Sekhar, Arjun (44940076) - Stat8123 Assignment 3

Arjun Sekhar

25/10/2021

The data set

```
setwd("~/OneDrive/Assessments/University/Year 4/Semester 2/Stat8123/Tutorials/Assignment 3")
pedestrians_2021 <- read.csv("pedestrians_2021.csv", header = TRUE)

library(pander)
pander(head(pedestrians_2021, n=5))
```

Sensor	Date_Time	Date	Time	Count
231 Bourke St	2020-12-31T13:00:00Z	2021-01-01	0	424
Alfred Place	2020-12-31T13:00:00Z	2021-01-01	0	14
Birrarung Marr	2020-12-31T13:00:00Z	2021-01-01	0	250
Bourke St - Spencer St (North)	2020-12-31T13:00:00Z	2021-01-01	0	213
Bourke St - Spencer St (South)	2020-12-31T13:00:00Z	2021-01-01	0	124

Question 1

In this question we intend to analyse our time series plot of pedestrian counts in the inner city of Melbourne during January to September 2021.

```
# Step 1: Import the 'ggplot2' and the 'dplyr' package.
library(ggplot2)
library(dplyr)
```

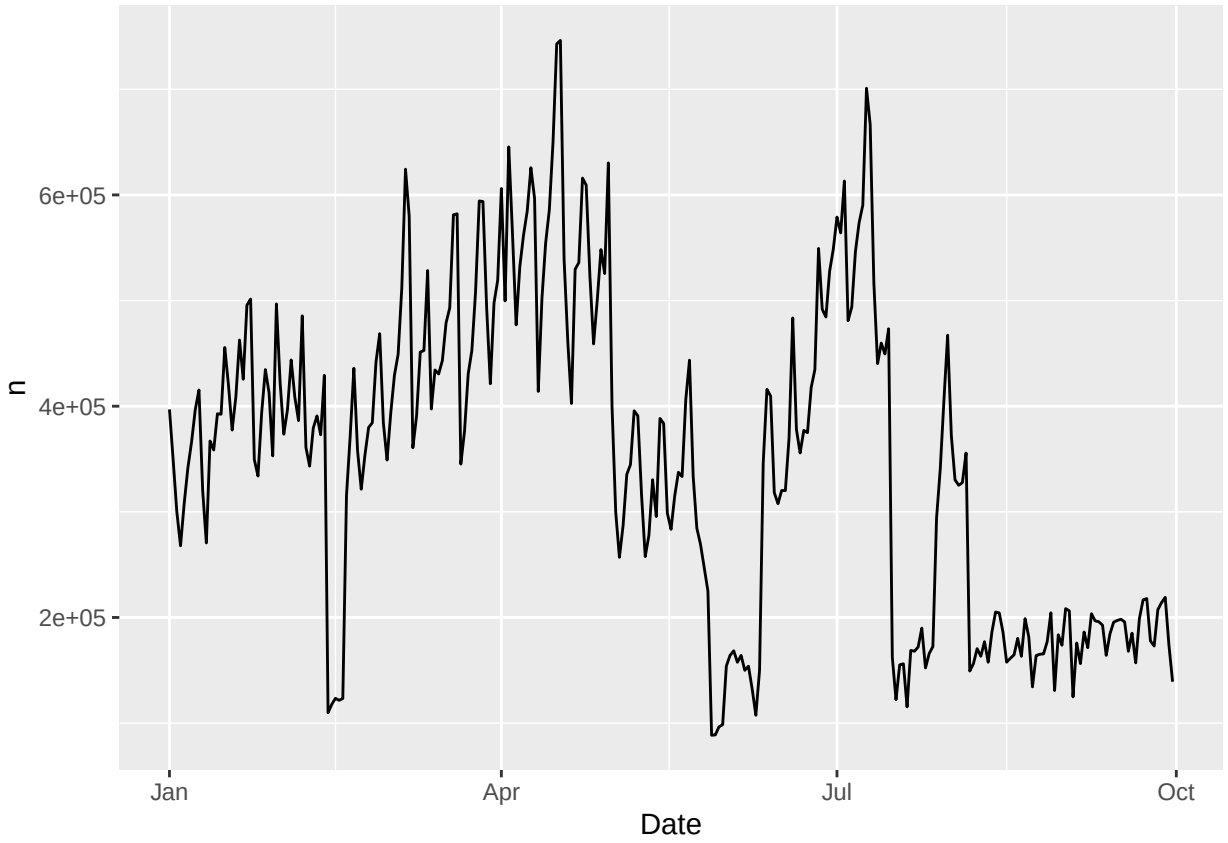
```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
# Step 2: Separate the date from being a string to separate values.
pedestrians_2021$Date <- as.Date(pedestrians_2021$Date)

# Step 3: Group, summarise the pedestrian information.
ped_ts <- pedestrians_2021 %>%
  group_by(Date) %>%
  summarise(n=sum(Count))
```

```
# Step 4: Plot the time series graph using 'geom_line()' to display the effect.
ggplot(ped_ts, aes(x=Date, y=n)) + geom_line()
```



From the plot itself, it is a time series plot which tells us the number of pedestrians in the inner city of Melbourne between January to September 2021.

In this plot we can notice that during the early part of the year (say January and February), the daily pattern is fairly standard, albeit a little lower than usual. There was a drop in the population around March and then it picked up again in April, before it came down drastically from May to June - this tells us that Melbourne was potentially amidst another one of their lockdowns, which basically dropped the pedestrian population by around 25%. Eventually once the lockdowns ended, it soared back up.

An interesting aspect to notice is that they had another lockdown around a month ago, but their pedestrian numbers in inner Melbourne was incredibly low, which aligned with the rise in COVID-19 cases in the state. There were protests happening but the numbers were no where enough to replicate the heights of the periods when there were no lockdowns.

