Chapter 11

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.

11.1 Permutation Tests and Randomization Tests

11.1.1 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 explanatory variable(s) and response 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 explanatory variable and response variable is destroyed.

- If either explanatory variable or response variable 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 explanatory and response 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.

11.1.2 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 response 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 explanatory variable is `group`, and the response 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.

Table 11.1: 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

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

The Greenhouse Data Again

With permutation and randomization tests, it's a tricky business to carry out a test for a set of explanatory variables while controlling for another set. It's easy to preserve the relationships among multiple explanatory variables or multiple response variables by keeping them together, but it's hard to preserve the relationship of the response variable to one set of explanatory 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 explanatory variables are completely unrelated to each other, so we can just randomize the response variable (or collection of response 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
```

The Twins Data

Sherlock Holmes and the hat.

Long ago, there was more space in journals, and a journal called *Human Biology* used to publish raw data. The `twin data` contains educational test scores and physical measurements for a sample of high school age identical and fraternal twin pairs. Members of each twin pair were of the same sex. Except for a few cases where the parents were not sure, Twin One was born first and Twin Two was born second. The variables are:

1. SEX: 0=Male, 1=Female
2. IDENT: 0=Fraternal 1=Identical
3. PROGMAT1: Progressive matrices (puzzle) score for twin 1

4. REASON1: Reasoning score for twin 1
5. VERBAL1: Verbal (reading and vocabulary) score for twin 1
6. PROGMAT2: Progressive matrices (puzzle) score for twin 2
7. REASON2: Reasoning score for twin 2
8. VERBAL2: Verbal (reading and vocabulary) score for twin 2
9. HEADLNG1: Head Length of Twin 1
10. HEADBRD1: Head Breadth of Twin 1
11. HEADCIR1: Head Circumference of Twin 1
12. HEADLNG2: Head Length of Twin 2
13. HEADBRD2: Head Breadth of Twin 2
14. HEADCIR2: Head Circumference of Twin 2

This is a subset of the original data. Some variables like height and weight are not included. The reference is Clark, P. J., Vandenberg, S. G., and Proctor, C. H. (1961), "On the relationship of scores on certain psychological tests with a number of anthropometric characters and birth order in twins," *Human Biology*, **33**, 163-180.

We want to see if performance on the educational tests is related to head size.

```
/res/jbrunner/www/442/S > head smalltwin.dat
sex ident progmat1 reason1 verbal1 progmat2 reason2 verbal2 headlng1 headbrd1
headcir1 headlng2 headbrd2 headcir2
    1    1   1  48  53  66  35  42  61   183 140 522 188 138 535
    2    1   1  47  69  88  53  74  84   189 137 542 186 140 543
    3    1   1  35  68  92  42  61  86   185 145 549 186 140 550
    4    1   1  34  42  73  26  38  68   183 151 544 185 147 545
    5    1   1  49  71  95  38  72  97   174 145 534 186 143 543
    6    1   1  50  90 122  46  82 101   191 143 551 191 141 552
    7    1   1  25  30  42  28  37  43   184 143 511 186 143 535
    8    1   1  25  74  64  41  78  65   180 146 532 179 144 527
    9    1   1  23  19  52  23  36  59   193 146 560 191 145 551

/res/jbrunner/www/442/S > R

> twinframe <-  read.table("smalltwin.dat")
> sex <- twinframe$sex ; ident <- twinframe$ident
> sexfac <- factor(twinframe$sex,levels=c(0,1),label=c("Male","Female"))
> identfac <- factor(twinframe$ident,levels=c(0,1),
+                  label=c("Fraternal","Identical"))
> table(sexfac,identfac)
        identfac
sexfac   Fraternal Identical
  Male          13        21
  Female        20        20
> mental <- twinframe[,3:8]  # All rows, cols 3 to 8
> phys   <- twinframe[,9:14] # All rows, cols 9 to 14
> cor(mental,phys)
          headlng1   headbrd1  headcir1  headlng2   headbrd2  headcir2
progmat1 0.1945786 0.02669260 0.2046808 0.2070390 0.09577333 0.2204541
reason1  0.1232977 0.03186775 0.2052615 0.0978289 0.04733736 0.1955942
verbal1  0.2259473 0.05372263 0.2452086 0.2132409 0.07487114 0.2333709
progmat2 0.2863199 0.19917360 0.3128950 0.3446627 0.22308623 0.3739253
reason2  0.2127977 0.06950846 0.2767257 0.1226885 0.11543427 0.2521013
verbal2  0.2933130 0.16693928 0.3242051 0.2537764 0.22801336 0.3350497

