

MATH 615: Approaches to Quantitative Analysis in the Life Sciences

Class 10

Non-parametric statistics — resampling, randomization and simulation

Packages

Introduction

In statistics, the term non-parametric statistics covers a range of topics:

- Distribution free methods which do not rely on assumptions that the data are drawn from a given probability distribution. As such it is the opposite of parametric statistics. It includes non-parametric statistical models, inference and statistical tests.
- Non-parametric statistic can refer to a statistic (a function on a sample) whose interpretation does not depend on the population fitting any parametrized distributions. Statistics based on the *ranks* of observations are one example of such statistics and these play a central role in many non-parametric approaches.

Traditional non-parametric statistics

■ Example 1: Mann-Whitney U test

Tests whether two samples come from the same distribution. It is therefore analogous to a simple one-way ANOVA with two factors (treatments), or a T -test. The regular ANOVA assumes that the data in each sample come from a normal distribution. The Mann-Whitney test does not. In fact, the data do not even have to be numbers, but can be *ranks*. The core of the test is the principle that if the two samples *are* from the same distribution (the null hypothesis), then the probability of a random value from one group being ranked higher than a random value from the other group will be 0.5.

Mandible (tooth) lengths of golden jackals from the British Museum (Natural History).

```
In[109]:= males = {120, 107, 110, 116, 114, 111, 113, 117, 114, 112};  
          females = {110, 111, 107, 108, 110, 105, 107, 106, 111, 111};
```

```
In[113]:= data = Join[Transpose[{males, Table["M", {Length[males]}]}],
  Transpose[{females, Table["F", {Length[females]}]}]];
TableForm[
  data]
```

Out[114]/TableForm=

```
120 M
107 M
110 M
116 M
114 M
111 M
113 M
117 M
114 M
112 M
110 F
111 F
107 F
108 F
110 F
105 F
107 F
106 F
111 F
111 F
```

Rank the data

```
In[117]:= TableForm[data = MapThread[Append[#1, #2] &, {Sort[data], Range[Length[data]}]]]
```

Out[117]/TableForm=

```
105 F 1
106 F 2
107 F 3
107 F 4
107 M 5
108 F 6
110 F 7
110 F 8
110 M 9
111 F 10
111 F 11
111 F 12
111 M 13
112 M 14
113 M 15
114 M 16
114 M 17
116 M 18
117 M 19
120 M 20
```

Resort into groups, keeping only rank

```
In[118]:= TableForm[data = Sort[data[[All, {2, 3}]]]]
```

```
Out[118]//TableForm=
```

```
F 1
F 2
F 3
F 4
F 6
F 7
F 8
F 10
F 11
F 12
M 5
M 9
M 13
M 14
M 15
M 16
M 17
M 18
M 19
M 20
```

Choose the sample for which the average rank is smaller (females), and for each observation that sample, count the number of observations in the other sample that are smaller than it (and count 0.5 for those that are equal).

```
Out[128]= {0, 0, 0, 0, 1, 1, 1, 2, 2, 2}
```

The sum of these is U .

```
Out[126]= 9
```

For large samples, under the null hypothesis, U is normally distributed with mean

$$m_U = \frac{n_1 n_2}{2}$$

and standard deviation

$$\sigma_U = \sqrt{\frac{n_1 n_2 (n_1 + n_2 + 1)}{12}}$$

which means that our calculated U can be tested for significance. For our data, $n_1 = n_2 = 10$, so $m_U = 50$ and $\sigma_U = 13.23$. The probability of getting value of 9 or smaller from a Normal distribution with these parameters is

```
Out[131]= 0.000970818
```

Note that the method relaxes the assumption of Normality of the data, making it more flexible, but still uses a known distribution to test the statistic. There are a large class of such non-parametric tests. The next sections will examine non-parametric methods that do not even require that kind of assumption.

Confidence intervals

Bias revisited

This is the formula for a standard deviation:

$$\sigma = \sqrt{\frac{\sum (x_i - \bar{x})^2}{n}}$$

where the x_i are the observations, \bar{x} is the mean of the observations, and n is the number of observations (the sample size). It's the square root of the sum of squared deviations divided by the sample size. This formula applies if what you want to know is just the standard deviation of some data — for example if your observations are complete, rather than being a sample — and is also called the maximum likelihood estimator, or MLE, of σ .

Here's the *Mathematica* code.

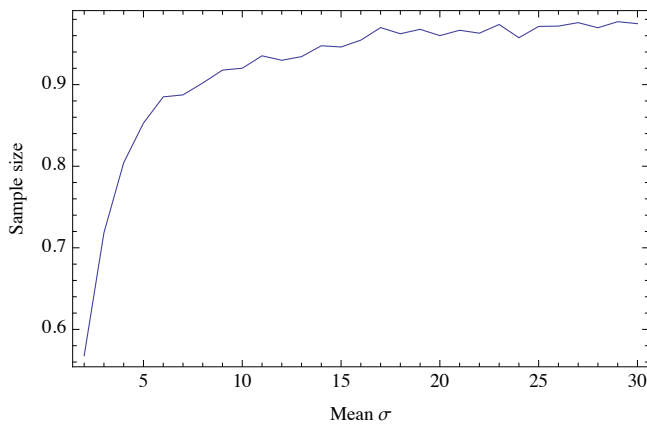
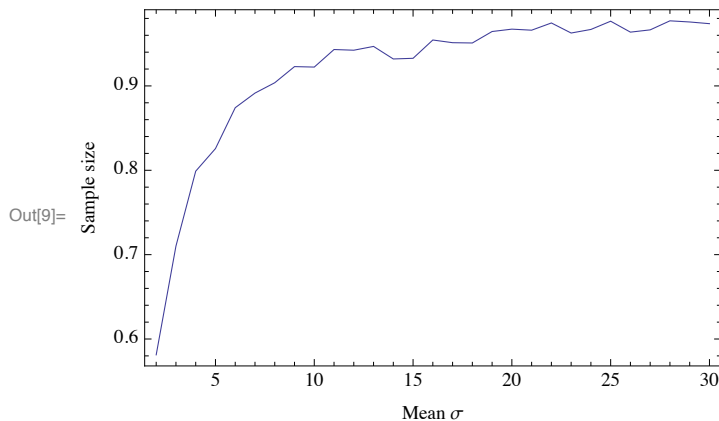
```
In[7]:= stdMLE[data_] := Sqrt[Total[(data - Mean[data]) ^ 2] / Length[data]]
```

