# Shiny – Case Study (Telecom App)

# Contents

# Making of a Shiny App - Recap

- **Structure of the app consists of a user interface and server script.** The user interface script is used for layout and appearance purposes.

- It is generally saved as **ui.R** file on the working directory. While the server script contains the instructions that the computer needs to build an app. The server script is usually named as **server.R** and saved in the working directory for the app being developed.

# Data Snapshot

Telecom Data

Variables

| CustID | Week | Calls | Minutes | Amt | Gender | Active | Age_Group |
|---|---|---|---|---|---|---|---|
| 1001 | 1 | 56 | 392 | 78.4 | F | Yes | 18-30 |
| 1001 | 2 | 49 | 735 | 154.35 | F | Yes | 18-30 |
| 1001 | 3 | 140 | 420 | 126 | F | Yes | 18-30 |
| 1001 | 4 | 182 | 1638 | 393.12 | F | Yes | 18-30 |

| Columns | Description | Type | Measurement | Possible values |
|---|---|---|---|---|
| CustID | Customer ID | Integer | - | - |
| Week | Week Number | Integer | 1 – 24 | 24 |
| Calls | Number of Calls | Integer | - | Positive values |
| Minutes | Time Spent on Calls in Minutes | Integer | minutes | Positive values |
| Amt | Amount Spent (in Rs.) | Numeric | Rs. | Positive values |
| Gender | Gender of customer | Charater | M, F | 2 |
| Active | Active status | Charater | Yes, No | 2 |
| Age_Group | Age Group | Charater | 18-30, 30-45, >45 | 3 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1001 | 24 | 70 | 980 | 264.6 | F | Yes | 18-30 |

* Here we continue to use previous data Telecom Data.

