Diabetes hospital admission

52103889

25/11/2021

Introduction

Open science Open science is the movement to make scientific research including data, physical samples, software, publications and its dissemination accessible to all levels of society. It promotes transparency, robustness and trust of research process and its conclusion. It also provides collaborative platform for sharing of scientific knowledge and ease of access of research materials for both amateur and professionals.

Open science should be encouraged as it provides following benefits to the researchers and to the society as a whole [1]

Efficiency of the research – Open science can dramatically improve effectiveness and productivity of the scientific research by reducing the duplication and cost of creating, transferring, and reusing data. It allows reusability of data for more research work and hence create new opportunity for global participation in the research process.

Quality and integrity of the research – Openness of research assets like data, code and results provides an opportunity for evolution and scrutiny by scientific community which allows accurate replication and validation of the research result. Openness can also identify malpractice of science early on and will contribute to maintain science's self-correction principle.

Economic benefits – Open science will foster innovation and scientific systems more broadly with increased awareness and conscious choices and hence benefit advanced economies and developing countries. Innovation and knowledge transfer – Open science offers swift path from research to innovation to produce new products and services.

Public disclosure and agreement – Open science involves public of the society which promotes awareness and trust of research results hance results in more public funded research.

Global benefits – Open science takes account of global issue like climate change or the ageing population and promotes collaborate effort and faster knowledge transfer

In simple terms open science can amplify the power of science by adopting reproducibility factor of scientific research like keeping research data, code, and environment information open and accessible to all levels of society.

R is open access easy to learn programming language. Along with its statistical utilities and libraries R also makes machine learning algorithms easy to learn.

Pipeline toolkit in R provides concrete evidence that the results are re-creatable from the starting material, and data analysis project does not need to rerun from scratch. [2]

R has various tools that ensures specific package version can be required for analysis and hence package reproducibility can be achieved. [2]

Successfully completing a data analysis project often requires much more than statistics and visualizations. Efficiently managing the code, data, and results as the project matures helps reduce stress and errors. The "workflow" packages assist the R programmer by managing project infrastructure and/or facilitating a reproducible workflow. [2]

It can be demonstrated with a simplistic exploratory data analysis in R markdown based on fake data set as an example. We have considered two data sets; one has sugar readings of 100 people, and another has age and gender information of those 100 people. We are interested to see if there is a relation between age and their sugar readings for both male and female.

R markdown

The objective of reproducibility in research can be achieved by combining specific instruction to experimental data and data analysis in one place for better understanding, verification, and recreation. This can be achieved by using R markdown which allows use of document that is combination of content and data analysis code. The Sweave function and the knitr package can be used to blend the subject matter and R code in a single document that defines content and the analysis. [2]

R markdown is a file format within R which allows creation of reproducible and dynamic documents It consist of three types of content YAML header,text and code chunk. Combination of these trio make R favorite data analysis programming language for health data scientists. R markdown can produce output/report in word, HTML, pdf, dashboards and slideshows etc which can be easily shared.The output is defined in the YAML header.

YAML header is a short text which gives information of title, author, date and output demarcated by three dashes (—) on either end of the markdown file

R markdown provides syntax to create different levels of headers. For example, the word "introduction" of this report has been created as main heading by using single "#" before the text "introduction" and other levels of headings can be created by increasing the number of "#" in front of the text. For example

heading 2 with "##" in front of text or phrase

heading 3 with "###" in front of text or phrase

Plain text can be written in *italics* by using "*" or "_" and keeping the text in between them and in **bold** by using "**" or "__" before and after the text. There are syntax for writing text in different styles for example

superscripts ^{2} by using "^"

~~strikethrough~~ by using "~~"

Unordered lists ca be written as follows by using "*" in front of the text and "+" in front and indenting the sub-unordered lists

- Unordered list
- Item 2
  - sub-item 1
  - sub-item 2

Ordered list can be written as following by writing order like "1." and sub items using "1." in front and indenting the sub-ordered lists

1. ordered list
2. item 2
    1. sub-item
    2. sub-item

Tables can also be created by using "-" more that three to underline the table header and then use "|" to separate table content on each side.

Table Header	Second Header
Cell 1	Cell 2
Cell 3	Cell 4

To create a blockquote, add a "> "in front of a paragraph. For example

> This is blockquote

R markdown facilitate code chunk which is simply a display of code for illustration and its output in the final document . The code chunk in R markdown is used for detailed snippets of code and it is achieved by using three back ticks in start of the code and finish with 3 back ticks (“‘)

Code chunk output can be customised with arguments set within the curly bracket { } for example

include = FALSE prevents code and results from appearing in the finished file. R Markdown still runs the code in the chunk, and the results can be used by other chunks.

echo = FALSE prevents code, but not the results from appearing in the finished file. This is a useful way to embed figures.

message = FALSE prevents messages that are generated by code from appearing in the finished file.

warning = FALSE prevents warnings that are generated by code from appearing in the finished.

R markdown also facilitates inline code by putting back ticks (') around part of the code line. In line code is usually for small command within the test of the report.

Look at the below code chunk, all the libraries that is used in this exploratory data analysis has been loaded with message = "FALSE" which will prevent message in finished file. R facilitate writing very concise text beside each line of code if it is needed to explain what that line of code is performing by using "#"

```
library(tidyverse)
library(plotly)    # interactive visualisations
library(eeptools)  # calculating age from DOB
```

```
## Warning: package 'eeptools' was built under R version 4.1.2
```

```
library(viridis)   # nice colour scheme
library(here)      # importing data
library(ggplot2)   # great for combining plots
```

Data preparation

A meta data is a data that provides information about one or more aspect of data. Meta data includes documentation of the data set contents, context, quality, structure and accessibility. It usually summaries the data and hence help in understanding and working with specific data easier. It makes data more traceable, findable and reusable.

