Randomization-Based Inference

MA386 - Statistical Programming

Summary: We have seen the use of numerical simulation for investigating a process or examining the properties of a statistical method. In this module, we discuss the application of simulations to randomization-based inference — a nonparametric approach to inference.

1 Bootstrapping

Summary: Statistical inference rests on modeling the sampling distribution of a test statistic. The Bootstrap is a method for constructing an empirical model for this distribution. It is primarily used to estimate the standard error of an estimator.

The pinacle of the introductory statistic statistics course is typically the Central Limit Theorem. This theorem is quite remarkable. Roughly, it states that if we were to resample from the population many times, each time computing the sample mean, these sample means would vary from one another according to a Normal distribution. This idea of resampling and observing how a statistic varies *across samples* is known as the sampling distribution. Developing a model for this distribution is the fundamental step when performing statistical inference. This model might be constructed to appealing to large-sample behavior (Central Limit Theorem); or, it might be developed by first assuming some properties about the population itself (assuming the population is Normal, for example, often occurs in ANOVA). The bootstrap procedure was developed as a method for constructing an empirical model for this sampling distribution under mild assumptions.

Instead of delving into the theoretical justification for the bootstrap procedure, we will appeal to a popular view of the procedure in which we visualize a "bootstrap world." Figure 1 illustrates the traditional nonparametric bootstrap procedure. We begin with a population (grey region in the figure), and suppose we are interested in some parameter μ characterizing that population. We conduct a study by taking a random sample from the population (orange region in figure) and compute our statistic $\bar{x}$ which estimates the parameter. We will refer to this sample as the *original sample.*

In order to obtain the sampling distribution, we would need to take additional random samples (blue regions in the figure), of the same size n as the original sample. In each of these additional samples, we would compute the statistic in the same way as our original sample. The sampling distribution would then be the distribution of these statistics; that is, imagine a histrgram made from $\bar{x}, \bar{x}_2, \bar{x}_3, \ldots, \bar{x}_m$ for some large number m. While conceptually this works, practically, it is infeasible. Due to limitations on time and resources, we cannot conduct our study a large number of times. That is, the additional samples are not realized.

However, we can emulate the process of constructing a sampling distribution. Hence, we enter the bootstrap world. In this alternate reality, the original sample becomes our population. It is the only thing that exists. We know everything there is to know about the population. We know which values are possible and how likely each is to occur. We now take a random sample (green region in the figure) from the original sample. During this step, we ensure the random sample of size n (same size as the original sample) is taken *with replication.* This is similar to catch and release fishing. Each value in our "population" (original sample) is eligible to enter our *re-sample* multiple times. These re-samples are referred to as *bootstrap samples.* As this re-sampling process can be done in the computer, it is very cheap to construct b bootstrap samples (where b is some large integer) very quickly.

Intuitively, this works because of the nature of random sampling. Each bootstrap sample is representative of the original sample, and the original sample is representative of the original population. So, the bootstrap samples then reflect what we would have expected to see if we had been able to repeatedly sample from the population. Using the *bootstrap statistics* $(\bar{x}_1^*, \bar{x}_2^*, \ldots, \bar{x}_b^*)$, we can model the sampling distribution of the original statistic $\bar{x}$!

