

**Task 1 (a)****User**

Write a R code to read Studentmarks.csv into dataframe df

**ChatGPT**

Here is a basic R code to read the `Studentmarks.csv` file into a dataframe called `df`:

```
```r
# Reading Studentmarks.csv into dataframe df
df <- read.csv("Studentmarks.csv")

# Viewing the first few rows of the dataframe
head(df)
```
```

Make sure the `Studentmarks.csv` file is located in your working directory. If it's in a different directory, you can provide the full path to the file like this:

```
```r
df <- read.csv("/path/to/your/Studentmarks.csv")
```
```

**User**

check the structure of df

**ChatGPT**

To check the structure of the dataframe `df` in R, you can use the `str()` function. Here's how you can do it:

```
```r
# Checking the structure of the dataframe df
str(df)
```
```

This will display the structure of `df`, including the types of each column, the number of observations, and a preview of the data in each column.

**User**

convert dob column into a date object

**ChatGPT**

To convert the `dob` (date of birth) column in the dataframe `df` into a date object in R, you can use the `as.Date()` function. Here's how you can do it:

```
```r
# Converting the dob column into a date object
df$dob <- as.Date(df$dob, format = "%Y-%m-%d") # Adjust the format as needed

# Checking the structure of the dataframe again to confirm the conversion
str(df)
```
```

**### Notes:**

- Adjust the `format` parameter in `as.Date()` according to the format of your `dob` values. For example:
  - If your dates are in the format `dd/mm/yyyy`, use `format = "%d/%m/%Y"`.
  - If they are in the format `mm/dd/yyyy`, use `format = "%m/%d/%Y"`.

Make sure to check the format of your dates in the `dob` column before applying the conversion.

**User**

Calculate current age from column 'dob' and store it in a new column 'age1' with 5 decimal points

**ChatGPT**

To calculate the current age from the `dob` column and store it in a new column called `age1` with five decimal points, you can follow these steps in R:

1. Calculate the age by subtracting the date of birth from the current date.
2. Convert the difference into years.
3. Store the result in the `age1` column.

Here's the code to accomplish that:

```
```r
# Load the necessary package
library(lubridate)

# Calculate the current age from the dob column and store it in a new column 'age1'
df$age1 <- round(as.numeric(difftime(Sys.Date(), df$dob, units = "days")) / 365.25, 5)

# View the updated dataframe
head(df)
```
```

**### Explanation:**

- `Sys.Date()` gets the current date.

- ``difftime()`` calculates the difference in days between the current date and the ``dob`` column.
- Dividing by ``365.25`` accounts for leap years.
- ``round(..., 5)`` rounds the age to five decimal points.

After running this code, you'll have a new column ``age1`` in your dataframe ``df`` that contains the age of each student calculated from their date of birth.

**User**

avoid lubridate library

**ChatGPT**

If you want to avoid using the ``lubridate`` library, you can calculate the current age directly using base R functions. Here's how to do it:

```
```r
# Calculate the current age from the dob column and store it in a new column 'age1'
df$age1 <- round(as.numeric(difftime(Sys.Date(), df$dob, units = "days")) / 365.25, 5)

# View the updated dataframe
head(df)
```

### Explanation:
- `Sys.Date()` gets the current date.
- `difftime()` computes the difference in days between the current date and the `dob` column.
- Dividing by `365.25` converts the difference from days to years, accounting for leap years.
- `round(..., 5)` rounds the age to five decimal points.
```

This will add the ``age1`` column to your dataframe ``df`` without using any external libraries.

