Resampling Stats Illustrations

The birthday problem

(program "birthday")

What is the probability that two or more people among a roomful of 25 have the same birthday? (In technical terms, this is the probability of duplication in a multi-outcome sample from an infinite universe. This famous examination question used in courses on probability shows how powerful and simple the resampling method is.)

We might examine a random number table, select the first 25 numbers falling between 001 and 365 (representing the days in the year), and record whether or not there is a match among the 25. We would then repeat the process enough times to get a reasonably stable probability estimate. Pose the question to a mathematical friend of yours. After watching her sweat, compare your answer to hers. You will find the correct answer very surprising. People who know how this problem works have been known to take unfair advantage of this knowledge by making and winning big bets.

The birthday problem is amazingly simple with RESAMPLING STATS. First, **GENERATE** 25 numbers between 1 and 365 (the number of days in a typical year) and store the results in **a**.

```
GENERATE 25 1,365 a
```

Next, using the **MULTIPLES** command, check to see whether any two or more people have the same birthday by searching for duplicate (or triplicate, quadruplicate, etc.) numbers and put the result in **b**.

```
MULTIPLES a >=2 b
```

We want to keep track of the individual results **b** and keep them in our special "score" vector **scrboard**. (Vector names can have up to 8 characters.)

```
SCORE b scrboard
```

Let's review the program to see where the **REPEAT**, **END** "loop" goes.

```
GENERATE 25 1,365 a
MULTIPLES a >=2 b
SCORE b scrboard
```

The loop begins before **GENERATE** and ends after **SCORE**. We **REPEAT** this loop 1000 times (or more). So, insert a line before **GENERATE** — put your cursor at the beginning of the **GENERATE** line and press return — and type in the **REPEAT** command.

```
REPEAT 1000
```

Then **END** the loop after **SCORE**.

```
END
```

Finally, we count how often **scrboard** recorded a value of 1 or more, indicating a trial with at least 1 replicated birthday, and divide by 1000 to express the result in conventional proportion terms. (Put another way, we look for rooms of 25 people where at least 1 birthday is shared.) Then we **PRINT** the result to the screen.

```
COUNT scrboard >=1 k
DIVIDE k 1000 prob
PRINT prob
```

Now try running the program. The answer may surprise you.

Quality control

(program: “larybird”)

This example might also be used with the later examples on hypothesis testing.

In the first three games of the 1988 NBA playoff series between Boston and Detroit, Larry Bird only got baskets 20 of the 57 shots he attempted in the first three games. Everybody said that Bird, normally a 48% shooter, was in a slump. The *Washington Post* said (May 30, 1988, p. D4):

“Larry Bird is so cold he couldn’t throw a beachball in the ocean… They fully expect Bird to come out of his horrendous shooting slump. It is safe to assume that if Bird doesn’t shake out of his slump Monday, it will be difficult and probably even impossible for Boston…”

By “slump” people meant that the chance of Bird scoring a basket during that period was lower than usual. And in such a case, coaches and players usually conclude that the player should take fewer shots than usual because he does not have a “hot hand.”

Another possibility is that Bird’s performance was not extraordinary and is the sort of thing that could happen just by chance due to ordinary random variability. So let’s see just how unusual it would be for a slot-machine that hits 48 percent of the time to show a “slump” like Bird’s.

Using RESAMPLING STATS, first we set up 1000 trials of this experiment.

This experiment consists of generating, for each trial, a series of 57 “shots,” represented by 57 numbers randomly generated between 1 and 100.

```
REPEAT 1000
     GENERATE 57 1,100 a
```

Next we count how many of those 57 shots were “baskets,” that is, were between 1 and 48 (remember that Bird is a 48% shooter on average). This constitutes our “core procedure,” highlighted above and below, and you could choose to use the “Repeat Wizard” and “Results Wizard” to complete the job. Instead, we will show the commands to do the repetition and scoring tasks.

```
     COUNT a between 1 48 b
```

Next, we **SCORE** the result, and **END** the **REPEAT** loop.

```
     SCORE b scrboard
END
```

Finally, we count the number of times that the result was 20 baskets or fewer.

```
COUNT scrboard <=20 k
```

To express our result as a proportion, we **DIVIDE** by 1000. To see our results, we add a **PRINT** command and a **HISTOGRAM** command.

```
DIVIDE k 1000 prob
PRINT prob
HISTOGRAM scrboard
```

Result:

```
prob = .038
```

The results show that in 38 series out of 1000, our simulated Larry Bird gets 20 or fewer baskets in a series of 57 shots. That means that even if nothing changes in his shooting, about 3 or 4 out of every 100 series of 57 shots, on average, he would shoot that poorly or worse. To verify and refine our estimate, we might run the program again several times, or increase the number of trials.

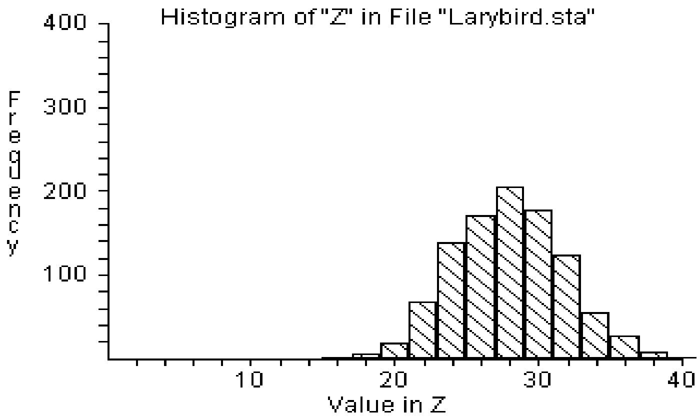


Figure 1: Frequency distribution, number of baskets in 57 shots

A confidence interval for a political survey
(program “bush”)

One of the Gallup polls for the 1988 U. S. presidential election showed 840 (56%) for Bush and 660 for Dukakis (a total of 1500 voters). Estimate bounds on the percentage of the entire electorate that favors Bush.

Put another way, we would like to learn how variable our sample result is. How much might one sample differ from another?

If we had unlimited time and money, we could take additional surveys of 1500 each to see how much they differ from one another. Lacking the ability to go back to the actual universe and draw more samples from it, we do the next best thing — we create a hypothetical “bootstrap” universe based on the sample data, and then draw samples from it.

We therefore use 56% as our best estimate of the percentage of the electorate that favors Bush, then observe how samples of size 1500 from such a hypothesized electorate behave. We want to see how variable one sample proportion is from another. Here we use the **MAXSIZE** command to create vector space for more than 1000 elements.

```
MAXSIZE a 1500
REPEAT 1000                         Do 1000 simulations
    GENERATE 1500 1,100 a           Generate a sample where
                                    1-56 = “favor Bush”
    COUNT a <= 56 b                 Count the Bush votes
    DIVIDE b 1500 c                 Convert to a proportion
    SCORE c scrboard                Keep score
END                                 End the simulation, go back
                                    and repeat until 1000 are
                                    complete
HISTOGRAM scrboard
PERCENTILE scrboard (2.5 97.5) interval
                                    Find the 2.5th and 97.5th
                                    percentiles
PRINT interval
```

Result:

```
interval = 0.539 to 0.584
```

interval is the estimated confidence interval.

NOTE

The shaded area above indicates one iteration of the experiment. An alternative approach is to write this as your "core procedure" (under "Wizards"), then use the "Repeat Wizard" to repeat it, and the "Results Wizard" to show you the percentiles.

Figure 2 reveals the distribution of these resampling proportions. To estimate a 95% confidence interval, we determine those values that enclose 95% of our results, "chopping off" 2.5% at either end. This agrees quite well with the interval obtained by a conventional Normal approximation: P ± 1.96 $(P(1-P)/n)^{½}$ = .535 to .585.

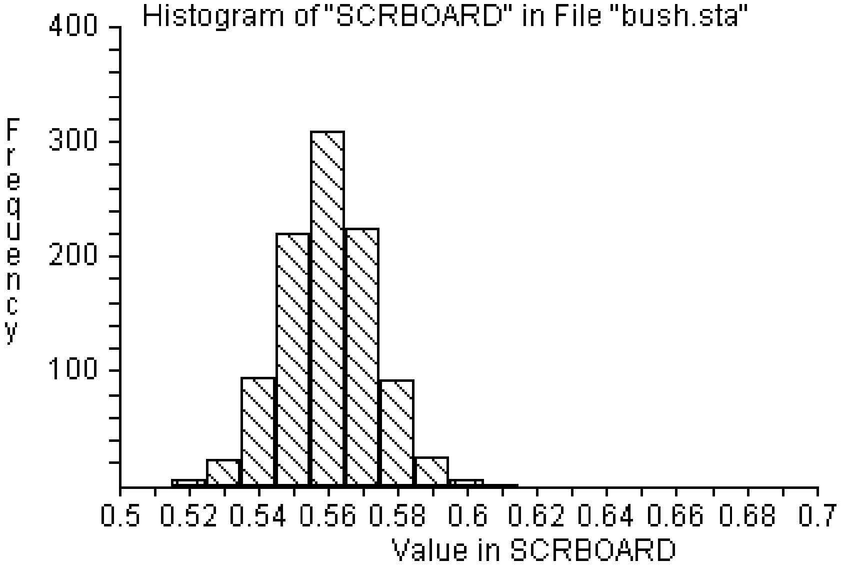


Figure 2: Proportion of favorables in samples of 1500, drawing from a universe that is 56% favorable

Pig weight gains — reliability of the estimate

(program “pigfood”)

(This bootstrap example is from *Basic Research Methods in Social Science*, Julian L. Simon, 1969.)