Good data quality starts with good metadata hence meta data forms an integral part of open and reproducible science. Making data and meta data available in open access repository along with the code that is used to do the data analysis will help in replicating the results effortlessly. This will also enhance robustness of code and trustworthiness of research results/outputs.

Within health care system where privacy of patient's personal information takes priority, sharing actual data in open access repository can not be possible all the times. A fake data can play a crucial role in this scenario. A fake data set is a data set that has the same format as the real data set but contains made-up data. Fake data with research code can be shared in any open access repository and checking the robustness of the code can be requested from intended audience.

It's not always feasible to share the data while working with huge data set and may require a second opinion or help with the code being used. Fake data similar to original big data along with code can be shared in some open platform/ repository and help can requested.

Working with fake data and analysing it and using the same code to analyse a data set that is kept in a very secure environment could make the process smother than working on secured data directly.

It is quite easy to put together a fake data set, we just need to follow our data dictionary

Similar to fake data, a tidy data is equally important for efficiency, reproducibility and collaboration. A tidy data is a way of organising your data, to enable easy manipulation and plotting in R. It has three simple rules

1. Each variable must have its own column.
2. Each observation must have its own row.
3. Each value must have its own cell

An untidy data set can be converted into tidy data set by pivoting it. There is a popular saying in Data Science like this "Data Scientists spend up to 80% of the time on data cleaning and 20% of their time on actual data analysis".

Having a tidy data with a standardised framework of how data should look like, we will have to spend less time on data cleaning and wrangling and more time on focusing analysis. Tidy data make data more reproducible and easily understood by others. Moreover, a tidy data is complemented with tools in r to work with. The dplyr package of R provides several functions like mutate (), filter (), group_by() and many more functions that help you complete the most common data manipulation.

When data has been cleaned and tidied up, the next important step is to read this data into R. Usually data is in table such as spreadsheet, database comma separated values, CSV in short, which is basically simplified spreadsheet. readr package of tidyverse of R provides a function read_csv () which will read the csv file kept in the working directory into R environment just by passing the name of csv. read_csv () is one of the quickest and efficient way to import a tibble of the data set. Here is the beauty of this function. The path of the file does not need to be hard coded within the read_csv (), In fact it is advised not to pass the file path through the function. This will help in reproducibility when someone else will try to read the data with your code.

Another package which is useful in reading data in R keeping reproducibility in mind is **here** package. The here package allows you to set the top level of your project folder as "here" and to specify where things live

relative to that location. Here package works from the project up and hence makes it easy to reference other sub folders in the directory. If here package is used, files outside of project will still work as it is supposed to work.

In the below code chunk two fake data set "demographic" and "hospital_admission" has been read into R using "here" package.

```
demographic <- read_csv(here("inputs/demographics.csv"))
Hospital_admission <- read_csv(here("inputs/Diabetes related hospital admission.csv"))
```

The dplyr package provides a function called glimpse (). This function displays a vertical preview of dataset which allows easy preview of data type and sample data. For example below glimpse() is giving a snapshot of data type of fake dataset that was loaded previously.

```
glimpse(demographic)
```

```
## Rows: 100
## Columns: 3
## $ `chi-number` <dbl> 4162785270, 1025086593, 6094223759, 1748543658, 953366253~
## $ sex          <chr> "male", "male", "female", "male", "male", "female", "fema~
## $ dob          <date> 2006-02-17, 1950-07-22, 1960-03-16, 1952-09-17, 1973-03-~
```

```
glimpse(Hospital_admission)
```

```
## Rows: 100
## Columns: 3
## $ chi    <dbl> 9988802102, 9956612897, 9906303504, 9877864738, 9733171929, 968~
## $ value  <dbl> 213, 235, 161, 224, 185, 71, 235, 135, 144, 135, 225, 168, 248,~
## $ result <chr> "positive", "positive", "negetive", "positive", "negetive", "ne~
```

checking missing values

Once data set has been read in R, next important thing is to have a good look at the data set, hunt for missing values and typos. R provides some very useful functions like summary () and table () whaich can be used to check missing values, typos and outliers.

```
demographic %>% summary()
```

```
##    chi-number           sex                dob
##  Min.   :1.025e+09   Length:100         Min.   :1950-07-22
##  1st Qu.:3.544e+09   Class :character   1st Qu.:1966-04-03
##  Median :6.177e+09   Mode  :character   Median :1979-07-06
##  Mean   :5.888e+09                      Mean   :1979-12-20
##  3rd Qu.:8.354e+09                      3rd Qu.:1994-08-07
##  Max.   :9.989e+09                      Max.   :2010-12-18
```

```
demographic$sex %>%table()   # any missing value in sex
```

```
## .
## female   male
##     48     52
```

For example, summary function on hospital_admission data set is clearly showing minimum value as 10. From the data dictionary we know the value column is for sugar reading in mg/dL and sugar reading as low as 10 mg/dL is unlikely. This can be interpreted as typos during data collection or an outlier

