Packages

1. Table 1 shows some of the packages used in this study

Table 1: Some of the packages used in the study

Package	Function	Purpose
ROSE	Ovun.sample	Corrects class imbalance using oversampling, undersampling, a mixture of over/undersampling and the generation of synthetic rows
Tidyr	'Spread'	
Reshape2	'Melt'	Changing rows to columns
vars	varselect	
kernlab	ksvm	Build ksvm models
plyr	revalue	Change names of factor levels
DMwR	knn	Build knn models
nnet	nnet	Build neural net models
caret	caret::train	Building C-Forest,
adabag	b	
bbmisc	normalize	For normalizing numerical variables with non-normal distribution
cowplot	Plot_grid	Build plot with multiple ggplots in one
plyr	revalue	Change
DT	data.table	Creates datable for dashboard
ROAuth		Twitter Authentication
translateR	translate	Translate non-English text
syuzhet	get_sentiment	Analyse the sentiment of a piece of text
wordcloud2	wordcloud2	Generating wordclouds
dplyr	%>%	Piping function for stringing tasks together
TwitteR	setup_twitter_oauth	Setting up authentication for twitter
tidytext	unnest_tokens	Break text into individual words for text mining
SuperLearner	SuperLearner	Build ensemble data mining models with multiple different data mining algorithms

tseries	adf.test	Statistical test of time series
MTS	diffM	
vars	VARselect	
forecast	meanf, rwf, snaive	Forecasting methods
partykit	cforest	Build C-Forest models
kernlab	ksvm	Build SVM models
shiny	shiny	Build interactive dashboards
Flexdashboard	Flexdashboard	Build interactive dashboards
gbm	gbm	Build SVM models

System Requirements

Hardware

- 64-bit Windows operating system (the R portion can be performed using Mac, but the persistent backend uses Microsoft SQL Server, which is not available for Mac)
- Windows 8.1 or higher
- 1.8GHz or faster processor
- Minimum 10GB, recommended 20-50GB hard disk space

- 4GB RAM (> 8GB recommended)

Software

- .Net Framework 4.6 (found at https://support.microsoft.com/en-us/help/3045560/microsoft-net-framework-4-6-web-installer-for-windows)

Installing SSMS

1. Go to https://www.microsoft.com/en-US/download/details.aspx?id=29062

2. Click on ENGLISH in the dropdown menu (or another language if preferred) then click ‘Download’

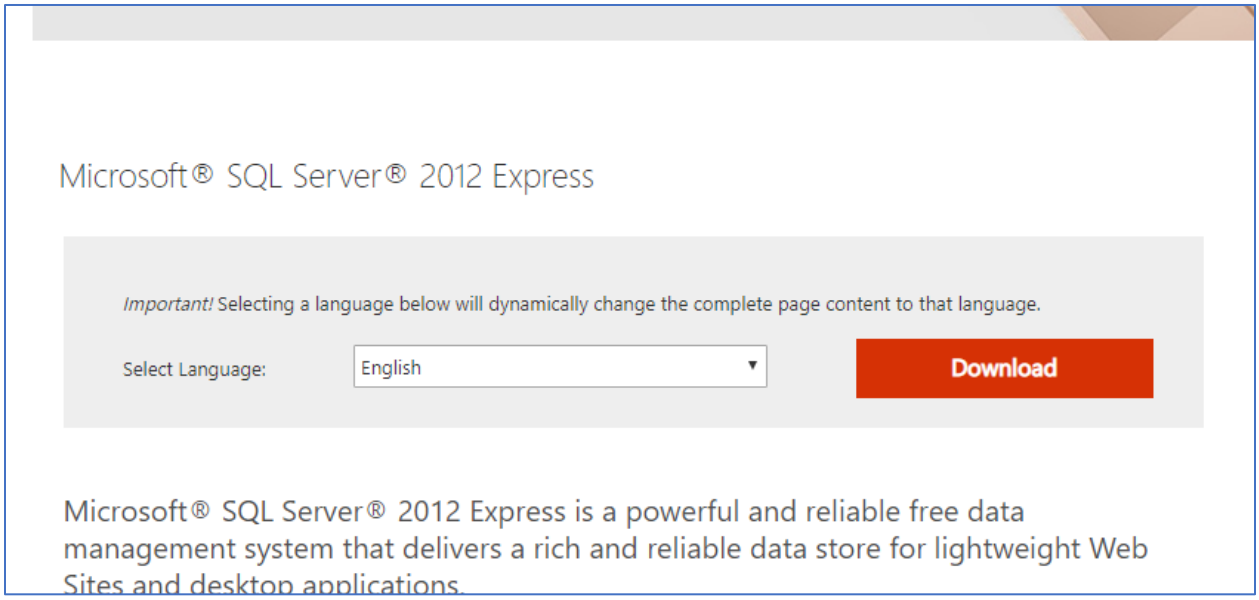

Figure 1: Choosing language for SQL Server

3. Click on 'ENU\x64\SQLEXPRADV_x64_ENU.exe'

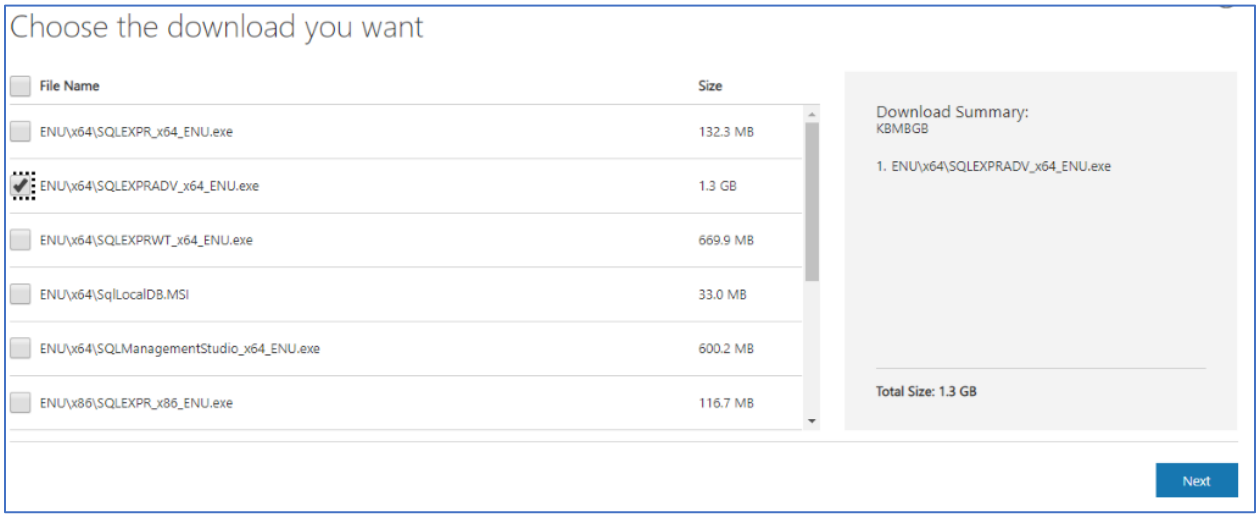

Figure 2. Choosing SSMS file to download

4. When the download has finished click on the .exe file (should be on the bottom left hand corner of the internet window, or in the download tab)