**User**

Extract day, month, and year into separate columns from dob columns

**ChatGPT**

To extract the day, month, and year from the ``dob`` column in the dataframe ``df`` and store them in separate columns, you can use the ``format()`` function in base R. Here's how you can do it:

```
```r
# Extract day, month, and year from the dob column into separate columns
df$day <- as.integer(format(df$dob, "%d")) # Extract day
df$month <- as.integer(format(df$dob, "%m")) # Extract month
df$year <- as.integer(format(df$dob, "%Y")) # Extract year

# View the updated dataframe
head(df)
```

### Explanation:
- `format(df$dob, "%d")` extracts the day from the date.
- `format(df$dob, "%m")` extracts the month.
- `format(df$dob, "%Y")` extracts the year.
- `as.integer()` converts the extracted values into integer format.
```

After running this code, your dataframe ``df`` will have three new columns: ``day``, ``month``, and ``year``, containing the respective components from the ``dob`` column.

**User**

get current year

**ChatGPT**

To get the current year in R, you can use the ``format()`` function with ``Sys.Date()``. Here's how to do it:

```
```r
# Get the current year
current_year <- as.integer(format(Sys.Date(), "%Y"))

# Print the current year
print(current_year)
```

### Explanation:
- `Sys.Date()` retrieves the current date.
- `format(..., "%Y")` extracts the year from that date.
- `as.integer()` converts the year into an integer.
```

After running this code, ``current_year`` will contain the current year as an integer.

**User**

get current year, no explanation needed and no print

**ChatGPT**

```
```r
current_year <- as.integer(format(Sys.Date(), "%Y"))
```
```

**User**

subtract `df$year` from current year and store in new column called `age2`

**ChatGPT**

```
```r
df$age2 <- current_year - df$year
```
```

**User**

drop dob column

**ChatGPT**

```
```r
df$dob <- NULL
```
```

**User**

Change the name of the column from day to date

**ChatGPT**

```
```r
colnames(df)[which(names(df) == "day")] <- "date"
```
```

**User**

Create a scatter plot for X2020,X2021,X2022 versus Studentname using ggplot

**ChatGPT**

Here's how to create a scatter plot for the columns `X2020`, `X2021`, and `X2022` versus `Studentname` using `ggplot2` in R:

```
```r
# Load ggplot2 library
library(ggplot2)

# Reshape the dataframe for plotting
library(reshape2)
df_long <- melt(df, id.vars = "Studentname", measure.vars = c("X2020", "X2021", "X2022"))

# Create scatter plot
ggplot(df_long, aes(x = variable, y = value, color = Studentname)) +
  geom_point() +
  labs(x = "Year", y = "Marks", title = "Scatter Plot of Marks by Student Name") +
  theme_minimal()
```

### Notes:
- This code first reshapes the dataframe from wide to long format using the `melt()` function from the `reshape2` package to make it suitable for plotting.
- Then, it creates a scatter plot using `ggplot2` where `variable` (years) is on the x-axis and `value` (marks) is on the y-axis, with points colored by `Studentname`.
```

**User**

Create a scatter plot for X2020,X2021,X2022 versus Studentname, with no reshaping, theme or labeling

**ChatGPT**

Here's how to create a scatter plot for `X2020`, `X2021`, and `X2022` versus `Studentname` without reshaping the data, adding themes, or labels:

```
```r
# Load ggplot2 library
library(ggplot2)

# Create scatter plot
ggplot(df, aes(x = Studentname)) +
  geom_point(aes(y = X2020, color = "blue")) +
  geom_point(aes(y = X2021, color = "red")) +
  geom_point(aes(y = X2022, color = "green"))
```
```

**User**

Create a scatter plot for x-axis: X2020,X2021,X2022 versus Studentname

**ChatGPT**

To create a scatter plot with `X2020`, `X2021`, and `X2022` on the x-axis versus `Studentname` on the y-axis, you can use the following code:

```
```r
# Load ggplot2 library
library(ggplot2)

# Create scatter plot
ggplot(df, aes(x = value, y = Studentname)) +
  geom_point(aes(x = X2020, color = "blue")) +
  geom_point(aes(x = X2021, color = "red")) +
  geom_point(aes(x = X2022, color = "green"))
```
```

This code assumes that you want to plot the three year columns (`X2020`, `X2021`, `X2022`) against the `Studentname`. However, to ensure they are plotted together clearly, you typically need to reshape the data. If reshaping is not desired, you can still plot them separately as shown, but with multiple layers.

