Monte Carlo Methods for Inference

Max Turgeon

STAT 3150–Statistical Computing

Lecture Objectives

- Recall the definition of a statistic and its sampling distribution
- Explain how simulations can be used for estimation and hypothesis testing
- Design and conduct a simulation study

Motivation

- Over the last few weeks, we talked about generating random variables and used them to estimate integrals.
- Starting this week, we will investigate how these ideas can be used for data analysis.
    - How to use simulations to estimate population parameters.
    - How to use simulations to perform hypothesis testing.

Definitions i

First, recall the following definitions:

- A **statistic** is a function of a sample, i.e. from a sample $X_1, \dots, X_n$ compute an output.
    - Sample mean, sample variance, etc.
    - Histogram, empirical CDF
- An **estimator** is a statistic $\hat{\theta}$ is a statistic used to estimate (or "approximate") a population parameter θ.
    - The sample mean estimates the population mean
    - The empirical CDF approximates the population CDF.

Definitions ii

- A statistic is a random variable, because it is a function of the sample. Therefore it has a distribution: the **sampling distribution**.
    - If $X_1, \ldots, X_n$ are $N(\mu, \sigma^2)$, then the sampling distribution for the sample mean is $N(\mu, \sigma^2/n)$

Remark

- The sampling distribution is often a function of unknown population parameters.
    - Or even the type of distribution may be unknown.
- Monte Carlo methods can be used to estimate the sampling distribution and derive quantities of interest.
    - E.g. Mean Squared Error, percentiles.

Example: 538's The Riddler i

- Refer to this post: `https://fivethirtyeight.com/features/can-you-parallel-park-your-car/`
- The population parameter we want to estimate is $P(\text{Have to parallel park})$.
- A sample is an arrangement of four cars in six parking spots, with each arrangement equally likely.
- From a sample, we can determine if the Riddler will have to parallel park or not.
    - Our statistic T is binary: *Yes* or *No*.

Example: 538's The Riddler ii

- This can be modeled using a Bernoulli distribution with parameter $p = P(T = \text{Yes})$.
    - Recall, this is the **sampling distribution**.
- To estimate p, we can simulate $B = 1000$ samples, compute T for each sample, and count the proportion $\hat{p}$ of samples for which $T = \text{Yes}$.
    - This is Monte Carlo integration!
- The estimate of the variance of T is $\hat{p}(1-\hat{p})$, and therefore our standard error for our estimate $\hat{p}$ is

$$se(\hat{p}) = \sqrt{\frac{\hat{p}(1-\hat{p})}{B}}.$$

Example i

- Assume we have a sample of size 2 from a standard normal distribution: X_1, X_2.
- We want to estimate the expected value of their absolute difference:

$$g(X_1, X_2) = |X_1 - X_2|.$$

- **How can we do this?** Monte Carlo integration!

Example ii

```r
B <- 1989
norm_vars1 <- rnorm(B)
norm_vars2 <- rnorm(B)
# Compute statistic
gvars <- abs(norm_vars1 - norm_vars2)
mean(gvars)
```

```
## [1] 1.124522
```

```r
sd(gvars)/sqrt(B)
```

```
## [1] 0.01919893
```

Exercise

Using Monte Carlo simulations, find the average Euclidean distance between two points uniformly and independently drawn from the unit square, i.e. both the x and the y coordinates for a single point are drawn from $U(0, 1)$.

To help you find the solution, try to answer the following questions:

- What constitutes a sample? (Is it one point? Two points?)
- What is the statistic?

Solution i

- Recall: if we have two points (x_1, y_1) and (x_2, y_2), their Euclidean distance is

$$dist = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}.$$

- The answer to the questions:
    - A sample is a *pair* of points.
    - The statistic is the Euclidean distance between the two points in the sample.

Solution ii

- The idea is therefore to generate multiple pairs of points and compute their Euclidean distance. We get an estimate of the average distance by taking the sample mean.

Solution iii

```
B <- 5000

# Using replicate, or a for loop
dist_vec <- replicate(B, {
    point1 <- runif(2)
    point2 <- runif(2)
    dist <- sqrt(sum((point1 - point2)^2))
    return(dist)
})

mean(dist_vec)
```