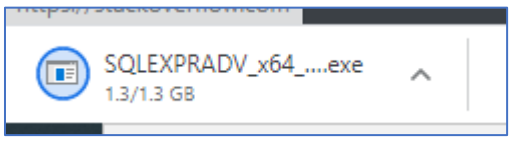

Figure 3. Download execution file for SSMS

5. Click ‘New SQL Server stand-alone installation or add features to an existing installation’

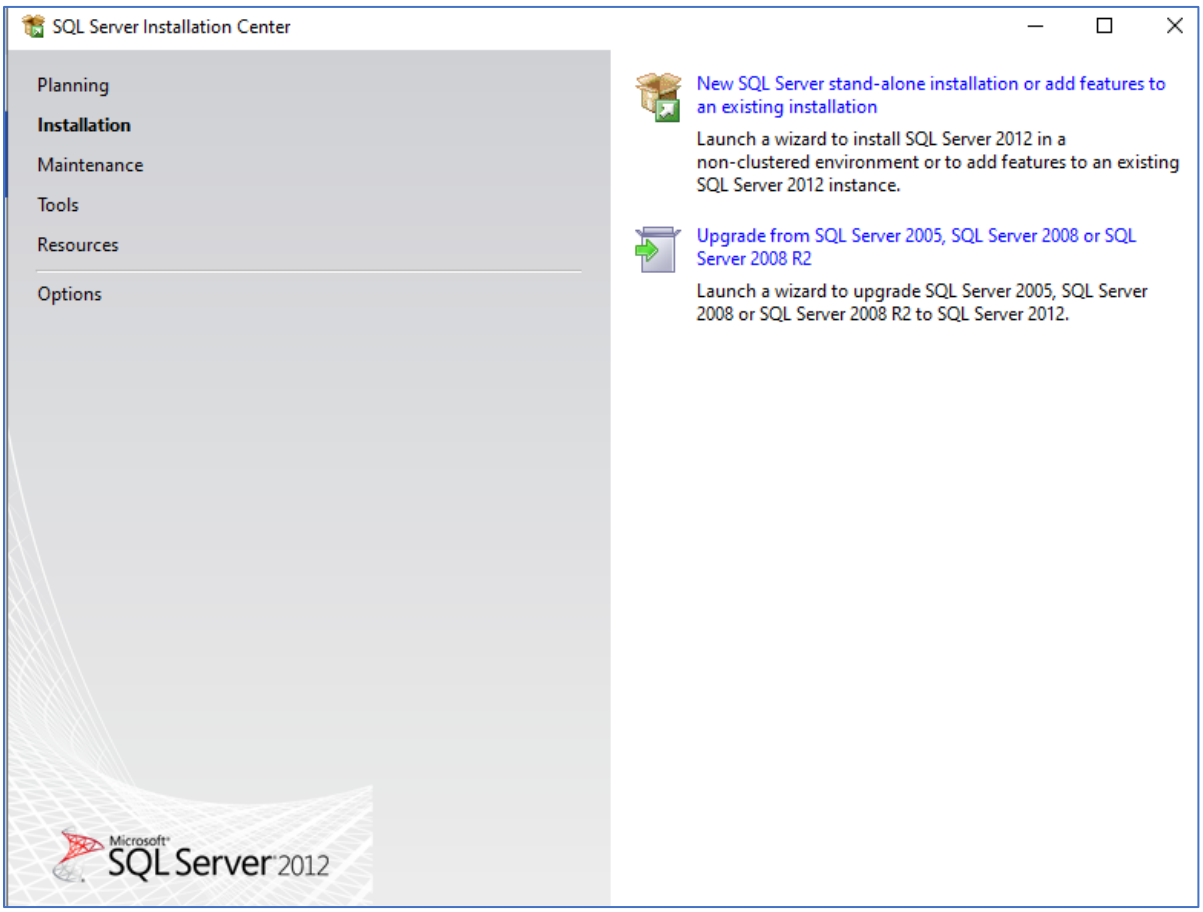

Figure 4: Installing standalone SSMS instance

6. Agree to the license terms then click ‘Next’

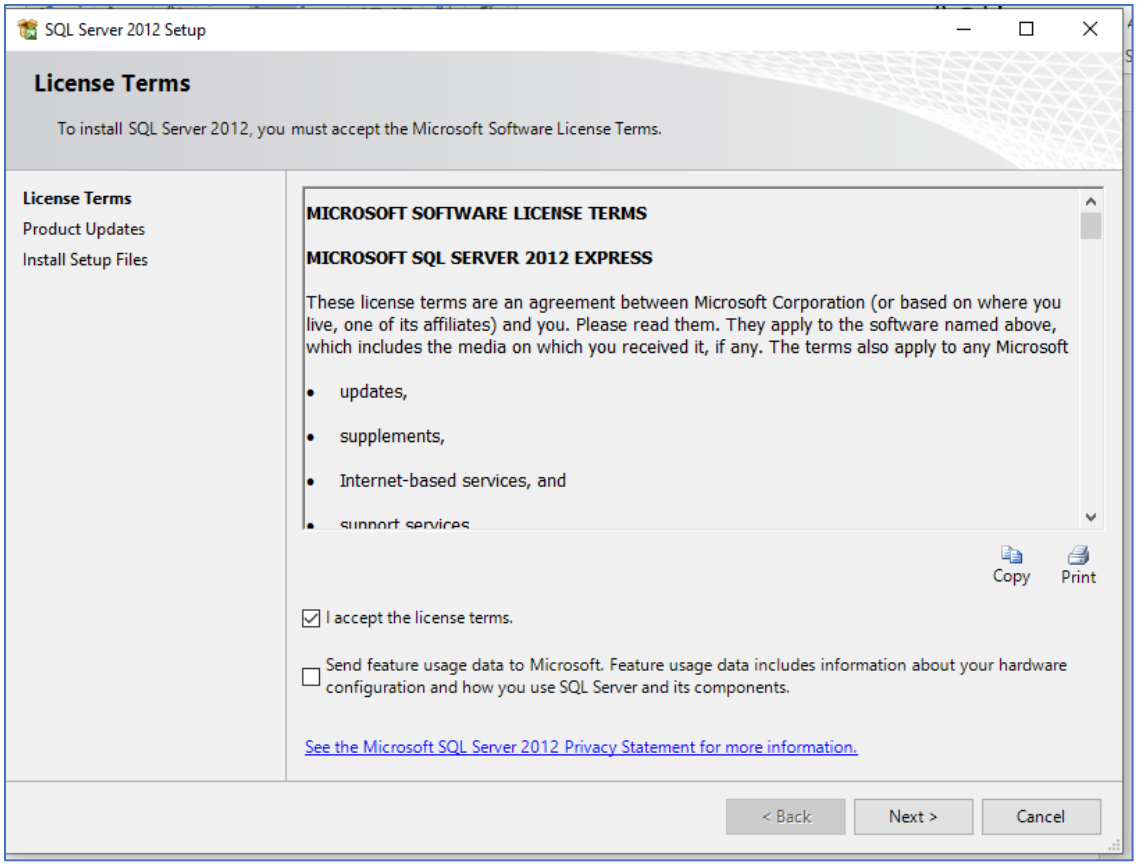

Figure 5. License terms screen for SSMS

7. Specify whether or not to include product updates (recommended). Then click ‘Next’

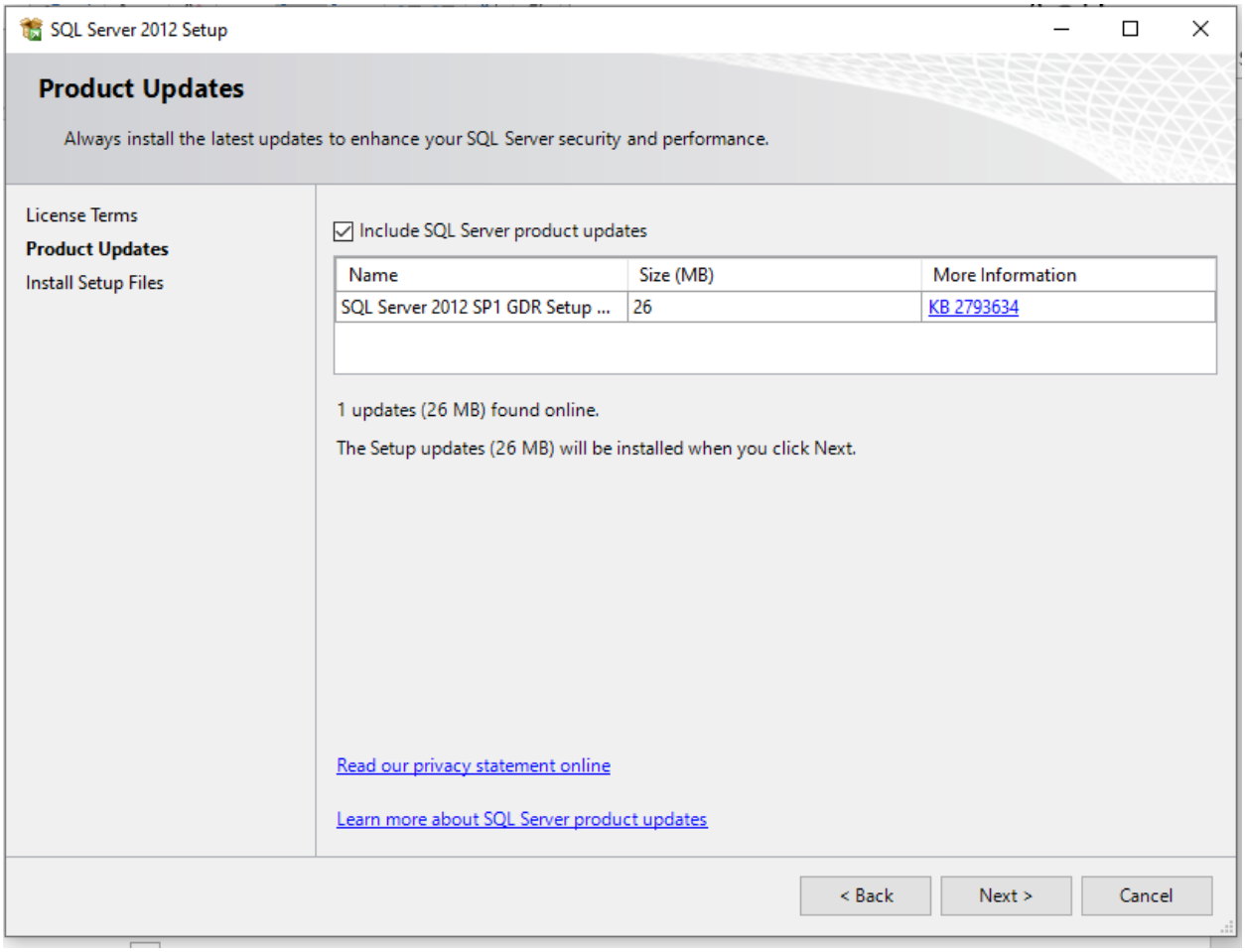

Figure 6: Product updates screen for SSMS

8. Select the features that you wish to install. In this case all features but 'Local DB' were selected then click 'Next'

