotides of Zyxin have equal probability. Let the probability of {A, C, T, G} to occur in the sequence be given by $(\pi_1, \pi_2, \pi_3, \pi_4)$. Then the null hypothesis to be tested is H$_0$: $(\pi_1, \pi_2, \pi_3, \pi_4)$ = $(\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4})$. The data for the gene is displayed in Table 2.1 (page 18) of the Applied Statistics for Bioinformatics using R for the sequence "X94991.1".

```
> install.packages(c("ape"),repo="http://cran.r-project.org",dep=TRUE)
> library(ape)
> table(read.GenBank(c("X94991.1"),as.character=TRUE))
  a   c   g   t
410 789 573 394
```

From the table, the total number of nucleotides is n = 2166, so that the expected frequencies $e_i = np_i$ are equal to 2166/4 = 541.5. Then, the q-value equals

$$q = \sum_{i=1}^{m} \frac{(o_i - e_i)^2}{e_i} = \frac{(410-541.5)^2}{541.5} + \frac{(789-541.5)^2}{541.5} + \frac{(573-541.5)^2}{541.5} + \frac{(394-541.5)^2}{541.5} = 187.0674 .$$

Since $P(\chi_3^2 \geq 187.0674)$ is close to zero, the null hypothesis is clearly rejected. The nucleotides of Zyxin do not occur with equal probability.

A more direct manner to perform the test is by using the built-in-function chisq.test(), as follows.

```
> zyxinfreq <- table(read.GenBank(c("X94991.1"), as.character=TRUE))
> chisq.test(zyxinfreq)
            Chi-squared test for given probabilities
data:   zyxinfreq
X-squared = 187.0674, df = 3, p-value < 2.2e-16
```

The p-value < 2.2e-16 is really small, and we reject the null hypothesis that nucleotides of Zyxin occur with equal probability.

Remark: the default option in chisq.test() is to test the null hypothesis of equal probabilities for all categories. In the above example, we will get same result if specify the probabilities explicitly as `chisq.test(zyxinfreq, p=c(1/4,1/4,1/4,1/4))`

Chi-square test for the 2 by 2 Contingency Table

We consider testing independence in a 2 by 2 contingency table. Suppose the first variable has two possible outcomes (A1 and A2), and the second variable has two possible outcomes (B1 and B2). Then there are four categories of possible outcomes in total. The 2 by 2 contingency table presents the count of number of observations in each category o_{ij} in a table.

Observed	B1	B2	Total
A1	o_{11}	o_{12}	R_1
A2	o_{21}	o_{22}	R_2
Total	C_1	C_2	N

The expected value for each category is the product of total observations N and the probability for that category.

Expected	B1	B2	Total
A1	$p_{11}N$	$p_{12}N$	$(p_{11}+p_{12})N$
A2	$p_{21}N$	$p_{22}N$	$(p_{21}+p_{22})N$
Total	$(p_{11}+p_{21})N$	$(p_{21}+p_{22})N$	N

We often wish to test the marginal independence of the contingency table. That is, if the probability for an observation falling into category o_{ij} (i =row, j=column) is just the product of the marginal probabilities for that row and column. The null hypothesis is H$_0$: $p_{11}=(p_{11}+p_{12})(p_{11}+p_{21})$. Since we would estimate the marginal probabilities by $(p_{11}+p_{11})=\frac{R_1}{N}$ and $(p_{11}+p_{21})=\frac{C_1}{N}$, the expected value for the first cell is:

$$e_{11}=p_{11}N=(\frac{R_1}{N})(\frac{C_1}{N})N\,.$$

The Chi-Square test statistic is thus calculated. The degrees of freedom for this test are the number of rows (r) minus 1 times the number of columns (c) minus 1, which is 1 in the case of 2 by 2 table:

$$q=\sum_{i=1}^{r}\sum_{j=1}^{c}\frac{(o_{ij}-e_{ij})^2}{e_{ij}} \quad \text{with} \quad df=(r-1)(c-1)\,.$$

To perform the test in R use the function chisq.test() on the appropriately formatted table.

Demonstration

Consider the CCND3 Cyclin D3 gene in the Golub data set. We wish to see if the proportion of gene expression greater than one is related to the disease groups ALL and AML. We first produce a 2 by 2 table to represent the data.

```
> data(golub, package='multtest')
> gene1<-golub[1042,]
> gol.fac <- factor(golub.cl,levels=0:1, labels= c("ALL","AML"))
> all<- gol.fac=="ALL"
> aml<- gol.fac=="AML"
> testTable<- matrix(c(sum(gene1[all]>1), sum(gene1[all]<=1),
sum(gene1[aml]>1), sum(gene1[aml]<=1)), nrow=2,
dimnames=list("Gene"=c(">1","<=1"), "Disease"=c("ALL","AML")))
> testTable
      Disease
Gene   ALL AML
  >1    26   3
  <=1    1   8
```

To test the null hypothesis of marginal independence, we use the chi-square test.

```
> chisq.test(testTable)
          Pearson's Chi-squared test with Yates' continuity correction
data:   testTable
X-squared = 16.9595, df = 1, p-value = 3.819e-05
Warning message:
In chisq.test(testTable) : Chi-squared approximation may be incorrect
```

Since the *p*-value is smaller than the significance level $\alpha = 0.05$, the null hypothesis of independence is rejected.

