Tutorial 09: Intro to 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 a simple linear regression model (movies only)

Using `titles.csv`, we will work with **movies only** and fit:

$$y = \beta_0 + \beta_1 x + \varepsilon$$

keeping y = imdb_score and x = runtime.

Your task: - Fit a regression line - Return a named numeric vector: `c(b0 = ..., b1 = ...)`

Photo by Thibault Penin on Unsplash

> **Info**
>
> A simple linear regression estimates a straight-line relationship between a predictor x and a response y.
>
> - b_0 (intercept): the predicted value of y when x = 0
> - b_1 (slope): the predicted change in y for a one-unit increase in x
>
> In R, once you fit the model, you can extract the two estimated coefficients from the fitted object.

Preview

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

table(df$type)

sub <- subset(df,
  type == "MOVIE" &
  is.finite(runtime) &
  is.finite(imdb_score) &
  runtime > 0
)

nrow(sub)
summary(sub$runtime)
summary(sub$imdb_score)

par(mfrow = c(1, 2))
plot(imdb_score ~ runtime, data = sub,
     xlab = "runtime (minutes)", ylab = "imdb_score",
     main = "IMDB score vs runtime (movies)")
hist(sub$imdb_score, xlab = "imdb_score", main = "Histogram: imdb_score")
par(mfrow = c(1, 1))
```

```
#| exercise: q1_titles_fit_lm
#| exercise.lines: 14
#| echo: false

df <- read.csv("titles.csv")

sub <- subset(df,
type == "MOVIE" &
is.finite(runtime) &
is.finite(imdb_score) &
runtime > 0
)

fit <- lm(___ ~ ___, data = sub)

b <- coef(fit)
c(b0 = unname(b[1]), b1 = unname(b[2]))
```

You need a fitted regression model object using the filtered movie data. Then extract the two coefficient estimates from that object (intercept first, slope second).

Solution.

```
#| exercise: q1_titles_fit_lm
#| solution: true

df <- read.csv("titles.csv")

sub <- subset(df,
type == "MOVIE" &
is.finite(runtime) &
is.finite(imdb_score) &
runtime > 0
)

fit <- lm(imdb_score ~ runtime, data = sub)

b <- coef(fit)
c(b0 = unname(b[1]), b1 = unname(b[2]))
```

```
#| exercise: q1_titles_fit_lm
#| check: true

gradethis::grade_this({
df <- tryCatch(read.csv("titles.csv"), error = function(e) NULL)
if (is.null(df)) fail("Couldn't read 'titles.csv'.")

sub <- subset(df,
type == "MOVIE" &
is.finite(runtime) &
is.finite(imdb_score) &
runtime > 0
)
if (nrow(sub) < 10) fail("Too few usable movie rows after filtering.")

fit <- lm(imdb_score ~ runtime, data = sub)
b <- coef(fit)
exp <- c(b0 = unname(b[1]), b1 = unname(b[2]))
```

```
res <- .result
if (!is.numeric(res) || length(res) != 2L || any(!is.finite(res))) {
fail("Return c(b0 = ..., b1 = ...) as a numeric vector of length 2.")
} else if (max(abs(res - exp)) < 1e-6) {
pass("Correct model fit and coefficients for imdb_score ~ runtime (movies).")
} else {
fail("Check the subset/filtering and that you used lm(imdb_score ~ runtime, data = sub).")
}
})
```

Q2 — Compute ($\hat{\beta}_1$) (slope) by computation

Compute the slope estimator using the formula:

$$\hat{\beta}_1 = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sum(x_i - \bar{x})^2}$$

Here, (x=runtime) and (y=imdb_score) (movies only). Return the numeric value `b1_hat`.

Info

Let:

- The numerator measures how x and y move together (a "covariance-like" quantity).
- The denominator measures how spread out x is (a "variance-like" quantity).
- The slope is the "co-movement" scaled by the spread in x.

Preview