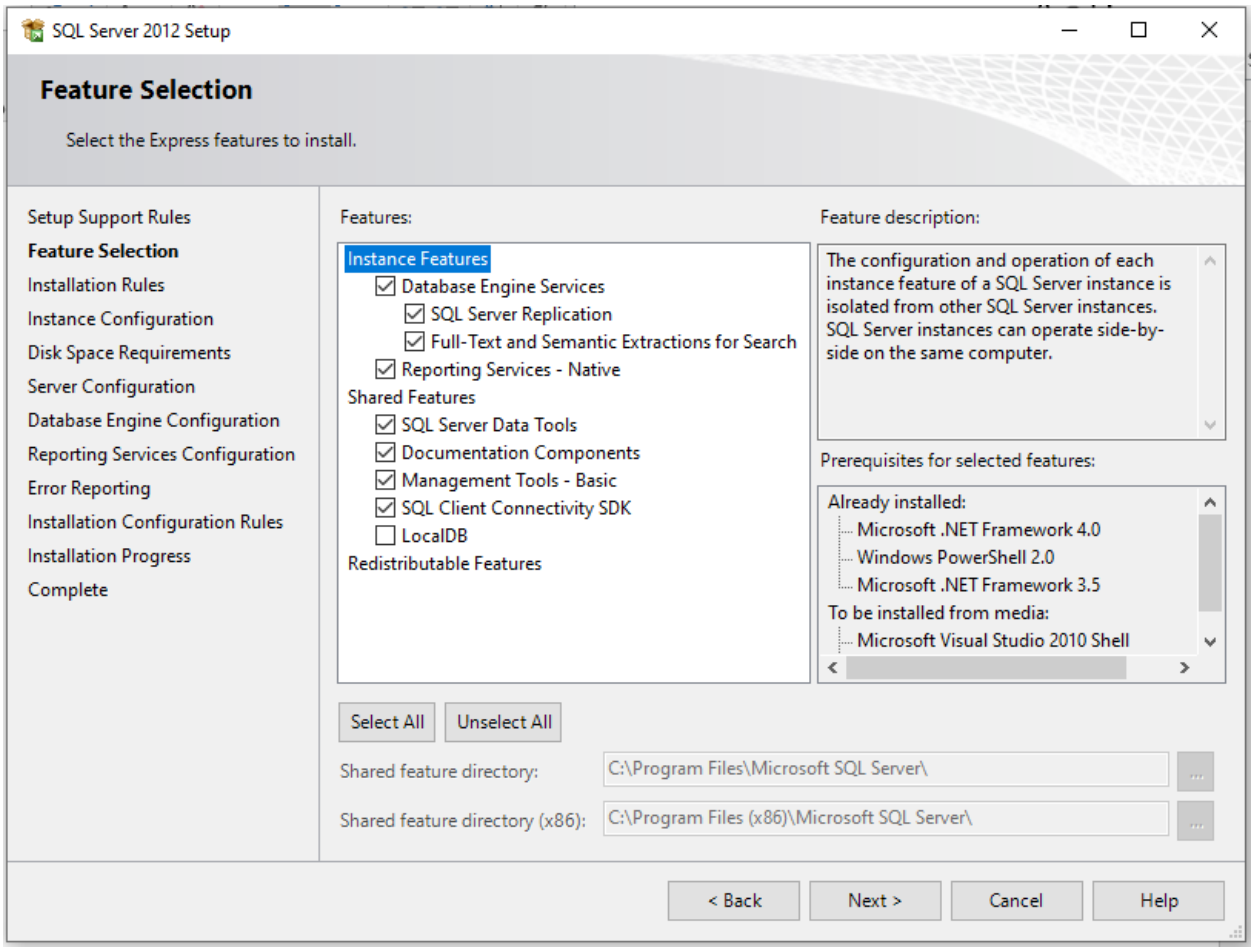

Figure 7: Selecting features to install

9. Specify the name and instance ID for the SQL Server instance then click ‘Next’

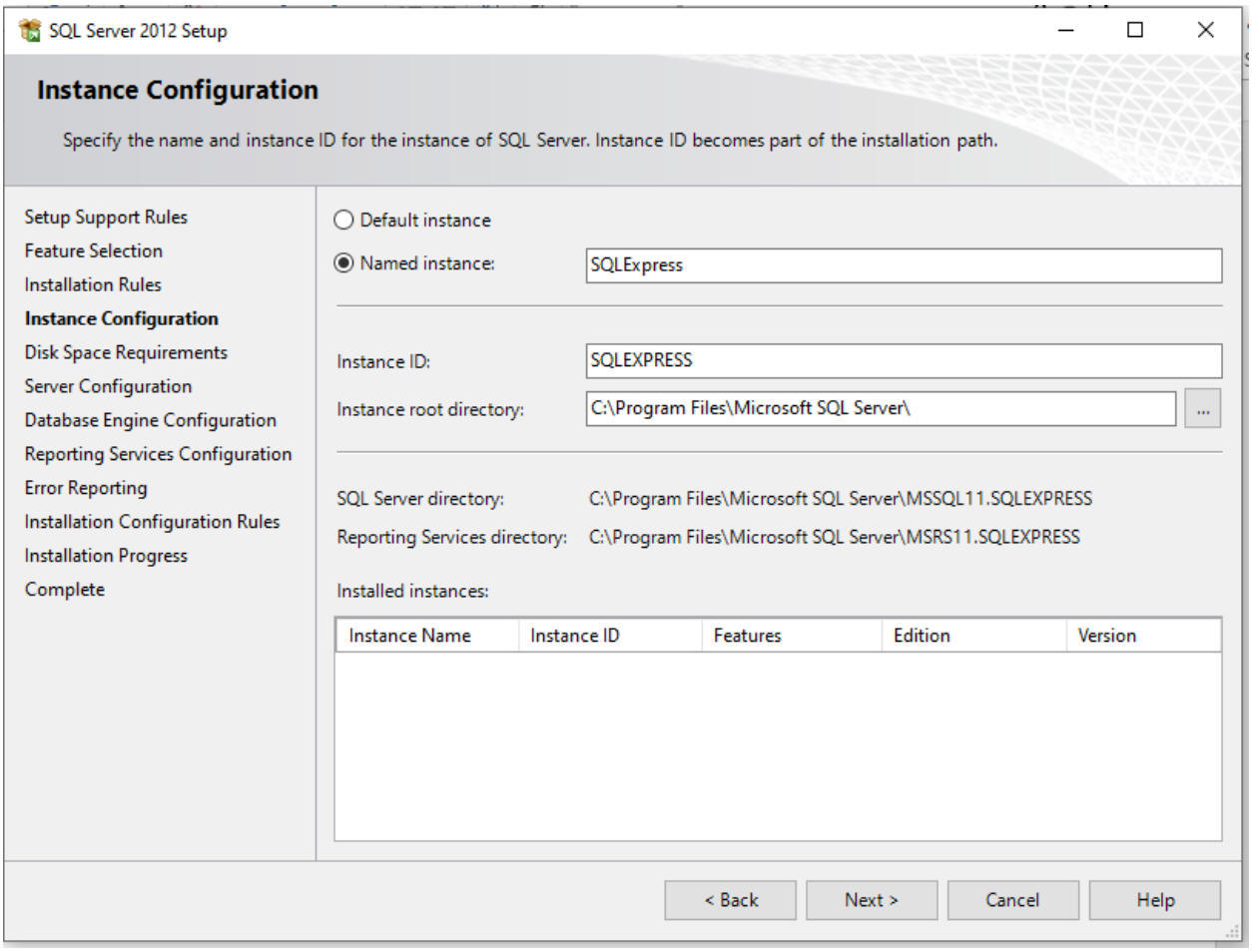

Figure 8. Specifying the name and ID for SQL instance

10. Specify the service account and collation configuration. The default settings that appear are suitable

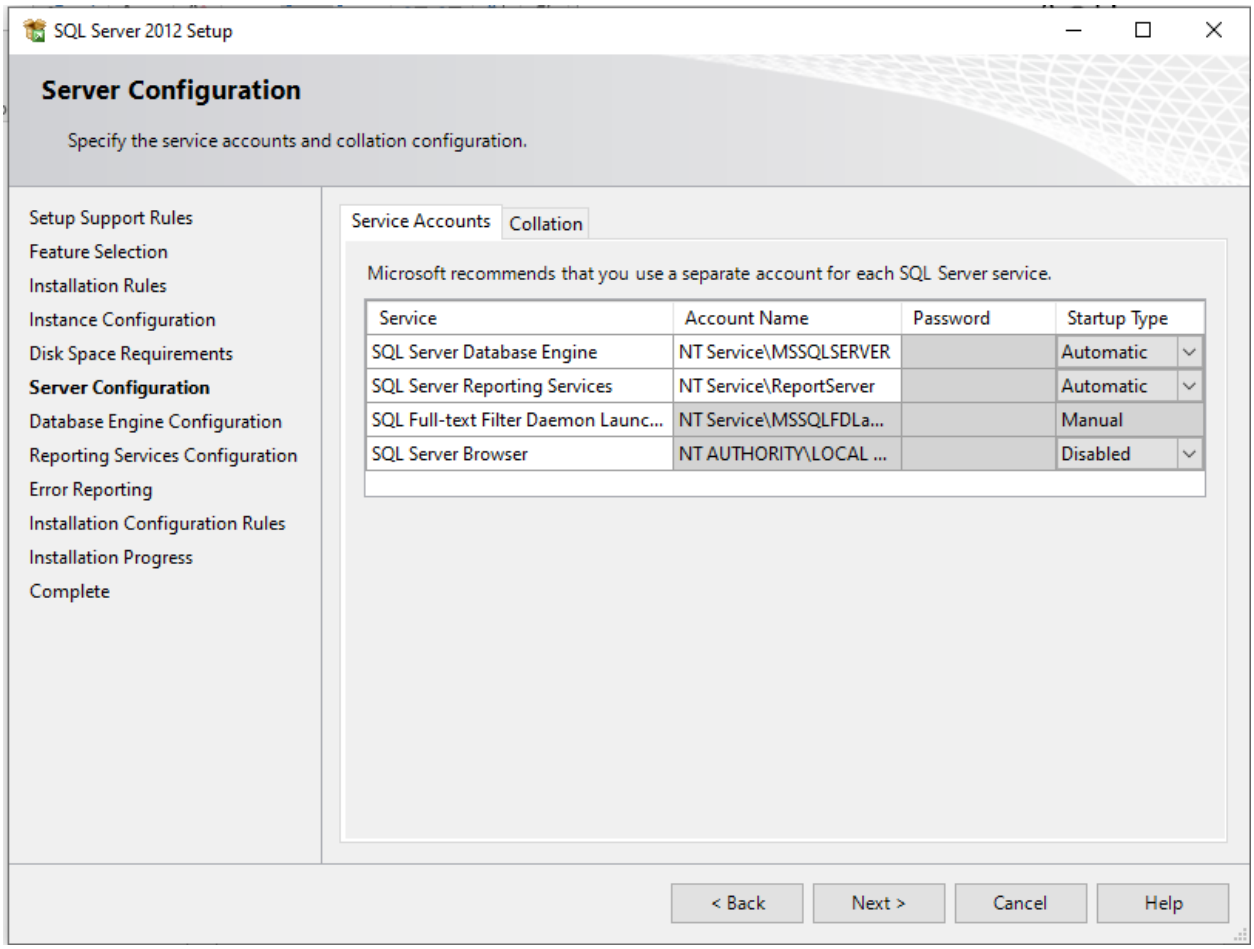

Figure 9: Specifying the service accounts and collation configuration

11. Select the authentication mode that you wish to use'. In this case Windows Authentication mode is used, which means the Windows user profile is used for authentication

