STATS 326/786

Code

Assignment 01

4/03/24

General comments:

- All the plots should be labelled appropriately (axes, legends, titles). There will be marks allocated for this.
- Please submit both your `.Rmd`, and the generated output file `.html` or `.pdf` on Canvas before the due date/time.
- Please make sure that the `.Rmd` file compiles without any errors. The marker will not spend time fixing the bugs in your code.
- Please avoid specifying absolute paths.
- Your submission must be original, and if we recognize that you have copied answers from another student in the course, we will deduct your marks.
- You must use `tidyverse` packages to answer the questions in this assignment. Please use `dplyr` for data wrangling/manipulation, `ggplot2` for data visualisation, and `lubridate` for dates/times. Some parts of Problem 2 will use plots from the `fpp3` packages.
- IMPORTANT NOTE: There are some questions that are for STATS 786 only. Students taking STATS 326, while you are welcome to attempt these questions, please do not submit answers to them.

Due: Friday 15 March 2023 at 16:00 PM (NZ time)

Problem 1: Kobe Bryant

The `lakers` data set (in the `lubridate` package) contains play-by-play statistics of each Los Angeles Lakers basketball game in the 2008-2009 regular season. It contains the following variables:

Variable	Description
`date`	Date of the game
`opponent`	Name of the opposition team
`game_type`	Home or away game
`time`	Time remaining on the game clock in a given period (counting down from 12 minutes)
`period`	The period of play (most games have four quarters, each 12 minutes in duration, noting that some games go into a 5-minute duration overtime if tied at the end of regular play)
`etype`	The type of play made (e.g., shot, turnover, rebound)
`team`	Name of the NBA team the player who made the play belongs to
`player`	Name of the player that the play was made by
`result`	Whether they won or lost the game
`type`	A more detailed description of the type of play made
`x`	The x-coordinate on the field of play (in ft)
`y`	The y-coordinate on the field of play (in ft)

1. 6 Marks

- Read in the `lakers` data set and convert this into a `tibble` object.
- Keep only the rows relating to Kobe Bryant. Name this object `kobe` .
- Transform the `date` variable into a `lubridate` date format (noting that it is currently in integer format).
- Shot location is given by `x` and `y` . The center of the hoop is located at the coordinates $(25, 5.25)$. Center the `x` and `y` variables to $(0, 0)$; you will want to overwrite `x` and `y` in your `kobe` data set.

Hide

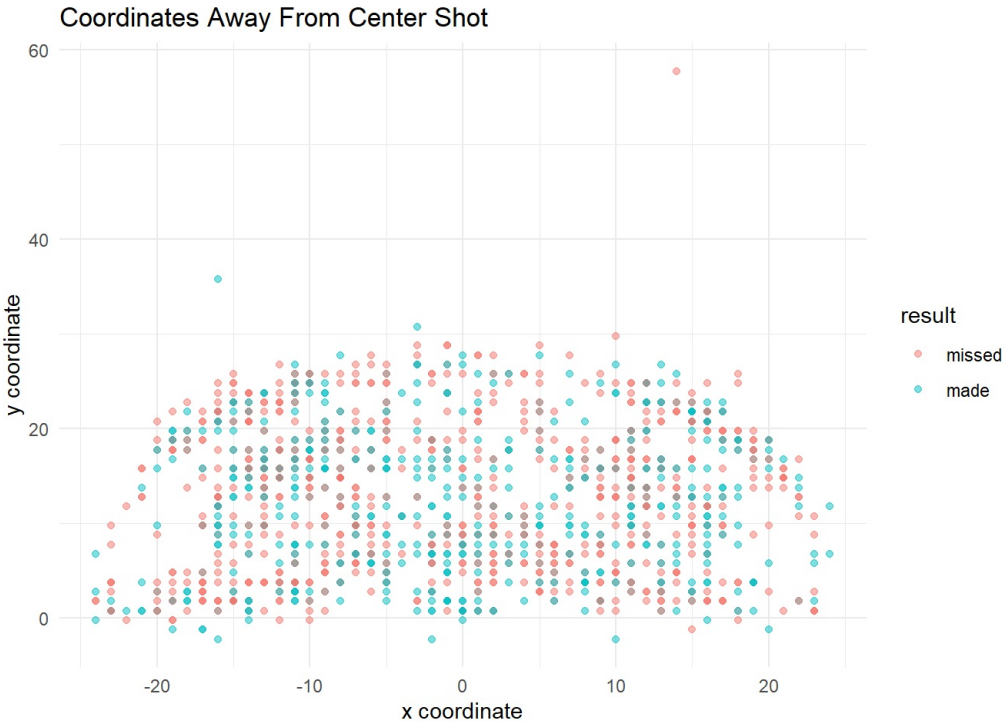
```
#Read lakers data set
data(lakers)
lakers.df <- as_tibble(lakers)

#filter for Kobe Bryant
kobe <- lakers.df %>% filter(player == 'Kobe Bryant')

#convert to date format
kobe$date <- ymd(kobe$date)

#center the x and y variables
kobe$x <- kobe$x - 25
kobe$y <- kobe$y - 5.25
```