Question 2

This question involves us creating plots for the monthly and weekly counts on the pedestrians and we want to find which month had the largest pedestrian count, as well as whether there was a clear pattern in terms of the day of the week.

A)

For the Monthly pattern, we can observe the following occur:

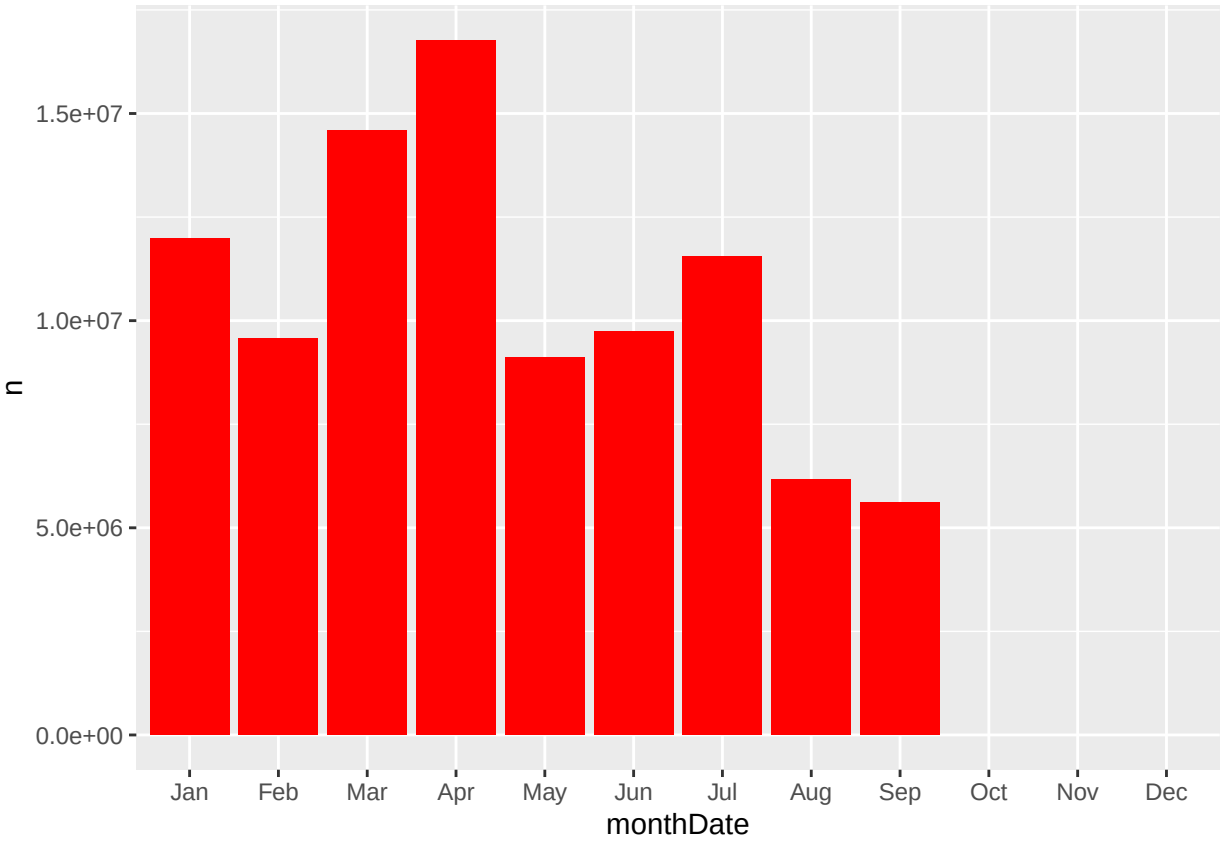
```
# Step 1: Load the packages 'ggplot2' and 'lubridate'
library(ggplot2)
```

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
# Step 2: Group the dates by creating another column named 'Date'.
pedestrians_2021$Date <- as.Date(pedestrians_2021$Date)

# Step 3: Analyse the pedestrian counts on the basis of the Month.
ped_month <- pedestrians_2021 %>%
  mutate(monthDate = month(Date)) %>%
  group_by(monthDate) %>%
  summarise(n=sum(Count))

