Tutorial 06: One Sample Hypothesis Tests

```
#| edit: false
#| output: false
webr::install("gradethis", quiet = TRUE)
library(gradethis)
options(webr.exercise.checker = function(
  label, user_code, solution_code, check_code, envir_result, evaluate_result,
  envir_prep, last_value, engine, stage, ...
) {
  if (is.null(check_code)) {
    # No grading code, so just skip grading
    invisible(NULL)
  } else if (is.null(label)) {
    list(
      correct = FALSE,
      type = "warning",
      message = "All exercises must have a label."
    )
  } else if (is.null(solution_code)) {
    list(
      correct = FALSE,
      type = "warning",
      message = htmltools::tags$div(
        htmltools::tags$p("A problem occurred grading this exercise."),
        htmltools::tags$p(
          "No solution code was found. Note that grading exercises using the ",
          htmltools::tags$code("gradethis"),
          "package requires a model solution to be included in the document."
        )
      )
    )
  } else {
    gradethis::gradethis_exercise_checker(
      label = label, solution_code = solution_code, user_code = user_code,
      check_code = check_code, envir_result = envir_result,
      evaluate_result = evaluate_result, envir_prep = envir_prep,
      last_value = last_value, stage = stage, engine = engine)
  }
})
```

Q1 — One Sample Hypothesis Test on the Mean (known)

For this question, we use the 2014 Soccer World Cup Tournament Predictions dataset. Test whether the average Soccer Power Index (spi) for a team equals 70 when the population standard deviation is known to be = 12.

We test

H : = 70

H :   70

Use a two-sided z-test with known.

Photo by CamYogi on Unsplash

Info

This is a one-sample test on a single numeric variable where the population standard deviation is assumed to be known. In that situation, we use a z-based procedure and compare the sample mean to the hypothesized value.

Preview

Run this code chunk to get a glimpse of the dataset. Feel free to change the values to visualize more/less number of rows.

```
#| echo: true
df <- read.csv("world_cup_predictions.csv")
head(df[, c("country","group","spi")], 10)
summary(df$spi)

par(mfrow = c(1,3))
hist(df$spi, main = "Histogram of SPI", xlab = "spi")
boxplot(df$spi, main = "Boxplot of SPI", ylab = "spi")
qqnorm(df$spi, main = "Q-Q Plot of SPI"); qqline(df$spi)
par(mfrow = c(1,1))
```

```
#| exercise: q1_ztest_spi
#| exercise.lines: 6
#| echo: false
df <- read.csv("world_cup_predictions.csv", stringsAsFactors = FALSE, check.names = FALSE)
spi_col <- df$spi #column vector for spi
mu_0 <-   #enter hypothesized mean
sigma <-
n <- sum(!is.na(spi_col)) #total number of teams
xbar <-
z <-  z-critical value
p_two <- #calculate 2-sided p-value

c(z = z, p_value = p_two)
```

Confirm you're comparing the means of two groups formed by a boolean condition on stars. Use the pooled-variance standard error and the appropriate two-sided t critical value for a 95% interval. Print just the lower and upper bounds as a numeric vector.

Solution.

```
#| exercise: q1_ztest_spi
#| solution: true
df <- read.csv("world_cup_predictions.csv", stringsAsFactors = FALSE, check.names = FALSE)
spi_col <- df$spi
mu_0 <- 70
sigma <- 12
n <- sum(!is.na(spi_col)) #total number of teams
xbar <- mean(spi_col, na.rm=TRUE)
z <- (xbar - mu_0) / (sigma / sqrt(n))
p_two <- 2 * (1 - pnorm(abs(z)))
```

```
c(z = z, p_value = p_two)
```

```
#| exercise: q1_ztest_spi
#| check: true
gradethis::grade_this({
df <- read.csv("world_cup_predictions.csv", stringsAsFactors = FALSE, check.names = FALSE)
spi_col <- df$spi; mu_0 <- 70; sigma <- 12
n <- sum(!is.na(spi_col))
xbar <- mean(spi_col, na.rm=TRUE)
exp_z <- (xbar - mu_0)/(sigma/sqrt(n))
exp_p <- 2*(1 - pnorm(abs(exp_z)))
r <- .result
ok <- is.numeric(r) && all(c("z","p_value") %in% names(r))
if (!ok) fail("Return a named numeric c(z=..., p_value=...).")
if (max(abs(r["z"] - exp_z), abs(r["p_value"] - exp_p)) < 1e-6) pass("  Correct z and p-value
else fail("Recheck x̄, n,  , and the two-sided p-value.")
})
```

Q2 — One-sample z-test on SPI (known, using BSDA)

For this question, we again use the 2014 Soccer World Cup Tournament Predictions dataset. We want to test whether the average Soccer Power Index (spi) for all teams equals 70 when the population standard deviation is known to be 12. This time we will use the z.test() function from the BSDA package to run the test in a single line, and you will just fill in the key arguments.