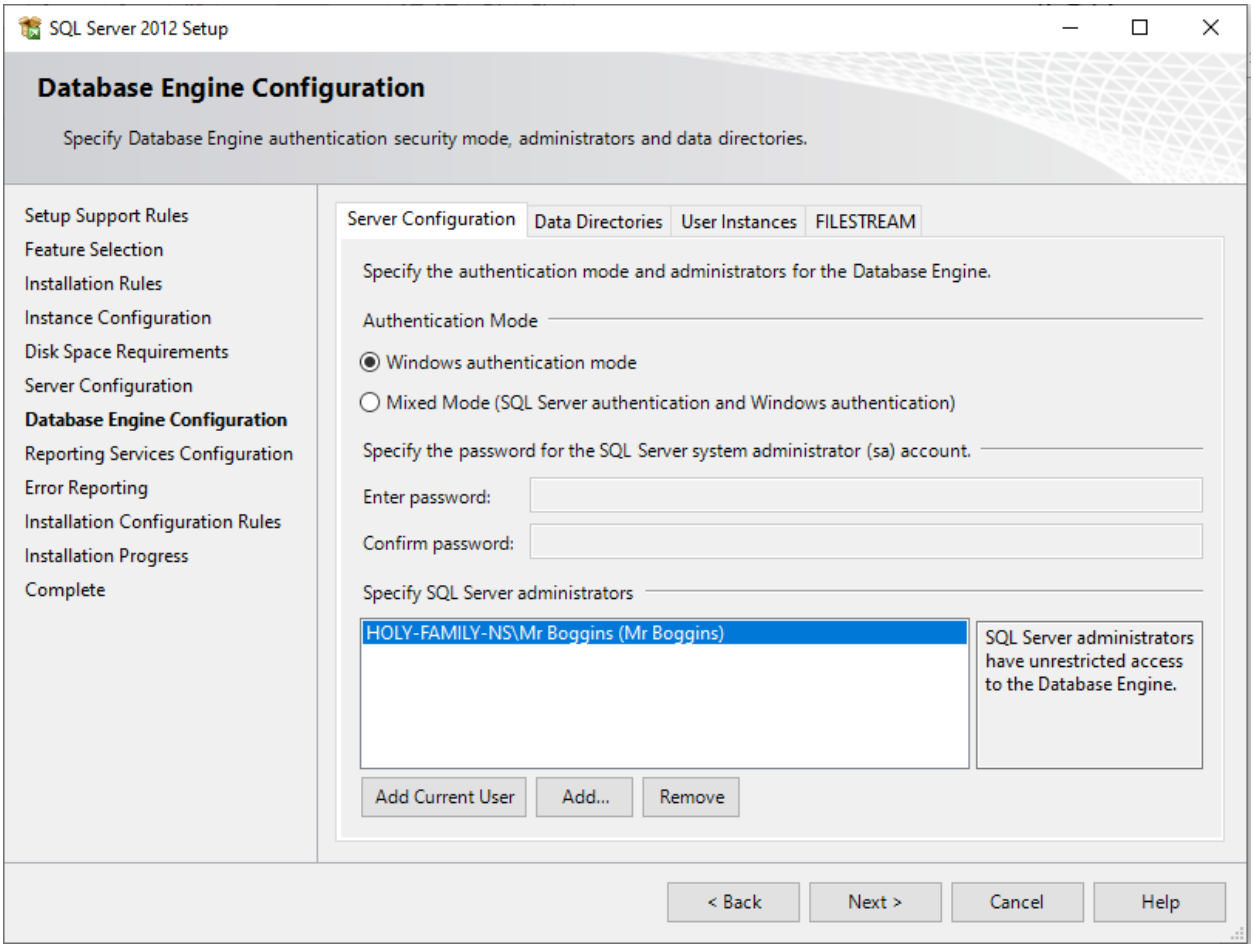

Figure 10: Database engine configuration

12. Select the ‘Install and Configure’ option then click ‘Next’

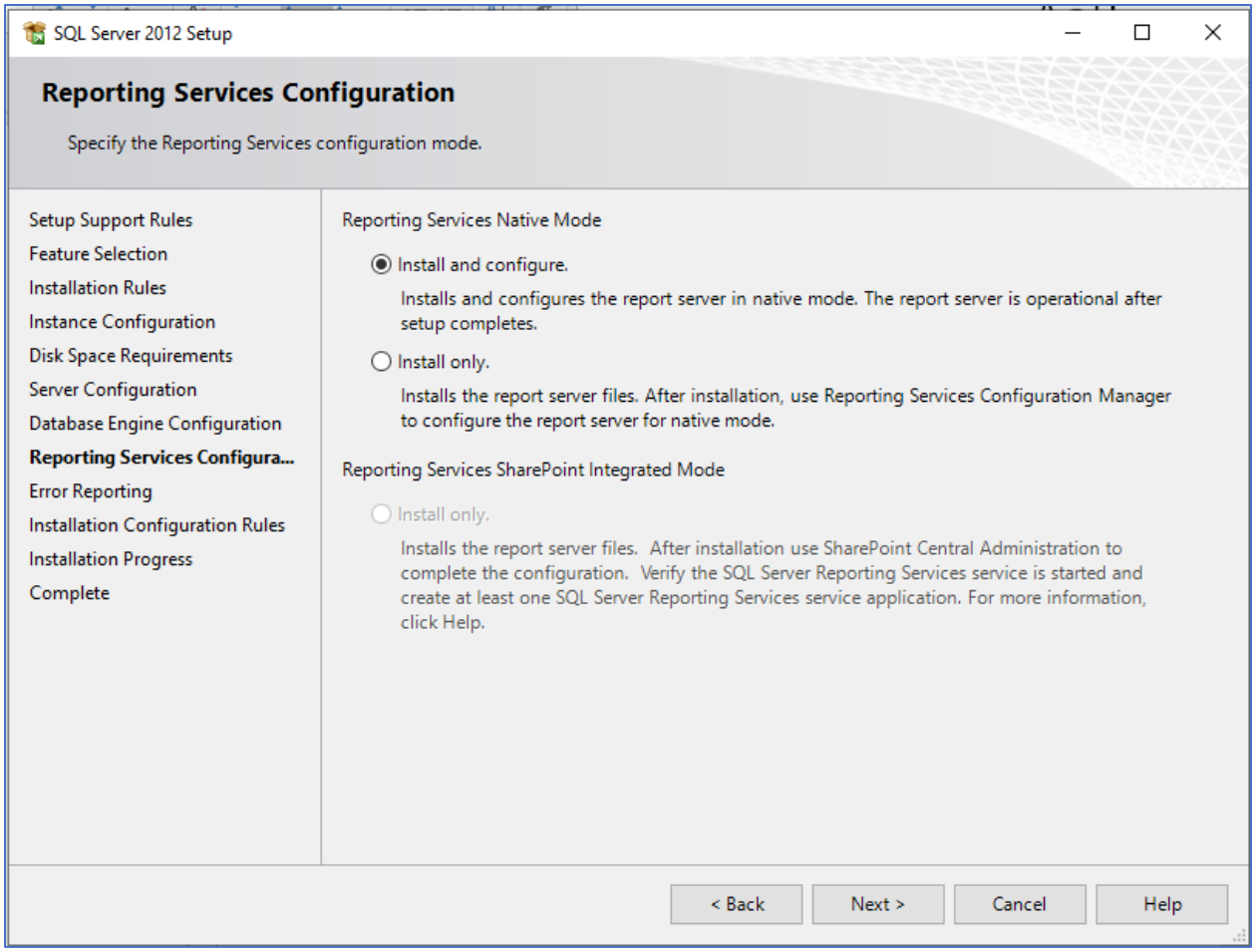

Figure 11: Configuration of reporting services

13. Click ‘Next’

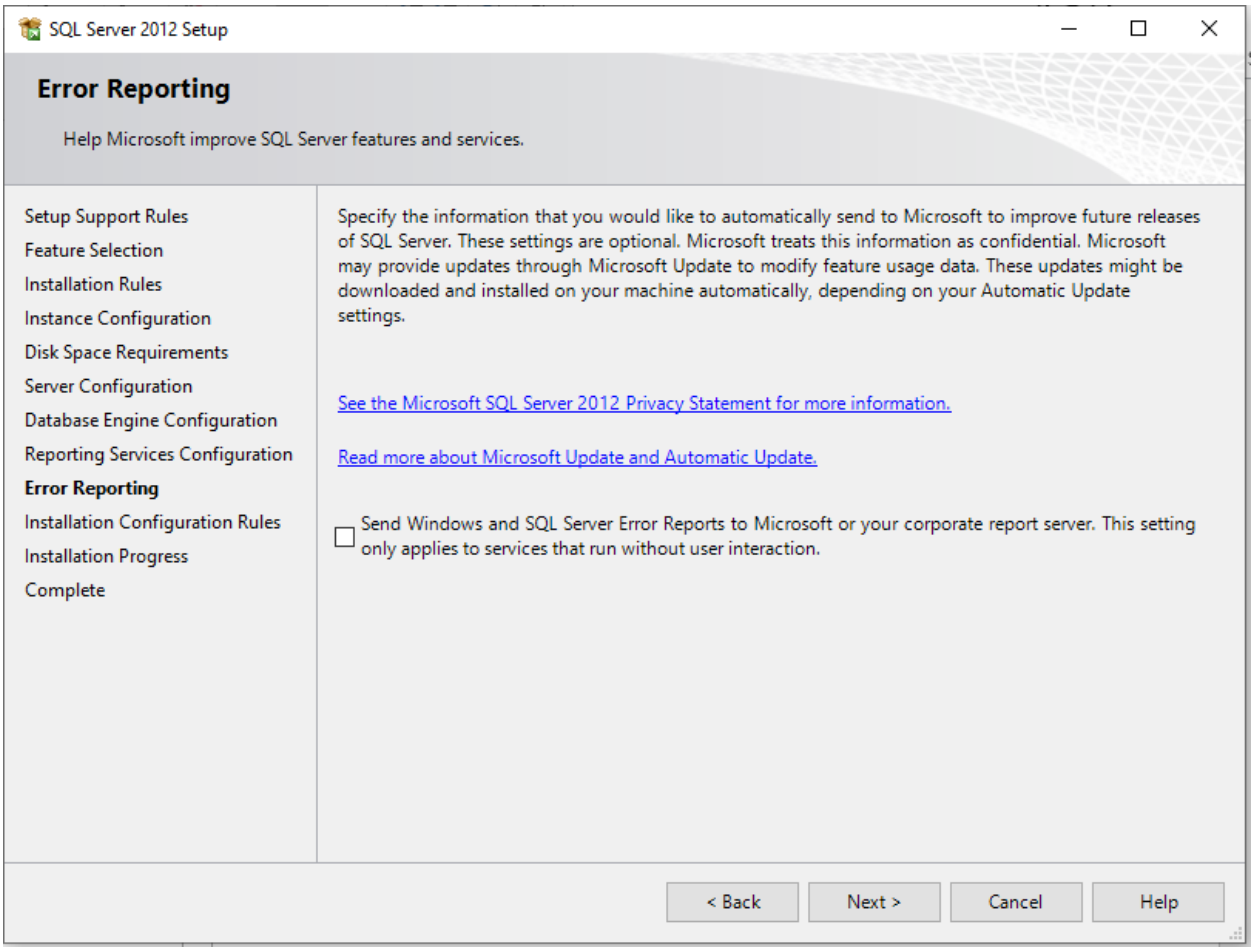

Figure 12: Error reporting screen

14. Click ‘Close’ after the installation has been completed

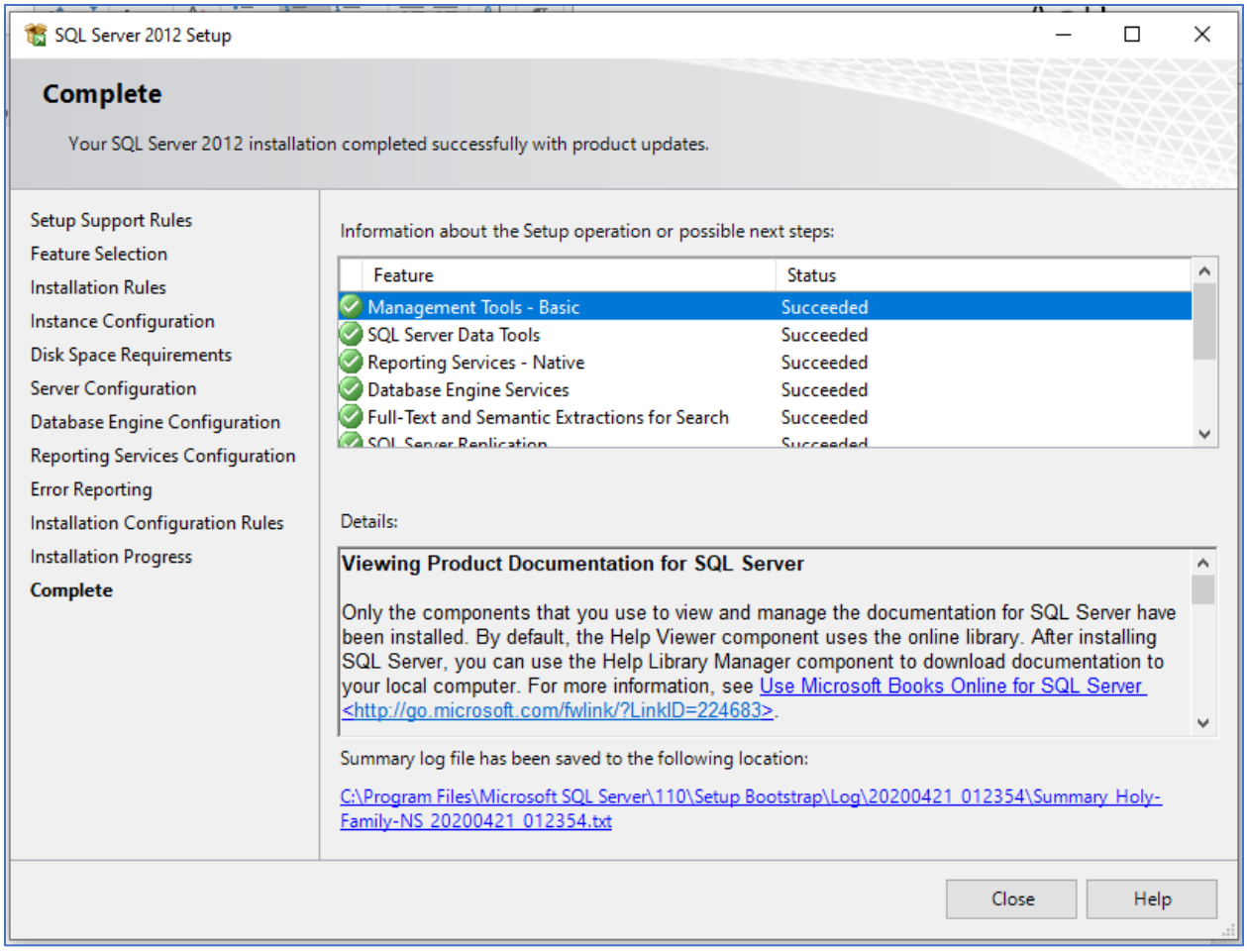

Figure 13: Completing installation of SSMS

15. Search for SQL Server in the start menu and click on ‘SQL Server Management Studio’

16. A popup screen appears asking if you want to import customized setting from 2008 SQL Server Management Studio. Click ‘Yes’