Here we generate sets of random numbers from a Normal(0,1) distribution. We generate sets of 2, 3, ..., 30 numbers, and calculate σ using the formula above. We do it 1000 times, and calculate the average of the σ values for each sample size.

```
In[8]:= stdMLEEstimates = Table[
  {n, Mean[Table[stdMLE[RandomReal[NormalDistribution[0, 1], n]], {1000}]}], {n, 2, 30}];
```

This is what they look like.

```
In[9]:= ListLinePlot[stdMLEEstimates, PlotRange -> All,
  Axes -> False, Frame -> True, FrameLabel -> {"Mean  $\sigma$ ", "Sample size"}]
```



We know the true value of σ ($\sigma = 1$), and so you can see that the calculated value is always less than that, and for small sample sizes, it can be much less. So while the formula is correctly calculating the standard deviation of each *sample* we created, it is underestimating the standard deviation of the *population* that our samples were drawn from. The standard deviation MLE is a *biased* estimator of the population standard deviation (meaning that it deviates systematically in a particular direction).

So, assuming that what we want to do is estimate the standard deviation of an (unknown) population, based on a sample, then we should try to correct for the bias at small sample sizes.

This is the *bias-corrected* formula for *estimating* a standard deviation:

$$\sigma = \sqrt{\frac{\sum (x_i - \bar{x})^2}{n - 1}}$$

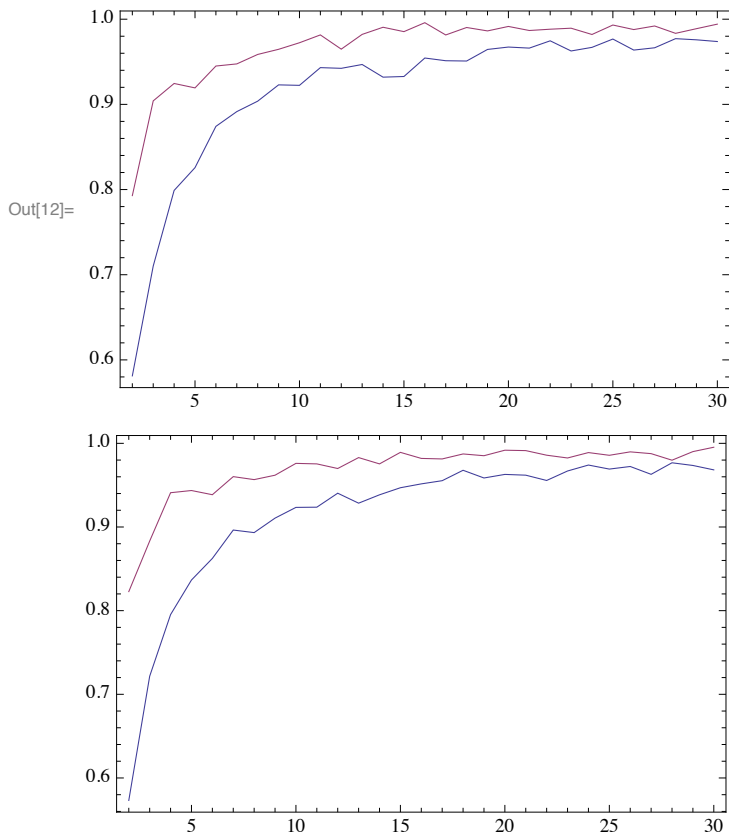
It's a small change — let's see how it does. Here is the *Mathematica* code:

```
In[10]:= stdC[data_] := Sqrt[Total[(data - Mean[data])^2] / (Length[data] - 1)]
```

We calculate a new set of values and plot them.

```
In[11]:= stdCEstimates =
  Table[{n, Mean[Table[stdC[RandomReal[NormalDistribution[0, 1], n]], {1000}]}], {n, 2, 30}];
```

```
In[12]:= ListLinePlot[{stdMLEEstimates, stdCEstimates}, PlotRange -> All, Axes -> False, Frame -> True]
```



You can see that the bias of the new estimator (the red line) is much reduced compared to the original estimator.

I'm explaining the concept of bias here because the next methods are ways to reduce it. The original standard deviation formula is based on the formula for a Normal distribution, but can be used on any set of data. However, the bias-corrected formula *assumes* a Normal distribution, and if your data are not normally distributed, then the correction won't work properly. The methods below are *non-parametric*, in that they can be used to reduce the bias in data without assuming a particular underlying distribution.