Info

Base R does not have a built-in one-sample z-test. The BSDA package provides z.test() which does: compute the z statistic, use the normal distribution, and give the p-value. We are using it here only to shorten the code; the logic is the same as in Q1.

Preview

```
#| echo: true
df <- read.csv("world_cup_predictions.csv")

head(df[, c("country","group","spi")], 10)

par(mfrow = c(1,2))
hist(df$spi, main = "Histogram of SPI", xlab = "spi")
boxplot(df$spi, main = "Boxplot of SPI", ylab = "spi")
par(mfrow = c(1,1))
```

```
#| exercise: q2_spi_z_bsda
#| exercise.lines: 8
#| echo: false

df <- read.csv("world_cup_predictions.csv")
x <- df$spi

library(BSDA)

out <- z.test(x,
mu =   ,       # hypothesized mean
sigma.x =   ,  # known population SD
alternative = "two.sided")

out$p.value
```

Use the same hypotheses as before (test against 70) and the same known   (12). Keep the test two-sided.

Solution.

```
#| exercise: q2_spi_z_bsda
#| solution: true

df <- read.csv("world_cup_predictions.csv")
x <- df$spi

library(BSDA)
```

```
out <- BSDA::z.test(x,
mu = 70,
sigma.x = 12,
alternative = "two.sided")

out$p.value
```

```
#| exercise: q2_spi_z_bsda
#| check: true
gradethis::grade_this({
df <- tryCatch(read.csv("world_cup_predictions.csv"), error = function(e) NULL)
if (is.null(df)) fail("Couldn't read 'world_cup_predictions.csv'. Make sure the file is avail
if (!"spi" %in% names(df)) fail("The CSV must contain a 'spi' column.")
x <- df$spi

suppressMessages(library(BSDA))
exp_out <- BSDA::z.test(x,
mu = 70,
sigma.x = 12,
alternative = "two.sided")
exp_p <- exp_out$p.value

res <- .result
if (!is.numeric(res) || length(res) != 1L) {
fail("Print a single numeric p-value from the test output.")
}

if (is.finite(res) && abs(res - exp_p) < 1e-6) {
pass("Correct p-value for the BSDA z-test with   = 70 and   = 12.")
} else {
fail("Not quite - check that you used mu = 70, sigma.x = 12, and alternative = 'two.sided'."
}
})
```

Q3 — One-sample test on SPI (unknown)

Now test the same hypothesis,

H :   = 70

H :   70

but now do not assume the population standard deviation is known. Use the sample SD, form the t statistic, and get the two-sided p-value from the t distribution.

Info

When is unknown, we use the sample SD to standardize and compare to a t distribution with n − 1 degrees of freedom.

Preview

Run this code chunk to get a glimpse of the dataset. Feel free to change the values to visualize more/less number of rows.

```
#| echo: true
df <- read.csv("world_cup_predictions.csv")
x <- df$spi

par(mfrow = c(1,2))
qqnorm(x, main = "Q-Q plot of SPI"); qqline(x)
hist(x, main = "Histogram of SPI", xlab = "spi")
par(mfrow = c(1,1))
```

Info

For a one-sample test with unknown σ, use $t = \dfrac{\bar{x} - \mu_0}{s/\sqrt{n}}$ with $df = n - 1$, and a two-sided p-value $p = 2 \times (1 - F_t(|t|))$.

```
#| exercise: q2_spi_ttest_unknown_sigma
#| exercise.lines: 10
#| echo: false

df <- read.csv("world_cup_predictions.csv")
spi_col <- df$spi # numeric SPI data

mu_0 <-   # hypothesized mean (70)
n <- sum(!is.na(spi_col)) # sample size
xbar <-  (spi_col, na.rm = TRUE) # sample mean
s <-  (spi_col, na.rm = TRUE) # sample standard deviation

dfree <- n - 1
```

```
t_stat <- (xbar - mu_0) / (s / sqrt(n))

p_two <- 2 * (1 - pt(abs(t_stat), df = dfree))

c(t = t_stat, df = dfree, p_value = p_two)
```

Use the SPI column, test against 70, compute mean and sd, then use the t formula and a two-sided p-value from pt().