Another example is table () on "result" column of hospital_admission data set clearly highlights that there is a missing value "*" in the "result "column.

```
Hospital_admission$result %>%table()  # any missing value in results
```

```
## .
##        * negetive positive
##        1       60       39
```

```
Hospital_admission %>% summary()  # any typos in hospital admission data set
```

```
##       chi                value          result
##  Min.   :1.025e+09   Min.   : 10.0   Length:100
##  1st Qu.:3.544e+09   1st Qu.:133.5   Class :character
##  Median :6.177e+09   Median :178.0   Mode  :character
##  Mean   :5.888e+09   Mean   :175.3
##  3rd Qu.:8.354e+09   3rd Qu.:222.5
##  Max.   :9.989e+09   Max.   :355.0
```

Both missing value and typo can be dropped out of the data set by first filtering it out from data using the filter () and dropping it out from data set by drop_na ()

```
tidy_hospital_admission <- Hospital_admission %>%
  drop_na()%>%
  filter(result != '*')

tidy_hospital_admission$result %>% table()
```

```
## .
## negetive positive
##       60       39
```

```
tidy_hospital_admission <- Hospital_admission %>%
  drop_na()%>%
  filter(value != '10')

tidy_hospital_admission %>% summary()
```

```
##       chi                value        result
##  Min.   :1.200e+09   Min.   : 70   Length:99
##  1st Qu.:3.613e+09   1st Qu.:134   Class :character
##  Median :6.260e+09   Median :178   Mode  :character
##  Mean   :5.937e+09   Mean   :177
##  3rd Qu.:8.362e+09   3rd Qu.:223
##  Max.   :9.989e+09   Max.   :355
```

joining the two data set

Now both the data set "demographic" and hospital admission" is now clean and tidy data set. We need a dataset that has information from both these dataset. R provides join function which can easily join two data set having one column information in common in each dataset. There are different types of joins available within R which are as follows

inner join - Returns only the rows in which the left table have matching data in the right table.

outer join - Returns all rows from both tables

left join - Returns all rows from the left table, and any rows with matching data from the right table

right join - Returns all rows from the right table, and any rows with matching data from the left table.

For our data set we want all the data from hospital_admission data set and corresponding /matching data from demographic dataset. Left join will be appropriate to join the two fake data set A new dataset has been created called admission_data, which contains all the data from tidy_hospital_admission and corresponding matching data from demographic dataset. Column name " value " has been renamed as " sugar_reading" for better understanding of the data using rename ().

```
admission_data <- left_join(tidy_hospital_admission,demographic,by = c("chi" = "chi-number"))

admission_data <- admission_data %>%
  rename(Sugar_reading = value)

glimpse(admission_data)
```

```
## Rows: 99
## Columns: 5
## $ chi           <dbl> 9988802102, 9956612897, 9906303504, 9877864738, 97331719~
## $ Sugar_reading <dbl> 213, 235, 161, 224, 185, 71, 235, 135, 144, 135, 225, 16~
## $ result        <chr> "positive", "positive", "negetive", "positive", "negetiv~
## $ sex           <chr> "male", "female", "male", "male", "male", "female", "fem~
## $ dob           <date> 1989-09-18, 1969-12-30, 1988-08-11, 1994-05-28, 1986-04~
```

save the clean data

Once the data set is joint, it is good idea to save the joint dataset somewhere safe, so that it can be referred in future, if needed. Write_csv() comes in handy which writes the clean, joint dataset in csv format. Our joint data set has been saved with the name "sugar reading" in csv format.

```
write_csv(admission_data, "Sugar reading.csv")
```

Data visualisation

R provides a plotting package ggplot which has helpful commands to create complex visually informative plots from data in the dataset.

