

# Using R to View Data

A demonstration by Jacob Hurley  
<https://github.com/jih5437/rDemo>



## Preview:

### Data Viewer

Select a dataset:

genre

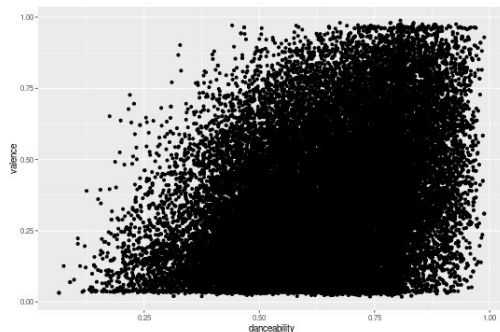
Select x variable:

danceability

Select y variable:

valence

danceability	energy	key	loudness	mode	speechiness	acousticness	instrumentalness
0.83	0.81	2	-7.36	1	0.42	0.06	0.00
0.72	0.49	8	-7.23	1	0.08	0.40	0.00
0.85	0.89	5	-4.78	1	0.06	0.01	0.00
0.48	0.78	0	-4.71	1	0.10	0.02	0.00
0.80	0.62	2	-7.67	1	0.29	0.22	0.00
0.72	0.57	0	-11.29	1	0.41	0.05	0.00



R Studio Data Viewer

<https://posit.cloud/content/5793375>

# Abstract

## Intro:

- What is R?

## Setup:

- Accessing RStudio

## Starting our first Project:

- How to make a project
- Choosing file format
- R Packages

## The R workspace:

- Becoming familiar with R
- Loading Data
- Viewing Data
- Making Graphs
- Additional Libraries

## Making a Simple R Application:

- R Script Setup
- Shiny Library
- User Interfaces
- Server Setup
- Running the application

## Additional Functionality

- APIs + More

## Conclusion:

- Final statements
- Questions & Open Discussion



# Intro



R Programming

R is a powerful statistical computing language that can help you analyze and visualize data

In this presentation, we will cover how to:

- Load data into R
- View data in R
- Make graphs with the data
- Use additional libraries to extend R's capabilities
- Build an example of an application using R

spotifyr 2.1.1



ggplot2



# Setup

In this demonstration we will be utilizing Posit Cloud's free to use RStudio cloud environment

This allows for many users who may have hardware limitations to program via the cloud

'Posit Cloud' also comes with everything we need for today's demonstration

Alternatively, you can also use the desktop version of RStudio with R installed

Let's open up RStudio/Posit Cloud and get started!

POSIT CLOUD

## Friction free data science

Posit Cloud lets you access Posit's powerful set of data science tools right in your browser—no installation or complex configuration required.

GET STARTED

ALREADY A USER? LOG IN



<https://posit.co/products/cloud/cloud/>

DOWNLOAD

## RStudio Desktop

Used by millions of people weekly, the RStudio integrated development environment (IDE) is a set of tools built to help you be more productive with R and Python.

### 1: Install R

RStudio requires R 3.3.0+. Choose a version of R that matches your computer's operating system.

DOWNLOAD AND INSTALL R

### 2: Install RStudio

DOWNLOAD RSTUDIO DESKTOP FOR WINDOWS

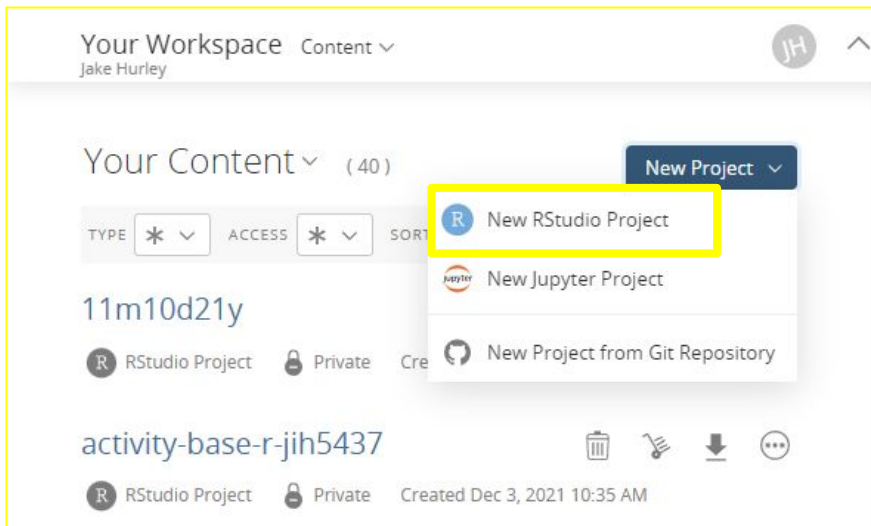
Size: 208.08 MB | SHA-256: 885432DB | Version: 2023.03.0+386 | Released: 2023-03-16

<https://posit.co/download/rstudio-desktop/>

# Starting our first Project



# Create a New Project



- We first select the “New Project” dropdown menu found in the top right of our window and then select “New RStudio Project”



- Once the page has loaded, save and name your project found in the top left of the window