>
> # But that’s IGNORING sex and ident-frat. Want to CONTROL for them.
> n <- length(sex)
> mf <- (1:n)[sex==0&ident==0] # mf are indices of male fraternal pairs
> mi <- (1:n)[sex==0&ident==1] # mi are indices of male identical pairs
> ff <- (1:n)[sex==1&ident==0] # ff are indices of female fraternal pairs
> fi <- (1:n)[sex==1&ident==1] # fi are indices of female identical pairs

> mf
 [1] 62 63 64 65 66 67 68 69 70 71 72 73 74

> # Sub-sample sizes
> nmf <- length(mf)  ; nmi <- length(mi)
> nff  <- length(ff) ; nfi <- length(fi)
> nmf ; nmi ; nff ; nfi
[1] 13
[1] 21
[1] 20
[1] 20
> table(sexfac,identfac)
        identfac
sexfac   Fraternal Identical
  Male          13        21
  Female        20        20

> # mentalmf are mental scores of male fraternal pairs, etc.
> mentalmf <- mental[mf,] ; physmf <- phys[mf,]
> mentalmi <- mental[mi,] ; physmi <- phys[mi,]
> mentalff <- mental[ff,] ; physff <- phys[ff,]
> mentalfi <- mental[fi,] ; physfi <- phys[fi,]

> mentalmf
   progmat1 reason1 verbal1 progmat2 reason2 verbal2
62       58      91     128       54      73     129
63       44      46      79       42      34      42
64       44      43      70       43      36      58
65       36      40      63       42      39      63
66       34      21      53       45      31      70
67       50      70      93       45      67     109
68       50      81     101       41      47      96
69       31      76     122       43      70      75
70       23      29      62       26      29      42
71       52      66     114       42      69     120
72       48      51      62       30      35      49
73       23      48      78       38      62      87
74       28      38      62       55      70     105

> # First three rows
> mentalmf[1:3,]
   progmat1 reason1 verbal1 progmat2 reason2 verbal2
62       58      91     128       54      73     129
63       44      46      79       42      34      42
64       44      43      70       43      36      58
> # Last 3 columns
> mentalmf[,4:6]
   progmat2 reason2 verbal2
62       54      73     129
63       42      34      42
64       43      36      58
65       42      39      63
66       45      31      70
67       45      67     109
68       41      47      96
69       43      70      75
70       26      29      42
71       42      69     120
72       30      35      49
73       38      62      87
74       55      70     105
> # Rows in random order
> mentalmf[sample(1:13),]
```

```
   progmat1 reason1 verbal1 progmat2 reason2 verbal2
71       52      66     114       42      69     120
73       23      48      78       38      62      87
66       34      21      53       45      31      70
69       31      76     122       43      70      75
65       36      40      63       42      39      63
68       50      81     101       41      47      96
64       44      43      70       43      36      58
70       23      29      62       26      29      42
62       58      91     128       54      73     129
63       44      46      79       42      34      42
74       28      38      62       55      70     105
67       50      70      93       45      67     109
72       48      51      62       30      35      49
>
```

That's how we'll randomize. Back to CONTROLLING for sex, ident.

```
> # mentalmf are mental scores of male fraternal pairs, etc.
> mentalmf <- mental[mf,] ; physmf <- phys[mf,]
> mentalmi <- mental[mi,] ; physmi <- phys[mi,]
> mentalff <- mental[ff,] ; physff <- phys[ff,]
> mentalfi <- mental[fi,] ; physfi <- phys[fi,]
>
> cor(mentalmf,physmf)
          headlng1    headbrd1   headcir1   headlng2   headbrd2   headcir2
progmat1 0.3534186 -0.53715165 0.05247501 -0.1486551 -0.3335911 -0.2541279
reason1  0.4784903 -0.04435345 0.40868525  0.2009069 -0.1853897  0.1574282
verbal1  0.3333061  0.02578888 0.36744645  0.1507982 -0.1958353  0.1267843
progmat2 0.5712273 -0.16389337 0.37080025  0.5622139 -0.1996214  0.4073323
reason2  0.4886337  0.38731941 0.63957418  0.4271557  0.2587126  0.6682264
verbal2  0.5278153  0.25599312 0.62836834  0.3403694  0.1966882  0.6113976
>
> # Don't want to correlate mental twin 1 with phys twin 2
>
> cor(mentalmf[,1:3],physmf[,1:3])
          headlng1    headbrd1   headcir1
progmat1 0.3534186 -0.53715165 0.05247501
reason1  0.4784903 -0.04435345 0.40868525
verbal1  0.3333061  0.02578888 0.36744645
> max(abs(cor(mentalmf[,1:3],physmf[,1:3])))
[1] 0.5371517
>
```

