

Example of Computer Randomization

Suppose you have $N = 31$ EU's you want to assign to 4 treatments, with $n_1 = n_2 = n_3 = n_4 = 7$ and $n_4 = 10$.

Lecture 3
September 9, 2002

Christopher Bingham, Instructor
612-625-7023
612-625-1024

Class Web Page

<http://www.stat.umn.edu/~kb/c1asses/5303>

© 2002 by Christopher Bingham

You could put slips numbered 1, 2, ..., 31 into a box, shuffle well, and draw numbers sequentially, assigning treatment 1 to EU's whose numbers were the first 7 drawn, treatment 2 to the next 7,

This procedure provides you a random permutation (reordering) of $\{1, 2, 3, \dots, 31\}$ with all permutations equally likely. You use the permuted set of numbers in assigning treatments.

Any other way of producing a random permutation could be used instead of drawing numbers from a box. One such way uses random numbers (more accurately *pseudo-random* numbers) generated on a computer.

Here is one way to do this in MacAnova.

```

Cmd> n <- vector(7, 7, 10) # sample sizes
Cmd> N <- sum(n); N # total number of EU's
      31
Cmd> treatments <- rep(0,N) # empty vector to be filled
Cmd> u <- runi(N) # random sample of uniform random variables
Cmd> J <- rank(u); J # their ranks are a random permutation
      21 17 16 14 11
      3 29 6 1 26
      27 25 13 28 2
      4 18 23 9 31
      10 30 7 5 24
      8 19 20 22 12
      15 underlining gives treatment groups
(31)
Cmd> treatments[J[run(n[1])]] <- 1 # Assign trt 1 to first 7
Cmd> treatments[J[n[1] + run(n[2])]] <- 2 # trt 2 to next 7
Cmd> treatments[J[n[1] + n[2] + run(n[3])]] <- 3 # 3 to next 7
Cmd> treatments[J[n[1] + n[2] + n[3] + run(n[4])]] <- 4
Cmd> print(paste(treatments)) # final assignment
2 3 1 3 4 2 4 4 3 3 1 4 2 1 4 1 1 3 4 4 1 4 3 4 2 2 2 2 1 4 3

```

With experience, you can use the looping facilities of MacAnova to do it more compactly.

```

Cmd> treatments <- rep(0,sum(n)); Ni <- 0
Cmd> for(i,1,rows(n)){
  treatments[J[Ni + run(n[i])]] <- i
  Ni <- Ni + n[i];;
}
Cmd> print(paste(treatments))
2 3 1 3 4 2 4 4 3 3 1 4 2 1 4 1 1 3 4 4 1 4 3 4 2 2 2 2 1 4 3

```

This works for any set of n's.

3

Randomization in Inference

Example from text.

Table 1 had data on the length of time it took each of 30 workers to runstitch a collar on a man's shirt, using a **standard** workplace and an **ergonomic** workplace, the two "treatments".

The data are paired, because each worker stitched once at each workplace.

Here is how you might randomize the order:

```

Cmd> N <- 30 # numbers of pairs
Cmd> first <- rbin(N, 1, .5) + 1 #first treatment
Cmd> print(paste(first))
1 1 2 1 2 2 2 2 1 1 2 2 1 2 2 1 1 2 2 1 1 2 2 1 1 2 2 2 2 1

```

Here `rbin(N, 1, .5)` generates 30 independent Bernoulli (binomial with $n = 1$) random variables with $p = .5$ (computer generated coin flips). Adding 1 turns the 0's and 1's into 1's and 2's.

4

The standard “normal theory” test is the paired t-test, essentially a one sample t-test on the differences $X_1 - X_2$, where X_1 and X_2 are times using the standard and ergonomic workplaces, respectively.

You can use `tval()` in MacAnova to do it.

```

Cmd> readdata("",standard,ergonomic)
Read from file "TP1:Stat5303:Data:Ch02:emp2-1.dat"
Column 1 saved as REAL vector standard
Column 2 saved as REAL vector ergonomic

Cmd> list(standard,ergonomic)
ergonomic      REAL      30
standard       REAL      30

Cmd> d <- standard - ergonomic # differences

Cmd> d
(1)  1.03      -0.04      0.26      0.3      -0.97
(6)  0.04      -0.57      1.75      0.01      0.42
(11) 0.45      -0.8      0.39      0.25      0.18
(16) 0.95      -0.18      0.71      0.42      0.43
(21) -0.48      -1.08      -0.57      1.1      0.27
(26) -0.45      0.62      0.21      -0.21     0.82

Cmd> n <- nrow(d); n # number of pairs
(1)  30

Cmd> tstat <- tval(d); df <- n - 1
(1)  1.49      29      0.14701

Cmd> vector(tstat,df,twotail(tstat,df))
(1)  1.49      29      0.14701
t-stat      DF      P-value
    
```

This paired t test assumes the d's are a random sample from $N(\mu_d, \sigma_d)$. It tests the null hypothesis $H_0: \mu_d = 0$ (no difference between workplace types), vs the alternative $H_a: \mu_d \neq 0$ (there is a difference).

Assuming the order was in fact randomized, you can do a randomization-based test whose only assumption is that the randomization was done properly.

Randomization inference is conditional on the actual data observed.

The null hypothesis tested is stated somewhat differently, but has the same interpretation as stating there was no treatment effect.

H_0 : the type of workplace is irrelevant so which number in each pair is labeled standard and which ergonomic is arbitrary.

