GOV 2000 Section 6:
Random Samples and Descriptive Inference (Regression)

Konstantin Kashin¹
Harvard University

October 10, 2012

¹These notes and accompanying code draw on the notes from Molly Roberts, Maya Sen, Iain Osgood, Brandon Stewart, and TF’s from previous years
Outline

Describing the Population

Estimating LCEF

Sampling Distributions

Hypothesis Testing
We are going to work with Leinhardt dataset from Leinhardt.RData.

- `lincome`: Log of per-capita income in U. S. dollars.
- `linfant`: Log of infant mortality rate per live births.
- `region`: A factor with levels: Africa; Americas; Asia, Asia and Oceania; Europe.
- `oil`: Oil-exporting country. A factor with levels: no, yes.

We want to regress log of infant mortality rate on log of per-capita income.
Data

We are going to work with Leinhardt dataset from Leinhardt.RData.
We are going to work with Leinhardt dataset from Leinhardt.RData.

- lincome: Log of per-capita income in U. S. dollars.
- linfant: Log of infant mortality rate per 1000 live births.
- region: A factor with levels: Africa; Americas; Asia, Asia and Oceania; Europe.
- oil: Oil-exporting country. A factor with levels: no, yes.
We are going to work with Leinhardt dataset from Leinhardt.RData.

- lincome: Log of per-capita income in U. S. dollars.
- linfant: Log of infant mortality rate per 1000 live births.
- region: A factor with levels: Africa; Americas; Asia, Asia and Oceania; Europe.
- oil: Oil-exporting country. A factor with levels: no, yes.

We want to regress log of infant mortality rate on log of per-capita income.
Relationship Between Economic Development and Infant Mortality

Log of per capita income (USD) vs. Log of infant-mortality rate per 1000 live births
**Population Linear Conditional Expectation Function**
Population Linear Conditional Expectation Function

\[ E[Y|X = x] = \beta_0 + \beta_1 x \]
Population Linear Conditional Expectation Function

\[ E[Y|X = x] = \beta_0 + \beta_1 x \]

In R:

```
lm(linfant ~ lincome, data=Leinhardt)
```
Population Linear Conditional Expectation Function

\[ E[Y|X = x] = \beta_0 + \beta_1 x \]

In R:

```r
lm(linfant ~ lincome, data=Leinhardt)
```

True population parameters:

\[ \beta_0 = 7.1458 \]
\[ \beta_1 = -0.5118 \]
Population Linear Conditional Expectation Function

Relationship Between Economic Development and Infant Mortality

Log of per capita income (USD)

Log of infant-mortality rate per 1000 live births
Outline

Describing the Population

Estimating LCEF

Sampling Distributions

Hypothesis Testing
Our Sample

Let’s take one sample of size 40 (without replacement) and run regression on it.
Let’s take one sample of size 40 (without replacement) and run regression on it.

```r
set.seed(01238)
my.samp <- Leinhardt[sample(nrow(Leinhardt), size = 40, replace = FALSE),]
lm.samp <- lm(linfant ~ lincome, data = my.samp)
```
Let’s take one sample of size 40 (without replacement) and run regression on it.

```r
set.seed(01238)
my.samp <- Leinhardt[sample(nrow(Leinhardt),size=40,
                         replace=FALSE),]
lm.samp <- lm(linfant ~ lincome, data=my.samp)
```

Estimated regression coefficients:
\[ \hat{\beta}_0 = 7.4462 \]
\[ \hat{\beta}_1 = -0.5660 \]
Estimated Linear Conditional Expectation Function

Relationship Between Economic Development and Infant Mortality

Log of per capita income (USD) vs Log of infant-mortality rate per 1000 live births.
Estimated Linear Conditional Expectation Function

Relationship Between Economic Development and Infant Mortality

Log of per capita income (USD) vs. Log of infant-mortality rate per 1000 live births.
This is what we get as regression output:

Coefficients:

|                | Estimate | Std. Error | t value | Pr(>|t|)  |
|----------------|----------|------------|---------|----------|
| (Intercept)    | 7.44624  | 0.41046    | 18.141  | < 2e-16  *** |
| lincome        | -0.56600 | 0.06412    | -8.827  | 9.72e-11 *** |

---

Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1 1
Regression Output

This is what we get as regression output:

Coefficients:

|               | Estimate | Std. Error | t value | Pr(>|t|)  |
|---------------|----------|------------|---------|----------|
| (Intercept)   |  7.44624 |  0.41046   | 18.141  | < 2e-16  *** |
| lincome       | -0.56600 |  0.06412   | -8.827  | 9.72e-11 *** |

---

Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1 1

Let’s focus on the standard errors for now...
Standard Errors of Regression Coefficients

- What do the standard errors of the regression coefficients represent?
What do the standard errors of the regression coefficients represent?

- Variability of sampling distributions of regression coefficients
What do the standard errors of the regression coefficients represent?

- Variability of sampling distributions of regression coefficients

How can we estimate the standard errors of the regression coefficients?
Standard Errors of Regression Coefficients

- What do the standard errors of the regression coefficients represent?
  - Variability of sampling distributions of regression coefficients

How can we estimate the standard errors of the regression coefficients?
- Theory
What do the standard errors of the regression coefficients represent?
  - Variability of sampling distributions of regression coefficients

How can we estimate the standard errors of the regression coefficients?
  - Theory
  - Boostrapping (resampling sample)
Outline

Describing the Population

Estimating LCEF

Sampling Distributions

Hypothesis Testing
Repeateed Sampling
Repeated Sampling

We can think of our sample as one of many possible samples we can draw from our population:
Repeated Sampling

We can think of our sample as one of many possible samples we can draw from our population:

<table>
<thead>
<tr>
<th>Samples from Pop.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td>$X_1, Y_1$</td>
</tr>
<tr>
<td>$X_2, Y_2$</td>
</tr>
<tr>
<td>$X_{40}, Y_{40}$</td>
</tr>
<tr>
<td>$(\bar{X}<em>{40}, \bar{Y}</em>{40})$</td>
</tr>
<tr>
<td>$(\bar{S}_X^2, \bar{Y}_X^2)$</td>
</tr>
<tr>
<td>$\hat{\beta}_0$</td>
</tr>
<tr>
<td>$\hat{\beta}_1$</td>
</tr>
</tbody>
</table>
Repeated Sampling

Now let’s take 10000 samples, each of size $n = 40$. For each sample, we will run a linear regression, as before. How do we do this in R?

First, create a matrix to hold the output:

```r
holder <- matrix(data = NA, ncol = 2, nrow = sims)
colnames(holder) <- c("intercept", "slope")
```

Now, using for loops:

```r
sims <- 10000
set.seed(02138)
for (i in 1: sims ) {
  my.samp <- Leinhardt[sample(nrow(Leinhardt), size = 40, replace = FALSE),]
  samp.lm <- lm(linfant ~ lincome, data = my.samp)
  holder[i,1] <- samp.lm$coefficients[1]
  holder[i,2] <- samp.lm$coefficients[2]
}
Now let’s take 10000 samples, each of size $n = 40$. For each sample, we will run a linear regression, as before. How do we do this in R?

First, create matrix that will hold output:

```r
holder <- matrix(data = NA, ncol = 2, nrow = sims)
colnames(holder) <- c("intercept", "slope")
```
**Repeated Sampling**

Now let’s take 10000 samples, each of size $n = 40$. For each sample, we will run a linear regression, as before. How do we do this in R?

First, create matrix that will hold output:

```r
holder <- matrix(data = NA, ncol = 2, nrow = sims)
colnames(holder) <- c("intercept", "slope")
```

Now, using for loops:
RePEATED SAMPLING

Now let’s take 10000 samples, each of size \( n = 40 \). For each sample, we will run a linear regression, as before. How do we do this in R?

First, create matrix that will hold output:

```r
holder <- matrix(data = NA, ncol = 2, nrow = sims)
colnames(holder) <- c("intercept", "slope")
```

Now, using for loops:

```r
sims <- 10000
set.seed(02138)
for(i in 1:sims){
  my.samp <- Leinhardt[sample(nrow(Leinhardt),size = 40,
                           replace=FALSE),]
  samp.lm <- lm(linfant ~ lincome, data=my.samp)
  holder[i,1] <- samp.lm$coefficients[1]
  holder[i,2] <- samp.lm$coefficients[2] }
```
More Efficient Repeated Sampling

But we know for loops are not the most efficient for this task, so how can we do this using `replicate()`?