An agricultural lab decides to experiment with a new pig ration — ration A — on twelve pigs. After 4 weeks, the pigs experience an average gain of 508 ounces. The weight gains of the individual pigs are as follows: 496, 544, 464, 416, 512, 560, 608, 544, 480, 466, 512, 496.

In presenting these results to major agricultural feed distributors, the lab wants to report not just the average estimated weight gain (as represented by the sample average), but also the possible range of sampling error.

How can we determine the extent to which one sample differs from another? (The reliability of our estimated mean weight gain.) If we had more time and money, we could try the ration on additional groups of 12 pigs, and see how the mean weight gain differed from group to group.

Lacking time and money, we will create a hypothetical “bootstrap” universe that we can draw re-samples from.

If the real universe is made up of pig weight gains like those we observed in our sample), we can represent this universe with, say, 1 million (or a billion, or a trillion) weight gains of 496 ounces, 1 million of 544 ounces, and so on for the observed weight gains. We could then draw repeated re-samples of 12 from this universe. Each time we draw a weight gain, each of the original weight gains has the same probability of being selected. As we draw these re-samples, we want to observe and record the means of each re-sample. At the conclusion of all the trials we identify those values that enclose, say, 95% of all the trial re-sample means. These values are the endpoints of a 95% estimated confidence interval around the sample mean.

Recognizing that it would be tedious to create a simulated universe with millions of values, we can achieve the same effect by selecting our new samples of 12 directly from the original sample randomly and *with replacement*. In that way, we have effectively created an infinite universe, “bootstrapped” from our sample. In RESAMPLING STATS:

First, we must record the initial data.

```
DATA (496 544 464 416 512 560 608 544 480 466 512 496) a
```

Record the data

Could use the **COPY** command also — it is the same as **DATA**

```
REPEAT 1000                 Do 1000 trials
    SAMPLE 12 a b           From "a," take a sample of 12 with
                            replacement and put it in "b"
    MEAN b c                Calculate the mean of the resample,
                            put the result in "c"
    SCORE c scrboard        Score the result
END                         End one trial, go back and repeat —
                            Keep repeating until all 1000 are
                            complete
HISTOGRAM scrboard          Produce a histogram of trial results
                            (Figure 3)
```

To help us pin down the confidence interval more precisely, we use the command **PERCENTILE**:

```
PERCENTILE scrboard (2.5 97.5) interval
                            Calculate the 2.5th and 97.5th
                            percentiles of trial results
PRINT interval
```

NOTE

The shaded area above indicates one iteration of our simulation experiment. An alternative approach is to enter those commands in the Wizards "core procedure" window, then use the "Repeat Wizard" to repeat the procedure and the "Results Wizard" to get the percentiles. (This would cause the data entry step to be needlessly repeated, but the only effect of that will be to slow down the computer a bit.)

Results:

interval = 480 to 537 = estimated 95% confidence interval

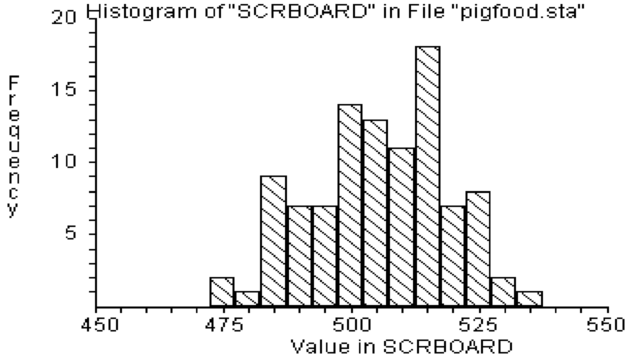


Figure 3: Average weight gain of 12 pigs, drawing bootstrap samples of size 12 from the given data, 1000 simulations

This agrees reasonably well, but is a bit narrower than the conventional t approximation of 476 to 540. In fact, Monte Carlo simulations show that in this case the bootstrap interval is slightly narrower than the "true" interval, a bias that diminishes with larger samples and less extreme confidence intervals. See Efron (1993) for a discussion of refinements of the bootstrap that produce greater accuracy.

How does this "bootstrap" approach relate to the conventional formulaic approach? In reality they are parallel methods, both seeking to answer the question "how would other samples behave when drawn from the same universe that spawned our original sample?" A "bootstrap" approach says "let's constitute a hypothetical universe — the sample itself — that represents our best guess about the real universe, then draw lots of samples from it." A conventional approach says "Let's describe a hypothetical distribution — say the Normal — that might have spawned our sample. We'll use the parameters mean and standard deviation for this description. Then we'll use someone else's description of how samples drawn from such a hypothetical universe behave, employing formulas and tables."

This same bootstrap sampling distribution could be used in answering a different question. Suppose we are now concerned with the merchant who will distribute the new feed. This firm wants to advertise an average weight gain level to minimize the probability that the real average is less than his advertised average. Specifically, it wants to select a value for advertising such that 95% of the samples had a higher average.

Again, we use the command **PERCENTILE**:

```
PERCENTILE scrboard 5 c5    Calculate the 5th percentile of
                            trial results

PRINT c5
```

Results:

```
c5 = 487
```

95% of the values lie above this point — it constitutes the lower bound of a one-tailed 95% confidence interval that includes the population mean.

NOTE: HOW PERCENTILES ARE CALCULATED

The PERCENTILE command first sorts the vector values, then calculates that value corresponding to a specified percentile ***X***. If there is a single value ***Y*** such that the percentage of the distribution below ***Y*** is less than ***X*** and the percentage of the distribution above ***Y*** is less than ***(1-X)***, that value ***Y*** is the percentile sought. If the specified percentile does not correspond exactly to a single position in the distribution of values, an average of the values bracketing the percentile position will be used. This procedure is analogous to the method for calculating the median.

Using the PERCENTILE command, we can also see more than one confidence interval at the same time:

```
PERCENTILE scrboard (.5 2.5 97.5 99.5) intervals
```

Gives you the percentiles corresponding to the above numbers — the resulting vector "intervals" contains the endpoints for 99% and 95% confidence intervals

Generalization — the bootstrap

The bootstrap procedure can be generalized to many other situations where you have taken a sample, calculated a statistic for the sample, and want to know how reliable that statistic is as a measure for the population from which the sample came. If it were possible, we would take an additional sample (or samples) from the population and see how the statistic varied. We don't have the population available — but we do have our sample, which is the best proxy to the population. We can expand, or bootstrap, our sample into the hypothetical population in which we are interested by replicating each member of the sample, say, a million times. Then we can proceed to take a new sample from this hypothetical population and see how the statistic changes.

A shortcut is to take the new sample from the sample itself, *with replacement*. Here's why: each time we select an observation for the new sample, we want each of the elements of our original sample to have an equal chance of being selected — hence our decision to sample with replacement. This is just about the same thing as replicating each member of the sample, say, a million times and sampling without replacement.

Of course, we can (and should) take more than one new sample — 1,000 or 10,000 can be done easily on the computer.

A confidence interval for a median
(program “profits”)

Sometimes, especially with highly skewed data such as incomes, the median is preferred to the mean as a measure of the distribution center. Constructing a confidence interval for the median is easy with RESAMPLING STATS.

Say you need to come up with a quick estimate of the profits of a typical American Fortune 1000 business, and the extent to which that estimate might be in error. You draw a random sample of 15 firms, finding their profits (in $ million) to be: 1315, 288, 155, 37, 99, 40, 170, 66, 500, 419, 125, -90, -63, 29, 966. We use RESAMPLING STATS to calculate the median profit, and construct a bootstrap confidence interval:

```
DATA (1315 288 155 ...) a            Record the profits of the 15 firms.
Could use the COPY command
also — it is the same as DATA

MEDIAN a meda                        Find the median, call it “meda”

REPEAT 1000                          Do 1000 trials

    SAMPLE 15 a aa                   Take a sample of 15 with replacement

    MEDIAN aa med$                   Find the median of the resample,
                                     call it “med$”

    SCORE med$ scrboard              Keep score of the trial result

END                                  End one trial, go back and repeat

PERCENTILE scrboard (5 95) interval
                                     Calculate the 5th and 95th percentiles

PRINT meda interval                  Print the actual sample median, plus
                                     the 5th and 95th percentiles of the
                                     resample medians
```

Results:

```
meda   = 125

interval    = 40  288
```

The sample we chose has a median of $125 million, and our 90% confidence interval runs from $40m to $288m.

NOTE

The shaded area above indicates one iteration of our simulation experiment. An alternative approach is to enter those commands in the “core procedure” window under “Wizards,” then use the “Repeat Wizard” to repeat it, and the “Results Wizard” to get the percentiles.

