SYNTHETIC DATA AND PARAMETER INFLUENCE IN LINEAR MODELS

Synthetic data and parameter influence in linear models

In this lab you will generate synthetic datasets and study how model parameters affect regression results, especially:

- estimated coefficients (`beta` values),
- p-values,
- confidence intervals,
- model fit indicators such as `R-squared`.

The main point is to understand *why* statistical outputs change when data-generating assumptions change.

1. Setup and first synthetic dataset

We start from a simple model:

$$y = \beta_0 + \beta_1 x + \varepsilon, \quad \varepsilon \sim N(0, \sigma^2).$$

```
set.seed(42)

n <- 120
beta0 <- 2
beta1 <- 1.5
sigma <- 2

x <- runif(n, min = 0, max = 10)
eps <- rnorm(n, mean = 0, sd = sigma)
y <- beta0 + beta1 * x + eps

df <- data.frame(x = x, y = y)
head(df)
```

Fit the model:

```
m <- lm(y ~ x, data = df)
summary(m)
confint(m)
```

Quick visualization:

```
plot(df$x, df$y, pch = 19, col = "steelblue",
     xlab = "x", ylab = "y", main = "Synthetic data")
abline(m, col = "red", lwd = 2)
```

2. A helper function to run experiments

To compare scenarios efficiently, create a small simulation function:

```
run_experiment <- function(n = 100, beta0 = 0, beta1 = 1, sigma = 1, x_max = 1
  x <- runif(n, 0, x_max)
  y <- beta0 + beta1 * x + rnorm(n, 0, sigma)
  fit <- lm(y ~ x)
  s <- summary(fit)

  c(
    n = n,
    beta1_true = beta1,
    beta1_hat = coef(fit)[["x"]],
    beta1_se = s$coefficients["x", "Std. Error"],
    p_value = s$coefficients["x", "Pr(>|t|)"],
    r_squared = s$r.squared
  )
}
```

Single run example:

```
run_experiment(n = 100, beta0 = 2, beta1 = 1.5, sigma = 2, x_max = 10)
```

3. Influence of noise level (sigma)

Now keep everything fixed except sigma:

```
set.seed(123)
sigma_grid <- c(0.5, 1, 2, 4, 8)

res_sigma <- do.call(
  rbind,
  lapply(sigma_grid, function(sig) {
    run_experiment(n = 120, beta0 = 2, beta1 = 1.5, sigma = sig, x_max =
  })
)

as.data.frame(res_sigma)
```

Interpretation guideline:

- higher noise usually increases standard errors,
- p-values often become larger,
- R-squared tends to decrease.

Task 3.1

1. Repeat each sigma case at least 100 times.
2. For each sigma, compute:
   - median p-value,
   - mean estimated slope (beta1_hat),
   - mean R-squared.
3. Comment on trends.

4. Influence of sample size (n)

Keep beta1 and sigma fixed, vary n:

```
set.seed(123)
n_grid <- c(20, 40, 80, 160, 320)

res_n <- do.call(
  rbind,
  lapply(n_grid, function(nn) {
    run_experiment(n = nn, beta0 = 2, beta1 = 1.5, sigma = 2, x_max = 10)
  })
)

as.data.frame(res_n)
```

Expected pattern:

- larger n usually gives smaller standard errors,
- p-values become smaller (more power),
- estimates are more stable across repetitions.

Task 4.1

For each value in n_grid, run 200 simulations and compute:

1. mean and standard deviation of beta1_hat,
2. fraction of simulations with p_value < 0.05,
3. average width of the 95% confidence interval for the slope.

5. Influence of true effect size (beta1)

Now vary the true slope:

```
set.seed(123)
beta1_grid <- c(0, 0.1, 0.3, 0.7, 1.5)
```

```
res_beta <- do.call(
  rbind,
  lapply(beta1_grid, function(b1) {
    run_experiment(n = 120, beta0 = 2, beta1 = b1, sigma = 2, x_max = 10
  })
)

as.data.frame(res_beta)
```

Interpretation:

- when `beta1 = 0`, the p-value should often be non-significant,
- as `beta1` increases, p-values should generally decrease and power should rise.

Task 5.1

1. For each `beta1` in `beta1_grid`, run at least 300 simulations.
2. Compute empirical power as:
   - proportion with `p_value < 0.05`.
3. Plot power vs true `beta1`.

6. Influence of predictor range (`x_max`)

A wider range of `x` can improve slope estimation.

```
set.seed(123)
xmax_grid <- c(2, 5, 10, 20)

res_xmax <- do.call(
  rbind,
  lapply(xmax_grid, function(xm) {
    run_experiment(n = 120, beta0 = 2, beta1 = 1.5, sigma = 2, x_max = x
  })
)

as.data.frame(res_xmax)
```


Task 6.1

Using repeated simulations (at least 200 per case), test whether increasing `x_max`:

1. decreases the standard error of `beta1_hat`,
2. increases the fraction of significant p-values.

7. Compact simulation loop (recommended workflow)

The following pattern makes comparisons easier:

```
set.seed(2026)
n_rep <- 300
sigma_grid <- c(0.5, 1, 2, 4)

all_res <- do.call(rbind, lapply(sigma_grid, function(sig) {
  reps <- replicate(
    n_rep,
    run_experiment(n = 120, beta0 = 2, beta1 = 1.5, sigma = sig, x_max = 10),
    simplify = "matrix"
  )
  out <- as.data.frame(t(reps))
  out$sigma <- sig
  out
}))

head(all_res)
```

Example summary by `sigma`:

```
aggregate(cbind(beta1_hat, p_value, r_squared) ~ sigma, data = all_res, mean)
aggregate(beta1_hat ~ sigma, data = all_res, sd)
```

8. Visual summaries

Useful plots:

```r
# p-values by sigma
boxplot(p_value ~ sigma, data = all_res,
        main = "p-values vs noise level",
        xlab = "sigma", ylab = "p-value")
abline(h = 0.05, lty = 2, col = "red")

# estimated slope by sigma
boxplot(beta1_hat ~ sigma, data = all_res,
        main = "Estimated slope vs noise level",
        xlab = "sigma", ylab = "beta1_hat")
abline(h = 1.5, lty = 2, col = "blue")
```

9. Final tasks

Task 9.1
Design your own experiment by varying *two* parameters jointly (for example `n` and `sigma`). Prepare a table with:

- average `beta1_hat`,
- standard deviation of `beta1_hat`,
- proportion of `p_value < 0.05`,
- average `R-squared`.

Task 9.2
In 6-10 sentences, summarize your conclusions:

1. What most strongly affects p-values?
2. What most strongly affects stability of `beta` estimates?
3. Which settings can produce misleading conclusions?

10. Wrap-up

By generating synthetic data, you can directly control the truth (`beta0`, `beta1`, `sigma`, sample size, and predictor range) and observe how inferential outputs react. This is one of the best ways to build intuition about p-values, coefficient uncertainty, and model reliability.