Solution.

```
#| exercise: q2_spi_ttest_unknown_sigma
#| solution: true
df <- read.csv("world_cup_predictions.csv")
spi_col <- df$spi

mu_0 <- 70
n <- sum(!is.na(spi_col))
xbar <- mean(spi_col, na.rm = TRUE)
s <- sd(spi_col, na.rm = TRUE)

dfree <- n - 1
t_stat <- (xbar - mu_0) / (s / sqrt(n))
p_two <- 2 * (1 - pt(abs(t_stat), df = dfree))

c(t = t_stat, df = dfree, p_value = p_two)
```

```
#| exercise: q2_spi_ttest_unknown_sigma
#| check: true
gradethis::grade_this({
df <- tryCatch(read.csv("world_cup_predictions.csv"), error = function(e) NULL)
if (is.null(df)) fail("Couldn't read 'world_cup_predictions.csv'.")
if (!"spi" %in% names(df)) fail("Need a 'spi' column.")

x <- df$spi
mu_0 <- 70
n <- sum(!is.na(x))
xbar <- mean(x, na.rm = TRUE)
s <- sd(x, na.rm = TRUE)
dfree <- n - 1

exp_t <- (xbar - mu_0) / (s / sqrt(n))
exp_p <- 2 * (1 - pt(abs(exp_t), df = dfree))

res <- .result
if (!is.numeric(res) || length(res) != 3L) fail("Return c(t = ..., df = ..., p_value = ...).
if (any(!is.finite(res))) fail("Values must be finite.")

t_user <- unname(res[1])
df_user <- unname(res[2])
p_user <- unname(res[3])

if (abs(t_user - exp_t) < 1e-6 &&
abs(df_user - dfree) < 1e-6 &&
abs(p_user - exp_p) < 1e-6) {
pass("Correct manual one-sample t-test.")
} else {
fail("Something is off - check mean, sd, df = n - 1, and the two-sided p-value using pt().")
}
})
```

Q4 — One-sample t-test on SPI (unknown, built-in)

For this question, we again use the 2014 Soccer World Cup Tournament Predictions dataset. We want to test whether the average Soccer Power Index (spi) for all teams equals 70, but we will let R do the one-sample t-test for us using the built-in t.test() function. This is the "practical" version of Q3.

> **Info**
>
> When the population standard deviation is not given, we use a one-sample t-test: t.test(x, mu =   , alternative = "two.sided") R will estimate the standard deviation, use df = n − 1, and return the p-value.

```
#| exercise: q4_spi_t_builtin
#| exercise.lines: 8
#| echo: false

df <- read.csv("world_cup_predictions.csv")
x <-

out <- t.test( ,
mu =  ,
alternative = "two.sided")

out$p.value
```

Use the SPI column from the data, test it against 70, and keep the test two-sided.

Solution.

```
#| exercise: q4_spi_t_builtin
#| solution: true

df <- read.csv("world_cup_predictions.csv")
x <- df$spi
out <- t.test(x, mu = 70, alternative = "two.sided")
out$p.value
```

```
#| exercise: q4_spi_t_builtin
#| check: true
gradethis::grade_this({
df <- tryCatch(read.csv("world_cup_predictions.csv"), error = function(e) NULL)
if (is.null(df)) fail("Couldn't read 'world_cup_predictions.csv'.")
if (!"spi" %in% names(df)) fail("The CSV must contain a 'spi' column.")
x <- df$spi

exp_out <- t.test(x, mu = 70, alternative = "two.sided")
exp_p <- exp_out$p.value

res <- .result
if (!is.numeric(res) || length(res) != 1L) {
fail("Print a single numeric p-value from the t-test.")
}

if (abs(res - exp_p) < 1e-6) {
pass("Correct p-value for the one-sample t-test on SPI.")
} else {
fail("Not quite - check that you used mu = 70 and alternative = 'two.sided'.")
}
})
```

Q5 — One-sample t-test on Track Score (unknown, manual, large n)

For this question, use the large music dataset. We want to test whether the average track score in this dataset equals 45.

H :   = 45 H :   45

We will compute the test statistic and the p-value manually.

Info

One-sample t (unknown):
$t = \frac{\bar{x}-\mu_0}{s/\sqrt{n}}$, with df = n-1, and two-sided p-value $p = 2 \times (1 - F_t(|t|))$.