2. 6 Marks

- Subset the `kobe` data set by only considering plays that are shot attempts (i.e., where `etype` is equal to `shot`). Name this new data set `kobe.shot`.
- Make a scatter plot of the centered shot location, colouring the points by `result`. You should use the `geom_point` layer.
- Set the transparency of the points to `alpha = 0.5`.
- Use the default colour scheme, but reverse the colour order so that shots made is green(ish) and shots missed is red(ish). Hint: You can use `scale_colour_discrete` with an additional argument.

Hide

```
#filter for attempted shots
kobe.shot <- kobe %>% filter(etype == 'shot')

#create scatter plot
kobe.shot %>% ggplot(mapping = aes(x = x, y = y, col = result)) +
  geom_point(alpha = 0.5) +
  scale_colour_discrete(limits = c("missed", "made")) +
  ggtitle("Coordinates Away From Center Shot") +
  xlab("x coordinate") +
  ylab("y coordinate") +
  theme(legend.title = element_blank()) +
  theme_minimal()
```


Coordinates Away From Center Shot
60
40
20
0
y coordinate
-20
-10
0
10
20
x coordinate
result
missed
made

3. 6 Marks

- Using the `kobe.shot` data set, produce a 2-dimensional density plot (with contours) of Kobe Bryant's shot locations. You will want to use both `geom_density_2d_filled` and `geom_density_2d`. Do not colour by `result`.
- Remove the legend using `legend.position` argument in the `theme` layer.

Hide

```
#create 2-dimensional density plot
kobe.shot %>% ggplot(mapping = aes(x = x, y = y)) +
  geom_density_2d_filled() +
  geom_density_2d(color = "white") +
  ggtitle("2D Contours of Shot Locations") +
  xlab("x coordinate") +
  ylab("y coordinate") +
  theme(legend.position = "none")
```