A confidence interval for net profit

(program "mailing")

To test potential response to a new book offer, a mail order company sends a mailing to 10,000 potential customers, randomly selected from lists of millions. If the offering is successful, it will be mailed to the much larger lists of recipients.

The results of the test are categorized into "no" (negative response), "silent" (no response), "order/return," "order/bad debt," and "order/pay." The value of a customer who orders the title is $45 (this includes the immediate profit from the book, plus the net present value of possible future orders). The customers who order and return, and those who order and never pay, are worth $8.50 and $9.50 respectively, reflecting the processing costs of the customer transactions and correspondence, and the value of material shipped. The silent customer costs the firm just the outgoing mailing costs — $.40. The customer who responds "no" costs costs the firm the outgoing mailing costs, plus the cost of the postpaid reply.

Action	number	proportion	profit rate	profit
No	500	0.05	-$0.95	-$475
Silent	9200	0.47	-$0.40	-$3,680
Order/return	90	0.009	-$8.50	-$765
Order/bad debt	30	0.003	-$9.50	-$285
Order/pay	180	0.018	$45.00	$8,100
Profit				$2,895

The profit from the test mailing is $2,895. How reliable an estimate is this? How different might it be with a different sample from the same population?

We answer this question by constituting a hypothetical population of outcomes, and drawing samples of 10,000 from it so we can see how those samples behave. Our best guess about what the population of outcomes looks like is the sample itself, so we could replicate the sample many times and constitute a hypothetical larger universe. Alternatively, we can sample with replacement from the sample itself (effectively the same thing as replicating the sample an infinite number of times and sampling without replacement).

Here are the specifics:

1. Constitute an urn with slips of paper recording the various outcomes (profits), in the quantities in which they occurred in the sample: 500 -.95's, 9200 -.4's, 90 -$8.5's, 30 -9.5's, and 180 45's.

2. Draw a sample of 10,000 randomly and with replacement.

3. Sum the profit values in the resample & record.

4. Repeat many times.

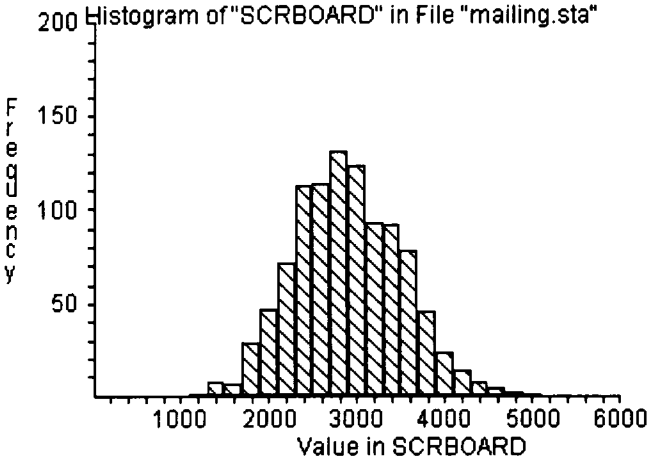
Here's the program:

```
MAXSIZE orders 10000 orders$ 10000
URN 500#-.95 9200#-.4 90#-8.5 30#-9.5 180#45 orders
REPEAT 1000
    SAMPLE 10000 orders orders$
    SUM orders$ profit$
    SCORE profit$ scrboard
END
HISTOGRAM scrboard
PERCENTILE scrboard (5 95) int
PRINT int
```

Results:

```
INT   =   1920   3896
```

The 90% bootstrap confidence interval for the profit runs from $1,920 to $3,896.



Confidence Interval for a Measure of Agreement
(program "kappa")

Radiographer A examines 160 slides, reports 18 positives and 142 negatives. Radiographer B examines the same slides and comes up with 8 positives and 152 negatives. Here's how the readings split up:

		Radiographer A		
		+	-	
Radiographer B	+	7	1	8
	-	11	141	152
		18	142	160

The Kappa statistic is used to measure the extent of agreement, where

$$\text{Kappa} = \frac{\text{\# agreements observed - \# agreements expected}}{\text{\# slides reviewed total - \# agreements expected}}$$

The term "expected" here means what would occur if the two radiographers did not agree with each other more than what chance would produce. For example, A had 142 negatives. Of these 142 negatives, just by chance, we would expect B to rate 95% (152/160) as negative, or 134.9. More formally,

The expected count for each cell in the 2x2 table is the product of the marginals for that cell, divided by the total. For the negative-negative cell:

(152*142)/160 = 134.9

Expected counts:

.9	7.1
17.1	134.9

Agreements are denoted by the upper left and lower right cells, so expected agreement is 135.8. Kappa is calculated as follows:

Kappa = (148-135.8)/(160-135.8) = .5

Kappa can range between -1 and 1; it tells us what proportion we have attained of possible total agreement in excess of chance.

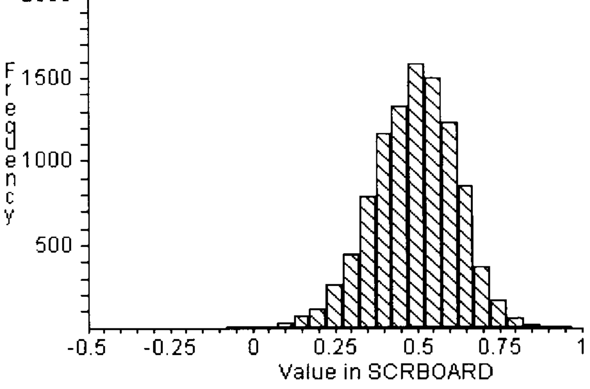
The observed value in this case is .5. How reliable is this estimate? Let's derive a bootstrap confidence interval for it.

Solution: We want to know how additional samples of 160 slides would fare, being examined by the same radiographers. Would the Kappa statistic vary greatly from one sample of 160 to another? We don't have additional sets of slides to look at, so we take additional samples from the universe suggested by our sample. We note that we had four types of slides: 7 pos-pos, 1 Aneg-Bpos, 11 Apos-Bneg, and 141 neg-neg. We could imagine marking 7 cards “pp”, 1 card “np”, 11 cards “pn”, and 141 cards “nn”, and shuffling them in a hat. We would draw a sample with replacement, count the numbers of various types of cards, recalculate the Kappa statistic, and record it. Do this many times, and observe the distribution of the resampled Kappa statistic.

```
MAXSIZE scrboard 10000

URN 7#1 1#2 11#3 141#4 a    Record the observed cases,
                            Letting #1 represent “pp”, #2
                            represent “np”, #3 “pn” and #4 “nn”.

REPEAT 10000

   SAMPLE 160 a b           Take a sample of 160 cases, with
                            replacement

   COUNT b =1 a11           Count the “pp” cases

   COUNT b =2 a12           Count the “np” cases

   COUNT b =3 a21           Count the “pn” cases

   COUNT b =4 a22           Count the “nn” cases

   LET xagrees =
   ((a11+a21)*(a11+a12)160)+((a12+a22)*(a21+a22) / 160)
                            calculate the expected agreements
                            (keep it all on one line, though)

   LET kappa = ((a11+a22)-xagrees) / (160-xagrees)
                            calculate kappa

   SCORE kappa scrboard

END

HISTOGRAM scrboard

PERCENTILE scrboard (5 95) int

PRINT int
```

Results:

```
int = 0.27711   0.68932
```



NOTE

The shaded commands above indicate one iteration of our simulation experiment. An alternative approach is to enter these commands in the “core procedure” window under “Wizards,” use the “Repeat Wizard” to repeat the procedure, and the “Results Wizard” to find out how often the simulation result was as extreme as (or more extreme than) the observed result.

Hypothesis test for a difference in proportions

(program “noshow”)

Fly-By-Night Airways wishes to determine whether its business routes have a different passenger no-show rate than its vacation routes. It divides its routes into business and vacation, setting aside those routes that are neither. Taking a sample of 1000 reservations for each group, it determines that 384 passengers failed to show up on the business routes, while 341 failed to show up on the vacation routes. How likely is it that such a difference (43 or greater) occurred by chance, if the routes do not differ in this respect?

If the two routes do not differ, then our best estimate of the overall no-show rate (our null hypothesis) is (384+341)/2000 = 36.3%. We now want to test whether 2 groups of 1000 drawn from this no-show rate are likely to show differences as great as those observed.

```
REPEAT 1000                          Do the experiment 1000 times
    GENERATE 1000 1,1000 business
                                     Generate 1000 bookings to represent
                                     the business routes
    GENERATE 1000 1,1000 vacation
                                     Same for vacation routes
    COUNT business between 1 363 bnoshow
                                     Count the business no-shows (recall
                                     that the null hypothesis overall
                                     no-show rate is 36.3%)
    COUNT vacation between 1 363 vnoshow
                                     Similarly for the vacation no-shows
    SUBTRACT bnoshow vnoshow dif
                                     Find the number of excess business
                                     no-shows
    SCORE dif scrboard               Keep score
END                                  Go back and repeat until 1000 trials
                                     are complete, then proceed
```

HISTOGRAM scrboard	Produce a histogram of the trial results (Figure 5)

How often was the difference >= the observed difference? These commands will calculate this for us:

COUNT scrboard >=43 k

DIVIDE k 1000 prob

PRINT prob

Result:

prob = .028

NOTE

The shaded commands above indicate one iteration of our simulation experiment. An alternative approach is to enter these commands in the "core procedure" window under "Wizards," use the "Repeat Wizard" to repeat the procedure, and the "Results Wizard" to find out how often the simulation result was as extreme as (or more extreme than) the observed result.

