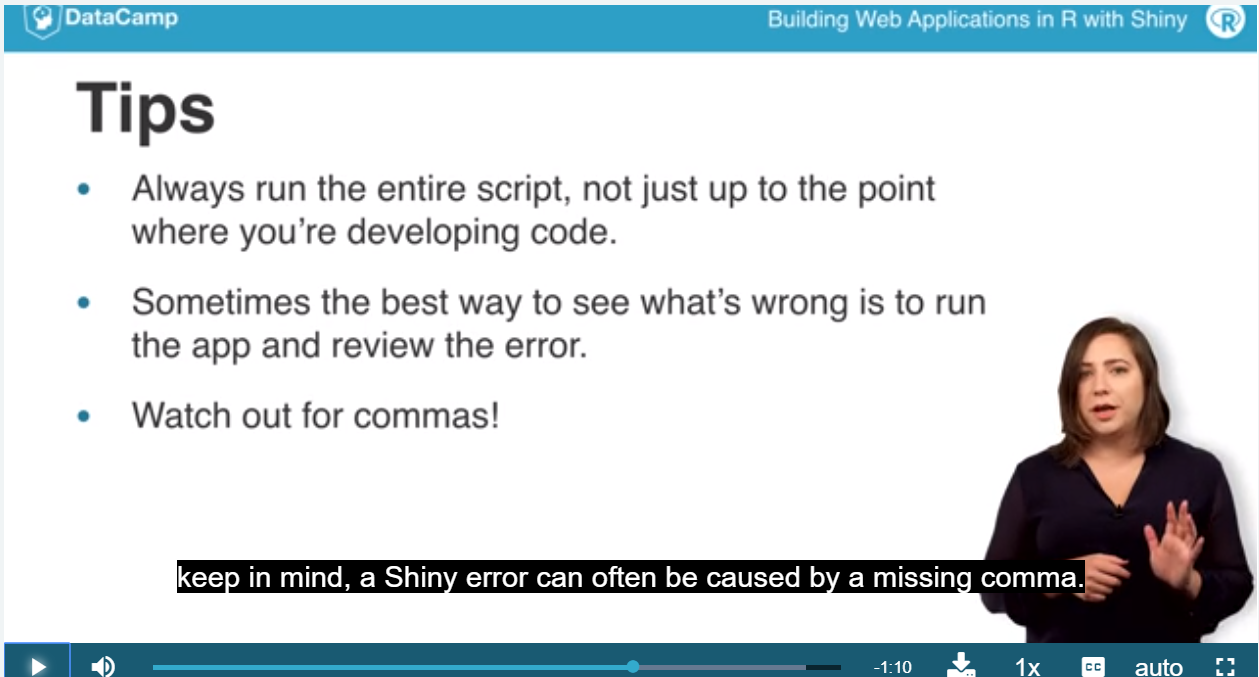
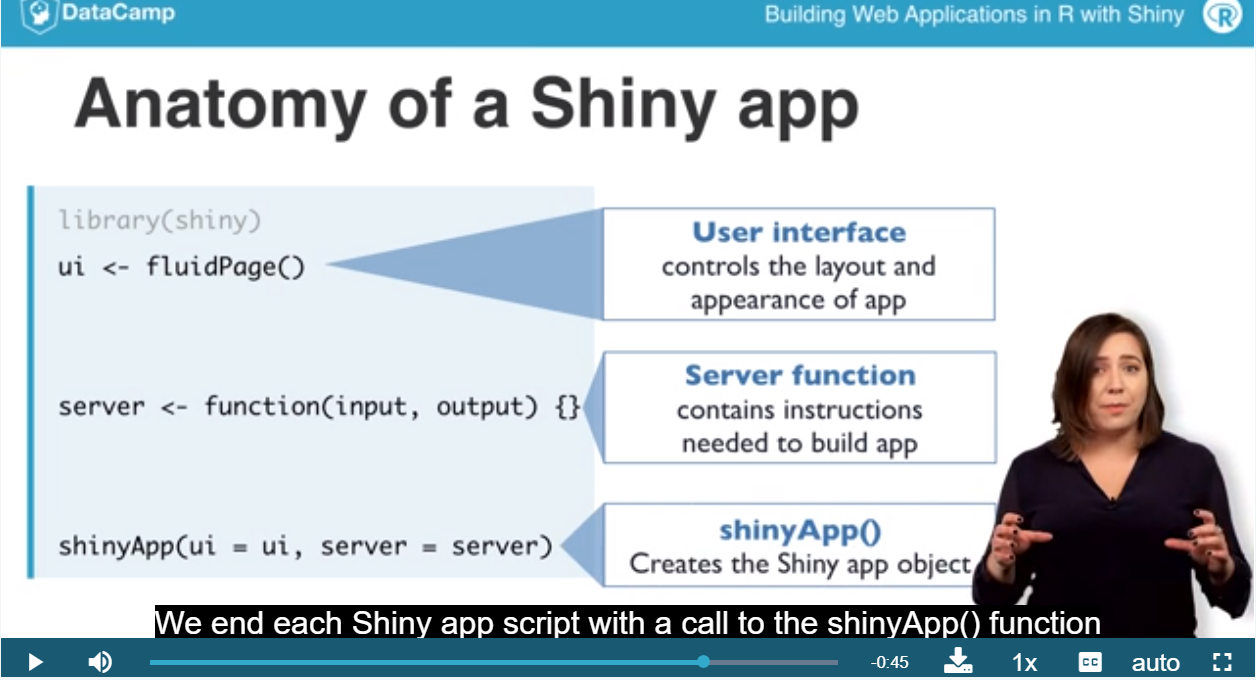
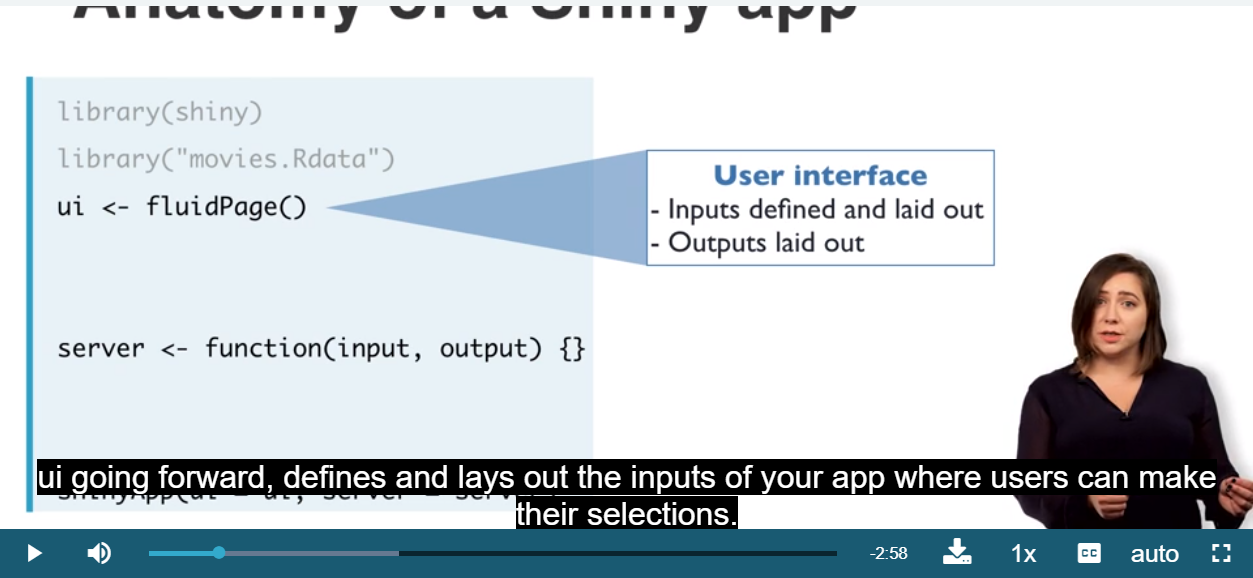
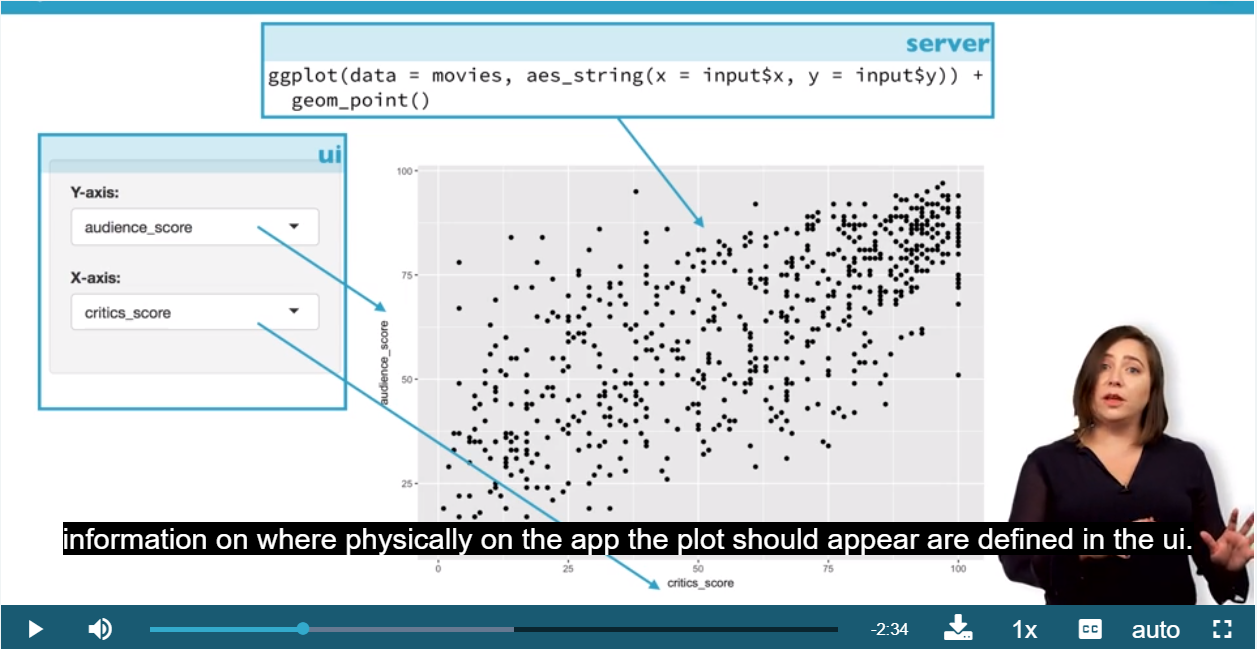
Shiny