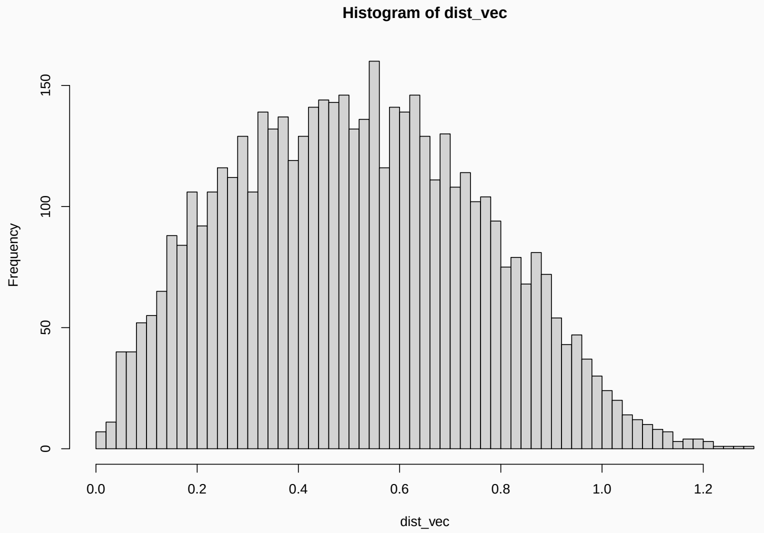
Solution iv

```
## [1] 0.5167341
```

- If we look at a histogram of the sampling distribution, we can see that it doesn't look like a normal distribution.

```r
hist(dist_vec, breaks = 50)
```

Solution v


Histogram of dist_vec
Frequency
0
50
100
150
0.0
0.2
0.4
0.6
0.8
1.0
1.2
dist_vec

Solution vi

- However, by the Central Limit Theorem, the sample mean of the distances approximately follows a normal distribution.
- We can use this result to construct a 95% confidence interval around our estimate.

```r
theta <- mean(dist_vec)
se_dist <- sd(dist_vec)/sqrt(B)

c(theta - 1.96*se_dist,
  theta + 1.96*se_dist)
```

```
## [1] 0.5099299 0.5235383
```

Solution vii

- **Be careful**: There are two sampling distributions here, and you should be able to distinguish between them.
    - The sampling distribution of the distances (doesn't depend on the sample size).
    - The sampling distribution of the *sample mean* of the distances (does depend on the sample size).

Mean squared error i

- Suppose we want to use an estimator $\hat{\theta}$ to estimate a parameter θ.
- Recall $\hat{\theta}$ is a random variable with a distribution. We say the estimator $\hat{\theta}$ is **unbiased** if its expected value is θ:

$$E(\hat{\theta}) = \theta.$$

- We can study the (un)biasedness of $\hat{\theta}$ by using the **mean squared error** (MSE):

$$MSE(\hat{\theta}) = E\left[\left(\hat{\theta} - \theta\right)^2\right].$$

Mean squared error ii

- **Why?** The MSE is related to the variance and the bias of $\hat{\theta}$:

$$MSE(\hat{\theta}) = \mathrm{Var}(\hat{\theta}) + \left(E(\hat{\theta}) - \theta\right)^2.$$

- This relates to what is called the **variance-bias tradeoff**:
    - For a fixed MSE, lower bias implies higher variance and vice-versa.

Example i

- The sample mean is an unbiased estimate of the population mean.
- However, it can be sensitive to outliers.

```r
mean(c(1,5,2,8, 4))
```

```
## [1] 4
```

```r
mean(c(1,5,2,8, 100))
```

```
## [1] 23.2
```

Example ii

- An estimator of the mean that is *less* sensitive to outliers is the **trimmed mean**.
- The idea is to remove the extreme values from the sample before taking the mean.
- More precisely: let $X_1, \ldots, X_n$ be a random sample, and let $k < 0.5n$ be a positive integer.
- The k-th level trimmed mean is defined as:

$$\bar{X}_{[k]} = \frac{1}{n - 2k} \sum_{i=k+1}^{n-k} X_{(i)},$$

where $X_{(i)}$ is the i-th order statistic.

Example iii

```r
# Generate a standard normal
# sample of size 4
(norm_vars <- rnorm(4))
```

```
## [1] 1.42856694 -0.05512797 -0.34324464
-0.32587459
```

```r
# Sort it
(norm_vars <- sort(norm_vars))
```

```
## [1] -0.34324464 -0.32587459 -0.05512797
1.42856694
```

Example iv

```r
# Compute 1st level trimmed mean
mean(norm_vars[c(-1, -4)])
```

```
## [1] -0.1905013
```

```r
# Compare to sample mean
mean(norm_vars)
```

```
## [1] 0.1760799
```

- We can generate a sample of size $n = 20$ and compare the MSE of the sample mean with the 1st-level trimmed mean.

Example v

```r
n <- 20
results <- replicate(3150, {
  norm_vars <- sort(rnorm(n))

c("TM" = mean(norm_vars[c(-1, -n)]),
  "SM" = mean(norm_vars))
})
```

```r
# Bias
rowMeans(results) - 0
```

Example vi

```
##            TM            SM
## -0.001538027 -0.001478366
```

```
# MSE
rowMeans((results - 0)^2)
```

```
##          TM         SM
## 0.05288323 0.05160470
```

- There isn't any outliers, so we get similar results for both types of means.