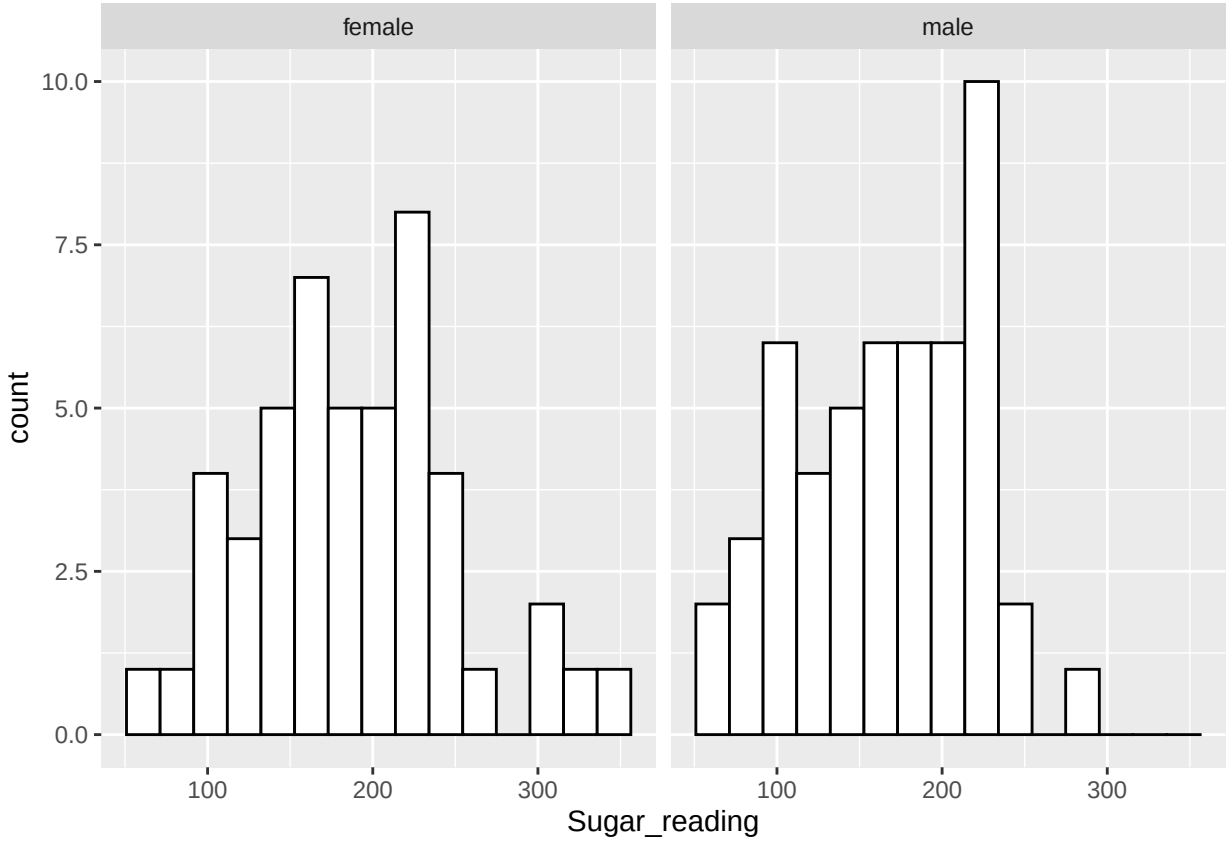
histogram

A histogram has been plotted by using the data from admission_data dataset. Passing "sugar_reading" through x axis and using geom_histogram () and using Facet_wrap () for creating two histogram one

for each sex. Plotted histogram shows distribution of sugar reading between males and females. Largest concentration of sugar reading in male and female both is between 220- 230 mg/dL .Also data is not normally distributed for both male and female sugar readings.

```
#histogram

admission_data %>%
    ggplot(aes(x= Sugar_reading))+
  geom_histogram( bins = 15,color = "black", fill = "white")+
  facet_wrap(~sex)
```



box plot of dataset

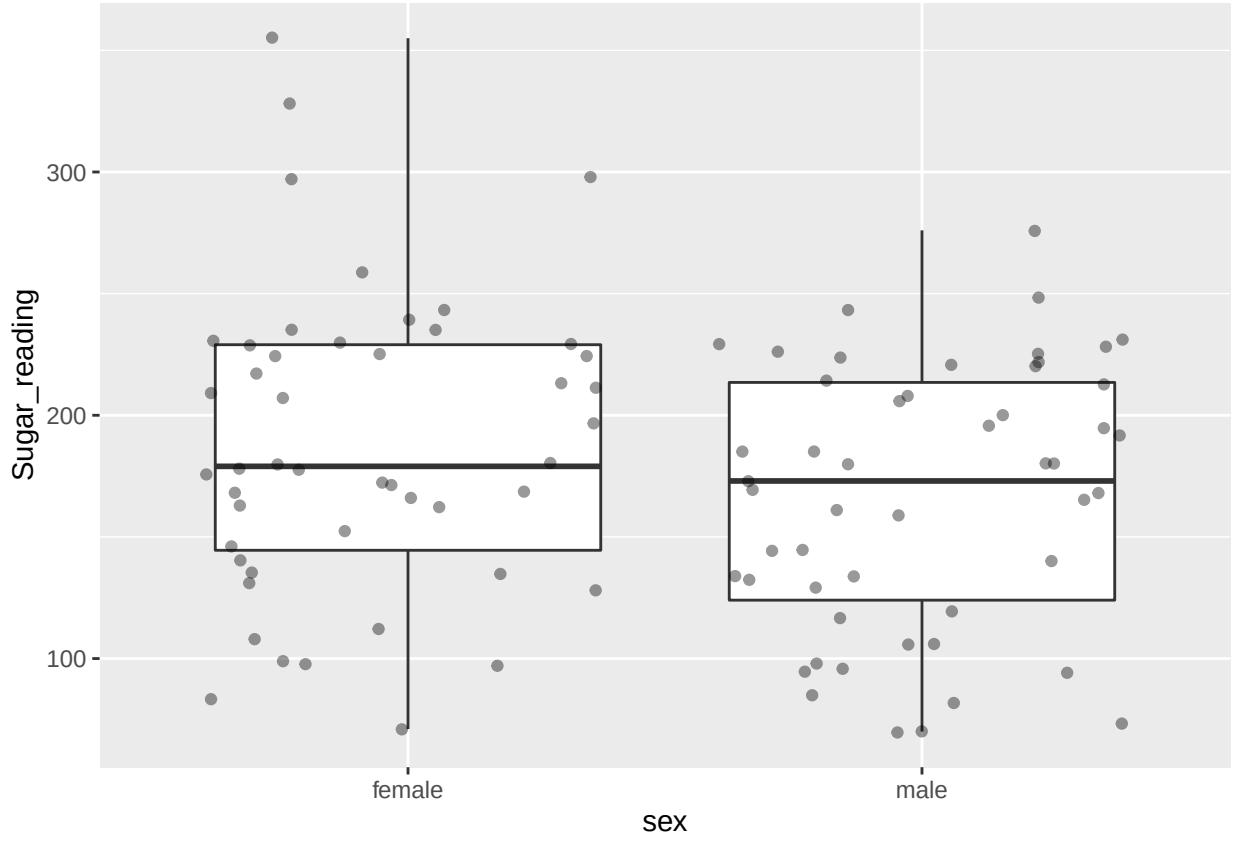
To see the median value of sugar reading between males and females a box plot has been plotted by passing sex through x axis and sugar reading in y axis and using the geom_boxplot (). Another function called geom_jitter() has been used to visualise individual sugar reading for both sex.