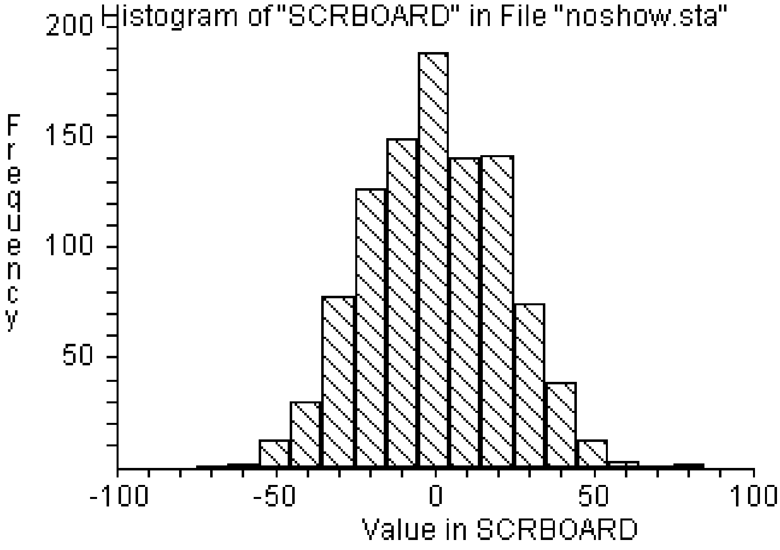


Figure 5: Difference in number of no-shows between two samples, drawing two samples of size 1000 where the no-show probability is 36.3% for each draw, 1000 simulations

From the histogram (Figure 5) and the result, we see that obtaining 43 or more "excess" business no-shows by chance is highly unlikely — it happened only 28 out of 1000 tries, for an estimated "p-value" of .028.

Hypothesis test for a difference in means
(program "battery")

Does one brand of car battery last longer than another? Here are the figures on 10 randomly selected batteries each for brand A and brand B. Is brand A's apparent advantage significant?

TABLE 2. BATTERY LIFE TIMES

brand	life (months)	aveage
A	30 32 31 28 31 29 29 24 30 31	29.5
B	28 28 32 31 24 23 31 27 27 31	28.2

A's advantage: 1.3 months

Our null hypothesis is that there is no difference between the two types of batteries. If this is the case, then we can consider them part of the same "population," writing their durations down on slips of paper and tossing the slips all in the same hat. Next we draw out 10 slips of paper to represent pseudo brand A, followed by 10 slips of paper to represent pseudo brand B. For purposes of this example, we will draw with replacement, since we want to make inferences to a larger population, process or system that produced these batteries. We calculate the average life time of each group, and determine whether they differ by as much as the observed data.

In making our draws, we have a choice—we could sample with replacement (a bootstrap procedure) or without replacement (a permutation or randomization test). A discussion of sampling with or without replacement can be found at www.resample.com\ permutation.htm. For purposes of this example we will try it both ways to see how much of a difference it makes. Since we're comparing two methods, let's set the number of trials at 15,000 (requiring the use of **MAXSIZE** at the outset of the program to increase the capacity of the scorekeeping vector "scrboard" from the default value of 1000 to 15,000).

In RESAMPLING STATS:

```
MAXSIZE scrboard 15000
COPY (30 32 31 28 31 29 29 24 30 31) a
                                      Record the data
COPY (28 28 32 31 24 23 31 27 27 31) b
CONCAT a b c                          Combine the data
REPEAT 15000                          Do 15000 trials
```

```
    SAMPLE 10 c d          Take a sample, size 10, with replacement,
                           and call it "d"
    SAMPLE 10 c e          Take our second sample
    MEAN d dd              Find the means of the two samples
    MEAN e ee
    SUBTRACT dd ee f       Find the difference between the two
                           sample means
    SCORE f scrboard       Keep the score of the difference
END                        End one trial, go back and repeat until
                           1000 are complete, then proceed
HISTOGRAM scrboard         Produce a histogram of the trial results
COUNT scrboard >=1.3 k
DIVIDE k 15000 prob
PRINT prob
```

How often did “D” group exceed “E” group by 1.3 or more?

Result:

prob = .147

NOTE

The shaded commands above indicate one iteration of our simulation experiment. An alternative approach is to enter these commands in the “core procedure” window under “Wizards,” use the “Repeat Wizard” to repeat the procedure, and the “Results Wizard” to find out how often the simulation result was as extreme as (or more extreme than) the observed result.

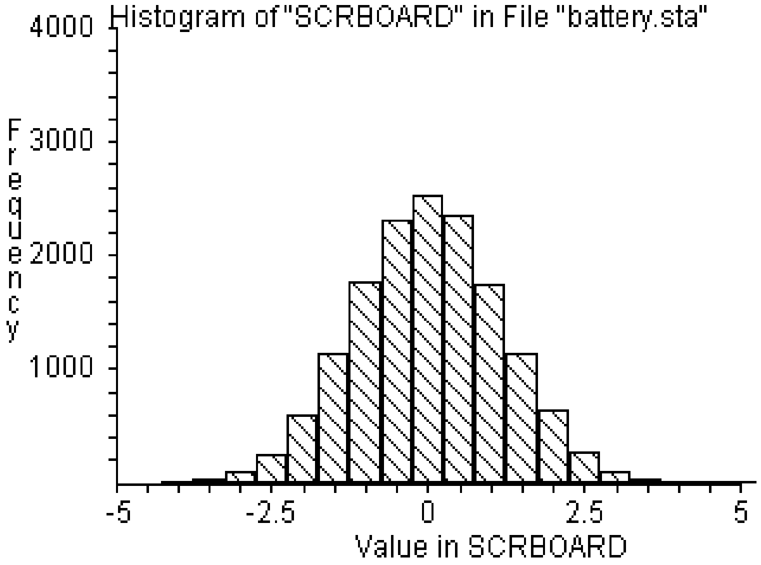


Figure 6: Difference in mean battery lifetime between two samples of size 10, each drawn from the given data, 15000 simulations

From the histogram (Figure 6) and the result, we see that a difference of 1.3 months would not be at all unusual if we draw random samples of 10 from our combined population. The cases where randomly-drawn group D’s mean exceeds randomly-drawn group E’s mean are not unusual — this occurs 14% of the time for an estimated “p-value” of .147.

Battery without replacement

Here is the same problem, the only change being that we use sampling without replacement instead of sampling with replacement.

The **TAKE** command selects specified elements from a vector. The **SHUFFLE / TAKE** combination is our way of sampling without replacement:

```
MAXSIZE scrboard 15000
COPY (30 32 31 28 31 29 29 24 30 31) A
COPY (28 28 32 31 24 23 31 27 27 31) B
CONCAT A B C
REPEAT 15000
```

```
    SHUFFLE c d
    TAKE d 1,10 e
    TAKE d 11,20 f
    MEAN e ee
    MEAN f ff
    SUBTRACT ee ff g
    SCORE g scrboard
END
HISTOGRAM scrboard
COUNT scrboard >= 1.3 k
DIVIDE k 15000 prob
PRINT prob
```

Result:

```
prob = .168
```

NOTE

The shaded commands above indicate one iteration of our simulation experiment. An alternative approach is to enter these commands in the “core procedure” window under “Wizards,” use the “Repeat Wizard” to repeat the procedure, and the “Results Wizard” to find out how often the simulation result was as extreme as (or more extreme than) the observed result.

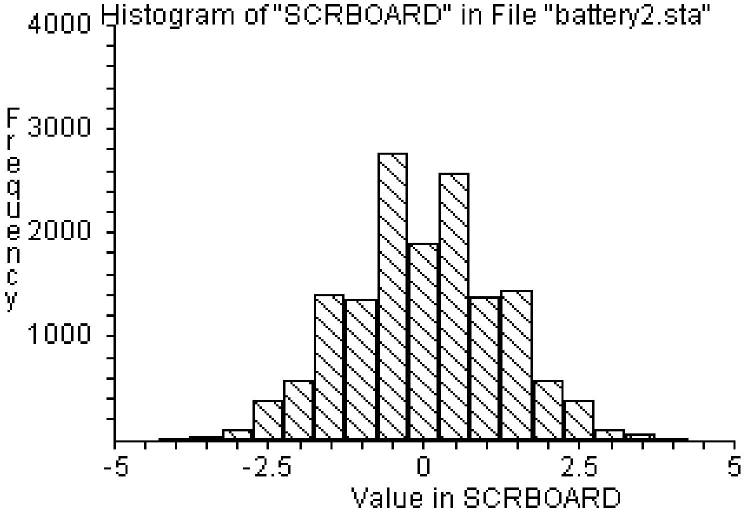


Figure 7: Difference in mean battery lifetime between two samples of size 10, each drawn from the given data, 15,000 simulations

While sampling without replacement yields the same conclusion – not statistically significant – as sampling with replacement, in this case there is a slight difference in p-values.