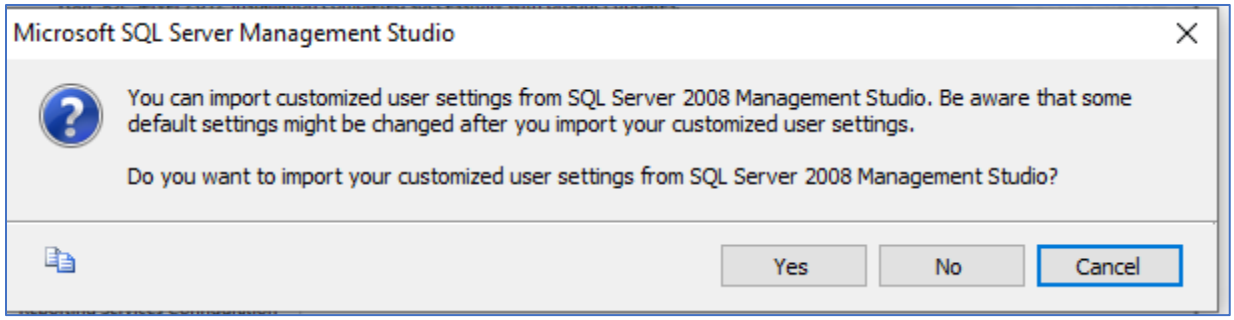

Figure 14. Importing settings from 2008 SSMS

17. Click on the ‘Databases’ dropdown to view all of the databases

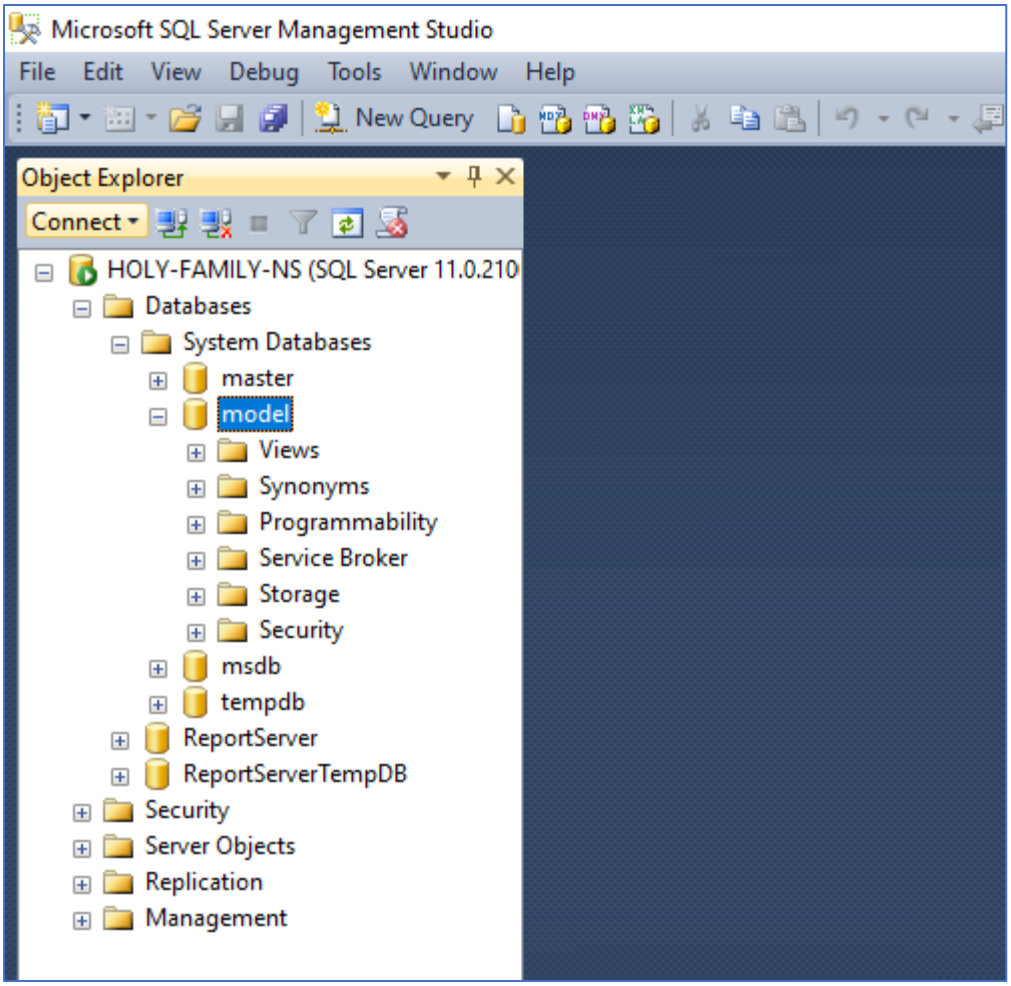

Figure 15: Viewing databases in SSMS

18. To create a new database right-click ‘Database’ and click ‘New Database’

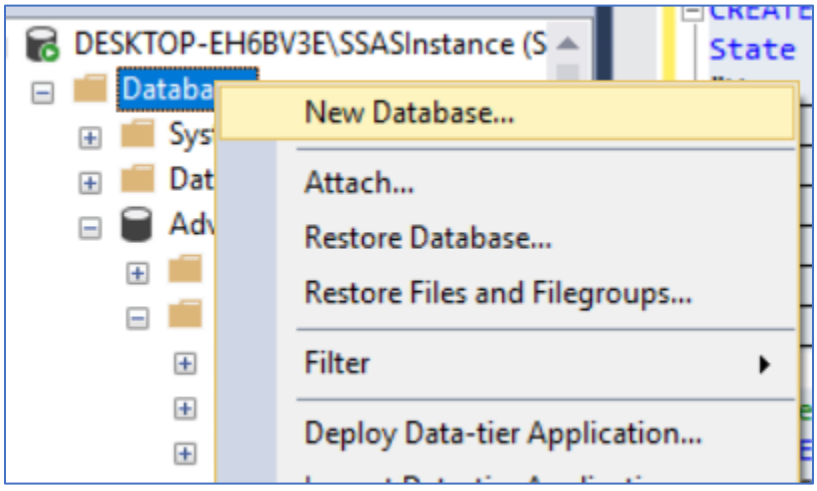

Figure 16: *Create a new database in SSMS*

19. To create a query right click on ‘Model’ and select ‘New Query’

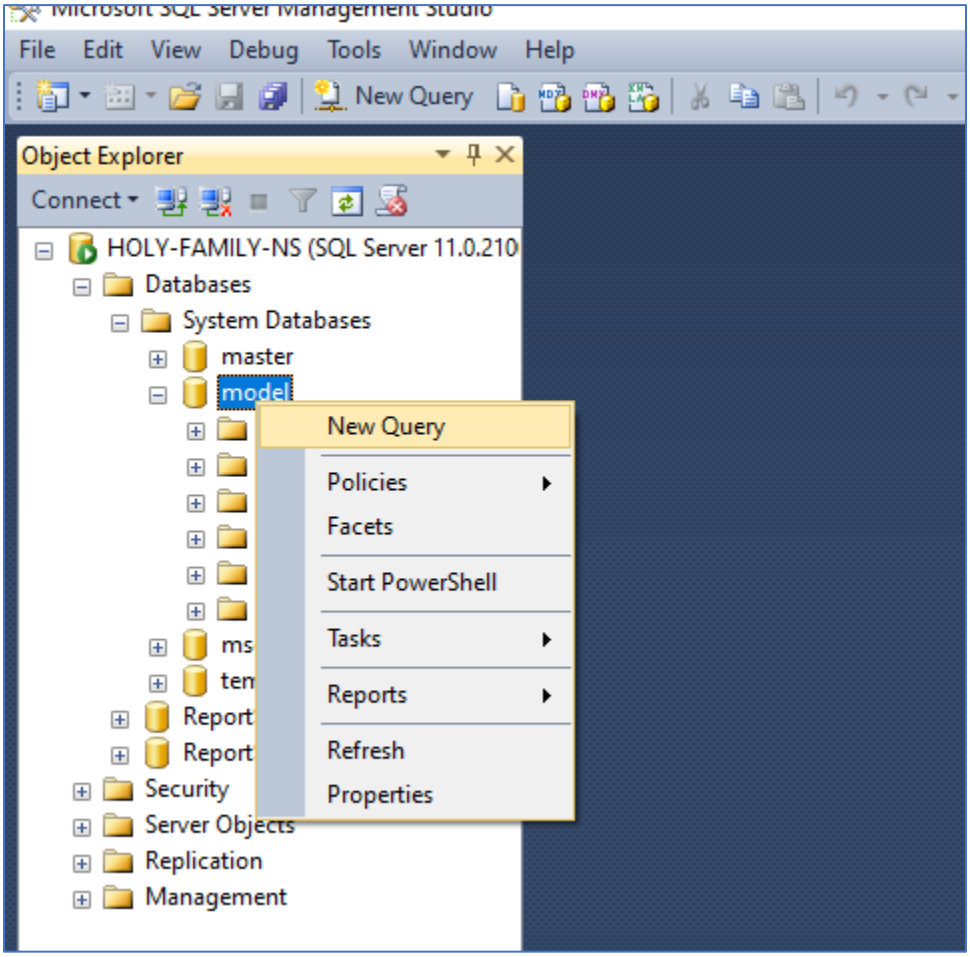

Figure 17: Creating new query in SSMS

Installing and Setting Up SSIS

1. Go to https://docs.microsoft.com/en-us/visualstudio/install/install-visual-studio?view=vs-2019

2. Click ‘Download Visual Studio’

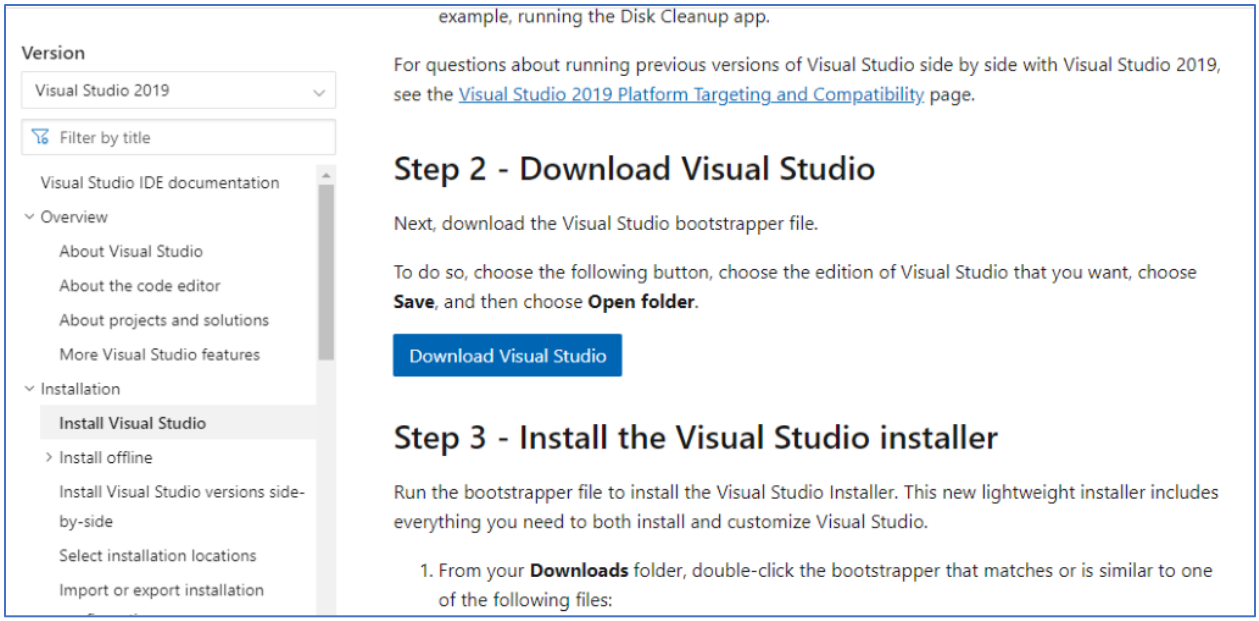

Figure 18. Download screen for Visual Studio

3. Click ‘Free Download’ under the ‘Community’ option

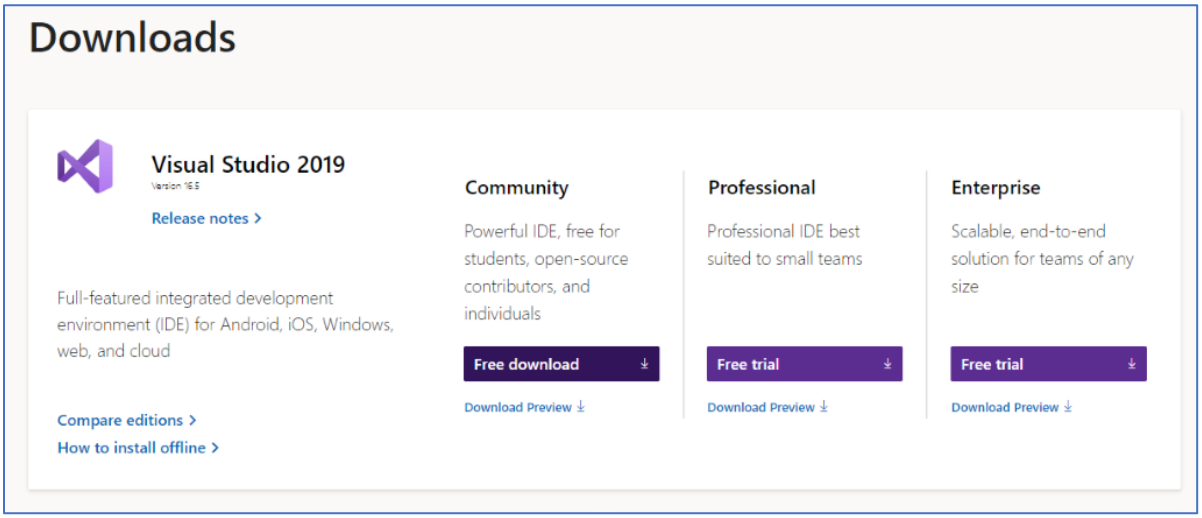

Figure 19: Downloading Community edition of visual studio

4. Click on the downloaded execution file