It is evident from the box plot that the median value of sugar reading for both male and female is same which around 175 mg/dL. The data is scattered for both males and females. The spread of interquartile range for sugar reading for both male and female is almost equal but the minimum and maximum value of sugar reading for female is little bit higher than male sugar readings

```
# boxplot

admission_data %>%
```

```
ggplot(aes(x=sex,
           y=Sugar_reading))+
geom_boxplot()+
geom_jitter(alpha = 0.4)+                    # add data points
theme(legend.position = "none")              # remove legend
```



calculating age from dob

A new variable has been calculated called “age” from the column “dob”

```
#admission_data$dob = as.Date(admission_data$dob)
admission_data$age = round(age_calc(admission_data$dob, enddate = Sys.Date(), units = 'years' ))
glimpse(admission_data)
```

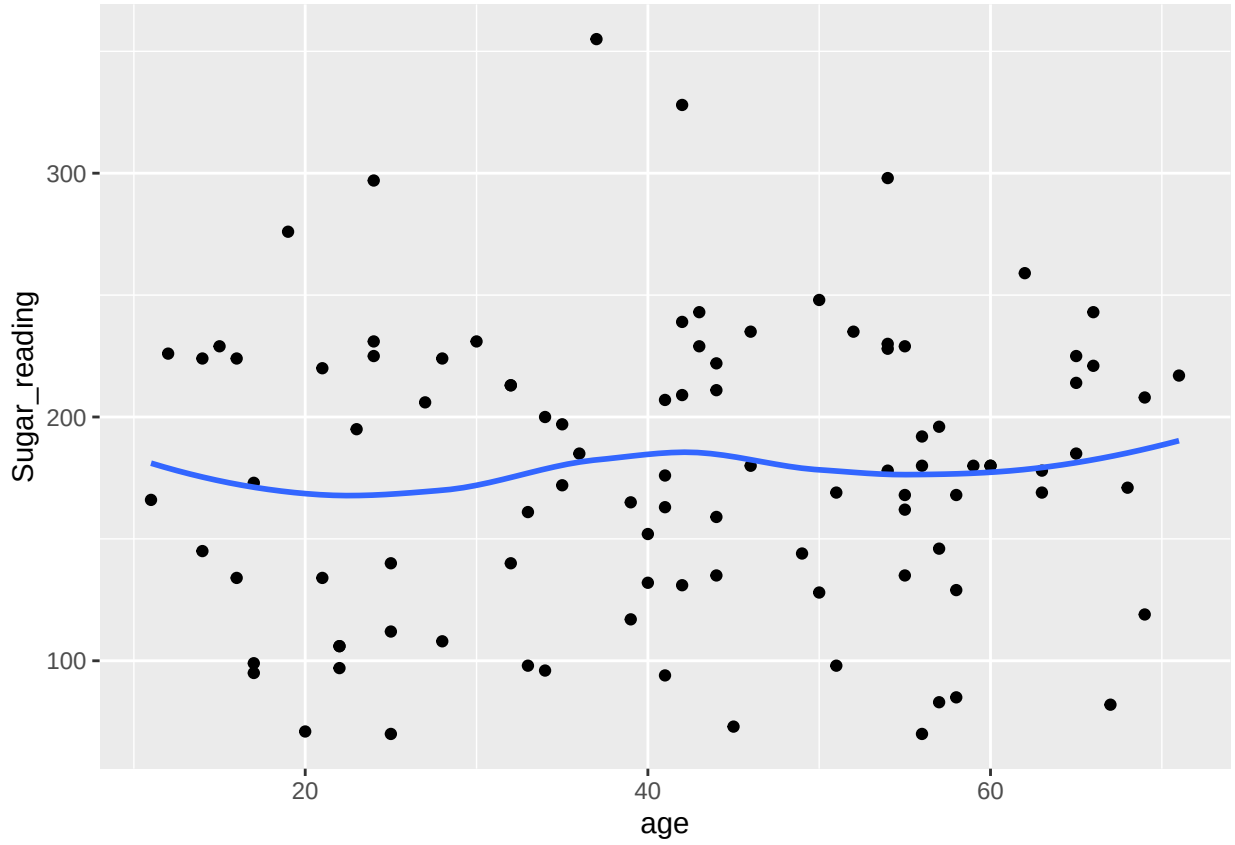
```
## Rows: 99
## Columns: 6
## $ chi           <dbl> 9988802102, 9956612897, 9906303504, 9877864738, 97331719~
## $ Sugar_reading <dbl> 213, 235, 161, 224, 185, 71, 235, 135, 144, 135, 225, 16~
## $ result        <chr> "positive", "positive", "negetive", "positive", "negetiv~
## $ sex           <chr> "male", "female", "male", "male", "male", "female", "fem~
## $ dob           <date> 1989-09-18, 1969-12-30, 1988-08-11, 1994-05-28, 1986-04~
## $ age           <dbl> 32, 52, 33, 28, 36, 20, 46, 44, 49, 55, 65, 55, 50, 32, ~
```

Scatterplot

To visualise relationship between age and sugar reading a scatter plot has been plotted using geom_point () and passing age in x axis and sugar reading in Y axis and drawing a line of best fit individually for male and female in different colours.

```
admission_data %>%
  ggplot(aes(x=age,
             y=Sugar_reading))+
  geom_point()+
  geom_smooth(se = FALSE)
```