# Choosing file format

When we first load into our project we are greeted with a blank workspace

First we will create a file to code in, an R Notebook

A R notebook is a user friendly way to program in R

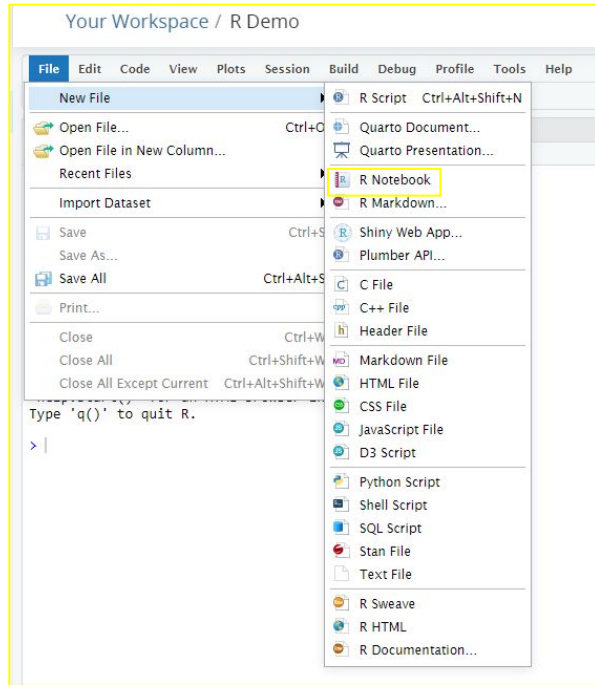
Code can be run in cell blocks and checked step-by-step

This allows new users of R to view data and compare different data frames in one document and export in an html (web based) format

- Select “File”, “New File” and choose “R Notebook”

Another way of coding in R is coding in “R Script”

R Script can run applications using packages like the ‘shiny’ library to create interactive applications with an user interface (UI)





# R Packages

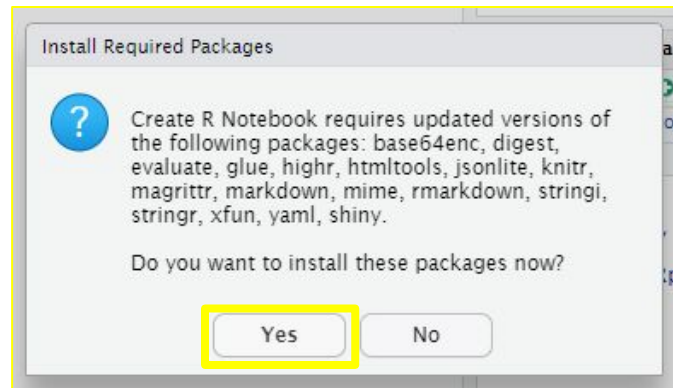
What are packages?

Packages are ways to extend capabilities of the R statistical programming language

Packages consist of code, data, and documentation that follow a standard collection format

- If prompted with "Install Required Packages", select Yes

	Name	Description	Version
<b>System Library</b>			
<input type="checkbox"/>	askpass	Safe Password Entry for R, Git, and SSH	1.1
<input type="checkbox"/>	assertthat	Easy Pre and Post Assertions	0.2.1
<input checked="" type="checkbox"/>	base	The R Base Package	4.2.3
<input type="checkbox"/>	base64enc	Tools for base64 encoding	0.1-3





# The R Studio Workspace



Your Workspace / R Demo

The R Workspace

RAM

Settings

More

JH Jake Hurley

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function

Addins

rDemo.Rmd

Source Visual

Outline

1 ---  
2 title: "R Demo"  
3 author: "Jacob Hurley"  
4 output: html\_notebook  
5 ---  
6  
7 {r}  
8 #CODE GOES HERE  
9 ---  
10

A. The coding workspace

- {language}  
<code>

8:16 Chunk 1

R Markdown

Console Terminal Background Jobs

R 4.2.3 . /cloud/project/

R version 4.2.3 (2023-03-15) -- "Shortstop Beagle"  
Copyright (C) 2023 The R Foundation for Statistical Computing  
Platform: x86\_64-pc-linux-gnu (64-bit)  
  
R is free software and comes with ABSOLUTELY NO WARRANTY.  
You are welcome to redistribute it under certain conditions.  
Type 'license()' or 'licence()' for distribution details.  
  
R is a collaborative project with many contributors.  
Type 'contributors()' for more information and  
'citation()' on how to cite R or R packages in publications.  
  
Type 'demo()' for some demos, 'help()' for on-line help, or  
'help.start()' for an HTML browser interface to help.  
Type 'q()' to quit R.  
  
> |

Environment History Connections Tutorial

R Global Environment

B. Location of Data Frames

Environment is empty

Files Plots Packages Help Viewer Presentation

Cloud project

	Name	Size	Modified
	..		
	.Rhistory	0 B	Apr 12, 2023, 12:08 PM
	project.Rproj	205 B	Apr 12, 2023, 12:09 PM
	rDemo.Rmd	97 B	Apr 12, 2023, 12:15 PM
	rDemo.nb.html	648.3 KB	Apr 12, 2023, 12:15 PM

D. File Directory



# Gathering Data

Loading data into RStudio is a simple process

Once we have obtained data we can begin the process of loading in our data

Data can be obtained from anywhere or collected and imported as long as it is a valid format

For this demonstration we will be utilizing a data set found from Kaggle's free to use and open source database which is in a CSV format (comma-separated values)

The dataset published by Andrii Samoshyn contains songs with their genres taken from Spotify's website.

Each song has multiple features and attributes assigned via Spotify's API



ANDRII SAMOSHYN · UPDATED  
2 YEARS AGO



New Notebook

Download (3 MB)



## Dataset of songs in Spotify

The full list of genres



Data Card

Code (40)

Discussion (6)

### About Dataset

The full list of genres included in the CSV are Trap, Techno, Techhouse, Trance, Psytrance, Dark Trap, DnB (drums and bass), Hardstyle, Underground Rap, Trap Metal, Emo, Rap, RnB, Pop and Hip-hop.