```r
First, define function that we will replicate:

```sample_lm_fxn``` <- function () {
  my_samp <- Leinhardt [ sample ( nrow ( Leinhardt ), size = 40 , replace = FALSE ) ,
    samp_lm <- lm( linfant ~ lincome , data =my_samp )
    holder <- samp_lm
    $coefficients [1]
    holder [2] <- samp_lm
    $coefficients [2]
    return ( holder )
}

Now, tell `replicate()` to repeat the function above one.oldstyle/zero.oldstyle/zero.oldstyle/zero.oldstyle/zero.oldstyle times:

```
set.seed(02138)
results <- replicate (10000 , sample_lm_fxn ()))
```
But we know for loops are not the most efficient for this task, so how can we do this using `replicate()`? First, define function that we will replicate:

```r
sample.lm.fxn <- function() {
  my.samp <- Leinhardt[sample(nrow(Leinhardt), size=40, replace=FALSE),]
  samp.lm <- lm(linfant ~ lincome, data=my.samp)
  holder <- samp.lm$coefficients[1]
  return(holder)
}
```

Now, tell `replicate()` to repeat the function above one time:

```r
set.seed(02138)
results <- replicate(10000, sample.lm.fxn())
```
More Efficient Repeated Sampling

But we know for loops are not the most efficient for this task, so how can we do this using `replicate()`?

First, define function that we will replicate:

```r
sample.lm.fxn <- function() {
  my.samp <- Leinhardt[sample(nrow(Leinhardt), size=40, replace=FALSE),]
  samp.lm <- lm(linfant ~ lincome, data=my.samp)
  holder <- samp.lm$coefficients[1]
  return(holder)
}
```

Now, tell `replicate()` to repeat the function above 10000 times:
**More Efficient Repeated Sampling**

But we know for loops are not the most efficient for this task, so how can we do this using `replicate()`?

First, define function that we will replicate:

```r
sample.lm.fxn <- function() {
    my.samp <- Leinhardt[sample(nrow(Leinhardt), size=40, replace=FALSE),]
    samp.lm <- lm(linfant ~ lincome, data=my.samp)
    holder <- samp.lm$coefficients[1]
    return(holder)
}
```

Now, tell `replicate()` to repeat the function above 10000 times:

```r
set.seed(02138)
results <- replicate(10000, sample.lm.fxn())
```
Sampling Distributions for Regression Coefficients

```r
holder <- t(results)
par(mfrow = c(1,2))
plot(density(holder[,1]), col = "black",
    main = "Sampling Distribution for Intercept",
    xlab = expression(beta[0]))
abline(v = mean(holder[,1]), col="red", lwd=2)