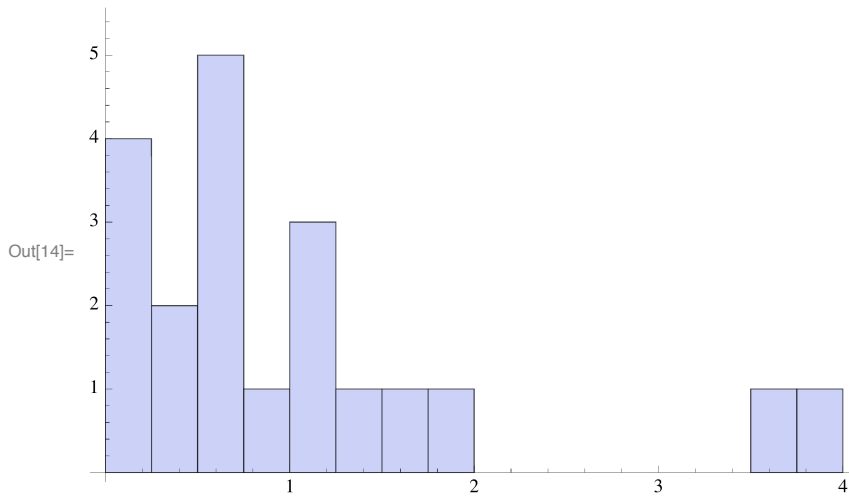
Resampling methods — Bootstrap and Jackknife

■ Example data

Here are some data (taken from the example in Chapter 2 of Manley 1997). They come from an exponential distribution with parameter $\lambda = 1$ (which therefore has a true mean and standard deviation of 1).

```
In[13]:= data = {3.56, 0.69, 0.10, 1.84, 3.93, 1.25, 0.18, 1.13,
                0.27, 0.50, 0.67, 0.01, 0.61, 0.82, 1.70, 0.39, 0.11, 1.20, 1.21, 0.72};
```

```
In[14]:= Histogram[data, {0, 4, 0.25}]
```



We know how to calculate the mean and standard deviation of the data, both in bias-corrected and uncorrected form

```
In[15]:= mean = Mean[data]
```

```
Out[15]= 1.0445
```

```
In[16]:= StandardDeviation[data] (* bias-corrected *)
```

```
Out[16]= 1.05968
```

```
In[17]:= stdMLE[data] (* uncorrected *)
```

```
Out[17]= 1.03285
```

■ The Jackknife

We also know how to calculate the standard error of the mean

```
In[18]:= n = Length[data];
          standardErrorOfMean = StandardDeviation[data] / Sqrt[n]
```

```
Out[19]= 0.236952
```

```
In[20]:= standardErrorOfStandardDeviation = StandardDeviation[data] / Sqrt[2 * n]
```

```
Out[20]= 0.16755
```

from which we can calculate 95% or other confidence intervals. But what if we are interested instead in the *standard deviation* of the data? We know how to calculate it, with either a bias-corrected or uncorrected formula, but what about a confidence on that estimate? There actually is a formula for the standard error of the standard deviation under conditions of normality, but let's ignore that for now. In 1958, Tukey proposed another way.

Let's suppose we remove one of our data points and recalculate the mean of the observations from the remaining observations. We'll remove the first one, the value of which is

```
Out[21]= 3.56
```

The new mean is

```
Out[22]= 0.912105
```

If we know the number of original observations, and these two means, we can reconstruct what the missing data point is:

```
In[23]:= Mean[data] * n - Mean[Rest[data]] * (n - 1)
```

```
Out[23]= 3.56
```

We could do this of each data point, reconstructing each one from the overall mean and the mean with that point missing. Since we know what the values are anyway, it's not a very useful exercise! But here's a trick. Suppose that instead of the mean, we were interested in the standard deviation of the data. Let's do the 'remove-one' exercise again, this time calculating the standard deviation (*without* bias correction). Here again is the case of dropping the first observation:

```
In[24]:= stdMLE[data] * n - stdMLE[Rest[data]] * (n - 1)
```

```
Out[24]= 3.95907
```

This is not one of the original data points any more, but a new 'pseudovalue' that is an independent estimate of the *standard deviation*. So we do this for all the data points, dropping out each one in turn.

```
Out[25]= {3.95907, 0.58556, 0.971105, 0.839844, 5.20211, 0.544031,
          0.897727, 0.526677, 0.823214, 0.670842, 0.592825, 1.06179, 0.617033,
          0.548094, 0.7376, 0.736943, 0.961562, 0.535059, 0.536654, 0.575414}
```

So now we can estimate the standard deviation by calculating the *mean* of these pseudovalues, and the standard error of the estimate in the same way that we calculate the standard error of a mean.

```
In[26]:= standardDeviationEstimate = Mean[pseudovalues]
```

```
Out[26]= 1.09616
```

```
In[27]:= standardErrorOfStandardDeviation = StandardDeviation[pseudovalues] / Sqrt[n]
```

```
Out[27]= 0.272804
```

These are called **Jackknife estimates**. As well as providing an estimate of the standard error of the standard deviation, our new estimated standard deviation is closer to the bias-corrected estimate (using the exact formula) than the uncorrected estimate. It turns out that the Jackknife procedure also corrects bias (up to a certain amount). It can be used, in principle, on any kind of statistic that we might calculate from a dataset.

■ The bottom line

- The jackknife method can potentially be used to bias correct, and obtain confidence intervals for, *any kind of statistic*.
- It essentially converts the problem of estimating the standard error of a statistic into one of estimating the standard error of a mean (which is easy).
- It is most useful for those statistics for which analytical formulas are complicated or unavailable.

■ The Bootstrap

Problems of inaccuracy of estimates, and bias, stem from the fact that our data are almost always a sample of a larger population. If we knew how our sample differed from the full population, we could correct the errors. Is there a way to do that without simply sampling the entire population? Efron (1979) proposed a simple solution — the best information we have about the population is our sample, so we could sample data *from our sample*, and examine how our sample-of-a-sample differs from the sample. This difference will, under many circumstances, be similar to the difference between the sample itself and the population.