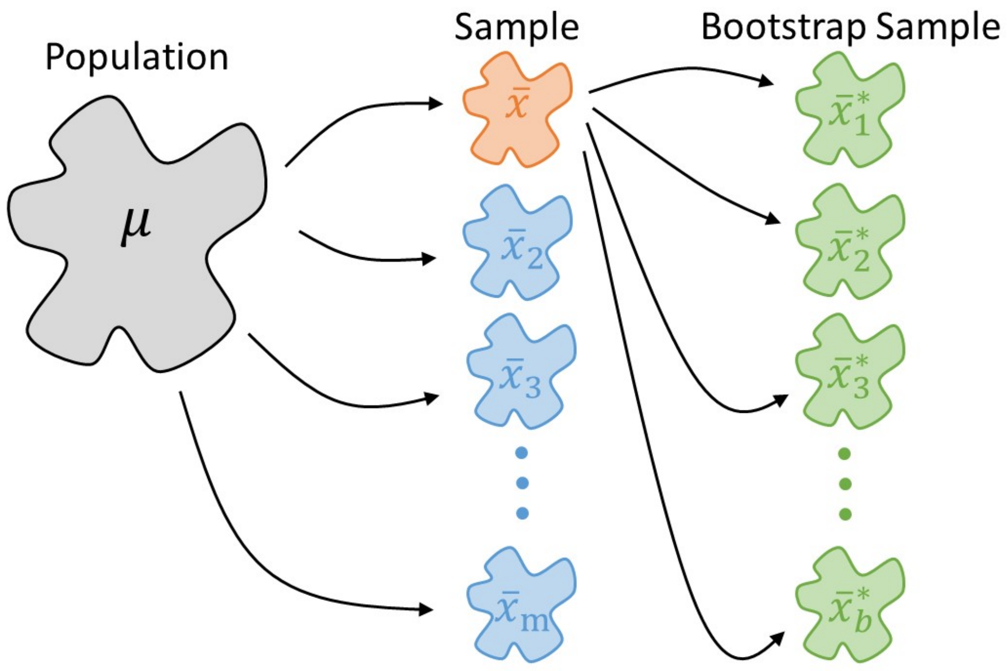

Figure 1: Illustration of bootstrap procedure.

In this alternate reality of ours, we have mimicked the true sampling design of the data and reproduced *the process* as closely as possible. **The standard deviation of the bootstrap statistics is an estimate of the standard error of the statistic of interest.**

Example: *(Guinea Pigs)* A research article investigated the body weight (grams) of Guinea Pigs at birth. Their primary question of interest was to estimate the variability in the body weights. The data is provided below:

Obs.	1	2	3	4	5
Weight	421.0	452.6	456.1	494.6	373.8
Obs.	6	7	8	9	10
Weight	90.5	110.7	96.4	81.7	102.4
Obs.	11	12	13	14	15
Weight	241.0	296.0	317.0	290.9	256.5
Obs.	16	17	18	19	20
Weight	447.8	687.6	705.7	879.0	88.8
Obs.	21	22	23	24	25
Weight	296.0	273.0	268.0	227.5	279.3
Obs.	26	27			
Weight	258.5	296.0			

The point estimate for the variability (quantified using the sample standard deviation) is easy to obtain.

```
# Read Data
Guinea.df <- data.frame(Pig = seq_len(27),
                        Weight = c(421.0, 452.6, 456.1, 494.6, 373.8,
                                    90.5, 110.7,  96.4,  81.7, 102.4,
                                   241.0, 296.0, 317.0, 290.9, 256.5,
                                   447.8, 687.6, 705.7, 879.0,  88.8,
                                   296.0, 273.0, 268.0, 227.5, 279.3,
                                   258.5, 296.0))


# Point Estimate
summarize(Guinea.df, SD.Weight = sd(Weight))
```

```
  SD.Weight
1  198.7855
```

Every point estimate should *always* be reported along with some measure of the variability in that estimate. We could report the standard error. However, unlike the sample mean, there is not a simple formula for the standard error of the standard deviation. So, we rely on bootstrapping to estimate this quantity. Our procedure will look as follows:

1. Resample the weights $B = 10000$ times.
2. For each resample, compute the standard deviation s_b^* for $b = 1, 2, \ldots, B$.
3. Take the standard deviation of these B statistics to estimate the standard error.

As always, there are several ways to implement the resampling in `R`; we will make use of many of the properties of the `dplyr` package and the `replicate()` function throughout this unit. For each bootstrap replication, we compute the quantity of interest and store the value. Since we are only storing a single value (the standard deviation) from each replicate, we will store them in a vector.

```
# Set Seed
set.seed(20160816)

# Resample Data
#  Ensure resampling is done with replacement and each sample is same size as
#  original sample.
Boot.sd <- replicate(10000, {
  # one bootstrap sample
  df.resample <- sample_n(Guinea.df, size=nrow(Guinea.df), replace=TRUE)

  # compute standard deviation
  #  ensure value is single number so that data goes into vector
  summarize(df.resample, SD=sd(Weight))[1,1]
})

# Compute Standard Error
sd(Boot.sd)
```

```
[1] 32.57505
```

More than providing an estimate of the standard error, we have an empirical model for the sampling distribution of the standard deviation. We can visualize the density graphically. From the plot, we see that in samples of this size, we would expect the value of the standard deviation to fall between 150 and 240, with a value near 200 to be most likely.

