FDS Course Work 1

Solution Template

2023-10-31

Instructions

This is the RMD file you will edit and submit for your coursework. Please save the edited file with the same name as this one but with your user name (i.e. university log in name, not student number) added to the beginning, e.g. "pk255_MA3419_CW1.Rmd" You will need to create a PDF file labelled in the same way: e.g. "pk255_MA3419_CW1.pdf".

Complete the course work by adding code where needed and typing text between the horizontal lines where indicated. You should not need to add any new code chunks. Many code chunks contain a variable called `default`answer' (which is NA). This is what you will need to replace with your own code.

Marks indicated for each section are out of a total of 80. There are 5 marks for submitting code that runs and knits and 5 for submitting a two files with the correct names and format. To show that your code runs, you should include some (appropriate) output below each code chunk. This should clearly show (when viewed in the PDF) that the code has produced the required result.

There are also 5 marks for the overall standard of your code i.e. is it laid out neatly and readable; have you been consistent in your your use of tidyverse functions etc.

There is an additional file on Blackboard called C1_InfoAndHints which gives more advice on what marks are awarded for. Please check this regularly because I may update it with the answers to any significant questions I get.

This work comprises 30% of the total assessment for the module.

It is very important that you work on your own for this assessment. No collaboration is allowed and anyone found doing so will be subject to the penalties in the University Regulations (they are very severe). Anti-plagiarism software will be used for this submission.

If you get **completely** stuck contact me and I will give you some guidance - if that will result in the loss of any marks I will tell you and give you the choice to continue on your own.

You must email me your RMD file and submit your PDF file on Turnitin by 16:00 on Wednesday 15 November. (Note, this is an extension of two days on the deadline originally given.)

Load Data October expenditure data for Leicester City Council: https://data.leicester.gov.uk/explore/dataset/expenditure-exceeding-gbp500-2023/export/?disjunctive.department&disjunctive.purpose_of_expenditure&disjunctive.merchant_category

```r
Leics2023 <- read_excel(here("Data", "Leics-exceeding-gbp500-2023.xlsx"))
Leics2022 <- read_excel(here("Data", "Leics-exceeding-ps500-2022.xlsx"))
Leics2021 <- read_excel(here("Data", "Leics-exceeding-ps500-2021.xlsx"))
Leics2020 <- read_excel(here("Data", "Leics-exceeding-ps500-2020.xlsx"))
```

Join the four data frames with 'bind_rows()'. [2 marks]

```r
Leics <- bind_rows(Leics2023, Leics2022, Leics2021, Leics2020)

# Question 1: How many rows are there in the combined data frame? [1 mark]
answer <- nrow(Leics)

glue("Answer 1, number of rows: {answer}")
```

```
## Answer 1, number of rows: 283452
```

Clean the column names with the 'janitor' package (use big_camel case) and print the names. [3 marks]

```r
Leics <- Leics %>%

 clean_names(case = "big_camel")

 names(Leics)
```

```
## [1] "PaymentDate"          "Department"           "Beneficiary"
## [4] "PurposeOfExpenditure" "Amount"               "Vat"
## [7] "MerchantCategory"
```

```r
# Question 2: How does the column name that was originally "Purpose of expenditure" appear
# after the names have been cleaned? [1 mark]

answer <- colnames(Leics)[4]

 glue("Answer 2: {answer}")
```

```
## Answer 2: PurposeOfExpenditure
```

Use mutate and the lubridate package to convert the type of PaymentDate to Date and create two new columns "Year" and "Month" containing the year and month number of each payment. [5 marks]

```r
library(lubridate)

Leics$PaymentDate <- ymd(Leics$PaymentDate)

Leics <- Leics %>%
 mutate(Year = year(PaymentDate), Month = month(PaymentDate))
```