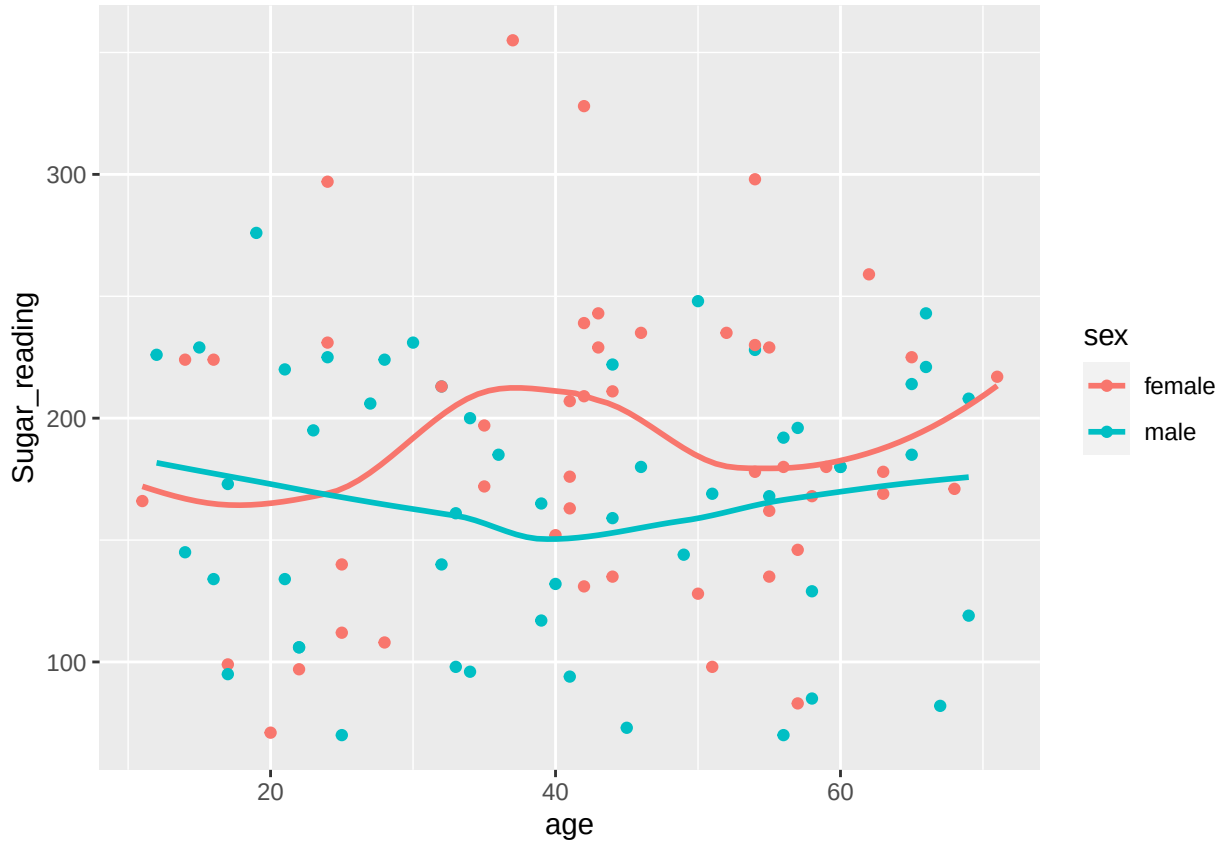
```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



line of best fit for males and females

```
admission_data %>%
  ggplot(aes(x=age,
             y=Sugar_reading,
             color =sex))+
  geom_point(alpha=2, size=1.5)+
  geom_smooth(se = FALSE)
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



A person is called diabetic if the sugar reading exceeds the reading 200 mg/dL.The results column says positive if the sugar reading is more than 200 mg/dL and negative otherwise. A new data set has been created with diabetic males and females called "diabetes". A similar scatter plot has been plotted by using geom_point () and passing age in x axis and sugar reading in y axis and drawing a line of best fit individually for males and females in different colours.

