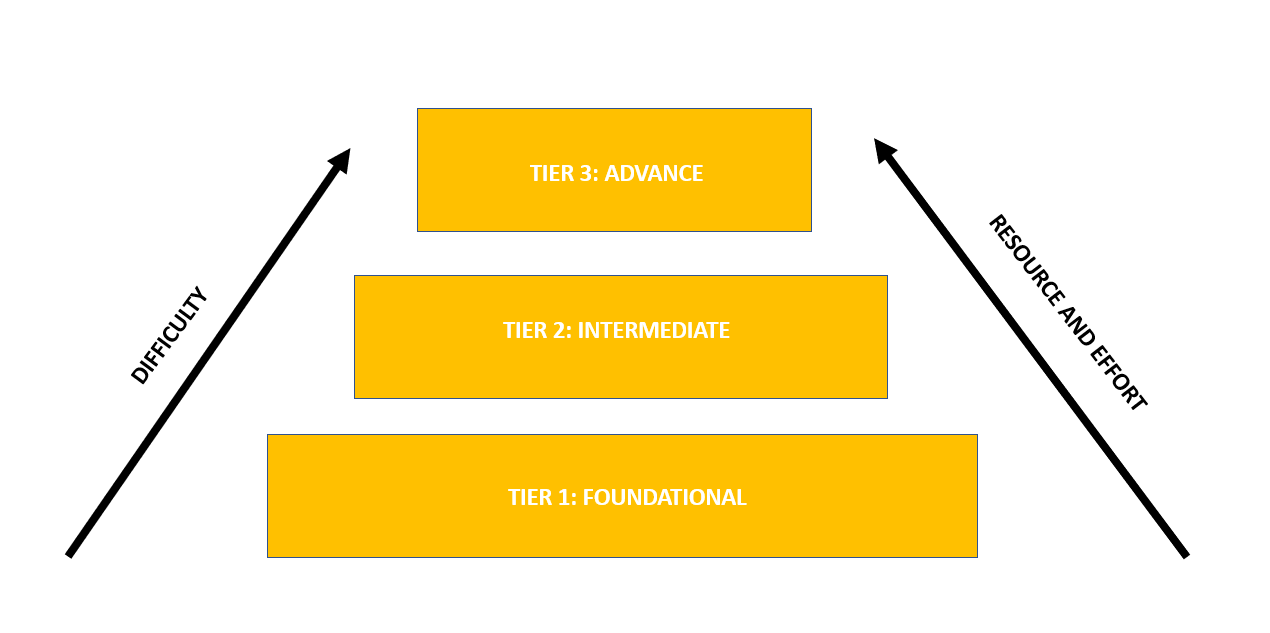


**“Moving on Up”: Upscaling your Analysis of Netflix Data with a Dashboard**

By Michael Hoang

Alright, we’re back. So, a quick recap of what happened. With one of my past articles, I went over the importance of building out your portfolio with your own unguided projects and figured it’s better to walk the walk instead of just talking about it. This means laying out three separate data projects from a single Netflix data set. Using R, the first project was a simple cleaning and exploration project that laid out the foundation for the other projects, including this one.



According to the above figure, what that previous project constituted to a “Tier-1 Foundational project”. Meaning that it’s fairly bare bones in nature, but nevertheless a very vital process for any data analyst/scientist. However, knowing that most people are card-carrying members of Team Python, I had enlisted help from some of the other community members to show that same process using Python given my ineptitude in this language. Thanks [*insert names here*].

Obviously, the next step from there is to advance to the next level, a “Tier-2” project. But what constitute one? Well, a tier-2 project is really anything that sits in a zone of being something of an upgrade from the “Tier-1” project but shouldn’t take longer than a week to complete (*assuming you’re putting in more than 2 hours/day*) outside of the initial cleaning process. Typically, this will take on the form of (1) a very-detailed exploration of the data (including comparative analyses) or (2) improving data visualization, but really anything can kind of fit in here. One example of this would be to put out a simple dashboard to show off the data.

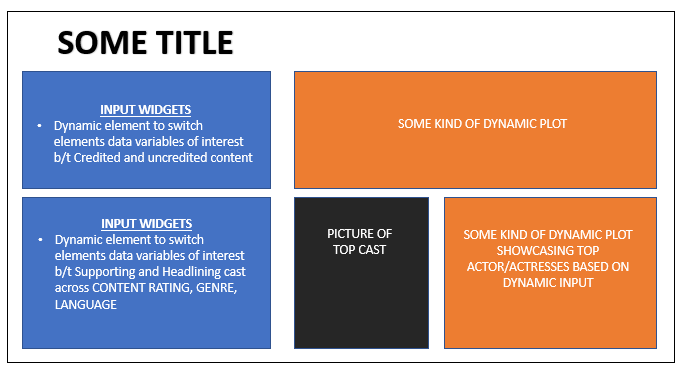
Full disclosure, I’ve been dipping my toes into Python recently and I’ll admit it’s growing on me. Not totally loving it as much as R, but it’s getting there with a few things. However, one of the few things that R still has an edge over Python is in the area of dashboarding with dynamic features compared to Dash.