Print the class of the PaymentDate column and the maximum and minimum values. [2 marks]

```r
class(Leics$PaymentDate)
```

```
## [1] "Date"
```

```r
min(Leics$PaymentDate)
```

```
## [1] "2020-01-03"
```

```r
max(Leics$PaymentDate)
```

```
## [1] "2023-09-29"
```

Use the count function to list the top 10 beneficiaries (by number of payments). Save this in a data frame called beneficiaries. [2 marks]

```r
beneficiaries <- Leics %>%
 count(Beneficiary, sort = TRUE) %>%
 top_n(10)
```

```
## Selecting by n
```

```r
# Question 3: What is the name of the beneficiary with the eighth highest
# number of payments. [1 mark]

eighth_highest_name_of_beneficiary <- Leics %>%
 count(Beneficiary, sort = TRUE) %>%
 slice(8)

answer <- eighth_highest_name_of_beneficiary$Beneficiary

glue("Answer 3: eighth beneficiary by number of payments: {answer}")
```

```
## Answer 3: eighth beneficiary by number of payments: Help At Home
```

Use a regular expression to filter the Leics data frame so that it only contains row where the PurposeOfExpenditure includes the word "care", irrespective of case. (Not "carer." Create a new data frame called Leics_care for this data.) [5 marks]

```r
library(stringr)

Leics_care <- Leics %>%
 filter(str_detect(PurposeOfExpenditure, "\\b(?i)care\\b"))

#I assumed that care in "Healthcare" is not separate word and so, I eliminated it.

# Question 4: How many rows are there in this data frame? [1 mark]
answer <- nrow(Leics_care)

 glue("Answer 4, number of rows: {answer}")
```

```
## Answer 4, number of rows: 51881
```

Group the data by Beneficiary and calculate the total amount paid to each Beneficiary. Call this data frame top_beneficiaries and print the first 5 rows of the sorted dataframe. [5 marks]

```r
top_beneficiaries <- Leics %>%
 group_by(Beneficiary) %>%
 summarise(Total_amount_paid = sum(Amount)) %>%
 arrange(desc(Total_amount_paid))

print(head(top_beneficiaries, 5))
```

```
## # A tibble: 5 x 2
##   Beneficiary                          Total_amount_paid
##   <chr>                                            <dbl>
## 1 Biffa Waste Services Ltd                     72996895.
## 2 Leicestershire County Council                43244087.
## 3 Leicestershire Partnership NHS Trust         35077201.
## 4 Leicester BSF Company 2 Ltd                  28979116.
## 5 Total Gas & Power Ltd                        28436335.
```

```
# Question 5: Who is the top beneficiary, by total amount? [1 mark]
top_beneficiary <- top_beneficiaries %>%
 slice(1)

answer <- top_beneficiary$Beneficiary


glue("Answer 5, top beneficiary: {answer}")
```

```
## Answer 5, top beneficiary: Biffa Waste Services Ltd
```

Group by year and month (in that order) and calculate the sum of all payments in each month for each year. Create a new data frame called LeicsYrTotals for this grouped data. [*The total should be in a column called MonthTotal and you should include the parameter .groups = "drop_last" in your summarise function.*] [3 marks]

```
LeicsYrTotals <- Leics %>%
 group_by(Year, Month) %>%
 summarise(MonthTotal = sum(Amount), .groups = "drop_last")

print(LeicsYrTotals)
```

```
## # A tibble: 45 x 3
## # Groups:   Year [4]
##     Year Month MonthTotal
##    <dbl> <dbl>      <dbl>
##  1  2020     1  23147226.
##  2  2020     2  21697258.
##  3  2020     3  37354648.
##  4  2020     4  34703055.
##  5  2020     5  21395680.
##  6  2020     6  33819018
##  7  2020     7  50498317.
##  8  2020     8  28120108.
##  9  2020     9  35178602.
## 10  2020    10  31927517.
## # i 35 more rows
```