This particular implementation of the bootstrap is known as a *nonparametric bootstrap* because we made no underlying assumptions regarding the distributional form of the underlying population. An alternative approach would make use of distributional assumptions on the population.

For example, suppose that researchers were willing to believe that the weight of the guinea pigs tends to follow a Normal distribution; however, the mean and variance of the distribution are unknown. As before, they are particularly interested in estimating the standard deviation. How might we model the sampling distribution capitalizing on the additional information provided by the model for the population?

When implementing the bootstrap procedure above, we resampled from the original sample. This was because we were using the original sample as our model for the population. For those familiar, we are essentially using the empirical cumulative distribution function to define a probabilistic model for the population. In this revised example, we want to make use of a "known" probability model — that of a Normal distribution. This changes how the resampling occurs.

```r
# Set Seed
set.seed(20160816)

# Summarize Sample
#  Use the mean and standard deviation of the sample as estimates for the
#  unknown population parameters.
stats <- summarize(Guinea.df, Mean=mean(Weight), SD=sd(Weight))

# Resample Data
#  Ensure resampling is done from assumed model with estimates in place of the
#  parameters.
Boot.sd2 <- replicate(10000, {
  # one bootstrap sample
  resample <- rnorm(nrow(Guinea.df), mean=stats$Mean, sd=stats$SD)

  # compute standard deviation
  sd(resample)
})

# Compute Standard Error
sd(Boot.sd2)
```

```
[1] 27.41761
```

The model for the sampling distribution from this *parametric bootstrap* procedure (named so because we are making use of distributional assumptions) is summarized graphically below.

Exercise: *(Death Penalty)* A survey of 25 students (14 U.S. students and 11 international students) was conducted regarding their view of the death penalty. Of the 14 U.S. students, 10 were against the death penalty; of the 11 international students, 4 were against the death penalty. We are interested in evaluating whether there is evidence that the proportion of students against the death penalty differs between U.S. and international students.

The first thing to consider is an appropriate statistic. Often, for categorical data such as this, the odds ratio is examined. Let p_1 be the proportion of U.S. students against the death penalty and let p_2 be the proportion of international students against the death penalty. Then, the odds ratio comparing U.S. to international students is given by

$$OR = \left(\frac{p_1}{1-p_1}\right) / \left(\frac{p_2}{1-p_2}\right) = 70/16 = 4.375$$

Since the OR is larger than 1, this suggests that US students are more likely to be against the death penalty. Use a nonparametric bootstrapping procedure to estimate the standard error of this estimate.

One of the key ideas behind bootstrapping is that by repeating the procedure, we get an idea of the possible results (statistics) we might observe if we were to repeat the study. So, in addition to the standard error, with a model for the sampling distribution, we can construct confidence intervals to produce interval estimates.

1.1 Bootstrap Percentile Interval

The simplest form of a $100c\%$ confidence interval is to simply take the $(1-c)/2$ and $(1+c)/2$ empirical percentiles of the bootstrapped statistics. That is, we isolate the center $100c\%$ of the sampling distribution from the bootstrap statistics. This is easily done with a single line of code making use of the `quantile()` function. For example, for the Guinea Pig dataset, if we want a 95% confidence interval for the standard deviation using the nonparametric bootstrap procedure, we would have the following:

```r
# 95% CI
#  We remove missing values (if any) when making this computation and specify
#  the definition of a percentile we wish to use (there are many).
quantile(Boot.sd, probs=c(0.025, 0.975), type=8, na.rm=TRUE)
```

This confidence interval has the same interpretation as those constructed using an analytical formula. It specifies the values of the parameter with which this data is consistent. Such an interval requires isolating $100c\%$ of the sampling distribution; there is no requirement that it be the middle percentage. Isolating the middle is typically done because sampling distributions are often unimodal and symmetric, and by specifying the center, we are grabbing the most common values from the sampling distribution. However, for the sampling distribution of the odds ratio from the Death Penalty example, we may consider other options since the sampling distribution is heavily skewed.

We should note that while intuitive, the percentile interval suffers from some limitations. Efron, who originally proposed the bootstrap procedure, actually recommended an alternative form known as the BC interval. Another approach is to consider constructing an interval using the components from the bootstrap sampling distribution but mimicking the formula from the confidence interval for the population mean when the underlying population is known to be Normal. It is appropriately named the Bootstrap-t Interval. There are other bootstrap intervals as well, but these seem to be those used most often in practice. While there has been a lot of work done on the theoretical properties of these intervals, there does not seem to be an "all-purpose" approach. For each of these intervals, there exist a series of counter-examples for which the performance is not very good.