* Ui:使用者可以操作的介面

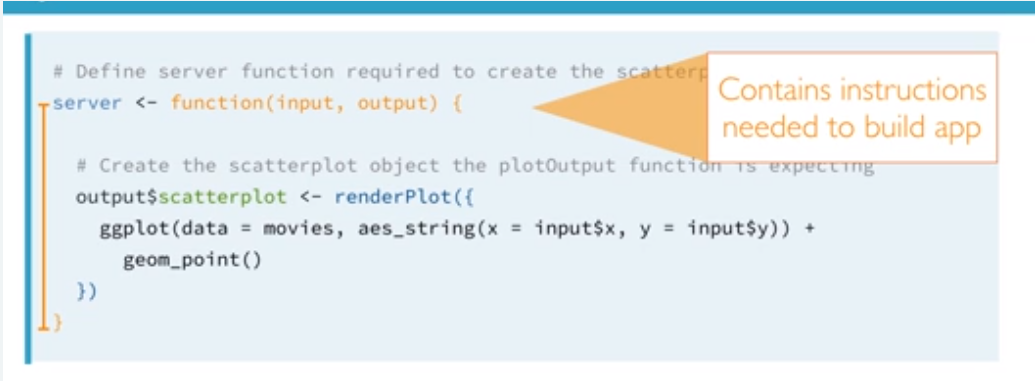
 In this app a selectInput widget is used to allow the user to select which variables should be plotted on the xand y axes of the scatterplot.