4. 9 Marks

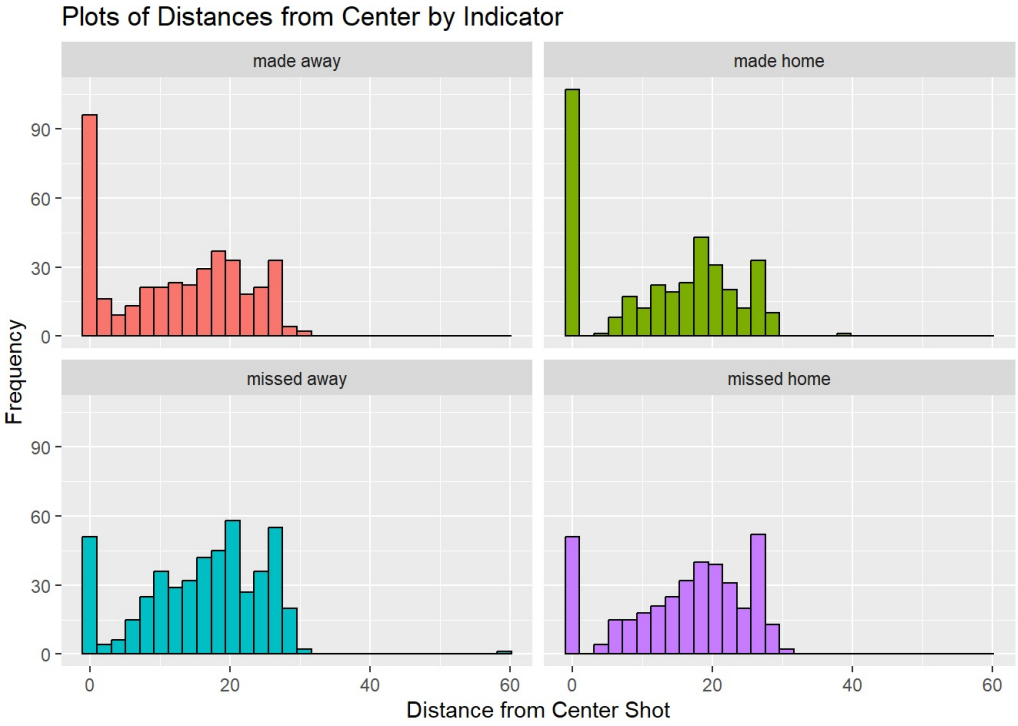
- Within the `kobe.shot` data set, create a variable called `distance` that calculates the distance a shot was taken from hoop. You will need to use Pythagoras' theorem, i.e., distance $= \sqrt{x^2 + y^2}$.
- Then create another variable within your `kobe.shot` data set called `indicator` that concatenates the values of `result` with `game_type`. Hint: You can use the `paste` function. You should end up with a variable in your data set that takes on the four values: "made home", "made away", "missed home", "missed away".
- Plot histograms showing the distribution of distance using `geom_histogram`. Use `facet_wrap` to create seperate panels for all values of `indicator`. (You should end up with four panels on the same figure).
- Fill the histograms by `indicator` such that the interior of the bars are different colours for the four different groups.
- Remove the legend.