If you would like a different representation or have any other specific requirements, let me know!

**User**

Create a scatter plot for x-axis: X2020,X2021,X2022 versus Studentname with no specific colors

**User**

Create a scatter plot for x-axis: X2020,X2021,X2022 versus Studentname with color difference of x-axis

**ChatGPT**

To create a scatter plot with `X2020`, `X2021`, and `X2022` on the x-axis versus `Studentname` on the y-axis, with different colors for each year, you can use the following code:

```
```r
# Load ggplot2 library
library(ggplot2)

# Reshape the data to long format for plotting
library(reshape2)
df_long <- melt(df, id.vars = "Studentname", measure.vars = c("X2020", "X2021", "X2022"))

# Create scatter plot with different colors for each year
ggplot(df_long, aes(x = value, y = Studentname, color = variable)) +
  geom_point()
```

### Explanation:
- The data is reshaped to long format using `melt()` so that `X2020`, `X2021`, and `X2022` can be treated as a single variable in the plot.
- The `color` aesthetic is set to differentiate the points based on the year variable.
```

This will create a scatter plot with different colors for each of the `X2020`, `X2021`, and `X2022` values, plotted against `Studentname`.

**User**

Create a scatter plot for x-axis: X2020,X2021,X2022 versus Studentname, only use ggplot2 library

**ChatGPT**

To create a scatter plot with `X2020`, `X2021`, and `X2022` on the x-axis versus `Studentname` on the y-axis using only the `ggplot2` library (without reshaping), you can layer multiple `geom\_point()` calls for each of the year columns. Here's how to do it:

```
```r
# Load ggplot2 library
library(ggplot2)

# Create scatter plot
ggplot() +
  geom_point(data = df, aes(x = X2020, y = Studentname), color = "red") +
  geom_point(data = df, aes(x = X2021, y = Studentname), color = "blue") +
  geom_point(data = df, aes(x = X2022, y = Studentname), color = "green")
```

### Explanation:
- Each `geom_point()` function call adds points for a specific year against `Studentname`, and you can choose different colors for each year.
- The `ggplot()` function initializes the plot without setting a default dataset, allowing you to specify different datasets in each `geom_point()` call.
```

This code will produce a scatter plot with different colors representing the values of `X2020`, `X2021`, and `X2022` against `Studentname`.

**User**

Create a scatter plot for x-axis: X2020,X2021,X2022 versus Studentname, only use ggplot2 library, no explanations needed and color=x

**ChatGPT**

```
```r
# Load ggplot2 library
library(ggplot2)

# Create scatter plot
ggplot(df) +
  geom_point(aes(x = X2020, y = Studentname, color = "X2020")) +
  geom_point(aes(x = X2021, y = Studentname, color = "X2021")) +
  geom_point(aes(x = X2022, y = Studentname, color = "X2022"))
```
```

**User**

Add label to X and Y as Marks and Studentsname

**ChatGPT**

```
```r
# Load ggplot2 library
library(ggplot2)

# Create scatter plot with labels
```

```
ggplot(df) +
  geom_point(aes(x = X2020, y = Studentname, color = "X2020")) +
  geom_point(aes(x = X2021, y = Studentname, color = "X2021")) +
  geom_point(aes(x = X2022, y = Studentname, color = "X2022")) +
  labs(x = "Marks", y = "Studentname")
```

```

**User**

add color legend title to "year" also

**ChatGPT**

```
```r
# Load ggplot2 library
library(ggplot2)

# Create scatter plot with labels and color legend title
ggplot(df) +
  geom_point(aes(x = X2020, y = Studentname, color = "X2020")) +
  geom_point(aes(x = X2021, y = Studentname, color = "X2021")) +
  geom_point(aes(x = X2022, y = Studentname, color = "X2022")) +
  labs(x = "Marks", y = "Studentname", color = "Year")
```

