Chapter 7

Computer-intensive Tests

This chapter covers two methods of statistical inference in which computing power and random number generation largely substitute for statistical theory: randomization tests and tests based on the bootstrap. These methods allow the creation of customized non-parametric tests without having to produce a new statistical theory each time.

7.1 Permutation Tests and Randomization Tests

7.1.1 Permutation Tests

Randomization tests use the Law of Large Numbers to approximate permutation tests, so we will begin with permutation tests. A **permutation** is an arrangement of a set of objects in some order; so for example, we say there are $5! = 5 \times 4 \times 3 \times 2 \times 1$ permutations of 5 objects. That is, 5 objects may be arranged in 120 different orders.

Permutation tests are most natural in the setting of a true experimental study with random assignment of subjects to treatments, so that all possible assignments are equally likely. The reasoning goes like this. If the treatment is completely ineffective, then the data are what they are, and the only reason that some test statistic might differ between treatments is by chance, because of the random assignment. This is the null hypothesis.

The set of all possible permutations of the data yields the set of all possible assignments to experimental conditions. Under the null hypothesis, these are equally likely. This does *not* mean that all values of the test statistic

are equally likely; not at all! Depending on the particular values of the data, there might be quite a few ties, and the distribution of the test statistic might have an arbitrarily peculiar shape. However, if we had enough time, we could calculate exactly what it is, as follows.

Generate all possible permutations of the data. For each permutation, compute the value of the test statistic. The histogram of the test statistic's values (to be precise, the relative frequency histogram of those values) is called the **permutation distribution** of the test statistic.

If the null hypothesis holds, the test statistic has the permutation distribution. If not, it has some other distribution. Suppose the observed value of the test statistic (that is, the one that we computed from the *unscrambled* data) is far out on the tail of the permutation distribution. Then the data may be deemed unlikely given the hull hypothesis — possibly unlikely enough so that the null hypothesis may be rejected, and we may conclude that the treatment has some effect.

In particular, the proportion of the permutation distribution at or beyond the observed test statistic will be called the **permutation p-value**. As usual,if $p < 0.05$, we'll claim statistical significance.

Don't you think this is more reasonable than doing an experiment with random assignment, and then proceeding to assume a normal distribution in some hypothetical "population" of subjects who *might* have received the various experimental treatments? Fisher (who came up with permutation tests as well as the F-test) thought so. In his classic *Statistical Methods for Research Workers* (1936) he wrote, after describing how to do a permutation test,

> Actually, the statistician does not carry out this very tedious process but his conclusions have no justification beyond the fact they could have been arrived at by this very elementary method.

To summarize, a permutation test is conducted by following these three steps.

1. Compute some test statistic using the set of original observations
2. Re-arrange the observations in all possible orders, computing the test statistic each time.
3. Calculate the permutation test p-value, which is the proportion of test statistic values from the re-arranged data that equal or exceed the value of the test statistic from the original data.

Several comments about permutation tests are in order.

- Please notice that no distribution at all is being assumed for the data. They are what they are, period. In fact, for observational data as well as experimental data, *permutation tests are distribution-free under the null hypothesis*. In this sense, permutation tests are non-parametric.

- For observational studies too, the null hypothesis is that the independent variable(s) and dependent variable(s) are independent.

- It's even better than that. Bell and Doksum (1967) proved that *any* valid distribution test of independence *must* be a permutation test (maybe a permutation test in disguise).

- Some non-parametric methods depend on large sample sizes for their validity. Permutation tests do not. Even for tiny samples, the chance of false significance cannot exceed 0.05.

- It doesn't matter if data are categorical or quantitative. By scrambling the data, any possible relationship between IV and DV is destroyed.

- If either IV or DV is multivariate, scramble *vectors* of data.

- The explanation of permutation tests referred to "the" test statistic, without indicating what that test statistic might be. In fact, the test statistic is up to you. No matter what you choose, the chance of false significance is limited to 0.05.

