Statistical Computing: Test One

Test One

[LBNSTE002 - Stephan Liebenberg]

2025-02-13

Question 1: Data Cleaning and Summary with *airquality* [15 Marks]

Load the `airquality` dataset, convert it to a `tibble` and convert all columns to numeric:

```r
data("airquality")
airquality <- as_tibble(airquality)
airquality <- airquality |>
  mutate(across(everything(), as.numeric))
airquality <- airquality[complete.cases(airquality), ]
```

Part A [3 Marks]

Question

Output: For each of the `Ozone`, `Solar.R`, `Wind` and `Temp` columns, impute missing values in the `airquality` dataset by replacing them with the median of the respective column.

Answer

```r
airquality <- airquality |>
  mutate(across(c(Ozone, Solar.R, Wind, Temp), ~ ifelse(is.na(.), median(., na.rm = TRUE),
```

Part B [4 Marks]

Question

Table: Find mean, sd, min, max for each of temperature and ozone level. Display the results in a table with appropriate formatting and a caption. Note: need not be cross-referenceable.

Answer

```r
library(dplyr)
library(knitr)

# Compute summary statistics for Temp and Ozone
summary_table <- airquality |>
  summarise(
    Mean_Temp = mean(Temp, na.rm = TRUE),
    SD_Temp = sd(Temp, na.rm = TRUE),
    Min_Temp = min(Temp, na.rm = TRUE),
    Max_Temp = max(Temp, na.rm = TRUE),
    Mean_Ozone = mean(Ozone, na.rm = TRUE),
    SD_Ozone = sd(Ozone, na.rm = TRUE),
    Min_Ozone = min(Ozone, na.rm = TRUE),
    Max_Ozone = max(Ozone, na.rm = TRUE)
  ) |>
  pivot_longer(cols = everything(), names_to = "Statistic", values_to = "Value")

# Display the table with formatting
kable(summary_table, caption = "Summary statistics for Temperature and Ozone levels")
```

Table 1: Summary statistics for Temperature and Ozone levels

Statistic	Value
Mean_Temp	77.792793
SD_Temp	9.529969
Min_Temp	57.000000
Max_Temp	97.000000
Mean_Ozone	42.099099
SD_Ozone	33.275969
Min_Ozone	1.000000
Max_Ozone	168.000000

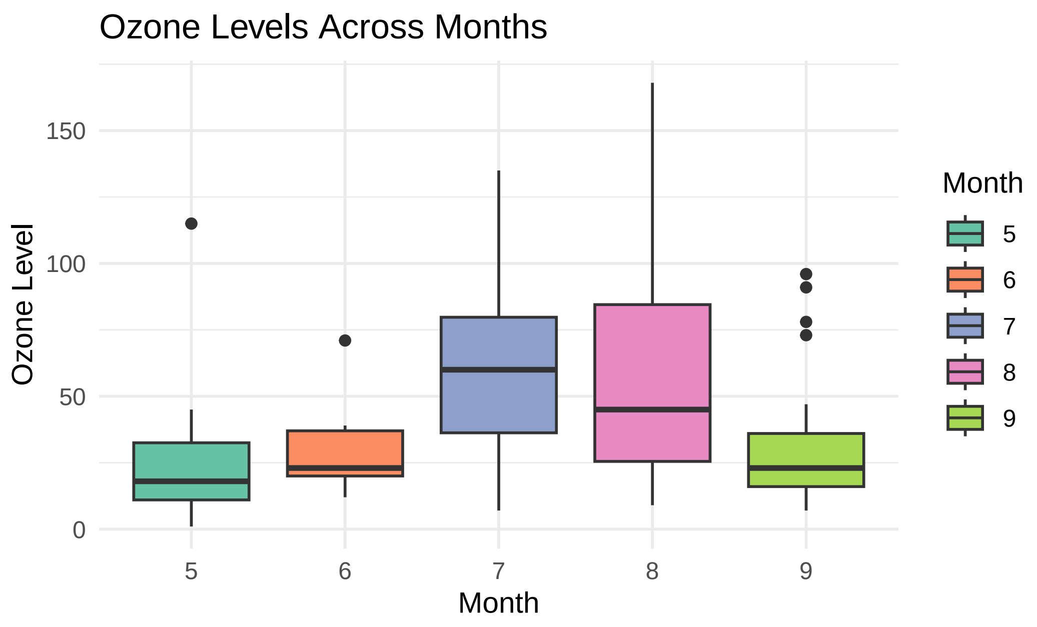
Part C [4 Marks]

Question

Plot: Plot ozone levels for each month *on the same plot* (i.e. don't facet) with `ggplot2`, using different `fill` colours for each month. Use a properly formatted plot with a caption.

