HUDM 6026
Computational Statistics

Week 2 – Monte Carlo Simulation Studies, Part II

R: User-written functions

One of the advantages of R is the user's ability to add functions. The structure of a function is given below.

```
myfunction <- function(arg1, arg2, ... )
{
  statements
  return(object)
}
```

arg1, arg2, … are arguments that are passed to the function when it is called, i.e., the input.

Objects in the function are local to the function. The returned object can be any data type, i.e., this is the output.

It can be instructive to look at the code of a function. In R, you can view a function's code by typing the function name without the ( ).

R Example

The function below, called “sumsq” calculates the sum of squared deviations from the sample mean of a vector.

```
sumsq <- function(vec)
{
  mn <- mean(vec)
  d <- vec - mn
  d2 <- d^2
  ss <- sum(d2)
  ss
}

set.seed(123)
sumsq(rnorm(100))
[1] 82.49005
```

Monte Carlo Simulation Studies

- Properties of statistical procedures are often available through large-sample theory (i.e., central limit theorem).
- However, the procedures are used with finite samples in cases where assumptions are violated. Often questions in applied statistics have to do with the properties of estimators, procedures, or hypothesis tests under varying conditions. Some examples:
  - Estimator: In linear regression, what happens to the estimates of regression coefficients and standard errors if the assumption of homogeneity of error variances is violated?
  - Test: In testing a difference in means with the two-sample *t*-test, how does it affect the power and Type I error rate if the samples were not generated from normal distributions and the sample size is small?
  - Procedure: In cluster analysis, how is classification accuracy affected when a method that assumes a linear decision boundary is used with data that actually have a nonlinear boundary?

Monte Carlo Simulation Studies

- These questions often have answers that are difficult or impossible to work out analytically.
- This is where simulation studies can help.
- A *Monte Carlo simulation study* is an experiment in which
  a) a computer is used to generate *R* replications of data from a known probability distribution (called the *data-generating process*; DGP),
  b) each of the *R* generated data sets are analyzed with one or more statistical tests or procedures of interest, and
  c) the results of the analyses are aggregated over the *R* replications and (if possible) compared to the known truth specified in the DGP.

Monte Carlo Simulation

- When the goal is to examine or compare the properties of one or more estimators, the performance outcomes are typically bias, standard deviation (or variance) and mean squared error (MSE),

a) $\text{Bias} = E\left[\hat{\theta} - \theta\right]$

b) $\text{Standard deviation} = \sqrt{E\left[\left(\hat{\theta} - \bar{\hat{\theta}}\right)^2\right]}$

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

- where θ is the true parameter value, $\hat{\theta}$ is the estimator, and $\bar{\theta}$ is the mean (or expected value) of the estimator.

Simple Example - Setup

- Two estimators for the variance of a population, σ^2, are the usual unbiased estimator s^2 and the maximum likelihood estimator $\hat{\sigma}^2$:

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2 \quad \text{and} \quad \hat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

- where $x_1, x_2, \ldots, x_n$ are an *iid* sample of size n from the population.
- Some facts:
  1. We can show, analytically, that s^2 is an unbiased estimator for σ^2 and that the MLE estimator has a bias of $n/(n - 1)$.
  2. We can also work out the variances and MSEs. If population is normal,

$$\operatorname{var}(s^2) = \frac{2\sigma^2}{n-1} \quad \text{and} \quad \operatorname{var}(\hat{\sigma}^2) = \frac{2\sigma^2(n-1)}{n^2}$$

$$\operatorname{MSE}(s^2) = \frac{2\sigma^2}{n-1} \quad \text{and} \quad \operatorname{MSE}(\hat{\sigma}^2) = \frac{(2n-1)\sigma^2}{n^2}$$