Preview

Run this code chunk to get a glimpse of the dataset. Feel free to change the values to visualize more/less number of rows.

```
#| echo: true
df <- read.csv("spotify-2024.csv")

look at the columns

names(df)

quick peek at Track.Score

summary(df$Track.Score)

par(mfrow = c(1,2))
hist(df$Track.Score, main = "Histogram of Track Score", xlab = "Track.Score")
boxplot(df$Track.Score, main = "Boxplot of Track Score", ylab = "Track.Score")
par(mfrow = c(1,1))
```

```
#| exercise: q5_trackscore_t_manual
#| exercise.lines: 14
#| echo: false
df <- read.csv("spotify-2024.csv")

numeric variable

x <-
```

```
mu_0 <-   # 45
x <- x[is.finite(x)] # drop NAs
n <- length(x)
xbar <-  (x) # mean
s <-  (x) # sample sd

dfree <- n - 1

t_stat <- (xbar - mu_0) / (s / sqrt(n))

p_two <- 2 * (1 - pt(abs(t_stat), df = dfree))

c(t = t_stat, df = dfree, p_value = p_two)
```

Use Track.Score as x, use mean() and sd() before plugging into the t formula. Make it two-sided.

Solution.

```
#| exercise: q5_trackscore_t_manual
#| solution: true

df <- read.csv("spotify-2024.csv")
x <- df$Track.Score
x <- x[is.finite(x)]

mu_0 <- 45
n <- length(x)
xbar <- mean(x)
s <- sd(x)

dfree <- n - 1
t_stat <- (xbar - mu_0) / (s / sqrt(n))
p_two <- 2 * (1 - pt(abs(t_stat), df = dfree))

c(t = t_stat, df = dfree, p_value = p_two)
```

```
#| exercise: q5_trackscore_t_manual
#| check: true
gradethis::grade_this({
df <- tryCatch(read.csv("spotify-2024.csv"), error = function(e) NULL)
if (is.null(df)) fail("Couldn't read 'spotify-2024.csv'.")
if (!"Track.Score" %in% names(df)) fail("The CSV must contain a 'Track.Score' column.")

x <- df$Track.Score
x <- x[is.finite(x)]

mu_0 <- 45
n <- length(x)
xbar <- mean(x)
s <- sd(x)
dfree <- n - 1

exp_t <- (xbar - mu_0) / (s / sqrt(n))
exp_p <- 2 * (1 - pt(abs(exp_t), df = dfree))

res <- .result
if (!is.numeric(res) || length(res) != 3L) {
fail("Return c(t = ..., df = ..., p_value = ...).")
}

t_user <- unname(res[1])
df_user <- unname(res[2])
p_user <- unname(res[3])

if (abs(t_user - exp_t) < 1e-6 &&
abs(df_user - dfree) < 1e-6 &&
abs(p_user - exp_p) < 1e-6) {
pass("Correct manual one-sample t-test for Track.Score.")
} else {
fail("Check mean, sd, df = n - 1, and the two-sided p-value.")
}
})
```

Q6 — One-sample t-test on Track Score (unknown, built-in)

Now run the same hypothesis test using R’s built-in t.test() on the same column.

H : = 45 H : 45

Info

t.test(x, mu = 45, alternative = “two.sided”) will estimate the SD, use df = n − 1, and give the p-value.

Preview

```
#| echo: true
df <- read.csv("spotify-2024.csv")
summary(df$Track.Score)
```

```
#| exercise: q6_trackscore_t_builtin
#| exercise.lines: 8
#| echo: false

df <- read.csv("spotify-2024.csv")
x <-

out <- t.test( ,
mu =  ,
alternative = "two.sided")

out$p.value
```

Use the Track.Score column.

Solution.

```
#| exercise: q6_trackscore_t_builtin
#| solution: true

df <- read.csv("spotify-2024.csv")
x <- df$Track.Score

out <- t.test(x,
mu = 45,
alternative = "two.sided")

out$p.value
```

```
#| exercise: q6_trackscore_t_builtin
#| check: true
gradethis::grade_this({
df <- tryCatch(read.csv("spotify-2024.csv"), error = function(e) NULL)
if (is.null(df)) fail("Couldn't read 'spotify-2024.csv'.")
if (!"Track.Score" %in% names(df)) fail("The CSV must contain a 'Track.Score' column.")
x <- df$Track.Score

exp_out <- t.test(x, mu = 45, alternative = "two.sided")
exp_p <- exp_out$p.value

res <- .result
if (!is.numeric(res) || length(res) != 1L) {
fail("Print a single numeric p-value from t.test().")
}

if (is.finite(res) && abs(res - exp_p) < 1e-6) {
pass("Correct p-value for the built-in one-sample t-test on Track.Score.")
} else {
fail("Not quite - check mu = 45 and alternative = 'two.sided'.")
}
})
```

Q7 — One-sample test on a proportion (base R `prop.test()`)

We use the video game sales dataset. Let's test whether **30% of the listed games were published by Nintendo**.

