Tutorial 3

ST2137-2420

Material

This tutorial covers the topics and concepts from chapter 3 of the course textbook. The questions offer practice on understanding numerical data through plots, numerical summaries, comparisons with theoretical distributions, and associations.

Dataset: Student Performance

R code

```r
stud_perf <- read.table("data/student/student-mat.csv", sep=";",
                        header=TRUE)
```

Python code

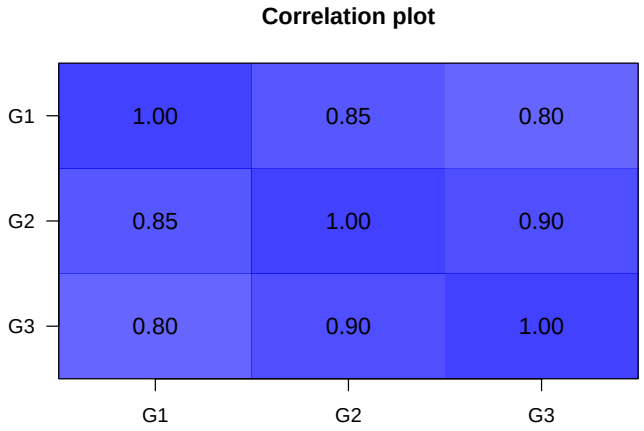
```python
import pandas as pd
import numpy as np

stud_perf  = pd.read_csv("data/student/student-mat.csv", delimiter=";")
```

1. Create and plot a correlation matrix for the output variables G1, G2 and G3 for the student performance data.

R code

```r
library(psych)
cor_g123 <- cor(stud_perf[, c("G1", "G2", "G3")])
corPlot(cor_g123, cex=0.8, show.legend = FALSE)
```



Python code

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import linregress
from scipy.special import gammaln

cor_g123 = stud_perf[['G1', 'G2', 'G3']].corr()
cor_g123 = stud_perf[['G1', 'G2', 'G3']].corr()
cor_g123.style.background_gradient(cmap='coolwarm_r',
                                   axis=None, vmin=-1, vmax=1)
```

Table 1

	G1	G2	G3
G1	1.000000	0.852118	0.801468
G2	0.852118	1.000000	0.904868
G3	0.801468	0.904868	1.000000

There is a high correlation between the three variables. The variable that has the highest association with G3 is G2 (second grade period).

2. Using the columns `G1`, `G2` and `G3`, create a new dataframe with 2 columns: `grade_type` and `grade_score`. There should be $395 \times 3 = 1185$ rows in the new dataframe. The unique values in `grade_type` column should be `final`, `second` and `first`. Here are some sample rows:

R code

```
grade_score <- c(stud_perf$G1, stud_perf$G2, stud_perf$G3)
grade_type <- rep(c("first", "second", "final"), each=395)
df2 <- data.frame(grade_type = grade_type, grade_score = grade_score)
```

Python code

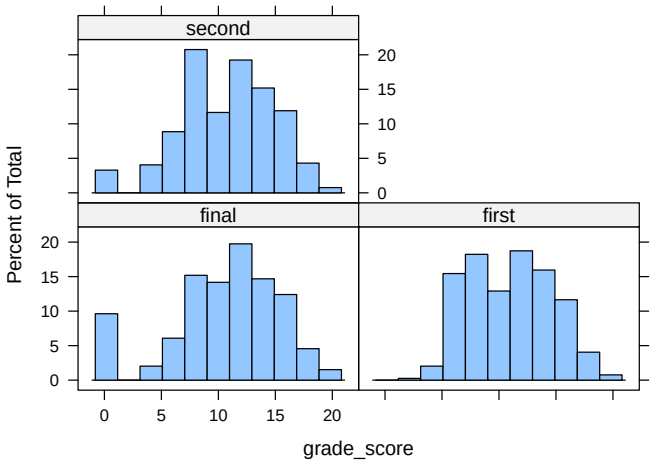
```
grade_type = pd.Series(np.repeat(['first', 'second', 'final'], repeats=395))
grade_score = pd.Series(stud_perf.G1.to_list() + stud_perf.G2.to_list() +
```

```
                        stud_perf.G3.to_list())
df2 = pd.DataFrame({'grade_score': grade_score, 'grade_type': grade_type})
```

4. Create histograms for G1, G2, G3; summarise and compare them.

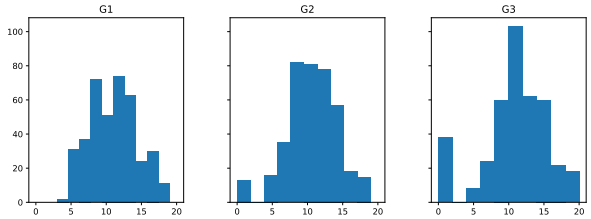
R code

