# EXPLORING NYCFLIGHTS13 AND SIMPLE LINEAR MODELS

## Exploring nycflights13 and simple linear models

In this instruction you will work with the `flights` table from the `nycflights13` package. The goals are:

1. compute simple summary statistics on the full table,
2. create a smaller subset that is easier to plot,
3. visualize the smaller dataset in 2D and 3D,
4. compare two linear models:
   - `air_time ~ distance`
   - `log(air_time) ~ log(distance)`

### 1. Load and inspect the dataset

Install and load the package (if needed):

```
install.packages("nycflights13")
```

```
library(nycflights13)
```

The package contains several related tables. We use `flights`:

```
data(flights)
dim(flights)
head(flights)
str(flights)
summary(flights)
```

Important columns for this lab:

- `air_time` - time spent in the air (minutes),
- `distance` - distance (miles),
- `dep_delay`, `arr_delay` - delays (minutes),
- `month`, `day`, `carrier`, `origin`, `dest` - useful grouping variables.

### 2. Simple summary statistics on `flights`

Start with quick summaries for selected numeric columns:

```
summary(flights[, c("air_time", "distance", "dep_delay", "arr_delay")])
```

Compute mean, median, and standard deviation (ignoring missing values):

```
vars <- c("air_time", "distance", "dep_delay", "arr_delay")

means <- sapply(flights[, vars], mean, na.rm = TRUE)
medians <- sapply(flights[, vars], median, na.rm = TRUE)
sds <- sapply(flights[, vars], sd, na.rm = TRUE)

means
medians
sds
```

Some simple grouped summaries:

```
# Average departure delay by month
tapply(flights$dep_delay, flights$month, mean, na.rm = TRUE)

# Number of flights by origin airport
table(flights$origin)

# Average air_time by origin
tapply(flights$air_time, flights$origin, mean, na.rm = TRUE)
```

**Tasks (summary statistics)**

**Task 2.1**

1. Report the mean and median of `distance` and `air_time`.
2. Compare mean vs median for `dep_delay` and comment briefly on skewness.

**Task 2.2**

1. Find the month with the largest mean departure delay.
2. Find which `origin` airport has the longest average `air_time`.

## 3. Create a smaller dataset for visualization

The full `flights` table is very large, so we create a smaller clean sample.

### 3.1 Keep complete rows for key variables

```
fl_clean <- flights[complete.cases(flights[, c("air_time", "distance", "
dim(fl_clean)
```

### 3.2 Subsample to a manageable size

Set a seed for reproducibility and sample rows:

```
set.seed(123)
n_small <- 3000

idx <- sample(seq_len(nrow(fl_clean)), size = n_small)
fl_small <- fl_clean[idx, ]

dim(fl_small)
head(fl_small)
```

Note: for large tables, you can use `sample.int()` for a small performance improvement:

```
idx <- sample.int(nrow(fl_clean), size = n_small)
```

Optional: focus only on selected months or one origin airport:

```
fl_small_jja <- fl_small[fl_small$month %in% c(6, 7, 8), ]    # summer flights
fl_small_ewr <- fl_small[fl_small$origin == "EWR", ]          # one airport
```

If you use the optional subsets, adjust later plots/models accordingly.

**Tasks (subsampling)**

**Task 3.1**

1. Create your own `fl_small` with size between 1500 and 5000.
2. Verify there are no missing values in `air_time` and `distance`:

```
colSums(is.na(fl_small[, c("air_time", "distance")]))
```

## 4. Visualize the smaller dataset in 2D

### 4.1 Scatter plot: `distance` vs `air_time`

```
plot(fl_small$distance, fl_small$air_time,
     pch = 19, cex = 0.5, col = "steelblue",
     xlab = "Distance (miles)",
     ylab = "Air time (minutes)",
     main = "Flights: air_time vs distance")
```

### 4.2 Color by `origin` (factor variable)

```
cols <- as.numeric(factor(fl_small$origin))
plot(fl_small$distance, fl_small$air_time,
     pch = 19, cex = 0.5, col = cols,
     xlab = "Distance (miles)", ylab = "Air time (minutes)",
     main = "air_time vs distance by origin")
legend("topleft", legend = levels(factor(fl_small$origin)),
       col = seq_along(levels(factor(fl_small$origin))),
       pch = 19, bty = "n")
```

### 4.3 Optional transformed-scale 2D plot

```
plot(fl_small$distance, fl_small$air_time,
     pch = 19, cex = 0.5, col = "darkgreen",
     log = "xy",
     xlab = "Distance (log scale)", ylab = "Air time (log scale)",
     main = "Log-log view")
```

### Tasks (2D visualization)

### Task 4.1

1. Produce at least two 2D plots:
   - one on original scale,
   - one on log-log scale.
2. Comment whether the relationship looks more linear after log transform.