**THE OBJECTIVE**

Now what I would like for this dashboard to do is give me a breakdown of the data as it pertains to the cast. Namely, I’m interested in looking at comparing Netflix content that differ from having credit cast and uncredited cast as it pertains to genre, content rating and content type. Additionally, I would also be interested in seeing if I can identify who are the top headlining cast member or top supporting cast member (*based on number of credits on Netflix*) across genre and/or language (i.e., English-speaking or non-English speaking).

**THE PROCESS**

Whenever you are creating a dashboard for the first time, the first thing you should do is do a quick sketch of how you would like it to look. In my case, I would like for it to look like the following:



NOTE: This may change in some regard in the end, but we’ll use this as a rough guide throughout the process

Now a lot of the groundwork was done previously with the Tier-1 project, so if you’re unsure about some of the code or process going forward, you can always just refer back to the previous article. Now the obvious first step for us starting out is to create a blank skeleton.

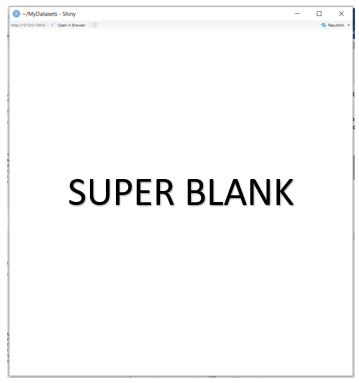
```

ui = fluidPage()

server = function(input, output) {}

shinyApp(ui = ui, server = server)

```



The next step would be to include a title and maybe a quick little description or blurb. What is noticeable is the use of the [HTML() function](https://shiny.rstudio.com/articles/html-tags.html) which is a way to utilize some HTML syntax to format the text on the Shiny app since it’s a web app.

```

ui= fluidPage(

wellPanel(

tags$h1("All about actors"),

tags$div(

HTML("<body>

<p style = 'font-size: 120%; text-align: justify'>

Let's be honest, what makes or breaks watching something is all about the cast. So why not look closer at this by checking out how certain cast-credited content differs from non-cast-credited content. Also, since we want to have a deeper dive, let's look at what are the most credited actor/actress across genre, language and casting type.

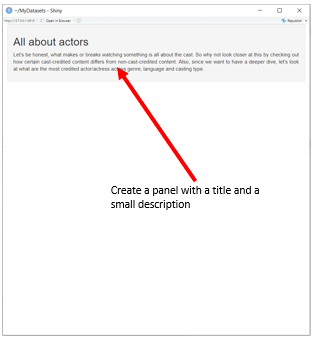
</p>

</body>")

))

)

```



Moving on, the next part would be to incorporate the input widgets and outputs that compares Netflix content differentiating based on whether there is a credited cast or not. What we’ll need to do is allocate a segment of the page to both these components, which is accomplished using [fluidRow() function](https://shiny.rstudio.com/reference/shiny/1.4.0/fluidPage.html). Additionally, we’ll use the [RadioButton()](https://shiny.rstudio.com/reference/shiny/1.6.0/radioButtons.html) widget to serve as the means of switching between plot outputs, which is wrapped around a well panel.

```

ui= fluidPage(

wellPanel(

tags$h1("All about actors"),

tags$div(

HTML("<body>

<p style = 'font-size: 120%; text-align: justify'>

Let's be honest, what makes or breaks watching something is all about the cast.

So why not look closer at this by checking out how certain cast-credited content differs from non-cast-credited content.

Also, since we want to have a deeper dive, let's look at what are the most credited actor/actress across genre, language and casting type.

</p>

</body>")

)),

fluidRow(

column(width = 3,

wellPanel(radioButtons(inputId = "cast\_charts",

label = "What sort of chart involving cast type are you interested in observing?",

choices = c("Just looking at cast counts" = "default",

"Across content type" = "type",

"Across content rating" = "content",

"English content vs. Non-English content" = "english",

"Across genres" = 'genre')))),

column(width = 8, plotOutput(outputId = "cast\_charts"))

),

tags$br()

) # ending fluidPage