```
> cor(mentalmf[,4:6],physmf[,4:6])
          headlng2   headbrd2  headcir2
progmat2 0.5622139 -0.1996214 0.4073323
reason2  0.4271557  0.2587126 0.6682264
verbal2  0.3403694  0.1966882 0.6113976
> max(abs(cor(mentalmf[,4:6],physmf[,4:6])))
[1] 0.6682264
>
>
> cor(mentalmi[,1:3],physmi[,1:3])
          headlng1   headbrd1  headcir1
progmat1 0.2334577 0.26536909 0.3193472
reason1  0.2622690 0.37549903 0.3534622
verbal1  0.4436284 0.06643773 0.3480645
> max(abs(cor(mentalmi[,1:3],physmi[,1:3])))
[1] 0.4436284
> cor(mentalmi[,4:6],physmi[,4:6])
          headlng2  headbrd2  headcir2
progmat2 0.3645763 0.2537397 0.3699872
reason2  0.1682737 0.4212712 0.3873012
verbal2  0.1814358 0.1590209 0.2112241
> max(abs(cor(mentalmi[,4:6],physmi[,4:6])))
[1] 0.4212712
>
> cor(mentalff[,1:3],physff[,1:3])
            headlng1  headbrd1  headcir1
progmat1 -0.09894825 0.1031112 0.1024857
reason1   0.10353527 0.1974691 0.2299249
verbal1   0.04068947 0.1458637 0.0710240
> max(abs(cor(mentalff[,1:3],physff[,1:3])))
[1] 0.2299249
> cor(mentalff[,4:6],physff[,4:6])
            headlng2  headbrd2  headcir2
progmat2 -0.05058245 0.3809976 0.1205803
reason2   0.19569669 0.3570053 0.2617820
verbal2   0.24212501 0.3964967 0.2463883
> max(abs(cor(mentalff[,4:6],physff[,4:6])))
[1] 0.3964967
>
> cor(mentalfi[,1:3],physfi[,1:3])
            headlng1    headbrd1     headcir1
progmat1 -0.01443227 -0.34580801 -0.004887716
reason1   0.15174745  0.04052029  0.304039946
```

