

Tutorial 10: Inference Techniques on Regression

```
#| 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 - Fit the Regression Model

For this tutorial, we will be using a Life Expectancy dataset from the World Health Organization (WHO). From your tutorial 9 knowledge, fit a simple linear regression model with Life Expectancy as X and Schooling as Y. Return the estimated slope.

Photo by Kenny Eliason on Unsplash

Preview

Feel free to run this code block to visualize the data.

```

#| echo: true
df <- read.csv("life_expectancy.csv", check.names = FALSE)
names(df) <- trimws(names(df))

le <- df[, c("Life expectancy", "Schooling")]
names(le) <- c("life_exp", "schooling")
le <- na.omit(le)

head(le, 8)

plot(le$schooling, le$life_exp,
     xlab = "Schooling (years)", ylab = "Life expectancy",
     main = "Life expectancy vs Schooling")

```

```

#| exercise: t10_q1_fit_slope
#| exercise.lines: 8
#| echo: false
df <- read.csv("life_expectancy.csv", check.names=FALSE)
names(df) <- trimws(names(df))
le <- df[, c("Life expectancy","Schooling")]
names(le) <- c("life_exp","schooling") #use these column names

```

```
le <- na.omit(le)

model <- ____ ( ____ ~ ____ , data = le)
coef(model)[____]
```

Fill `coef(model)[_____]` with right column name.

Solution.

```
#| exercise: t10_q1_fit_slope
#| solution: true
df <- read.csv("life_expectancy.csv", check.names=FALSE)
names(df) <- trimws(names(df))
le <- df[, c("Life expectancy", "Schooling")]
names(le) <- c("life_exp", "schooling")
le <- na.omit(le)

model <- lm(life_exp ~ schooling, data = le)
coef(model)["schooling"]
```

```
#| exercise: t10_q1_fit_slope
#| check: true
gradethis::grade_this({
df <- read.csv("life_expectancy.csv", check.names=FALSE)
names(df) <- trimws(names(df))
le <- df[, c("Life expectancy", "Schooling")]
names(le) <- c("life_exp", "schooling")
le <- na.omit(le)
mod <- lm(life_exp ~ schooling, data = le)
exp <- coef(mod)["schooling"]

x <- .result
ok <- is.numeric(x) && length(x) == 1L && is.finite(x)
if (!ok) fail("Return the slope as a single numeric value.")
else if (abs(x - exp) < 1e-10) pass("Correct slope.")
else fail("Slope mismatch. Check your lm() formula and coefficient name.")
})
```

Q2 — Interpret the slope:

A policy increases schooling by **2 years** for every country. Using the fitted regression model, compute the predicted change in life expectancy caused by this policy.

Do this in two ways:

- 1) Compute two fitted values at x_0 and $x_0 + 2$, then take the difference.
- 2) Use the model's slope to compute the same change.

Photo by MD Duran on Unsplash

Info

In a simple linear regression line, the “effect” of increasing (x) by a fixed amount (x) can be seen by comparing fitted values at two (x)-values separated by (x).

A good way to sanity-check an interpretation is to compute the same effect using (1) fitted values and (2) the slope coefficient, and confirm they agree.

Preview

Feel free to run this code block to visualize the data.

```
#| echo: true
df <- read.csv("life_expectancy.csv", check.names = FALSE)
names(df) <- trimws(names(df))

le <- df[, c("Life expectancy", "Schooling")]
names(le) <- c("life_exp", "schooling")
le <- na.omit(le)

plot(le$schooling, le$life_exp,
     xlab = "Schooling (years)", ylab = "Life expectancy",
     main = "Life expectancy vs Schooling")

model <- lm(life_exp ~ schooling, data = le)
coef(model)

x0 <- mean(le$schooling)
x0
```

```
#| exercise: t10_q2_two_year_effect_demo
#| exercise.lines: 16
```

```

#| echo: false

df <- read.csv("life_expectancy.csv", check.names = FALSE)
names(df) <- trimws(names(df))

le <- df[, c("Life expectancy", "Schooling")]
names(le) <- c("life_exp", "schooling")
le <- na.omit(le)

model <- lm(life_exp ~ schooling, data = le)

# pick a reference schooling level

x0 <- mean(le$schooling)

# method 1: compare fitted values at x0 and x0 + 2

yhat0 <- ___
yhat2 <- ___
effect_via_preds <- ___

# method 2: use the slope coefficient

b1 <- ___
effect_via_slope <- ___

c(effect_via_preds = effect_via_preds,
  effect_via_slope = effect_via_slope)

```

Use the idea “effect = fitted value at (new x) minus fitted value at (old x)”. You can get fitted values either from the line equation or from `predict()` using `newdata`. For the slope method, interpret the slope as “change in fitted response per 1 unit of x”.

Solution.

