Lab Notes

Chapter 4

OpenIntro Biostatistics

Overview

1. Sampling Variability
   - *OI Biostat* Section 4.1
2. Confidence Intervals
   - *OI Biostat* Section 4.2
3. Hypothesis Testing
   - *OI Biostat* Sections 4.3.1 - 4.3.2
4. Inference Concept Check
   - *OI Biostat* Sections 4.3.3 - 4.3.5

Lab 1 illustrates the idea of sampling variability through simulation and explores the relationship between a point estimate and population parameter.

Lab 2 introduces the calculation and interpretation of confidence intervals.

Lab 3 introduces the mechanics of formal hypothesis testing.

Lab 4 examines some conceptual details of inference, including the relationship between hypothesis tests and confidence intervals.

These labs demonstrate inference with the t-distribution, rather than the normal distribution. While using the normal distribution is a convenient approximation when doing calculations without access to software, R offers functions that compute confidence intervals and p-values based on the t-distribution. The t-distribution is formally introduced in Chapter 5 of the text.

Lab 1: Sampling Variability

The first part of the lab covers taking a single sample from yrbss. In the second part, many samples are taken and their means calculated via a for loop.

The key point to understand is that the subset yrbss.sample is created in two distinct steps:

1. Select the row numbers. The vector sample.rows is a random sample of 10 numbers from 1 to n, where n represents the total number of rows in the yrbss dataframe.
2. Extract the rows corresponding to the selected row numbers. The dataframe yrbss.sample is created using the bracket notation first introduced in the Chapter 1 Lab Notes.

Enclosing these two steps in the for loop (Question 3) allows for a different set of 10 numbers and thus, a different set of 10 observations, to be drawn with each iteration of the loop. The loop contains a slightly more compact version of the code shown in Question 1:

```
for(k in 1:replicates){

  sample.rows = sample(1:nrow(yrbss), sample.size)
  sample.means[k] = mean(yrbss$weight[sample.rows], na.rm = TRUE)

}
```

- The second line in the loop calculates the mean of the weight variable in yrbss, but only for the indices specified by the numbers in sample.rows. Note how no comma is needed in the bracket notation because yrbss$weight specifies a vector.
- One alternative way to write the line is as follows, where the outer object is still the yrbss matrix, and the bracket notation specifies the rows as sample.rows and the column as weight.

```
sample.means[k] = mean(yrbss[sample.rows, "weight"], na.rm = TRUE)
```

The function **abline()** is used to draw a straight line through a plot. It has the generic structure

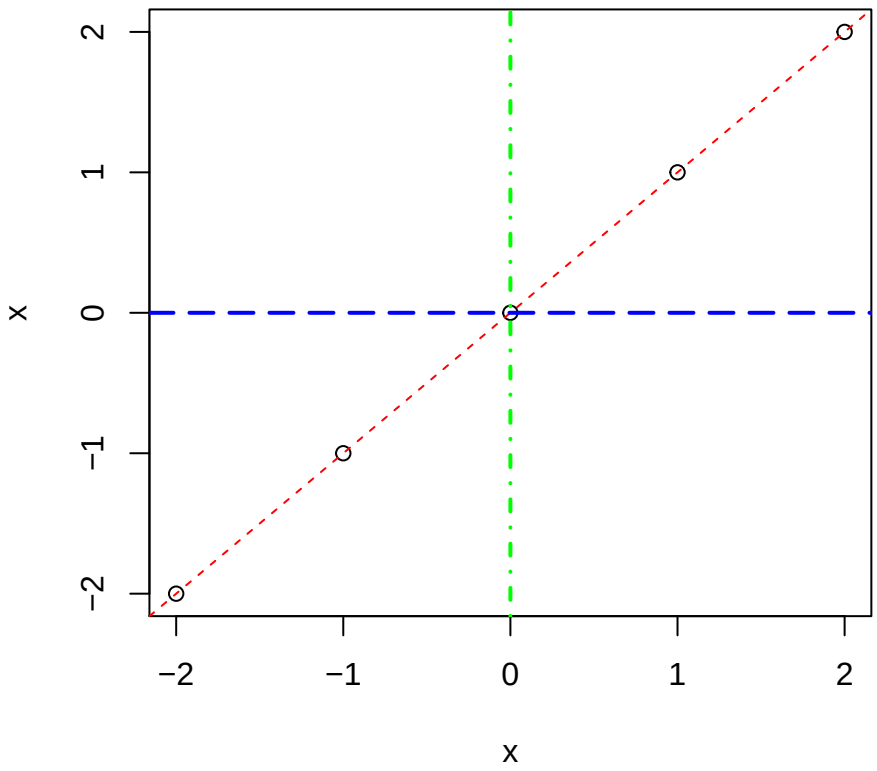
```
abline(a, b, h, v)
```

and additional parameters like color (col), line type (lty), and line width (lwd) can also be specified. The first two arguments specify an intercept and slope, respectively; h refers to the y-value for a horizontal line and v refers to the x-value for a vertical line.