Usability ⓘ

10.00

License

CC0: Public Domain

Expected update frequency

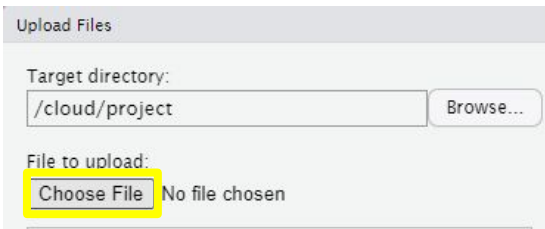
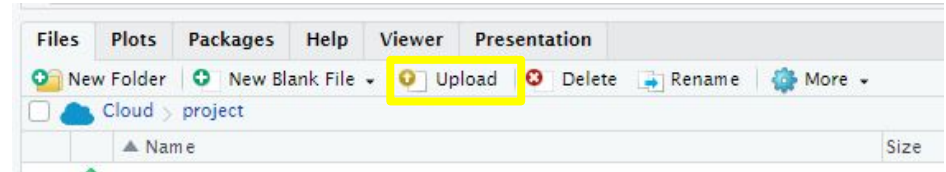
Annually

<https://www.kaggle.com/datasets/mrmorj/dataset-of-songs-in-spotify>

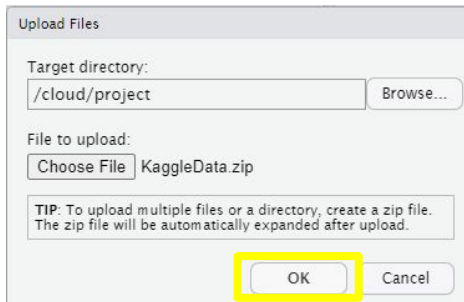
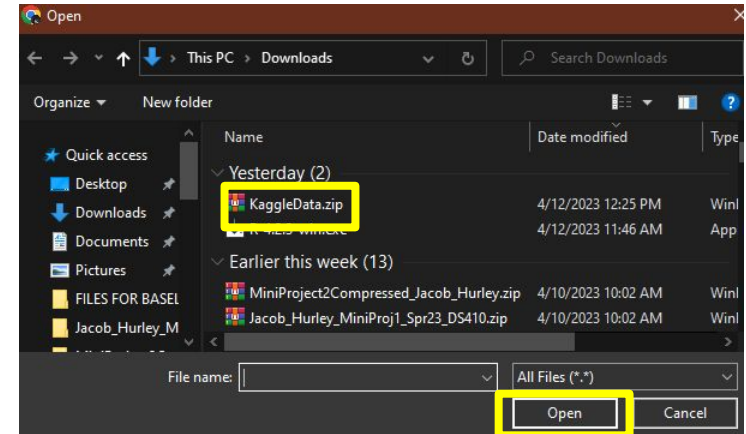
# Uploading our Data



After downloading our dataset, we can then upload our data by selecting the upload button found in the bottom left window



When prompted, select 'Choose File' and select the file to upload to the project folder



Press 'OK'

# Loading Data Into the R Environment

We can enter our upload files into the R Environment by typing the following into the code block : **dataFrameName <- read.csv("filepath/file.csv", header = TRUE)**

```
#Loading in our data
Kaggle Data is used for our initial demonstration -> https://www.kaggle.com/datasets/mrmorj/dataset-of-songs-in-spotify

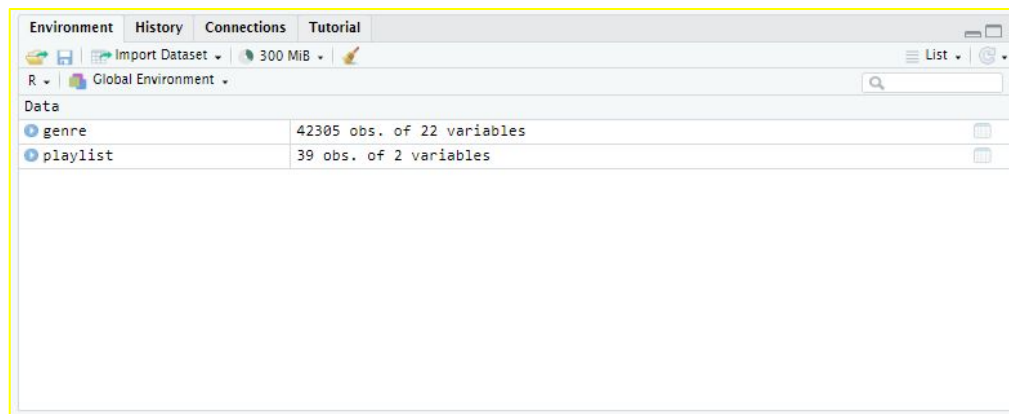
Upload these files to the main project folder and call them here:
```{r}
#Loading our data into the R Global Environment
genre <- read.csv("genres_v2.csv", header = TRUE)
playlist <- read.csv("playlists.csv", header = TRUE)
```
```

Coding Notes:

'<-' : Is the "**Assignment Operator**", to assign values to the variables.