  What choice is best? It depends on the exact way in which the independent and dependent variables are related. A test statistic that captures the nature of the dependence will yield a more powerful, and hence a better test. So one option is to use your intuition, and make something up. Another option is to look in a book like Good's *Permutation Tests*. There, you'll find good suggestions for a lot of common hypothesis-testing problems. These suggestions are not just based on hunches. They are based on research, in which the statistical researcher has tried to derive a test statistic with maximum power for some class of alternative hypotheses. If you think the null hypothesis might be false in the specified way, such a test statistic will likely perform better than anything you happen to come up with.

Many scientists who use permutation tests just compute something traditional like an F statistic, but compare it to a permutation distribution rather than the F distribution. You usually can't go too far wrong with this approach. It's optimal when the traditional assumptions hold, quite good when they almost hold, and the resulting tests tend to become very powerful for a broad range of alternative hypotheses as the sample size increases.

Another advantage of using traditional test statistics is that everyone has heard of them, and they do not arouse suspicion. If you make up something strange, people may think that you tried more traditional quantities first, and then eventually found a statistic that made the test significant. There's no doubt about it; you *can* fraudulently obtain significance with a permutation test by fishing for a test statistic until you find one that exploits a chance pattern in the data.

- Even with some combinatoric simplification (you can often get away without listing *all* the permutations) and a lot of computing power, permutation tests are not easy to do in practice. Fisher himself considered permutation tests to be entirely hypothetical, but that was before computers.

- One way around the computational problem is to convert the data to ranks, and then do it. Then, permutation distributions can be figured out in advance, by a combination of cleverness and brute force. All the common non-parametric rank tests are permutation tests carried out on ranks. Fisher's exact test is a permutation test for categorical data.

  Often, you'll see Z or chi-square statistics for the rank tests. Since the normal and chi-square distributions are continuous, while permutation distributions are always discrete, you know these have to be large-sample approximations based somehow on the Central Limit Theorem. But aren't permutation tests valid for small samples? Yes! The way it works is that good nonparametric books have tables that give exact critical values for small samples; the Z and chi-square approximations are used once the sample size becomes big enough for the approximations to be valid – and big enough so that the exact permutation distribution (even of the ranks) is hard to compute. But statistical *software* often gives you p-values based on the large-sample approximation, regardless of what the sample size is. This throws away the small-sample virtues

of the tests. If you use rank tests with small samples, it's up to you to find the appropriate table and learn how to use it.

- The modern way around the computational problem is to approximate (that is, estimate) the p-value of a permutation test using the Law of Large Numbers. That's called a randomization test, and it's the topic of the next section.

7.1.2 Randomization Tests

The permutation test p-value is the area under the curve (relative frequency histogram) of the permutation distribution, at or beyond the observed value of the test statistic. When we approximate the p-value of a permutation test by simulation, it's called a **randomization test**. Here's how to do it.

- Place the values of the dependent variable in a random order.
- Compute the test statistic for the randomly shuffled data.

In this way, we have randomly sampled a value of the test statistic from its permutation distribution. Carry out this procedure a large number of times. By the Law of Large Numbers, the the permutation p-value is approximated by the proportion of randomly generated values that exceed or equal the observed value of the test statistic. This proportion is the p-value of the randomization test.

The approximation gets better as the Monte Carlo sample size increases. We'll denote the Monte Carlo sample size by m, the permutation test p-value by p, and the randomization test p-value by $\widehat{p}$.

How big should the Monte Carlo sample size be? Here's one approach. As usual, it's based on a normal approximation to the binomial distribution.

```
############################################################
# Choose Monte Carlo sample size for a randomization     #
# test. Estimate p (p-value of permutation test) with    #
# p-hat. For a given true p (default = 0.04) and         #
# a given alpha (default = 0.05), returns the MC sample  #
# size needed to get p-hat < alpha with probability cc   #
# (default = .99).                                       #
############################################################
randm <- function(p=.04,alpha=0.05,cc=.99)
```

```
    {
    randm <- qnorm(cc)^2 * p*(1-p) / (alpha-p)^2
    randm <- trunc(randm+1) # Round up to next integer
    randm
    } # End of function randm

> probs <- c(.01,.02,.03,.04,.045,.049)
> cbind(probs,randm(p=probs)) # Use default values of alpha and cc
       [,1]   [,2]
[1,]  0.010     34
[2,]  0.020    118
[3,]  0.030    394
[4,]  0.040   2079
[5,]  0.045   9304
[6,]  0.049 252189
```