To create a sample-of-a-sample, we sample *with replacement* from our data. (In other words, we treat our data as if it were a much larger sample than it actually is, and therefore closer to the size of the population.) Let's do it for our example data, focussing again on the standard deviation. After randomly sampling n points from the distribution, we calculate the (uncorrected) standard deviation, and do that a veyr large number of times.

```
In[28]:= bootstrapValues = Table[stdMLE[RandomChoice[data, Length[data]]], {10 000}];
```

Here is the mean of our bootstrap estimates...

```
Out[29]= 0.977715
```

...and here is the standard deviation, which is a measure of the standard error of the mean.

```
Out[30]= 0.244303
```

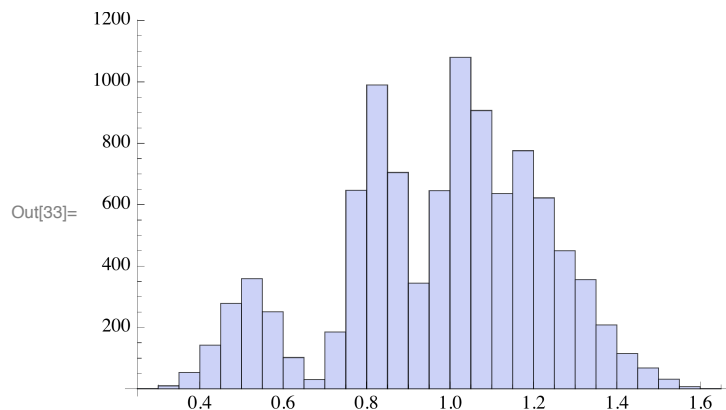
Note that the mean of the bootstrap standard deviations is lower than the standard deviation of the original data (1.033). That's because standard deviations are biased downwards! The *difference* is an *estimate of the bias*.

```
Out[31]= -0.0551329
```

We therefore correct our original estimate of the standard deviation by subtracting the bias

```
Out[32]= 1.08798
```

Of we look at the distribution of all 10,000 bootstrap estimates, it is not really normally distributed, so we might think about getting confidence intervals from the 2.5% and 97.5% qunatiles of the bootstrap distribution.



The procedure is actually a bit tricky, because the bootstrap values are biased downwards, so we have to correct for that. Here is one method, as described in Manly, 1996:

```
In[34]:= p = N[Length[Select[bootstrapValues, # > stdMLE[data] &]] / Length[bootstrapValues]]
```

```
Out[34]= 0.4547
```

```
In[35]:= z0 = x /. Solve[CDF[NormalDistribution[0, 1], x] == (1 - p), x][[1]]
```

```
Out[35]= 0.113795
```

```
In[36]:= 2 * z0 - 1.96
```

```
Out[36]= -1.73241
```

```
In[37]:= a1 = CDF[NormalDistribution[0, 1], (2 * z0 - 1.96)]
         a2 = CDF[NormalDistribution[0, 1], (2 * z0 + 1.96)]
```

```
Out[37]= 0.0416004
```

```
Out[38]= 0.98565
```

```
In[39]:= Quantile[bootstrapValues, {a1, a2}]
```

```
Out[39]= {0.488974, 1.42885}
```

■ The bottom line

- The bootstrap method can potentially be used to bias correct, and obtain confidence intervals for, *any kind of statistic*.
- It essentially converts the problem of estimating the standard error of a statistic into one of estimating the standard deviation of the resampled statistic (which is easy).
- It is most useful for those statistics for which analytical formulas are complicated or unavailable.

■ Example: estimating species richness

The previous example, of estimating the standard error of a standard deviation and correcting its bias, is not very compelling because a) there is a formula for the standard error of a standard deviation, and b) there exists a bias-corrected formula for the standard deviation, both of which apply under certain conditions. However, we may be confronted with a statistic for which formulas for standard errors and bias don't exist, perhaps because they are complicated and therefore extremely difficult to obtain. This is often the case in ecology.

Consider the problem of estimating the number of species in an area. In the field, you sample individuals, identify them to species, and then count how many species you have altogether. The problem is that many species are rare, and so you are likely to have missed a few of the rarest. And you can't count more species than there are, therefore, estimates of species of rich are *biased downwards* — they are *underestimates* of the true species richness.

The degree of bias obviously has to do with how many rare species there are, and we have some information about that from our sample (in which some will be abundant and some rare). Here is an example dataset: the abundances of seeds of different species in a soil sample.

```
In[40]:= seedBankData = {209, 109, 91, 75, 66, 49, 44, 43, 32, 20, 29,
                        15, 23, 11, 14, 11, 14, 14, 12, 4, 9, 10, 8, 7, 4, 6, 6, 4, 3, 4, 2, 2, 1, 1};
```

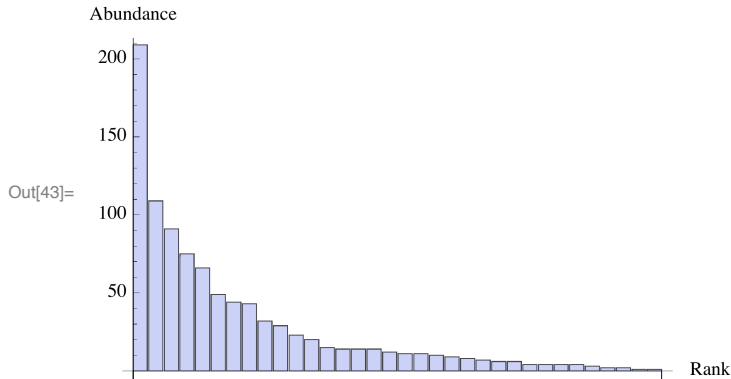
The total number of individuals sampled was

```
Out[41]= 952
```

and the total number of species in the sample was

```
Out[42]= 34
```

Here we plot the abundances in order from largest to smallest — a *rank-abundance plot*.