The following plot illustrates how abline() can be used.

```
#draw a plot
x = c(-2, -1, 0, 1, 2)
plot(x , x)

#draw lines
abline(a = 0, b = 1, col = "red", lty = 2)
abline(h = 0, col = "blue", lty = 5, lwd = 2)
abline(v = 0, col = "green", lty = 4, lwd = 2)
```


x
2
1
0
−1
−2
−2
−1
0
1
2
x

Lab 2: Confidence Intervals

The simulation code from the previous lab is expanded in this lab. For each sample, not only is the mean calculated, but also the standard deviation; these values are then used to compute a confidence interval for each sample.

Unlike the previous simulation which drew samples from `yrbss`, this simulation draws only from the rows of `yrbss` for which a `weight` value has been recorded. This simplifies the calculation of confidence intervals, since the sample size will be consistent from sample to sample rather than being affected by missing data values.

The **`complete.cases()`** function returns a logical vector specifying which observations or rows have no missing values. The following example illustrates how `complete.cases` returns `TRUE` for the first three values in the vector `x` and `FALSE` for the last two values; bracket notation is then used to extract the values of `x` for which the values are not missing (i.e., the cases are complete).

Generally, missing data should be handled with care, since data that is selectively missing from a particular subset may mean that the complete cases do not represent the target population. If, for instance, a large number of adolescents who were overweight were more likely to refuse having their weight recorded, the average weight in the complete cases would be too small. Since sophisticated methods for conducting inference in the presence of missing data are beyond the scope of this course, however, this lab uses complete cases for instructional purposes.

```
#create data vector
x = c(50, 3, 1.2, NA, NA)

#view output of complete.cases()
complete.cases(x)
```

```
## [1]  TRUE  TRUE  TRUE FALSE FALSE
```

```
#extract non-missing values of x
x[complete.cases(x)]
```

```
## [1] 50.0  3.0  1.2
```

The **`t.test()`** function performs a t-test; its arguments will be explained in more detail in the next section. In this lab, the function is used to compute a confidence interval with a specific confidence level. It has the following generic structure, where the first argument is a numeric vector of data values, and the confidence level is entered as a decimal. The `$conf.int` syntax is used to print out only the confidence interval from the complete output.

```
t.test(x, conf.level = 0.95)$conf.int
```

The `for` loop in Question 5 stores output in two vectors, `sample.means` and `m`; `sample.means` is a vector of the 1,000 sample means, and `m` is a vector of the margin of error calculated from each sample, where $m = z^\star \times s/\sqrt{n}$. These two vectors are used to generate the upper and lower bounds of the confidence interval for each sample.

Each confidence interval is then checked for whether or not it contains `mu`, the mean weight in the `yrbss.complete` population. The logical vector `contains.mu` returns `TRUE` if `mu` is both greater than the lower bound and less than the upper bound.

Lab 3: Hypothesis Testing

This lab introduces the essential functions for conducting hypothesis tests (for a population mean) with R. The t-distribution functions are useful for hand calculations, while the `t.test()` function performs hypothesis tests directly on data.

T Distribution Functions

The function **`pt()`** used to calculate $P(X \leq k)$ or $P(X > k)$ has the generic structure

```
pt(q, df, lower.tail = TRUE)
```

where `q` is k and `df` is the degrees of freedom. By default (`lower.tail = TRUE`), R calculates $P(X \leq k)$. In order to compute $P(X > k)$, specify `lower.tail = FALSE`.

The following code shows how to calculate $P(X \leq 1.20)$ and $P(X > 1.20)$ for $X \sim t_{df=20}$.

```
#probability X is less than (or equal to) 1.20
pt(1.20, df = 20)
```

```
## [1] 0.8779192
```

```
#probability X is greater than 1.20
pt(1.20, df = 20, lower.tail = FALSE)
```

```
## [1] 0.1220808
```

The function **`qt()`** used to identify the observation that corresponds to a particular probability p has the generic structure

```
qt(p, df, lower.tail = TRUE)
```

where `p` is p and `df` is the degrees of freedom. By default (`lower.tail = TRUE`), R identifies the observation that corresponds to area p in the lower tail (i.e., to the left). To identify the observation with area p in the upper tail, specify `lower.tail = FALSE`.