```



From here, the next step is to just add the corresponding component on the server side. This is simply accomplished using a combination of the if/else statements in conjunction with [ggplot()](https://ggplot2.tidyverse.org/reference/ggplot.html) to make each individual plot dynamically react to the input widget.

```

server = function(input, output) {

output$cast\_charts = renderPlot({

if (input$cast\_charts == "default") {

netflix\_cast\_split %>%

mutate(

any\_cast = ifelse(cast == "Unknown/No Cast", "no", "yes")

) %>%

group\_by(any\_cast) %>%

summarise(count = n()) %>%

ggplot(aes(x = reorder(any\_cast, count), y = count, fill = any\_cast)) +

geom\_bar(stat = 'identity', color = 'black') +

scale\_fill\_manual(values = c("yes" = 'firebrick2', 'no' = 'slateblue3')) +

theme\_classic() +

theme(

axis.line = element\_line(color = 'black'),

axis.text = element\_text(color = 'black', size = 15),

axis.text.y = element\_text(size = 15),

axis.text.x = element\_text(size = 15),

plot.title = element\_text(size = 20, hjust = 0.5),

axis.title.x = element\_text(size = 15),

axis.title.y = element\_text(size = 15),

strip.text.x = element\_text(size = 15, color = 'black')

) +

guides(fill = FALSE) +

labs(x = "Presence of a Credited Cast?", y = "Number of Netflix Content", title = "Breakdown of Netflix Content by Presence of Credited Cast") +

coord\_flip()

} else if (input$cast\_charts == "english") {

netflix\_countryxcast\_df %>%

mutate(

any\_cast = ifelse(cast == "Unknown/No Cast", "no", "yes")

) %>%

mutate(

english\_or\_not =

ifelse(c(country\_type == "main country" & country\_name == "United States"), "English Speaking",

ifelse(c(country\_type == "main country" & country\_name == "Canada"), "English Speaking",

ifelse(c(country\_type == "main country" & country\_name == "United Kingdom"), "English Speaking",

ifelse(c(country\_type == "main country" & country\_name == "Australia"), "English Speaking",

ifelse(c(country\_type == "main country" & country\_name == "New Zealand"), "English Speaking",

ifelse(c(country\_type == "main country" & country\_name == "Ireland"), "English Speaking",

ifelse(c(country\_type == "main country" & country\_name == "Jamaica"), "English Speaking",

ifelse(c(country\_type == "main country" & country\_name == "Barbados"), "English Speaking",

ifelse(c(country\_type == "main country" & country\_name == "Belize"), "English Speaking",

ifelse(country\_type == "main country", "Non-English Speaking", NA))))))))))

) %>%

filter(country\_type == "main country") %>%

filter(!is.na(english\_or\_not)) %>%

group\_by(any\_cast, english\_or\_not) %>%

summarise(count = n()) %>%

ggplot(aes(x = reorder(any\_cast, count), y = count, fill = any\_cast)) +

geom\_bar(stat = 'identity', color = 'black') +

scale\_fill\_manual(values = c("yes" = 'firebrick2', 'no' = 'slateblue3')) +

theme\_classic() +

labs(x = "Presence of a Credited Cast?", y = "Number of Netflix Content", title = "Breakdown of Netflix Content by Presence of Credited Cast and Language") +

theme(

axis.line = element\_line(color = 'black'),

axis.text = element\_text(color = 'black', size = 15),

axis.text.y = element\_text(size = 15),

axis.text.x = element\_text(size = 15),

plot.title = element\_text(size = 20, hjust = 0.5),

axis.title.x = element\_text(size = 15),

axis.title.y = element\_text(size = 15),