In the bootstrap approach, we assume the abundance distribution of our sample is representative of the true abundance distribution, and we resample the same total number of seeds (with replacement) from our sample distribution. Here is one example of a sample:

```
In[44]:= bootstrapSample =
RandomChoice[seedBankData → Range[1, Length[seedBankData]], Total[seedBankData]]

Out[44]= {1, 9, 6, 18, 3, 1, 4, 1, 21, 1, 4, 9, 4, 1, 3, 1, 2, 1, 13, 1, 7, 2, 3, 18, 10, 1, 4, 1, 7, 1, 7, 7, 19,
1, 5, 29, 18, 16, 8, 1, 1, 13, 7, 1, 1, 1, 7, 1, 1, 22, 2, 5, 4, 4, 4, 1, 26, 8, 5, 6, 2, 1, 22,
1, 17, 23, 18, 5, 15, 6, 18, 9, 30, 9, 16, 3, 13, 5, 2, 5, 17, 11, 4, 2, 1, 5, 3, 3, 27, 2, 1,
7, 18, 1, 2, 1, 1, 8, 7, 1, 10, 1, 5, 10, 11, 14, 3, 5, 1, 9, 8, 10, 1, 10, 1, 14, 3, 16, 24,
4, 3, 10, 7, 31, 3, 1, 2, 3, 6, 3, 1, 7, 1, 5, 4, 6, 3, 1, 1, 21, 2, 3, 9, 5, 1, 19, 5, 13, 5,
8, 5, 6, 6, 4, 1, 6, 9, 2, 2, 3, 6, 3, 1, 1, 2, 1, 23, 1, 2, 4, 2, 8, 1, 9, 31, 10, 2, 8, 3, 2,
5, 4, 13, 1, 9, 3, 1, 3, 4, 8, 1, 2, 1, 7, 1, 7, 1, 4, 1, 2, 2, 3, 9, 8, 3, 1, 1, 2, 3, 1, 6, 22,
24, 17, 5, 14, 1, 9, 6, 18, 1, 3, 2, 2, 1, 11, 1, 1, 11, 16, 16, 9, 29, 4, 7, 5, 1, 1, 3, 7, 3,
8, 1, 19, 7, 4, 1, 23, 1, 13, 1, 22, 9, 15, 26, 5, 6, 2, 6, 3, 1, 2, 1, 2, 3, 15, 17, 1, 10, 4,
3, 7, 3, 1, 16, 2, 6, 2, 3, 1, 6, 9, 4, 9, 1, 5, 1, 2, 29, 3, 5, 16, 2, 8, 1, 4, 3, 3, 1, 5, 7,
1, 1, 30, 11, 2, 4, 31, 15, 2, 8, 6, 9, 3, 1, 4, 31, 1, 9, 1, 11, 1, 5, 2, 2, 11, 2, 5, 7, 3,
2, 8, 8, 1, 6, 4, 1, 13, 1, 3, 3, 5, 6, 2, 5, 6, 3, 3, 4, 2, 4, 6, 5, 16, 1, 5, 1, 4, 2, 6, 4,
32, 1, 2, 1, 18, 2, 11, 1, 1, 2, 24, 1, 1, 1, 18, 4, 5, 1, 8, 1, 7, 3, 6, 8, 11, 21, 2, 1, 4,
8, 3, 1, 3, 2, 5, 8, 22, 5, 2, 3, 1, 1, 16, 7, 2, 7, 26, 5, 2, 15, 1, 5, 3, 3, 1, 8, 6, 1, 5,
2, 3, 1, 30, 1, 1, 11, 4, 24, 2, 1, 5, 8, 4, 1, 2, 7, 7, 2, 4, 2, 4, 2, 2, 2, 2, 3, 7, 7, 1, 4,
2, 5, 7, 1, 13, 3, 22, 4, 2, 3, 5, 2, 13, 1, 4, 8, 7, 7, 22, 2, 7, 4, 8, 9, 1, 8, 22, 14, 5, 5,
1, 2, 6, 1, 32, 2, 3, 4, 1, 23, 4, 12, 2, 23, 1, 1, 9, 30, 3, 2, 2, 7, 2, 5, 1, 7, 5, 1, 17, 1,
8, 1, 6, 12, 6, 9, 1, 1, 8, 2, 1, 5, 2, 28, 4, 1, 2, 5, 8, 10, 12, 26, 2, 14, 3, 10, 1, 5, 9,
23, 1, 15, 2, 1, 1, 6, 7, 24, 3, 2, 24, 16, 1, 3, 21, 19, 13, 5, 1, 1, 15, 19, 6, 16, 3, 23,
20, 5, 13, 15, 8, 1, 8, 2, 2, 1, 17, 1, 4, 1, 3, 1, 5, 2, 18, 1, 6, 4, 13, 6, 4, 3, 19, 4, 2,
15, 2, 8, 21, 6, 12, 1, 15, 21, 5, 8, 23, 19, 33, 1, 1, 1, 11, 11, 1, 3, 1, 3, 15, 1, 4, 1, 14,
11, 1, 2, 2, 5, 3, 4, 1, 5, 2, 8, 11, 1, 2, 7, 1, 5, 11, 23, 6, 1, 4, 6, 1, 3, 9, 12, 2, 13, 4,
10, 1, 1, 5, 1, 5, 1, 6, 29, 9, 1, 2, 9, 4, 2, 1, 6, 2, 1, 1, 7, 11, 1, 2, 7, 28, 19, 5, 1, 23,
6, 5, 1, 3, 1, 1, 2, 19, 2, 1, 2, 3, 1, 2, 6, 5, 1, 18, 2, 3, 27, 2, 3, 19, 15, 2, 8, 5, 5, 4,
4, 1, 18, 1, 1, 17, 1, 3, 19, 4, 5, 6, 1, 2, 7, 8, 2, 2, 17, 2, 4, 13, 1, 1, 1, 8, 5, 13, 9, 3,
8, 3, 6, 10, 14, 4, 5, 9, 1, 1, 3, 2, 4, 7, 28, 2, 5, 1, 7, 1, 14, 3, 1, 25, 4, 4, 6, 14, 1, 1,
1, 8, 1, 1, 24, 2, 3, 1, 19, 4, 7, 6, 1, 5, 13, 14, 1, 1, 4, 4, 6, 2, 14, 1, 2, 1, 3, 2, 15, 13,
1, 5, 3, 7, 4, 1, 16, 8, 1, 3, 1, 22, 1, 3, 1, 2, 4, 1, 8, 1, 2, 3, 2, 7, 4, 24, 6, 1, 6, 8, 5,
9, 1, 15, 19, 3, 1, 6, 10, 30, 6, 1, 4, 22, 7, 2, 4, 1, 2, 8, 4, 2, 6, 3, 5, 3, 16, 4, 14, 1,
7, 8, 18, 10, 12, 3, 4, 1, 1, 3, 9, 11, 19, 5, 1, 29, 2, 1, 2, 4, 5, 24, 29, 9, 4, 2, 5, 17, 9,
3, 2, 1, 2, 3, 2, 1, 3, 4, 1, 1, 7, 6, 5, 4, 2, 1, 5, 18, 2, 1, 1, 6, 13, 10, 3, 9, 6, 6, 12,
3, 7, 5, 7, 5, 1, 17, 8, 2, 26, 1, 1, 4, 3, 5, 5, 23, 4, 16, 2, 17, 4, 1, 27, 21, 2, 13, 8, 2}
```