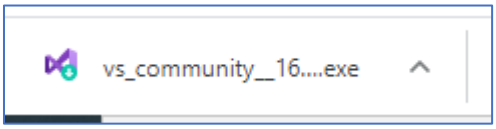

Figure 20. Downloaded visual studio file

5. The popup screen for configuring the Visual Studio Installer should appear. Click ‘Continue’

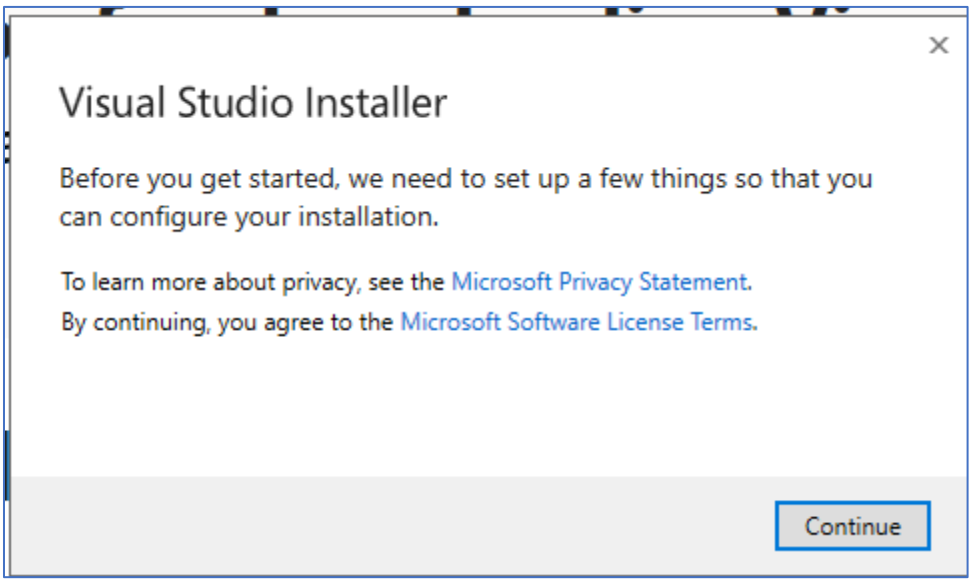

Figure 21. Screen for configuring visual studio installer

6. Click on the ‘Data storage and processing’ option

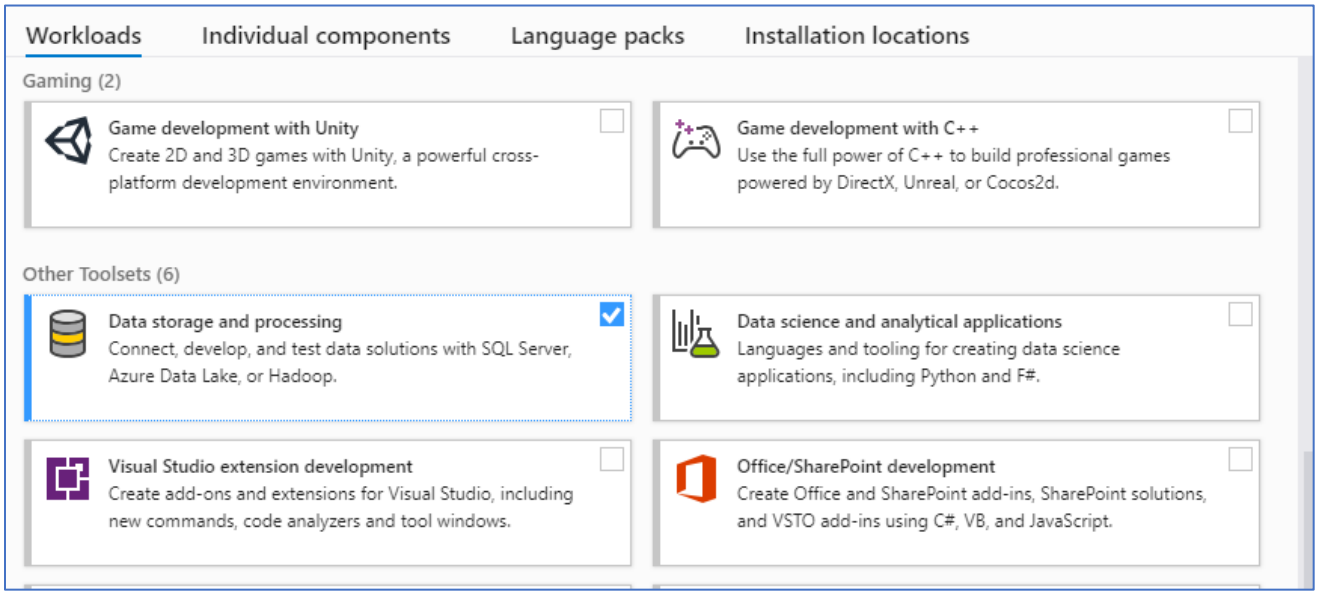

Figure 22: Selecting data storage and processing option

7. Click ‘Install while downloading’ next to the install button, then click ‘Install’

Opening Visual Studio and SSIS

1. Click on the start screen and scroll down to and select ‘Visual Studio 2019’

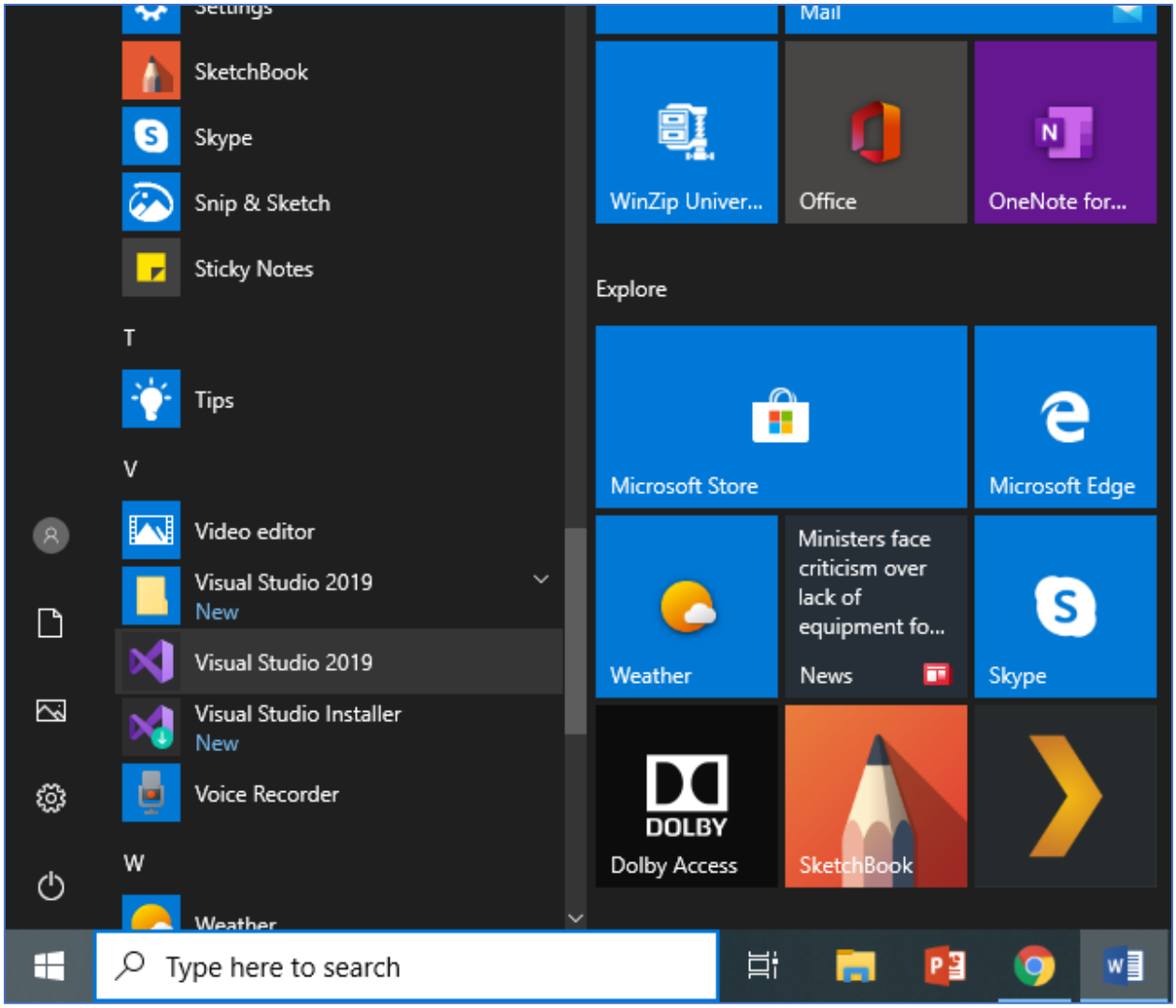

Figure 23: Selecting visual studio from Windows menu

2. The sign in screen for visual studio appears. Sign in with a Microsoft account. If you do not have a Microsoft account create one by clicking the ‘Create One!’ button. The steps for creating a Microsoft account guides you through creating an account.

