486 Class Project

Che Zhang

Ge Qu

Resampling Method in R

What we are going to discuss today is how the resampling method is solved by R code. Using R to generate many randomizations and compute p-value, we use a **for** loop to create many random permutations and corresponding mean differences, store all the mean differences and then compute the p-value.

**The problem is** :

compare two methods in an experiment.

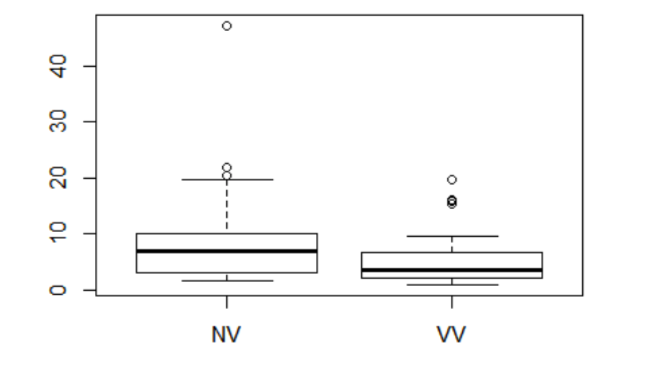
A dataset contains results from an experiment in visual perception using random dot stereograms, such as that shown below. Both images appear to be composed entirely of random dots. However, they are constructed so that a 3D image (of a diamond) will be seen, if the images are viewed with a stereo viewer, causing the separate images to fuse. Another way to fuse the images is to fixate on a point between them and defocus they eyes, but this technique takes some effort and practice. An experiment was performed to determine whether knowledge of the form of the embedded image affected the time required for subjects to fuse the images. One group of subjects (group NV-43 subjects) received either no information or just verbal information about the shape of the embedded object. A second group (group VV-35 subjects) received both verbal information and visual information (e.g., a drawing of the object).

Solution: we use the median difference as the test statistic rather than mean difference. The p- value is estimated based on 10,000 randomly sampled randomizations.

First we read the dataset in R and represent an present an boxplot

fusion=**read.table**("C:/Users/sjricht2/Documents/DataSets/Independent samples T‐test/Fusion\_data.txt", header=TRUE)

**boxplot**(fusion$time~fusion$treatment)



Then we use the R code to compare the treatment group and control group. Treat 1 and Treat 2 are different groups.

**names**(fusion)

## [1] "time" "treatment"

N <‐ **length**(fusion$time)

Treat1 <‐ **subset**(fusion, Select=time, treatment=="NV", drop=T) Treat2 <‐ **subset**(fusion, Select=time, treatment=="VV", drop=T) N1 <‐ **length**(Treat1)

**head**(Treat1,5)

## time treatment

## 1 47.20001 NV

## 2 21.99998 NV

## 3 20.39999 NV

## 4 19.70001 NV

## 5 17.40000 NV

**head**(Treat2,5)

## time treatment

## 44 19.70001 VV

## 45 16.19998 VV

## 46 15.90000 VV

## 47 15.40002 VV

## 48 9.70000 VV

Third, we need to use the median difference between this two groups. So we calculate the median. The **set**. **Seed** function **sets** the starting number used to generate a sequence of random numbers.

observed.fusion <‐ **median**(Treat1$time)‐**median**(Treat2$time) observed.fusion

## [1] 3.3

nperms <‐ 9999

**set.seed**(4132)

result <‐ **numeric**(nperms) for(i in 1:nperms)

{

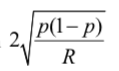
index <‐ **sample**(N, size=N1, replace = FALSE)

result[i] <‐ **median**(fusion$time[index]) ‐ **median**(fusion$time[‐index]) }

(**sum**(result >= observed.fusion)+1)/(nperms + 1) ## [1] 0.3213

**Literature Cited**: [Cleveland, W. S. (1993). *Visualizing Data*. Original source: Frisby, J. P. and Clatworthy, J.L., "Learning to see complex random-dot stereograms," *Perception*, 4, (1975), pp. 173-178]

**Result**: When the sample size is medium to large, enumerating all possible arrangements can be time-consuming at most, and almost impossible at worst. Randomly sample the population of permutations. While enumerating 1 trillion permutations may be computationally time-prohibitive, in many cases enumerating 10,000 or even 100,000 is not.