Hide

```
#create distance variable
kobe.shot <- kobe.shot %>% mutate(distance = (x^2 + y^2)^(0.5))

#create indicator variable
kobe.shot <- kobe.shot %>% mutate(indicator = paste(result, game_type, sep = " "))

#create histograms
kobe.shot %>% ggplot(mapping = aes(x = distance, fill = indicator, col = indicator)) +
  geom_histogram(alhpa = 0.25, color = "black") +
  xlab("Distance from Center Shot") +
  ylab("Frequency") +
  ggtitle("Plots of Distances from Center by Indicator") +
  theme(legend.position = "none") +
  facet_wrap(~indicator)
```

```
## Warning in geom_histogram(alhpa = 0.25, color = "black"): Ignoring unknown
## parameters: `alhpa`
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

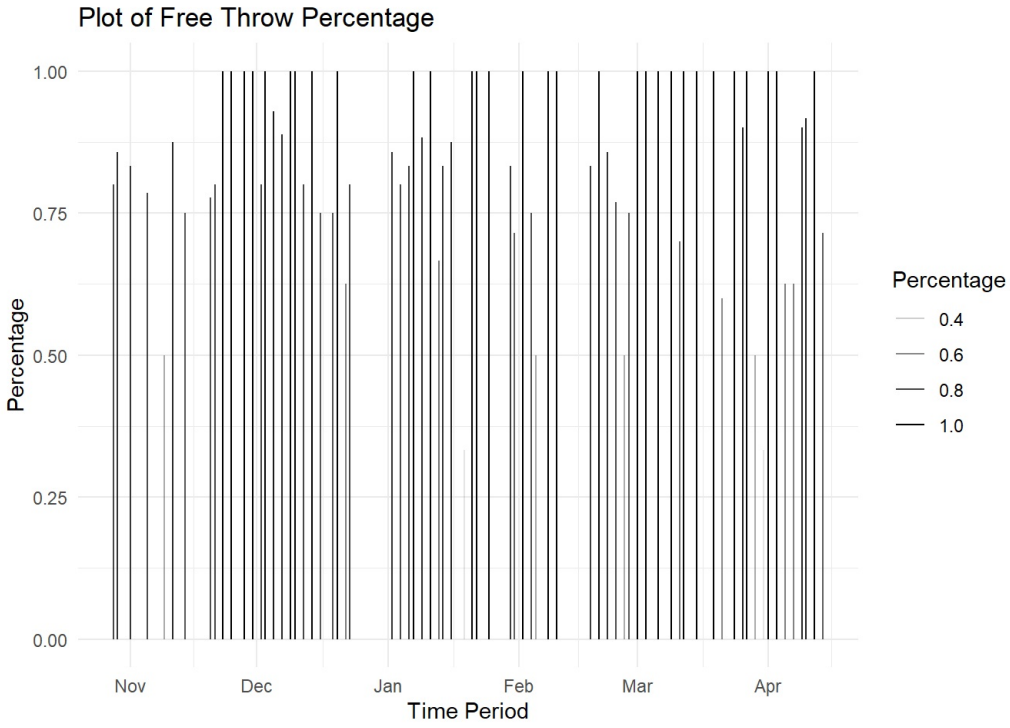


5. 11 Marks

- Subset the original `kobe` data set (not the `kobe.shots` data set) by considering plays that are only free throws (i.e., where `etype` is equal to `free throw`). Call this new data set `kobe.free`.
- Within the `kobe.free` data set, calculate the total number of points from free throws per game as well as the free throw percentage per game. You will want to use the `group_by`, `summarise`, `sum`, and `n` functions.
- Plot Kobe Bryant's free throw percentage per game using `geom_segment` to create vertical line segments from 0 to the free throw percentage. Your x-axis should be `date` and your y-axis should be free throw percentage.
- Add transparency proportional to the total number of points per game. (i.e., a larger number of points should have darker line segments).

Hide

```
#subset data set for free throws
kobe.free <- kobe %>% filter(etype == "free throw")

#calculate total points per game, and free throw percentage
kobe.free <- kobe.free %>% group_by(date) %>%
  summarise(Total_Score = sum(points),
            Number_Throws = n())

kobe.free <- kobe.free %>% mutate(Percentage = Total_Score/Number_Throws)