## 5. Visualize the smaller dataset in 3D

We will use three continuous variables:

- `x = distance`,
- `y = air_time`,
- `z = dep_delay`.

### 5.1 3D scatter with `scatterplot3d`

Install and load package:

```
install.packages("scatterplot3d")
library(scatterplot3d)
```

Plot:

```
scatterplot3d(
  x = fl_small$distance,
  y = fl_small$air_time,
  z = fl_small$dep_delay,
  pch = 16, cex.symbols = 0.5, color = "steelblue",
  xlab = "distance", ylab = "air_time", zlab = "dep_delay",
  main = "3D scatter: distance, air_time, dep_delay"
)
```

### 5.2 Optional interactive 3D with `rgl`

```
install.packages("rgl")
library(rgl)

cols <- as.numeric(factor(fl_small$origin))

plot3d(
  x = fl_small$distance,
  y = fl_small$air_time,
  z = fl_small$dep_delay,
  col = cols,
  size = 3,
  xlab = "distance",
  ylab = "air_time",
  zlab = "dep_delay"
)

legend3d("topright",
         legend = levels(factor(fl_small$origin)),
         pch = 16,
         col = seq_along(levels(factor(fl_small$origin))),
         cex = 1)
```

### Tasks (3D visualization)

### Task 5.1

1. Create one static 3D plot using `scatterplot3d`.
2. If possible, create one interactive 3D plot using `rgl`.
3. Explain briefly what extra insight (if any) the 3D view gives compared to 2D.

## 6. Linear models for `air_time` vs `distance`

Now fit and compare two models on the same cleaned data.

For fairness, we ensure positive values and remove missing values:

```
model_df <- fl_small[
  complete.cases(fl_small[, c("air_time", "distance")]) &
    fl_small$air_time > 0 &
    fl_small$distance > 0,
  c("air_time", "distance", "origin")
]

dim(model_df)
```

### 6.1 Model A: original scale

```
m_lin <- lm(air_time ~ distance, data = model_df)
summary(m_lin)
```

Plot with fitted line:

```
plot(model_df$distance, model_df$air_time,
     pch = 19, cex = 0.5, col = "gray40",
     xlab = "distance", ylab = "air_time",
     main = "Model A: air_time ~ distance")
abline(m_lin, col = "red", lwd = 2)
```

### 6.2 Model B: log-log scale

```
m_log <- lm(log(air_time) ~ log(distance), data = model_df)
summary(m_log)
```

Log-log diagnostic plot:

```
plot(log(model_df$distance), log(model_df$air_time),
     pch = 19, cex = 0.5, col = "gray40",
     xlab = "log(distance)", ylab = "log(air_time)",
     main = "Model B: log(air_time) ~ log(distance)")
abline(m_log, col = "blue", lwd = 2)
```

### 6.3 Compare model quality

Compare basic indicators:

```
summary(m_lin)$r.squared
summary(m_log)$r.squared

AIC(m_lin, m_log)
```

Residual checks:

```
par(mfrow = c(1, 2))
plot(fitted(m_lin), resid(m_lin),
     pch = 19, cex = 0.5, col = "gray40",
     xlab = "Fitted", ylab = "Residuals",
     main = "Residuals: Model A")
abline(h = 0, lty = 2)

plot(fitted(m_log), resid(m_log),
     pch = 19, cex = 0.5, col = "gray40",
     xlab = "Fitted", ylab = "Residuals",
     main = "Residuals: Model B")
abline(h = 0, lty = 2)

par(mfrow = c(1, 1))
```

## 7. Tasks for model comparison

### Task 7.1

1. Fit `m_lin` and `m_log`.
2. Report slope estimates from both models.
3. Report `R-squared` values and compare them.

### Task 7.2

1. Inspect residual plots for both models.
2. Decide which model gives residuals that look closer to random noise.
3. Explain in 2-4 sentences which model you would prefer and why.

### Task 7.3 (optional extension)

Add `origin` as a factor predictor and check whether it improves the fit:

```
m_lin2 <- lm(air_time ~ distance + origin, data = model_df)
m_log2 <- lm(log(air_time) ~ log(distance) + origin, data = model_df)

summary(m_lin2)
summary(m_log2)
```

## 8. Wrap-up

In this lab you:

- explored a large real dataset from `nycflights13`,
- created a smaller sample for visualization,
- visualized relationships in 2D and 3D,
- compared two related linear models on original and log-transformed scales.

The same workflow (cleaning -> subsampling -> visualization -> modeling) is common in practical data analysis.