```
verbal1   0.22504203 -0.01581501  0.341174647
> max(abs(cor(mentalfi[,1:3],physfi[,1:3])))
[1] 0.345808
> cor(mentalfi[,4:6],physfi[,4:6])
          headlng2    headbrd2  headcir2
progmat2 0.4030654 -0.02036423 0.4244152
reason2  0.3233766  0.05661767 0.4178053
verbal2  0.2702130  0.15930201 0.4025376
> max(abs(cor(mentalfi[,4:6],physfi[,4:6])))
[1] 0.4244152
>
> # test sta will be absobs = 0.6682264
> obsmax <- max( c(
+                  cor(mentalmf[,1:3],physmf[,1:3]),
+                  cor(mentalmf[,4:6],physmf[,4:6]),
+                  cor(mentalmi[,1:3],physmi[,1:3]),
+                  cor(mentalmi[,4:6],physmi[,4:6]),
+                  cor(mentalff[,1:3],physff[,1:3]),
+                  cor(mentalff[,4:6],physff[,4:6]),
+                  cor(mentalfi[,1:3],physfi[,1:3]),
+                  cor(mentalfi[,4:6],physfi[,4:6])    )   )
>
> obsmax
[1] 0.6682264
>
> obsmin <- min( c(
+                  cor(mentalmf[,1:3],physmf[,1:3]),
+                  cor(mentalmf[,4:6],physmf[,4:6]),
+                  cor(mentalmi[,1:3],physmi[,1:3]),
+                  cor(mentalmi[,4:6],physmi[,4:6]),
+                  cor(mentalff[,1:3],physff[,1:3]),
+                  cor(mentalff[,4:6],physff[,4:6]),
+                  cor(mentalfi[,1:3],physfi[,1:3]),
+                  cor(mentalfi[,4:6],physfi[,4:6])    )   )
> obsmin
[1] -0.5371517
>
> absobs <- max(abs(obsmax),abs(obsmin)) # Test Statistic
> absobs
[1] 0.6682264
>
> ####
> # Here’s how we’ll sample. Recall mentalmf <- mental[mf,]
> ####
> mf
 [1] 62 63 64 65 66 67 68 69 70 71 72 73 74
> mentalmf
   progmat1 reason1 verbal1 progmat2 reason2 verbal2
62       58      91     128       54      73     129
63       44      46      79       42      34      42
64       44      43      70       43      36      58
65       36      40      63       42      39      63
66       34      21      53       45      31      70
67       50      70      93       45      67     109
68       50      81     101       41      47      96
69       31      76     122       43      70      75
70       23      29      62       26      29      42
71       52      66     114       42      69     120
72       48      51      62       30      35      49
73       23      48      78       38      62      87
74       28      38      62       55      70     105
> mental[sample(mf),]
   progmat1 reason1 verbal1 progmat2 reason2 verbal2
72       48      51      62       30      35      49
66       34      21      53       45      31      70
62       58      91     128       54      73     129
69       31      76     122       43      70      75
70       23      29      62       26      29      42
71       52      66     114       42      69     120
67       50      70      93       45      67     109
74       28      38      62       55      70     105
63       44      46      79       42      34      42
68       50      81     101       41      47      96
73       23      48      78       38      62      87
65       36      40      63       42      39      63
64       44      43      70       43      36      58
>
> rmentalmf <- mental[sample(mf),]
> rmentalmi <- mental[sample(mi),]
> rmentalff <- mental[sample(ff),]
> rmentalfi <- mental[sample(fi),]
>
> rcorrs <- c(
+      cor(rmentalmf[,1:3],physmf[,1:3]),
+      cor(rmentalmf[,4:6],physmf[,4:6]),
+      cor(rmentalmi[,1:3],physmi[,1:3]),
+      cor(rmentalmi[,4:6],physmi[,4:6]),
+      cor(rmentalff[,1:3],physff[,1:3]),
+             cor(rmentalff[,4:6],physff[,4:6]),
+             cor(rmentalfi[,1:3],physff[,1:3]),
+             cor(rmentalfi[,4:6],physff[,4:6])  )
>
> min(rcorrs) ; max(rcorrs)
[1] -0.5673855
[1] 0.5166834
> rmin <- NULL ; rmax <- NULL ; rabs <- NULL
>
> # Now simulate
> M <- 200 ; set.seed(4444)
> for(i in 1:M)
+     {
+     rmentalmf <- mental[sample(mf),]
+     rmentalmi <- mental[sample(mi),]
+     rmentalff <- mental[sample(ff),]
+     rmentalfi <- mental[sample(fi),]
+     rcorrs <- c(
+             cor(rmentalmf[,1:3],physmf[,1:3]),
+             cor(rmentalmf[,4:6],physmf[,4:6]),
+             cor(rmentalmi[,1:3],physmi[,1:3]),
+             cor(rmentalmi[,4:6],physmi[,4:6]),
+             cor(rmentalff[,1:3],physff[,1:3]),
+             cor(rmentalff[,4:6],physff[,4:6]),
+             cor(rmentalfi[,1:3],physff[,1:3]),
+             cor(rmentalfi[,4:6],physff[,4:6])  )
+     rmin <- c(rmin,min(rcorrs))
+     rmax <- c(rmax,max(rcorrs))
+     rabs <- c(rabs,max(abs(min(rcorrs)),abs(max(rcorrs))))
+     }
> cbind(rmin,rmax,rabs)[1:20,] # First 20 rows
            rmin      rmax      rabs
 [1,] -0.6521097 0.6024060 0.6521097
 [2,] -0.4410713 0.6091124 0.6091124
 [3,] -0.5635999 0.3953340 0.5635999
 [4,] -0.6655059 0.6937127 0.6937127
 [5,] -0.5110777 0.3692450 0.5110777
 [6,] -0.4513148 0.7600707 0.7600707
 [7,] -0.3180858 0.5724620 0.5724620
 [8,] -0.6258317 0.4013421 0.6258317
 [9,] -0.4061387 0.5174977 0.5174977
[10,] -0.5004209 0.4688702 0.5004209
[11,] -0.6437074 0.3458846 0.6437074
[12,] -0.4065318 0.2945435 0.4065318
```