```
#| echo: true
df <- read.csv("titles.csv")
sub <- subset(df,
type == "MOVIE" &
is.finite(runtime) &
is.finite(imdb_score) &
runtime > 0
)

x <- sub$runtime
y <- sub$imdb_score

c(n = length(x), xbar = mean(x), ybar = mean(y))
```

```
#| exercise: q2_titles_b1_manual
#| exercise.lines: 18
#| echo: false

df <- read.csv("titles.csv")

sub <- subset(df,
type == "MOVIE" &
is.finite(runtime) &
is.finite(imdb_score) &
runtime > 0
)

x <- sub$runtime
y <- sub$imdb_score

xbar <- ___(x)
ybar <- ___(y)

num <- sum((x - xbar) * (y - ybar))
den <- sum((x - xbar)^2)

b1_hat <- ___ / ___
b1_hat
```

Start by centering both variables around their sample means. Then compute the two sums

shown in the formula (top and bottom) and combine them to get the slope.

Solution.

```
#| exercise: q2_titles_b1_manual
#| solution: true

df <- read.csv("titles.csv")

sub <- subset(df,
type == "MOVIE" &
is.finite(runtime) &
is.finite(imdb_score) &
runtime > 0
)

x <- sub$runtime
y <- sub$imdb_score

xbar <- mean(x)
ybar <- mean(y)

num <- sum((x - xbar) * (y - ybar))
den <- sum((x - xbar)^2)

b1_hat <- num / den
b1_hat
```

```
#| exercise: q2_titles_b1_manual
#| check: true

gradethis::grade_this({
df <- tryCatch(read.csv("titles.csv"), error = function(e) NULL)
if (is.null(df)) fail("Couldn't read 'titles.csv'.")

sub <- subset(df,
type == "MOVIE" &
is.finite(runtime) &
is.finite(imdb_score) &
runtime > 0
)
x <- sub$runtime
y <- sub$imdb_score
```

```
xbar <- mean(x); ybar <- mean(y)
exp <- sum((x - xbar) * (y - ybar)) / sum((x - xbar)^2)

res <- .result
if (!is.numeric(res) || length(res) != 1L || !is.finite(res)) {
fail("Return a single numeric value b1_hat.")
} else if (abs(res - exp) < 1e-6) {
pass("Correct slope estimator (b1) by computation.")
} else {
fail("Check numerator/denominator in the b1_hat formula.")
}
})
```

Q3 — Compute ($\hat{\beta}_0$) (intercept) by computation

Compute the intercept estimator:

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$$

Return the numeric value $\hat{\beta}_0$.

> **Info**
>
> Conceptually:
>
> - The fitted regression line must pass through the point $(\bar{x}, \bar{y})$.
> - Once you have the slope, the intercept is whatever value makes the line go through that mean point.

Preview

```
#| echo: true
df <- read.csv("titles.csv")
sub <- subset(df,
type == "MOVIE" &
is.finite(runtime) &
is.finite(imdb_score) &
runtime > 0
)
c(xbar = mean(sub$runtime), ybar = mean(sub$imdb_score))
```

```
#| exercise: q3_titles_b0_manual
#| exercise.lines: 18
#| echo: false

df <- read.csv("titles.csv")

sub <- subset(df,
type == "MOVIE" &
is.finite(runtime) &
is.finite(imdb_score) &
runtime > 0
)

x <- sub$runtime
y <- sub$imdb_score

xbar <- mean(x)
ybar <- mean(y)

b1_hat <- sum((x - xbar) * (y - ybar)) / sum((x - xbar)^2)

b0_hat <- ___ - ___ * ___
b0_hat
```

Use the fact that the fitted line goes through $(\bar{x}, \bar{y})$. If you already computed the slope in Q2, you only need sample means to determine the intercept.

Solution.