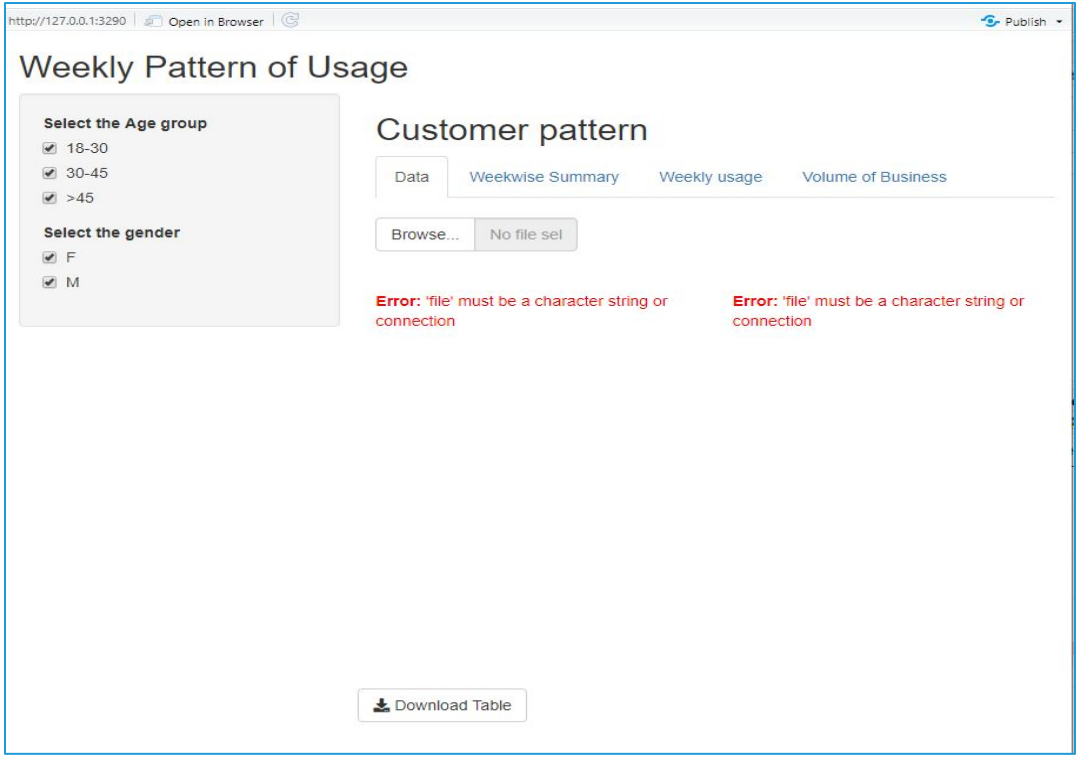
# The Telecom App Structure

- Title panel has name of the application.
- Sidebar layout has sidebar panel with input widgets and main panel with outputs.
- Main panel of the app has four tabs:
  1. Data:
     - Table for weekly usage pattern of the customers across various demographics where one imports data & can also download the table.
     - Pie chart of Active customers
  2. **Weekwise Summary:** Summary of data.
  3. **Weekly Usage:** Line chart showing weekly usage pattern of the customers across various demographics.
  4. Volume of Business.

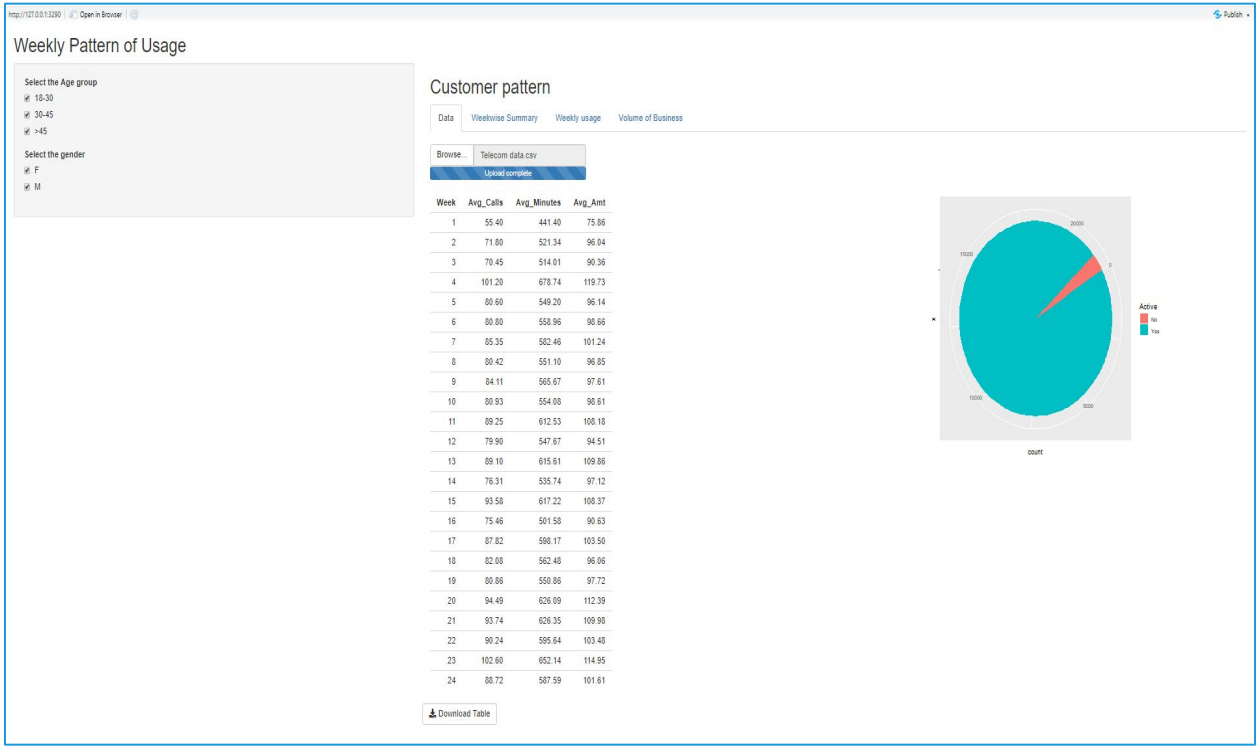
Let's have a look at the app design first and then begin to create our app.