Historically, the two represent different strands of thinking in statistics.

Sampling with replacement is a bootstrap procedure, first published by Julian Simon in 1969 (Simon, 1969) and named and developed in the literature by Bradley Efron in the late 1970’s and early 1980’s (Efron, 1979 and 1982).

Sampling without replacement is the same thing as a Monte Carlo or approximate version of a permutation test (also called a randomization test).

Suggested by R.A. Fisher (Fisher, 1935) and elaborated by E. J. G. Pitman (Pitman, 1937), permutation tests involve exhaustively enumerating all the ways the combined battery data could be split into two sub-samples of size 10 each, to use the current example as an illustration.

Meyer Dwass (Dwass, 1957) later suggested that random samples selected Monte Carlo style from all these permutations could substitute for an exhaustive enumeration of them. This is the “sampling without replacement” procedure used above.

Permutation tests and their Monte Carlo or “approximate” counterparts are “exact” tests – they can be counted on to produce Type-I error rates at or below the nominal level of the test. For example, if the level of the test is .05, an exact test produces (erroneous) “significant” results 5% of the time or less when testing a null model.

Owing to the long history of permutation procedures and their exact nature, many practitioners of resampling generally prefer sampling without replacement (a permutation test) rather than with replacement (a bootstrap procedure) whenever such a procedure makes sense in the context of the data and problem.

Baseball payroll—hypothesis test for correlation using the Pearson correlation coefficient

(program "baseball")

Is a baseball team's performance directly related to its payroll? (In technical terms, is there a correlation between two variables, or are they independent?) Specifically, we want to know whether baseball teams with high payrolls also tend to be the better performing teams.

The following data are from the *Washington Post*, March 27, 1998, page F2, and were compiled by the *Post* according to the formula of the Player Relations Council. Performance is ranked by the teams' won-loss records; note that good performance is denoted by a *low* rank number.

TABLE 3. MAJOR LEAGUE PAYROLL AND WON-LOSS RANKS 1995–1997

	Total Payroll	Rank*
NY Yankees	192.7	3
Baltimore	179.5	4
Atlanta	164.8	1
Cleveland	155.7	2
Chicago WS	150.3	14
Cincinnati	143	9.5
Texas	139.9	11
Colorado	138.3	8
Toronto	137.4	25
St. Louis	137.3	19.5
Seattle	137.1	6
Boston	131.8	7
Los Angeles	128.3	5
San Francisco	124	18
Chicago Cubs	123	21
Florida	122.8	12
Anaheim	116	15.5
Houston	115.4	9.5
Philadelphia	109.9	26
San Diego	104.5	13
NY Mets	104.2	17
Kansas City	101.1	22
Minnesota	94.6	27
Oakland	85.5	23.5
Detroit	84	28
Milwaukee	78.5	19.5
Pittsburgh	67.7	23.5
Montreal	67.6	15.5

*Rank in games won and lost over the 3-year period.

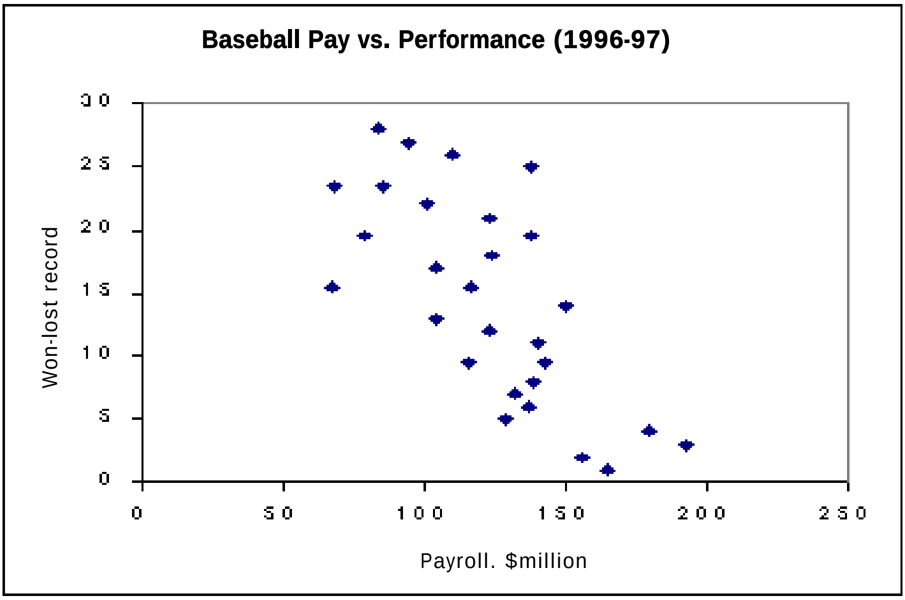


Figure 8: Scatterplot, baseball payroll vs. performance

We can guess that if there is a relationship here, it might well be a linear one. Statisticians typically measure the strength of a linear relation ship with the Pearson correlation coefficient, which can range between -1 (strong negative correlation) through 0 (no correlation) to +1 (strong positive correlation). The Pearson correlation coefficient for these data is -.71. Is that statistically significant? Might a correlation coefficient this negative have occurred just by chance? In other words, might there be no relationship between the two variables and the apparent correlation be the result of the "luck of the draw?"

Here is a resampling procedure to test this possibility:

1. Write down all the payroll numbers on one set of cards, one number per card. Do likewise with the performance ranks.
2. Shuffle the "pay" cards (or both sets of cards—either procedure will achieve full randomization of one variable relative to the other), and deal out the "rank" cards along side the "pay" cards.
3. Calculate the correlation coefficient of the shuffled arrays, and record.
4. Repeat steps 2-3 many times, say 1000 times.
5. Find out how often the shuffled correlation coefficient is as low as or lower than the observed value of -.71.

The data presented in the above table are in a data file called "baseball.dat," which is an ASCII (text) file with two columns of numbers. The first column is the payroll numbers, and the second

```
READ file "baseball.dat" pay rank

CORR pay rank r              Calculate the observed correlation
                             coefficient

REPEAT 1000

    SHUFFLE rank rank$       Shuffle the ranks, call the result rank$
                             (the "$" is often used to denote the
                             resampling counterpart to an
                             observed variable)

    CORR pay rank$ r$        Find the correlation between the
                             shuffled ranks and pay; call this
                             resampled correlation coefficient r$

    SCORE r$ scrboard        Keep score of this resampled
                             correlation coefficient in scrboard

END

HISTOGRAM scrboard

COUNT scrboard <=r k         How often did the resampled
                             (shuffled) correlation coefficient fall at
                             or below the value of the observed
                             correlation coefficient?

DIVIDE k 1000 prob           Convert to a proportion for an
                             estimated p-value

PRINT r prob
```

Result:

```
prob = 0
```

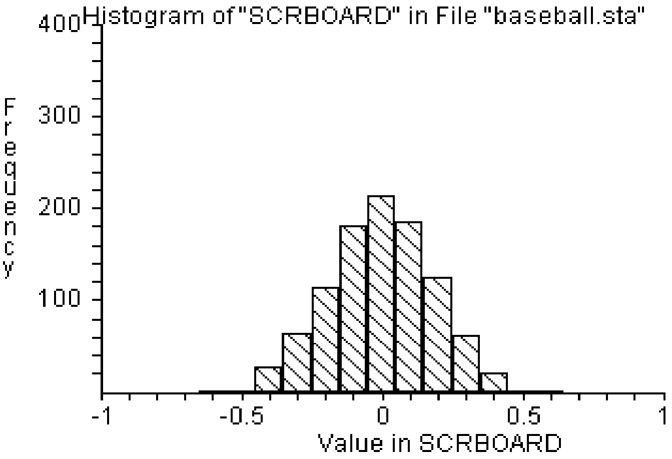


Figure 9: Correlation after shuffling

No shuffling produced a correlation coefficient as negative as the observed value, -.71, so we conclude that there is significant correlation between payroll and performance. The higher the payroll, the lower the rank number (i.e. the better the performance).

Baseball payroll—testing for correlation using the sum-of-products statistic

(program “basebal2”)

We can also conduct this test with a statistic other than the correlation coefficient, the calculation of which is not wholly transparent. We will use the “sum of products” statistic. (Interestingly, the multiplications used in the sum-of-products test also appear in the calculation of the Pearson correlation coefficient, which then goes on to scale them to between -1 and +1.)

Consider the array:

1

2

3

Let’s multiply it by another array which is arranged in the same order (lowest to highest):

1	**x**	**2**	**=**	**2**
2	**x**	**4**	**=**	**8**
3	**x**	**6**	**=**	**18**
		sum	**=**	**28**

Now rearrange the second array and perform the multiplication again. You will note that the sum of products is at its highest when the two arrays are in the same order. Conversely, you will note that the sum of products is at its lowest when the two arrays are in perfect opposite order.

To repeat: When there is perfect positive correlation (high numbers go with high, low numbers with low), the sum of products is at its highest. When there is perfect negative correlation (high with low, low with high), this sum of products is at its lowest.