strip.text.x = element\_text(size = 15, color = 'black')

) +

guides(fill = FALSE) +

coord\_flip() +

facet\_wrap(vars(english\_or\_not))

} else if (input$cast\_charts == "type") {

netflix\_cast\_split %>%

mutate(

any\_cast = ifelse(cast == "Unknown/No Cast", "no", "yes")

) %>%

group\_by(any\_cast, type) %>%

summarise(count = n()) %>%

ggplot(aes(x = reorder(any\_cast, count), y = count, fill = any\_cast)) +

geom\_bar(stat = 'identity', color = 'black') +

scale\_fill\_manual(values = c("yes" = 'firebrick2', 'no' = 'slateblue3')) +

theme\_classic() +

labs(x = "Presence of a Credited Cast?", y = "Number of Netflix Content", title = "Breakdown of Netflix Content by Presence of Credited Cast and Content Type") +

theme(

axis.line = element\_line(color = 'black'),

axis.text = element\_text(color = 'black', size = 15),

axis.text.y = element\_text(size = 15),

axis.text.x = element\_text(size = 15),

plot.title = element\_text(size = 25, hjust = 0.5),

axis.title.x = element\_text(size = 15),

axis.title.y = element\_text(size = 15),

strip.text.x = element\_text(size = 15, color = 'black')

) +

guides(fill = FALSE) +

coord\_flip() +

facet\_wrap(vars(type))

} else if (input$cast\_charts == "genre") {

netflix\_genrexcast\_df %>%

mutate(

any\_cast = ifelse(cast == "Unknown/No Cast", "no", "yes")

) %>%

group\_by(any\_cast, genre) %>%

summarise(count = n()) %>%

ggplot(aes(x = reorder(any\_cast, count), y = count, fill = any\_cast)) +

geom\_bar(stat = 'identity', color = 'black') +

scale\_fill\_manual(values = c("yes" = 'firebrick2', 'no' = 'slateblue3')) +

theme\_classic() +

labs(x = "Presence of a Credited Cast?", y = "Number of Netflix Content", title = "Breakdown of Netflix Content by Presence of Credited Cast and Genre") +

theme(

axis.line = element\_line(color = 'black'),

axis.text = element\_text(color = 'black', size = 15),

axis.text.y = element\_text(size = 15),

axis.text.x = element\_text(size = 15, angle = 90),

plot.title = element\_text(size = 25, hjust = 0.5),

axis.title.x = element\_text(size = 15),

axis.title.y = element\_text(size = 15),

strip.text.x = element\_text(size = 8, color = 'black')

) +

guides(fill = FALSE) +

coord\_flip() +

facet\_wrap(vars(genre))

} else if (input$cast\_charts == "content") {

netflix\_cast\_split %>%

mutate(

any\_cast = ifelse(cast == "Unknown/No Cast", "no", "yes")

) %>%

group\_by(any\_cast, rating) %>%

summarise(count = n()) %>%

ggplot(aes(x = reorder(any\_cast, count), y = count, fill = any\_cast)) +

geom\_bar(stat = 'identity', color = 'black') +

scale\_fill\_manual(values = c("yes" = 'firebrick2', 'no' = 'slateblue3')) +

theme\_classic() +

labs(x = "Presence of a Credited Cast?", y = "Number of Netflix Content", title = "Breakdown of Netflix Content by Presence of Credited Cast and Content Rating") +

theme(

axis.line = element\_line(color = 'black'),

axis.text = element\_text(color = 'black', size = 15),

axis.text.y = element\_text(size = 15),

axis.text.x = element\_text(size = 15),

plot.title = element\_text(size = 20, hjust = 0.5),

axis.title.x = element\_text(size = 15),

axis.title.y = element\_text(size = 15),

strip.text.x = element\_text(size = 15, color = 'black')

) +

guides(fill = FALSE) +

coord\_flip() +

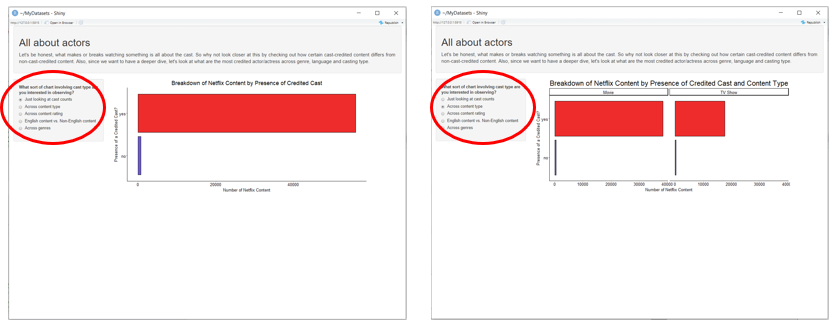
facet\_wrap(vars(rating))

}

})

}