# App Design – Tab 1

On opening the App & before importing data :

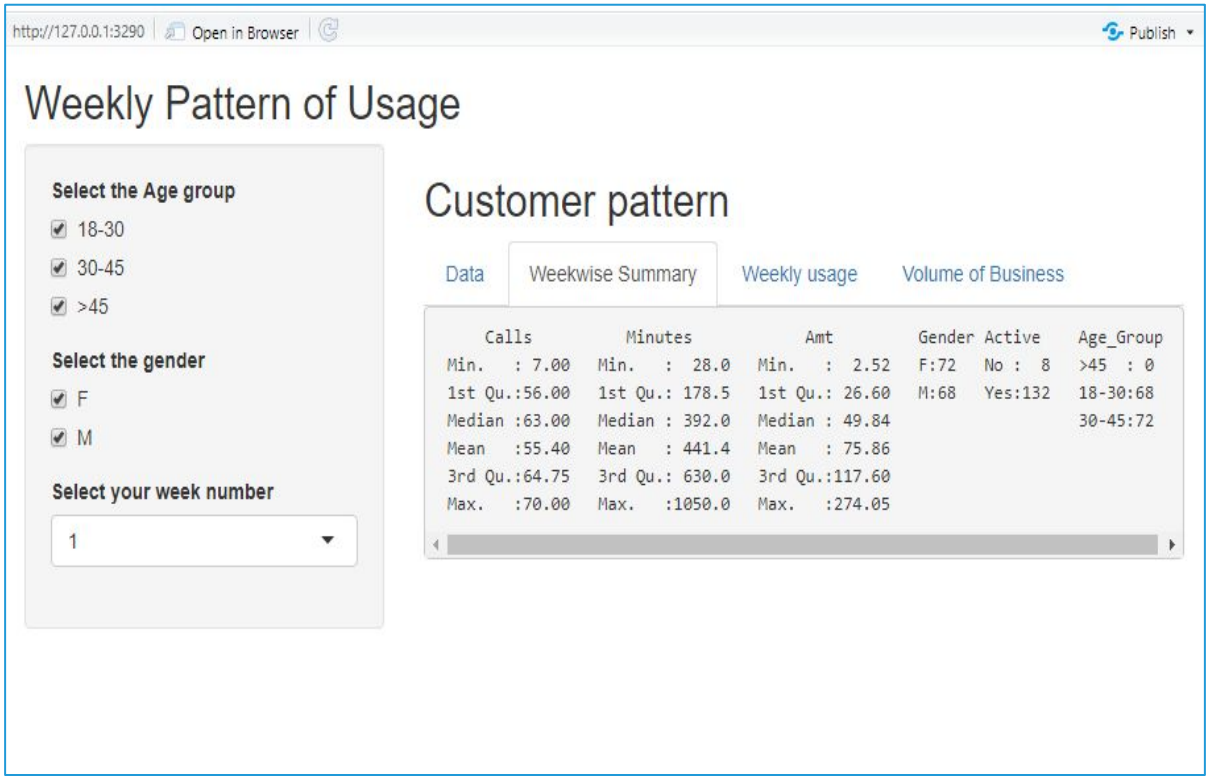


# App Design – Tab 1

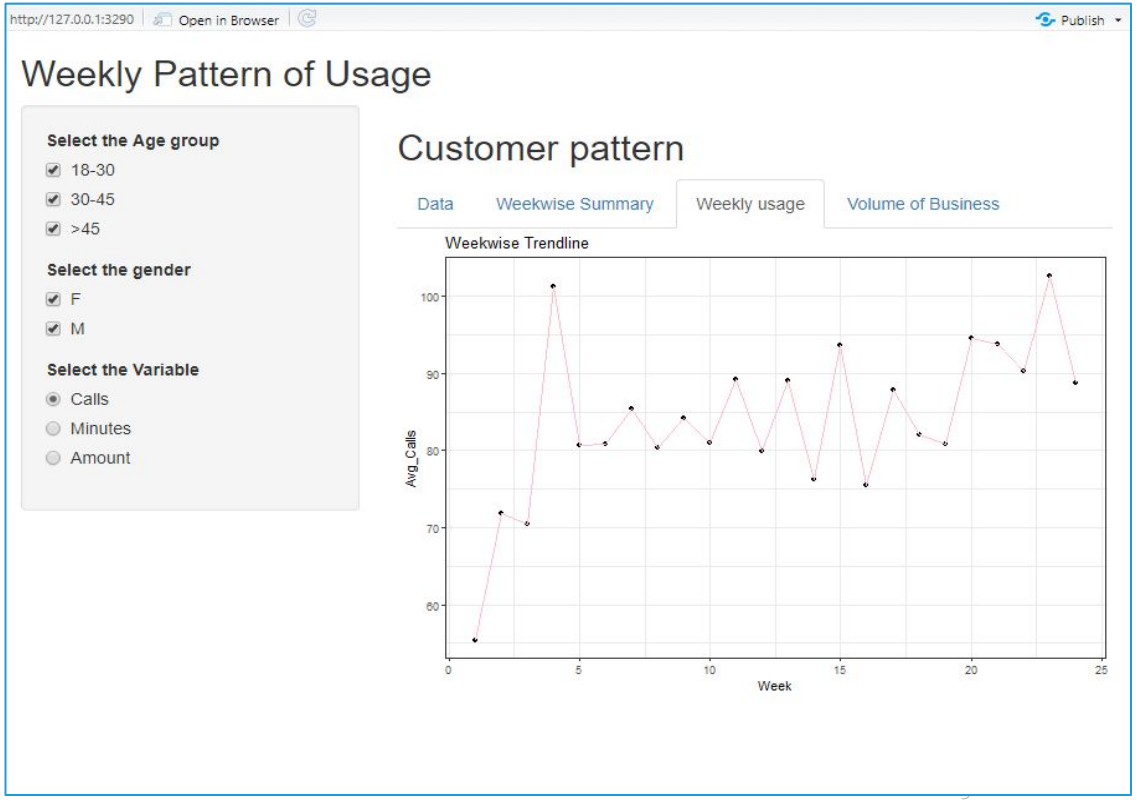



| Week | Avg_Calls | Avg_Minutes | Avg_Amt |
|---|---|---|---|
| 1 | 55.40 | 441.40 | 75.86 |
| 2 | 71.60 | 521.34 | 96.04 |
| 3 | 70.45 | 514.01 | 90.36 |
| 4 | 101.20 | 678.74 | 119.73 |
| 5 | 80.60 | 549.20 | 96.14 |
| 6 | 80.80 | 558.96 | 98.66 |
| 7 | 85.35 | 582.46 | 101.24 |
| 8 | 80.42 | 551.10 | 96.85 |
| 9 | 84.11 | 565.67 | 97.61 |
| 10 | 80.93 | 554.08 | 98.61 |
| 11 | 89.25 | 612.53 | 108.18 |
| 12 | 79.90 | 547.67 | 94.51 |
| 13 | 89.10 | 615.61 | 109.86 |
| 14 | 76.31 | 535.74 | 97.12 |
| 15 | 93.58 | 617.22 | 108.37 |
| 16 | 75.46 | 501.58 | 90.63 |
| 17 | 87.82 | 598.17 | 103.50 |
| 18 | 82.08 | 562.48 | 96.06 |
| 19 | 80.86 | 550.86 | 97.72 |
| 20 | 94.49 | 626.09 | 112.39 |
| 21 | 93.74 | 626.35 | 109.98 |
| 22 | 90.24 | 595.64 | 103.48 |
| 23 | 102.60 | 652.14 | 114.95 |
| 24 | 88.72 | 587.59 | 101.61 |