For this sample, we calculate number of different species it contains.

```
Out[45]= 33
```

We repeat this process many times. Here are 100 bootstrap species richnesses

```
Out[46]= {32, 34, 32, 33, 33, 34, 33, 33, 33, 33, 33, 32, 34, 33, 34, 33, 32, 32, 33, 32, 32, 33, 32, 32, 34, 33,
          32, 32, 33, 32, 34, 32, 34, 33, 34, 33, 33, 33, 32, 34, 32, 34, 31, 34, 33, 34, 32, 34, 33, 32, 32,
          34, 33, 33, 33, 32, 34, 33, 34, 30, 33, 32, 34, 34, 32, 33, 34, 34, 34, 32, 32, 33, 32, 34, 33, 34,
          34, 33, 31, 34, 33, 33, 31, 34, 33, 34, 33, 33, 33, 33, 33, 34, 33, 33, 33, 33, 34, 32, 34, 34}
```

Actually, we will do 10,000 and calculate the mean

```
Out[48]= 32.8623
```

This is different than the full sample richness of 34 species by

```
Out[49]= -1.1377
```

This is an estimate of the bias in our original sample, so our bootstrap bias-corrected estimate of the true species richness is

```
Out[58]= 35.1377
```

Confidence intervals of the richness estimate

```
Out[59]= {33.2754, 36.2754}
```

■ The uses of the methods

These methods are most commonly used for correcting bias and estimating confidence intervals, but they can also be used for hypothesis testing in some simple circumstances. For example, if we have two samples, and we calculate the confidence intervals around the means (using any method), we can test whether they are likely to have come from the same population or not based on the overlap of their confidence distributions. We could also get confidence intervals on the slope parameter of a simple linear regression, and use these to decide if the slope is significantly different from zero. This only works for simple tests.

Randomization

■ Data 1

Mandible (tooth) lengths of golden jackals from the British Museum (Natural History).

```
In[61]= males = {120, 107, 110, 116, 114, 111, 113, 117, 114, 112};
          females = {110, 111, 107, 108, 110, 105, 107, 106, 111, 111};
```

```
In[63]:= data = Join[Transpose[{Table["M", {Length[males]}], males}],
  Transpose[{Table["F", {Length[females]}], females}]];
TableForm[
  data]
```

Out[64]//TableForm=

```
M 120
M 107
M 110
M 116
M 114
M 111
M 113
M 117
M 114
M 112
F 110
F 111
F 107
F 108
F 110
F 105
F 107
F 106
F 111
F 111
```

■ Hypothesis

Male jackals have larger teeth than females.

■ Traditional parametric analysis (ANOVA)

```
In[65]:= ANOVA[data]
```

```
Out[65]= {ANOVA →
  DF      SumOfSq  MeanSq  FRatio  PValue
  Model   1      115.2   115.2   12.1405  0.00264727
  Error   18     170.8   9.48889
  Total   19     286.
  All     111.
  CellMeans → Model[F] 108.6
               Model[M] 113.4
```

By default, this is a two-way test. Given the hypothesis, a one-way test is appropriate, in which case $p = 0.0013$.