Simple Example - Setup

- For our simulation we will carry out the three steps as follows:
  1. Generate 10000 replications (R = 10000) of samples of size 10 from a standard normal distribution.
  2. For each replication i in 1, …, R, determine s_i^2 and $\hat{\sigma}_i^2$, the values of s^2 and $\hat{\sigma}^2$, respectively, for each replication.
  3. Estimate bias, variance, and MSE as follows. For example for s^2,

$$\bar{s}^2 = \frac{1}{R}\sum_{i=1}^{R} s_i^2$$

$$\text{Bi}\hat{\text{a}}\text{s}\left(s^2\right) = \bar{s}^2 - \sigma^2$$

$$\text{S}\hat{\text{D}}\left(s^2\right) = \sqrt{\frac{1}{R-1}\sum_{i=1}^{R}\left(s_i^2 - \bar{s}^2\right)^2}$$

$$\text{M}\hat{\text{S}}\text{E}\left(s^2\right) = \sqrt{\frac{1}{R}\sum_{i=1}^{R}\left(s_i^2 - \sigma^2\right)^2} \approx \left(\text{S}\hat{\text{D}}\left(s^2\right)\right)^2 + \left[\text{Bi}\hat{\text{a}}\text{s}\left(s^2\right)\right]^2$$

Simple Example - Simulation

```
### Data generation function for standard normal
dg0 <- function(ss) {
  out <- rnorm(ss, mean = 0, sd = 1)
}

### Analysis function to estimate s^2 and σ̂²
simFun0 <- function(dat) {
  nn <- length(dat)
  s2 <- var(dat)
  sigma2hat <- s2*((nn-1)/nn)
  out <- c(s2, sigma2hat)
  out
}

### Define number of replications and sample size
R <- 10000
n <- 10
```

Simple Example - Simulation

```
### Create a matrix to store output
out0 <- matrix(0, R, 2)

### Set the seed for reproducibility
set.seed(145389)

### Run the simulation within a for loop
for (i in 1:R) {
  out0[i,] <- simFun0( dg0(n) )
}

### Estimate bias, var, sd, mse
(means0 <- apply(out0, 2, mean))
(bias0 <- means0 - c(1,1))
(var0 <- apply(out0, 2, var))
(sd0 <- apply(out0, 2, sd))
(mse0 <- apply(out0, 2, function(vec) mean((vec - 1)^2)))
### Can also calculate the simulation standard error for the mean by
### dividing sd0 by the square root of R.
(se0 <- sd0/sqrt(R))
```

Simple Example - Simulation

```
output <- rbind(means0, bias0, var0, mse0)
rownames(output) <- c("Means", "Bias", "Var", "MSE")
colnames(output) <- c("s2", "sigma2hat")
output

> output
                s2    sigma2hat
Means  0.998941171  0.899047054
Bias  -0.001058829 -0.100952946
Var    0.223121767  0.180728631
MSE    0.223100575  0.190902055
SimSE  0.004723577  0.004251219
```

Typically, we would stop with the above output and interpret the results. However, in this case, the analytical formulas are available so we can compare. For normally distributed data,

$$\operatorname{var}\left(s^2\right)=\frac{2\sigma^2}{n-1} \quad \text{and} \quad \operatorname{var}\left(\hat{\sigma}^2\right)=\frac{2\sigma^2(n-1)}{n^2}$$

$$\operatorname{MSE}\left(s^2\right)=\frac{2\sigma^2}{n-1} \quad \text{and} \quad \operatorname{MSE}\left(\hat{\sigma}^2\right)=\frac{(2n-1)\sigma^2}{n^2}$$

Example 2: Simple Linear Regression

- Simple linear regression involves only one outcome Y and one predictor X. We assume: (a) the relationship between X and Y is linear plus random noise, (b) that the random noise has mean zero, and (c) that the random noise has constant variance σ^2.

$$Y = \beta_0 + \beta_1 X + \varepsilon$$

- Usually we assume $\varepsilon \sim N(0, \sigma^2)$
- The unknown parameters are the betas and σ^2.
- We gather data in the form of n pairs $(x_1, y_1), \ldots, (x_n, y_n)$ of observations and use the data to get estimates $\hat{\beta}_0$ & $\hat{\beta}_1$.
- Then, the predicted values can be computed using the estimates:

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i + \varepsilon_i$$

- Finally, the residuals are the differences between actual and predicted:

$$r_i = y_i - \hat{y}_i$$

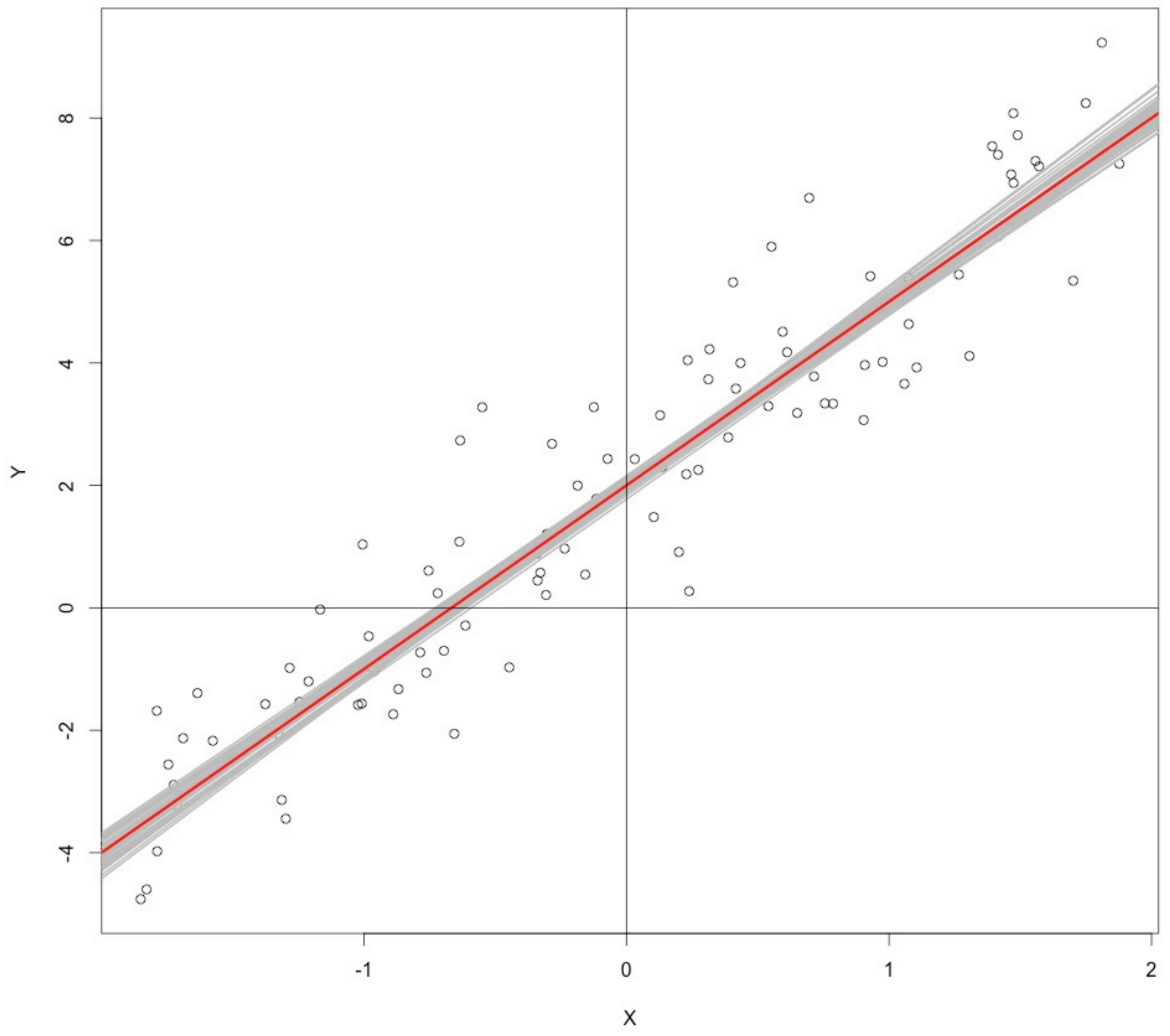
Variability of the estimates

```
### A simulation study to assess model accuracy
### Suppose the true model is Y = 2 + 3X + e, where e ~ N(0,1)
and X is fixed.
set.seed(136344)
# X could be from any distribution (or even deterministic)
X <- runif(100, min = -2, max = 2)

### Generate Y using random e
Y <- 2 + 3*X + rnorm(100, mean = 0, sd = 1)
plot(X, Y)

### Plot the true regression line and the estimated one
abline(a = 2, b = 3, lwd = 3, col = 2)
abline(lm(Y ~ X), lwd = 2, col = "gray")
```