Most often, though, when we randomly shuffle the second array, the sum of products will be neither at its highest nor at its lowest — it will be somewhere in the middle.

To test for positive correlation, we see how often random shuffling produces a sum of products as high as (or higher than) the observed sum of products. If it hardly ever does, then we can say that there is significant positive correlation.

To test for negative correlation, we see how often random shuffling produces a sum of products as low as (or lower than) the observed sum of products. If it hardly ever does, then we can say that there is significant negative correlation.

TABLE 4. BASEBALL PAYROLL AND RANK DATA WITH PRODUCTS 1995–1997

	Total Payroll	Rank	Product
NY Yankees	192.7	3	578.1
Baltimore	179.5	4	718
Atlanta	164.8	1	164.8
Cleveland	155.7	2	311.4
Chicago WS	150.3	14	2104.2
Cincinnati	143	9.5	1358.5
Texas	139.9	11	1538.9
Colorado	138.3	8	1106.4
Toronto	137.4	25	3435
St. Louis	137.3	19.5	2677.35
Seattle	137.1	6	822.6
Boston	131.8	7	922.6
Los Angeles	128.3	5	641.5
San Francisco	124	18	2232
Chicago Cubs	123	21	2583
Florida	122.8	12	1473.6
Anaheim	116	15.5	1798
Houston	115.4	9.5	1096.3
Philadelphia	109.9	26	2857.4
San Diego	104.5	13	1358.5
NY Mets	104.2	17	1771.4
Kansas City	101.1	22	2224.2
Minnesota	94.6	27	2554.2
Oakland	85.5	23.5	2009.25
Detroit	84	28	2352
Milwaukee	78.5	19.5	1530.75
Pittsburgh	67.7	23.5	1590.95
Montreal	67.6	15.5	1047.8
Sum of products:			44858.7

Is the observed sum of products, 44,858.7, lower than what might be obtained by random shuffling of the data as we did before? The same resampling procedure is used, except that we calculate the sum-of-products in step 3 instead of the correlation coefficient.

1. Write down all the payroll numbers on one set of cards, one number per card. Do likewise with the performance ranks.

2. Shuffle the "pay" cards (or both sets of cards—either procedure will achieve full randomization of one variable relative to the other), and deal out the "rank" cards along side the "pay" cards.

3. Calculate the sum-of-products of the shuffled arrays, and record.

4. Repeat steps 2-3 many times, say 1000 times.

5. Find out how often the shuffled correlation coefficient is as negative as or lower than the observed value of -.71.

Here is a RESAMPLING STATS program:

```
READ file “baseball.dat” pay rank
                              Read the data into two vectors, pay
                              and rank

REPEAT 1000

    SHUFFLE rank rank$        Shuffle the ranks, call the result rank$
                              (the “$” is often used to denote the
                              resampling counterpart to an
                              observed variable)

    MULTIPLY pay rank$ products
                              Multiply the pay vector by the
                              shuffled rank vector (resulting in 28
                              products, placed in a vector of that
                              name)

    SUM products sumprod      Sum those products

    SCORE sumprod scrboard    Keep score of the sum

    END

COUNT scrboard <= 44858.7 k   How often was the sum as low as or
                              lower than the observed sum?

DIVIDE k 1000 prob

PRINT prob
```

Result:

```
prob = .0001
```

Only one shuffling out of 1000 produced a sum of products as low as the observed value, so we conclude that there is significant correlation between payroll and performance. The higher the payroll, the lower the rank number (i.e. the better the performance).

Sample size

A useful application of the resampling method is in determining appropriate sample size.

EXAMPLE 1 (WITH A PROPORTION)

Consider the problem of a political candidate interested in taking a poll to learn her standing in the race. Suppose now that she has not yet taken the poll, and wants to determine how much to spend on it (i.e., how many people to query).

We will begin here with a trial-and-error approach, considering how much error there might be in a poll of 100 people. But what universe should we draw from? Continuing our trial-and-error approach, our candidate guesses that she is in a fairly tight race so we draw from a 50/50 universe to observe how samples from that universe behave. (Later we could modify that to draw samples from a different universe.) Here’s the program:

```
REPEAT 1000                 Do 1000 trials

    GENERATE 100 1,2 a      Generate 100 numbers (the sample of
                            voters) randomly selecting “1” or “2.”
                            We let 1 = favorable, 2 = unfavorable

    COUNT a =1 b            Count the number of favorables in the
                            sample

    SCORE b scrboard        Keep score

END                         End the trial, go back and repeat until
                            1000 are complete
```

Next, we want to measure how variable these results are. The following commands calculate the 5th and 95th percentile of the trial results:

```
PERCENTILE scrboard (5 95) interval

PRINT interval
```

We could then run the program again with a sample size of 500 instead of 100.

```
REPEAT 1000                 Do 1000 trials

    GENERATE 500 1,2 a      Generate 500 numbers (the sample of
                            voters) randomly selecting “1” or “2.”
                            We let 1 = favorable, 2 = unfavorable

    COUNT a =1 b            Count the number of favorables in the
                            sample

    SCORE b scrboard        Keep score

END                         End the trial, go back and repeat until
                            1000 are complete
```

```
PERCENTILE scrboard (5 95) interval
PRINT interval
```

Running the program 3 or 4 times may suffice for a quick estimate of the appropriate sample size. For more exhaustive estimating, the following "nested loop" procedure is suggested:

```
COPY 100 s                          Set the sample size indicator at 100
REPEAT 20                           Experiment with 20 different sample
                                    sizes
    ADD 20 s s                      Add 20 to the sample size (our first
                                    experiment will actually be with a
                                    sample size of 120)
    REPEAT 1000
        GENERATE s 1,2 a            Use "s" as the sample size in each set
                                    of 1000 trials

                                    HINT:  Note that GENERATE can
                                    accept a variable as its sample size.
                                    SAMPLE is the same way.
        COUNT a =1 b
        SCORE b scrboard
    END
    PERCENTILE scrboard (5 95) interval
                                    We calculate the percentiles for each
                                    of the 20 sample sizes
    DIVIDE interval s prop          Convert to a proportion so we can
                                    easily compare different sample sizes
    PRINT s                         Print the sample size
    PRINT prop                      Print the interval
    CLEAR scrboard                  Clear the scorekeeping vector used in
                                    the "inner" loop
END
```

HINT

Take care to **CLEAR** the scorekeeping vector used in a nested loop before beginning another iteration of the nested loop.

It is then easy to look at the intervals and sample sizes to make a judgement about whether the cost of additional sampling is worth the improved accuracy.

EXAMPLE 2 (WITH MEASURED DATA)

Consider again the feed merchant who is marketing a new pig ration. Seeing that the resamples of 12 produce considerable variability from sample to sample, he wants to know whether a larger sample would produce less variability. This would allow him to establish a higher bound for the minimum performance of his new product. We expect that it would, so the important question becomes how much an increase in sample size reduces variability. Let us try the same procedure again, only taking resamples of size 40 instead of size 12 from the original set of 12 observations:

```
DATA (496 544 464 416 512 560 608 544 480 466 512 496) a
```

Could also use the **COPY** command —
it's the same as **DATA**

```
REPEAT 1000
          SAMPLE 40 a b
          MEAN b bb
          SCORE bb scrboard
END
HISTOGRAM scrboard
```

In running this experiment, we found that 499 ounces was the cutoff point for a 90% one-sided confidence interval: 900 of our 1000 trials produced sample means at or above this level. This is an improvement over our result (491 ounces) with a re-sample size of 12. The merchant must weigh the costs of further sampling against the benefits of being able to narrow his confidence interval.

Unusual Statistics

(program "firings")

One of the advantages of resampling is its suitability for use with non-standard statistics. Here is an illustration of a statistic designed to meet the needs of a specific situation:

A company has been accused of firing workers (it has 50) when they get close to the level of seniority at which their pension would be vested (25 years). The union notes that the levels of seniority of 7 fired workers in the last 12 months were unusually close to 25 years.

Seniority at discharge (years):

23	19	24	23	25	2	5

Seniority of all workers:

11	8	24	36	20	19	11	9	10	9	5
4	2	1	9	21	16	17	11	1	1	23
19	24	40	28	5	7	1	34	20	16	31
23	50	4	1	8	8	14	12	32	1	15
12	25	19	5	24	2					

Note: A "25" indicates the worker's pension has vested.

The company counters that operational considerations were the only factors in each of the firings and that the proximity of the firing dates to pension vesting dates was purely coincidental, the result of random chance.

Can we assess whether this claim is reasonable?

To solve this problem, we need a measure of the degree to which firing dates cluster just below 25 years seniority. Here's one possible measure:

Let's subtract from 25 the tenure of each fired, unvested employee then sum them. The lower that sum, the more the firings cluster just below 25. What about the workers already vested? They are all equal evidence against the union's claim, and probably equivalent in meaning to a very junior worker being fired – whatever the reason for firing a very junior or a vested worker, we can be pretty confident that it is unrelated to pension vesting. So, we add to our sum 25 for each worker whose pension is vested. If the overall sum is low, that indicates the firings cluster just below 25.

