Medical Statistics – Lab 2

R version

Welcome to lab 2 in the medical statistics course. For today's exercises, we will continue exploring the `lowbwt.sav` dataset.

```r
library(haven)
library(dplyr)
library(ggplot2)

# Load the dataset
lowbwt <- read_sav("lowbwt.sav")

# Convert alll labelled variables into factor variables
lowbwt <- lowbwt %>% mutate(across(where(is.labelled), as_factor))
```

> **Working Directory**
>
> The code above assumes that the file `lowbwt.sav` is in your current working directory. If you encounter an error saying the file cannot be found, you have two options:
>
> 1. Set your working directory to the folder containing the dataset using `Session > Set Working Directory > To Source File Location` in RStudio, or
> 2. Use an absolute path to the file, for example: `read_sav("C:/Users/YourName/Documents/datasets/lowbwt.sav")`

As a reminder, the dataset includes the following variables (see the previous lab for more details):

Variable	Abbreviation
Identification Code	ID
Low Birth Weight (0 = Birth Weight 2500g, 1 = Birth Weight < 2500g)	low

Variable	Abbreviation
Age of the Mother in Years	age
Weight in Pounds at the Last Menstrual Period	lwt
Ethnicity (1 = Caucasian, 2 = Afro-American, 3 = Asian)	ethnicity
Smoking Status During Pregnancy (1 = Yes, 0 = No)	smoke
History of Premature Labor (0 = None, 1 = One, etc.)	ptl
History of Hypertension (1 = Yes, 0 = No)	ht
Presence of Uterine Irritability (1 = Yes, 0 = No)	urirr
Number of Physician Visits During the First Trimester (0 = None, 1 = One, 2 = Two, etc.)	pvft
Birth Weight in Grams	bwt

Point Estimates and 95% Confidence Intervals for Population Means

We will start by analyzing the variable 'birth weight in grams' (bwt), which is the main outcome of this study.

To calculate the mean, standard deviation, and standard error of the mean, you can use the following R code:

```
bwt_summary <- lowbwt |>
  summarise(
    n = sum(!is.na(bwt)),
    mean = mean(bwt, na.rm = TRUE),
    sd = sd(bwt, na.rm = TRUE),
    se = sd / sqrt(n)
  )

# Convert to data frame to display more decimal places
as.data.frame(bwt_summary)
```

> **i** Sequential Calculation in summarise()
>
> Notice that in the `summarise()` function above, the `se` calculation references both `sd` and `n` that were defined earlier in the same call. The `summarise()` function evaluates expressions sequentially from top to bottom, making newly created variables available for

use in subsequent calculations within the same function call. This sequential evaluation also applies to other dplyr functions like `mutate()`, which we will use later in this lab.

Question 1

Based on these summary statistics, what is the estimated mean birth weight for the population?

Question 2

Calculate the corresponding 95% confidence interval based on the normal approximation.

You can also use R to calculate the 95% confidence interval by using the `qt()` function to determine the appropriate t-value. This approach uses the t-distribution, which provides a more accurate confidence interval when the population standard deviation is unknown and the sample size is small.

```
bwt_ci <- bwt_summary |>
  mutate(
    t_value = qt(0.975, df = n - 1),
    lower_ci = mean - t_value * se,
    upper_ci = mean + t_value * se
  )

as.data.frame(bwt_ci)
```

Explanation:

The `qt()` function in R is used to obtain the critical value from the t-distribution. In this case, we use `qt(0.975, df = n - 1)` to get the t-value for a 95% confidence interval, where `0.975` corresponds to the upper tail probability for a two-sided confidence level of 95%, and `df = n - 1` represents the degrees of freedom (sample size minus one).

Question 3

How does the 95% confidence interval based on the t-distribution compare to the 95% confidence interval based on the normal approximation that you manually computed?

One-Sample t-Test

To determine whether the population mean birth weight differs significantly from a hypothesized value of 3000 grams, we use the `t.test()` function to conduct a one-sample t-test:

```r
t_test_result <- t.test(lowbwt$bwt, mu = 3000)
t_test_result
```

> Question 4
>
> You see that the test has 188 degrees of freedom. Why?