# Step 3: Plot the bar graph.
ggplot(ped_month, aes(x = monthDate, y = n)) +
  geom_bar(stat = "identity", fill = "red") +
    scale_x_discrete(limits = month.abb)
```

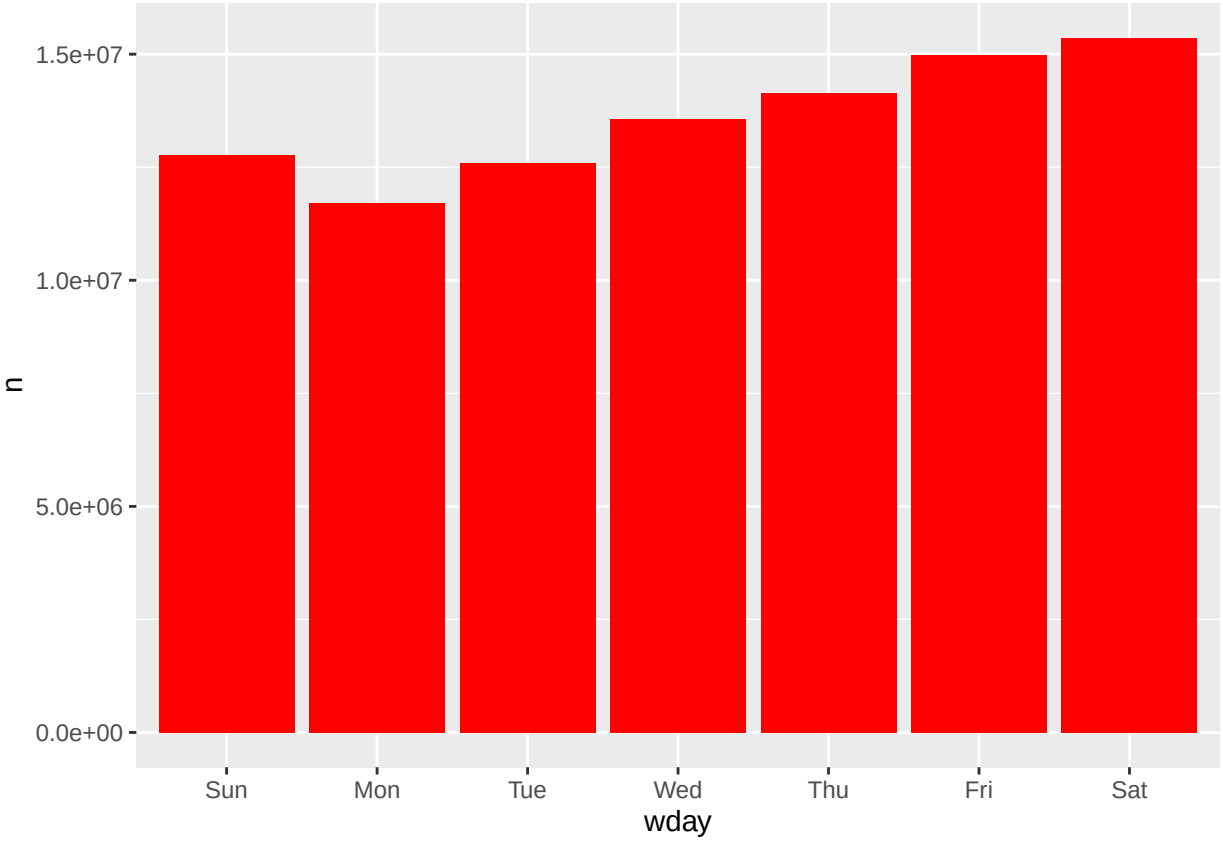


Observing this chart by eyeballing, we can see that the month with the largest pedestrian count was on the 4th month of the year, which is **April**. Some other popular months in this study were March, January and July - all of which were during the immediate aftermaths of a lockdown.

B)

We can now go forth in creating our *weekly* pedestrian count and subsequently we can go about analysing whether there is a clear day of the week pattern that is visible for the pedestrian count.

```
# Step 1: Import the packages 'ggplot2' and 'lubridate'.
library(ggplot2)
library(lubridate)

# Step 2: Although we have done this before, we group the date.
pedestrians_2021$Date <- as.Date(pedestrians_2021$Date)

# Step 3: Analyse the pedestrian count based on the days of the week.
ped_days <- pedestrians_2021 %>%
  mutate(wday = wday(Date, label = TRUE)) %>%
  group_by(wday) %>%
  summarise(n=sum(Count))

# Step 4: Plot the bar chart as follows.
ggplot(ped_days, aes(x = wday, y = n)) + geom_bar(stat = "identity", fill = "red")
```



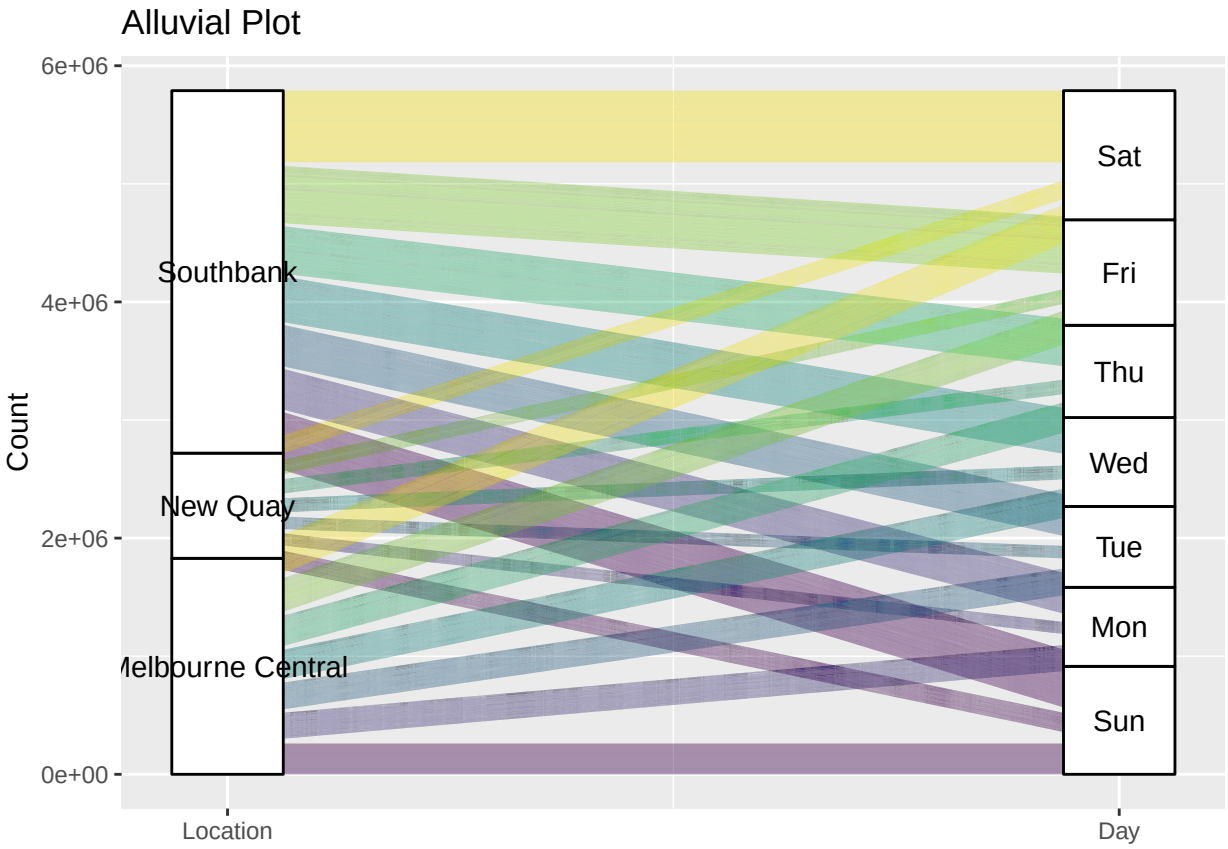
This plot has a progressive day of the week pattern, which means that while Mondays are least popular (possibly due to the lack of motivation to move on a Monday morning), as the week wears on, the movement increases - going as far as peaking on a Saturday. The increase in movement could be matched with the level of enthusiasm among the citizens of Melbourne as the week progresses. More so, it goes to show that due to people being quite busy during early parts of the week, as it wears on, there is more of a movement that is happening.

Question 3

The alluvial plot here shall describe the traffic flow at the locations Melbourne Central, New Quay and Southbank and we aim to analyse it by the days of the week. Our main package shall be the *ggalluvial* package, which will generate our Alluvial plot.