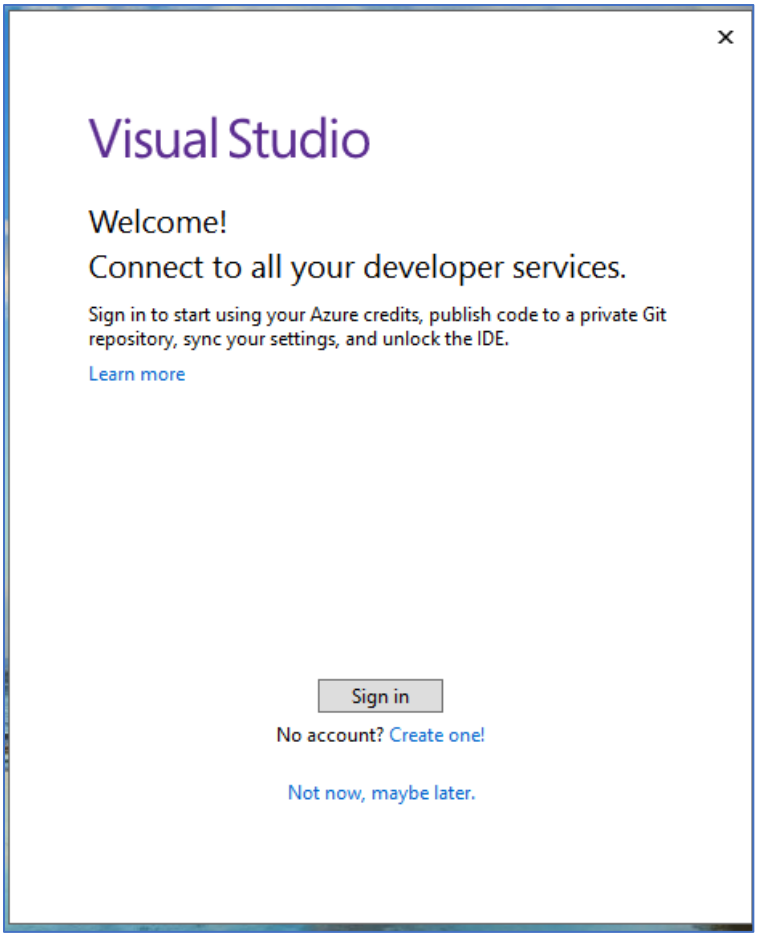

Figure 24: Signing into visual studio

3. Sign in with your Microsoft account

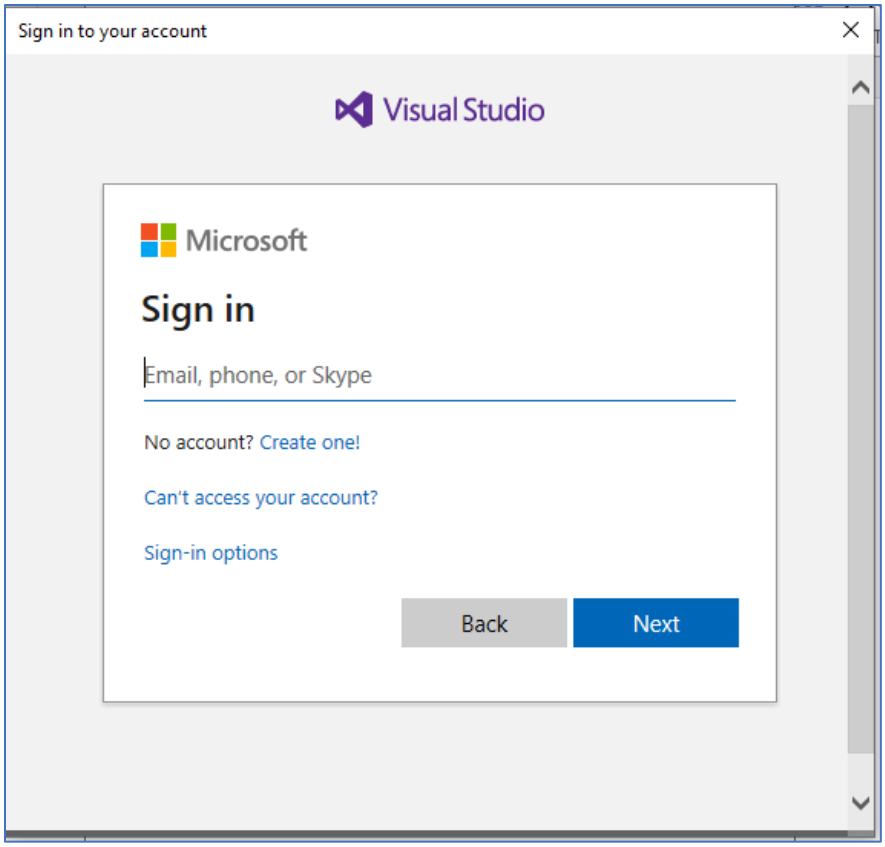

Figure 25: Entering Microsoft email address for visual studio sign in

4. Enter your Microsoft password

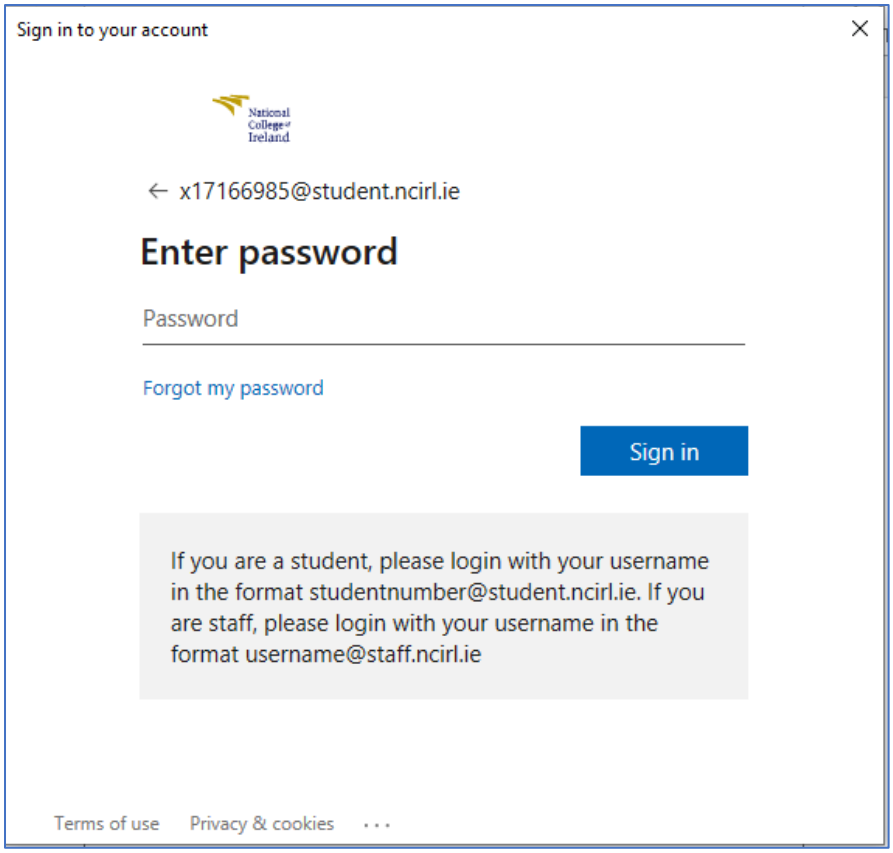

Figure 26: Entering password for Microsoft account

5. Set the default colour settings, then click “Start Visual Studio”

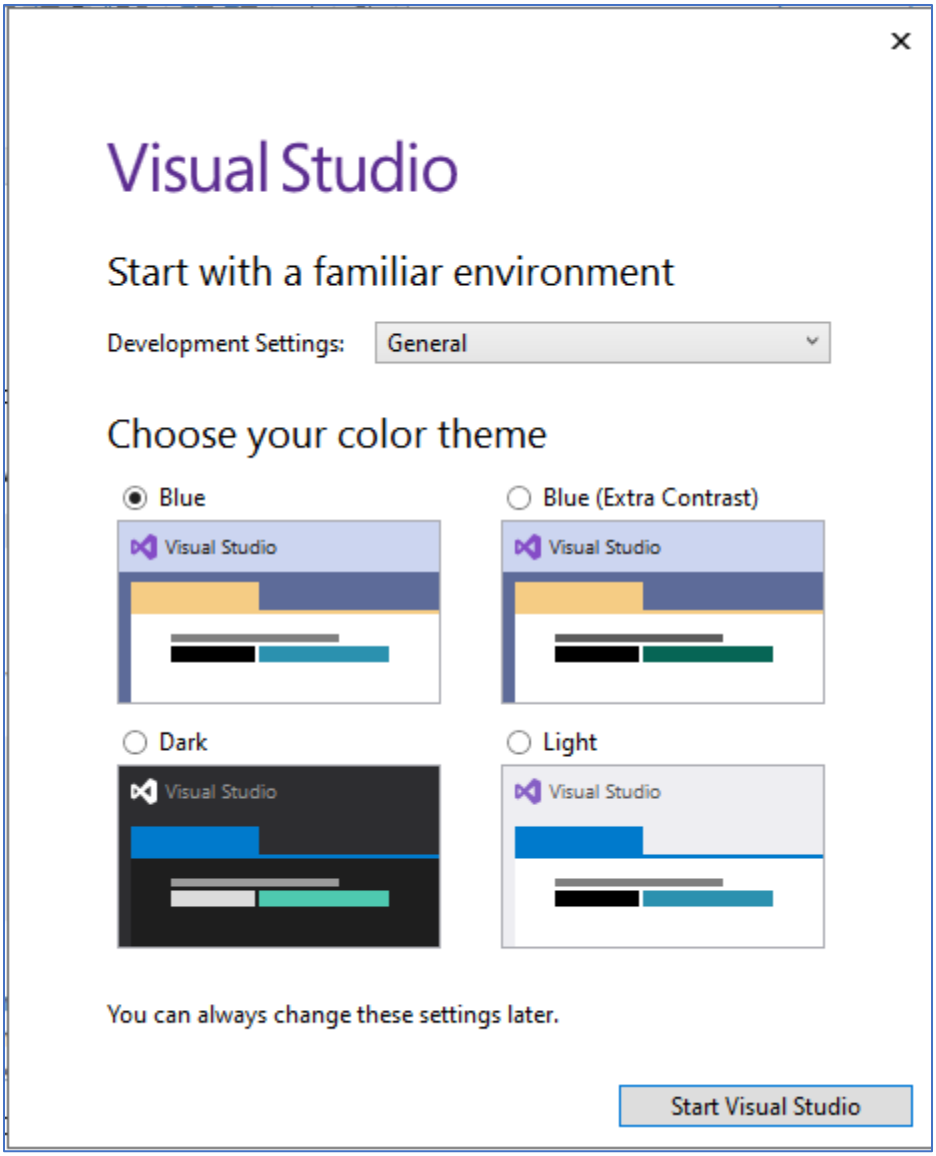

Figure 27: Setting default themes

6. Click ‘Install’

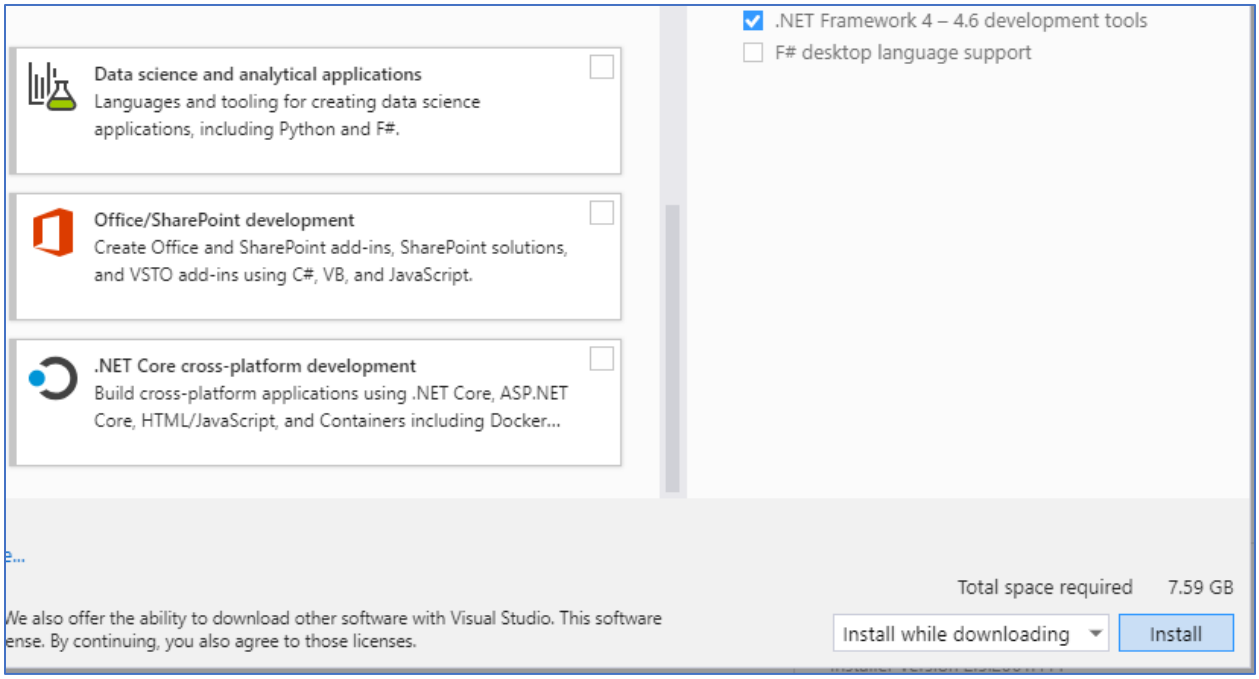

Figure 28: Installing data storage and processing in Visual Studio

7. Once the installation is complete a popup screen appears that says the system requires a reboot for completion of installation. Click 'Restart'

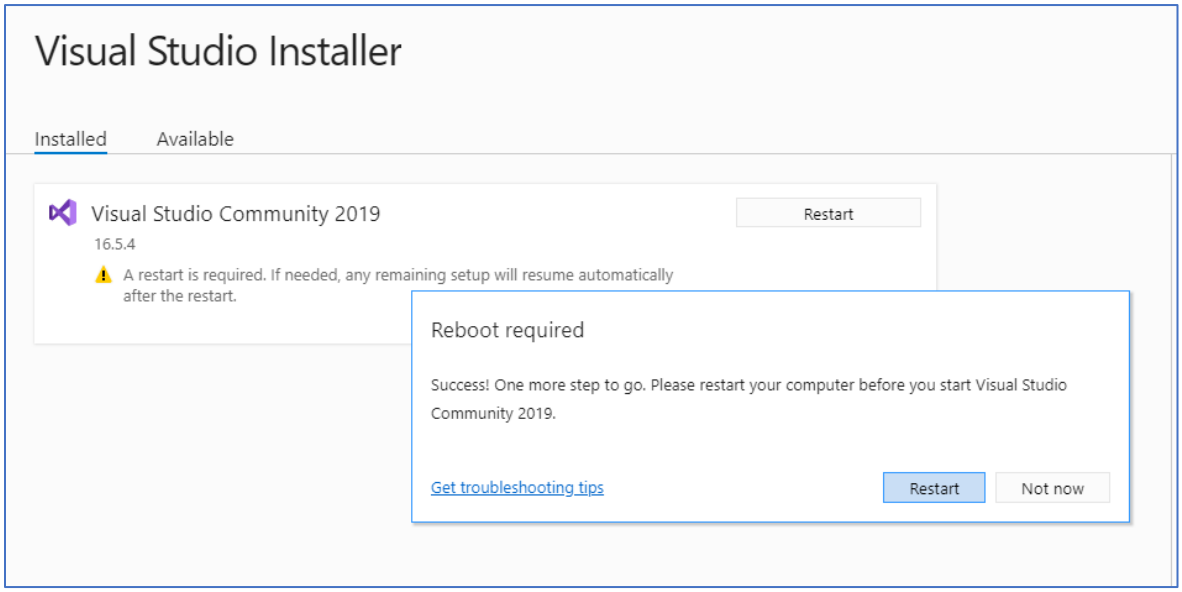

Figure 29: Reboot popup for SSIS

Installing R

For Mac

1. Download R from https://cran.r-project.org/bin/macosx/
2. Click on the appropriate R package for your system (catalina is the most up to date version)