Notice the warning message that R gives: Chi-squared approximation may be incorrect. Recall that chi-square test is an asymptotic nonparametric test. In this case, there are small numbers in some cells, and an exact test would be better suited.

Fisher's Exact Test for the 2 by 2 Contingency Table

The Fisher's test for marginal independence is a permutation test. To answer the question "how extreme is this observed table", we can consider all possible permutations of the table of size N that have the same marginal counts R_1, R_2, C_1 and C_2. The statistic to use is the odds ratio $o_{11}o_{22}/(o_{12}o_{21})$. The p-value is calculated as the proportion among the possible permutations that have odds ratios more extreme than the observed table.

The null hypothesis of marginal independence H_0: $p_{11} = (p_{11} + p_{12})(p_{11} + p_{21})$ is equivalent to

$H_0 : \frac{p_{11}p_{22}}{p_{12}p_{21}} = 1$, the population odds ratio equals to one.

The Fisher's exact test is done in R with fisher.test().

Demonstration (continued)

Consider the 2 by 2 table we used earlier in the chi-square test.

```
> testTable
        Disease
Gene    ALL AML
   >1    26   3
   <=1    1   8
```

Apply Fisher’s exact test using R:

```
> fisher.test(testTable)
             Fisher's Exact Test for Count Data
data:   testTable
p-value = 2.767e-05
alternative hypothesis: true odds ratio is not equal to 1
95 percent confidence interval:
      5.170859 3104.396070
sample estimates:
odds ratio
   55.92246
```

We reject the null hypothesis of independence due to the small *p*-value.

In this example, the Fisher test gives a slightly different p-value from the chi-square test. But the conclusions are the same. Generally, we would trust the Fisher’s test more since it is an exact test and here the sample sizes are small, particularly in two cells (1 and 3).

	Disease	
Gene	ALL	AML
>1	26	3
<=1	1	8

Lesson Summary

We have covered the two nonparametric tests for independence on 2 by 2 tables. You should know how to apply these tests with R, and when to use which test.

In next lesson, we consider some diagnostic test to check assumptions for the parametric tests.

Lesson 4: Diagnostic tests: Non-normality Test and Outliers

Objectives

By the end of this lesson you will have had opportunity to:

- Conduct Shapiro-Wilk test for normality
- Conduct outlier test

Overview

The main difference between the parametric and nonparametric tests is the model assumption, particularly the normal distribution assumption. Therefore, diagnostic tests for checking the model assumptions are useful.

Before, we used the q-q plot to visually check the normality of the data. The degree of linearity in a q-q plot can be made into an exact Shapiro-Wilk test for normality. In R, this is done by shapiro.test().

Demonstration
To test the hypothesis that the ALL gene expression values
of CCND3 Cyclin D3 from Golub et al. (1999) are normally distributed, the **Shapiro-Wilk test** can be used as follows.

```
> shapiro.test(golub[1042, gol.fac=="ALL"])
              Shapiro-Wilk normality test
data:    golub[1042, gol.fac == "ALL"]
W = 0.9466, p-value = 0.1774
```

Since the p-value is greater than 0.05, the conclusion is not to reject the null hypothesis that CCND3 Cyclin D3 expression values follow from a normal distribution.

Outliers Test

When gene expression values are not normally distributed, then outliers may appear with large probability. We can decide whether a certain set of gene expression values is contaminated by an outlier or not. Such a hypothesis can be tested by the Grubbs (1950) test. This can be tested by the function grubbs.test() of the outliers package.

Example 7.4.1. From Figure 2.4 (page 21) of the Applied Statistics for Bioinformatics using R we observe that expression values of gene CCND3 Cyclin D3 may contain outliers with respect to the left tail. This can be tested by grubbs.test().

```
> library(outliers)
> grubbs.test(golub[1042, gol.fac=="ALL"])
            Grubbs test for one outlier
data:    golub[1042, gol.fac == "ALL"]
G = 2.9264, U = 0.6580, p-value = 0.0183
alternative hypothesis: lowest value 0.45827 is an outlier
```

Since the p-value is smaller than 0.05, the conclusion is to reject the null hypothesis of no outliers.

In the case the data are normally distributed, the probability of outliers is small. Hence, extreme outliers indicate that the data is likely non-normally distributed. In such case, the Wilcoxon tests are preferred.

Diagnostic Tests and Selection of Nonparametric Tests

When the diagnostic tests find outliers or non-normally distributed data, we prefer the nonparametric Wilcoxon tests over parametric tests.

However, even if the diagnostic test does not reject the normality assumption, it may be due to low power for small sample size. It does not prove the data is indeed normally distributed. Logically, not enough evidence for non-normality is not enough evidence for normality either. Therefore, for small sample size, it is always better to use nonparametric exact tests.

Lesson Summary

This lesson taught two diagnostic tests: Shapiro-Wilk test for testing normality assumption, the Grubbs test for outliers. You should know how to use these tests for checking the model assumptions of parametric tests. A data analyst should always check the model assumptions for the statistical procedures applied. You will encounter more diagnostic tests in later modules.

Module Summary

This module covered several nonparametric hypothesis tests. You should know the difference between parametric tests and nonparametric tests. You should know how to use diagnostic tests to check the parametric assumptions.

You should know how to apply the appropriate nonparametric tests in R using wilcox.test(), chisq.test() and fisher.test().

We have covered the concept of permutation tests. You should be able to program simple permutation tests in R.

Students interested in programming of more advanced permutation tests can study the ‘coin’ package in R by yourself.