Student's Sleep Data

This example is simple as well as classical, but its simplicity allows the examination of basic issues. The data are from a paper by William Gossett, who published anonymously under the name "Student," and after whom the *Student's t* distribution is named. The data show the effect of two soporific drugs (increase in hours of sleep) on groups consisting of 10 patients each. The independent variable is `group`, and the dependent variable is `extra` (for extra hours of sleep). The source is Student (1908) The probable error of the mean. *Biometrika*, **6**, 20.

```
credit.erin > cat sleep.dat
   extra group
1    0.7     1
2   -1.6     1
3   -0.2     1
4   -1.2     1
5   -0.1     1
6    3.4     1
7    3.7     1
8    0.8     1
```

```
9    0.0     1
10   2.0     1
11   1.9     2
12   0.8     2
13   1.1     2
14   0.1     2
15  -0.1     2
16   4.4     2
17   5.5     2
18   1.6     2
19   4.6     2
20   3.4     2

credit.erin > R --vanilla < randex1.R > randex1.out
credit.erin > cat randex1.out

R : Copyright 2001, The R Development Core Team
Version 1.4.0  (2001-12-19)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for a HTML browser interface to help.
Type 'q()' to quit R.

> # randex1.R :  First randomization test example, with Student's Sleep Data
> # Monte Carlo sample size m may be set interactively
> set.seed(4444) # Set seed for random number generation
>
> # Define margin of error functions
> merror <- function(phat,M,alpha) # (1-alpha)*100% merror for a proportion
+     {
+     z <- qnorm(1-alpha/2)
```

```
+     merror <- z * sqrt(phat*(1-phat)/M)  # M is (Monte Carlo) sample size
+     merror
+     }
> mmargin <- function(p,cc,alpha)
+            # Choose m to get (1-alpha)*100% margin of error equal to cc
+            {
+            mmargin <- p*(1-p)*qnorm(1-alpha/2)^2/cc^2
+            mmargin <- trunc(mmargin+1) # Round up to next integer
+            mmargin
+            } # End definition of function mmargin
> ###############
> sleepy <- read.table("sleep.dat")
> t.test(extra ~ group, var.equal=TRUE, data = sleepy)

        Two Sample t-test

data:  extra by group
t = -1.8608, df = 18, p-value = 0.07919
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -3.3638740  0.2038740
sample estimates:
mean in group 1 mean in group 2
           0.75            2.33

> t.test(extra ~ group, var.equal=TRUE, data = sleepy)[1]
$statistic
        t
-1.860813

> # It's a list element, not a number
> ObsT <- t.test(extra ~ group, var.equal=TRUE, data = sleepy)[[1]]
> ObsT
        t
-1.860813
>
> # If M is not assigned, it's 1210
> if(length(objects(pattern="M"))==0) M <- 1210
```

```
> cat("Monte Carlo Sample size M = ",M,"\n")
Monte Carlo Sample size M =  1210
> dv <- sleepy$extra ; iv <- sleepy$group
> trand <- numeric(M)
> for(i in 1:M)
+     { trand[i] <- t.test(sample(dv) ~ iv, var.equal=TRUE)[[1]] }
> randp <- length(trand[abs(trand)>=abs(ObsT)])/M
> margin <- merror(randp,M,.01)
>
> cat ("\n")

> cat ("Randomization p-value = ",randp,"\n")
Randomization p-value =  0.08429752
> cat("99% CI from ",(randp-margin)," to ",(randp+margin),"\n")
99% CI from  0.06372398  to  0.1048711
> cat ("\n")

>
> # Now try difference between medians
> cat("\n")

> cat("Median extra sleep for Group = 1: ",median(dv[iv==1]),"\n")
Median extra sleep for Group = 1:  0.35
> cat("Median extra sleep for Group = 2: ",median(dv[iv==2]),"\n")
Median extra sleep for Group = 2:  1.75
> ObsMedDif <- abs(median(dv[iv==1])-median(dv[iv==2]))
> cat("Absolute difference is ",ObsMedDif,"\n")
Absolute difference is  1.4
> cat("\n")

> trand2 <- numeric(M)
> for(i in 1:M)
+     {
+     rdv <- sample(dv)
+     trand2[i] <- abs(median(rdv[iv==1])-median(rdv[iv==2]))
+     }
> randp2 <- length(trand2[abs(trand2)>=abs(ObsMedDif)])/M
> margin <- merror(randp2,M,.01)
```

```
>
> cat ("\n")

> cat ("Randomization p-value for diff bet medians  = ",randp2,"\n")
Randomization p-value for diff bet medians  =  0.2090909
> cat("99% CI from ",(randp2-margin)," to ",(randp2+margin),"\n")
99% CI from  0.1789778  to  0.239204
> cat ("\n")


$
```