```
#| exercise: q3_titles_b0_manual
#| solution: true

df <- read.csv("titles.csv")

sub <- subset(df,
type == "MOVIE" &
is.finite(runtime) &
is.finite(imdb_score) &
runtime > 0
)

x <- sub$runtime
y <- sub$imdb_score

xbar <- mean(x)
ybar <- mean(y)

b1_hat <- sum((x - xbar) * (y - ybar)) / sum((x - xbar)^2)

b0_hat <- ybar - b1_hat * xbar
b0_hat
```

```
#| exercise: q3_titles_b0_manual
#| check: true

gradethis::grade_this({
df <- tryCatch(read.csv("titles.csv"), error = function(e) NULL)
if (is.null(df)) fail("Couldn't read 'titles.csv'.")

sub <- subset(df,
type == "MOVIE" &
is.finite(runtime) &
is.finite(imdb_score) &
runtime > 0
)
x <- sub$runtime
y <- sub$imdb_score

xbar <- mean(x); ybar <- mean(y)
b1 <- sum((x - xbar) * (y - ybar)) / sum((x - xbar)^2)
exp <- ybar - b1 * xbar
```

```
res <- .result
if (!is.numeric(res) || length(res) != 1L || !is.finite(res)) {
fail("Return a single numeric value b0_hat.")
} else if (abs(res - exp) < 1e-6) {
pass("Correct intercept estimator (b0) by computation.")
} else {
fail("Check: b0_hat = ybar - b1_hat * xbar.")
}
})
```

Q4 — Cross-check ($\hat{\beta}_0$, $\hat{\beta}_1$) using summary(lm)

Fit the same model as Q1 and extract coefficients from:

`summary(fit)$coefficients`

Return `c(b0 = ..., b1 = ...)`.

Info

Most regression summaries include a coefficient table where:

- each row corresponds to a coefficient (intercept + predictor)
- the main estimate column gives the fitted coefficient values

Your goal is to extract the two estimates for this model.

Preview

```
#| echo: true
df <- read.csv("titles.csv")
sub <- subset(df,
type == "MOVIE" &
is.finite(runtime) &
is.finite(imdb_score) &
runtime > 0
)

fit <- lm(imdb_score ~ runtime, data = sub)
summary(fit)$coefficients
```

```
#| exercise: q4_titles_summary_extract
#| exercise.lines: 16
#| echo: false

df <- read.csv("titles.csv")

sub <- subset(df,
type == "MOVIE" &
is.finite(runtime) &
is.finite(imdb_score) &
runtime > 0
)

fit <- lm(imdb_score ~ runtime, data = sub)

tab <- summary(fit)$coefficients

b0 <- tab["___", "___"]
b1 <- tab["___", "___"]

c(b0 = b0, b1 = b1)
```

Look for the coefficient table in the model summary. It will have a row for the intercept and a row for the predictor, and an "estimate"-type column containing the fitted values.

Solution.

```
#| exercise: q4_titles_summary_extract
#| solution: true

df <- read.csv("titles.csv")

sub <- subset(df,
type == "MOVIE" &
is.finite(runtime) &
is.finite(imdb_score) &
runtime > 0
)

fit <- lm(imdb_score ~ runtime, data = sub)

tab <- summary(fit)$coefficients
```

```
b0 <- tab["(Intercept)", "Estimate"]
b1 <- tab["runtime", "Estimate"]

c(b0 = b0, b1 = b1)
```

```
#| exercise: q4_titles_summary_extract
#| check: true

gradethis::grade_this({
df <- tryCatch(read.csv("titles.csv"), error = function(e) NULL)
if (is.null(df)) fail("Couldn't read 'titles.csv'.")

sub <- subset(df,
type == "MOVIE" &
is.finite(runtime) &
is.finite(imdb_score) &
runtime > 0
)
fit <- lm(imdb_score ~ runtime, data = sub)
tab <- summary(fit)$coefficients

exp <- c(
b0 = tab["(Intercept)", "Estimate"],
b1 = tab["runtime", "Estimate"]
)