For our purposes, the percentile interval is often sufficient.

1.2 Bag of Little Bootstraps

In recent years, data has grown rapidly in size. We no longer think in megabytes or gigabytes but terabytes and petabytes. These massive datasets require specialized approaches. Many common statistical models (linear regression, logistic regresion) have been parallelized to be "scalable" to these larger datasets. However, not all statistical approaches can be implemented on large datasets. The bootstrap, in the form described above does not lend itself to being implemented at scale. To those familiar with the concepts of parallel programming, this may seem somewhat surprising. Since bootstrapping works on each resample independently and then recombines, this seems primed for parallelization; in fact, it describes the MapReduce paradigm. However, transfer of the data is typically the most expensive portion of data analysis. And, since we resample with replacement such that each resample has the same dimension as the original sample, the data transfer required for bootstrapping is expensive.

One approach to solving this problem (at least at the terabyte scale) is the "bag of little bootstraps" approach. The tweak is somewhat simple and yet extraordinarily powerful. The algorithm has the following approach:

1. Randomly partition the original dataset into k smaller datasets, each of size m where $mk = n$ (for simplicity here).
2. For each partition, perform a bootstrap with B replicates and compute the CI of interest, for example.
3. Recombine the CI's from each partition for an overall result, by averaging the endpoints of the interval, for example.

The key step is actually hidden above. When constructing the bootstrap replicates, resample the partition of size m so that the bootstrap sample has size n (the size of the original sample). This novel step ensures the standard error computation has been appropriately computed. This takes us from small data back to a large dataset, but we have saved effort on data transfer, which is the most costly endeavor.

As an example of the process, consider all flights that departed New York City from a major airport in 2013. The dataset, available with the `nycflights13` package, can be loaded in the following way:

```
# Load Data
data(flights, package="nycflights13")
```

Suppose we are interested in constructing a confidence interval for the median time (minutes) a flight spends in the air. A traditional bootstrap would be easy to implement on a large machine, but will struggle on a laptop. To truly implement the bag of little bootstraps procedure, we would want our code to operate in parallel; as this is beyond the scope of this course, we simply illustrate the idea.

```
# Specify Bootstrap Parameters
B <- 500
n <- nrow(flights)

# Partition Data
#  We break it into 498 groups of size 675 and 1 group of size 626.  Each
#  record is randomly assigned to one of these groups.
flights.list <- flights %>%
  mutate(Boot.Groups =
           sample(seq(from=0, to=n-1) %/% 498, size=n, replace=FALSE)) %>%
  split(.$Boot.Groups)


# Bootstrap
#  Store the results for each component of partition in separate element of
#  list.
Boot.list <- flights.list %>%
  lapply(function(u){
    # bootstrap each piece separately
    replicate(B, {
      # resample with larger size
      sample_n(u, size=n, replace=TRUE) %>%
        "[["("air_time") %>%
        median(na.rm=TRUE)
    })
  })


# Compute CI
#  Compute CI for each component of the partition and then merge.
Boot.list %>%
  sapply(quantile, probs=c(0.025, 0.975), type=8, na.rm=TRUE) %>%
  apply(1, mean)
```

We leave it up to the reader to verify that the results obtained by the little bag of bootstraps procedure is similar to those that would be obtained if we had conducted a traditional bootstrap using the original dataset. We now turn our attention to conducting hypothesis tests.

2 Permutation Tests

Summary: When we have observational data, we think of the observed data as a random sample from some underlying population. In this way, bootstrapping is an analogous procedure — we are mimicking the sampling process. But, in a controlled experiment, the key step is not the random sampling but the randomization of subjects to the treatment groups. Mimicking this randomization process leads to what are known as randomization-based tests.

The key idea behind conducting a controlled experiment is that the randomization to groups should create an "even playing field." That is, the two (or more) treatment groups will have roughly the same demographics; therefore, any differences in the response between the two groups must be due to the grouping and not any underlying demographics. This eliminates the effects of potential confounders. However, it is possible that by chance alone, one group ends up having a larger response on average even though in the general population the two groups are similar. The idea in a permutation test is to compute this likelihood of the difference occuring by chance alone. That is, we want to determine the probability that we would end up with this large of a group or larger given the null hypothesis — a p-value.