Variability of Estimates

```
### Write a function to automate this process
genDat <- function(R)
{
  ### R is the number of replications
  ### Create a matrix for storing output
  out <- matrix(0, R, 2)
  ### Plot the estimated regression lines
  for (i in 1:R) {
    if(i == 1) {plot(X, Y)}
    Y <- 2 + 3*X + rnorm(100, mean = 0, sd = 1)
    abline(lm(Y ~ X), lwd = 2, col = "gray")
    ### Record the intercepts and slopes for output
    out[i,] <- lm(Y ~ X)$coef
    }
  ### Plot the true regression line
  abline(a = 2, b = 3, lwd = 3, col = 2)
  ### Return the output
  out
}
```

How variable are the estimates of the intercept and slope



Standard Errors

- In order to formulate statistical tests of the regression coefficients, we need to quantify their variability.
- To do this we will use the assumptions that the error term has constant variance σ^2, that is, $\text{var}(\varepsilon_i) = \sigma^2$, and that the errors are uncorrelated with one another, that is, $\text{cov}(\varepsilon_j, \varepsilon_i) = 0$, for $i \neq j$.
- We will need a couple of facts:

$$\sum_{i=1}^{n} (x_i - \bar{x})\bar{y} = 0$$

$$\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y}) = \sum_{i=1}^{n} (x_i - \bar{x})(\beta_0 + \beta_1 x_i + u_i)$$

- Using these facts we can show that

$$\text{var}(\beta_1) = \frac{\sigma^2}{\sum_{i=1}^{n} (x_i - \bar{x})^2} \quad \text{and} \quad \text{var}(\beta_0) = \sigma^2 \left[\frac{1}{n} + \frac{\bar{x}^2}{\sum_{i=1}^{n} (x_i - \bar{x})^2} \right]$$

Standard Errors

- Finally, we need an estimator for σ^2. We use the following estimator:

$$\hat{\sigma}^2 = \frac{\sum_{i=1}^{n}\left(y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i\right)^2}{n-2}$$

because it is unbiased (we will not prove this).

- Plugging in the estimator, we get

$$\hat{\text{var}}\left(\hat{\beta}_1\right) = \frac{\hat{\sigma}^2}{\sum_{i=1}^{n}\left(x_i - \bar{x}\right)^2} \quad \text{and} \quad \hat{\text{var}}\left(\hat{\beta}_0\right) = \hat{\sigma}^2\left[\frac{1}{n} + \frac{\bar{x}^2}{\sum_{i=1}^{n}\left(x_i - \bar{x}\right)^2}\right]$$

- Note the hats on the variances now, indicating that it is no longer a theoretical identity, but now a quantity estimated from an actual sample of data.

Inverse Transform Method

- Method for generating a sample from a given distribution if you *know the formula* of the cdf $F(x)$

- Step 1: solve for x the equation $u = F(x)$
  That is, obtain $x = F^{-1}(u)$, where F^{-1} is the *inverse* function of $F(x)$

- Step 2: simulate $u \sim U(0,1)$
  That is, u is a random uniform variable

- Step 3: plug in the simulate u in the inverse equation from Step 1.
  That is, you have one simulated x from the desired distribution

- Step 4: Repeat steps 2 & 3 R times as needed

Example (Pr. 3.4 on p. 94): Inverse Transform Method

- Let's simulate a sample from the Rayleigh distribution:

$$f(x) = \frac{x}{\sigma^2} e^{-x^2/(2\sigma^2)}, \qquad x > 0$$
$$F(x) = 1 - e^{-x^2/(2\sigma^2)}, \qquad x > 0$$

- Step 1: solve for *x* the equation $u = F(x)$
  That is, obtain $x = F^{-1}\,(u)$, where F^{-1} is the *inverse* function of $F(x)$:

$$x = \sigma\sqrt{-2\log(1-u)}$$

- Step 2: simulate $u \sim$ U(0,1)
  That is, *u* is a random uniform variable

- Step 3: plug in the simulate *u* in the inverse equation from Step 1.
  That is, you have one simulated *x* from the desired distribution

- Step 4: Repeat steps 2 & 3 *R* times as needed