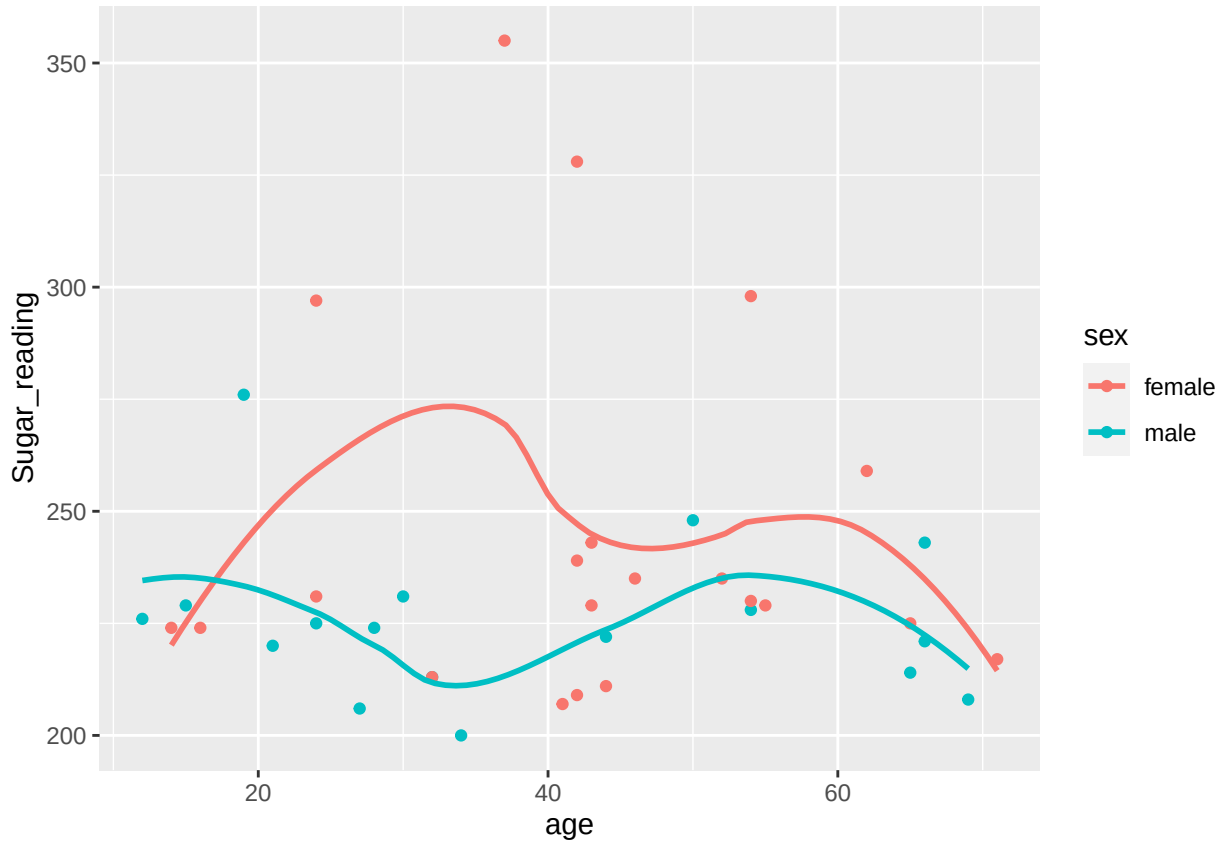
```
diabetes <- admission_data %>%
  filter(result == "positive")
glimpse(diabetes)
```

```
## Rows: 38
## Columns: 6
## $ chi           <dbl> 9988802102, 9956612897, 9877864738, 9634128581, 94410542~
## $ Sugar_reading <dbl> 213, 235, 224, 235, 225, 248, 243, 211, 221, 243, 355, 2~
## $ result        <chr> "positive", "positive", "positive", "positive", "positiv~
## $ sex           <chr> "male", "female", "male", "female", "female", "male", "f~
## $ dob           <date> 1989-09-18, 1969-12-30, 1994-05-28, 1976-03-19, 1957-04~
## $ age           <dbl> 32, 52, 28, 46, 65, 50, 43, 44, 66, 66, 37, 54, 19, 24, ~
```

```
diabetes %>%
  ggplot(aes(x=age,
             y=Sugar_reading,
             color =sex))+
```

```
  geom_point(alpha=2, size=1.5)+
  geom_smooth(se = FALSE)
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

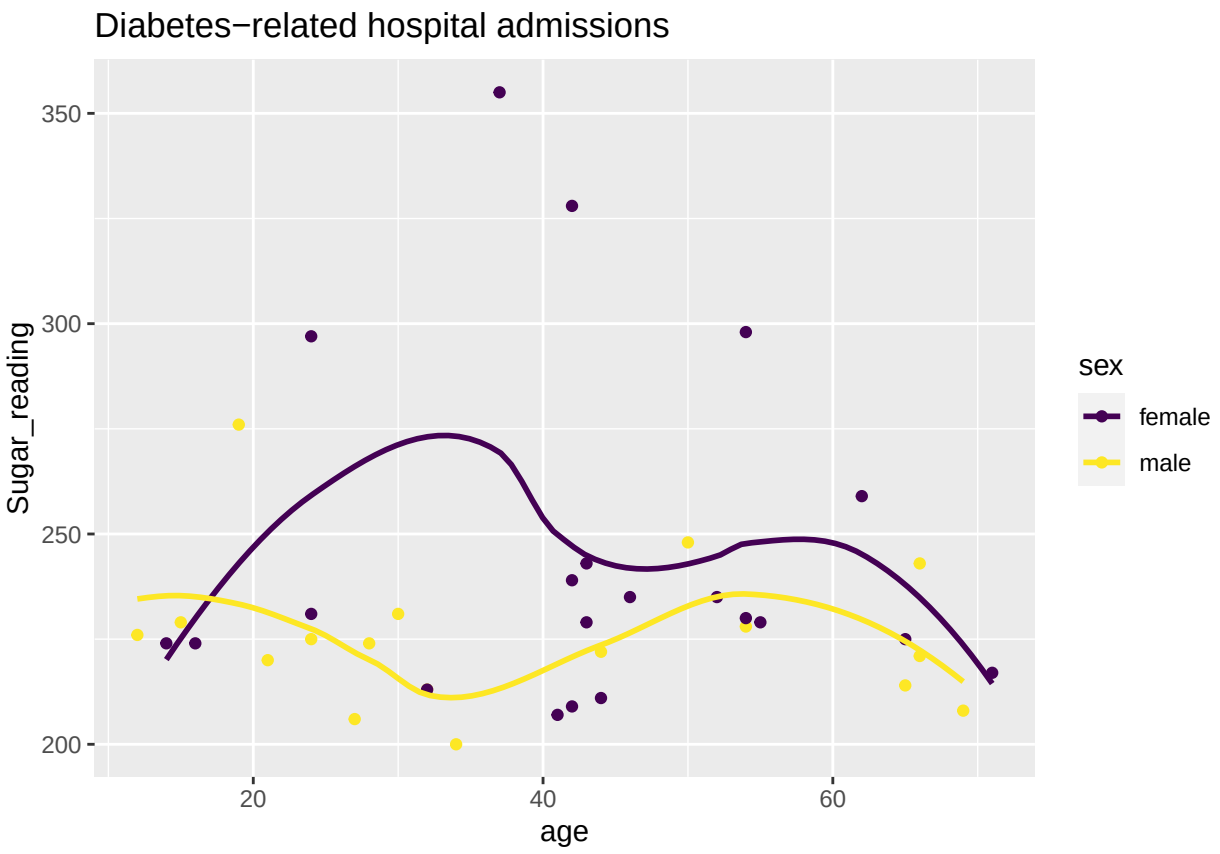


using viridis colour scheme

Colours use in scatter plot for males and females are red and green. Most common colour blindness is known as "red/green colour blindness". Colouring the plot with red and green might make this useless for the people with colour blindness. Viridis colour scheme has been used to colour the scatter plot which provide a wide perceptual range and rely more on blue - yellow contrast.

