ST2137 AY 24/25 Sem 2

Assignment 1 Review

Contents

1 R portion

The file `aircraft_failure.txt` contains information on the time between breakdowns of 10 aircrafts. Each line in the file corresponds to a particular aircraft. For instance, aircraft number 9 first broke down 418 hours after it was commissioned. Following that repair, it broke down 18 hours later again, and so on.

1.1 Question 1

1. Read the data into R and create a dataframe named `ftimes_df` with two columns. Here are the first few rows of the dataframe:

The tricky thing about this data was that it was not in a nice tabular format. The code below uses `sapply` to split each line by a space. `vapply` is another form of `sapply` - it is used to extract the number of failures for each aircraft.

Throughout lectures and tutorials, I have been stressing the importance of using the same folder structure. When you work in teams, you will have to follow similar rules as well. If you had used a different path setting, or used an absolute path, you would get 1 mark deducted.

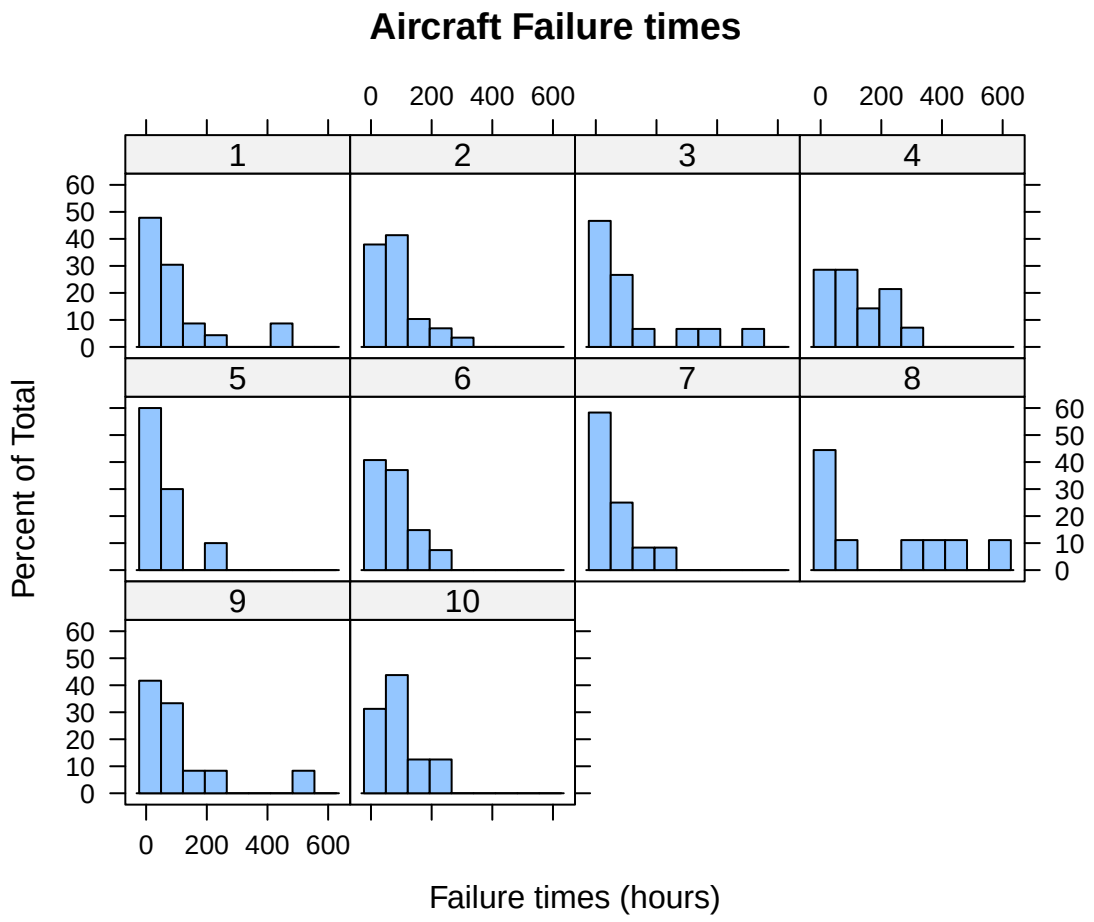
```
tmp <- readLines("data/aircraft_failure.txt",warn = FALSE)
ftimes_list <- sapply(tmp, function(x) as.integer(strsplit(x, " ")[[1]]),
                      USE.NAMES = FALSE)
d_len <- vapply(ftimes_list, length, 1L)
a_names_vec <- rep(1:10, d_len)

ftimes_df <- data.frame(aircraft = a_names_vec, failure_times = unlist(ftimes_list))
head(ftimes_df)
```

```
  aircraft failure_times
1        1           413
2        1            14
3        1            58
4        1            37
5        1           100
6        1            65
```

1.2 Question 2

2. Create a lattice plot of histograms for each aircraft. Ensure that the plot has a title and proper axis labels.

```
library(lattice)
histogram( ~ failure_times | as.factor(aircraft),
           xlab="Failure times (hours)",
           main="Aircraft Failure times",
           data=ftimes_df, as.table=TRUE)
```



Many of you used incorrect y-axis here. The default plot from this function is to use "percent of total". However, many changed this to "frequency" or "count". This is incorrect. To obtain density or count, add an argument `type="density"` or `type="count"`.

1.3 Question 3

Overall, this question was well done. However, several students implemented the function wrongly, using `var(x)` instead of `mean(x^2)`. The answer for aircraft one had been provided for you as a first check, but it was overlooked.