```

#| exercise: t10_q2_two_year_effect_demo
#| solution: true

df <- read.csv("life_expectancy.csv", check.names = FALSE)
names(df) <- trimws(names(df))

le <- df[, c("Life expectancy", "Schooling")]
names(le) <- c("life_exp", "schooling")

```

```

le <- na.omit(le)

model <- lm(life_exp ~ schooling, data = le)

x0 <- mean(le$schooling)

yhat0 <- unname(predict(model, newdata = data.frame(schooling = x0)))
yhat2 <- unname(predict(model, newdata = data.frame(schooling = x0 + 2)))
effect_via_preds <- yhat2 - yhat0

b1 <- unname(coef(model)["schooling"])
effect_via_slope <- 2 * b1

c(effect_via_preds = effect_via_preds,
  effect_via_slope = effect_via_slope)

```

```

#| exercise: t10_q2_two_year_effect_demo
#| check: true
gradethis::grade_this({
df <- read.csv("life_expectancy.csv", check.names = FALSE)
names(df) <- trimws(names(df))

le <- df[, c("Life expectancy", "Schooling")]
names(le) <- c("life_exp", "schooling")
le <- na.omit(le)

mod <- lm(life_exp ~ schooling, data = le)

x0 <- mean(le$schooling)

exp_preds <- unname(predict(mod, newdata = data.frame(schooling = x0 + 2))) -
unname(predict(mod, newdata = data.frame(schooling = x0)))

exp_slope <- 2 * unname(coef(mod)["schooling"])

exp <- c(effect_via_preds = exp_preds, effect_via_slope = exp_slope)

res <- .result
ok <- is.numeric(res) && length(res) == 2L && all(is.finite(res)) &&
identical(names(res), names(exp))

if (!ok) fail("Return c(effect_via_preds = ..., effect_via_slope = ...) (two numbers, named)

```

```
else if (max(abs(res - exp)) < 1e-8) pass("Correct. Both methods agree.")
else fail("Mismatch. Re-check how you computed fitted values and/or extracted the slope.")
})
```

Q3 — Compute R^2

Return the model's R^2

Info

R^2 is the proportion of variability in life expectancy explained by schooling.

Preview

Feel free to run this code block to visualize the data.

```
##| echo: true
df <- read.csv("life_expectancy.csv", check.names = FALSE)
names(df) <- trimws(names(df))

le <- df[, c("Life expectancy", "Schooling")]
names(le) <- c("life_exp", "schooling")
le <- na.omit(le)

head(le, 8)

plot(le$schooling, le$life_exp,
     xlab = "Schooling (years)", ylab = "Life expectancy",
     main = "Life expectancy vs Schooling")
```

```
##| exercise: t10_q3_r2
##| exercise.lines: 7
##| echo: false
df <- read.csv("life_expectancy.csv", check.names = FALSE)
names(df) <- trimws(names(df))
le <- df[, c("Life expectancy", "Schooling")]
names(le) <- c("life_exp", "schooling")
le <- na.omit(le)
model <- lm(life_exp ~ schooling, data = le)
```

```
___(model)$___
```

Produce model summary and return the appropriate statistic from it.

Solution.