The assumptions of this analysis are:

- 1) Random sampling from the populations of interest
- 2) Equal population standard deviations
- 3) Normal distribution of mandible lengths within groups

■ Randomization test version

Calculate actual difference between means

```
In[66]:= observedDifference = N[Mean[males] - Mean[females]]
```

```
Out[66]= 4.8
```

Randomize observations into groups and calculate the difference

```
In[67]:= allData = Join[males, females]
```

```
Out[67]= {120, 107, 110, 116, 114, 111, 113, 117, 114,
  112, 110, 111, 107, 108, 110, 105, 107, 106, 111, 111}
```

```
In[68]:= randomOrdering = allData[[Ordering[Table[RandomReal[], {Length[allData]}]]]]
```

```
Out[68]= {111, 114, 120, 113, 112, 111, 111, 116, 107,
  111, 110, 114, 117, 110, 107, 107, 110, 105, 106, 108}
```

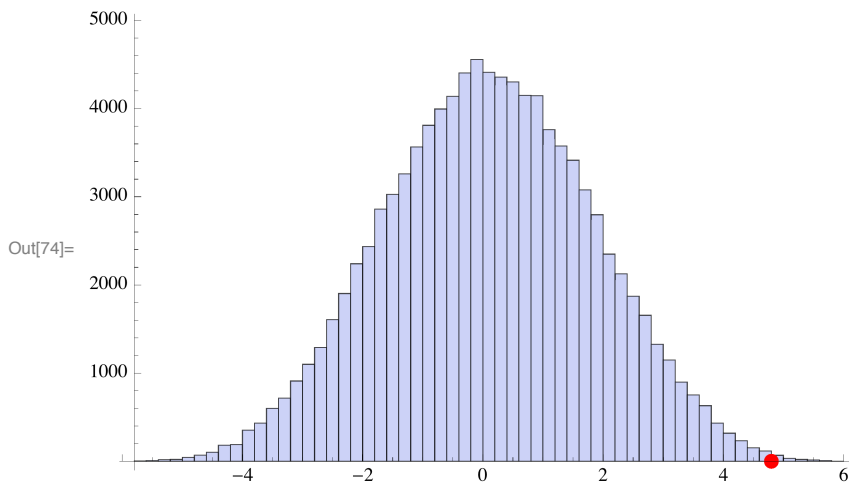
```
In[69]:= randomDifference = N[Mean[Take[randomOrdering, 10]] - Mean[Take[randomOrdering, {11, 20}]]]
```

```
Out[69]= 3.2
```

Do this many times!

```
In[73]:= randomDifferences = Table[
  randomOrdering = allData[[Ordering[Table[RandomReal[], {Length[allData]}]]]];
  N[Mean[Take[randomOrdering, 10]] - Mean[Take[randomOrdering, {11, 20}]]], {99 999}];
```

```
In[74]:= Show[Histogram[randomDifferences],
  ListPlot[{{observedDifference, 0}], PlotStyle -> {Red, PointSize[0.02]}]]
```



Calculate p as the fraction of random differences that are the same size or larger than the observed difference.

```
In[76]:= p = N[(Length[Select[randomDifferences, # >= observedDifference &]] + 1) /
  (Length[randomDifferences] + 1)]
```

```
Out[76]= 0.00147
```

Randomization produces a null distribution of the statistic of interest, preserving the essential characteristics of the data (because the data themselves are used — the randomization simply breaks the hypothesised organization of the data into groups).

```
In[134]:= Permutations[Range[4]]
```

```
Out[134]= {{1, 2, 3, 4}, {1, 2, 4, 3}, {1, 3, 2, 4}, {1, 3, 4, 2}, {1, 4, 2, 3}, {1, 4, 3, 2},
  {2, 1, 3, 4}, {2, 1, 4, 3}, {2, 3, 1, 4}, {2, 3, 4, 1}, {2, 4, 1, 3}, {2, 4, 3, 1},
  {3, 1, 2, 4}, {3, 1, 4, 2}, {3, 2, 1, 4}, {3, 2, 4, 1}, {3, 4, 1, 2}, {3, 4, 2, 1},
  {4, 1, 2, 3}, {4, 1, 3, 2}, {4, 2, 1, 3}, {4, 2, 3, 1}, {4, 3, 1, 2}, {4, 3, 2, 1}}
```

Data 2

The first dataset is a measure of the taxonomic similarity of the earwig assemblage on different land masses.

Out[142]//TableForm=

	E&A	AFR	MAD	FEA	AUS	NZ	SA	NA
Europe and Asia	0	0	0	0	0	0	0	0
Africa	0.3	0	0	0	0	0	0	0
Magagascar	0.14	0.5	0	0	0	0	0	0
Far East	0.23	0.5	0.54	0	0	0	0	0
Australia	0.3	0.4	0.5	0.61	0	0	0	0
New Zealand	-0.04	0.04	0.11	0.03	0.15	0	0	0
South Africa	0.02	0.09	0.14	-0.16	0.11	0.14	0	0
North America	-0.09	-0.06	0.05	-0.16	0.03	-0.06	0.36	0

The second dataset is a measure of the number of 'steps' required to jump from one land mass to another, assuming their current positions.

Out[143]//TableForm=

	E&A	AFR	MAD	FEA	AUS	NZ	SA	NA
Europe and Asia	0	0	0	0	0	0	0	0
Africa	1	0	0	0	0	0	0	0
Magagascar	2	1	0	0	0	0	0	0
Far East	1	2	3	0	0	0	0	0
Australia	2	3	4	1	0	0	0	0
New Zealand	3	4	5	2	1	0	0	0
South Africa	2	3	4	3	4	5	0	0
North America	1	2	3	2	3	4	1	0

The third dataset is a measure of the number of 'steps' required to jump from one land mass to another, assuming their positions in the geologic past when Gondwanaland was a large land mass that connected the various regions.