```
[13,] -0.6115288 0.5631299 0.6115288
[14,] -0.4709578 0.5452405 0.5452405
[15,] -0.6060098 0.6110585 0.6110585
[16,] -0.4220454 0.3177893 0.4220454
[17,] -0.3407132 0.5021933 0.5021933
[18,] -0.5861414 0.3645763 0.5861414
[19,] -0.6137978 0.4693924 0.6137978
[20,] -0.4509271 0.4157352 0.4509271
>
> length(rabs[rabs>=absobs])/M # Two sided
[1] 0.135
> length(rmin[rmin<=obsmin])/M # Lower tailed
[1] 0.395
> length(rmax[rmax>=obsmax])/M # Upper tailed
[1] 0.07
```

Now let's put the wole thing together. Make a file that just does the analysis and prints the results. How many simulations should we use? I'd like to make sure that $\widehat{P}$ is significantly different from 0.07, so I run

```
> 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
>
> findm(pp=.07)
[1] 1506
```

and choose $m = 1600$. First I'll show you the output, then a listing of the program `twins.R`.

```
> source("twins.R")
Male Fraternal
   Twin 1
          headlng1    headbrd1   headcir1
progmat1 0.3534186 -0.53715165 0.05247501
```

```
reason1  0.4784903 -0.04435345 0.40868525
verbal1  0.3333061  0.02578888 0.36744645

   Twin 2
          headlng2   headbrd2  headcir2
progmat2 0.5622139 -0.1996214 0.4073323
reason2  0.4271557  0.2587126 0.6682264
verbal2  0.3403694  0.1966882 0.6113976


Male Identical
   Twin 1
          headlng1   headbrd1  headcir1
progmat1 0.2334577 0.26536909 0.3193472
reason1  0.2622690 0.37549903 0.3534622
verbal1  0.4436284 0.06643773 0.3480645
   Twin 2
          headlng2  headbrd2  headcir2
progmat2 0.3645763 0.2537397 0.3699872
reason2  0.1682737 0.4212712 0.3873012
verbal2  0.1814358 0.1590209 0.2112241

Female Fraternal
   Twin 1
            headlng1  headbrd1  headcir1
progmat1 -0.09894825 0.1031112 0.1024857
reason1   0.10353527 0.1974691 0.2299249
verbal1   0.04068947 0.1458637 0.0710240
   Twin 2
            headlng2  headbrd2  headcir2
progmat2 -0.05058245 0.3809976 0.1205803
reason2   0.19569669 0.3570053 0.2617820
verbal2   0.24212501 0.3964967 0.2463883

Female Identical
   Twin 1
            headlng1    headbrd1     headcir1
progmat1 -0.01443227 -0.34580801 -0.004887716
reason1   0.15174745  0.04052029  0.304039946
verbal1   0.22504203 -0.01581501  0.341174647
   Twin 2
          headlng2    headbrd2  headcir2
progmat2 0.4030654 -0.02036423 0.4244152
reason2  0.3233766  0.05661767 0.4178053
```

```
verbal2  0.2702130  0.15930201 0.4025376


Correlations Between Mental and Physical


     Minimum Observed Correlation:  -0.5371517
     Randomization p-value (one-sided): p-hat =  0.416875
     Plus or minus 99% Margin of error =  0.03174979

     Maximum Observed Correlation:  0.6682264
     Randomization p-value (one-sided): p-hat =  0.10625
     Plus or minus 99% Margin of error =  0.01984402

     Maximum Observed Absolute Correlation:  0.6682264
     Randomization p-value (two-sided): p-hat =  0.199375
     Plus or minus 99% Margin of error =  0.02572806
```

And here is a listing of the program.

```
# twins.R
# Just do the analysis - no examples or explanation with source("twins.R")
twinframe <-  read.table("smalltwin.dat")
sex <- twinframe$sex ; ident <- twinframe$ident
mental <- twinframe[,3:8]  # All rows, cols 3 to 8
phys   <- twinframe[,9:14] # All rows, cols 9 to 14
n <- length(sex)
mf <- (1:n)[sex==0&ident==0] # mf are indices of male fraternal pairs
mi <- (1:n)[sex==0&ident==1] # mi are indices of male identical pairs
ff <- (1:n)[sex==1&ident==0] # ff are indices of female fraternal pairs
fi <- (1:n)[sex==1&ident==1] # fi are indices of female identical pairs
# Sub-sample sizes
nmf <- length(mf)  ; nmi <- length(mi)
nff  <- length(ff) ; nfi <- length(fi)
# mentalmf are mental scores of male fraternal pairs, etc.
mentalmf <- mental[mf,] ; physmf <- phys[mf,]
mentalmi <- mental[mi,] ; physmi <- phys[mi,]
mentalff <- mental[ff,] ; physff <- phys[ff,]
mentalfi <- mental[fi,] ; physfi <- phys[fi,]