```



What’s likely notable here is that when creating each of these individual plots, I’ve used different datasets that had been separated out based on certain variables. This is the reason why we had to create individualized earlier on.

The following part would be to add the input widgets for determining who are the top credited actors/actresses (either headlining, supporting or both) that’s based on genre, content rating and/or language. As usual, we’ll start with the UI side of things by adding the necessary codes for the input widgets, which will be a combination of RadioButtons() and [selectInput().](https://shiny.rstudio.com/reference/shiny/1.6.0/selectInput.html)

```

# On the UI Side

fluidRow(

column(width = 4, wellPanel(

selectInput(inputId = "cast\_comparisons", label = "Observe the top Actors/Actresses by what?",

choices = c("Genre: TV Action & Adventure" = "TV Action & Adventure",

"Genre: Movie Action & Adventure" = "Action & Adventure",

"Genre: TV Drama" = "TV Dramas",

"Genre: Movie Drama" = "Dramas",

"Genre: TV Comedy" = "TV Comedies",

"Genre: Movie Comedy" = "Comedies",

"Genre: TV Horror" = "TV Horror",

"Genre: Movie Horror" = "Horror Movies",

"Genre: Stand-Up/Skits/Talk Shows" = "Stand-Up Comedy & Talk Shows",

"Genre: Stand-Up Specials" = "Stand-Up Comedy",

"Genre: Music & Musicals" = "Music & Musicals",

"Genre: Sports" = "Sports Movies",

"Genre: LGBTQ" = "LGBTQ Movies",

"Genre: Independent Movies" = "Independent Movies",

"Genre: Classic & Cult TV" = "Classic & Cult TV",

"Genre: Cult Movies" = "Cult Movies",

"Genre: Classic Movies" = "Classic Movies",

"Genre: Korean TV" = "Korean TV Shows",

"Genre: British TV" = "British TV Shows",

"Genre: Spanish TV" = "Spanish-Language TV Shows",

"Genre: Children & Family Movies" = "Children & Family Movies",

"Genre: Kids' TV" = "Kids' TV",

"Genre: Teen TV" = "Teen TV Shows",

"Genre: TV Thriller" = "TV Thrillers",

"Genre: Movie Thriller" = "Thrillers",

"Genre: Reality TV" = "Reality TV",

"Genre: International Movie" = "International Movies",

"Genre: International TV" = "International TV Shows",

"Genre: TV SciFi & Fantasy" = "TV Sci-Fi & Fantasy",

"Genre: Movie SciFi & Fantasy" = "Sci-Fi & Fantasy",

"Genre: Science & Nature TV" = "Science & Nature TV",

"Genre: TV Romance" = "Romantic TV Shows",

"Genre: Movie Romance" = "Romantic Movies",

"Genre: Anime Features" = "Anime Features",

"Genre: Anime Series" = "Anime Series",

"Genre: Crime TV" = "Crime TV Shows",

"Genre: Mystery TV" = "TV Mysteries",

"Genre: Documentaries" = "Documentaries",

"Genre: Docuseries" = "Docuseries",

"Genre: Faith & Spirituality" = "Faith & Spirituality")),

radioButtons(inputId = "cast\_language", label = "Are you interested in English-Only Actors/Actresses?",

choices = c("Yes, just English" = "English Speaking",

"No, only Non-English" = "Non-English Speaking",

"Both" = "both")),

radioButtons(inputId = "cast\_type", label = "What cast billing are you interested in?",

choices = c("Headliners Only" = "headliner",

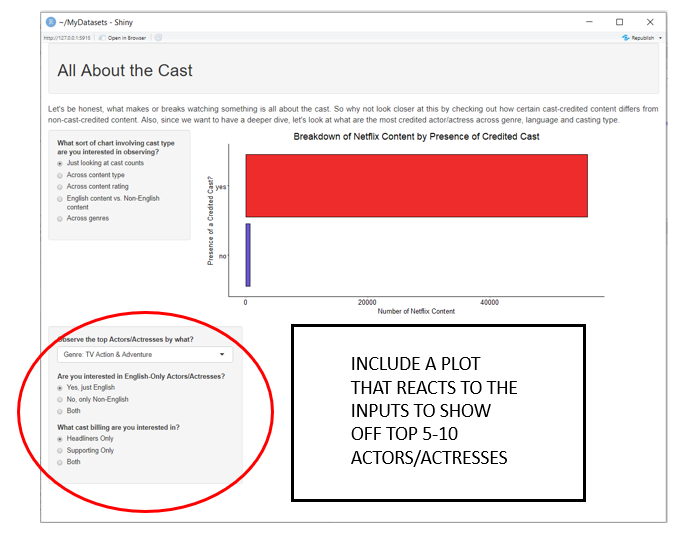
"Supporting Only" = "supporting cast",

"Both" = "both")))),

column(width = 4, plotOutput(outputId = "top\_cast")),

column(width = 4, uiOutput(outputId = "cast\_image")))

```



As for the server side of things, it’ll start with creating a reactive element that enable the switch between desired variables. This is possible with the [switch() function](https://www.datamentor.io/r-programming/switch-function/) that evaluates an expression based on its value and return from any existing list the corresponding element. In our case, it would return any element within the genre with the corresponding element from the dataset.

```

# on the server side

cast\_n\_director\_genre\_2 = reactive({switch(input$cast\_comparisons, as.character(input$cast\_comparisons))})