'header = TRUE' - the first row is used for the names of the columns

Run code chunks by pressing the arrow here



The screenshot shows the RStudio Environment pane with the following data frames loaded:

| Environment | History                    | Connections | Tutorial |
|-------------|----------------------------|-------------|----------|
| R           | Global Environment         |             |          |
| Data        |                            |             |          |
| genre       | 42305 obs. of 22 variables |             |          |
| playlist    | 39 obs. of 2 variables     |             |          |

When loaded in, we will see our dataframes with the names given in the environment window on the right hand side of the workspace



# Viewing Our Data

Once you have loaded your data into R, you can view it using a variety of functions

Here are some examples of functions that can be used to view data in R:

```
## {r}
head(genre)
```

Description: df [6 x 22]

|   | danceability | energy |
|---|--------------|--------|
| 1 | 0.831        | 0.814  |
| 2 | 0.719        | 0.493  |

```
## {r}
tail(genre)
```

Description: df [6 x 22]

|       |       |       |
|-------|-------|-------|
| 42300 | 0.423 | 0.904 |
| 42301 | 0.528 | 0.693 |

```
## {r}
str(genre)
```

'data.frame': 42305 obs. of 22 variables:

```
$ danceability : num 0.831 0.719 0.85 0.
$ energy       : num 0.814 0.493 0.893 0.
$ key          : int 2 8 5 0 2 0 8 8 1 1
$ loudness     : num -7.36 -7.23 -4.78 -
$ mode        : int 1 1 1 1 1 1 1 1 1 1
$ speechiness : num 0.42 0.3704 0.0627
```

```
## {r}
summary(genre)
```

|          | danceability |
|----------|--------------|
| Min.     | :0.0651      |
| 1st Qu.: | 0.5240       |
| Median : | 0.6460       |
| Mean :   | 0.6394       |
| 3rd Qu.: | 0.7660       |
| Max.     | :0.9880      |

In this example, we use the head(), tail(), str(), and summary() functions to view the data in different ways.

The head() and tail() functions show the first and last few rows of the data, respectively.

The str() function shows the structure of the data, including the variable names and data types.

The summary() function shows summary statistics for the data, including the minimum, maximum, and median values.

# Making Basic R Graphs

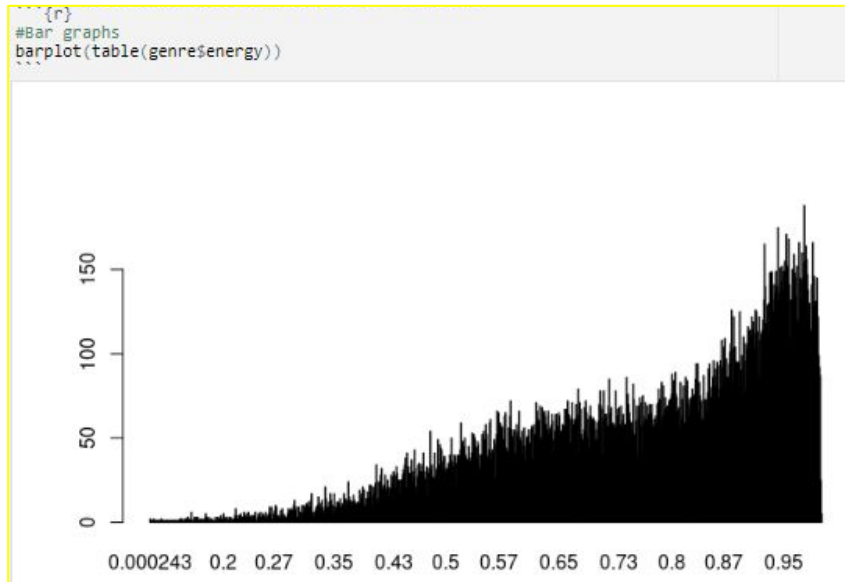
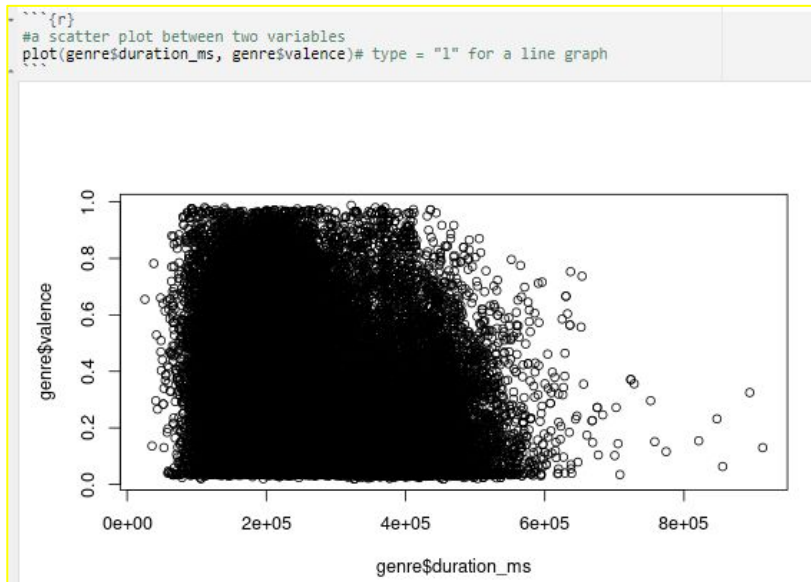
Coding Notes:

`$` - is an operator in R is used to access specific columns or variables in a data frame.

Once you have loaded and confirmed your data is in R, you can now make graphs to visualize it

R has a variety of built-in functions for creating graphs, including bar graphs, line graphs, and scatter plots

Here are some examples of functions that can be used to make graphs in R:



In this example, we use the `plot()` and `barplot()` functions to create different types of graphs.

The `barplot()` function creates a bar graph of a categorical variable, the `plot()` function creates a scatterplot of two variables.