```
library(lattice)
histogram(~ grade_score | grade_type, data=df2)
```



Python code

```
stud_perf[['G1', 'G2', 'G3']].hist(layout=(1,3), figsize=(12, 4),
                                   grid=False, sharex=True, sharey=True);
```



The histograms for `first` appears unimodal and symmetric. The tails are thin. The histogram for `second` appears more left-skewed than the one for `first`. There are also students with a score of 0. The histogram for `final` has a large number of 0 scores. Apart from those students, the rest of the histogram looks unimodal and symmetric.

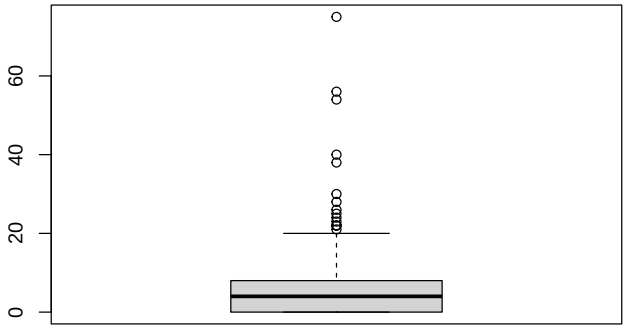
5. Create a boxplot of the single numeric variable `absences` using `boxplot()`, and also the 5-number summary using `summary()`. Extract the rows corresponding to the outliers and study them. What would you investigate next? *Hint: see the help page for information on the object returned by* `boxplot`.

R code

```
summary(stud_perf$absences)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.000   0.000   4.000   5.709   8.000  75.000
```

```r
box_out <- boxplot(stud_perf$absences)
```

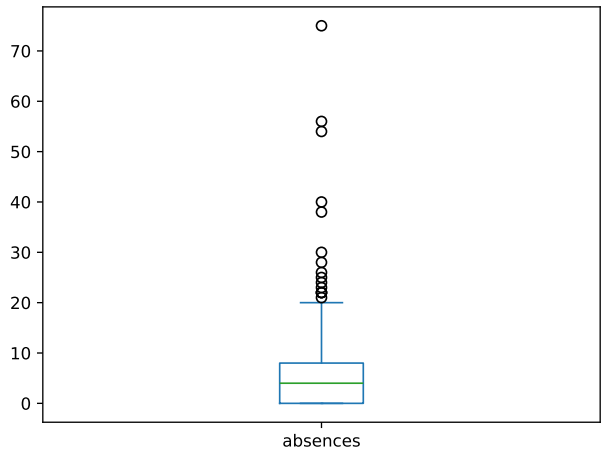



```r
outlier_points <- stud_perf[stud_perf$absences >= min(box_out$out),]
```

Python code

```python
stud_perf.absences.plot(kind='box')
iqr = np.quantile(stud_perf.absences, 0.75) - np.quantile(stud_perf.absences, 0.25)
upper_fence = np.quantile(stud_perf.absences, 0.75) + 1.5*iqr

#stud_perf[stud_perf.absences > upper_fence]
```



Among the outliers, there seems to be a higher proportion of females than in the general population, and there also seems to be a higher proportion of romantically involved students than the general population. These could be follow-up hypotheses to consider.

Assessing "Poisson-ness" of a Dataset

The dataset in `er_arrivals.csv` contains the number of arrivals to an Emergency Room in U.K. over 13 months in the 1960s. It is commonly assumed (to begin with) that the number of arrivals on each day Y_j follows a Poisson pmf:

$$P(Y_j = k) = \frac{e^{-\lambda}\lambda^k}{k!}, \quad k = 0, 1, 2, ...$$

To test/visualise this assumption, we can make a Poisson-ness plot. Here's how it works. First, suppose we observe $Y_1, Y_2, ..., Y_N$ and we wish to assess if it follows a Poisson distribution. We compute the count of counts:

$$X_k = \sum_{j=1}^{N} I(Y_j = k) \quad \text{for } k = 0, 1, 2, ...$$

where

$$I(Y_j = k) = \begin{cases} 1, & Y_j = k \\ 0, & Y_j \neq k \end{cases}$$

Thus X_k will be the number of times that Y_j took on the value of k. We let L be the maximum observed value of k, i.e. $\sum_{k=0}^{L} X_k = N$. We have observed our entire sample, so we consider N to be a fixed integer.

$$\begin{aligned} E(X_k) &= N \times \frac{e^{-\lambda}\lambda^k}{k!} \\ \text{(Taking logs on both sides)} \quad \ln E(X_k) &= \ln N - \lambda + k\ln(\lambda) - \ln(k!) \end{aligned}$$

Thus, using X_k as an estimate of $E(X_k)$, a plot of $\phi_k = \ln(k!X_k/N)$ (on the y-axis) against k (on the x-axis) should yield a straight line with slope equals to $\ln(\lambda)$. Here is how this plot can be used:

- Systematic deviations, e.g. curvature from this line indicate that the Poisson distribution is unsuitable for this data.
- A regression can be fitted to estimate λ (slope will be $\ln(\lambda)$).

6. Read the data as a data frame `er_arrivals` in R/Python, ensuring that the date column is in a suitable format.

R code