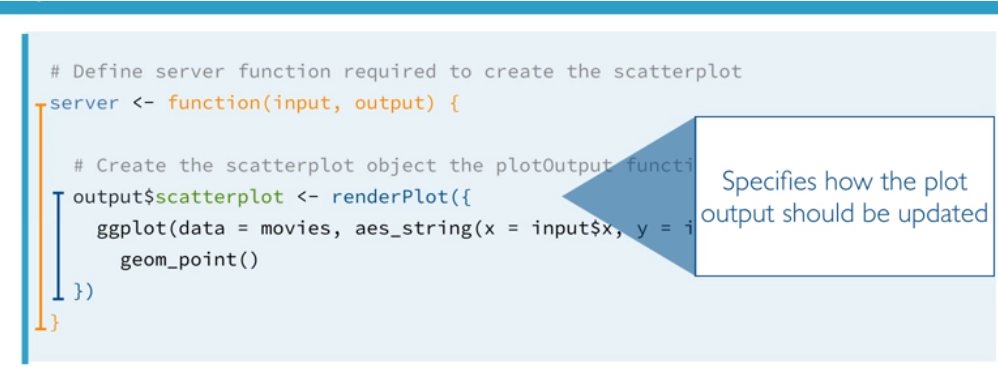
The selectInput function has four arguments: an inputId that is used to refer to the input parameter when building the scatterplot, a label that is displayed in the app, a list of choices to pick from, and a selected choice for when the app first launches. Note that choices takes a named vector, and the name rather than the value (which must match variable names in the data frame) is displayed to the user.

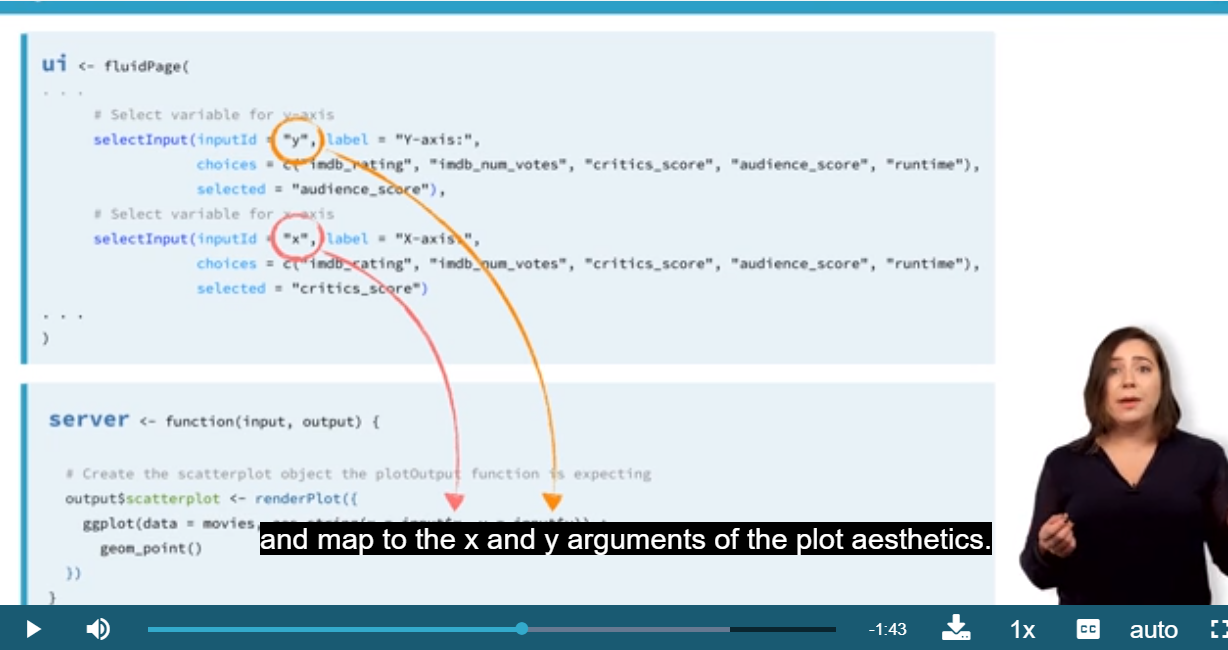
讓標籤可以跟變數本身名字不一樣。更易於閱讀

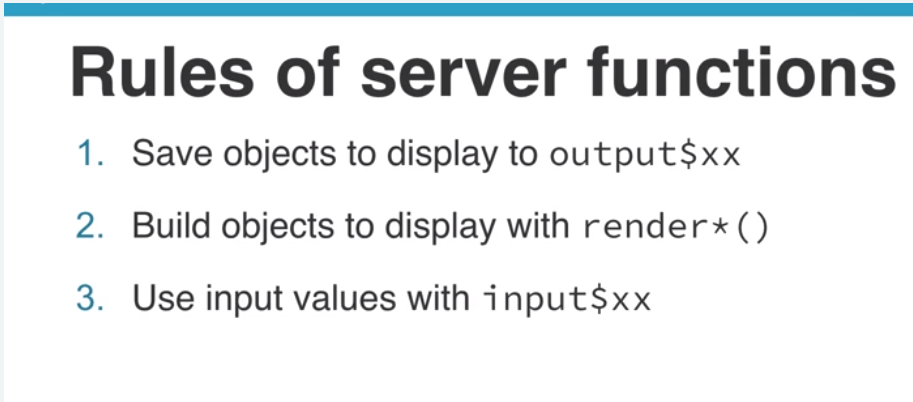
The potential variables the user can select for the xand y axes and color currently appear in the UI of the app the same way that they are spelled in the data frame header. However we might want to label them in a way that is more human readable. We can achieve this using named vectors for the choices argument, in the format of "Human readable label" = "variable\_name"