cat("Male Fraternal \n")
cat("   Twin 1   \n")
print(cor(mentalmf[,1:3],physmf[,1:3]))
cat("   Twin 2   \n")
print(cor(mentalmf[,4:6],physmf[,4:6]))
```

```
cat(" \n")

cat("Male Identical \n")
cat("   Twin 1   \n")
print(cor(mentalmi[,1:3],physmi[,1:3]))
cat("   Twin 2   \n")
print(cor(mentalmi[,4:6],physmi[,4:6]))
cat(" \n")

cat("Female Fraternal \n")
cat("   Twin 1   \n")
print(cor(mentalff[,1:3],physff[,1:3]))
cat("   Twin 2   \n")
print(cor(mentalff[,4:6],physff[,4:6]))
cat(" \n")

cat("Female Identical \n")
cat("   Twin 1   \n")
print(cor(mentalfi[,1:3],physfi[,1:3]))
cat("   Twin 2   \n")
print(cor(mentalfi[,4:6],physfi[,4:6]))
cat(" \n")

# test sta will be absobs = 0.6682264
# Keep track of minimum (neg corr: obsmin =  -0.5371517) and max too.


obsmax <- max( c(
                cor(mentalmf[,1:3],physmf[,1:3]),
                cor(mentalmf[,4:6],physmf[,4:6]),
                cor(mentalmi[,1:3],physmi[,1:3]),
                cor(mentalmi[,4:6],physmi[,4:6]),
                cor(mentalff[,1:3],physff[,1:3]),
                cor(mentalff[,4:6],physff[,4:6]),
                cor(mentalfi[,1:3],physfi[,1:3]),
                cor(mentalfi[,4:6],physfi[,4:6])    )   )
obsmin <- min( c(
                cor(mentalmf[,1:3],physmf[,1:3]),
                cor(mentalmf[,4:6],physmf[,4:6]),
                cor(mentalmi[,1:3],physmi[,1:3]),
                cor(mentalmi[,4:6],physmi[,4:6]),
                cor(mentalff[,1:3],physff[,1:3]),
                cor(mentalff[,4:6],physff[,4:6]),
                cor(mentalfi[,1:3],physfi[,1:3]),
                cor(mentalfi[,4:6],physfi[,4:6])    )   )

absobs <- max(abs(obsmax),abs(obsmin)) # Test Statistic

rmin <- NULL ; rmax <- NULL ; rabs <- NULL
# Now simulate. Want p-hat sig diff from 0.07. Use findm(pp=.07), get
# 1506, so use m=1600

M <- 1600 ; set.seed(4444)
for(i in 1:M)
    {
    rmentalmf <- mental[sample(mf),]
    rmentalmi <- mental[sample(mi),]
    rmentalff <- mental[sample(ff),]
    rmentalfi <- mental[sample(fi),]
    rcorrs <- c(
            cor(rmentalmf[,1:3],physmf[,1:3]),
            cor(rmentalmf[,4:6],physmf[,4:6]),
            cor(rmentalmi[,1:3],physmi[,1:3]),
            cor(rmentalmi[,4:6],physmi[,4:6]),
            cor(rmentalff[,1:3],physff[,1:3]),
            cor(rmentalff[,4:6],physff[,4:6]),
            cor(rmentalfi[,1:3],physff[,1:3]),
            cor(rmentalfi[,4:6],physff[,4:6])  )
    rmin <- c(rmin,min(rcorrs))
    rmax <- c(rmax,max(rcorrs))
    rabs <- c(rabs,max(abs(min(rcorrs)),abs(max(rcorrs))))
    }

twot <- length(rabs[rabs>=absobs])/M # Two sided
lowt <- length(rmin[rmin<=obsmin])/M # Lower tailed
upt <- length(rmax[rmax>=obsmax])/M # Upper tailed

