

1. Create a R Markdown document using html output
2. Install R package [randomNames](#)
3. Generate 100 student names (first and last names) using random names package.
4. Generate 100 grades for above students. Random number should have a mean of 70 with standard distribution of 15. Below functions, [rnorm](#) and [floor](#) would be useful.

`rnorm(n, mean=a, sd=b) floor`

5. Put student names and grades to one tibble (data frame) You can use [as\\_tibble](#) function for this purpose. Your tibble should have at least 2 columns (names, grades).
6. export this data frame as csv file, student\_grades.csv.

Note the difference between [write.csv](#) which is in base R library and [write\\_csv](#) which is readr library in tidyverse.

7. Deploy your R Markdown document to Rpubs

1. please install gapminder package. Gapminder contains life expectancy, GDP per capita, and population by country.

```
install.packages("gapminder")  
library(tidyverse)  
library(gapminder)  
data(gapminder)
```

2. using dataset of gapminder, plot the population vs time of Germany. You should use
  - geom\_point
  - geom\_line

which one is better?

3. Create same plots for your own country.
4. Make a histogram of for life expectancy values for year 2007. Do not forget that you need to subset your dataset
5. Create a facets of histograms life expectancy values by year
6. Create a line plot of life expectancy values for the countries. In the aesthetics, you need to set x,y and by
7. Create a line plot of life expectancy values for the countries. Set the color for the continent values In the aesthetics, you need to set x,y, by and color.
8. Create a line plot of life expectancy values for the countries. create the facets for the continent values
9. Create a R Markdown document that shows your plots
10. Deploy your R Markdown document to Rpubs

1. create a R Studio Project named Lab-2021-10-19
2. create a R Markdown document using html output
3. rename YAML part of your document with following

---

**title: "Lab-2021-10-19" author: "Put your name here" output: html\_document**

4. Write following setup code in your r markdown document

```
5.      #install.packages("gapminder")
6.      library(tidyverse)
7.      library(gapminder)
8.      data(gapminder)
```

9. create headers using ## for every question of this lab. like

## Question 5

6. Write necessary code so that following \_\_\_\_ parts will be filled. You need to use inline code for this purpose.

Gapminder dataset contains information about countries and their life expectancy. It has \_\_\_\_ rows and \_\_\_\_ variables.

7. print columns of gapminder dataset as html table. Use both normal output and knitr::kable output.
8. Create a plot of the population vs time of France
9. knit your document to html
10. knit your document to word
11. end your document with running following code.

```
sessionInfo()
```

12. Deploy your R Markdown document to Rpubs

1. create a R Studio Project named Lab-2021-10-19
2. create a shiny document
3. give it Application Name: AppTemplate
4. choose Application Type: Single File (app.R) option
5. Run application and see your web application
6. Change your slider input and see your histogram change
7. create an another shiny document
8. give it Application Name: AppHelloWorld
9. choose Application Type: Multiple File (ui.R/server.R) option
10. in the ui.R remove all code and copy paste following code

```
library(shiny)

shinyUI(
  fluidPage("Hello World")
)
```

11. Run application and see your web application, saying Hello World
12. Stop your application and change ui.R with following code

```
library(shiny)
shinyUI(
  fluidPage(
    textInput("name", "Enter your Name"),
    textOutput("outputHello")
  )
)
```

We are using a simple [text input](#) here.

13. Run application and see your input textbox. It does nothing right now but can enter text in it.
14. replace your server.R code with the following code

```
library(shiny)

shinyServer(function(input, output) {

  output$outputHello <- renderText({
    paste("Hello ", input$name)
  })

})
```

15. Deploy your application to <http://www.shinyapps.io/>

---

5

1. install e1071 package in R

```
install.packages("e1071")
```

2. run following R file in the RStudio

[classification\\_iris\\_full\\_data.R](#)

3. In our R codes, we will follow the same approach always. Our classification codes will call the function and give which column should be predicted. Here we are trying to predict Species. Using data argument we set our data set.

```
model = naiveBayes(Species~.,data=df)
```

For prediction, we use predict function and give our model and test data.

```
predict(model,data)
```

In this file, we use following line, since only first 4 columns contains our X values. Our target is 5th column; therefore, we exclude it from our data frame.

```
predict(model,df[,1:4])
```

4. run following R file in the RStudio

[classification\\_iris\\_train\\_test\\_split.R](#)

Since R do not have formal train test split function. We use sample function of R to sample rows from R data frame. Using negative indexing we get test data as in the below code.

```
test_data = df[-train_rows,]
```

Other differences in our code to use train and test data in our model and predict codes. Since we have to use train and test data in our calls.

```
model = naiveBayes(Species~.,data=train_data)
predicted_values = predict(model,test_data[,1:4])
```

```
correctly_predicted = sum(predicted_values == test_data[, "Species"])
```

5. install rpart package in R

```
install.packages("rpart")
```

6. save as [classification\\_iris\\_full\\_data.R](#) code as classification\_iris\_full\_data\_dt.R

7. You need to change the code so that you will be using rpart instead of naiveBayes in your classification

8. `predict.rpart` does not work like `naiveBayes`. You need to give one more argument to `predict` part, like below.

```
predicted_values = predict(model,df[,1:4],type = "class")
```

9. save as [classification\\_iris\\_train\\_test\\_split.R](#) as `classification_iris_train_test_split_dt.R`

and do similar changes to it so that it will work as decision tree `rpart`.

10. Create a R markdown document that shows above steps in your document

11. Deploy your R Markdown document to Rpubs

---

6

1. load the [example-data-bad.csv](#) file to rstudio. You will have problems loading this file. You need to customize `read_csv` call so that you can load this file.
2. Create a R markdown document
3. In this R Markdown document show `example-data-bad` contents in table
4. Deploy your R Markdown document to Rpubs

---

7

1. Do either one of the following
  - a. goto <https://www.kaggle.com/drubal/top-100-global-steel-producers-20112016> download the dataset. You may need to create a account in kaggle
  - b. [Download from github](#)
2. import the dataset in a tibble
3. this dataset is not in tidy format. Transform the dataset so that it is in tidy format.
4. using `ggplot2` create a bar chart which shows total tonnage of steel produced between 2011-2016. That is you must sum these years to one column. For example Turkey should show 53.21.
5. Create a pie chart which shows percentage of the whole. This figure will show same information as in previous part, but it shows it differently.

6. Create a RMarkdown file.
7. Put these two figures to this markdown file.
8. Deploy your application to Rpubs