```
#| exercise: t10_q3_r2
#| solution: true
df <- read.csv("life_expectancy.csv", check.names = FALSE)
names(df) <- trimws(names(df))
le <- df[, c("Life expectancy", "Schooling")]
names(le) <- c("life_exp", "schooling")
le <- na.omit(le)
model <- lm(life_exp ~ schooling, data = le)

summary(model)$r.squared
```

```
#| exercise: t10_q3_r2
#| check: true
gradethis::grade_this({
df <- read.csv("life_expectancy.csv", check.names = FALSE)
names(df) <- trimws(names(df))
le <- df[, c("Life expectancy", "Schooling")]
names(le) <- c("life_exp", "schooling")
le <- na.omit(le)
mod <- lm(life_exp ~ schooling, data = le)
exp <- summary(mod)$r.squared

x <- .result
ok <- is.numeric(x) && length(x) == 1L && is.finite(x)
if (!ok) fail("Return a single numeric R-squared value.")
else if (abs(x - exp) < 1e-12) pass("Correct R-squared.")
else fail("R-squared mismatch.")
})
```

Q4 — Computing P-value

Compute the p-value for b_1 for testing $H_0 : b_1 = 0$.

Info

This tests whether there is evidence of a linear relationship between schooling and life expectancy.

Preview

Feel free to run this code block to visualize the data.

```
#| echo: true
df <- read.csv("life_expectancy.csv", check.names = FALSE)
names(df) <- trimws(names(df))

le <- df[, c("Life expectancy", "Schooling")]
names(le) <- c("life_exp", "schooling")
le <- na.omit(le)

head(le, 8)

plot(le$schooling, le$life_exp,
     xlab = "Schooling (years)", ylab = "Life expectancy",
     main = "Life expectancy vs Schooling")
```

```
#| exercise: t10_q4_slope_pval
#| exercise.lines: 9
#| echo: false
df <- read.csv("life_expectancy.csv", check.names = FALSE)
names(df) <- trimws(names(df))
le <- df[, c("Life expectancy", "Schooling")]
names(le) <- c("life_exp", "schooling")
le <- na.omit(le)
mod <- lm(life_exp ~ schooling, data = le)

summary(mod)$coefficients[___, ___]
```

The second argument in coefficients represents the column 'Pr(>|t|)'.

Solution.

```
#| exercise: t10_q4_slope_pval
#| solution: true
df <- read.csv("life_expectancy.csv", check.names = FALSE)
names(df) <- trimws(names(df))
le <- df[, c("Life expectancy", "Schooling")]
names(le) <- c("life_exp", "schooling")
le <- na.omit(le)
mod <- lm(life_exp ~ schooling, data = le)

summary(mod)$coefficients["schooling", "Pr(>|t|)"]
```

```
#| exercise: t10_q4_slope_pval
#| check: true
gradethis::grade_this({
df <- read.csv("life_expectancy.csv", check.names = FALSE)
names(df) <- trimws(names(df))
le <- df[, c("Life expectancy", "Schooling")]
names(le) <- c("life_exp", "schooling")
le <- na.omit(le)
mod <- lm(life_exp ~ schooling, data = le)
exp <- summary(mod)$coefficients["schooling", "Pr(>|t|)"]

x <- .result
ok <- is.numeric(x) && length(x) == 1L && is.finite(x)
if (!ok) fail("Return a single numeric p-value.")
else if (abs(x - exp) < 1e-12) pass("Correct p-value. A really low or 0 p-value suggests that")
else fail("P-value mismatch.")
})
```

Q5 — 95% CI for the slope

Return a 95% confidence interval for the slope using the R built-in command `confint()`.

Info

The `confint()` function in R is used to compute confidence intervals for one or more parameters in a fitted model.

Preview

Feel free to run this code block to visualize the data.

```
#| echo: true
df <- read.csv("life_expectancy.csv", check.names = FALSE)
names(df) <- trimws(names(df))

le <- df[, c("Life expectancy", "Schooling")]
names(le) <- c("life_exp", "schooling")
le <- na.omit(le)

head(le, 8)

plot(le$schooling, le$life_exp,
     xlab = "Schooling (years)", ylab = "Life expectancy",
     main = "Life expectancy vs Schooling")
```

```
#| exercise: t10_q5_ci_slope
#| exercise.lines: 9
#| echo: false
df <- read.csv("life_expectancy.csv", check.names = FALSE)
names(df) <- trimws(names(df))
le <- df[, c("Life expectancy", "Schooling")]
names(le) <- c("life_exp", "schooling")
le <- na.omit(le)
model <- lm(life_exp ~ schooling, data = le)

as.numeric(confint(___, ___))
```

Enter the right column names and research how `confint()` is used here.

Solution.