Hypotheses:

$(H_0 : p = 0.045)$ $(H_1 : p \neq 0.045)$

We will form a success/failure variable: "Publisher is Nintendo" = success.

Info

For a one-sample test on a proportion with counts (x) out of (n), use `prop.test(x, n, p = p0, alternative = "two.sided")`. This uses a chi-square test with 1 degree of freedom.

Preview

```
#| echo: true
vg <- read.csv("vgsales.csv")

head(vg[, c("Name","Platform","Publisher")], 10)

table(vg$Publisher)[1:10]   # peek at some publishers
mean(vg$Publisher == "Nintendo", na.rm = TRUE)
```

```
#| exercise: q7_vgs_prop_base
#| exercise.lines: 10
#| echo: false

vg <- read.csv("vgsales.csv")

x <- sum( , na.rm = TRUE)

# total trials
n <- sum(!is.na(vg$Publisher))

out <- prop.test(x = x,
                 n = n,
                 p =  ,
                 alternative =  )

# print p-value
```

Count how many rows have Publisher == “Nintendo”, test against 0.045, and keep it two-sided.

Solution.

```
#| exercise: q7_vgs_prop_base
#| solution: true

vg <- read.csv("vgsales.csv")

x <- sum(vg$Publisher == "Nintendo", na.rm = TRUE)
n <- sum(!is.na(vg$Publisher))
```

```
out <- prop.test(x = x,
                 n = n,
                 p = 0.045,
                 alternative = "two.sided")

out$p.value
```

```
#| exercise: q7_vgs_prop_base
#| check: true
gradethis::grade_this({
  vg <- tryCatch(read.csv("vgsales.csv"), error = function(e) NULL)
  if (is.null(vg)) fail("Couldn't read 'vgsales.csv'.")
  if (!"Publisher" %in% names(vg)) fail("The CSV must contain a 'Publisher' column.")

  x <- sum(vg$Publisher == "Nintendo", na.rm = TRUE)
  n <- sum(!is.na(vg$Publisher))

  exp_out <- prop.test(x = x, n = n, p = 0.045, alternative = "two.sided")
  exp_p <- exp_out$p.value

  res <- .result
  if (!is.numeric(res) || length(res) != 1L) {
    fail("Print a single numeric p-value.")
  }

  if (abs(res - exp_p) < 1e-6) {
    pass("Correct p-value for the one-sample proportion test.")
  } else {
    fail("Check x, n, and p = 0.045.")
  }
})
```

Q8 — From chi-square to z (follow-up on the same test)

`prop.test()` reports a chi-square statistic with 1 df. For a 1-df test, (^2 = z^2). Let's extract that chi-square value and take the square root.

We use the **same dataset and the same hypothesis** as Q7.

Info

If a one-sample proportion test gives (χ^2) with 1 df, then ($z = \sqrt{\chi^2}$) (keep the sign if you need direction).

```
#| exercise: q8_vgs_prop_chisq_to_z
#| exercise.lines: 10
#| echo: false

vg <- read.csv("vgsales.csv")

x <- sum(vg$Publisher == "Nintendo", na.rm = TRUE)
n <- sum(!is.na(vg$Publisher))

out <-

chisq_val <-
z_val <-

c(chisq = chisq_val, z_from_chisq = z_val)
```

Re-run the same `prop.test()` as Q7 and take `sqrt()` of the test statistic.