res <- .result
if (!is.numeric(res) || length(res) != 2L || any(!is.finite(res))) {
fail("Return c(b0 = ..., b1 = ...) as a numeric vector of length 2.")
} else if (max(abs(res - exp)) < 1e-6) {
pass("Correct extraction from summary(fit)$coefficients.")
} else {
fail("Check the row/column names: '(Intercept)' and 'runtime', column 'Estimate'.")
}
})
```

Q5 — Point prediction by hand (using b0, b1)

Using the same filtered movie subset and the fitted line, compute the predicted IMDB score when:

- $x_0 = 90$ minutes

Return a single numeric value: `yhat_90`.

> **Info**
>
> A point prediction on a fitted line is the model's estimated mean response at a chosen x_0. Conceptually: "plug in x_0 into the fitted line."

> **Preview**

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

sub <- subset(df,
  type == "MOVIE" &
  is.finite(runtime) &
  is.finite(imdb_score) &
  runtime > 0
)

fit <- lm(imdb_score ~ runtime, data = sub)
coef(fit)

plot(imdb_score ~ runtime, data = sub,
     xlab = "runtime (minutes)", ylab = "imdb_score",
     main = "IMDB score vs runtime (movies)")
abline(fit)
```

```
#| exercise: q5_titles_pred_hand
#| exercise.lines: 14
#| echo: false

df <- read.csv("titles.csv")

sub <- subset(df,
type == "MOVIE" &
is.finite(runtime) &
is.finite(imdb_score) &
runtime > 0
)
```

```
fit <- lm(imdb_score ~ runtime, data = sub)
b <- coef(fit)

x0 <- 90
yhat_90 <- ___
yhat_90
```

A point prediction uses the fitted line's two coefficients and a chosen (x_0). Use the intercept + slope idea (one part is the "baseline", the other is the "change per minute" times (x_0)).

Solution.

```
#| exercise: q5_titles_pred_hand
#| solution: true

df <- read.csv("titles.csv")

sub <- subset(df,
type == "MOVIE" &
is.finite(runtime) &
is.finite(imdb_score) &
runtime > 0
)

fit <- lm(imdb_score ~ runtime, data = sub)
b <- coef(fit)

x0 <- 90
yhat_90 <- b[1] + b[2] * x0
yhat_90
```

```
#| exercise: q5_titles_pred_hand
#| check: true

gradethis::grade_this({
df <- tryCatch(read.csv("titles.csv"), error = function(e) NULL)
if (is.null(df)) fail("Couldn't read 'titles.csv'.")

sub <- subset(df,
type == "MOVIE" &
is.finite(runtime) &
is.finite(imdb_score) &
```

```
runtime > 0
)
fit <- lm(imdb_score ~ runtime, data = sub)
b <- coef(fit)

x0 <- 90
exp <- unname(b[1] + b[2] * x0)

res <- .result
if (!is.numeric(res) || length(res) != 1L || !is.finite(res)) {
fail("Return a single numeric value yhat_90.")
} else if (abs(res - exp) < 1e-6) {
pass("Correct point prediction at runtime = 90.")
} else {
fail("Prediction is off. Re-check how you used the two fitted coefficients with x0.")
}
})
```

Q6 — Point prediction using predict()

Use the fitted model to predict the IMDB score at:

- $(x_0 = 90)$ minutes

This time, use `predict()` and a `newdata` data frame.

Return a single numeric value: `yhat_90`.

> **Info**
>
> Built-in prediction methods require:
>
> - a fitted model object
> - a `newdata` data frame with the predictor column named exactly as in the model

Preview

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

sub <- subset(df,
type == "MOVIE" &
is.finite(runtime) &
is.finite(imdb_score) &
runtime > 0
)

fit <- lm(imdb_score ~ runtime, data = sub)

new <- data.frame(runtime = c(90, 120))
new
```

```
#| exercise: q6_titles_pred_predict
#| exercise.lines: 12
#| echo: false

df <- read.csv("titles.csv")

sub <- subset(df,
type == "MOVIE" &
is.finite(runtime) &
is.finite(imdb_score) &
runtime > 0
)

fit <- lm(imdb_score ~ runtime, data = sub)

new <- data.frame(runtime = 90)
yhat_90 <- ___
yhat_90
```

Create a one-row `newdata` data frame whose column name matches the predictor used in the model, then call the model's prediction function and extract the numeric value.

Solution.