#create plot of date and free throw percentage
ggplot(data = kobe.free, mapping = aes(x = date, y = 0)) +
  geom_segment(mapping = aes(xend = date, yend = Percentage, alpha = Percentage)) +
  xlab("Time Period") +
  ylab("Percentage") +
  ggtitle("Plot of Free Throw Percentage") +
  theme_minimal()
```

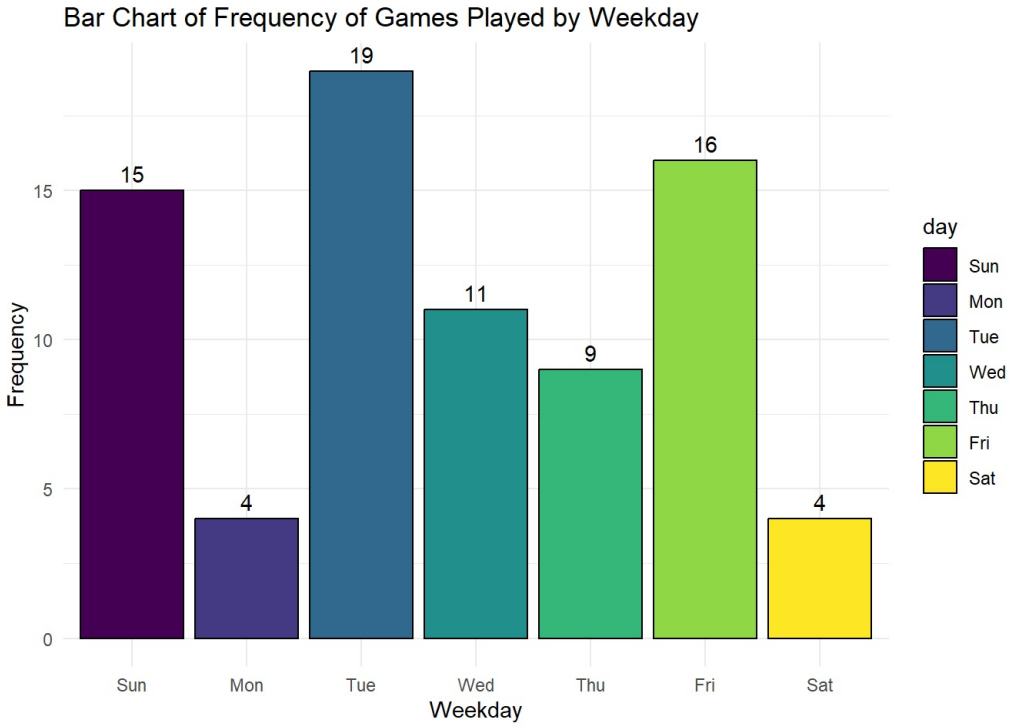


6. 7 Marks

- Using the `kobe` data set, find the unique dates that Kobe Bryant played in the 2008-2009 regular season. Hint: You will want to use the `group_by`, `summarise`, and `n_distinct` functions. You should end up with a data set with 78 rows. Name this data set `kobe.week`.
- Then create a variable that tells you the day of the week the game was played. You will need an appropriate `lubridate` function.
- Plot a bar chart that shows the frequency of games played on each of the seven days of the week.
- Comment on the most common and least common game days.

Hide

```
#Subset unique dates from 2008-2009
kobe.week <- kobe %>% group_by(date) %>%
  summarise(games_played = n_distinct(date))

#Create day of week variable
kobe.week <- kobe.week %>% mutate(day = wday(kobe.week$date, label = TRUE))

#Create bar chart
ggplot(data = kobe.week, mapping = aes(x=day, fill = day)) +
  geom_bar(col = "black") +
  ggtitle("Bar Chart of Frequency of Games Played by Weekday") +
  xlab("Weekday") +
  ylab("Frequency") +
  geom_text(stat = "count", aes(label = after_stat(count)), vjust = -0.5) +
  theme_minimal()
```



In observing our bar chart of the frequency of games played by Kobe Bryant during the 2008-2009 seasons, it appears that Tuesday had highest number of games which was 19 games. In constrast, Monday and Saturday only had 4 games which were the least number of games.

> Total possible marks for **Problem 1**: 45 Marks

Problem 2: Monthly average temperatures in Auckland

The data set `auckland_temps.csv` contains the monthly average temperatures in Auckland from July 1994 until January 2024. [Data source: https://cliflo.niwa.co.nz (https://cliflo.niwa.co.nz)]

1. 5 Marks

- Read in the data using `read_csv` (don't use `read.csv`).
- Convert the `Month` variable into the correct date format. Hint: You will need to use a function from the `tsibble` package.
- Coerce your `tibble` to a `tsibble` object with `Month` as the index.

Hide

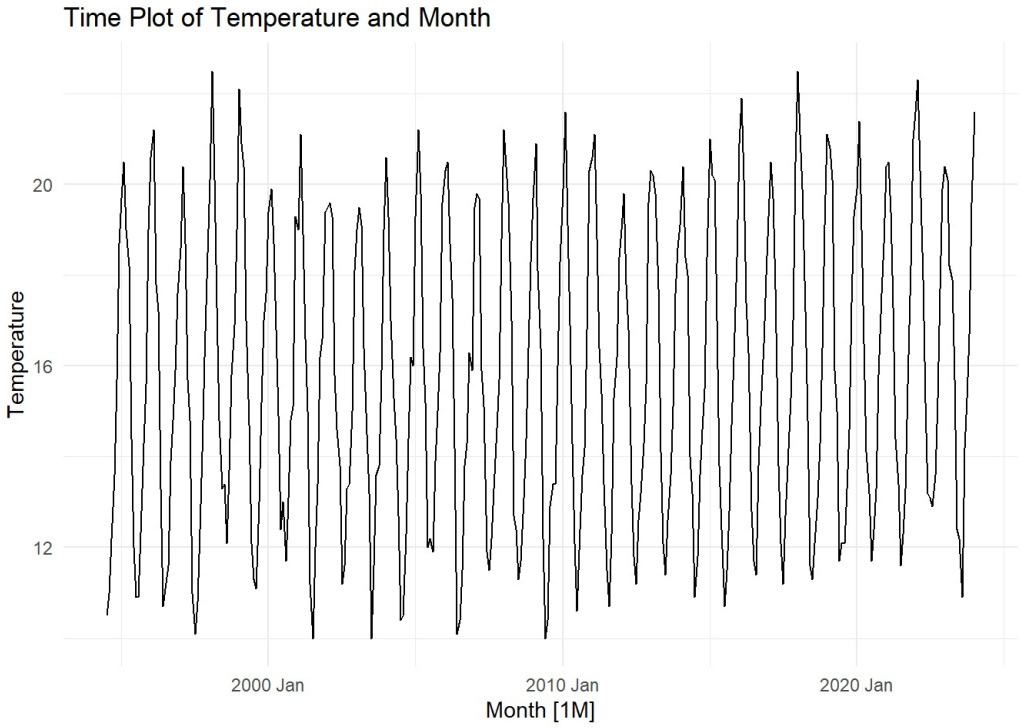
```
#read csv file
auckland_temps <- read_csv("auckland_temps.csv", show_col_types = FALSE)