Solution.

```
#| exercise: q8_vgs_prop_chisq_to_z
#| solution: true

vg <- read.csv("vgsales.csv")

x <- sum(vg$Publisher == "Nintendo", na.rm = TRUE)
n <- sum(!is.na(vg$Publisher))

out <- prop.test(x = x,
                 n = n,
                 p = 0.045,
                 alternative = "two.sided")

chisq_val <- out$statistic
z_val <- sqrt(chisq_val)

c(chisq = chisq_val, z_from_chisq = z_val)
```

```
#| exercise: q8_vgs_prop_chisq_to_z
#| check: true
gradethis::grade_this({
  vg <- tryCatch(read.csv("vgsales.csv"), error = function(e) NULL)
  if (is.null(vg)) fail("Couldn't read 'vgsales.csv'.")
  x <- sum(vg$Publisher == "Nintendo", na.rm = TRUE)
  n <- sum(!is.na(vg$Publisher))

  exp_out <- prop.test(x = x, n = n, p = 0.045, alternative = "two.sided")
  exp_chisq <- as.numeric(exp_out$statistic)
  exp_z <- sqrt(exp_chisq)

  res <- .result
  if (!is.numeric(res) || length(res) != 2L) {
    fail("Return c(chisq = ..., z_from_chisq = ...).")
  }

  if (abs(res[1] - exp_chisq) < 1e-6 && abs(res[2] - exp_z) < 1e-6) {
    pass("Correctly extracted chi-square and its square root.")
  } else {
    fail("Check that you ran the same prop.test and used sqrt() on the statistic.")
  }
})
```

Q9 — One-sample proportion using `prop_test()` (rstatix)

Now we repeat the same test — “is the proportion of Nintendo-published games 30%?” — but use the `prop_test()` function from the **rstatix** package, which gives the **z** statistic directly.

We’ll create a logical column and then test it.

Info

`rstatix::prop_test()` can test a single proportion vs a hypothesized value and returns a z statistic and p-value. We’ll test against 0.045.

```
#| exercise: q9_vgs_prop_rstatix
#| exercise.lines: 12
#| echo: false
```

```
library(rstatix)

vg <- read.csv("vgsales.csv")

vg$is_nintendo <- vg$Publisher == "Nintendo"

out <- rstatix::prop_test(
  x = sum( , na.rm = TRUE),
  n = sum( ),
  p =
)
```

Use the logical column `is_nintendo`.

Solution.

```
#| exercise: q9_vgs_prop_rstatix
#| solution: true
library(rstatix)

vg <- read.csv("vgsales.csv")

vg$is_nintendo <- vg$Publisher == "Nintendo"

out <- rstatix::prop_test(
  x = sum(vg$is_nintendo, na.rm = TRUE),
  n = sum(!is.na(vg$is_nintendo)),
  p = 0.045
)

out
```

```
#| exercise: q9_vgs_prop_rstatix
#| check: true
gradethis::grade_this({
  vg <- tryCatch(read.csv("vgsales.csv"), error = function(e) NULL)
  if (is.null(vg)) fail("Couldn't read 'vgsales.csv'.")
  if (!"Publisher" %in% names(vg)) fail("CSV must contain 'Publisher'.")
```

```
suppressMessages(library(rstatix))

# Build the expected result using rstatix::prop_test(x, n, p)
is_nintendo <- vg$Publisher == "Nintendo"
x <- sum(is_nintendo, na.rm = TRUE)
n <- sum(!is.na(is_nintendo))

exp_out <- rstatix::prop_test(x = x, n = n, p = 0.045)

res <- .result

# Accept either:
# 1) the rstatix tibble/data.frame output, or
# 2) a base stats::prop.test() htest object (in case someone used that)
if (is.data.frame(res) && all(c("statistic", "p") %in% names(res))) {
  got_z <- as.numeric(res$statistic[1])
  got_p <- as.numeric(res$p[1])

  exp_z <- as.numeric(exp_out$statistic[1])
  exp_p <- as.numeric(exp_out$p[1])

  if (isTRUE(all.equal(got_z, exp_z, tolerance = 1e-6)) &&
      isTRUE(all.equal(got_p, exp_p, tolerance = 1e-6))) {
    pass("Correct: rstatix::prop_test output matches expected z and p-value.")
  } else {
    fail("Close, but the z statistic / p-value don't match the expected test (p = 0.045)."
  }

} else if (inherits(res, "htest")) {
  got_z <- unname(as.numeric(res$statistic))
  got_p <- unname(as.numeric(res$p.value))

  exp_z <- as.numeric(exp_out$statistic[1])
  exp_p <- as.numeric(exp_out$p[1])

  if (isTRUE(all.equal(got_z, exp_z, tolerance = 1e-6)) &&
      isTRUE(all.equal(got_p, exp_p, tolerance = 1e-6))) {
    pass("Correct test result (matches expected z and p-value).")
  } else {
    fail("You ran a proportion test, but the result doesn't match the expected test (p = 0
  }
  } else {
    fail("Make sure your last line prints the result of rstatix::prop_test(...).")
  }
})
```