# Additional Libraries

While R has many built-in functions for viewing and graphing data, there are also additional libraries that can extend R's capabilities

These libraries can be downloaded and installed using the `install.packages()` function in R console

Here are some examples of additional libraries that can be used to enhance R's functionality:

- Install the `ggplot2` library for advanced plotting
  - `install.packages("ggplot2")`
    - Load Via: `library(ggplot2)`
- Install the `dplyr` library for data manipulation
  - `install.packages("dplyr")`
    - Load Via: `library(dplyr)`
- Install the `tidyr` library for data tidying
  - `install.packages("tidyr")`
    - Load Via: `library(tidyr)`

```
#Install library (enter into console)
```{r}
#install.packages('ggplot2')
#install.packages('dplyr')
#install.packages('tidyr')
```

#Load our library|
```{r}
library(ggplot2)
library(dplyr)
library(tidyr)
```
```

In this example, we use the `install.packages()` function to download and install the `ggplot2`, `dplyr`, and `tidyr` libraries. These libraries can be used for advanced plotting, data manipulation, and data tidying, respectively.

Information on commands and more to be used with these libraries can be done by using the command `help(library_name)`



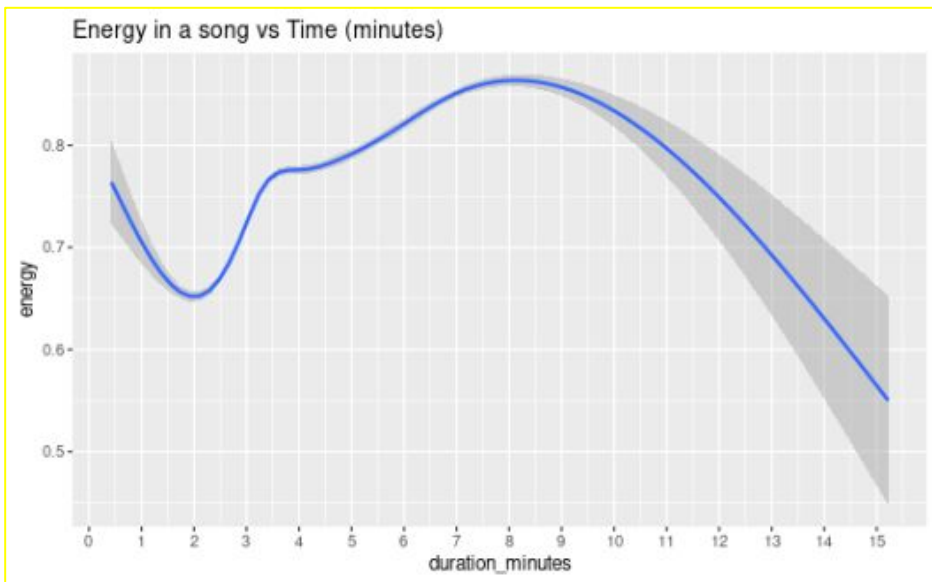
# Graphics with Libraries

Using additional libraries, we can extend the capabilities of the programming language R to give us more advanced graphs

Here in this demonstration we can see how dplyr and ggplot are used to enhance the functionality of R

With the addition help of these new libraries, we can create advanced graphs that we were not able to create before

```
{r}
genre %>% #dplyr command to load in data aka 'piping'
  mutate(duration_minutes = duration_ms/60000) %>% #more piping
  ggplot(aes(x = duration_minutes, y = energy)) +
  ggtitle("Energy in a song vs Time (minutes)") +
  geom_smooth() +
  scale_x_continuous(breaks=seq(0,15,1))
```





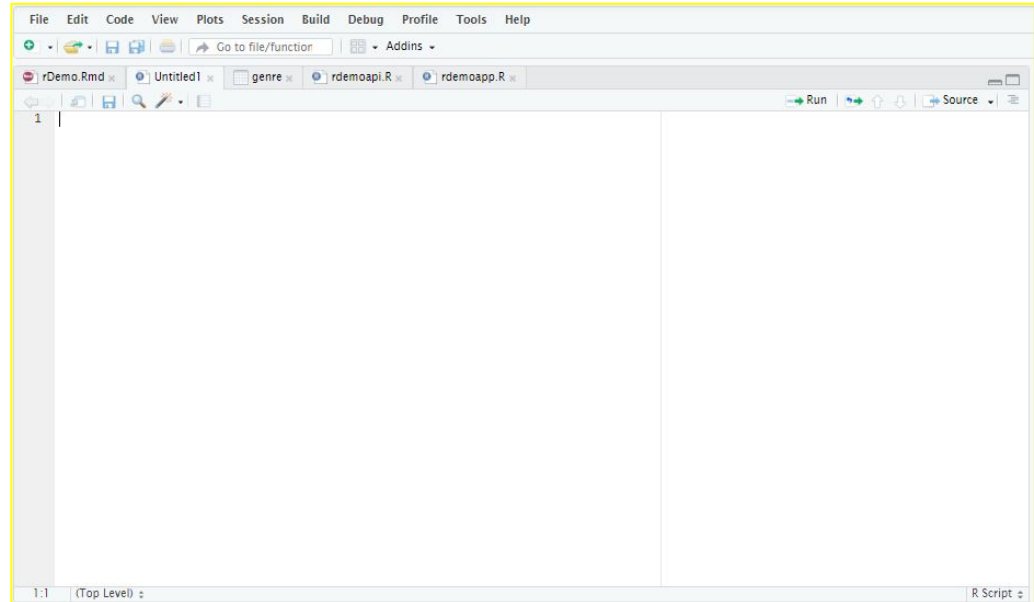
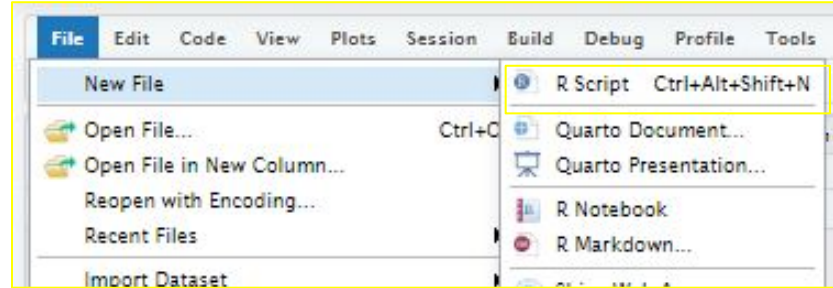
# Making a Simple R Application