```

From here, the if/else statements are used to indicate which sort of plot output would result based on the widget inputs. In our case, since the genre will serve as a means of filtering out a given data set, the language and cast type input would serve in determining which of the many different separated data set to be used. Since each has three possible options, this means we have 8 possible options. Since this is a lot of syntax, you can check out the [code here](https://github.com/Vibe1990/Shiny-Project-Ideas/blob/main/Netflix%20Project/Tier%202%20Project:%20Dynamic%20Visualization%20with%20Shiny%20(CODE)).

```

output$top\_cast = renderPlot({

if(input$cast\_language == "both" & input$cast\_type == "headliner") {

netflix\_genrexcountryxcast\_df %>%

filter(cast != "Unknown/No Cast") %>%

filter(cast\_type == "headliner") %>%

filter(genre == as.character(cast\_n\_director\_genre\_2())) %>%

group\_by(cast) %>%

summarise(count = n()) %>%

arrange(desc(count)) %>%

filter(count > 1) %>%

top\_n(5) %>%

ggplot(aes(x = reorder(cast, count), y = count, fill = cast)) +

geom\_bar(stat = 'identity', color = 'black') +

guides(fill = F) +

theme\_classic() +

theme(

axis.text = element\_text(color = 'black'),

plot.caption = element\_text(color = 'black', hjust = 0)

) +

coord\_flip() +

labs(x = "Top Headlining Actors/Actresses", y = "Number of Netflix Content", caption = "\* Only actors with at least 2 credited roles on Netflix")

} else if (input$cast\_language == "English Speaking" & input$cast\_type == "headliner") {

netflix\_genrexcountryxcast\_df %>%

filter(cast != "Unknown/No Cast") %>%

filter(cast\_type == "headliner") %>%

filter(genre == as.character(cast\_n\_director\_genre\_2())) %>%

mutate(

english\_or\_not =

ifelse(c(country\_type == "main country" & country\_name == "United States"), "English Speaking",

ifelse(c(country\_type == "main country" & country\_name == "Canada"), "English Speaking",

ifelse(c(country\_type == "main country" & country\_name == "United Kingdom"), "English Speaking",

ifelse(c(country\_type == "main country" & country\_name == "Australia"), "English Speaking",

ifelse(c(country\_type == "main country" & country\_name == "New Zealand"), "English Speaking",

ifelse(c(country\_type == "main country" & country\_name == "Ireland"), "English Speaking",

ifelse(c(country\_type == "main country" & country\_name == "Jamaica"), "English Speaking",

ifelse(c(country\_type == "main country" & country\_name == "Barbados"), "English Speaking",

ifelse(c(country\_type == "main country" & country\_name == "Belize"), "English Speaking",

ifelse(country\_type == "main country", "Non-English Speaking", NA))))))))))

) %>%

filter(country\_type == "main country") %>%

filter(english\_or\_not == "English Speaking") %>%

group\_by(cast) %>%

summarise(count = n()) %>%

arrange(desc(count)) %>%

filter(count > 1) %>%

top\_n(5) %>%

ggplot(aes(x = reorder(cast, count), y = count, fill = cast)) +

geom\_bar(stat = 'identity', color = 'black') +

guides(fill = F) +

theme\_classic() +

theme(

axis.text = element\_text(color = 'black'),

plot.caption = element\_text(color = 'black', hjust = 0)

) +

coord\_flip() +

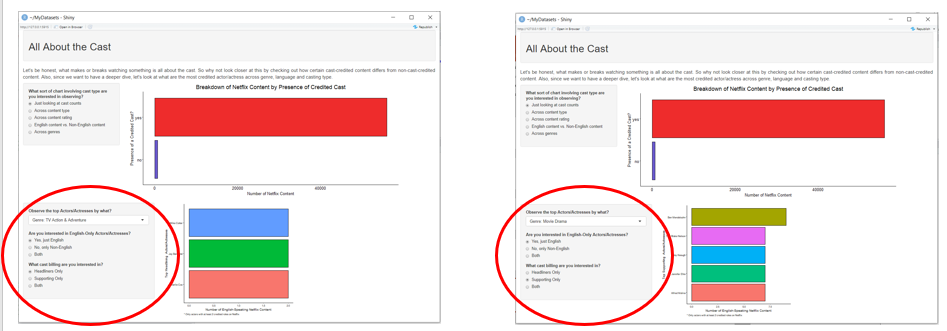
labs(x = "Top Headlining Actors/Actresses", y = "Number of English-Speaking Netflix Content", caption = "\* Only actors with at least 2 credited roles on Netflix")

# Repeat the process for the 6 other possibilities

}

})

```



So far, we’ve got a lot done with the dashboard as outlined earlier. The only thing left to do is input the appropriate image showcasing the top actor/actress as respective to the widget inputs. Now, normally, this wouldn’t be a problem since the process behind it is really just a matter of using the appropriate tag function (this case tags$img()) and if/else statement that match to the given inputs as shown in a previous DQ article. However, if you’ve been following along, you’ll notice that this data set doesn’t have such data. All it has is the listed cast member.