Answer

```r
library(ggplot2)

# Create the plot
ggplot(airquality, aes(x = factor(Month), y = Ozone, fill = factor(Month))) +
  geom_boxplot() +
  labs(
    title = "Ozone Levels Across Months",
    x = "Month",
    y = "Ozone Level",
    fill = "Month",
    caption = "Boxplot of Ozone levels for each month using different fill colors."
  ) +
  theme_minimal() +
  scale_fill_brewer(palette = "Set2")
```



Boxplot of Ozone levels for each month using different fill colors.

Part D [4 Marks]

Question

One of two thermometers was randomly selected to measure the temperature each day, as follows:

```r
thermometer_data <- structure(list(Month = c(5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,
5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 6, 6,
6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6,
6, 6, 6, 6, 6, 6, 6, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7,
7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 8, 8, 8, 8,
8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8,
8, 8, 8, 8, 8, 8, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9,
9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9), Day = c(1, 2, 3,
4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20,
21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 1, 2, 3, 4, 5, 6,
7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22,
23, 24, 25, 26, 27, 28, 29, 30, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26,
27, 28, 29, 30, 31, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,
14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29,
30, 31, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16,
17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30), Thermometer = c("A",
"A", "A", "A", "A", "A", "B", "A", "B", "A", "B", "A", "A", "B",
"A", "A", "A", "A", "A", "A", "A", "A", "B", "B", "A", "A", "B",
"A", "B", "A", "A", "B", "A", "A", "B", "A", "B", "B", "B", "B",
"B", "A", "B", "B", "B", "A", "A", "A", "A", "A", "A", "A", "A",
"A", "A", "B", "A", "A", "B", "A", "B", "B", "B", "B", "B", "B",
"B", "B", "B", "B", "A", "B", "B", "A", "B", "B", "B", "B", "B",
"B", "B", "A", "B", "B", "B", "A", "B", "B", "B", "A", "B", "B",
"B", "B", "A", "B", "A", "B", "A", "B", "B", "B", "B", "B", "B",
"B", "B", "B", "A", "A", "A", "A", "B", "A", "A", "A", "A", "B",
"A", "B", "B", "A", "B", "B", "B", "B", "B", "B", "B", "B", "B",
"A", "B", "B", "A", "A", "B", "A", "A", "A", "B", "A", "B", "A",
"B", "A", "A", "A", "A", "A", "A", "A", "A")), row.names = c(NA,
-153L), class = c("tbl_df", "tbl", "data.frame"))
```

i. **Output**: Merge these data with the `airquality` dataset. [1 Mark]

ii. **Output**: Find the average temperature and the standard deviation of temperature for each month and thermometer, and display the results with the following columns: `Mean Temp (A), Mean Temp (B), SD Temp (A), SD Temp (B)`. Use `tidyr::pivot_wider()` to reshape the data. [3 Marks]

Answer

```r
library(dplyr)

# Merge datasets on Month and Day
airquality <- left_join(airquality, thermometer_data, by = c("Month", "Day"))

library(tidyr)

# Group by Month and Thermometer, then compute mean and standard deviation
temp_stats <- airquality %>%
  group_by(Month, Thermometer) %>%
  summarise(
    Mean_Temp = mean(Temp, na.rm = TRUE),
    SD_Temp = sd(Temp, na.rm = TRUE)
  ) %>%
  pivot_wider(
    names_from = Thermometer,
    values_from = c(Mean_Temp, SD_Temp),
    names_glue = "{.value} Temp ({Thermometer})"
  )
```

```
`summarise()` has grouped output by 'Month'. You can override using the
`.groups` argument.
```

```r
# Print results
print(temp_stats)
```

```
# A tibble: 5 x 5
# Groups:   Month [5]
  Month `Mean_Temp Temp (A)` `Mean_Temp Temp (B)` `SD_Temp Temp (A)`
  <dbl>                <dbl>                <dbl>              <dbl>
1     5                 66.6                 66.2               6.45
2     6                 72.6                 85.2               4.72
3     7                 83                   84.1               5.79
4     8                 81                   85.8               6.85
5     9                 71.9                 82.3               5.40
# i 1 more variable: `SD_Temp Temp (B)` <dbl>
```

Question Two: Time series model [5 Marks]

Time series analysis is the analysis of a sequence of random variables over some time interval, such as daily temperatures, stock prices, or website traffic. Some time series exhibit patterns where the current value depends on either past observation or past noise, or both. An **ARMA(1,1) model** (AutoRegressive Moving Average) is one such time series model.

In simple terms:

- **Autoregressive (AR)** means that the value today is influenced by yesterday's value.
- **Moving Average (MA)** means that today's value is also influenced by the random component from the past.

The ARMA(1,1) model follows this equation:

$$X_t = \phi X_{t-1} + \epsilon_t + \theta \epsilon_{t-1},$$

where:

- X_t is the value at time t.
- ϕ is the AR(1) coefficient (influence from the previous value).
- θ is the MA(1) coefficient (influence from past random effects).
- ϵ_t is a random noise term (new shock at time t), $\epsilon_t \sim N(0, \sigma^2)$.
- ϵ_{t-1} is the previous noise term.
- $X_0 = 0$.