So, while bootstrapping models the sampling distribution of the statistic, randomization tests model the null distribution.

Example: *(Reading)* An educator conducted an experiment to test whether new directed reading activities in the classroom will help elementary school pupils improve some aspects of their reading ability. A group of 44 students were randomly assigned to one of two classrooms. One group (consisting of 21 students) was followed the activities over an 8-week period. The other group (consisting of 23 students) acted as the control group, following the same curriculum without the activities. At the end of the 8 weeks, all students took a Degree of Reading Power (DRP) test, which measures the aspects of reading ability that the treatment is designed to improve. Is there evidence of a difference between the scores of the two groups on average?

The idea here is to really pick apart the definition of a p-value. If the null hypothesis is really true, then there is no difference between the two groups. Further, any difference observed is due to chance; that is, it is due to the randomization process. And, if the null is true, the response should have been the same regardless of the treatment group assignment. Therefore, we replicate the randomization process. What if it had gone differently? Could we have seen a more extreme result? A less extreme result? By re-doing the randomization, we are essentially forcing the null to be true because we are acting as if the result of the individual would have been the same regardless. And, any differences we see are due only to chance!

We take all subjects within the same pool and randomize them to two groups (of the same sizes represented in the original study). Then, we compute some statistic which measures the effect observed. While any statistic will work (difference in sample means or medians, for example), using a standardized statistic tends to have better properties.

Before we actually conduct the randomization, we need to consider the definition of a p-value: it is the probability of observing a statistic as contrary or moreso to the null hypothesis *as the one observed*. This last part, with emphasis added, suggests our first step is to compute the test statistic for our original sample.

```
# Read Data
Read.df <- data.frame(
  Score=c(24, 43, 58, 71, 43, 49, 61, 44, 67, 49,
          53, 56, 59, 52, 62, 54, 57, 33, 46, 43,
          57, 42, 43, 55, 26, 62, 37, 33, 41, 19,
          54, 20, 85, 46, 10, 17, 60, 53, 42, 37,
```

```r
          42, 55, 28, 48),
  Trt.Grp=rep(c("Treated", "Control"), times=c(21, 23)))


# Test Statistic: Original Sample
#  Compute the traditional two-sample standardized statistic.
stat <- Read.df %>%
  group_by(Trt.Grp) %>%
  summarize(Mean=mean(Score), Var=var(Score), N=n()) %>%
  summarize(Stat = diff(Mean)/sqrt(sum(Var/N))) %>%
  as.numeric()
```

With the original statistic computed, we now construct the randomizations to each group. For each new randomization, we re-compute the statistic and store the result.

```r
# Set Seed
set.seed(20160818)

# Specify Parameters
m <- 5000

# Compute Randomizations
#  Re-randomize the subjects to two groups as above and recompute the statistic.
#  The values stay the same, so we need to scramble the groups.
test.stats <- replicate(m, {
  # scramble groups, then compute test stat as above
  Read.df %>%
    mutate(Trt.Grp = sample(Trt.Grp, size=length(Trt.Grp), replace=FALSE)) %>%
    group_by(Trt.Grp) %>%
    summarize(Mean=mean(Score), Var=var(Score), N=n()) %>%
    summarize(Stat = diff(Mean)/sqrt(sum(Var/N))) %>%
    as.numeric()
})
```

The p-value is then computed by determining what fraction of these values are more extreme than that observed.

```r
# Compute p-value
round(mean(abs(test.stats) >= abs(stat)), 4)
```

```
[1] 0.0278
```

Notice we take the absolute value because the alternative hypothesis was two-sided in this case. So, we want to be extreme in either direction. This result can also be visualized graphically:

The most important part, and most difficult part in general, of the above process is generating data *under the null hypothesis*. In the two-sample setting, if the null hypothesis is true, then the means are the same. Therefore, putting the two samples in the same group forces them to have the same mean. Then, we randomize them to two groups. This mimicks the experiment conducted. The test statistic is then computed when the null hypothesis is true (by design) in each replication.

The subtlety in the above is that we are assuming the variances of the two groups are equal (even if the test-statistic we chose does not make that assumption in the two-sample t framework). Again, when we combine the two groups into a single sample, the population now has the same mean and the same variance.