# New R Script

Now we will create an R Script to utilize the “shiny” library

Select “File” On the top left side of your work space, Select “New File” then “R Script”

Notice the other project is open in a different tab, we can refer to this if we need to

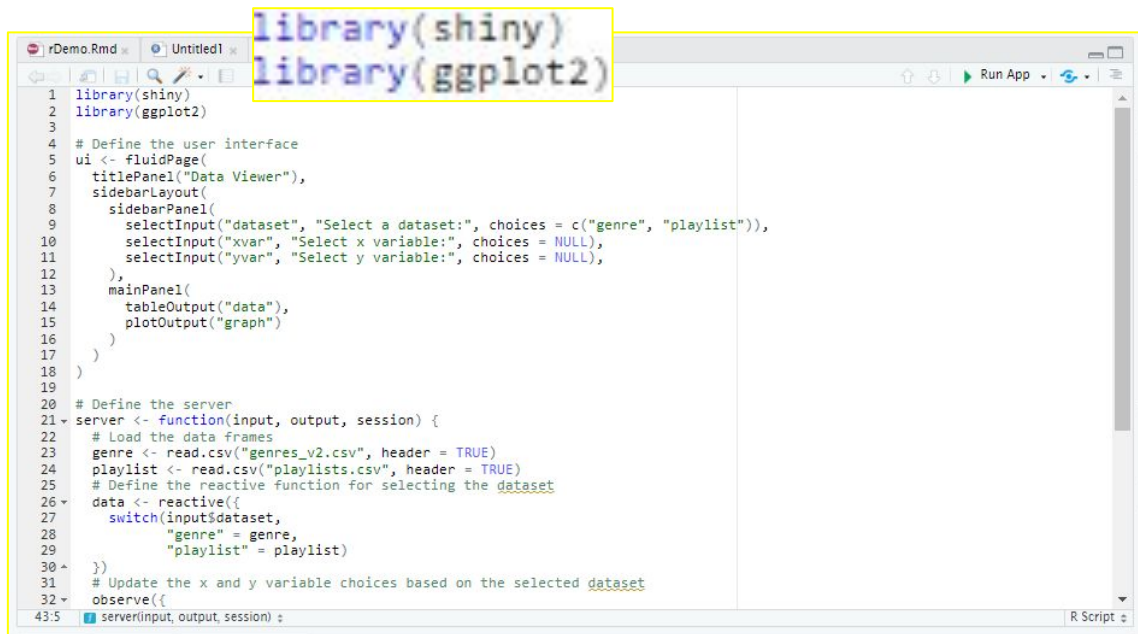


# Shiny Library

Upon Starting:

- Install and load all necessary libraries for your project prior to coding
- RStudio may have shiny already installed, to check what packages we have installed we can enter **installed.packages()**
- Load any other libraries you would like to use in your application using **library(Package\_Name)**

After you have the shiny library installed, we can now set up a server to run our R application



```
library(shiny)
library(ggplot2)

# Define the user interface
ui <- fluidPage(
  titlePanel("Data Viewer"),
  sidebarLayout(
    sidebarPanel(
      selectInput("dataset", "Select a dataset:", choices = c("genre", "playlist")),
      selectInput("xvar", "Select x variable:", choices = NULL),
      selectInput("yvar", "Select y variable:", choices = NULL),
    ),
    mainPanel(
      tableOutput("data"),
      plotOutput("graph")
    )
  )
)

# Define the server
server <- function(input, output, session) {
  # Load the data frames
  genre <- read.csv("genres_v2.csv", header = TRUE)
  playlist <- read.csv("playlists.csv", header = TRUE)
  # Define the reactive function for selecting the dataset
  data <- reactive({
    switch(input$dataset,
      "genre" = genre,
      "playlist" = playlist)
  })
  # Update the x and y variable choices based on the selected dataset
  observe({
    server(input, output, session)
  })
}
```

For our demonstration we will utilize libraries “Shiny” And “GGPlot2”

The shiny library can be called by entering:

**library(shiny)**

And with GGPlot installed,

**library(ggplot2)**

# Define the user interface

First we need to define the user interface. This can be done by the following

```
# Define the user interface
ui <- fluidPage(
  titlePanel("Data Viewer"),
  sidebarLayout(
    sidebarPanel(
      selectInput("dataset", "Select a dataset:", choices = c("genre", "playlist")),
      selectInput("xvar", "Select x variable:", choices = NULL),
      selectInput("yvar", "Select y variable:", choices = NULL),|
    ),
    mainPanel(
      tableOutput("data"),
      plotOutput("graph")
    )
  )
)
```

We first define in our code 'ui' with shiny's "fluidPage" function. We then can refer to the shiny library and fill out our first application user interface (UI)

Coding Notes:

'<-' : Is the "**Assignment Operator**", to assign values to the variables.