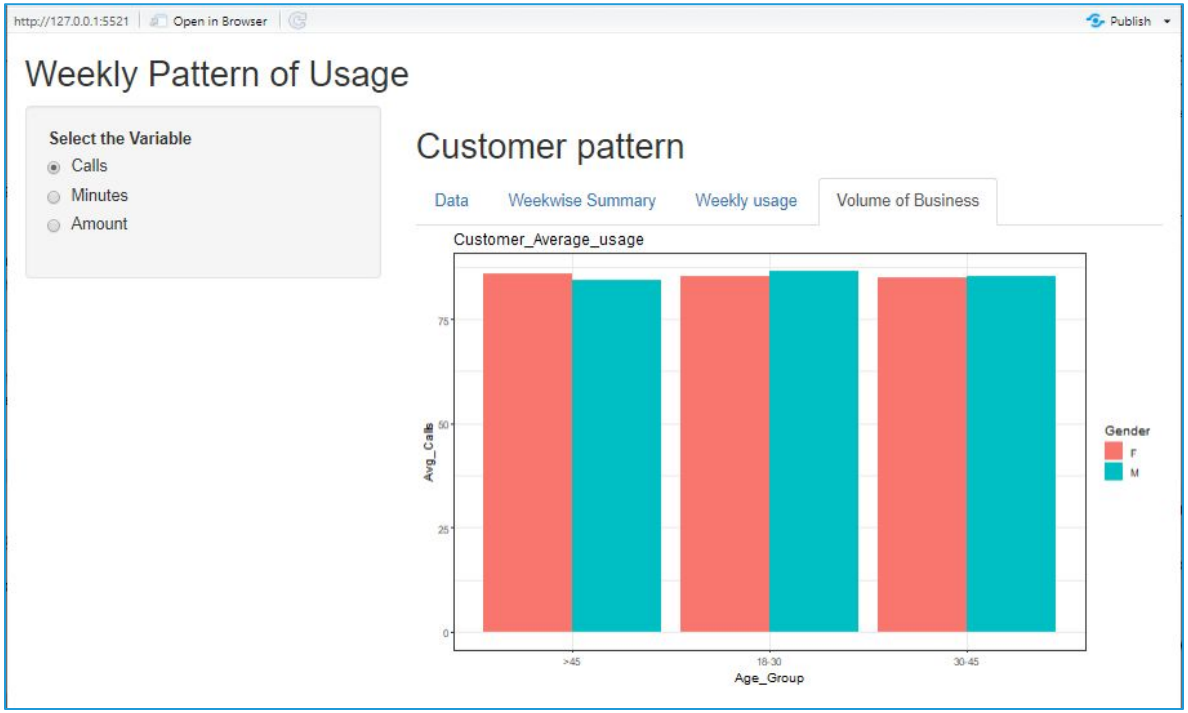
# App Design – Tab 2



# App Design – Tab 3



# App Design – Tab 4




**Note : Always have the R codes ready before starting shiny app coding.**

# ui.R

```
# ui.R

#Load the required packages.
library(shiny)
library(shinyjs)

fluidPage(

useShinyjs(),
```

- Package **shiny** is used for creating Shiny App.
- Package **shinyjs** allows us to perform JavaScript operations in Shiny apps that greatly improves our apps without having to know any JavaScript.

**fluidPage()** makes the app page responsive.

Every time any function from the **shinyjs()** is used, **useShinyjs()** has to be mentioned.

# ui.R

```
# ui.R continued
titlePanel("Weekly Pattern of Usage"),
  sidebarLayout(
    sidebarPanel(
      hidden(checkboxGroupInput("age","Select the Age
group",choices=c("18-30","30-45",">45"),selected=c("18-30","30-45",">4
5"))),
      hidden(checkboxGroupInput("gender","Select the
gender",choices=c("F","M"),selected=c("F","M"))),
```

The **titlePanel** and the **sidebarPanel** consists of widgets for age, gender, week number and variable.

- **hidden()** function from **shinyjs** package makes the Shiny tag invisible when Shiny app starts.
- **checkboxGroupInput()** creates a group of checkboxes.
- **First argument** is the ID with which **server.R** will identify the input
- **Second argument** is label for the widget
- **choices=** is for the list of values to show check boxes for
- **selected=** is for the value that should be initially selected.

# ui.R

```
hidden(
       selectInput("ID","Select your week number",choices=c(1:24))),
hidden(
        radioButtons("var",label="Select the Variable",
        choices=c("Calls"=2,"Minutes"=3,"Amount"=4)))
       ),
```

- **selectInput()** creates a select list control that can be used to choose a single or multiple items from a list of values.
- **radioButtons()** creates a set of radio buttons used to select  an item from a list.

- With this **`sidebarPanel()`** body ends and now we will define **`mainPanel()`**.