Server:背後操作的部分(使用者看不到)











library(shiny)

library(ggplot2)

load(url("http://s3.amazonaws.com/assets.datacamp.com/production/course\_4850/datasets/movies.Rdata"))

# Define UI for application that plots features of movies

ui <- fluidPage(

# Sidebar layout with a input and output definitions

sidebarLayout(

# Inputs

sidebarPanel(

# Select variable for y-axis

selectInput(inputId = "y",

label = "Y-axis:",

choices = c("IMDB rating" = "imdb\_rating",

"IMDB number of votes" = "imdb\_num\_votes",

"Critics score" = "critics\_score",

"Audience score" = "audience\_score",

"Runtime" = "runtime"),

selected = "audience\_score"),

# Select variable for x-axis

selectInput(inputId = "x",

label = "X-axis:",

choices = c("IMDB rating" = "imdb\_rating",

"IMDB number of votes" = "imdb\_num\_votes",

"Critics score" = "critics\_score",

"Audience score" = "audience\_score",

"Runtime" = "runtime"),

selected = "critics\_score"),

# Select variable for color

selectInput(inputId = "z",

label = "Color by:",

choices = c("Title type" = "title\_type",

"Genre" = "genre",

"MPAA rating" = "mpaa\_rating",

"Critics rating" = "critics\_rating",

"Audience rating" = "audience\_rating"),

selected = "mpaa\_rating")

),

# Outputs

mainPanel(

plotOutput(outputId = "scatterPlot")

)

)

)

# Define server function required to create the scatterplot

server <- function(input, output) {

# Create the scatterplot object the plotOutput function is expecting

output$scatterplot <- renderPlot({

ggplot(data = movies, aes\_string(x = input$x, y = input$y,

color = input$z)) +

geom\_point()

})

}

# Create a Shiny app object

shinyApp(ui = ui, server = server)

* Building a reactive widget

reactivity is established by linking an input with an output via a render\*() function.

如果要畫兩張圖

\*,height = 200限制高度

library(shiny)

library(ggplot2)

load(url("http://s3.amazonaws.com/assets.datacamp.com/production/course\_4850/datasets/movies.Rdata"))

# Define UI for application that plots features of movies

ui <- fluidPage(

# Sidebar layout with a input and output definitions

sidebarLayout(

# Inputs

sidebarPanel(

# Select variable for y-axis

selectInput(inputId = "y",

label = "Y-axis:",

choices = c("imdb\_rating", "imdb\_num\_votes", "critics\_score", "audience\_score", "runtime"),

selected = "audience\_score"),

# Select variable for x-axis

selectInput(inputId = "x",

label = "X-axis:",

choices = c("imdb\_rating", "imdb\_num\_votes", "critics\_score", "audience\_score", "runtime"),

selected = "critics\_score")

),

# Outputs

mainPanel(

plotOutput(outputId = "scatterplot"),

plotOutput(outputId = "densityplot",height = 200)

)

)

)

# Define server function required to create the scatterplot

server <- function(input, output) {

# Create scatterplot

output$scatterplot <- renderPlot({

ggplot(data = movies, aes\_string(x = input$x, y = input$y)) +

geom\_point()

})

# Create densityplot

output$densityplot <- renderPlot({

ggplot(data = movies, aes\_string(x = input$x)) +

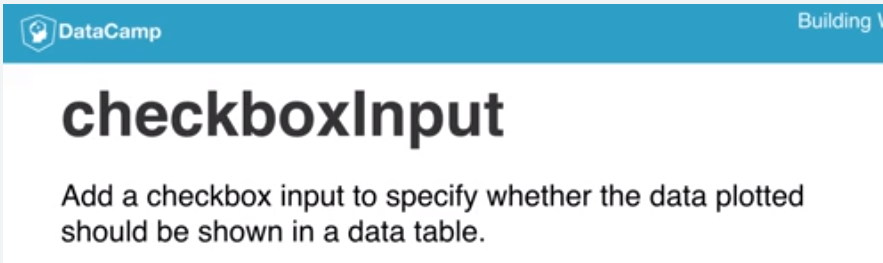
geom\_density()

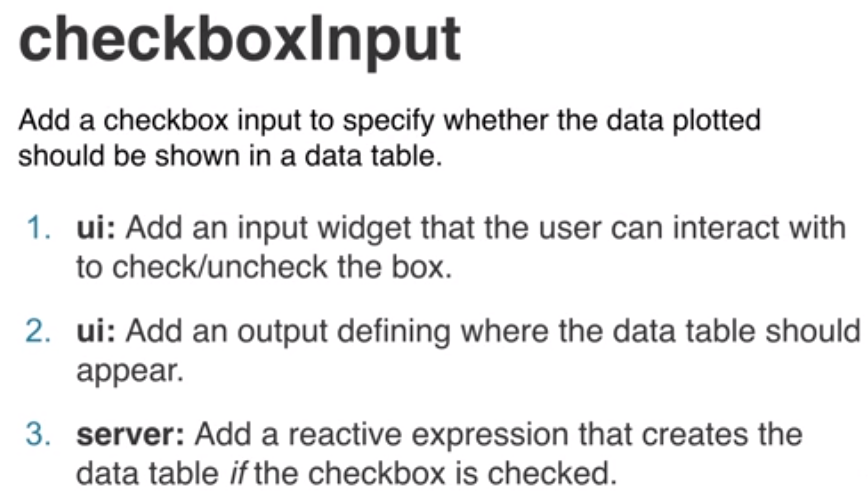
})