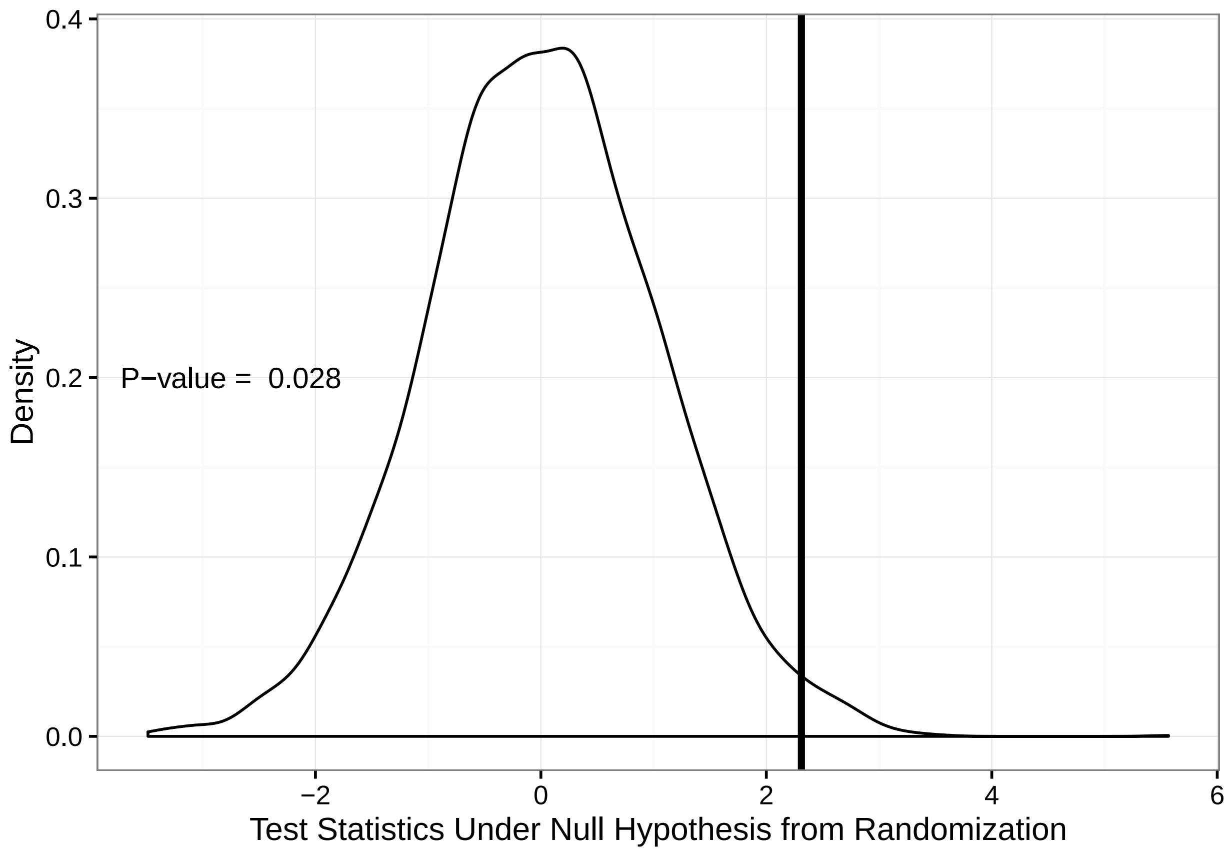

Figure 2: Illustration of the extreme results obtained from the study. This p-value is computed via a randomization test.

Notice that in theory, an exact p-value could be computed because there are a finite number of different scenarios (randomization to groups) that could occur. In practice, we care little about trying to examine all such permutations and simply consider a random sample of them. The differences in the p-values obtained are generally negligible.

Exercise: *(M&M's)* On Halloween my daughter was excited to get packs of fun-size M&M's. Suppose we are interested in determining if there is evidence that the proportion of at least one color differs from the rest. In our fun-size bag, we observed 3 orange, 2 yellow, 2 brown, 2 green, 1 blue, and 5 red candies. Is it reasonable that the proportion of each color in fun-size bags are all equal?

Before writing code to address this question, answer the following two questions:

1. How can I generate data under the null hypothesis (all proportions are equal)?
2. What test statistic will I use? It should be large when H_0 is false and small when H_0 is true.