'titlePanel'- Enter "title"

'sidebarLayout' - setting up the "**sidebar**"

'sidebarPanel'- inputs a specified "**panel**" on the **sidebar**

'selectInput'- allows for inputs for the **panel**

'mainPanel'- displays the final graphs in the main window

# Making our server

```
# Define the server
server <- function(input, output, session) {
  # Load the data frames
  genre <- read.csv("genres_v2.csv", header = TRUE)
  playlist <- read.csv("playlists.csv", header = TRUE)
  # Define the reactive function for selecting the dataset
  data <- reactive({
    switch(input$dataset,
           "genre" = genre,
           "playlist" = playlist)
  })
  # Update the x and y variable choices based on the selected dataset
  observe({
    updateSelectInput(session, "xvar", choices = names(data()))
    updateSelectInput(session, "yvar", choices = names(data()))
  })
  # Render the table output
  output$data <- renderTable({
    head(data())
  })
  # Render the plot output
  output$graph <- renderPlot({
    ggplot(data(), aes_string(x = input$xvar, y = input$yvar)) + geom_point()
  })
}

# Run the Shiny app
shinyApp(ui = ui, server = server)
```

Coding Notes:

'server' - where we will host all of our functions that update our output

'function' - creates a basic function with inputs, outputs and the session of the server

'data' - creating our data environment within the server

'observe' - allows for the server to react to changes made during use of the application

'output' - displays a specified output

'\$' - is an operator in R is used to access specific columns or variables in a data frame.

'header = TRUE' - the first row is used for the names of the columns

'shinyApp' - runs the shiny application in a window

# Running the Application

First, click  in the top right of the coding window

Here we can see the code in action!

`titlePanel("Data Viewer")`

Data Viewer

```
sidebarLayout(  
  sidebarPanel(  
    selectInput("dataset", "Select a dataset:", choices = c("genre", "playlist"),  
    selectInput("xvar", "Select x variable:", choices = NULL),  
    selectInput("yvar", "Select y variable:", choices = NULL),  
    # Update the x and y variable choices based on the selected dataset  
    observe({  
      updateSelectInput(session, "xvar", choices = names(data()))  
      updateSelectInput(session, "yvar", choices = names(data()))  
    })  
  )  
)
```

```
# Render the table output  
output$data <- renderTable({  
  head(data())  
})
```

```
# Render the plot output  
output$graph <- renderPlot({  
  ggplot(data(), aes_string(x = input$xvar, y = input$yvar)) + geom_point()  
})
```

APP: <https://posit.cloud/content/5793375>

Select a dataset:

genre

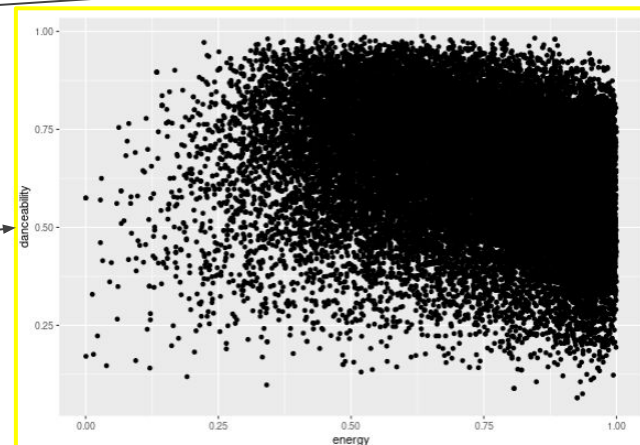
Select x variable:

energy

Select y variable:

danceability

| danceability | energy | key | loudness | mode | speechiness | acousticness |
|--------------|--------|-----|----------|------|-------------|--------------|
| 0.83         | 0.81   | 2   | -7.36    | 1    | 0.42        | 0.06         |
| 0.72         | 0.49   | 8   | -7.23    | 1    | 0.08        | 0.40         |
| 0.85         | 0.89   | 5   | -4.78    | 1    | 0.06        | 0.01         |
| 0.48         | 0.78   | 0   | -4.71    | 1    | 0.10        | 0.02         |
| 0.80         | 0.62   | 2   | -7.67    | 1    | 0.29        | 0.22         |
| 0.72         | 0.57   | 0   | -11.29   | 1    | 0.41        | 0.05         |







**Congratulations on completing your first R application!!**



# Additional Functions

```
library(shiny)
library(spotifyr)
library(tibble)
library(class)

Sys.setenv(SPOTIFY_CLIENT_ID = [REDACTED])
Sys.setenv(SPOTIFY_CLIENT_SECRET = [REDACTED])

access_token <- get_spotify_access_token()
```

Many libraries allow us to connect with multiple APIs (**Application Programming Interface**).

An example of one of these libraries is the “spotifyr” library that allows R to connect with the Spotify API.

Other libraries open to the public offer more ways for R programmer to connect to their favorite APIs.