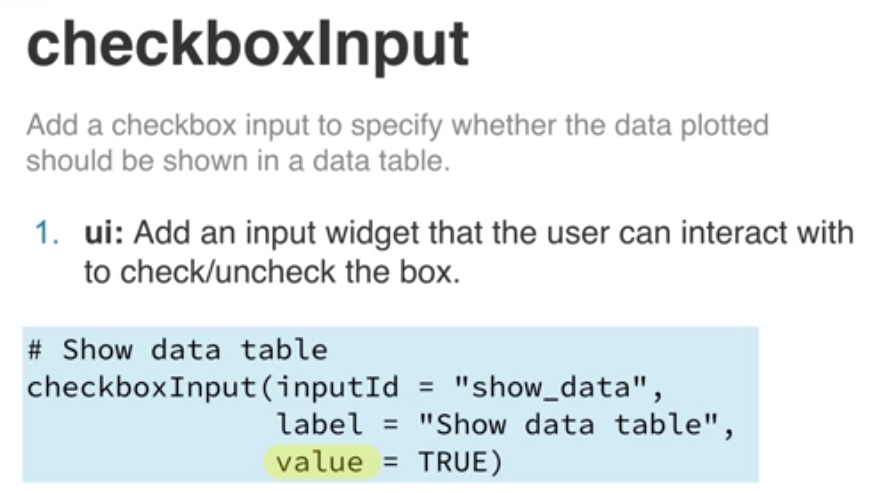
}

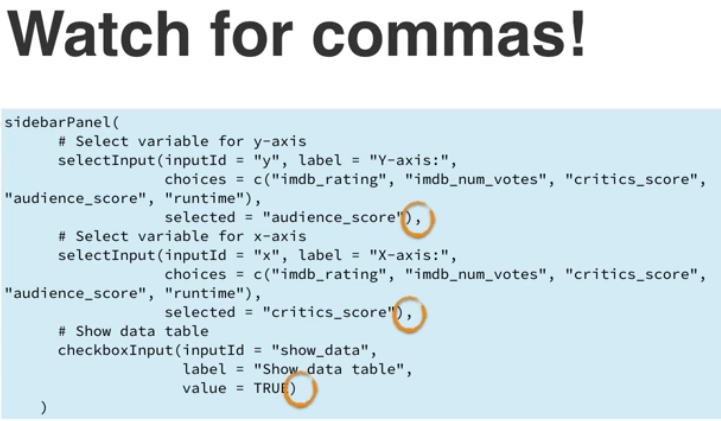
# Create the Shiny app object

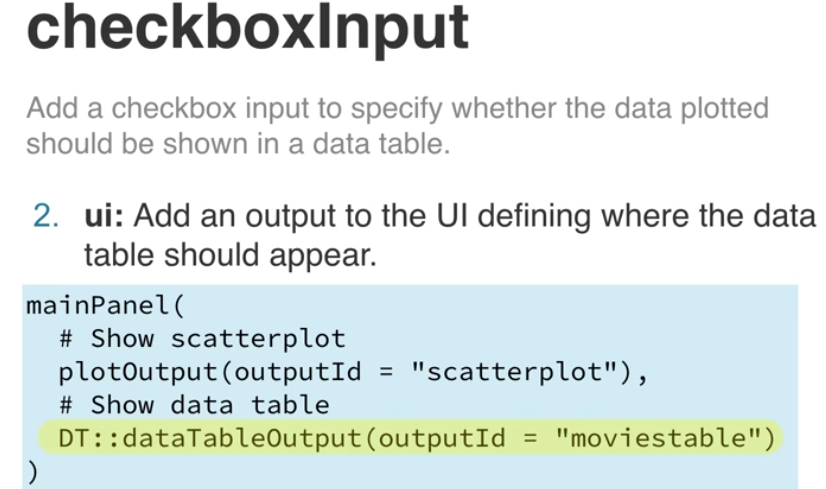
shinyApp(ui = ui, server = server)

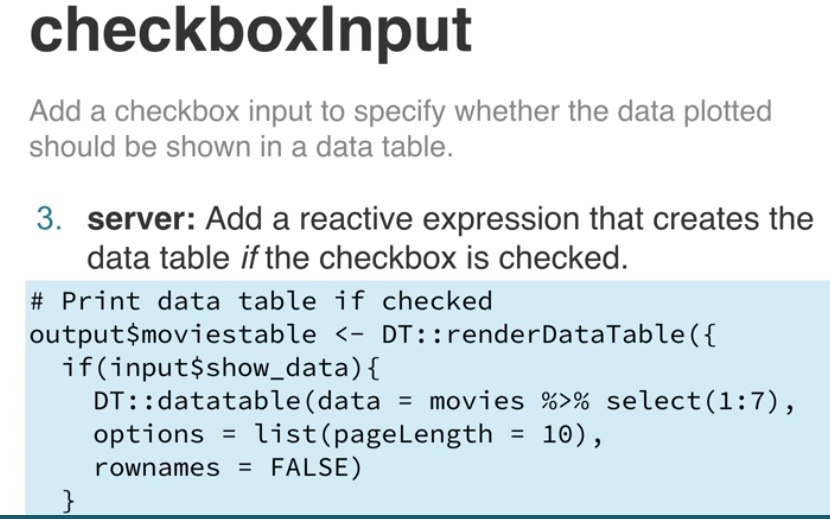


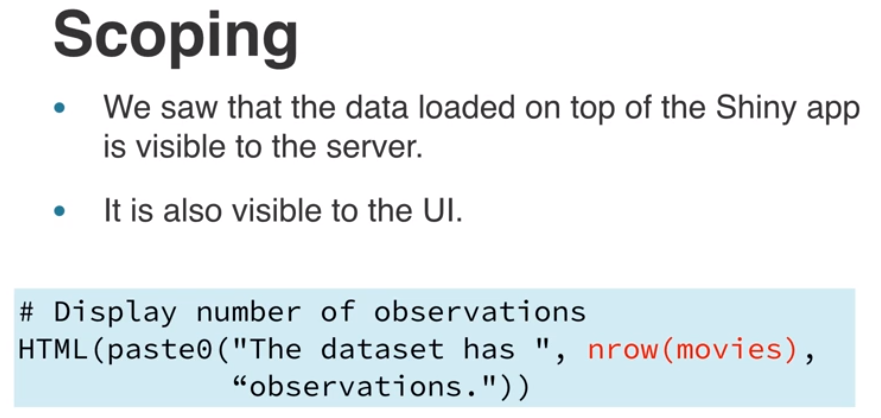












Checkboxinput的作用是顯示出符合選擇條件的數據(table)

**req, a.k.a. your best friend**

The app on the right is the one you developed in the previous exercise. Run the code using the "Run Code" button. Then, in the Shiny app (not in the code), delete the numeric value (30) in the sample size box. You will encounter an error: Error: size is not a numeric or integer vector.