# ui.R

```
# Define the main panel with various tabs
mainPanel(
  h2("Customer pattern"),
  tabsetPanel(id="tabs",type="tabs",
    tabPanel("Data",value="data",
                 fluidRow(column(width=4,
                                 fileInput("file1","", multiple = TRUE,
                                           accept = c("text/csv",
"text/comma-separated-values,text/plain", ".csv")
                                      )
                                  )),
                 fluidRow(column(6,tableOutput("weeks")),
                          column(6,plotOutput("pie"))),
                 fluidRow(downloadButton('download',"Download Table",
class = "butt"))
        ),
```

# ui.R

```
# Define the main panel with various tabs
tabPanel("Weekwise Summary",value="summa", verbatimTextOutput("sum")),
tabPanel("Weekly usage",value="Usag",plotOutput("usage")),
tabPanel("Volume of Business",value="vol",plotOutput("bar"))
        )
    )
  ))
```

- **mainPanel()** creates a main panel containing output elements that can in turn be passed to **sidebarLayout()**.
- HTML content can be added to Shiny app by placing it inside a ***Panel()** function
- **h2()** is a Shiny's HTML tag function which creates a second level header.
- **tabsetPanel()** is to create multiple tabs with **tabPanel**(). It takes the arguments as a set of **tabPanels**, **id=** for the server logic to determine which of the current tabs is active and **type=** for the type of tabs.
- In **tabPanel()** first argument is the display title for the tab,
- **value=** value that should be sent when **tabsetPanel()** reports that this tab is selected,
- **last argument** is the placeholder for the output.

# server.R

```
# Load the required packages

library(ggplot2)
library(dplyr)
library(reshape2)

function(input,output,session){
```

Defining the **function** and **session** argument for shiny server. This is for the use of **ObserveEvent()** command & **shinyjs()** function.

```
# Coding for tab1 i.e. Data

# import data
  imported_data <- reactive({

      df <- read.csv(input$file1$datapath,header = TRUE)
      df$Gender <- as.factor(df$Gender)
      df$Active <- as.factor(df$Active)
      df$Age_Group <- as.factor(df$Age_Group)
      return(df)
    })
```

**reactive()** function helps executing codes based on the data imported.

# server.R

```
# Coding for tab1 i.e. Data
```

```
week_Avg <- reactive({
    teledata <- imported_data()

    xy <- subset(teledata, (Age_Group %in% input$age) & (Gender %in%
input$gender))

summarise(group_by(xy,Week),Avg_Calls=mean(Calls),Avg_Minutes=mean(Min
utes),Avg_Amt=mean(Amt))

  })

  output$weeks<-renderTable({
     week_Avg()
     })
```

- **renderTable()** displays table as an output.
- **xy** is the subset of **imported_data()** where the age and gender inputs are taken from the user through the widgets.
- **summarise(group_by())** is displaying the weekly Average no. calls, Average time spent on calls (in Minutes) and Average Amount spent by the customers.

# server.R

```
# Coding for tab1 i.e. Data
```

```
output$download <- downloadHandler(
    filename = function(){
      paste("Weekly Summary.csv")
    },
    content = function(file){
      write.csv(week_Avg(), file, row.names = FALSE)
    },
    contentType = "text/csv"
  )
```

- **downloadHanderler()** allows content from the Shiny application to be made available to the user as file downloads.
- **filename=** A string of the filename, including extension, that the user's web browser should default to when downloading the file; or a function that returns such a string.
- **content=** A function that takes a single argument file that is a file path (string) of a nonexistent temp file, and writes the content to that file path.
- **contentType=** A string of the download's content type, for example "text/csv" or "image/png"

# server.R

```
# Coding for tab1 i.e. Data
```

```
# Pie Chart representing Active Status of Customers :

output$pie <- renderPlot({
    teledata <- imported_data()

    xy <- subset(teledata, (Age_Group %in% input$age) & (Gender %in%
input$gender))

    pie <- ggplot(xy, aes(x="",fill=Active))+
           geom_bar(width = 1)+
           coord_polar(theta = "y", start = pi/3)
    pie

  })
```

# server.R

```
# Coding for tab2 i.e Weekwise Summary
```

```
output$sum<-renderPrint({
    teledata <- imported_data()

    summary(subset(teledata,(Week %in% input$ID) & (Age_Group %in%
input$age) & (Gender %in% input$gender), select=c(-CustID,-Week)))

      })
```

- Here we have defined the function for giving summary as the output **"as-it-is "**, hence used **renderPrint()** and not **renderText().**
- It will print the **summary()** of the subset of **teledata** where the age, gender and week inputs are taken from the user through input widgets.