Enter artist name

eric brewer and friends

Danceability: 0.3 0.8 1

Energy: 0.3 0.8 1

Valence: 0.3 0.8 1

Submit

Show 25 entries Search:

| artist_name             | danceability | energy | valence | track_name           | external_urls.spotify   |
|-------------------------|--------------|--------|---------|----------------------|---|
| Eric Brewer and Friends | 0.386        | 0.734  | 0.714   | 79th Street Shuffle  | <a href="https://open.spotify.com/track/6tml5b8b0QSugn59yaX5a5">https://open.spotify.com/track/6tml5b8b0QSugn59yaX5a5</a> |
| Eric Brewer and Friends | 0.573        | 0.629  | 0.565   | Benson               | <a href="https://open.spotify.com/track/5v9loVCywo9rvjZ4mdrpyz">https://open.spotify.com/track/5v9loVCywo9rvjZ4mdrpyz</a> |
| Eric Brewer and Friends | 0.506        | 0.735  | 0.423   | More After Diss      | <a href="https://open.spotify.com/track/4wutvza0OitFmbxTPLTnjc">https://open.spotify.com/track/4wutvza0OitFmbxTPLTnjc</a> |
| Eric Brewer and Friends | 0.359        | 0.679  | 0.454   | Rosy Dan             | <a href="https://open.spotify.com/track/2VgIL8NKFRRio8Nkxki4bs">https://open.spotify.com/track/2VgIL8NKFRRio8Nkxki4bs</a> |
| Eric Brewer and Friends | 0.423        | 0.763  | 0.678   | 79th. Street Shuffle | <a href="https://open.spotify.com/track/0fdGQ0mf19Jw5Pz6iMTnJ">https://open.spotify.com/track/0fdGQ0mf19Jw5Pz6iMTnJ</a>   |
| Eric Brewer and Friends | 0.531        | 0.728  | 0.480   | Benson               | <a href="https://open.spotify.com/track/3f1v17ESpL0ME0m99ZySw">https://open.spotify.com/track/3f1v17ESpL0ME0m99ZySw</a>   |
| Eric Brewer and Friends | 0.442        | 0.799  | 0.398   | F.U.A.               | <a href="https://open.spotify.com/track/4nV0rsSHbuyggEs8MnEQt">https://open.spotify.com/track/4nV0rsSHbuyggEs8MnEQt</a>   |
| Eric Brewer and Friends | 0.458        | 0.767  | 0.642   | Friends              | <a href="https://open.spotify.com/track/3pqd1gnYVI9HgdCjEJlRk">https://open.spotify.com/track/3pqd1gnYVI9HgdCjEJlRk</a>   |



# Conclusion

R is a powerful tool for data analysis and visualization

With R, you can load data from various sources, view and manipulate data, and create sophisticated graphs and visualizations

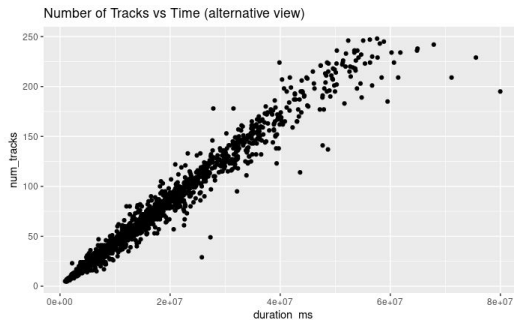
There are many additional libraries available to extend R's functionality, and functions available for data manipulation and analysis

R also allows for the creation of interactive applications and dashboards using libraries like shiny

With its many capabilities and a large and active user community, R is a great choice for data analysis and visualization in various industries and fields.

## Plotting Points Now Using Averages

```
AvgPlot <-  
ggplot(data=sPlaylistsTotal, aes(x=duration_ms, y=num_tracks)) +  
  ggtitle("Number of Tracks vs Time (alternative view)") +  
  geom_point(stat="identity")  
AvgPlot
```



## A For loop to Cycle Through Each Row in AvgData (no filter, hiphop, chill, rap)

```
for (i in 1:nrow(AvgData)) {  
  print(paste("the average playlist length in minutes for", AvgData$Catagory[i], "is:", AvgData$Time_in_Minutes[i], "the  
average playlist length in songs", AvgData$Catagory[i], " is:", AvgData$Songs_in_Playlist[i], "the average song length for",  
AvgData$Catagory[i], "is:", AvgData$Song_Length_Minutes[i]))  
}
```

```
[1] "the average playlist length in minutes for No Filter is: 259.051164533333 the average playlist length in songs No Filter  
is: 66.1305 the average song length for No Filter is: 3.91727212909827"  
[1] "the average playlist length in minutes for hiphop is: 417.685958333333 the average playlist length in songs hiphop is:  
105 the average song length for hiphop is: 3.97796150793651"  
[1] "the average playlist length in minutes for Chill is: 338.672039873418 the average playlist length in songs Chill is: 8  
4.4556962025316 the average song length for Chill is: 4.0100556279976"  
[1] "the average playlist length in minutes for Rap is: 327.082179545455 the average playlist length in songs Rap is: 83.38  
63636363636 the average song length for Rap is: 3.92249002452984"
```



# References

R Core Team (2021). R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria. URL <https://www.R-project.org/>

Wickham, H. (2016). ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York. ISBN 978-3-319-24277-4, URL <https://ggplot2.tidyverse.org/>

Chang, W., Cheng, J., Allaire, J., Xie, Y., & McPherson, J. (2021). shiny: Web Application Framework for R. R package version 1.6.0. URL <https://CRAN.R-project.org/package=shiny>

Wickham, H., François, R., Henry, L., & Müller, K. (2021). dplyr: A Grammar of Data Manipulation. R package version 1.0.7. URL <https://CRAN.R-project.org/package=dplyr>

Wickham, H. (2021). tidyr: Tidy Messy Data. R package version 1.1.4. URL <https://CRAN.R-project.org/package=tidyr>

Christian P. (2020). R API Tutorial: Getting Started with APIs in R URL <https://www.dataquest.io/blog/r-api-tutorial/>

# Questions & Open Discussion