In order to avoid such errors, which users of your app could very easily encounter, we need to hold back the output from being calculated if the input is missing. The req function is the simplest and best way to do this, it ensures that values are available ("**[truthy](https://shiny.rstudio.com/reference/shiny/latest/req.html" \t "_blank)**") before proceeding with a calculation or action. If any of the given values is not truthy, the operation is stopped by raising a "silent" exception (neither logged by Shiny, nor displayed in the Shiny app's UI).

練習

View the help function for the selectInput widget by typing ?selectInput in the console, and figure out how to enable the selectize and multiple selection options (or whether they are enabled by default).

In the UI: Based on your findings add the necessary arguments to the selectInput widget.

In the server: Add a call to the req function, just like you did in the previous exercise, but this time requiring that input$studio be available. Update the call to filter()to use the logical operator %in% (instead of ==) so that it works when multiple studios are selected.

Run the app and (1) confirm that you can select multiple studios, (2) start typing "Warner Bros" to confirm selectize works, and (3) delete all selections to confirm req is preventing an error from being displayed when no studio input is provided.

library(shiny)

library(ggplot2)

library(dplyr)

library(DT)

?selectInput

load(url("http://s3.amazonaws.com/assets.datacamp.com/production/course\_4850/datasets/movies.Rdata"))

all\_studios <- sort(unique(movies$studio))

# UI

ui <- fluidPage(

sidebarLayout(

# Input(s)

sidebarPanel(

selectInput(inputId = "studio",

label = "Select studio:",

choices = all\_studios,

selected = "20th Century Fox",

multiple=TRUE)可以一次選很多

),

# Output(s)

mainPanel(

DT::dataTableOutput(outputId = "moviestable")

)

)

)

# Server

server <- function(input, output) {

# Create data table

output$moviestable <- DT::renderDataTable({

req(input$studio)

movies\_from\_selected\_studios <- movies %>%

filter(studio %in% input$studio) %>%

select(title:studio)

DT::datatable(data = movies\_from\_selected\_studios,

options = list(pageLength = 10),

rownames = FALSE)

})

}

# Create a Shiny app object

shinyApp(ui = ui, server = server)