```
# Step 1: Input the 'ggalluvial' package
library(ggalluvial)

# Step 2: Mutate the date by rearranging it a little
ped_2021 <- pedestrians_2021 %>% mutate(date = ymd(Date))
ped_2021 <- ped_2021 %>% mutate(wday = wday(Date, label = TRUE))

# Step 3: Pick out the 3 locations (Melbourne Central, New Quay and Southbank)
ped_2021 <- ped_2021[ped_2021$Sensor %in% c("Melbourne Central","New Quay","Southbank"), ]

# Step 4: Use the 'ggplot' package with the alluvium plot.
ggplot(ped_2021, aes(y = Count, axis1 = Sensor, axis2 = wday)) +
  geom_alluvium(aes(fill = wday), width = 0, knot.pos = 0, reverse = FALSE) +
  guides(fill = FALSE) +
  geom_stratum(width = 1/8, reverse = FALSE) +
  geom_text(stat = "stratum", aes(label = after_stat(stratum)), reverse = FALSE) +
  scale_x_continuous(breaks = 1:3, labels = c("Location", "Day", "Time")) +
  ggtitle("Alluvial Plot")
```

```
## Warning: `guides(<scale> = FALSE)` is deprecated. Please use `guides(<scale> =
## "none")` instead.
```



Describing our plot, we can see that our main variables to consider are the Location and Days (both in the

x-axis), and then the Count (in the y-axis). Basically, the plot itself is a useful ranking tool for us to see which days that a particular location is popular. It thereby allows us to infer the pedestrian traffic flow.

The following facts that can be inferred are that:

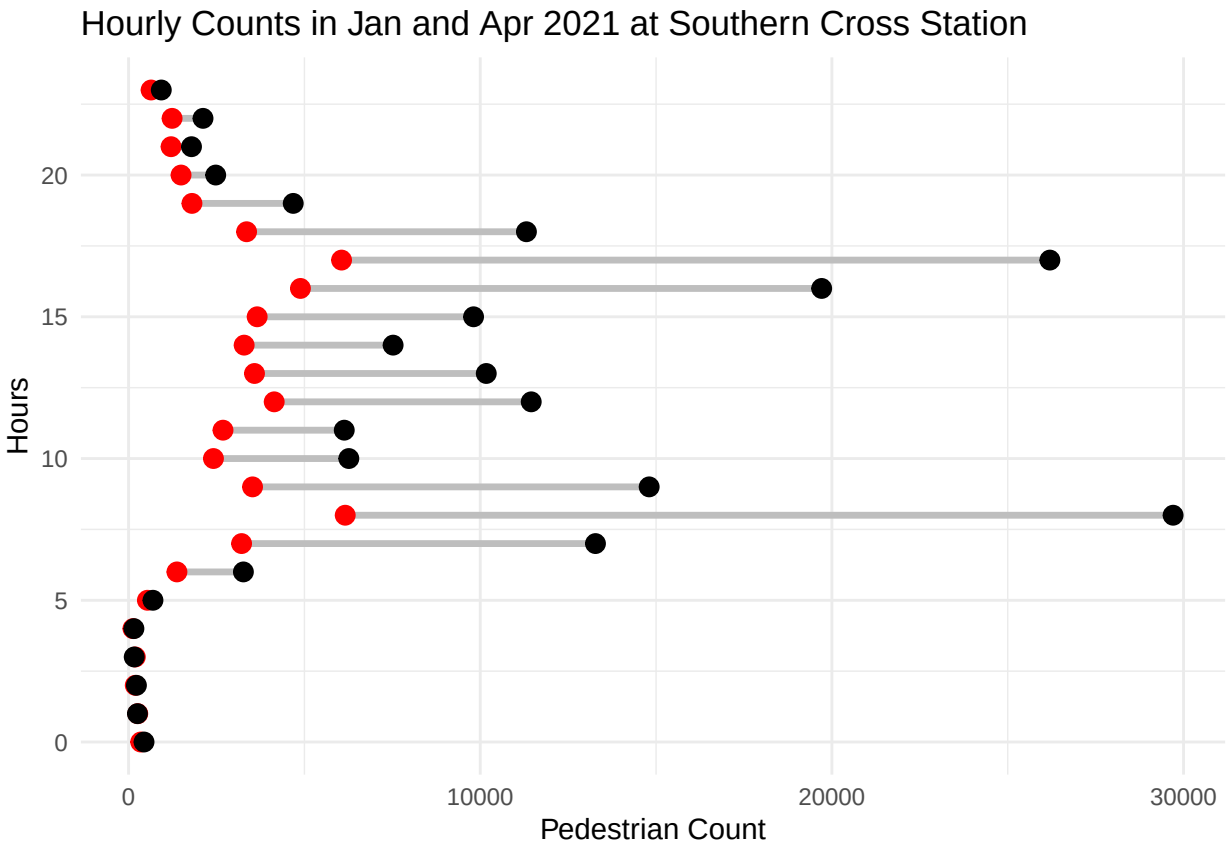
- Southbank is a popular spot for pedestrians to walk through.
- Sundays are understandably the least popular days for the pedestrians to cross, given there is little activity on Sundays.
- New Quay is a more popular destination during the weekends, potentially because it attracts visitors, given it is by the sea side.

The above aspects prove important because they tell us the travel patterns of pedestrians, but also, given the frequency of COVID-19 lockdowns in Melbourne, the consistency of these visits could also be questionable.

Question 4

Our aim here is to analyse hourly pedestrian counts during January and April 2021 at Southern Cross Station using a dumbbell plot. We adopt the following steps below to approach this.