To sum up, we subtract each worker's tenure from 25, convert all results <=0 to 25, then sum.

For the observed data, this sum is 79.

Resampling Procedure:

1. Put the tenures of the 50 workers in a hat.
2. Select 7 at random, without replacement.
3. Subtract each worker's tenure from 25, convert all results <=0 to 25, then sum. Record this sum.
4. Repeat steps 2-4 many times.
5. How often did we get a result <= the observed value of 79?

```
DATA (11 8 24 36 20 19 11 9 10 9 5 4 21 9 21 16 17 11 1 1 23 19
24 40 28 5 7 1 34 20 16 31 23 50 4 1 8 8 14 12 32 1 15 12 25 19 5
24 2) a

REPEAT 1000

    SHUFFLE a aa               Shuffle the data

    TAKE aa 1,7 b              Take the first 7 (note that aa remains
                               unchanged)

    SUBTRACT 25 b c            Subtract each of the 7 tenures from 25

    RECODE c <=0 25 d          Recode all non-positive results as 25

    SUM d e                    Find the sum

    SCORE e scrboard           Keep score

END

COUNT scrboard <= 79 k         How often did the resampled result
                               fall at or below the observed result of
                               79?

DIVIDE k 1000 prob

PRINT prob
```

Result:

```
prob = .11
```

The estimated p-value is .11, indicating that a sum as low as the observed value of 79 might happen 11% of the time, simply drawing workers at random. We conclude the evidence is not strong that there was systematic firing of those close to vesting.

Regression

The command **REGRESS** performs multiple linear regression. RESAMPLING STATS allows you to repeat the regression many times with randomized or bootstrapped values to assess the significance of your results.

Bootstrapping a regression – cases

(program "news")

We can also use resampling to establish confidence intervals for the coefficients and intercept of a regression equation. For example, we may wish to predict a city's weekend newspaper circulation on the basis of retail sales and population density. Consider these data:

city	circulation (000)	$ million retail sales	pop. / sq. mile
1	3.0	21.7	47.8
2	3.3	24.1	51.3
3	4.7	37.4	76.8
4	3.9	29.4	66.2
5	3.2	22.6	51.9
6	4.1	32.0	65.3
7	3.6	26.4	57.4
8	4.3	31.6	66.8
9	4.7	35.5	76.4
10	3.5	25.1	53.0
11	4.0	30.8	66.9
12	3.5	25.8	55.9
13	4.0	30.3	66.5
14	3.0	22.2	45.3
15	4.5	35.7	73.6
16	4.1	30.9	65.1
17	4.8	35.5	75.2
18	3.4	24.2	54.6
19	4.3	33.4	86.7
20	4.0	30.0	64.8
21	4.6	35.1	74.7
22	3.9	29.4	62.7
23	4.3	32.5	67.6
24	3.1	24.0	51.3
25	4.4	33.9	70.8

(Problem is from Terrell, Daniel, *Business Statistics*, 1975, Houghton Mifflin, p. 269)

A multiple linear regression of circulation in thousands (circ) on retail sales (sales) and population per square mile (pop) yields the following estimate of a linear relationship:

circ = .0575 (sales) + .0300 (pop) + .3446

Realizing that our estimate of this relationship may not be accurate, we wish to establish confidence intervals around the estimated coefficients for sales and pop, as well as the constant. In common sense terms, we ask ourselves "What would the estimate have been had we not picked these data points, but some others?" It is impractical to gather additional data, but, following the logic presented earlier in the section on the bootstrap, we can resample with replacement from the data set we do have. In doing so, we are letting our sample stand in as a proxy for the universe from which it was drawn. Sampling with replacement allows the sample to serve effectively as an infinite universe.

NOTE: "NOPRINT" OPTION

Each time you run a regression, RESAMPLING STATS automatically prints out such basic information as the number of observations involved. When you run repeated randomized or bootstrapped regressions, you may wish to omit the reporting of this information for ***each trial***. Otherwise, the program will be delayed as the results for each trial are shown on the screen. To eliminate this display, use the noprint option (see program below).

Maintaining case correspondence

When we bootstrap a regression relationship, of course, we must retain the correspondence between the observations for the different variables — we are sampling cases. In other words, we might or might not select Seattle for a particular re-sample (or we might select it twice or more). If we do select it though, we must select its circulation, sales and population data and keep them in a row so that Seattle sales always goes with Seattle circulation, etc. For this reason, we do not use the **SAMPLE** command — we would have no way of ensuring that it took the same elements in the same order from the circulation, sales and population data vectors.

Rather, we **GENERATE** 25 random numbers between 1 and 25 (with replacement, of course), and use those numbers to indicate the positions of data points to take from *each* of the data vectors.

In RESAMPLING STATS:

```
READ file "news" circc sales pop        Reads the data (first column
                                        into a vector "circ," second
                                        column into a vector "sales,"
                                        and third column into a vector
                                        "pop")
REGRESS circ sales pop a                Regress circulation on sales
                                        and population, yielding the
                                        vector "a" of coefficients and
                                        constant
REPEAT 1000
    GENERATE 25 1,25 b              Generate 25 numbers randomly
                                    between 1 and 25; these will be the
                                    positions of the records we take for
                                    this bootstrap resample
    TAKE circ b circ$               Take the circulation data points
                                    corresponding to the positions
                                    specified by "b"
    TAKE sales b sales$             Similarly for sales
    TAKE pop b pop$                 Similarly for population
    REGRESS noprint circ$ sales$ pop$ a$
                                    Regress the bootstrapped circulation
                                    data on the bootstrapped sales and
                                    population data
    SCORE a$ z1 z2 z3               Score the bootstrap regression result
                                    vector to "z1" (sales coef) "z2" (pop
                                    coef) and "z3" (constant)
END
PRINT a                             Print the actual regression estimate
PERCENTILE z1 (5 95) c1             Calculate the 5th and 95th percen tiles
                                    of the trial results for the sales
                                    coefficient
PERCENTILE z2 (5 95) c2             Similarly for the population
                                    coefficient
PERCENTILE z3 (5 95) c3             And the constant
PRINT c1 c2 c3
```

C1 and C2 are the confidence limits for the first and second coefficients, respectively, while C3 is the confidence limits for the constant.

Bootstrapping a regression – residuals

(program “cheese”)

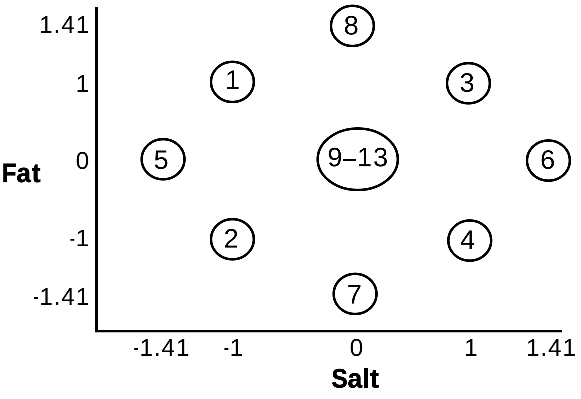
Bootstrapping cases makes sense when the observations can be thought of as a random sample from a population. Often this is not the case. Consider these hypothetical data, which are the results of a focus group studying how consumer acceptance of a cheese product is affected by relative content of salt and fat.

A food company experiments with different levels of salt and fat in a product (measured from a baseline that we’ll call “0 salt, 0 fat,” though there are positive levels of each). There are 13 different trials resulting in 13 observations (five of which are all at the baseline level). Data on the levels of salt & fat and on resulting consumer acceptance are in file “cheese.dat,” and below:

SALT	FAT	Consumer Acceptance
-1	1	4.2
-1	-1	2.8
1	1	7.4
1	-1	6.1
-1.41	0	3.4
1.41	0	6.6
0	-1.41	4.6
0	1.41	7
0	0	5.2
0	0	5.6
0	0	5.4
0	0	6.0
0	0	5.6

In the model used by the firm, product acceptance (PA) is regressed on Fat, Salt, F*S, Fat squared, and Salt squared.

Clearly the salt and fat content values are not randomly chosen; in fact they were carefully selected to yield good baseline information plus maximum information about varying levels of salt and fat, while minimizing costs. Looked at on a two-dimensional plot, the various combinations of salt and fat form a circle, with the 5 baseline observations in the center.