The following code shows how to calculate the value of the observation with 0.841 area to the left (and 0.159 area to the right) on a t-distribution with 20 degrees of freedom.

```
#identify X value
qt(0.841, df = 20)
```

```
## [1] 1.024135
```

```
qt(0.159, df = 20, lower.tail = FALSE)
```

```
## [1] 1.024135
```

Hypothesis Testing with `t.test()`

The **`t.test()`** function has the following generic structure:

```
t.test(x, alternative = "two.sided", mu = , conf.level = 0.95)
```

where `x` is a numeric vector of data values, `alternative` specifies the form of the alternative hypothesis, `mu` is μ_0, and `conf.level` refers to the confidence level. The argument for `alternative` can be either `"two.sided"` ($H_A : \mu \neq \mu_0$), `"less"` ($H_A : \mu < \mu_0$), or `"greater"` ($H_A : \mu > \mu_0$). By default, confidence level is set to 95% and a two-sided alternative is tested.

The following example shows a hypothesis test for mean standing height in centimeters in the artificial NHANES population, using a random sample of 135 adults. The null hypothesis is that the population mean height is equal to 168 cm (a little over 5 feet, 6 inches). A one-sided alternative is tested against the null, $H_A : \mu > 168$ cm; the output includes the t-statistic, degrees of freedom, p-value, 90% confidence interval, and the sample mean of the data.

```
#load the data
library(oibiostat)
data("nhanes.samp.adult")

#conduct test
t.test(nhanes.samp.adult$Height, mu = 168, alternative = "greater", conf.level = 0.90)
```

```
##
##  One Sample t-test
##
## data:  nhanes.samp.adult$Height
## t = 1.737, df = 134, p-value = 0.04234
## alternative hypothesis: true mean is greater than 168
## 90 percent confidence interval:
##  168.3846      Inf
## sample estimates:
## mean of x
##  169.4874
```

The output of `t.test()` is organized as a list object, and so specific pieces can be extracted using the dollar sign (`$`) and the name of the desired component. The possible components include the t-statistic (`statistic`), degrees of freedom (`parameter`), p-value (`p.val`), sample mean (`estimate`), and μ_0 (`null.value`).

The following examples show the t-statistic and p-value being selectively output from a test of the two-sided alternative against the null, using the same data as the previous example.

```
t.test(nhanes.samp.adult$Height, mu = 168)$statistic
```

```
##        t
## 1.736988
```

```
t.test(nhanes.samp.adult$Height, mu = 168)$p.val
```

```
## [1] 0.08468723
```

Lab 4: Inference Concept Check

This lab takes a closer look at some concepts in inference that may be confusing or seem counter-intuitive. The labs in the next unit will continue exploring conceptual details in the two-sample context.

The first section examines the relationship between a hypothesis test and a confidence interval associated with the same significance level (e.g., $\alpha = 0.05$ and 95% confidence) by illustrating how the margin of error is the common quantity for both tests and intervals.

The second section provides practice calculating and interpreting one-sided hypothesis tests and confidence intervals. Two important take-away points from Question 4: 1) statistical significance does not automatically imply practical significance, 2) care should be exercised when calculating one-sided p-values, because the area corresponding to the p-value is in the direction specified by the alternative hypothesis, and this area will not always be the smaller tail area.

The third section features an example of a scenario where context informs whether a one-sided or or two-sided test is more appropriate. The example also illustrates the terminology of a two-sided test as more "conservative" than a one-sided test.

The fourth section uses simulation to explore the idea of Type I error and α as the probability of making a Type I error. As in the previous simulation, samples are drawn repeatedly from `yrbss.complete`. A t-statistic is calculated from each sample to test the null hypothesis $H_0 : \mu =$ 67.91 kg. The simulation can be thought of as a simulation "under the null", in the sense that it is known that the null hypothesis is true: in the artificial `yrbss.complete` population, the mean weight is 67.91 kg. Thus, any samples that produce extreme t-statistics and indicate that the null should be rejected represent instances of making a Type I error. When the rejection region is defined by $\alpha = 0.05$, on average, 5% of the samples will have extreme t-statistics even though the population mean is not different from 67.91 kg. The simulation explicitly demonstrates that this error percentage changes as α changes.