Oh no, we’ve reached a major dilemma in this project. What can we do here?



Alright, after a deep breath, let’s take a quick inventory of what we have and what we want:

WE HAVE:

* A data set with a list of every credited cast member where the first listed cast = headliner and the rest are supporting, all of which are correctly listed with another variables (cast type)

WHAT WE WANT:

* To showcase the top credited headlining/supporting cast across each genre type differentiated by language and/or cast type with a visual

Now there are a few possible solutions to this:

1. Drop the picture aspect completely as it would be too much effort (the easiest option particularly if time is of the essence)
2. Create another column that includes a link for each actor/actress. While it would result in a picture, this will be the most meathead/intensive process as there may be 100s, if not 1000s to search for where a picture may not even be available on the internet. Since I’m not getting paid for any of this, this is likely not worth the effort.
3. Let’s split the difference here and just figure out which are the top listed actor/actress for each given input options. With 8 datasets used and about 39 possible genres, we will have at most 312 possible individual option.

As tempting it is to go with option 1, we’ll go with option 3 since there are a few things that it has going for it to be a more viable route aside from my desire to have a pictorial element in the dashboard. For one, compared to the pain that would come out of option 2, having about a few hundred is a much better route to take compared to a few thousands. Also, while it’s true there are a possible of 312 unique results, the majority of the outcomes will likely be the same actor/actress for multiple genres. Lastly, by individually going through each outcome, you can get actually go into a deep dive into the data to see if what the actual outcomes makes sense or not given you base level of knowledge about the TV & Film industry to sus out suspected findings.

So, after playing around with the various different inputs, you’ll recognize that there are fair number of ties as well as recurring actors/actresses that are listed as the top-credited headliner/supporting member based on number of Netflix credits. In this case, there are 79 unique answers. While it’s a lot, but way more manageable even when compared to some of the step during the data wrangling process.