The main conclusion here is that the difference between group means is *not* significant. The traditional t-test (in fact, the first published t-test!) and the randomization test both have p-values around 0.08. This is not too surprising. We randomized the t statistic, and the traditional t-test is going to be appropriate for these data.

Then we try another test statistic — the difference between medians. This time we get a p-value near 0.21. This probably reflects lower power of the randomization test when we test medians rather than means on data that are actually normal.

Another thing to notice is that the 99% confidence interval for p does not include 0.05. This means that $\widehat{p}$ is not just less than 0.05, it's *significantly* less than 0.05 (at the 0.01 level). This is good. In fact, maybe it should be obligatory.

If it's really obligatory, then we need some kind of power analysis for choosing m. Letting p denote the true p-value from the permutation test, and letting α denote the significance level (for us, $\alpha = 0.05$ unless we're applying a Bonferroni correction), the traditional statistic for testing whether p is different from α would be

$$Z^* = \frac{\widehat{P} - \alpha}{\sqrt{\frac{\alpha(1-\alpha)}{m}}},$$

which has a standard normal distribution under the null hypothesis. Some medium-grade calculations show that the probability that $\widehat{P}$ will be *significantly* different from α at level L (i.e., the power) with a true p-value of p is

approximately

$$1-Pr\left\{\frac{\sqrt{m}(\alpha-p)}{\sqrt{p(1-p)}}-z_{1-L/2}\sqrt{\frac{\alpha(1-\alpha)}{p(1-p)}}<Z<\frac{\sqrt{m}(\alpha-p)}{\sqrt{p(1-p)}}+z_{1-L/2}\sqrt{\frac{\alpha(1-\alpha)}{p(1-p)}}\right\}$$

where Z has a standard normal distribution, and the approximation is excellent for m larger than a few hundred.

The preceding formula is just for the record, and to provide another opportunity to illustrate how a formula can be transcribed more or less directly into an S function.

```
# Power for detecting p-hat significantly different from alpha at
# significance level L, given true p and MC sample size M.
randmpow <- function(M,alpha=0.05,p=0.04,L=0.01)
    {
    z <- qnorm(1-L/2)
    left <- sqrt(M)*(alpha-p)/sqrt(p*(1-p))
    right <- sqrt( alpha/p * (1-alpha)/(1-p) )
    randmpow <- 1 - pnorm(left+z*right) + pnorm(left-z*right)
    randmpow
    } # End function randmpow
```

The function `findm` uses `randmpow` to search for the Monte Carlo sample size needed for a specified power. Again, the *power* we're talking about here is the power of a test for whether the randomization test p-value $\widehat{P}$ is different from 0.05.

```
findm <- function(wantpow=.8,mstart=1,aa=0.05,pp=0.04,LL=0.01)
    {
    pow <- 0
    mm <- mstart
    while(pow < wantpow)
        {
        mm <- mm+1
        pow <- randmpow(mm,aa,pp,LL)
        } # End while
    findm <- mm
    findm
    } # End function findm
```

Table 7.1.2 shows the result of applying the function `findm` to a selected set of true p values and desired power values.