```

**User**

Without creating new data frame, create a new column called total marks with sum of X2020, X2021, X2022 using mutate function, use filter function to filter total marks>=200 , no boiler plate code and no theme, no explanations

**ChatGPT**

```
```r
library(dplyr)

df <- df %>%
  mutate(total_marks = X2020 + X2021 + X2022) %>%
  filter(total_marks >= 200)
```

```

**User**

plot as bar diagram , Studentsname vs Total Marks

**ChatGPT**

```
```r
ggplot(df, aes(x = Studentname, y = total_marks)) +
  geom_bar(stat = "identity")
```

```

**User**

Without creating new data frame or updating data frame, create a new column called total marks with sum of X2020, X2021, X2022 using mutate function, use filter function to filter total marks>=200 and plot as bar diagram , Studentsname vs Total Marks , no boiler plate code and no theme, no explanations

**ChatGPT**

```
```r
library(dplyr)
library(ggplot2)

df %>%
  mutate(total_marks = X2020 + X2021 + X2022) %>%
  filter(total_marks >= 200) %>%
  ggplot(aes(x = Studentname, y = total_marks)) +
  geom_bar(stat = "identity")
```

```

**User**

Arrange totalmarks in descending order as to show in a plot

**ChatGPT**

```
```r
df %>%
  mutate(total_marks = X2020 + X2021 + X2022) %>%
  filter(total_marks >= 200) %>%
  arrange(desc(total_marks)) %>%
  ggplot(aes(x = Studentname, y = total_marks)) +
  geom_bar(stat = "identity")
```

```

**User**

show bar chart in descending order of total marks from left to right

**ChatGPT**

```
```r
df %>%
  mutate(total_marks = X2020 + X2021 + X2022) %>%
  filter(total_marks >= 200) %>%
  arrange(desc(total_marks)) %>%
  ggplot(aes(x = reorder(Studentname, total_marks), y = total_marks)) +
  geom_bar(stat = "identity")
```

```

```
  geom_bar(stat = "identity")
...`
```

**User**

show highest totalmarks in left and so on

**ChatGPT**

```
```r
df %>%
  mutate(total_marks = X2020 + X2021 + X2022) %>%
  filter(total_marks >= 200) %>%
  arrange(total_marks) %>%
  ggplot(aes(x = reorder(Studentname, total_marks), y = total_marks)) +
  geom_bar(stat = "identity")
...`
```

**User**

remove arrange function and alternate the reorder function with -totalmarks

**ChatGPT**

```
```r
df %>%
  mutate(total_marks = X2020 + X2021 + X2022) %>%
  filter(total_marks >= 200) %>%
  ggplot(aes(x = reorder(Studentname, -total_marks), y = total_marks)) +
  geom_bar(stat = "identity")
...`
```

**User**

label X=Students name and y= Total Marks

**ChatGPT**

```
```r
df %>%
  mutate(total_marks = X2020 + X2021 + X2022) %>%
  filter(total_marks >= 200) %>%
  ggplot(aes(x = reorder(Studentname, -total_marks), y = total_marks)) +
  geom_bar(stat = "identity") +
  labs(x = "Students Name", y = "Total Marks")
...`
```

**User**

Give Title of the bar chart as "Total Marks of Students (>=200) - Descending Order"

**ChatGPT**

```
```r
df %>%
  mutate(total_marks = X2020 + X2021 + X2022) %>%
  filter(total_marks >= 200) %>%
  ggplot(aes(x = reorder(Studentname, -total_marks), y = total_marks)) +
  geom_bar(stat = "identity") +
  labs(x = "Students Name", y = "Total Marks",
       title = "Total Marks of Students (>=200) - Descending Order")
...`
```