* dateInput :選擇時間

  dateRangeInput:選擇時間範圍

# Date input

dateRangeInput(inputId = "date",

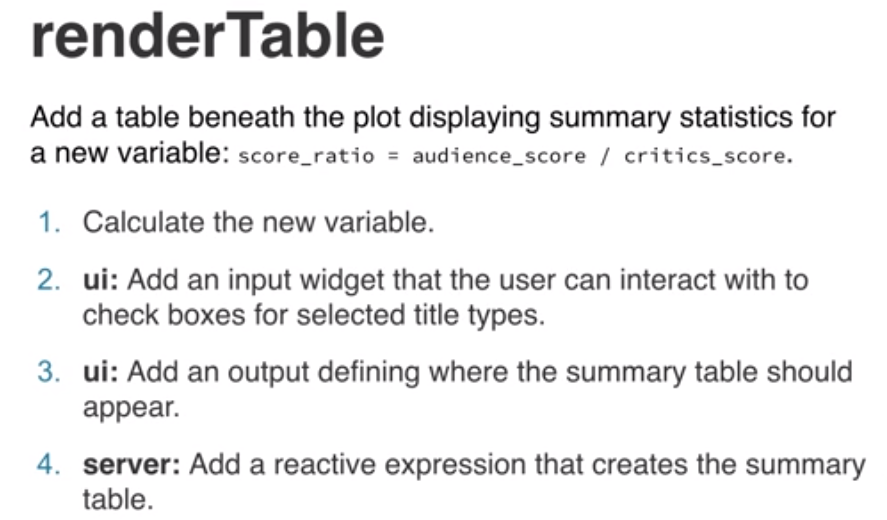
label = "Select dates:",

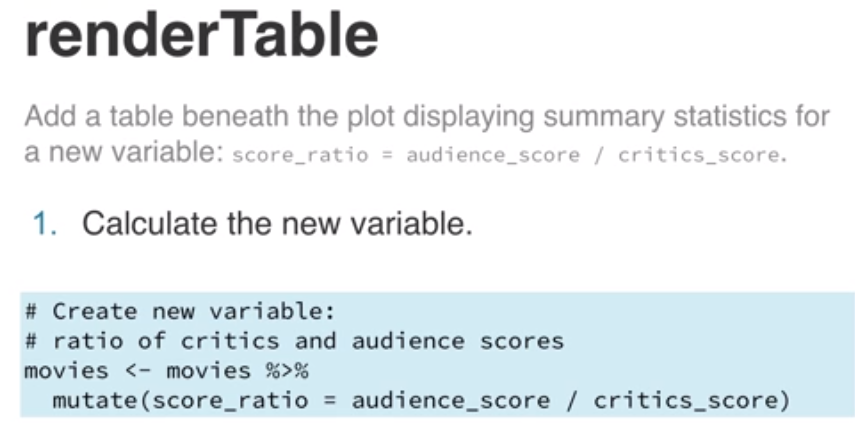
start = "2013-01-01",

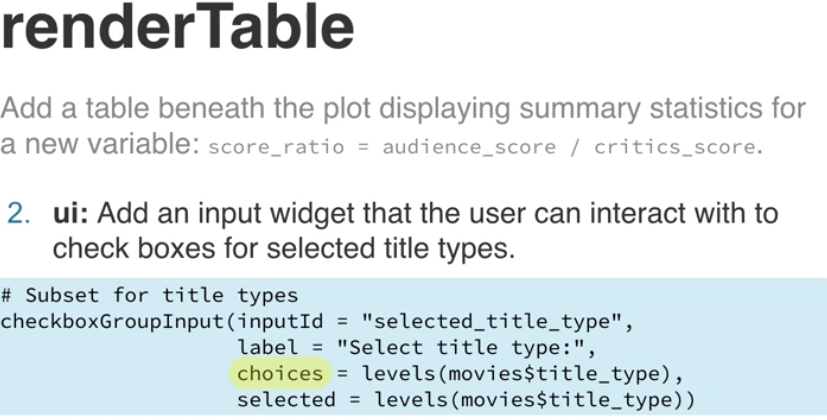
end = "2014-01-01",

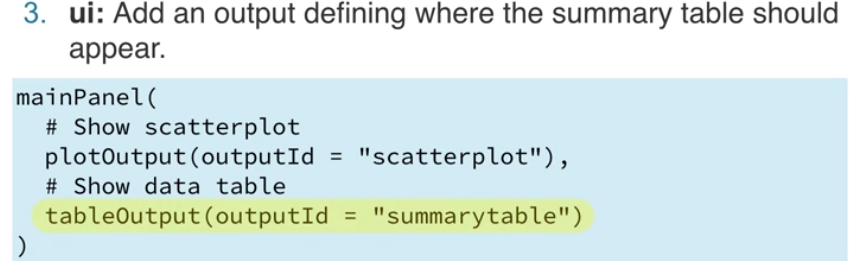
min = min\_date, max = max\_date,

startview="year")













意思就是用Rander開頭的東西要加{()}

* 若想加入文字，可以用paste0()

library(shiny)

library(ggplot2)

load(url("http://s3.amazonaws.com/assets.datacamp.com/production/course\_4850/datasets/movies.Rdata"))

# UI

ui <- fluidPage(

sidebarLayout(

# Input(s)

sidebarPanel(

# Select variable for y-axis

selectInput(inputId = "y",

label = "Y-axis:",

choices = c("imdb\_rating", "imdb\_num\_votes", "critics\_score", "audience\_score", "runtime"),

selected = "audience\_score"),

# Select variable for x-axis

selectInput(inputId = "x",

label = "X-axis:",

choices = c("imdb\_rating", "imdb\_num\_votes", "critics\_score", "audience\_score", "runtime"),

selected = "critics\_score")

),

# Outputs

mainPanel(

plotOutput(outputId = "scatterplot"),

textOutput(outputId = "correlation")

)

)

)

# Server

server <- function(input, output) {

# Create scatterplot object the plotOutput function is expecting

output$scatterplot <- renderPlot({

ggplot(data = movies, aes\_string(x = input$x, y = input$y)) +

geom\_point()

})

# Create text output stating the correlation between the two ploted

output$correlation <- renderText({

r <- round(cor(movies[, input$x], movies[, input$y], use = "pairwise"), 3)

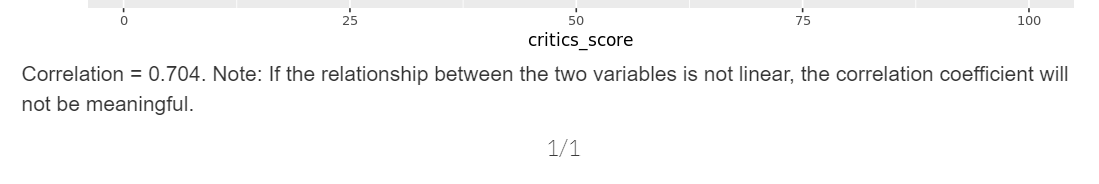
paste0("Correlation = ", r, ". Note: If the relationship between the two variables is not linear, the correlation coefficient will not be meaningful.")

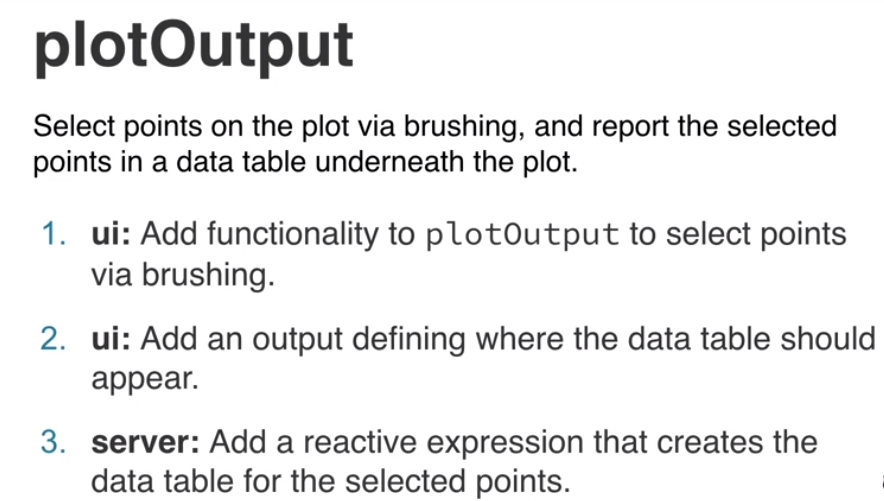
})

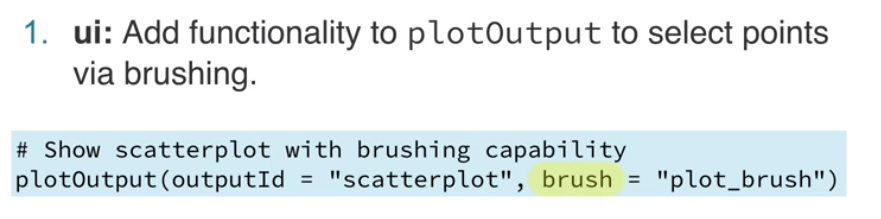
}

# Create a Shiny app object

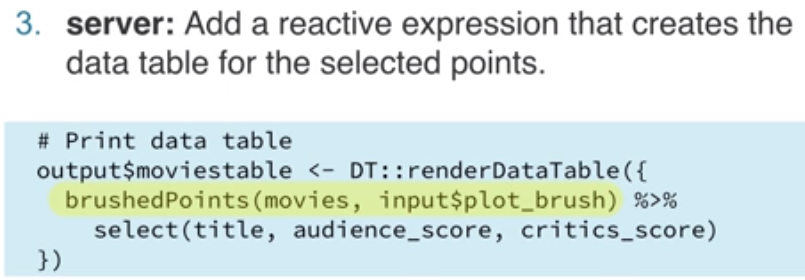
shinyApp(ui = ui, server = server)