Table 7.1: Monte Carlo sample size required to have specified probability that $\widehat{P}$ will be significantly different from 0.05 at the 0.01 level, when the true p-value is P

	Probability of Significance				
P	0.70	0.75	0.80	0.85	0.90
0.0001	129	130	131	132	133
0.0010	140	142	144	148	151
0.0050	177	184	191	199	210
0.0100	236	247	261	276	297
0.0200	448	478	513	555	610
0.0300	1,059	1,144	1,243	1,363	1,522
0.0400	4,411	4,811	5,276	5,845	6,602
0.0450	17,962	19,669	21,660	24,103	27,362
0.0550	18,548	20,459	22,697	25,452	29,143
0.0600	4,705	5,207	5,796	6,522	7,496
0.0700	1,209	1,345	1,506	1,705	1,974
0.0800	551	616	693	789	919
0.0900	317	356	403	461	539
0.1000	207	234	265	305	358
0.3000	11	13	15	18	22
0.5000	4	4	5	6	8

The Greenhouse Data Again

With permutation and randomization tests, it's a tricky business to carry out a test a set of independent variables while controlling for another set. It's easy to preserve the relationships among multiple independent variables or multiple dependent variables by keeping them together, but it's hard to preserve the relationship of the dependent variable to one set of independent variables while destroying its relationship to another set by randomization.

There's one very important case where this is *not* a problem. In factorial designs with equal or proportional sample sizes, the independent variables are completely unrelated to each other, so we can just randomize the dependent variable (or collection of dependent variables). Here's an example from the greenhouse data.

```
credit.erin > head green.dat
                              PLANT    MCG    MEANLNG

                           1      1       7      50.714
                           2      1       9      10.793
                           3      3       8     106.514
                           4      3       7     102.243
                           5      3       9      73.214
                           6      1       3      10.471
                           7      2       2      13.536
credit.erin > R

> green <- read.table("green.dat")
> plant <- factor(green$PLANT) ; mcg <- factor(green$MCG)
> meanlng <- green$MEANLNG #$
> obs <- anova(lm(meanlng ~ plant*mcg))
> obs
Analysis of Variance Table

Response: meanlng
          Df Sum Sq Mean Sq  F value    Pr(>F)
plant      2 221695  110848 113.9032 < 2.2e-16 ***
mcg        5  58740   11748  12.0719 5.894e-09 ***
plant:mcg 10  47581    4758   4.8893 1.273e-05 ***
```

```
Residuals 90  87586     973
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
> # This agrees with what we got from SAS

> obsF <- obs[1:3,4]
> obsF
         1          2          3
113.903170  12.071871   4.889303
>
> set.seed(4444)
> M <- 500 ; simf <- NULL
> for(i in 1:M)
+    {
+    simf <- rbind(simf,anova(lm(sample(meanlng)~plant*mcg))[1:3,4])
+    } # Next i (next simulation)

>
> plantp <- length(simf[,1][simf[,1]>=obsF[1]])/M ; plantp
[1] 0
> max(simf[,1])
[1] 7.460185
> min(simf[,1])
[1] 0.0003066219
> mcgp <- length(simf[,2][simf[,2]>=obsF[2]])/M ; mcgp
[1] 0
> intp <- length(simf[,3][simf[,3]>=obsF[3]])/M ; intp
[1] 0
> max(simf[,2])
[1] 4.54209
> max(simf[,3])
[1] 3.209669
```

The randomization p-value is approximately zero. We can't compute a meaningful confidence interval (why not?) but we can conclude that the permutation p-value is less than 0.05, because

```
> .05*sqrt(500)/sqrt(.05*.95)
[1] 5.129892
```