Out[144]//TableForm=

	E&A	AFR	MAD	FEA	AUS	NZ	SA	NA
Europe and Asia	0	0	0	0	0	0	0	0
Africa	1	0	0	0	0	0	0	0
Magagascar	2	1	0	0	0	0	0	0
Far East	1	1	1	0	0	0	0	0
Australia	2	1	1	1	0	0	0	0
New Zealand	3	2	2	2	1	0	0	0
South Africa	2	1	2	2	2	3	0	0
North America	1	2	3	2	3	4	1	0

■ Traditional parametric analysis

Cannot use a parametric method to test the association between the values in the distance matrices because they are not independent.

■ Randomization test — Mantel test

■ Functions to randomize a matrix and find the correlation between two matrices

■ Analysis

For the first distance matrix.

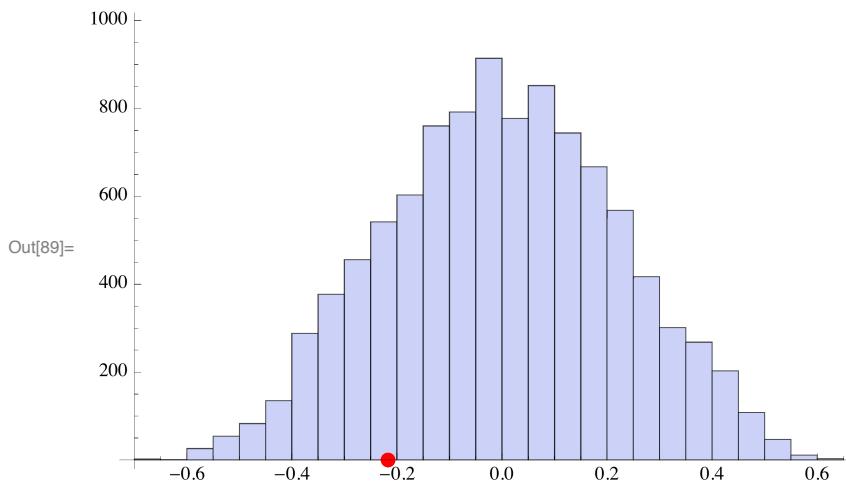
```
In[87]:= observedCorrelation1 = matrixCorrelation[associationMatrix, distanceMatrix1]
```

```
Out[87]= -0.216964
```

The observed correlation is negative, as expected (large geographic distances are associated with less taxonomic similarity).

```
In[88]:= randomizedCorrelations1 = Table[  
  matrixCorrelation[associationMatrix, randomizedDiagonalMatrix[distanceMatrix1]], {9999}];
```

```
In[89]:= Show[Histogram[randomizedCorrelations1],  
  ListPlot[{{observedCorrelation1, 0}}, PlotStyle -> {Red, PointSize[0.02]}]]
```



```
In[90]:= p = N[(Length[Select[randomizedCorrelations1, # <= observedCorrelation1 &]] + 1) /  
  (Length[randomizedCorrelations1] + 1)]
```

```
Out[90]= 0.1789
```

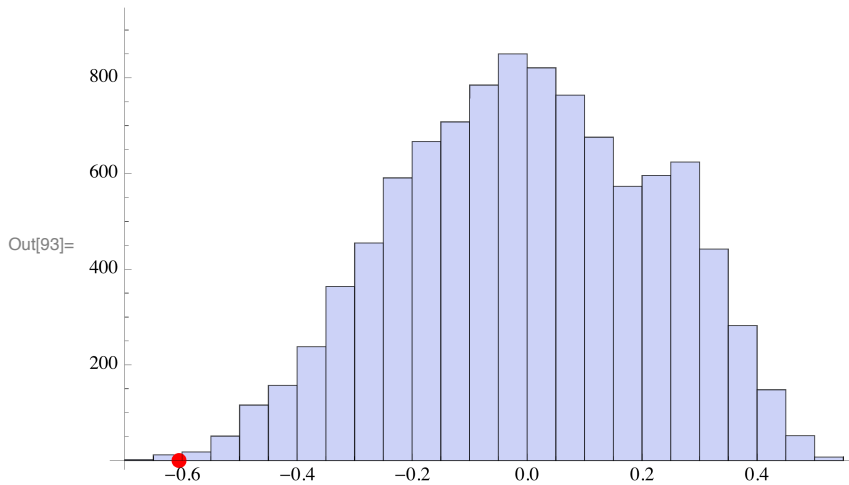
For the second distance matrix.

```
In[91]:= observedCorrelation2 = matrixCorrelation[associationMatrix, distanceMatrix2]
```

```
Out[91]= -0.605379
```

```
In[92]:= randomizedCorrelations2 = Table[  
  matrixCorrelation[associationMatrix, randomizedDiagonalMatrix[distanceMatrix2]], {9999}];
```

```
In[93]:= Show[Histogram[randomizedCorrelations2],
  ListPlot[{{observedCorrelation2, 0}}, PlotStyle -> {Red, PointSize[0.02]}]]
```



```
In[95]:= p = N[(Length[Select[randomizedCorrelations2, # <= observedCorrelation2 &]] + 1) /
  (Length[randomizedCorrelations2] + 1)]
```

Out[95]= 0.0012

Simulation — Monte-Carlo

Caveats

Non-parametric tests require fewer assumptions, and so are more flexible. Why not always use one? The answer is power. The more you know about your data, the easier it is to detect patterns, relationships, etc. So if you have a good reason to think that your data are Normally distributed (for example), you should use a parametric test — which includes that knowledge — because it will have greater power. The trade-off is the risk that you are wrong about your data being Normal, in which case your parametric test will have a higher power to give the wrong answer!