Example vii

- Let's introduce outliers through a *contaminated normal* distribution:

$$X \sim pN(0, 1) + (1 - p)N(0, 100).$$

- In other words, X follows a mixture distribution.
    - The second component, $N(0, 100)$, is responsible for the outliers in the sample.
- We can generate from a mixture distribution as follows:
    - Generate from a Bernoulli distribution with probability p.
    - If $Y = 0$, generate from the first component $N(0, 1)$.
    - If $Y = 1$, generate from the second component $N(0, 100)$.

Example viii

```r
p <- 0.9
n <- 20; B <- 2209

results <- replicate(B, {
  sigmas <- sample(c(1, 10), n, replace = TRUE,
                   prob = c(p, 1 - p))
  contnorm_vars <- rnorm(n, sd = sigmas)
  contnorm_vars <- sort(contnorm_vars)
  c("TM" = mean(contnorm_vars[c(-1, -n)]),
    "SM" = mean(contnorm_vars))
})
```

Example ix

```r
# Bias
rowMeans(results) - 0
```

```
##          TM          SM
## 0.009495568 0.003335998
```

```r
# MSE
rowMeans((results - 0)^2)
```

```
##        TM        SM
## 0.1868042 0.5267837
```

Example x

- As we can see, the two types of means have similar bias.
- But the trimmed mean has a lower MSE than the sample mean.
    - And therefore it has lower variance.
- **Conclusion**: With finite samples, we can sometimes find more efficient estimates of the mean.

Hypothesis testing i

- In hypothesis testing, we start with a **null hypothesis** about our parameter θ:
$$H_0 : \theta = \theta_0.$$
- We then use a **test statistic** to determine whether we should reject or not the null hypothesis.
    - A test statistic can also be an estimator, but more often it's a transformation thereof.
- If we know the sampling distribution of our test statistic when H_0 holds (i.e. $\theta = \theta_0$), then we can compute *how likely* it is to observe some given values of a test statistic.

Hypothesis testing ii

- This gives rise to the notion of a **p-value**: if your test statistic is T, and the observed value (i.e. after you've plugged in your data) is t, then the p-value is the following conditional probability:

$$P(T > t \mid H_0 \text{ hold}).$$

- Finally, we can reject the null hypothesis if the p-value is smaller than a predetermined level of significance α.
- With hypothesis testing, we can make two types of error:
    - **Type I error**: Rejecting the null hypothesis when it holds. This is typically controlled by our decision rule (i.e. when we call a p-value significant).

Hypothesis testing iii

    - **Type II error**: Not rejecting the null hypothesis when it doesn't hold. All things being equal, we would prefer a test with a smaller Type II error rate.

- **Power** is 1 minus the Type II error rate. By minimizing the latter, we increase power. All things being equal, we would prefer a test with higher power.
    - Note: Power typically increases with the sample size. The larger the sample size, the more likely we will reject the null hypothesis.

Example i

- We can use simulations to estimate the type I error rate.
- Here's the general idea:
    - Simulate data assuming the null hypothesis holds.
    - Perform a hypothesis test on the simulated data.
    - Count the proportion of our simulations that lead to a rejection of the null hypothesis.
- Note: To estimate power, simulate data when the null hypothesis doesn't hold.
- Consider two normal distributions $N(\mu_1, \sigma_1^2)$ and $N(\mu_2, \sigma_2^2)$, and assume that the null hypothesis is $H_0 : \mu_1 = \mu_2$.

Example ii

- We can generate from these two distributions by using the same mean, and use a t-test to decide whether we reject H_0 or not.
    - This can be done by comparing our p-value to our significance level α.
- Our estimate of the type I error rate would be the proportion of simulated datasets that led to a rejected t-test.

Example iii

```r
# Number of simulations
B <- 1000
# Sample size for data
n <- 20
# Same mean
mu1 <- mu2 <- 0
# Same variance; could also be different
sigma1 <- sigma2 <- 1
```

Example iv

```r
results <- replicate(B, {
  # Generate two samples
  norm_vars1 <- rnorm(n, mu1, sigma1)
  norm_vars2 <- rnorm(n, mu2, sigma2)
  # Perform t-test
  output <- t.test(norm_vars1, norm_vars2)
  # alpha = 0.05
  return(output$p.value < 0.05)
})

table(results)/B
```

Example v

```
## results
## FALSE  TRUE
## 0.954 0.046
```

- Our estimate of the Type I error rate is close to our significance level $\alpha = 0.05$.
- **Question**: To increase the accuracy, should we increase `B` or `n`?