```
#| exercise: q6_titles_pred_predict
#| solution: true

df <- read.csv("titles.csv")

sub <- subset(df,
type == "MOVIE" &
is.finite(runtime) &
is.finite(imdb_score) &
runtime > 0
)

fit <- lm(imdb_score ~ runtime, data = sub)

new <- data.frame(runtime = 90)
yhat_90 <- predict(fit, newdata = new)
yhat_90
```

```
#| exercise: q6_titles_pred_predict
#| check: true

gradethis::grade_this({
df <- tryCatch(read.csv("titles.csv"), error = function(e) NULL)
if (is.null(df)) fail("Couldn't read 'titles.csv'.")

sub <- subset(df,
type == "MOVIE" &
is.finite(runtime) &
is.finite(imdb_score) &
runtime > 0
)
fit <- lm(imdb_score ~ runtime, data = sub)

exp <- unname(predict(fit, newdata = data.frame(runtime = 90)))

res <- .result
if (!is.numeric(res) || length(res) != 1L || !is.finite(res)) {
fail("Return a single numeric prediction yhat_90.")
} else if (abs(res - exp) < 1e-6) {
pass("Correct predict() result at runtime = 90.")
} else {
fail("Prediction mismatch. Check newdata column name and the predict() call.")
```

```
}
})
```

Q7 — Compare two predictions (difference in fitted values)

Using the fitted model, compute:

$$\hat{y}(120) - \hat{y}(80)$$

Return a single numeric value: `diff_hat`.

> **Info**
>
> This question is about comparing fitted values at two predictor values. Conceptually: "how much does the model expect the response to change when (x) changes from 80 to 120?"

> **Preview**
>
> ```
> #| echo: true
> df <- read.csv("titles.csv")
>
> sub <- subset(df,
> type == "MOVIE" &
> is.finite(runtime) &
> is.finite(imdb_score) &
> runtime > 0
> )
>
> fit <- lm(imdb_score ~ runtime, data = sub)
>
> predict(fit, newdata = data.frame(runtime = c(80, 120)))
> ```

```
#| exercise: q7_titles_pred_diff
#| exercise.lines: 12
#| echo: false

df <- read.csv("titles.csv")

sub <- subset(df,
```

```
type == "MOVIE" &
is.finite(runtime) &
is.finite(imdb_score) &
runtime > 0
)

fit <- lm(imdb_score ~ runtime, data = sub)

yhat <- ___
diff_hat <- ___
diff_hat
```

Get two fitted values for (x=80) and (x=120) (any reasonable method), then subtract in the requested order.

Solution.

```
#| exercise: q7_titles_pred_diff
#| solution: true

df <- read.csv("titles.csv")

sub <- subset(df,
type == "MOVIE" &
is.finite(runtime) &
is.finite(imdb_score) &
runtime > 0
)

fit <- lm(imdb_score ~ runtime, data = sub)

yhat <- predict(fit, newdata = data.frame(runtime = c(80, 120)))
diff_hat <- yhat[2] - yhat[1]
diff_hat
```

```
#| exercise: q7_titles_pred_diff
#| check: true

gradethis::grade_this({
df <- tryCatch(read.csv("titles.csv"), error = function(e) NULL)
if (is.null(df)) fail("Couldn't read 'titles.csv'.")
```

```
sub <- subset(df,
type == "MOVIE" &
is.finite(runtime) &
is.finite(imdb_score) &
runtime > 0
)
fit <- lm(imdb_score ~ runtime, data = sub)

yhat <- predict(fit, newdata = data.frame(runtime = c(80, 120)))
exp <- unname(yhat[2] - yhat[1])

res <- .result
if (!is.numeric(res) || length(res) != 1L || !is.finite(res)) {
fail("Return a single numeric value diff_hat.")
} else if (abs(res - exp) < 1e-6) {
pass("Correct difference in predictions: yhat(120) - yhat(80).")
} else {
fail("Difference is off. Check that you predicted at both x values and subtracted in the rig
}
})
```