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
     } # End function merror

cat("Correlations Between Mental and Physical \n")
cat(" \n") ; cat(" \n")
cat("     Minimum Observed Correlation: ",obsmin,"\n")
cat("     Randomization p-value (one-sided): p-hat = ",lowt," \n")
cat("     Plus or minus 99% Margin of error = ",merror(lowt,M,0.01),"\n")
cat(" \n")
cat("     Maximum Observed Correlation: ",obsmax,"\n")
cat("     Randomization p-value (one-sided): p-hat = ",upt," \n")
cat("     Plus or minus 99% Margin of error = ",merror(upt,M,0.01),"\n")
cat(" \n")
cat("     Maximum Observed Absolute Correlation: ",absobs,"\n")
cat("     Randomization p-value (two-sided): p-hat = ",twot," \n")
cat("     Plus or minus 99% Margin of error = ",merror(twot,M,0.01),"\n")
cat(" \n")
```

11.2 Bootstrap

To appreciate the bootstrap, recall the idea of a *sampling distribution.*

If the sample size is large enough, the histogram of the sample data is a lot like the histogram of the entire population. Thus, sampling from the sample *with replacement* is a lot like sampling from the population. Sampling from the sample is called **resampling**. One can approximate the sampling distribution of a statistic as follows.

- Select a random sample of size n from the sample data, *with replacement.*
- Compute the statistic from the resampled data.
- Do this over and over again, accumulating the values of the statistic.
- A histogram of the values you have accumulated will resemble the sampling distribution of the statistic.

```
> # boot1.R    Working on the bootstrap
> # Run with    R --vanilla < boot1.R > boot1.out &
> # grades.dat has 4 columns: ID, Verbal SAT, Math SAT and 1st year GPA
>
> marks <- read.table("grades.dat")
> n <- length(marks$verbal) #$
> n
[1] 200
> marks[1:10,]
   verbal math gpa
1     623  509 2.6
2     454  471 2.3
3     643  700 2.4
4     585  719 3.0
5     719  710 3.1
6     693  643 2.9
7     571  665 3.1
8     646  719 3.3
9     613  693 2.3
10    655  701 3.3
> obscorr <- cor(marks)
> obscorr
          verbal      math       gpa
verbal 1.0000000 0.2746341 0.3224477
math   0.2746341 1.0000000 0.1942431
gpa    0.3224477 0.1942431 1.0000000
> # Question: Is the correlation between Verbal SAT and GPA the same as
> # the correlation between math SAT and GPA?
> # What is the sampling distribution of the difference between correlation
> # coefficients?
> #
> obsdiff <- obscorr[3,1]-obscorr[3,2] # Verbal minus math
> obsdiff
[1] 0.1282046
> # The strategy will be to obtain a 95% bootstrap confidence interval for
> # the difference between verbal correlation and math correlation. This
> # confidence interval will be approximately centered around the observed
> # difference obsdiff = .128.  If the confidence interval does not include
> # zero, we will conclude that the observed difference is significantly
> # different from zero.
>
> BOOT <- 1000  ;  bsdiff <- NULL ; set.seed(9999)
> # Accumulate bootstrap values in bsdiff
> # For clarity, do operations in several separate steps inside the loop
> for(i in 1:BOOT)
+     {
+     bootmarks <- marks[sample(1:n,replace=TRUE),] # sample rows with
+                                                   # replacement
+     bcorr <- cor(bootmarks)  # Correlation matrix of bootstrap sample
+     bdiffer <- bcorr[3,1]-bcorr[3,2] # Differencce between correlation
+                                      # coefficients
+     bsdiff <- c(bsdiff,bdiffer) # Add bdiffer to the end of bsdiff
+     } # Next bootstrap sample
> bsdiff <- sort(bsdiff)
> # Now get lower and upper limits of 95% CI
> low <- bsdiff[.025*BOOT] ; up <- bsdiff[.975*BOOT + 1]


> low ; up
[1] -0.03643594
[1] 0.3032818
> (low+up)/2
[1] 0.1334230
> obsdiff
[1] 0.1282046
> write(bsdiff,"bsdiff.dat") # Maybe for later analysis
> pdf("bsdiff.pdf") # Send graphics output to pdf file
> hist(bsdiff)
```

Bootstrap regression tests Fit the reduced model. Assemble resampled data sets by sampling with replacement from the residuals, and forming $\widehat{Y}$ plus the residual. Test full vs reduced model each time. Proportion of simulated F statistics at or above observed F is the bootstrap p-value.