Now, I’m not going to be showing the entire process here as it would be too long, so check it out on my GitHub instead. However, I will run down the actual steps process with some placeholder variables:

```

Output$cast\_image = renderUI({

# Step 1: put the link for each of the unique actors/actresses that were found to have the highest credits for a given input into a variable

url\_tie = “some\_url\_for\_tie\_image”

url\_for\_actor\_2 = “actor\_2\_url”

url\_for\_actor\_3 = “actor\_3\_url”

url\_for\_actor\_4 = “actor\_4\_url”

url\_for\_actor\_5 = “actor\_5\_url”

url\_for\_actor\_22 = “actor\_22\_url”

url\_for\_actor\_33 = “actor\_33\_url”

url\_for\_actor\_44 = “actor\_44\_url”

url\_for\_actor\_55 = “actor\_55\_url”

url\_for\_actor\_12 = “actor\_12\_url”

url\_for\_actor\_13 = “actor\_13\_url”

url\_for\_actor\_14 = “actor\_14\_url”

url\_for\_actor\_1 = “actor\_1\_url”

# Step 2: using a series of ifelse functions, you will make a variable that provides one of the url\_variables above as a potential output THAT DOESN’T RESULT IN A TWO-PERSON TIE (will show what to do with these later).

# There will be eight of these variables, each corresponding to one of the eight possible combinations b/t language input and cast type input

english\_headliner\_img =

ifelse(c(input$cast\_language == "English Speaking" & input$cast\_type == "headliner" & (input$cast\_comparisons == "Action & Adventure" | input$cast\_comparisons == "Thrillers" | input$cast\_comparisons == "Sci-Fi & Fantasy")), url\_for\_actor\_1,

ifelse(c(input$cast\_language == "English Speaking" & input$cast\_type == "headliner" & (input$cast\_comparisons == "International Movies" | input$cast\_comparisons == "Documentaries" |

input$cast\_compaisons == "Docuseries")), url\_for\_actor\_22,

ifelse(c(input$cast\_language == "English Speaking" & input$cast\_type == "headliner" & input$cast\_comparisons == "TV Mysteries"), url\_for\_actor\_33, url\_tie)))

# repeat for English & supporting, English & both, non-English & supporting, non-English & headliner, non-English & both, both & headliner, both & supporting and overall

# Step 3: using a series of if and else if statements that is used to address any sort of two-way tie where I will result in a “tag” specific output. This would be followed with the findings that specifically resulted in a tie involving at least 3 actors/actresses and lastly followed-up with the unique singular outputs as done in step 2

if (input$cast\_language == "both" & input$cast\_type == "both" & input$cast\_comparisons == "TV Action & Adventure") {

tags$div(

HTML("<div class = 'row'>

<img src = 'actor\_4\_url' width='250' height='400'>

<img src = ‘actor\_33\_url' width='250' height='400'>

</div>")

)

} else if (input$cast\_language == "both" & input$cast\_type == "both" & input$cast\_comparisons == "TV Comedies") {

tags$div(

HTML("<div class = 'row'>

<img src = actor\_22\_url' width='250' height='400'>

<img src = 'actor\_33\_url’ width='250' height='400'>

</div>"))

} else if (input$cast\_language == "both" & input$cast\_type == "both" & input$cast\_comparisons == "Stand-Up Comedy") {

tags$div(

HTML("<div class = 'row'>

<img src = 'actor\_11\_url' width='250' height='400'>

<img src = 'actor\_44\_url' width='250' height='400'>

</div>"))

} else if (input$cast\_language == "both" & input$cast\_type == "both" & (input$cast\_comparisons == "TV Horror" |

input$cast\_comparisons == "Music & Musicals" |

input$cast\_comparisons == "LGBTQ Movies" |

input$cast\_comparisons == "Classic & Cult TV" |

input$cast\_comparisons == "Cult Movies" |

input$cast\_comparisons == "Korean TV Shows" |

input$cast\_comparisons == "Spanish-Language TV Shows" |

input$cast\_comparisons == "Teen TV Show" |

input$cast\_comparisons == "TV Thrillers" |

input$cast\_comparisons == "Reality TV" |

input$cast\_comparisons == "TV Sci-Fi & Fantasy" |

input$cast\_comparisons == "Sci-Fi & Fantasy" |

input$cast\_comparisons == "Crime TV Shows" |

input$cast\_comparisons == "TV Mysteries" |

input$cast\_comparisons == "Faith & Spirituality")) {

tags$img(src = url\_tie , width = '800', height = '400')

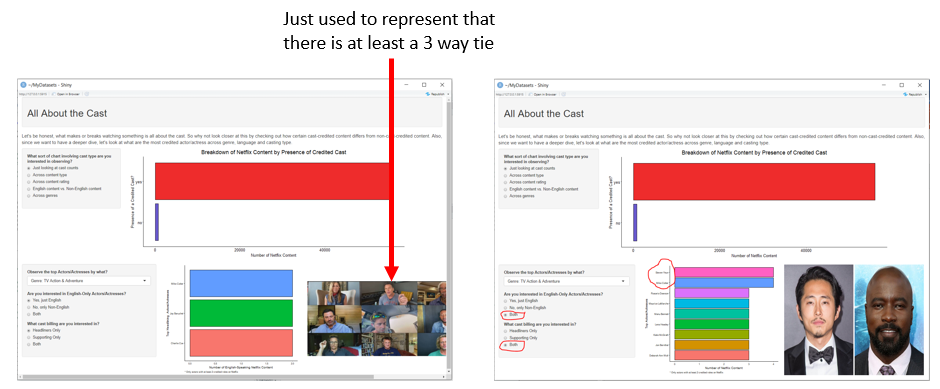
} else if (input$cast\_language == "English Speaking" & input$cast\_type == "headliner") {

tags$img(src = english\_headliner\_img, width = "275", height = '400')

})

```

The end product would look something like this:



And boom, we have ourselves a nice little Tier-2 project. While relatively simplistic in terms of type of functions used, it is undoubtedly an upgrade from the more static form of data reporting with the use of reactive elements. Additionally, we were also able to overcome one major roadblock through a rather slick way in order to achieve our desired outcome. Obviously, we can go a step further with improving upon the visual of this dashboard in terms of font styling and size, but you’ll need some additional knowledge in the [area of HTML and CSS to do so](https://www.youtube.com/watch?v=mU6anWqZJcc), which really is in the area of a UI/UX designer or a front-end developer but not necessarily a data analyst or scientist.

Having shown a Tier-1 and Tier-2 down, the next project will be that almighty Tier-3 project. A fitting grand finale of sorts. What exactly that I’ll be doing for it will be a surprise, so be on the lookout in the coming weeks as we cap off this little series along with some of the other articles from some of the other community members helping me out with showing off these projects on the Python side.

If you’re interested in check out some of my other projects, you can head over to my [GitHub](https://github.com/Vibe1990) to check some of them out. Alternatively, if you’re got some idea on a collaborative project or just want to connect, hit me up on my [LinkedIn](https://www.linkedin.com/in/michael-hoang-3222a220/).

Thanks for the read.