LeicsYrTotals is still grouped by Year. Apply summarise() again to calculate the number of months in each year and the total payment in each year. [*Include the parameter .groups = "drop".*] [2 marks]

```
LeicsYrTotals1 <- LeicsYrTotals %>%
  summarise(NumMonths = n(), YearTotal = sum(MonthTotal), .groups = "drop")

print(LeicsYrTotals1)
```

```
## # A tibble: 4 x 3
##    Year NumMonths  YearTotal
##   <dbl>     <int>      <dbl>
## 1  2020        12 392976423.
## 2  2021        12 424426000.
## 3  2022        12 498950477.
## 4  2023         9 358902313.
```

Create a column "AnnualisedTotal" that adjusts the year total to account for the number of months of data. Call your new data frame LeicsAnnTotals. [2 marks]

```
LeicsAnnTotals <- LeicsYrTotals1 %>%
 mutate(AnnualisedTotal = YearTotal / NumMonths * 12)

print(LeicsAnnTotals)
```

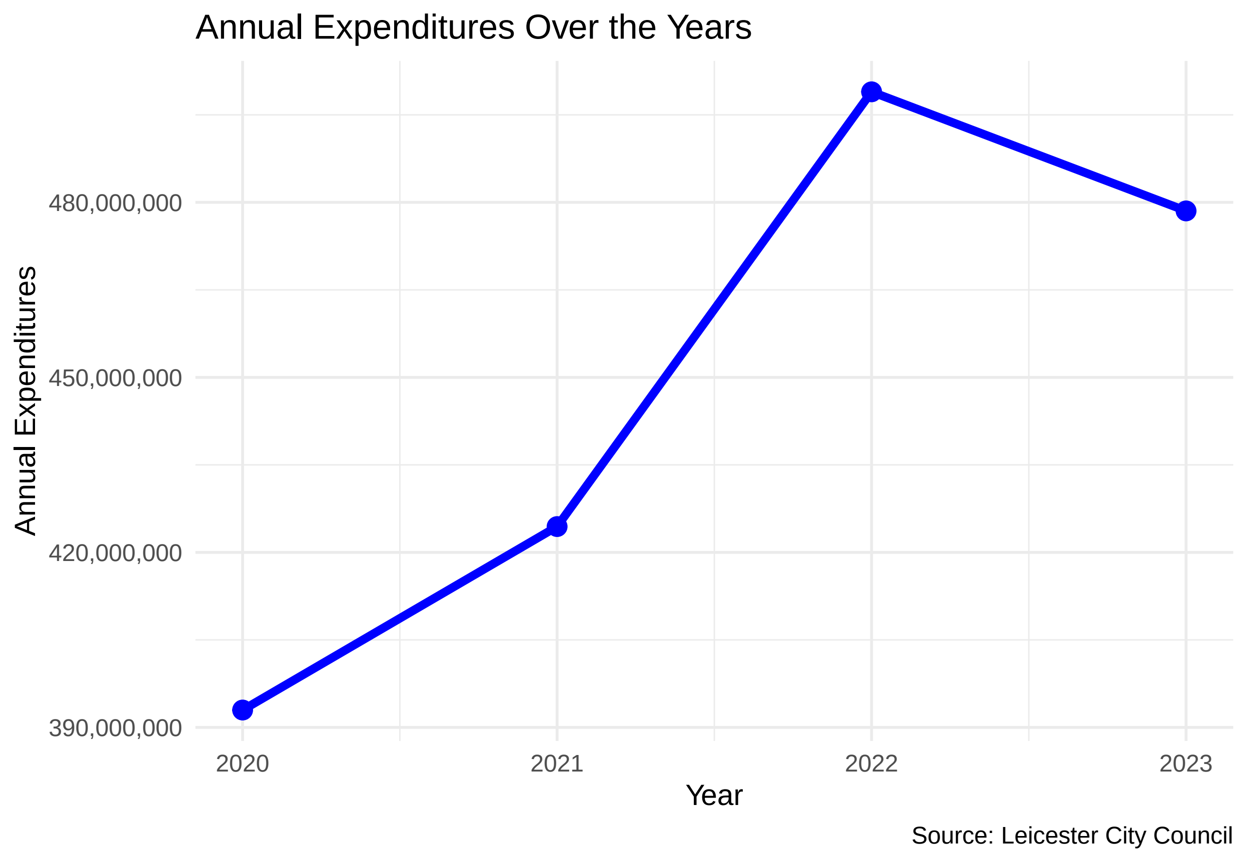
```
## # A tibble: 4 x 4
##    Year NumMonths  YearTotal AnnualisedTotal
##   <dbl>     <int>      <dbl>           <dbl>
## 1  2020        12 392976423.      392976423.
## 2  2021        12 424426000.      424426000.
## 3  2022        12 498950477.      498950477.
## 4  2023         9 358902313.      478536418.
```