#convert 'Month' string to date objects
auckland_temps$Month <- yearmonth(auckland_temps$Month)

#convert to tsibble object
auckland_tsibbles <- as_tsibble(auckland_temps, index = Month)
```

2. 7 Marks

- Create a time plot, seasonal plot, and subseries plot of the data.
- Comment on the seasonality in the plots. Which month has the highest average temperatures, and which month has the lowest?
- Comment on whether there is a trend in the data and if so, in what direction.

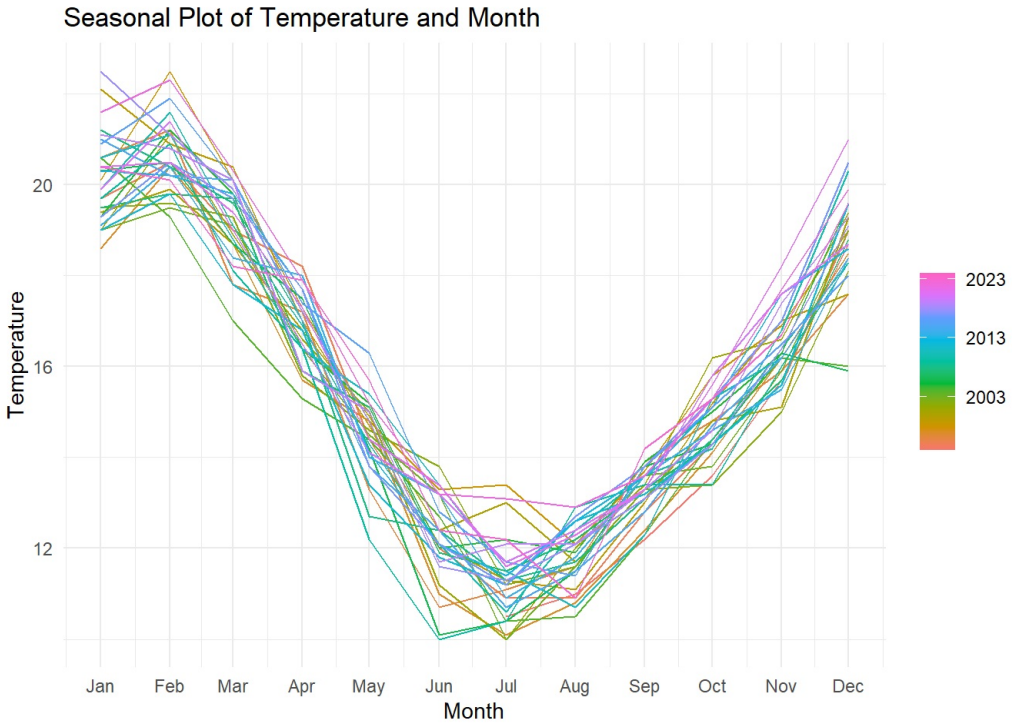
Hide