```
# Step 1: Import the packages 'tidyr', 'dplyr', 'readr' and 'ggalt'.
library(tidyr)
library(dplyr)
library(readr)
library(ggalt)

# Step 2: Mutate the 'Date' variable.
ped_hour <- pedestrians_2021 %>% mutate(date = ymd(Date))

# Step 3: Assign this mutation as a new column 'month'.
ped_hour$month <- month(ped_hour$date)

# Step 4: Filter out the hourly counts for Southern Cross Station
hourly_counts <- filter(ped_hour,
                        Sensor == "Southern Cross Station",
                        month %in% c(1, 4)) %>% select(Time, month, Count)

# Step 5: Group these hourly counts by 'Time' and 'month'
hourly_counts <- hourly_counts %>%
  group_by(Time, month) %>%
  summarise(n = sum(Count))

# Step 6: Pivot the hourly counts on the basis of the months.
pivot <- pivot_wider(hourly_counts, names_from = month, values_from = n)
names(pivot) <- c("hour", "Jan", "Apr")

# Step 7: Plot the dumbbell ggplot.
ggplot(pivot, aes(y = hour, x = Jan, xend = Apr)) +
  geom_dumbbell(size = 1.2, size_x = 3, size_xend = 3, colour = "grey",
                colour_x = "red", colour_xend = "black") +
  theme_minimal() +
  labs(title = "Hourly Counts in Jan and Apr 2021 at Southern Cross Station",
       x = "Pedestrian Count", y = "Hours")
```



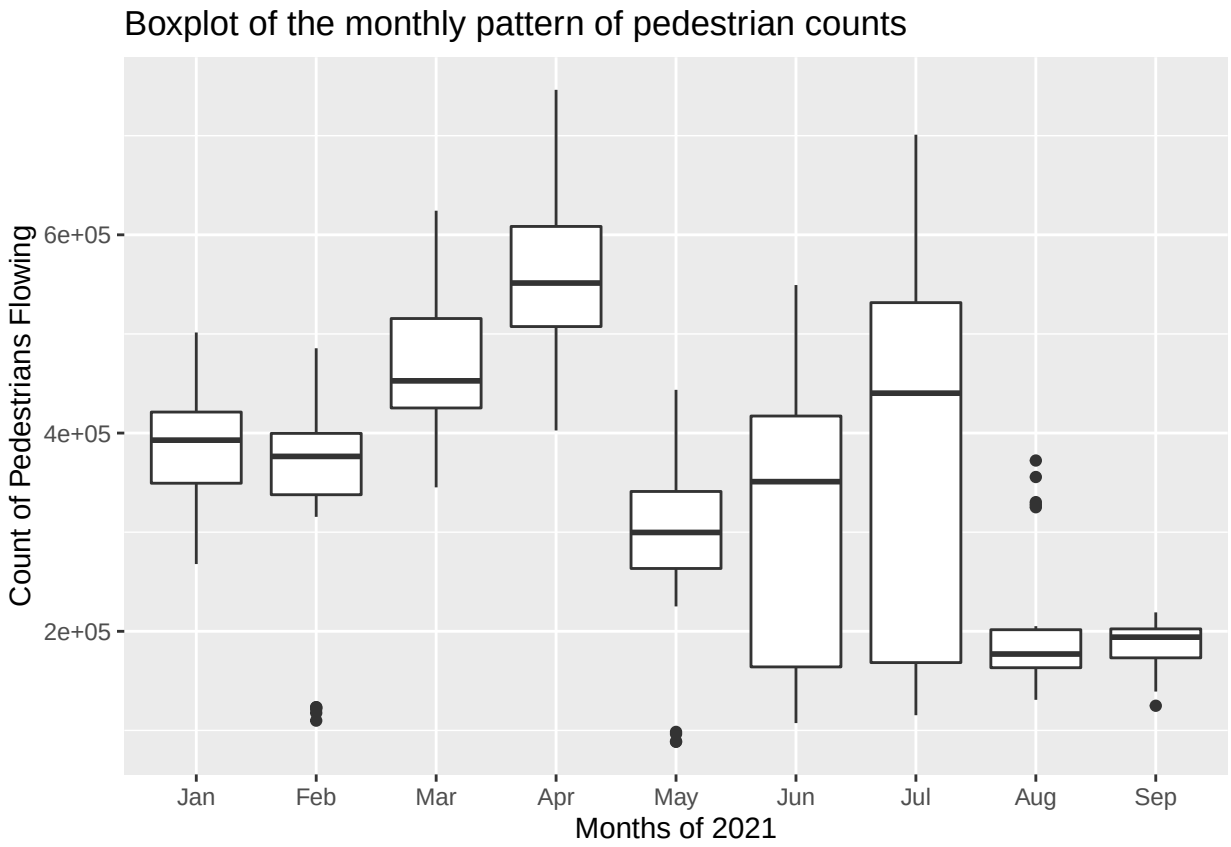
The above plot displays a span of the pedestrian count across the 24 hours of the day. As expected, we can see that during the timings late at night and the early morning the spread of the dumbbells is rather small, which is definitely an unsurprising result. In contrast though as we go from 08:00 to 17:00, there is a definitive activity. The activity is rather slow during the day because people would be working and so have less time to stroll around.

Question 5

Firstly, we take the boxplot to compare the monthly pattern of the pedestrian counts and from that, the following can be noted.