We want to concentrate on the random component here and bootstrap that. We can think of each PA value as comprised of an underlying component (which is a function of the chosen fat and salt level), and a random component or error term:

$$\begin{tikzcd}
\text{Obs. Value} \; = & \text{underlying value} \arrow[d] & + & \varepsilon \arrow[d] \\
 & \text{use predicted value} \arrow[d] & & \text{use bootstrapped residual} \arrow[d] \\
\text{Bootstrapped value} = & \hat{y} & + & \varepsilon^*
\end{tikzcd}$$

While we don’t know what the underlying value is, we can estimate it through our regression (y). And the residuals from the regression will be our estimate of the random component. Our procedure is, therefore:

1. Find residuals from original regression

2. Take a bootstrap sample from them, add to the predicted values to create a bootstrap sample of y-values

3. Re-run the regression with the bootstrap sample of y-values, record the new coefficients

4. Repeat 2-3 many times

5. Take appropriate percentiles of the recorded bootstrap coefficients

```
READ file “cheese.dat” salt fat accept
          ‘First we need to read in the data, and create the various
          interaction and squared terms

MULTIPLY salt fat saltfat

SQUARE salt saltsq

SQUARE fat fatsq

REGRESS accept salt fat saltfat saltsq fatsq model
          ‘Calculate the original regression; “model” is a vector holding
          the coefficients for salt, fat, salt*fat salt squared, and fat squared.

          ‘Next create Resampling Stats vector names for the coefficients in
          “model”:

TAKE model 1 b1

TAKE model 2 b2

TAKE model 3 b3

TAKE model 4 b4

TAKE model 5 b5

TAKE model 6 b0

LET predict = b1*salt+b2*fat+b3*saltfat+b4*saltsq+b5*fatsq+b0
          ‘Find the predicted values and, next, the residuals
```

```
LET resid = accept-predict
REPEAT 1000
    SAMPLE 13 resid resid$         'bootstrap the residuals
    ADD resid$ predict accept$
                  'add them to the predicted values to create a new vector
                  "accept$" of bootstrap Product Acceptance scores
    REGRESS noprint accept$ salt fat saltfat saltsq fatsq
    model$
                  'Regress the bootstrapped Product Acceptance scores on
                  the original values for salt, fat, etc.; use the noprint option
                  to suppress screen output in the repeat loop
    SCORE model$ b1 b2 b12 b11 b22 constant
                  'keep score of the bootstrapped coefficients "b1" (salt),
                  "b2" (fat), etc.
END
PERCENTILE b1 (5 95) c1
                  'Find the the intervals enclosing 90% of the bootstrapped
                  coefficients for salt, fat, etc.
PERCENTILE b2 (5 95) c2
PERCENTILE b12 (5 95) c12
PERCENTILE b11 (5 95) c11
PERCENTILE b22 (5 95) c22
PERCENTILE constant (5 95) const
PRINT model c1 c2 c12 c11 c22 const
                  'Print the original regression coefficients, then the intervals
```

NOTE

You may need to add a "MAXSIZE model$ 10000" statement to the beginning if you encounter "Vector maximum size exceeded" errors on your computer.

Results:

```
MODEL  =  1.3806  0.76277  -0.025  -0.35092  0.051472
5.5607
```

(= coefficients and constant in the original regression)

```
c1     =  1.2184      1.5388
c2     =  0.59429     0.92848
c12    =  -0.2686     0.20648
c11    =  -0.53587    -0.17147
c22    =  -0.11535    0.23294
const  =  5.34        5.77
```

Conclusion: The intervals for salt (c1) and fat (c2) are significantly positive. For salt-squared (c11) it is significantly negative. For the interaction term (c12) and for the fat-squared term (c22) the intervals span zero.

Serial correlation in a time series

Consider the following data on quarterly GNP growth and the following question:

Are the points serially correlated? Is a high likely to be followed by a high and a low by a low?

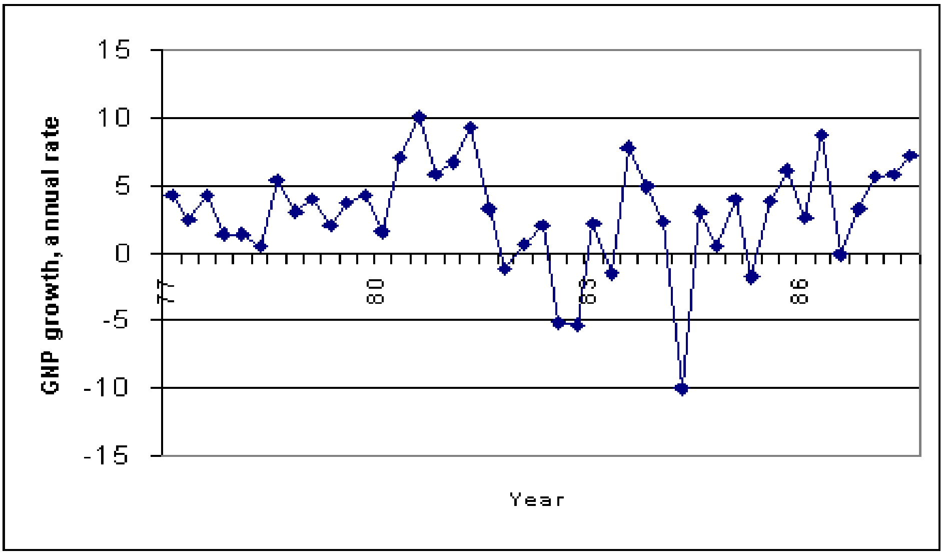


Figure 9: Quarterly U.S. GNP growth at annual rates

Serial correlation

(program "bus-cycl")

To answer this question, we will use the same "sum of products" technique we developed earlier to deal with correlation. In this case, we are wondering whether points are correlated with their neighbors. As before, we will use the fact that, when an array of numbers is multiplied by another array in the same order (high numbers matching high numbers and vice versa), the sum of the products is higher than when they are rearranged in any other order. In this case, we will be multiplying observations by their neighbors.

Therefore, the data are arranged (in a spreadsheet) in two columns such that each number is opposite the number that precedes it:

TABLE 8.

a_2	a_1
a_3	a_2
a_4	a_3
.	.
.	.
.	.
a_n	a_{n-1}

> (The same effect could be achieved with RESAMPLING STATS by creating a **COPY** of our original data vector then **CONCAT**enating a dummy number onto the end of the original and the beginning of the copy. Here, if we use 0 as a dummy, we can then **MULTIPLY** the two vectors and **WEED** out the zeroes, leaving us the product of each element and the one that follows.)

First, we note the sum of the observed products. Then, after **READ**ing the data into RESAMPLING STATS, we repeatedly reshuffle one of the vectors, keeping score of the sum of the shuffled products. If they are always less than the observed sum, we can be confident that the apparent serial correlation did not occur by chance.

In RESAMPLING STATS:

```
READ file "gnp.dat" gnp gnplag
                              Read our data file "gnp.dat" into
                              vectors called "gnp" and "gnplag"
                              (which is "gnp" lagged by one period)
MULTIPLY gnp gnplag q         Multiply the "gnp" vector by the
                              "lagged gnp" vector
SUM q qsum                    Sum those products
REPEAT 1000                   Repeat the following experiment 1000
                              times
    SHUFFLE gnplag gnplag$    Shuffle the "lagged gnp" vector
    MULTIPLY gnp gnplag$ q$
                              Multiply the original "gnp" vector by
                              a shuffled "lagged" vector
    SUM q$ qsum$              Sum those products
```

SUBTRACT qsum qsum$ result	
	Subtract the trial sum of products from the actual sum of products. If there is serial correlation in the data, the actual sum should exceed nearly all of the random sums
SCORE result scrboard	Keep score of the result
END	
HISTOGRAM scrboard	Produce a histogram of the amounts by which actual sums exceed trial sums

From the histogram (Figure 10), we can easily see how often the observed sum of products exceeds the randomized sums of products.

Most of the time, the actual sum of products was higher than the trial sum of products. However, 8% of our chance trials (at and to the left of 0), produced sums of products as high as or higher than the *observed result*. Here our estimated "p-value" is .08. We should be wary of drawing any conclusions, though we might consider the matter worthy of further investigation.

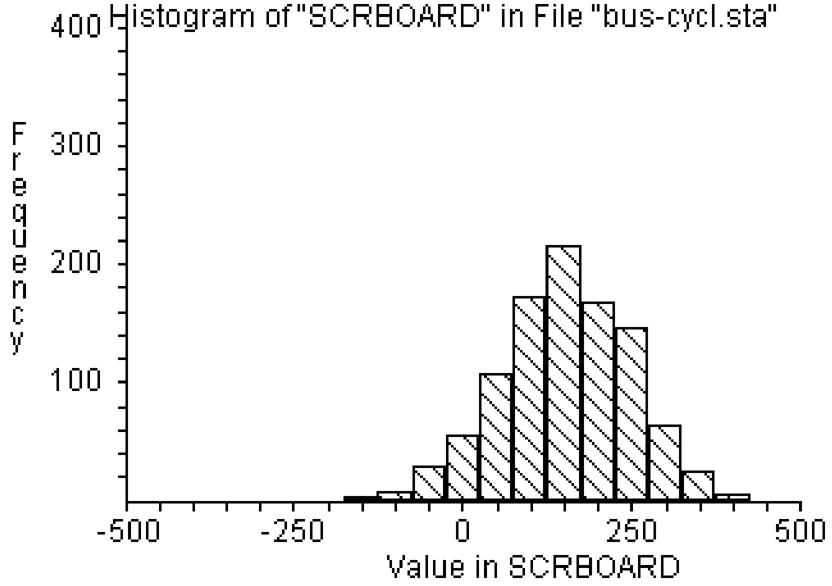


Figure 10: Sum of products of quarterly growth rates and lagged growth rates, shuffling the lagged rates, 1000 simulations