Plot a line chart showing the annualised expenditure for each year. [6 marks]

```
plot_1 <- ggplot(LeicsAnnTotals, aes(x = Year, y = AnnualisedTotal)) +
  geom_line(color = "Blue", size = 1.5) +
  geom_point(color = "Blue", size = 3, show.legend = FALSE) +
  theme_minimal() +

  # I have used ChatGpt in this part for developing my code and for improving my
  # graph's visualization.For example, I asked from ChatGpt how I can add title,
  # write source of the data and give names to axis and etc. After ChatGpt updated
  # this code I analyzed the final code for fully understanding its meaning.

  labs(
    title = "Annual Expenditures Over the Years",
    x = "Year",
    y = "Annual Expenditures",
    caption = "Source: Leicester City Council"
  ) +
  scale_y_continuous(
    labels = scales::comma_format()
  )
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
print(plot_1)
```



Comment on your plot of annualised expenditure. [6 marks]

This graph shows annual expenditures of Leicester City Council from 2020 until 2023. We can see from the graph that annual expenditure increases sharply from roughly 390 mln. to 498 mln between 2020 and 2022. But, in 2023, we see slight decrease in this trend. But I think it may not be downward trend if we would have real expenses of last quarter of 2023. Because we do not have real expenditure amounts of that period, we just predicted last 3 months of 2023 by finding average monthly expenditure amount and then applied it proportionally for the month of 10, 11 and 12 of 2023. If we look through previous years last quarter expenditures they are generally are higher than previous quarters. Generally, organizations which their budgets funded by governments try to spend cautiously at the first months of the year and spend more money in the last months of the budget years. So, if we would have real data of last quarter expenditures of 2023, we can observe higher annual expenditures in 2023 than all previous years. We can dive and prognoses the last quarter of 2023 expenditures by comparing all previous years last quarter expenditures.

Use just the three years for which you have 12 months of data. Group by month and calculate the sum of all payments in each month. Call this LeicsMonthTotals and print it. [2 marks]

```r
LeicsMonthTotals <- Leics %>%
  filter(Year < 2023) %>%
  group_by(Month) %>%
  summarise(Monthly_payments = sum(Amount))

print(LeicsMonthTotals)
```

```
## # A tibble: 12 x 2
##    Month Monthly_payments
##    <dbl>            <dbl>
##  1     1        85716463.
##  2     2       104536948.
##  3     3       131564547.
##  4     4       117173871.
##  5     5        87503735.
##  6     6        96369027.
##  7     7       133561167.
##  8     8        96334916.
##  9     9       102723200.
## 10    10       101891303.
## 11    11       123999170.
## 12    12       134978552.
```

Calculate the average for each month (across all three years). Call this LeicsMonthMeans and print it.[1 mark]