# server.R

```
# Coding for tab3 i.e. Weekly Usage Summary
  output$usage<-renderPlot({
    teledata <- imported_data()
    xy<-subset(teledata,(Age_Group %in% input$age) & (Gender %in%
input$gender))

    xyz <- as.data.frame(summarise(group_by(xy,Week),
                 Avg_Calls=mean(Calls), Avg_Minutes=mean(Minutes),
                 Avg_Amt=mean(Amt)))

    qplot(x=Week, y=xyz[,as.numeric(input$var)],data=xyz, xlab="Week",
          ylab=paste(names(xyz[as.numeric(input$var)])),
          main="Weekwise Trendline")+
          geom_line(size=0.7,colour="pink") + theme_bw()
      })
```

- **renderPlot()** displays a plot as an output.
- **xy** is the subset of **teledata** which uses the age, gender taken from the user through the widgets.
- **xyz** has the weekly Average no. calls, Average time spent on called (in Minutes) and Average Amount spent by the customers.
- **qplot()** is displaying line chart representing **xyz** for the selected variable.

# server.R

```
# Coding for tab4 i.e. Volume of Business

output$bar <- renderPlot({
    teledata <- imported_data()

xy<-subset(teledata,(Age_Group %in% input$age) & (Gender %in%
input$gender))

xyz <- as.data.frame(summarise(group_by(xy,Week),
                    Avg_Calls=mean(Calls), Avg_Minutes=mean(Minutes),
                    Avg_Amt=mean(Amt)))

teledata_new <- subset(teledata, select=c(-Week))

data <-with(teledata_new, tapply(teledata_new[,as.numeric(input$var)],
           list(Gender,Age_Group),mean))

data.m <- melt(data,id.vars=Gender)
colnames(data.m)[1] <- "Gender"
```

**renderPlot()** displays plot.

Object **data** uses the third input – variable.

* Note: All inputs are taken from the user at one time but are used at different stages in the backend.

# server.R

```
# Coding for tab4 i.e. Volume of Business

      ggplot(data.m, aes(Var2,value)) +
      geom_bar(aes(fill=Gender), position="dodge", stat="identity")+
      labs(title="Customer_Average_usage")+
      labs(x="Age_Group",y=names(xyz[as.numeric(input$var)]))+
      theme_bw()

})
```

# server.R

```
observeEvent(input$tabs, {
   if(input$tabs=="data"){
     show("age")
     show("gender")
     hide("ID")
     hide("var")
   }})
observeEvent(input$tabs, {
   if(input$tabs=="summa"){
     show("age")
     show("gender")
     show("ID")
     hide("var")
   }})
observeEvent(input$tabs, {
    if(input$tabs=="Usag"){
      show("age")
      show("gender")
      show("var")
      hide("ID")
    }})
```

- **observeEvent()** is an event handler used for giving condition/s for a particular event to happen.
- Here we have given conditions for all the tabs

- **show()** makes the widgets for **age** and **gender** visible
- **hide()** makes **var** and **ID** invisible when **data** tab is selected.
- Likewise, we have given the conditions for other tabs

# server.R

```
observeEvent(input$tabs, {
   if(input$tabs=="vol"){
     show("var")
     hide("age")
     hide("gender")
     hide("ID")
}})
}
```

# Deploy the App

To upload the app:

- Go to http://www.shinyapps.io/ and create a free or professional account.
- Make sure all your app files are in an isolated folder saved in your working directory.
- In order to upload the app on shinyapp.io, you need to install package **devtools** and **rsconnect**.

```
# This R file is created for making the app live
```

```
library(rsconnect)
rsconnect::setAccountInfo(name=<name>, token=<token> secret=<token>)

rsconnect::deployApp("TelecomApplication")
```

- **deployApp()** function automatically syncs up to the shinyapps.io server, and opens the shinyapps.io website on your browser.
- Once your app is uploaded, it is on the internet and can be viewed by anybody with access to it.