plot(density(holder[,2]), col = "black",
    main = "Sampling Distribution for Slope",
    xlab = expression(beta[1]))
abline(v = mean(holder[,2]), col="red", lwd=2)
```
Means of the sampling distributions are plotted in red.
Now add the true, population regression parameters in blue:
Let’s plot the first 100 regression lines (in red) and population regression line (in blue):
We know, from theory, that the true sampling distribution of $\hat{\beta}_1$ is:

$$\hat{\beta}_1 \sim \mathcal{N}(\beta_1, \frac{\sigma^2}{\sum_i (x_i - \bar{x})^2})$$
Why might the standard error of the simulated sampling distribution not match the theoretical standard error?
Why might the standard error of the simulated sampling distribution not match the theoretical standard error?

From theory: \( SE(\beta_1) = 0.0507 \)
From simulation: \( SE(\beta_1) = 0.0623 \)
Why might the standard error of the simulated sampling distribution not match the theoretical standard error?

From theory: $SE(\beta_1) = 0.0507$
From simulation: $SE(\beta_1) = 0.0623$

Possible explanations:
- Sampling without replacement
- Heteroskedasticity / non-constant variance
**Estimating Sampling Distribution for \( \hat{\beta}_1 \)**

We can estimate sampling distribution for \( \hat{\beta}_1 \) as:

\[
\hat{\beta}_1 \sim \mathcal{N}(\beta_1, \frac{\hat{\sigma}^2}{\sum_i(x_i - \bar{x})^2})
\]
Estimating Sampling Distribution for $\hat{\beta}_1$

We can estimate sampling distribution for $\hat{\beta}_1$ as:

$$\hat{\beta}_1 \sim \mathcal{N}(\hat{\beta}_1, \frac{\hat{\sigma}^2}{\sum_i (x_i - \bar{x})^2})$$

$$\hat{\sigma}^2 = \frac{\sum_{i=1}^{n} u_i^2}{n - 2} = \frac{SSR}{n - 2}$$
We can estimate sampling distribution for $\hat{\beta}_1$ as:

$$\hat{\beta}_1 \sim N(\hat{\beta}_1, \frac{\hat{\sigma}^2}{\sum_i (x_i - \bar{x})^2})$$

$$\hat{\sigma}^2 = \frac{\sum_{i=1}^{n} u_i^2}{n-2} = \frac{SSR}{n-2}$$

To get estimated standard error of $\hat{\beta}_1$ in R:

```r
sigma2.hat <- sum(residuals(samp.lm)^2)/(samp.lm$df)
denom <- sum((my.samp$lincome - mean(my.samp$lincome))^2)
sqrt(sigma2.hat/denom)
```
OUTLINE

Describing the Population

Estimating LCEF

Sampling Distributions

Hypothesis Testing
This is what we get as regression output:

Coefficients:

| Estimate | Std. Error | t value | Pr(>|t|) |
|----------|------------|---------|----------|
| (Intercept) | 7.44624 | 0.41046 | 18.141 | < 2e-16 *** |
| lincome | -0.56600 | 0.06412 | -8.827 | 9.72e-11 *** |

---

Signif. codes: 0 *** 0.001 ** 0.01 * 0.05 . 0.1 1

How do we get the test statistic and the p-value?
The test statistic for the null hypothesis that $\beta_1 = 0$ is just given by:

$$T = \frac{\hat{\beta}_1 - \beta_{1,H_0}}{SE(\hat{\beta}_1)} = \frac{\hat{\beta}_1 - 0}{SE(\hat{\beta}_1)}$$

Moreover, we know that $T \sim t_{n-2}$ under the null.
Regression Output: Test Statistic

The test statistic for the null hypothesis that $\beta_1 = 0$ is just given by:

$$T = \frac{\hat{\beta}_1 - \beta_{1,H_0}}{SE(\hat{\beta}_1)} = \frac{\hat{\beta}_1 - 0}{SE(\hat{\beta}_1)}$$

Moreover, we know that $T \sim t_{n-2}$ under the null.
**Regression Output: Test Statistic**

The test statistic for the null hypothesis that $\beta_1 = 0$ is just given by:

$$T = \frac{\hat{\beta}_1 - \beta_{1,H_0}}{SE(\hat{\beta}_1)} = \frac{\hat{\beta}_1 - 0}{SE(\hat{\beta}_1)}$$

Moreover, we know that $T \sim t_{n-2}$ under the null.

In R:

```r
library(tidyverse)
t.stat <- (summary(samp.lm)$coefficients[2,1] - 0) / summary(samp.lm)$coefficients[2,2]
```

$$T_{obs} = -8.236$$
**Regression Output: Test Statistic**

The test statistic for the null hypothesis that $\beta_1 = 0$ is just given by:

$$T = \frac{\hat{\beta}_1 - \beta_{1,H_0}}{\hat{SE}(\hat{\beta}_1)} = \frac{\hat{\beta}_1 - 0}{\hat{SE}(\hat{\beta}_1)}$$

Moreover, we know that $T \sim t_{n-2}$ under the null.

In R:

```r
# Assuming samp.lm is the fitted model

# Extracting coefficients and standard errors
b1_hat <- coefficients(samp.lm)[2,1]
b1_true <- coefficients(samp.lm)[2,2]
b1_SE <- summary(samp.lm)$coefficients[2,2]

# Calculating the test statistic
T_stat <- (b1_hat - b1_true) / b1_SE

# Calculating the observed value
T_obs <- T_stat
```

$T_{obs} = -8.8269$
This is what we get as regression output:

Coefficients:

|                | Estimate | Std. Error | t value | Pr(>|t|)   |
|----------------|----------|------------|---------|------------|
| (Intercept)    | 7.44624  | 0.41046    | 18.141  | < 2e-16    |
| lincome        | -0.56600 | 0.06412    | -8.827  | 9.72e-11   |

---

Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1 1
**Regression Output: Test Statistic**

We can plot the observed test statistic on top of the sampling distribution of the test statistic, which is a t-distribution with $n - 2$ degrees of freedom:
The definition of the p-value is the probability of obtaining a value of the test statistic \( \text{at least as extreme} \) as the one you observed:

\[
p = P(|T| \geq |T_{obs}|) = 2 \cdot P(T \geq |T_{obs}|)
\]
The definition of the p-value is the probability of obtaining a value of the test statistic \textit{at least as extreme} as the one you observed:

\[ p = P(|T| \geq |T_{obs}|) = 2 \cdot P(T \geq |T_{obs}|) \]

In R:

\[ 2*pt(-\text{abs(t.stat)}, \text{df}=38) \]
The definition of the p-value is the probability of obtaining a value of the test statistic \textit{at least as extreme} as the one you observed:

\[ p = P(|T| \geq |T_{obs}|) = 2 \cdot P(T \geq |T_{obs}|) \]

In R:

\[
2*pt(-\text{abs}(t.stat), \text{df}=38)
\]

p-value = 9.7185e−11
This is what we get as regression output:

Coefficients:

|                                | Estimate | Std. Error | t value | Pr(>|t|)     |
|--------------------------------|----------|------------|---------|--------------|
| (Intercept)                    | 7.44624  | 0.41046    | 18.141  | < 2e-16 ***  |
| lincome                        | -0.56600 | 0.06412    | -8.827  | 9.72e-11 *** |

---

Signif. codes:   0 *** 0.001 ** 0.01 * 0.05 . 0.1 1

p-value is less than $\alpha = 0.001$, so we can reject at $\alpha = 0.001$ level!
Suppose we wanted a two-sided 95% confidence interval for $\beta_1$: 
Suppose we wanted a two-sided 95% confidence interval for $\beta_1$:

$$\hat{\beta}_1 \pm t_{\alpha/2} \cdot \hat{SE}[\hat{\beta}_1]$$
Suppose we wanted a two-sided 95% confidence interval for $\beta_1$:

$$\hat{\beta}_1 \pm t_{\alpha/2} \cdot \hat{SE}[^1]$$

In R:

```r
coef(samp.lm)[2] + qt(0.025, df=38) * summary(samp.lm)$coefficients[2,2]
coef(samp.lm)[2] - qt(0.025, df=38) * summary(samp.lm)$coefficients[2,2]
```
**Regression Output: Confidence Intervals**

Suppose we wanted a two-sided 95% confidence interval for $\beta_1$:

$$\hat{\beta}_1 \pm t_{\alpha/2} \cdot \hat{SE}[\hat{\beta}_1]$$

In R:

```r
coef(samp.lm)[2] + qt(0.025, df=38) * summary(samp.lm)$coefficients[2,2]
c coef(samp.lm)[2] - qt(0.025, df=38) * summary(samp.lm)$coefficients[2,2]
```

95% CI: $[-0.6958, -0.4362]$