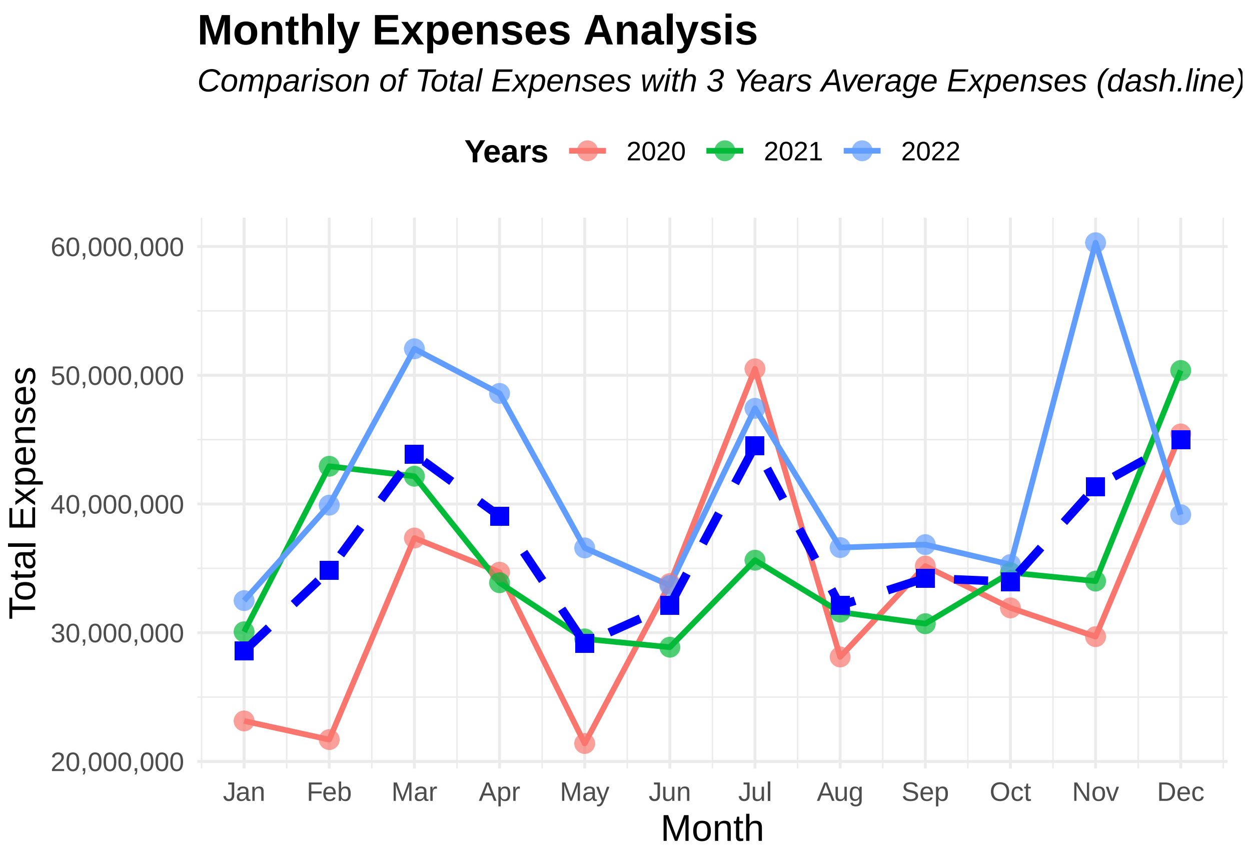
```
LeicsMonthMeans <- LeicsMonthTotals %>%
  mutate(Average_Monthly_payments = Monthly_payments / 3)

print(LeicsMonthMeans)
```

```
## # A tibble: 12 x 3
##    Month Monthly_payments Average_Monthly_payments
##    <dbl>            <dbl>                    <dbl>
##  1     1        85716463.                28572154.
##  2     2       104536948.                34845649.
##  3     3       131564547.                43854849.
##  4     4       117173871.                39057957.
##  5     5        87503735.                29167912.
##  6     6        96369027.                32123009.
##  7     7       133561167.                44520389.
##  8     8        96334916.                32111639.
##  9     9       102723200.                34241067.
## 10    10       101891303.                33963768.
## 11    11       123999170.                41333057.
## 12    12       134978552.                44992851.
```