> Question 5
>
> Based on the results of the test, does the population mean significantly differ from 3000?

One of the assumptions underlying the one-sample t-test is that the data are normally distributed. We can check this assumption by creating a histogram:

```r
ggplot(lowbwt, aes(x = bwt)) +
  geom_histogram(bins = 20, fill = "blue", color = "black") +
  labs(title = "Histogram of Birth Weights",
    x = "Birth Weight (grams)",
    y = "Frequency")
```

> Question 6
>
> Looking at the histogram, would you say that the data are normally distributed?

Point Estimates and 95% Confidence Intervals for Population Proportions

Next, we will explore the variable 'low birth weight' (low), which is a dichotomous variable with the value 'yes (bwt $<$ 2500)' if the baby had a low birth weight (defined as a birth weight $<$ 2500g) and a value of 'no (bwt $>=$ 2500 g)' otherwise.

We start by using the `table()` function to calculate the frequency of each category of the 'low' variable:

```r
freq_table <- table(lowbwt$low)
freq_table
```

> Question 7
>
> Based on these frequencies, what is the estimated proportion of low birth weight babies in the population?

Question 8

Calculate the corresponding 95% confidence interval based on the Normal approximation.

Binomial Test

Subsequently, we perform an exact binomial test to assess whether the proportion of low birth weight babies in the population differs significantly from a hypothesized value of 30%. The `binom.test()` function requires two key pieces of information:

- `x`: the number of "successes" (in this case, babies with low birth weight)
- `n`: the total number of observations

We can extract these values from the frequency table we created earlier:

```
# Extract the number of low birth weight babies (k)
k <- freq_table["yes (bwt < 2500)"]

# Extract the total number of observations (n)
n <- sum(freq_table)

# Perform the binomial test
binom_test_result <- binom.test(x = k, n = n, p = 0.30)
binom_test_result
```

Explanation:

- `freq_table["yes (bwt < 2500)"]` extracts the count of babies with low birth weight from our frequency table
- `sum(freq_table)` calculates the total number of observations by summing all frequencies
- `binom.test(x = k, n = n, p = 0.30)` performs the exact binomial test, where `p = 0.30` specifies the hypothesized population proportion (30%)

The output displays, among other statistics, the two-sided p-value and an exact 95% confidence interval calculated using the Clopper and Pearson procedure.

Question 9

Does the proportion of low birth weight babies differ significantly from 30%?

Question 10

The Dutch government intends to start a campaign against drinking alcoholic beverages if over 50% of the adolescents drink alcoholic beverages regularly (at least once a week). A random sample of 200 adolescents is taken and 128 admit that they drink alcohol regularly (we assume all 200 speak the truth). Test the null hypothesis that 50% of the Dutch adolescents drink alcohol, using a significance level of 5%. Use the exact binomial test for this question.

Question 11

Rather than using an exact binomial test, we can also use the normal approximation of the binomial distribution to obtain an approximate p-value for the above hypothesis test. Manually calculate this approximate p-value and compare it to the p-value obtained from the binomial test. Is the use of the normal approximation appropriate in this case?

Differences in Two-Sided P-Value Calculation Between SPSS and R

When conducting statistical tests, it is important to understand that different software packages can calculate two-sided p-values in slightly different ways, which may lead to variations in results. A key difference exists between how SPSS and base R handle this calculation:

- SPSS often calculates two-sided p-values by doubling the one-sided p-value. Specifically, SPSS determines the probability of the observed outcome in one direction (greater or less than a given value) and then multiplies this value by 2. This approach assumes that the distribution of the test statistic is symmetric under the null hypothesis. While this method is straightforward, it can be misleading if the distribution is skewed or the sample size is small, as it may not fully account for the asymmetry in the data.
- Base R (e.g., the `binom.test()` function) uses a more exact method for calculating two-sided p-values. R's approach sums the probabilities of observing outcomes that are as extreme as, or more extreme than, the observed value in both directions (both tails of the distribution). This method does not assume symmetry and provides a more accurate p-value, particularly for small samples or skewed distributions.