When estimating an exact *p*-value based on a random sample of *R* permutations, the exact *p*-value would be expected to be within

with 95% confidence. For example, if the true *p*-value is *p*  0.05 , the estimate would be

expected to be within the margins of error below:

|  |  |
| --- | --- |
| R | 95% margin of error |
| 1000 | 0.013784 |
| 5000 | 0.006164 |
| 10000 | 0.004359 |
| 100000 | 0.001378 |

**Our Discussion:**

It uses the difference of mean to locate the test,giving exact p-value regardless of distribution of population. Power will be similar to t-test. It have higher power than tests on means, especially for skewed and heavy-tailed distributions

The WRS test has been studied extensively in relation to the t-test. The t-test tends to have higher power for symmetric distributions, especially for lighter tailed distributions and smaller sample sizes. The WRS test generally has higher power for heavier-tailed distributions and moderate to large sample sizes.

When specific hypotheses are being tested by the permutation test, no population distribution is assumed, and thus it does not make sense to test parameters (e.g., equality of means).

To present resampling techniques, there are four main methods: randomization, tests cross-validation, jackknife, bootstrap.

**Literature Cited**: Introduction to Resampling Methods Using R

<https://www.uncg.edu/mat/qms/Resampling%20Methods%20Using%20R.pdf>

Appendices:

Rationale of supporting resampling

* Empirical : empirical-based resampling do not require assumptions on the sample or the population.
* Clarity: resampling is clean and simple. High mathematical background is not required to comprehend it
* Small sample size: Distributional assumptions required by classical procedures are usually met by a large sample size. Bootstrapping could treat a small sample as the virtual population to "generate" more observations
* Non-random sample: Resampling is valid for any kind of data, including random and non-random data.
* Large sample size: Given a very large sample size, one can reject virtually any null hypothesis → divide the sample into subsets, and then apply a simple or double cross-validation.
* Replications: Repeated experiments in resampling such as cross-validation and bootstrap can be used as internal replications.

Criticisms of resampling

* Assumption: "You’re trying to get something for nothing". Every theory and procedure is built on certain assumptions and requires a leap of faith to some degree. Indeed, the classical statistics framework requires more assumptions than resampling does
* Generalization: resampling is based on one sample and therefore the generalization cannot go beyond that particular sample.
* Bias: confidence intervals obtained by simple bootstrapping are always biased though the bias decreases with sample size. (for normal case the bias in is at least n/(n-1))
* Accuracy: for small sample resampling may be less accurate than conventional parametric methods. Not very convincing argument because today computers are very powerful.
* pros and cons in both traditional and resampling methods carry certain valid points. → the appropriateness of the methodology highly depends on the situation

R packages for resampling methods

* Boot: quite a wide variety of bootstrapping tricks.
* Bootstrap: relatively simple functions for bootstrapping and related techniques.
* coin: permutation tests
* Design: includes bootcov for bootstrapping the covariance of estimated regression parameters and validate for testing the quality of fitted models by cross validation or bootstrapping.
* MChtest: Monte Carlo hypothesis tests: tests using some form of resampling.
* meboot: a method of bootstrapping a time series.
* permtest: a function for permutation tests of microarray data.
* resper: for doing restricted permutations.
* scaleboot: produces approximately unbiased hypothesis tests via bootstrapping.
* simpleboot: performs bootstraps in simple situations: one and two samples, and linear regression.

Work cited: Vivien Rossi, Introduction to resampling methods,CIRAD - UMR Ecologie des forêts de Guyane