SIMPLE LINEAR MODELS WITH LM IN R

Simple linear models with `lm` in R

This instruction introduces simple linear regression in R using the `lm()` function. You will:

- start with manufactured (simulated) data, focusing on continuous variables,
- then fit the same model on a standard built-in dataset,
- and finally include an example where a **factor** variable is added to the model.

1. What `lm()` fits (conceptually)

The simple linear regression model has the form:

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

where:

- `x` and `y` are continuous variables,
- `beta0` and `beta1` are the intercept and slope,
- `eps` is random noise.

In R, you specify the model using a formula:

```
fit <- lm(y ~ x, data = df)
```

Interpretation:

- `y ~ x` means "model `y` as a linear function of `x`".
- `lm()` estimates `beta0` and `beta1` using least squares.

2. Manufactured data (continuous variables only)

2.1 Simulate a dataset

Create a dataset where the true relationship is linear:

```
set.seed(1)
n <- 60

x <- runif(n, min = 0, max = 10)           # continuous predictor
beta0_true <- 2
beta1_true <- 3
sigma_true <- 1.0

eps <- rnorm(n, mean = 0, sd = sigma_true) # noise
y <- beta0_true + beta1_true * x + eps

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

Optional sanity check:

```
plot(df$x, df$y, pch = 19, col = "steelblue",
     xlab = "x", ylab = "y", main = "Manufactured data")
```

2.2 Fit the simple linear model with `lm()`

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

Key quantities you should look for in `summary(fit)`:

- the estimated intercept (`(Intercept)`) and slope (`x`),
- their standard errors and p-values,
- the `R-squared` (how much of the variance is explained by the linear model),
- the residual standard error (related to the noise level).

You can also view coefficients directly:

```
coef(fit)
```

2.3 Visualize the fitted line and residuals

Regression line:

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

Residuals vs fitted values (useful to check for obvious patterns):

```
plot(fitted(fit), resid(fit),
     pch = 19, col = "gray40",
     xlab = "Fitted values", ylab = "Residuals",
     main = "Residuals vs fitted")
abline(h = 0, lty = 2)
```

2.4 Predict at new x-values

```
new_df <- data.frame(x = seq(0, 10, length.out = 100))
pred <- predict(fit, newdata = new_df)
```

3. Tasks for manufactured data

Task 2.1: Compare the estimated coefficients with the true values.

1. Print `coef(fit)` and check how close the estimates are to:
   - `beta0_true = 2`
   - `beta1_true = 3`
2. Which estimate (intercept or slope) do you think is closer to the truth, and why might that happen?

Task 2.2: Change the noise level and observe what happens.

1. Re-run the simulation with a different `sigma_true` (for example `0.3`, `1.5`, `3.0`).
2. For each case, record:
   - the residual standard error from `summary(fit)`,
   - and whether the slope estimate becomes more or less stable.

Task 2.3: Visual comparison.

1. For your best fit, create a plot with the observed points and the fitted line (`abline(fit)`).
2. Comment briefly on whether the scatter looks "random around the line" or shows a systematic pattern.

4. Standard dataset (continuous variables): Iris

4.1 Load the dataset

The Iris dataset is built-in in R:

```
data(iris)
str(iris)
```

We will model `Sepal.Length` as a linear function of `Petal.Length`.

4.2 Fit a continuous-only linear model

```
m1 <- lm(Sepal.Length ~ Petal.Length, data = iris)
summary(m1)
```

Plot observed points and fitted line:

```
plot(iris$Petal.Length, iris$Sepal.Length,
     pch = 19, col = "steelblue",
     xlab = "Petal.Length", ylab = "Sepal.Length")
abline(m1, col = "red", lwd = 2)
```

4.3 Interpret the slope

The slope coefficient (`Petal.Length`) answers:

"By how much does the expected `Sepal.Length` change when `Petal.Length` increases by `1` (one unit)?"

5. Tasks for the continuous-only model

Task 4.1: Report and interpret.

1. Report the estimated slope and intercept from `summary(m1)` (or `coef(m1)`).
2. Identify the `R-squared` value.
3. In one or two sentences: does the fitted line look like a good description of the scatter?

Task 4.2: Predict at one value.

1. Choose a value of `Petal.Length`, for example `x0 <- 4.0`.
2. Compute `predict(m1, newdata = data.frame(Petal.Length = x0))`.
3. Pick one Iris observation with `Petal.Length` close to `x0` and compare the observed `Sepal.Length` to the prediction.

6. Adding a factor variable: `Species`

Now we extend the model by adding `Species` (a factor with 3 levels). This is an example where the response (`Sepal.Length`) is continuous, but one predictor is categorical.

6.1 Fit the model with `Species`

```r
m2 <- lm(Sepal.Length ~ Petal.Length + Species, data = iris)
summary(m2)
```

Notes:

- `Species` is automatically treated as a factor by R because it is stored as such in `iris`.
- One species level is used as the reference (the "baseline"), and the other levels get coefficient(s) describing differences from that baseline.

To see the reference levels explicitly:

```r
levels(iris$Species)
```

6.2 Compare models (`Species` added or not)

```r
anova(m1, m2)
```

This compares the continuous-only model (`m1`) with the extended model (`m2`).

6.3 Visualize regression lines by species (base R)

```r
sp_levels <- levels(iris$Species)
cols <- 1:length(sp_levels)

plot(iris$Petal.Length, iris$Sepal.Length,
     pch = 19, col = as.numeric(iris$Species),
     xlab = "Petal.Length", ylab = "Sepal.Length")
legend("topleft",
       legend = sp_levels,
       col = cols, pch = 19, bty = "n")

xx <- seq(min(iris$Petal.Length), max(iris$Petal.Length), length.out = 100)
for (k in seq_along(sp_levels)) {
  sp <- sp_levels[k]
  new_df <- data.frame(Petal.Length = xx, Species = sp)
  yy <- predict(m2, newdata = new_df)
  lines(xx, yy, col = k, lwd = 2)
}
```

7. Tasks for the factor-variable example

Task 6.1: Identify species effects.

1. Look at `summary(m2)` and find which `Species` coefficients have small p-values.
2. In plain words: which species differ from the reference species (after accounting for `Petal.Length`)?

Task 6.2: Compare predictions across species.

1. Choose a Petal length value, for example `x0 <- 4.5`.
2. Compute predictions for each species using `predict(m2, ...)`.
3. Which species has the largest predicted `Sepal.Length` at `Petal.Length = 4.5`?

8. Wrap-up

With `lm()` you can:

- model continuous predictors with formulas like `y ~ x`,
- inspect results using `summary()`,
- check fit visually using scatter plots and residual plots,
- include categorical predictors (factors) directly, e.g. `y ~ x + Species`.

If you want to model *different slopes* for different factor levels, you can use interactions such as `y ~ x * Species` (optional extension).