<http://shiny.rstudio.com/articles/selecting-rows-of-data.html>

(burush的介紹，簡單來說bru就是選取到那個點就顯示那點的資料)

用hover的:

# Load packages

library(shiny)

library(ggplot2)

library(tidyverse)

library(DT)

# Load data

load(url("http://s3.amazonaws.com/assets.datacamp.com/production/course\_4850/datasets/movies.Rdata"))

# Define UI for application that plots features of movies

ui <- fluidPage(

br(),

# Sidebar layout with a input and output definitions

sidebarLayout(

# Inputs

sidebarPanel(

# Select variable for y-axis

selectInput(inputId = "y", label = "Y-axis:",

choices = c("imdb\_rating", "imdb\_num\_votes", "critics\_score", "audience\_score", "runtime"),

selected = "audience\_score"),

# Select variable for x-axis

selectInput(inputId = "x", label = "X-axis:",

choices = c("imdb\_rating", "imdb\_num\_votes", "critics\_score", "audience\_score", "runtime"),

selected = "critics\_score")

),

# Output:

mainPanel(

# Show scatterplot with brushing capability

plotOutput(outputId = "scatterplot", hover = "plot\_hover"),

# Show data table

dataTableOutput(outputId = "moviestable"),

br()

)

)

)

# Define server function required to create the scatterplot

server <- function(input, output) {

# Create scatterplot object the plotOutput function is expecting

output$scatterplot <- renderPlot({

ggplot(data = movies, aes\_string(x = input$x, y = input$y)) +

geom\_point()

})

# Create data table

output$moviestable <- DT::renderDataTable({

nearPoints(movies, input$plot\_hover) %>%

select(title, audience\_score, critics\_score)

})

}

# Create a Shiny app object

shinyApp(ui = ui, server = server)

* textOutput: 在應用程序頁面中將反應性輸出變量渲染為文本。 默認情況下，文本將包含在HTML div標記中。
* verbatimTextOutput: 在應用程序頁面中將反應性輸出變量渲染為逐字文本。 該文本將包含在HTML預標記中。

以上折線圖呈現了各區Ubike站車輛數隨著時間變化的折線圖。其中每天凌晨1點到6點的部分是缺少紀錄的。

從各區的折線圖，能推測出每區ubike使用熱度，甚至是人潮流動的狀況與特徵。

大安區:平常日每天大致上有三次的波動。第一次波動在早上上班時間，每天都呈現巨大的落差，車輛急遽減少，推測可能是因為居民在這段時間通勤至此區上班。而到了下午三點左右，空位數有一波增加趨勢，隨後又減少，接著在下班時間，約七點左右，又有一波明顯的增加，並在九點之後又減少，推測可能是上班族騎著Ubike回家了，造成短時間的車輛增加。到了假日，雖然在早上仍有車輛減少的情況，但也許是因為假日休息的緣故，落差不如平常日巨大。

士林區、中山區、信義區、松山區:平常日從早上開始呈現下降的趨勢，直到下午都是起起伏伏的狀態，然而到了大約七點左右的下班時間，車輛數目會急遽下降，顯示晚間的ubike熱度明顯上升，此區的人們使用ubike的活動增加，或者是人潮開始向外移動。到了假日，早上呈現的不再是緩降的趨勢，反而是一直保持在相對高點，並有些起伏。推測假日早上ubike的使用率不如平常日高。此外，信義區平均車輛數明顯高於其他兩區。

(大同與士林的差別是士林區假日用量較明顯的減少，車位數比較多)

大同區、中正區:假日跟平常日的趨勢比較相像。在早上接近中午的時候會有一小段時間腳踏車數短暫下降，之後又回升，推測也許是午餐時間到了，外出騎乘腳踏車用餐，導致腳踏車數微幅下降後又回升。從早上開始站點大致上呈現車輛數下降的趨勢，尤其到了下班時間，數目急遽下降，顯示此時是ubike的使用高峰期。大同區的總車數比較少，中正區則比較多。

內湖區、南港區:假日與工作日ubike平均數落差明顯。內湖區平常日的腳踏車平均每個站點不超過13台，然而假日時，平均卻可高達17輛，而且不低於12輛。顯示內湖是屬於上班族的領域，平常通勤頻繁，使用率高，到了假日使用熱度便會冷卻。與其他區一樣，每天晚上9點10點左右，車輛數來到一日最低點，之後回升。而南港平日假日雖然落差沒有內湖這麼大，但依然可以看出假日的車輛數比平常日多。此外，南港與內湖早上時段車輛數有一點傾向成反比的狀況。

文山區:可以看出工作日的使用熱度有明顯別於假日。在工作日中，每天接近中午會有個凹陷，顯示這段時間車輛數有下降再回升的情形。而不管是工作日還是假日，一天中車輛數最低點同樣出現在晚上9點10點之間。在假日中，白天使用ubike的熱度較低，因此車輛數維持在高處，導致與晚上的落差非常巨大。

北投區、萬華區:相較其他區變化比較規律，推測使用率不高造成。直到中午才出現第一次回彈，並在下午時段車輛數也有一次小小下降後回升的狀況，直到夜晚跟其他地區一樣，車輛數來到一日中的最低點。北投與萬華區差別在於萬華區的車輛變化量較大。

總體來說，台北市的ubike使用狀況大致上是從早是開始趨於頻繁，有些地區在中午時也會出現一波