```
#time plot
auckland_tsibbles %>% autoplot(Temperature) +
  ggtitle("Time Plot of Temperature and Month") +
  theme_minimal()
```



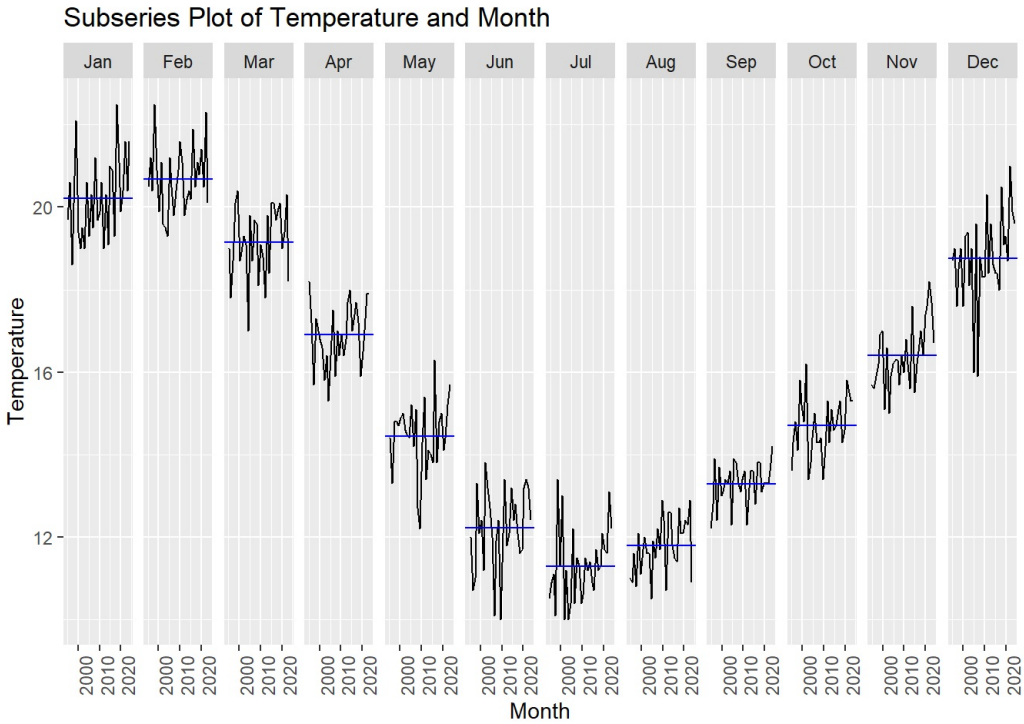
Hide

```
#seasonal plot
auckland_tsibbles %>% gg_season(Temperature) +
  ggtitle("Seasonal Plot of Temperature and Month")+
  theme_minimal()
```



Hide