The numbers observed are considered as if they were fixed. Inferences considers all possible results that might have occurred for the possible outcomes of the randomization.

To simplify, suppose we had only the first $n = 5$ pairs.

Then there are 32 different possible assignments of treatments to the first and second elements of the pairs.

Each possible assignment is essentially a specification of the signs for all the differences.

```
Cmd> d1 <- d[run(5)] # d1 is first 5 differences
Cmd> tvalobs <- tval(d1); tvalobs # paired t
(1) 0.35852
Cmd> twotailr(tvalobs,4) # two tail P-value
(1) 0.73808
```

Omitting some steps, I created a 32 by matrix signs, each row of which contains a different ordered set of -1's and +1's.


```
Cmd> twotailt(tvalobs,4) # normal theory P-value
(1) 0.73808
Cmd> sum(tstats >= tvalobs) # counts number ≥ observed
(1) 10
Cmd> 2*sum(tstats >= tvalobs)/32 # randomization P-value
(1) 0.625
```

It's not the same as the normal theory but gives the same conclusion. For larger samples it is usually quite close.

In fact, with many randomization tests, you don't need to compute the t-statistics, since there is a simpler statistic that is monotonically related.

Paired t statistic is $t = \frac{\bar{d}}{\sqrt{\sum(d-d)^2/(n-1)}}$

A little algebra shows that

$$t = \sqrt{\{n-1\}} \tau / \sqrt{\{1 - \tau^2\}}$$

$$\tau = \sqrt{n} \bar{d} / \sqrt{\{\sum d^2\}} = \sum d / \sqrt{\{n \sum d^2\}}$$

$n \sum d^2$ is the same for all sets of signs, so t depends only on $\sum d$

```
Cmd> sums <- rep(0,32) # place to put sums
Cmd> for(i,1,32){sums[i] <- sum(d1*vector(Signs[i]))}
Cmd> stemleaf(sums) #distribution of sums
 2  -2. | 65
 5  -2* | 000
 6  -1. | 9
 8  -1* | 44
11  -0. | 655
16  -0* | 41000
16  +0* | 00014
11  +0. | 556
 8  1* | 44
 6  1. | 9
 5  2* | 000
 2  2. | 56
```

1*|1 represents 1.1 Leaf digit unit = 0.1

```
Cmd> sumobs <- sum(d1); sumobs # observed value of sum
(1) 0.58
```

```
Cmd> sums >= sumobs # compare with observed
(1) F F F F F F F F
(8) F F F F F F F F
(15) F F F F F F F F
(22) F T T T T T F F
(29) T F F T T T F F
```

This is the same pattern of T's and F's as for the t-statistics.

```
Cmd> 2*sum(sums >= sumobs)/32 # same P-value, too
(1) 0.625
```

There is a macro `randsign()` that does this for you automatically.

```

Cmd> stuff <- randsign(d1)
WARNING: searching for unrecognized macro randsign near
stuff <- randsign(
Cmd> list(stuff)
stuff          REAL      32
Cmd> stemleaf(stuff) # exactly the same as before
  2  -2. | 65
  5 -2* | 000
  6 -1. | 9
  8 -1* | 44
 11 -0. | 655
 16 -0* | 41000
 16 +0* | 00014
 11 +0. | 556
   8  1* | 44
   6  1. | 9
   5  2* | 000
   2  2. | 56
1*|1 represents 1.1 Leaf digit unit = 0.1
Cmd> 2*sum(stuff >= abs(sum(d1)))/32
(1) 0.625
    
```

Unfortunately, you can't do this for the complete data set, since with $n = 30$ there are $2^{30} = 1073741824 = 1.07 \times 10^9$ different assignments of signs. You can, however, randomly select a large number of these. You use keyword `trials`.

```

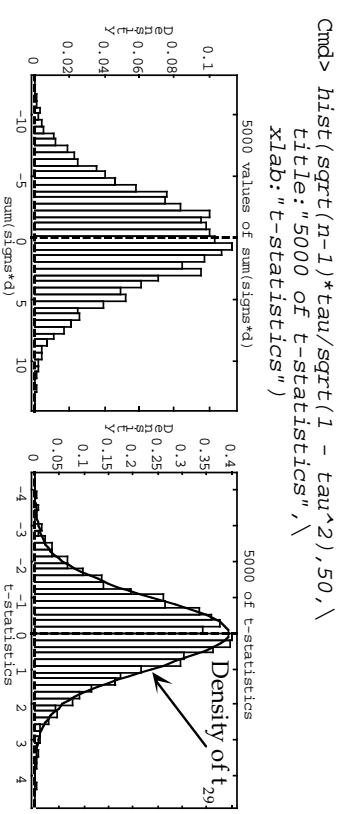
Cmd> stuff <- randsign(d, trials:5000); list(stuff)
stuff          REAL      5000
Cmd> 2*sum(stuff >= abs(sum(d)))/5000 # approximate P-value
(1) 0.1508
Cmd> twotailt(tval(d), n-1) # normal theory P-value
(1) 0.1868
    
```

`randsign()` generated 5000 sums of signed differences.

Here are histograms of the $\sum \pm d_i$ as well as the equivalent t statistics computed as $t = \sqrt{(n-1)}\tau / \sqrt{(1 - \tau^2)}$.

```

Cmd> tau <- stuff/sqrt(30*sum(d^2))
Cmd> hist(stuff, 50, xlab:"sum(signs*d)", \
title:"5000 values of sum(signs*d)")
    
```



On the right, I added the density for t_{29} . You see it is a good approximation to the histogram.