```r
er_arrivals <- read.csv("data/er_arrivals.csv")
er_arrivals$date <- as.Date(er_arrivals$date)
```

Python code

```python
er_arrivals = pd.read_csv("data/er_arrivals.csv")
er_arrivals['date'] = pd.to_datetime(er_arrivals['date'])
```

7. Create a Poisson-ness plot for this data.

R code

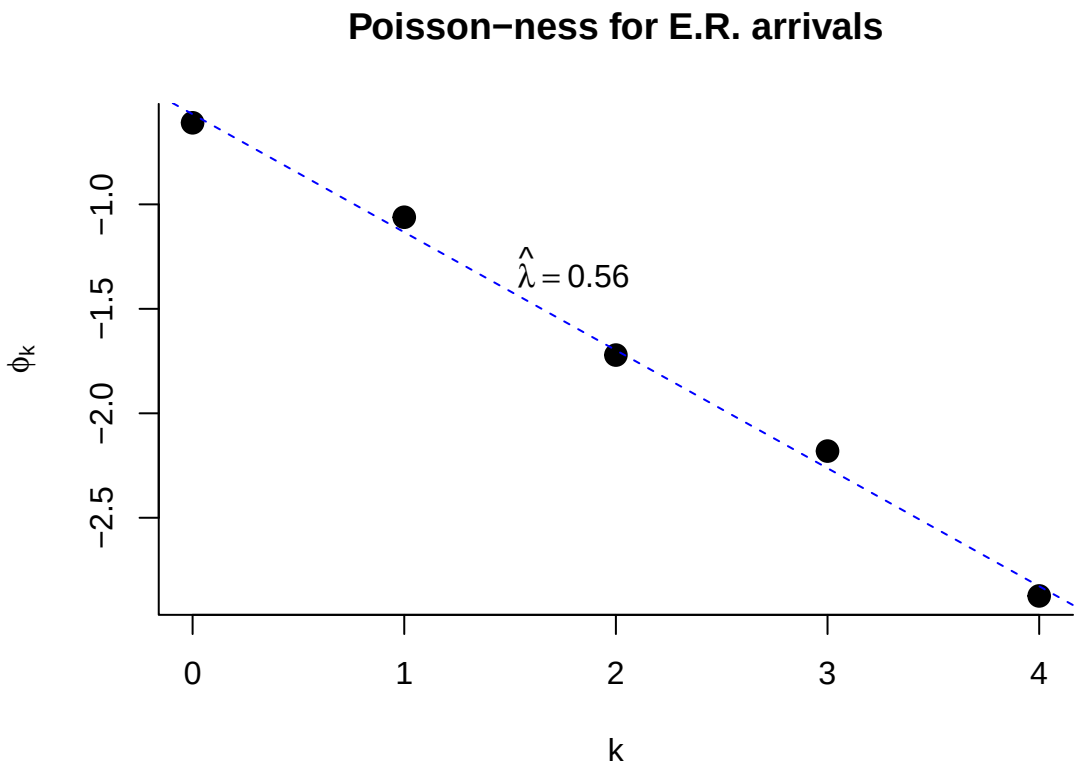
```r
Yk <- er_arrivals$num_arrivals
Xk <- table(Yk)
k <- as.integer(names(Xk))

Xk <- as.vector(Xk)
N <- length(Yk)

phi <- lfactorial(k) + log(Xk/N)
```

```r
# compute lam_hat from slope
lm1 <- lm(phi ~ k)
slope <- unname(coef(lm1)[2])
lam_hat <- exp(slope)

plot(k, phi, ylab=expression(phi[k]), pch=19,  cex=1.5, bty='l',
     main="Poisson-ness for E.R. arrivals")
abline(b=slope, a=-lam_hat, lty=2, col="blue")
text(1.8, -1.3, labels=expression(hat(lambda) == "0.56"))
```