```
#| exercise: t10_q5_ci_slope
#| solution: true
df <- read.csv("life_expectancy.csv", check.names = FALSE)
names(df) <- trimws(names(df))
le <- df[, c("Life expectancy", "Schooling")]
names(le) <- c("life_exp", "schooling")
le <- na.omit(le)
```

```
mod <- lm(life_exp ~ schooling, data = le)
```

```
as.numeric(confint(mod, "schooling"))
```

```
##| exercise: t10_q5_ci_slope
##| check: true
gradethis::grade_this({
  df <- read.csv("life_expectancy.csv", check.names = FALSE)
  names(df) <- trimws(names(df))
  le <- df[, c("Life expectancy", "Schooling")]
  names(le) <- c("life_exp", "schooling")
  le <- na.omit(le)
  mod <- lm(life_exp ~ schooling, data = le)
  exp <- as.numeric(confint(mod, "schooling"))

  x <- .result
  ok <- is.numeric(x) && length(x) == 2L && all(is.finite(x))
  if (!ok) fail("Return a numeric vector of length 2: c(lower, upper).")
  else if (max(abs(x - exp)) < 1e-10) pass("Correct CI.")
  else fail("CI mismatch.")
})
```

Q6 — Predict the mean life expectancy

Predict the mean life expectancy at schooling = 12 years.

Photo by Kenny Eliason on Unsplash

i Info

A point prediction is essentially \hat{y} at a certain x value. In this case, $x=12$.

i Preview

Feel free to run this code block to visualize the data.

```

#| echo: true
df <- read.csv("life_expectancy.csv", check.names = FALSE)
names(df) <- trimws(names(df))

le <- df[, c("Life expectancy", "Schooling")]
names(le) <- c("life_exp", "schooling")
le <- na.omit(le)

head(le, 8)

plot(le$schooling, le$life_exp,
     xlab = "Schooling (years)", ylab = "Life expectancy",
     main = "Life expectancy vs Schooling")

```

```

#| exercise: t10_q6_predict_12
#| exercise.lines: 8
#| echo: false
df <- read.csv("life_expectancy.csv", check.names = FALSE)
names(df) <- trimws(names(df))
le <- df[, c("Life expectancy", "Schooling")]
names(le) <- c("life_exp", "schooling")
le <- na.omit(le)
model <- lm(life_exp ~ schooling, data = le)

predict(___, newdata = data.frame(schooling = ___)) #enter schooling value

```

Look up how to use predict().

Solution.

```

#| exercise: t10_q6_predict_12
#| solution: true
df <- read.csv("life_expectancy.csv", check.names = FALSE)
names(df) <- trimws(names(df))
le <- df[, c("Life expectancy", "Schooling")]
names(le) <- c("life_exp", "schooling")
le <- na.omit(le)
model <- lm(life_exp ~ schooling, data = le)

predict(model, newdata = data.frame(schooling = 12))

```

```

#| exercise: t10_q6_predict_12
#| check: true
gradethis::grade_this({
df <- read.csv("life_expectancy.csv", check.names = FALSE)
names(df) <- trimws(names(df))
le <- df[, c("Life expectancy", "Schooling")]
names(le) <- c("life_exp", "schooling")
le <- na.omit(le)
mod <- lm(life_exp ~ schooling, data = le)
exp <- as.numeric(predict(mod, newdata = data.frame(schooling = 12)))

x <- .result
ok <- is.numeric(x) && length(x) == 1L && is.finite(x)
if (!ok) fail("Return a single numeric prediction.")
else if (abs(x - exp) < 1e-10) pass("Correct prediction.")
else fail("Prediction mismatch.")
})

```

Q7 — Residuals sum

Confirm if the residuals in the model sum to 0 or not. Round the sum.

Info

Usually with an intercept in the model, the residuals sum up to 0 (with a tiny numerical error).

Preview

```
#| echo: true
df <- read.csv("life_expectancy.csv", check.names = FALSE)
names(df) <- trimws(names(df))

le <- df[, c("Life expectancy", "Schooling")]
names(le) <- c("life_exp", "schooling")
le <- na.omit(le)

mod <- lm(life_exp ~ schooling, data = le)

# Key idea: residuals balance around 0 when there's an intercept

hist(resid(mod),
main = "Histogram of residuals",
xlab = "Residuals")

boxplot(resid(mod), horizontal = TRUE,
main = "Residuals boxplot")

# A visual "sum to zero" intuition: mean residual should be ~ 0
mean(resid(mod))
```

```
#| exercise: t10_q7_resid_sum
#| exercise.lines: 8
#| echo: false
df <- read.csv("life_expectancy.csv", check.names = FALSE)
names(df) <- trimws(names(df))
le <- df[, c("Life expectancy", "Schooling")]
names(le) <- c("life_exp", "schooling")
le <- na.omit(le)
model <- lm(life_exp ~ schooling, data = le)

round(sum(___(___)), 6)
```

Use `resid()` to get residuals.

Solution.