```
# Step 1: Load the packages 'ggplot2', 'lubridate' and 'modelr'.
library(ggplot2)
library(lubridate)
library(modelr)

# Step 2: Analyse the pedestrian counts on the basis of the month.
ped_month <- pedestrians_2021 %>%
  mutate(monthDate = ymd(Date)) %>%
  group_by(monthDate) %>%
  summarise(n=sum(Count)) %>%
  mutate(month = month(monthDate, label = TRUE))

# Step 3: Plot the boxplot for the months.
ggplot(ped_month, aes(x = month, y = n)) +
  geom_boxplot() +
  ggtitle("Boxplot of the monthly pattern of pedestrian counts") +
  ylab("Count of Pedestrians Flowing") +
```

```
  xlab("Months of 2021")
```



We can see here that early in the year (January and February), the spread of the pedestrian count is minimal and we can see that the median of the pedestrian flow was fairly similar. There was then an increase in the median and a moderate spread in the box-and-whisker of the March and April months but this was before a sharp drop in May.

The drop in May was arguably due to another lockdown that occurred in that period, which deeply hit the confidence of pedestrians to travel actively along inner Melbourne. Subsequently in June and July once that lockdown ended, there was a wide-spread in the pedestrian flow of June and July - possibly signalling the joy of the post-lockdown freedoms.

Unfortunately the increase of cases in the Delta strain resulted in August having a stunning decline and by September the citizens of Melbourne were hit with yet another lockdown, which effectively carried over till the conclusion of this study.

```
linear_mod <- lm(n ~ month - 1, data = ped_month)
summary(linear_mod)
```

```
##
## Call:
## lm(formula = n ~ month - 1, data = ped_month)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -257465  -37761    5166   50473  328191
##
## Coefficients:
##          Estimate Std. Error t value Pr(>|t|)
```

```
## monthJan   386700      18889  20.472   <2e-16 ***
## monthFeb   342186      19875  17.217   <2e-16 ***
## monthMar   470807      18889  24.925   <2e-16 ***
## monthApr   559024      19201  29.114   <2e-16 ***
## monthMay   293846      18889  15.557   <2e-16 ***
## monthJun   324457      19201  16.898   <2e-16 ***
## monthJul   372809      18889  19.737   <2e-16 ***
## monthAug   199136      18889  10.543   <2e-16 ***
## monthSep   186721      19201   9.724   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 105200 on 264 degrees of freedom
## Multiple R-squared:  0.9261, Adjusted R-squared:  0.9236
## F-statistic: 367.5 on 9 and 264 DF,  p-value: < 2.2e-16
```

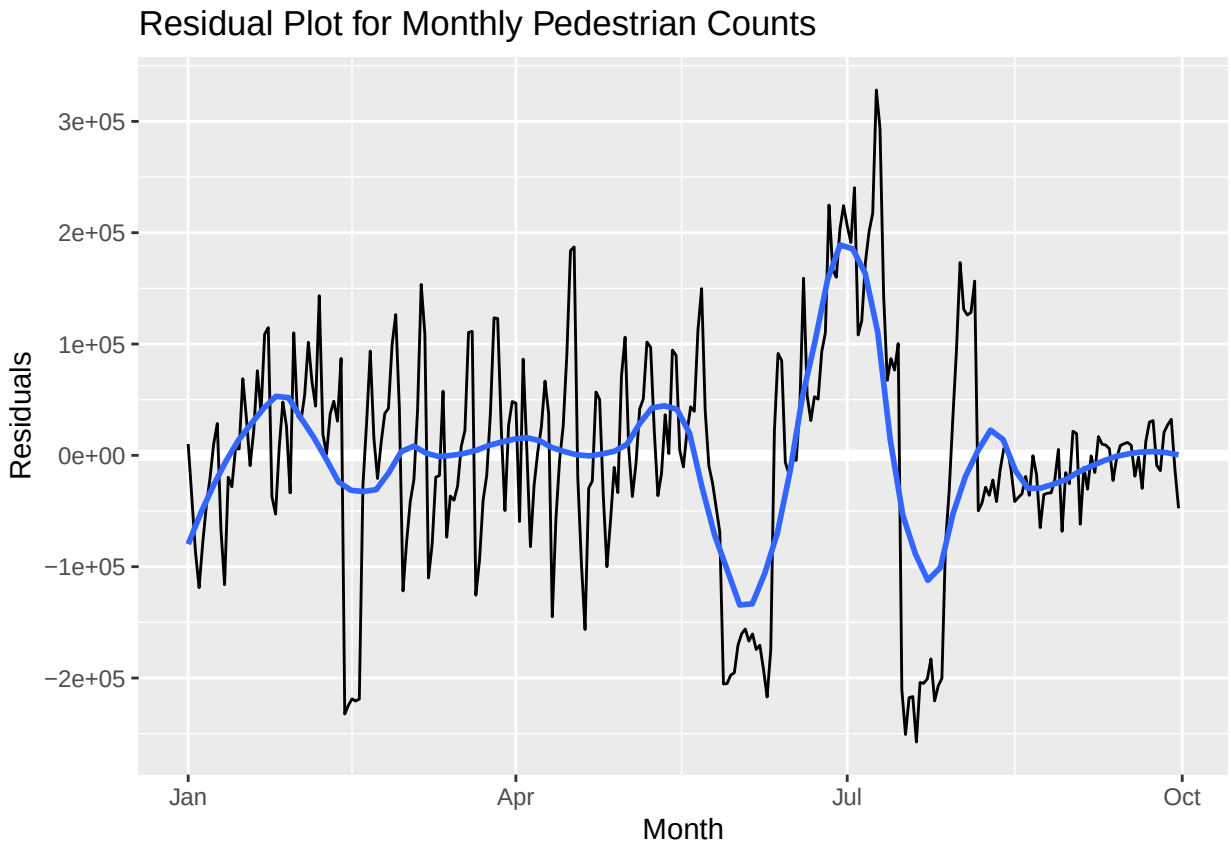
Above we have fitted a linear model to depict the monthly pedestrian counts and we can see that the variables signify the following:

$\widehat{Count}_i = 386700(Jan_i) + 342186(Feb_i) + 470807(Mar_i) + 559024(Apr_i) + 293846(May_i) + 324457(Jun_i) + 372809(Jul_i) + 199136(Aug_i) + 186721(Sep_i)$

- $Count_i$ = Number of Pedestrians that pass all through Melbourne
- Jan_i = Month of January
- Feb_i = Month of February
- Apr_i = Month of April
- May_i = Month of May
- Jun_i = Month of June
- Jul_i = Month of July
- Aug_i = Month of August
- Sep_i = Month of September