Another difference was that several functions returned a dataframe with one row. No one lost marks for this, but you would have observed a different output (there would have been a row number in front of the output).

```
mom_gamma <- function(x) {
  x_bar <- mean(x)
  x2_bar <- mean(x^2)
```

```r
  alpha_hat <- 1/(x2_bar/x_bar^2 - 1)
  beta_hat <- x_bar/alpha_hat

  out <- c(alpha_hat, beta_hat)
  names(out) <- c("alpha", "beta")
  out
}
ac1 <- ftimes_df$failure_times[ftimes_df$aircraft == 1]
mom_gamma(ac1)
```

```
     alpha        beta
  0.6727961 142.2357426
```

1.4 Question 4

Here is one approach to solving this problem:

```r
ac_vec <- rep(0.0, 10)

for (i in 1:10) {
  x  <- ftimes_df$failure_times[ftimes_df$aircraft == i]
  ac_vec[i] <- cor(x[-1], x[-length(x)])
}
```

1.5 Overall

Overall, I believe the R portion was very well done. I do believe most people are following the material well; just pay attention to small details, like path settings and formulas used.

In terms of marking scheme, there was 1 mark for each question. If you did not use the correct path setting, or there were errors in your code, you would have lost the final mark.

2 Python portion

2.1 Question 1

To read in the data and create the grouped version, it is best to use the `groupby` method of the dataframe.

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats

housing_df = pd.read_excel("data/housing.xlsx")
grouped_df = housing_df.groupby(['housing', 'contact', 'satisfaction'],
                                as_index=False).sum('count')
```

2.2 Question 2

The purpose of the statement is to reshape the data, from long format into a wide format. The second form is the typical form we see contingency tables presented to us in, and it is the form that is needed for the subsequent barcharts to be created.

```python
housing_types = grouped_df['housing'].unique()
housing_data = grouped_df[grouped_df.housing == housing_types[0]]
housing_data
```

```
      housing contact satisfaction  count
```

```
0  apartments    high         high    191
1  apartments    high          low    141
2  apartments    high       medium    116
3  apartments     low         high    111
4  apartments     low          low    130
5  apartments     low       medium     76
```

```
housing_data_pivot = housing_data.pivot(index='contact',
                                        columns='satisfaction', values='count')
housing_data_pivot
```

```
satisfaction  high  low  medium
contact
high           191  141     116
low            111  130      76
```

2.3 Question 3

Here is the suggested solution for this question:

```
# Define the order for 'contact' and 'satisfaction'
contact_order = ['low', 'high']
satisfaction_order = ['low', 'medium', 'high']
housing_order = grouped_df['housing'].unique()

# Create a 4x2x3 numpy array with zeros
array_4x2x3 = np.zeros((len(housing_order), len(contact_order), len(satisfaction_order)))

# Populate the array
for i, housing in enumerate(housing_order):
    for j, contact in enumerate(contact_order):
        for k, satisfaction in enumerate(satisfaction_order):

            value = grouped_df[
                (grouped_df['housing'] == housing) &
                (grouped_df['contact'] == contact) &
                (grouped_df['satisfaction'] == satisfaction)]['count'].values

            # Assign the value to the array (default to 0 if no value is found)
            array_4x2x3[i, j, k] = value[0] if len(value) > 0 else 0

array_4x2x3
```

```
array([[[130.,  76., 111.],
        [141., 116., 191.]],

       [[ 27.,  24.,  31.],
        [ 37.,  55.,  65.]],

       [[ 40.,  24.,  31.],
        [ 93.,  50.,  39.]],

       [[ 65.,  54., 100.],
        [ 34.,  47., 100.]]])
```

2.4 Question 4

For this question, it is important to go through all five steps.

Step 2

The null and alternative hypotheses are:

$$\begin{aligned} H_0 &: \text{Contact and satisfaction are indepdendent.} \\ H_1 &: \text{Contact and satisfaction are not indepdendent.} \end{aligned}$$

The tests are to be conducted at 5% significance level.

Steps 1, 3 & 4

```
for i, housing in enumerate(housing_order):
    print(f"For {housing}:")
    chisq_output = stats.chi2_contingency(array_4x2x3[i])
    print(f"The p-value is {chisq_output.pvalue:.4f}.")
    print(f"The test statistic value is {chisq_output.statistic:.4f}.")
    if np.any(chisq_output.expected_freq < 5):
        print("Expected cell count assumption violated!")
    else :
        print("Expected cell counts all at least 5.")
    print("---")
```

```
For apartments:
The p-value is 0.0206.
The test statistic value is 7.7670.
Expected cell counts all at least 5.
---
For atrium houses:
The p-value is 0.2898.
The test statistic value is 2.4771.
Expected cell counts all at least 5.
---
For terraced houses:
The p-value is 0.1185.
The test statistic value is 4.2656.
Expected cell counts all at least 5.
---
For tower blocks:
The p-value is 0.0361.
The test statistic value is 6.6422.
Expected cell counts all at least 5.
---
```

Step 5

We conclude that at 5% level, we conclude that for *apartments* and for *tower blocks*, the variables contact and satisfaction are not independent.

At the same significance level, for *atrium houses* and *terraced houses*, we do not reject the null hypothesis of independence between contact and satisfaction.

2.5 Overall

A very common mistake made here was to comment on observed cell counts, and not *expected* cell counts.

In terms of marking, there questions 1 - 3 were worth 1 mark each, and the final question was worth 2 marks.