```r
cat("poissonness lam-hat is ", sprintf("%.3f", lam_hat), ".\n", sep="")
```

```
poissonness lam-hat is 0.568.
```

Python code

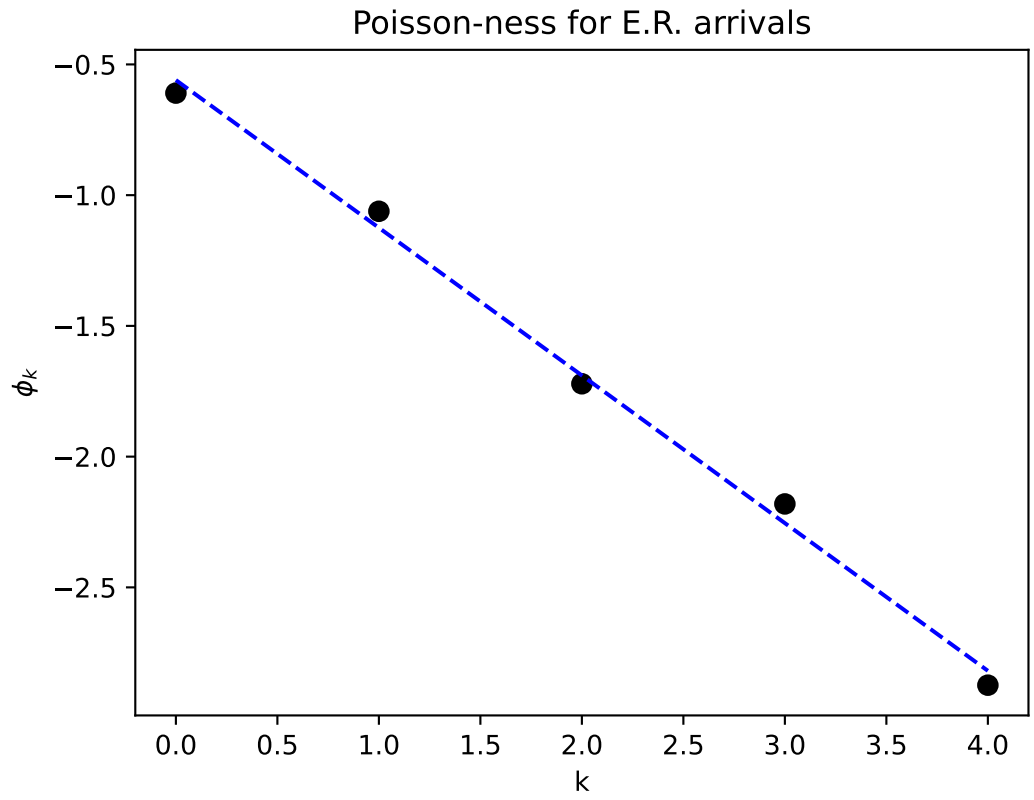
```python
# Step 1: Extract Yk and compute Xk
Yk = er_arrivals['num_arrivals']
Xk = Yk.value_counts().sort_index()  # Frequency table
k = Xk.index.to_numpy()  # Unique arrival values
Xk = Xk.to_numpy()  # Counts for each unique value

# Step 2: Compute N and phi
N = len(Yk)
phi = gammaln(k + 1) + np.log(Xk / N)

# Step 3: Compute lam_hat from slope
slope, intercept, _, _, _ = linregress(k, phi)
```

```
lam_hat = np.exp(slope)

# Step 4: Plotting
plt.scatter(k, phi, label=r'$\phi_k$', color='black', s=50);
plt.plot(k, slope * k + intercept, linestyle='--', color='blue', label=f"Slope: {slope:.2f}");
plt.xlabel('k');
plt.ylabel(r'$\phi_k$');
plt.title("Poisson-ness for E.R. arrivals");
#plt.legend()
```



8. Compute two different estimates of λ:
    1. Using the sample mean (this is also the MLE).
    2. Using the slope from the Poisson-ness plot

R code

```
    lam_hat_mean <- mean(Yk)
    lam_hat_slope <- lam_hat
    cat("Estimate from mean is ", sprintf("%.2f", lam_hat_mean), ".\n",
        "Estimate from slope is ", sprintf("%.2f", lam_hat_slope), ".\n", sep="")
```

```
Estimate from mean is 0.59.
Estimate from slope is 0.57.
```

Python code

```
lam_hat_mean = Yk.mean()
lam_hat_slope = lam_hat
```

```
print(f"""
Estimate from mean is {lam_hat_mean:.2f}.
Estimate from slope is {lam_hat_slope:.2f}.
""")
```

```
Estimate from mean is 0.59.
Estimate from slope is 0.57.
```