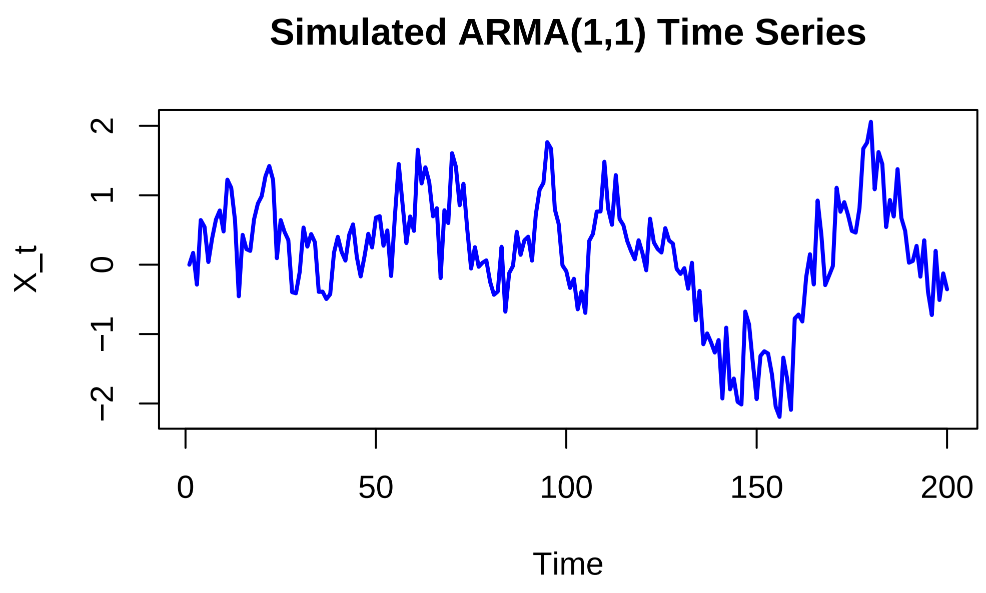
Question [5 Marks]

Plot: Simulate and plot an ARMA(1,1) time series of 200 time units by creating your own function instead of using any R time series packages. For your simulation, let $\phi = 0.9$, $\theta = -0.25$, and $\sigma = 0.5$.

Note: do not delete the code to set the seed below.

Answer

```r
set.seed(1)

simulate_ARMA11 <- function(n, phi, theta, sigma) {
  X <- numeric(n)
  epsilon <- rnorm(n, mean = 0, sd = sigma)

  for (t in 2:n) {
    X[t] <- phi * X[t-1] + epsilon[t] + theta * epsilon[t-1]
  }

  return(X)
}

# Simulating ARMA(1,1)
n <- 200
phi <- 0.9
theta <- -0.25
sigma <- 0.5
arma_series <- simulate_ARMA11(n, phi, theta, sigma)

# Plot the time series
plot.ts(arma_series, main = "Simulated ARMA(1,1) Time Series", ylab = "X_t", xlab = "Time"
```



Question Three: Poisson likelihood [10 Marks]

Suppose you have collected data on the number of website visits per minute, and you believe the number of visits follows a **Poisson distribution** with an unknown rate λ, which represents the expected number of visits per minute.

The **probability mass function** (PMF) of a Poisson-distributed variable is given by:

$$P(X = x|\lambda) = \frac{\lambda^x e^{-\lambda}}{x!}$$

Assume that observations are **independent**, then the likelihood function is:

$$L(\lambda) = \prod_{i=1}^{n} \frac{\lambda^{x_i} e^{-\lambda}}{x_i!}$$

which implies that the **log-likelihood function** is:

$$\ell(\lambda) = \sum_{i=1}^{n} (x_i \log \lambda - \lambda - \log x_i!)$$

Question [10 Marks]

1. Write a custom R function to compute the Poisson likelihood for a given λ, without using `dpois()`. [2 marks]
2. **Plot**: Using **both** the small and large data defined below, plot the likelihood for $\lambda \in [2, 4]$, indicating λ which **visually** maximises the function (i.e. the approximate MLE) and true value for λ. This should be visible for the small data, but R may have difficulty with larger data. [4 marks]
3. **Plot**: Write a custom R function to compute the log-likelihood instead. Again, plot the log-likelihood for both datasets and visually indicate the approximate MLE and true value for λ. [4 marks]

Hint: The functions `factorial` and `lfactorial` may be beneficial.

```r
set.seed(2)
lamTrue <- runif(1, 2, 4)
xSmall <- rpois(10, lamTrue)

# Example large dataset
xLarge <- rpois(1000, lamTrue)
```

Answer

```r
poisson_likelihood <- function(lambda, x) {
  if (lambda <= 0) {
    stop("Lambda must be positive")
  }

  likelihood <- prod((lambda^x * exp(-lambda)) / factorial(x))
  return(likelihood)
}
```