```

#| exercise: t10_q7_resid_sum
#| solution: true
df <- read.csv("life_expectancy.csv", check.names = FALSE)
names(df) <- trimws(names(df))
le <- df[, c("Life expectancy", "Schooling")]
names(le) <- c("life_exp", "schooling")
le <- na.omit(le)
model <- lm(life_exp ~ schooling, data = le)

round(sum(resid(model)), 6)

```

```

#| exercise: t10_q7_resid_sum
#| check: true
gradethis::grade_this({
df <- read.csv("life_expectancy.csv", check.names = FALSE)
names(df) <- trimws(names(df))
le <- df[, c("Life expectancy", "Schooling")]
names(le) <- c("life_exp", "schooling")
le <- na.omit(le)
mod <- lm(life_exp ~ schooling, data = le)
exp <- round(sum(resid(mod)), 6)

x <- .result
ok <- is.numeric(x) && length(x) == 1L && is.finite(x)
if (!ok) fail("Return one rounded number.")
else if (abs(x - exp) < 1e-12) pass("Correct.")
else fail("Mismatch.")
})

```

Q8 — Standardized residuals

Compute standardized residuals and figure out how many satisfy the condition $|r_i| > 2$.

Info

Standardized residuals are useful for spotting outliers in data. If the condition $|r_i| > 2$ is satisfied, it could be a common red flag.

Preview

```
#| echo: true
df <- read.csv("life_expectancy.csv", check.names = FALSE)
names(df) <- trimws(names(df))

le <- df[, c("Life expectancy", "Schooling")]
names(le) <- c("life_exp", "schooling")
le <- na.omit(le)

mod <- lm(life_exp ~ schooling, data = le)

r <- rstandard(mod)

# Show the distribution and typical range without counting > 2

hist(r,
main = "Standardized residuals",
xlab = "rstandard(mod)")
abline(v = c(-2, 2), lwd = 2)

# Show the top few absolute values (but don't reveal the count)

head(sort(abs(r), decreasing = TRUE), 10)
```

```
#| exercise: t10_q8_std_resid_count
#| exercise.lines: 9
#| echo: false
df <- read.csv("life_expectancy.csv", check.names = FALSE)
names(df) <- trimws(names(df))
le <- df[, c("Life expectancy", "Schooling")]
names(le) <- c("life_exp", "schooling")
le <- na.omit(le)
model <- lm(life_exp ~ schooling, data = le)

r <- ___(model)
sum(abs(r) > ___)
```

Use `rstandard()` to get standardized residuals.

Solution.

```

#| exercise: t10_q8_std_resid_count
#| solution: true
df <- read.csv("life_expectancy.csv", check.names = FALSE)
names(df) <- trimws(names(df))
le <- df[, c("Life expectancy", "Schooling")]
names(le) <- c("life_exp", "schooling")
le <- na.omit(le)
model <- lm(life_exp ~ schooling, data = le)

r <- rstandard(model)
sum(abs(r) > 2)

```

```

#| exercise: t10_q8_std_resid_count
#| check: true
gradethis::grade_this({
df <- read.csv("life_expectancy.csv", check.names = FALSE)
names(df) <- trimws(names(df))
le <- df[, c("Life expectancy", "Schooling")]
names(le) <- c("life_exp", "schooling")
le <- na.omit(le)
mod <- lm(life_exp ~ schooling, data = le)
exp <- sum(abs(rstandard(mod)) > 2)

x <- .result
ok <- is.numeric(x) && length(x) == 1L && is.finite(x)
if (!ok) fail("Return a single integer count.")
else if (abs(x - exp) < 1e-12) pass("Correct.")
else fail("Count mismatch.")
})

```