```
#subseries plot
auckland_tsibbles %>% gg_subseries(Temperature) +
  ggtitle("Subseries Plot of Temperature and Month")
```



In our seasonal plot of temperature and month, it appears that the temperature peaks around January and February, and dips around June to August. This shows an annual cycle with the warmer temperatures typically observed around the start of the year and colder temperatures in the middle of the year.

In our subseries plot of temperature and months, it appears that February had the highest average temperature, while July had the lowest average temperature.

As we observe our time plot of temperature and month, there doesn't appear to be a trend and the average temperature seems to be consistent. However, when observing our seasonal plot, it appears that the temperatures of every month from the most recent years (as indicated by purple and pink lines) are higher than the temperatures from later years (as indicated by green and orange lines).

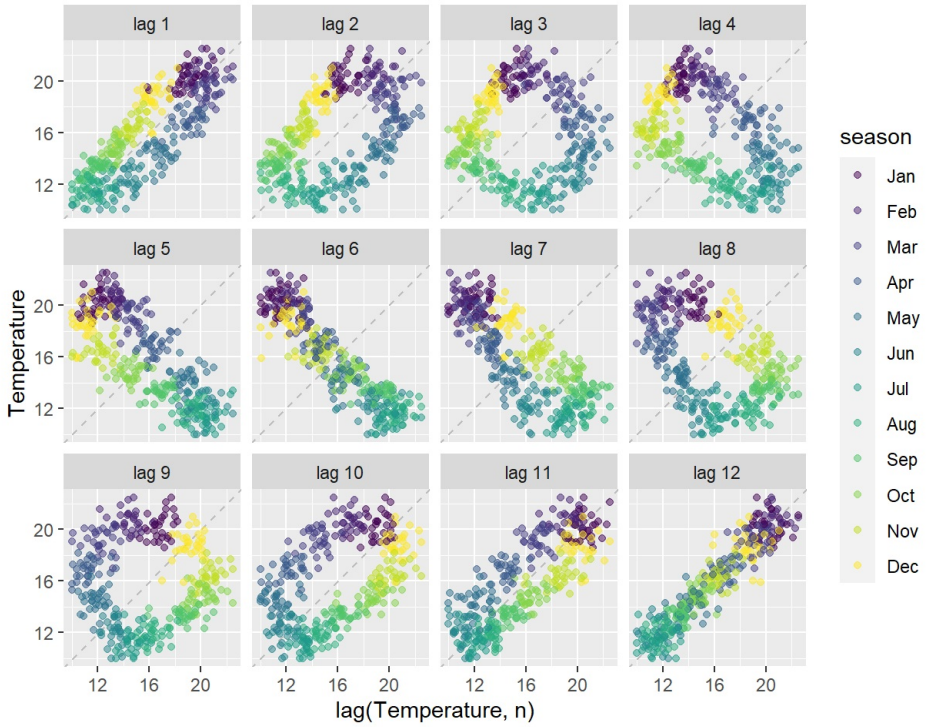
In addition, from our subseries plot, it appears that the temperatures of January, July, August, November and December appears be be increasing over time. These observations indicates a trend in our data where the temperature appears to be increasing over the years.

3. 8 Marks

- Create a lag plot for the first 12 lags. Use the `point` geometry and set `alpha = 0.5`.
- Write a sentence or two explaining what autocorrelation is.
- Comment on the patterns you observe in the lag plot, explaining why we see these specific autocorrelation patterns for lags 1, 6, and 12.

Hide

```
auckland_tsibbles %>% gg_lag(Temperature, geom = "point", alpha = 0.5, lag = 1:12)
```



Autocorrelation explores how the current value of a variable is correlated with its past values at different time lags. It measures how the past value influences the present value, which is important in helping identify trends and patterns within the data-set.

In lag 1, a positive correlation is observed between the lagged values and present values because the temperatures are similar for consecutive months. For example in lag 1 where the lagged values are shifted by 1 month, the temperature of January are paired with February, the similarity of their value creates a positive correlation.

In contrast for lag 6, where the lagged temperature values are shifted by 6 months, there is a negative correlation. This is expected because the temperature of January, occurring in summer, is plotted with the temperature of July, which is in winter. The temperatures in opposite seasons should result in a negative relationship.

In lag 12, a more linear and positive correlation is observed between the lagged and present values. This is because the lagged temperature values are shifted by a complete cycle of 12 months, which means the temperature of January is paired with the temperature of January from the following year. This is expected to be positive correlation and more linear.

Total possible marks for **Problem 2**: 20 Marks for 326 30 Marks for 786

Total possible marks for **Assignment 1**: 65 Marks for 326 75 Marks for 786