```
diabetes %>%
 ggplot(aes(x=age,
            y=Sugar_reading,
            color =sex))+
 geom_point()+
 geom_smooth(se = FALSE)+
  ggtitle("Diabetes-related hospital admissions") +  # add title
 scale_color_viridis(discrete=TRUE)         # use the viridis colour scheme
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



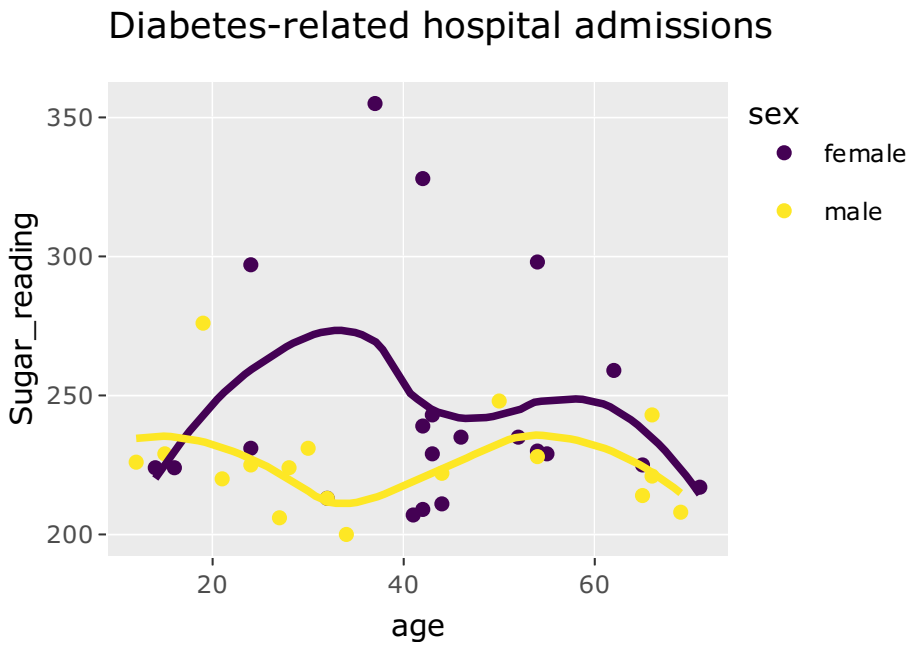
Applying interactive tooltip

R provides a fantastic graphical user interface called tooltip. When hovered over, it presents a brief informative message about that point. Tooltip has been applied by passing x axis and y axis values through tooltip in the ggplotly().

```
p <- diabetes %>%
  ggplot(aes(x=age,
             y=Sugar_reading,
             color =sex))+
  geom_point()+
  geom_smooth(se = FALSE)+
   ggtitle("Diabetes-related hospital admissions") +  # add title
  scale_color_viridis(discrete=TRUE)         # use the viridis colour scheme

ggplotly(p,tooltip = c("x","y"))
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



Summary and reflection

Reproducibility means research data and code are made open and available to all levels of society so that others can reach the same results as are claimed in the scientific output. R is one of best choice when it comes to reproducibility as it free and open-source software along with some other like python. R has scope of writing code and its documentation all in one place with desired format of its output like pdf, word, HTML etc. One of the most powerful features of R is its ability to deal and visualize tabular data and statistical function to do statistical analysis.

Personally, the features that was very helpful was the combination of code chunk and text where the documentation of code can be explained clearly and provision of adding comments next to the code which is very useful in explaining what that particular line of code is doing in the code chunk.

This report has not been created in one single stretch, hence every time the work was in progress a small snippet of information in form of text was added so that whenever the coding resumes, I will have clear picture where did I leave with what conditions and what was planned next in coding. This feature is not just helpful for reproducibility for other, It is also useful in keeping track of what has been done so far.

While creating fake data set with csv file, fake chi numbers were created which are comparatively large number. In csv file these large numbers are converted into scientific or exponential numbers in the cell. This is how large numbers are displayed in csv and reflection of this can be seen in this report as well.

knitting the R markdown file can be tricky sometimes. Absence of some additional package can lead to missing of plot/graph in final document. while knitting this document two graph were missing in the final pdf document which was later resolved by istalling a package called “webshot” that helped in printing the missing graph in final pdf.

References

[1] https://www.fosteropenscience.eu/content/what-are-benefits-open-science

[2] https://cran.r-project.org/web/views/ReproducibleResearch.html