Plot a line chart showing the 12 monthly totals as a separate line for each of the three years and as a separate line for the average amount for each month.[8 marks]

```
# I have used ChatGpt in this code also for developing my code and for improving
# my graph's visualization. For example, I asked from ChatGpt how I can add title,
# write source of the data and give names to axis and etc. After ChatGpt updated
# this code I analyzed the final code for fully understanding its meaning.

Leics1 <- LeicsYrTotals %>%
filter(Year < 2023)

plot_2 <- ggplot() +

  # Firstly I added Line and point geom for Leics1 dataset
  geom_line(
```

```r
    data = Leics1,
    aes(x = Month, y = MonthTotal, color = as.factor(Year), group = Year),
    linetype = "solid", size = 1
  ) +
  geom_point(
    data = Leics1,
    aes(x = Month, y = MonthTotal, color = as.factor(Year)),
    size = 3, alpha = 0.7
  ) +

  # Then in order to show the 4th line (Average_Monthly_payments) in same graph I added
  # Line and point geom for LeicsMonthMeans data set
  geom_line(
   data = LeicsMonthMeans,
   aes(x = Month, y = Average_Monthly_payments, color = "3 Years Average Payments"),
    linetype = "dashed", color = "blue", size = 1.5
  ) +
  geom_point(
    data = LeicsMonthMeans,
    aes(x = Month, y = Average_Monthly_payments, color = "3 Years Average Payments"),
    color = "blue", size = 3, shape = 15
  ) +

  # I made some modification for increasing readability of x and y axis. In x axis,
  # I changed numbers (1 to 12) to months and in y axis I extended figures to millions

  scale_x_continuous(breaks = 1:12, labels = month.abb) +
  scale_y_continuous(labels = scales::comma_format()) +

  # Theme and other adjustments (I added some titles and subtitles to the graph)
  labs(
    title = "Monthly Expenses Analysis",
    subtitle = "Comparison of Total Expenses with 3 Years Average Expenses (dash.line)",
    x = "Month",
    y = "Total Expenses",
    color = "Years"
  ) +
  theme_minimal() +
  theme(
    legend.position = "top",
    plot.title = element_text(size = 16, face = "bold"),
    plot.subtitle = element_text(size = 12, face = "italic"),
    axis.title = element_text(size = 14),
    axis.text = element_text(size = 10),
    legend.title = element_text(size = 12, face = "bold"),
    legend.text = element_text(size = 10)
  ) +

  # Legend title
  guides(color = guide_legend(title = "Years"))

print(plot_2)
```



Comment on your chart of expenditure per month. [6 marks]

From the graph, we observe that Leicester City Council 2022 total expenses is higher than 3 years (2020, 2021 and 2022) average monthly expenses except the month of December. in January of 2022, the expenses were just over 30 mln. and this increased until March to over 50 mln. and then showed downward trend until June which decreased that year's minimum point of approximately 34 mln. After that time, this indicator showed some fluctuations until October and increased sharply and reached its and also all 3 years maximum of 60 mln and then plummet down roughly just under 40 mln and also first time in that year it decreased under average expenses of 3 years.
2021 monthly expenses show almost similar trend with 3 years average monthly expense trend. The expenses in 2021 were higher than average indicator from January of 2021 until March and December of that year. From March 2021, the monthly expenses start about 42 mln and were under average indicator until mid of December and exceeded the average after that time and reached its maximum point of 50 mln.
In 2020, the monthly expenses show volatility , but most time of that year the expense figures were under average (maybe because of Covid 19).The expenses were roughly 24 mln in January of 2020 which were minimum of all 3 years indicators. Then after slight decrease,this increased sharply to 37 mln and then decreased its minimum point of just over 21 mln and from that point it rocketed up and passed the average indicator first time in that year and reached its maximum point of 50 mln. and then it decreased significantly and again got under the average until the end of that year except the month of September.