```r
# Step 1: Indicate the residuals by taking the 'modelr' package.
library(modelr)

# Step 2: Filter out the residuals from our linear model.
ped_month <- ped_month %>% add_residuals(linear_mod)

# Step 3: Plot the residuals by also adding a loess curve.
ped_month %>%
  ggplot(aes(monthDate, resid)) +
  geom_ref_line(h=0) +
  geom_line() +
  ggtitle("Residual Plot for Monthly Pedestrian Counts") +
  xlab("Month") +
  ylab("Residuals") +
  geom_smooth(method = "loess", se = FALSE, span = 0.2)
```

```
## `geom_smooth()` using formula 'y ~ x'
```



Analysing this residual plot, we can still see that the plot itself aligns with what we noticed in the earlier analysis of the respective months aligning with the lockdown times. It goes to show that months where there were lockdowns (i.e. an inactivity), the residual diminishes drastically. For example, months such as June and September were highly inactive unlike April and July, which peaked in pedestrian activity. This in summary tells us that the model has performed in consistency with the ground reality.

Question 6

```
# Step 1: Filter out the 10 dates where linear_mod fails
ped_filter <- ped_month %>%
  slice_max(n = 10, abs(resid))

# Step 2: Display the data
library(pander)
pander(ped_filter)
```

monthDate	n	month	resid
2021-07-09	701000	Jul	328191
2021-07-10	666602	Jul	293793
2021-07-20	115344	Jul	-257465
2021-07-17	122095	Jul	-250714
2021-07-03	613166	Jul	240357
2021-02-13	109875	Feb	-232311
2021-06-26	549334	Jun	224877
2021-02-14	117671	Feb	-224515
2021-06-30	548724	Jun	224267
2021-02-16	121527	Feb	-220659

Based on my knowledge about Melbourne, we can see that the residual values that have been filtered are mostly from July, as well as February and June. The interesting aspect of this is that most of these values are from periods that COVID-19 cases were low in Melbourne (and Victoria) and lockdowns did not occur. Right after these periods was when cases rose and restrictions were tightened, signifying that there was a sharp change in the flow of pedestrians.

Possible reasons for the misfit itself would be because the data was not standardised and that a transformation (particularly of the response variable) would have been useful. This passively tells us to consider model transformations of variables, as well as to consider alternative distributions as well.

A linear model is not quite the best choice for this kind of data because it is not completely standardised and besides we need to worry about the distribution of the data. The data as a linear model was not tested in those ways, which meant that the cleanliness of the model was not quite substantial.

Question 7

Research Question: Develop a linear model to describe the monthly pattern of pedestrians in Southbank.

Approach: Firstly we shall establish our model for Southbank, wherein we use the *Date* (as a function of the months) and *Count* variables to express this relationship.

```
# Step 1: Filtering out the 'Southbank' data in a frame.
Southbank <- pedestrians_2021[pedestrians_2021$Sensor %in% c("Southbank"), ]

# Step 2: Expressing this as a linear model
Southbank_lm <- lm(Count ~ months(Date)-1, data = Southbank)
summary(Southbank_lm)
```

```
##
## Call:
## lm(formula = Count ~ months(Date) - 1, data = Southbank)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1077.4  -408.3  -108.7   329.7  3214.5
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## months(Date)April       1087.42      32.54   33.41   <2e-16 ***
## months(Date)August       386.42      25.21   15.33   <2e-16 ***
## months(Date)February     550.25      44.56   12.35   <2e-16 ***
## months(Date)January      623.32      23.41   26.63   <2e-16 ***
## months(Date)July         671.24      22.64   29.65   <2e-16 ***
## months(Date)June         591.93      23.01   25.72   <2e-16 ***
## months(Date)May          943.48      22.64   41.68   <2e-16 ***
## months(Date)September    387.53      23.01   16.84   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 617.5 on 4768 degrees of freedom
## Multiple R-squared:  0.5477, Adjusted R-squared:  0.547
## F-statistic: 721.8 on 8 and 4768 DF,  p-value: < 2.2e-16
```

We can see from the above that March is missing, which is rather unusual given we are trying to model it across the 9 months. In saying that, we can also note that the model picks out the month of *April* as the

contrast month (which is can be viewed like an intercept). This means that the model itself can be expressed as follows

$\widehat{Count}_i = 623.3(Jan_i)+550.2(Feb_i)+1087.4(April_i)+386.4(May_i)+591.9(Jun_i)+671.2(Jul_i)+386.4(Aug_i)+387.5(Sep_i)$

Where (once again):

- $Count_i$ = Number of Pedestrians that pass through Southbank
- Jan_i = Month of January
- Feb_i = Month of February
- Apr_i = Month of April
- May_i = Month of May
- Jun_i = Month of June
- Jul_i = Month of July
- Aug_i = Month of August
- Sep_i = Month of September

We can now tidy up our model output through the process of using the *broom* package. It can be displayed and expressed as follows:

```
library(broom)
```

```
##
## Attaching package: 'broom'
```

```
## The following object is masked from 'package:modelr':
##
##     bootstrap
```

```
library(pander)
pander(tidy(Southbank_lm), caption = "Coefficients for Southbank")
```

Table 3: Coefficients for Southbank

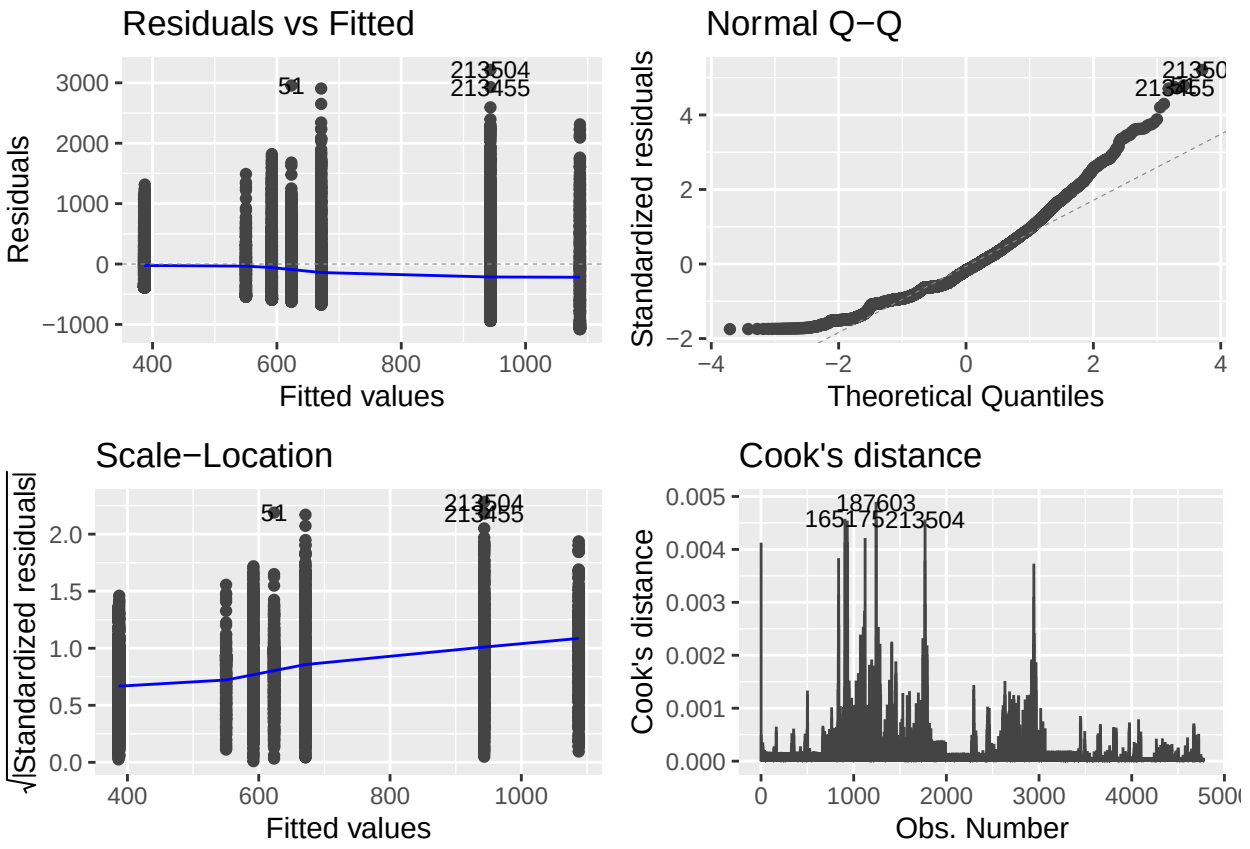
term	estimate	std.error	statistic	p.value
months(Date)April	1087	32.54	33.41	3.961e-220
months(Date)August	386.4	25.21	15.33	8.291e-52
months(Date)February	550.2	44.56	12.35	1.682e-34
months(Date)January	623.3	23.41	26.63	8.566e-146
months(Date)July	671.2	22.64	29.65	1.813e-177
months(Date)June	591.9	23.01	25.72	9.563e-137
months(Date)May	943.5	22.64	41.68	0
months(Date)September	387.5	23.01	16.84	7.392e-62

Now analysing the model diagnostics, we can see the following:

```
# We choose 'ggfortify' to plot our residuals using a ggplot technique.
library(ggfortify)
```

```
## Registered S3 method overwritten by 'ggfortify':
##   method        from
##   fortify.table ggalt
```

```
autoplot(Southbank_lm, which = 1:4, label.size = 3)
```



Analysing each of these plots, we can see the following:

- **Residuals vs. Fitted:** There is not much of a random scatter around 0, due to there being an observable pattern and no constant variance. We can also notice some values standing out (it is premature in this study to call them outliers), but they are highly negligible.
- **Normal Q-Q Plot:** There is a clear departure from normality that can be noticed along the top-right and the bottom-left of the plot. This is a sign that the model can be better improved if further analysed.
- **Scale Location:** From this plot, while there is approximately a constant variance about 1 the smoothing line does not depict this quite as much because it indicates the variance ranging between 0.75 and 1.00 (it is approaching 1.00).
- **Cook's Distance:** As spotted in the Residuals vs. Fitted plot, we could see that there were some values that stood out, but the Cook's Distance tells us that they are not quite large (usually these need to be greater than 1). It means that we can conclude that the Cook's Distance reveals no substantial outliers.

We can overall see that while the above model is an average model for the sake of the study, it could be improved by fitting another distribution as a generalised linear model (GLM). It could be noted that certain diagnostics were not met, which is not a deterrant to choosing the model altogether but in a further study it would be beneficial to tame the model.