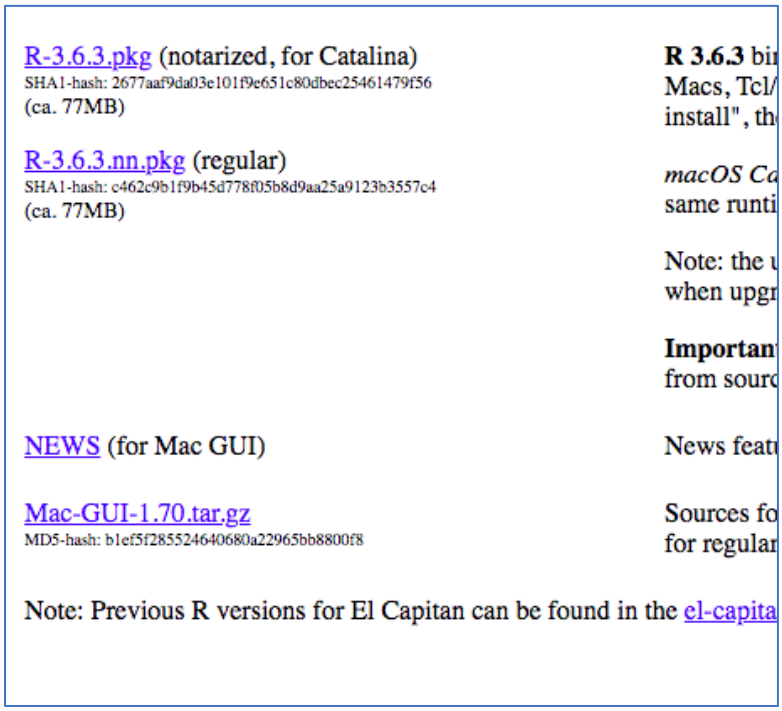

Figure 30: Selecting R package for Mac

3. When the introduction screen for R appears, click 'Continue'

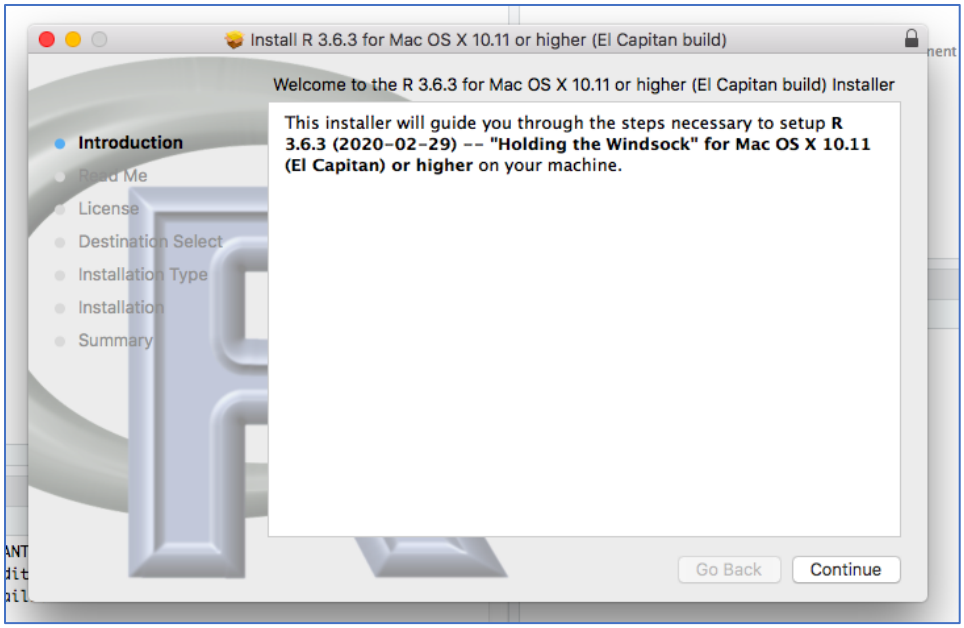

Figure 21: R introduction screen

4. Read the Read Me then click continue

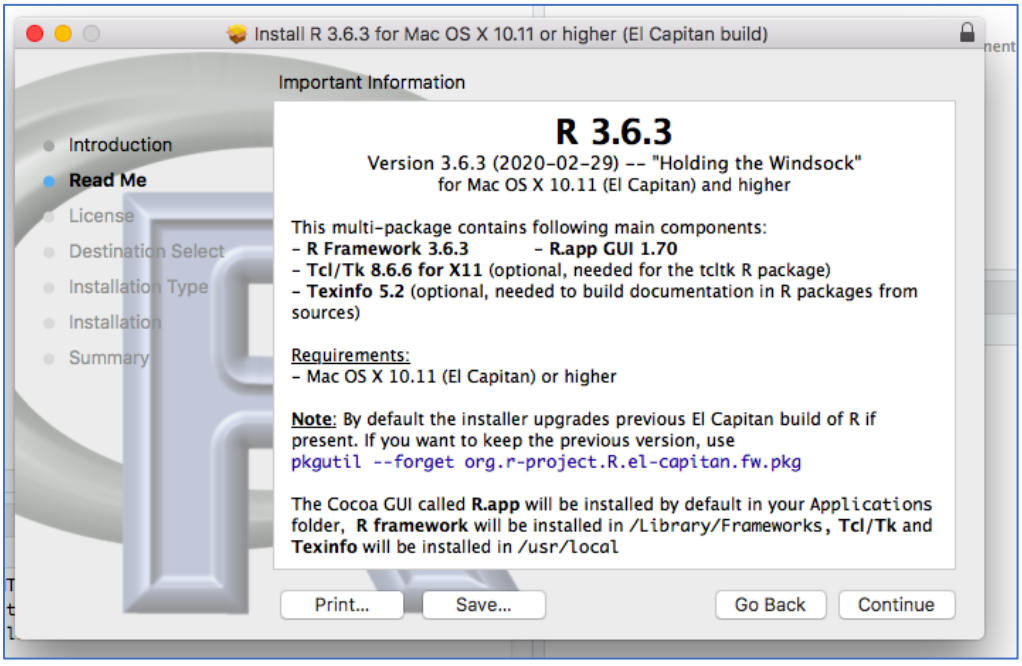

Figure 32: Read Me in R setup

5. Read and agree to the licence then click continue

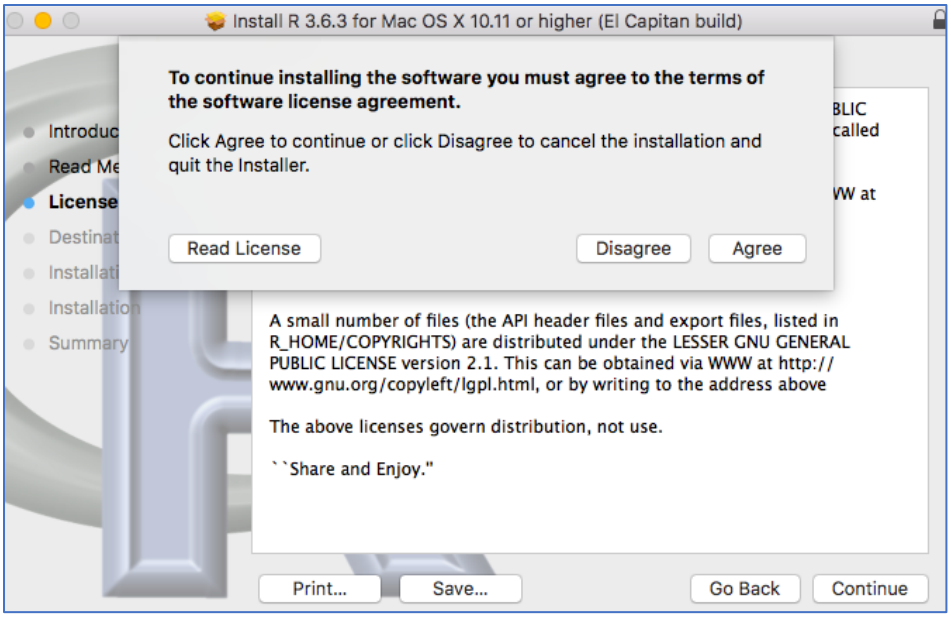

Figure 33: Agreeing to the license

6. Click ‘Install’

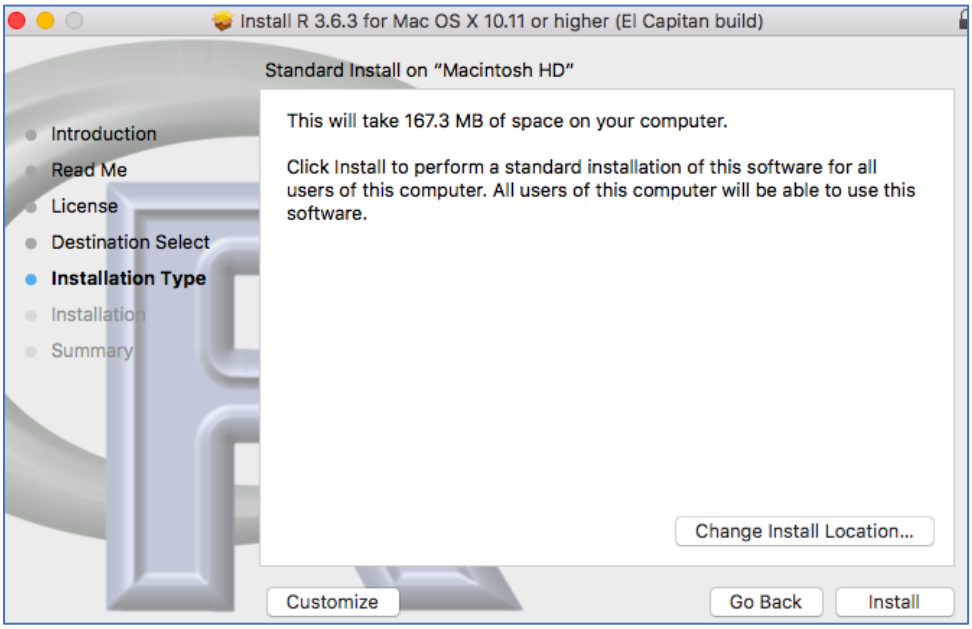

Figure 34: Installing r on Mac

7. The success message should appear

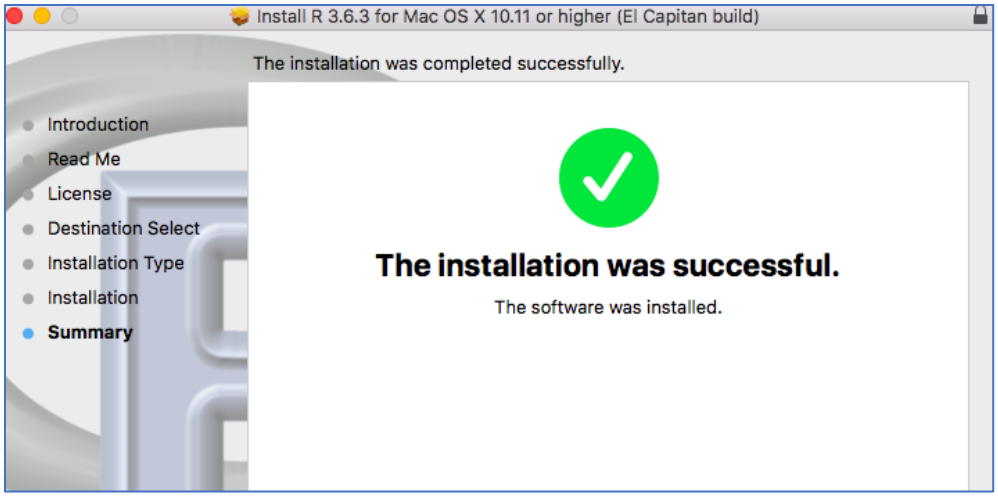

Figure 35: Success message

For Windows

1. Download R from https://cran.r-project.org/bin/windows/base/
2. Click on the 'Download R 4.0.2 For Windows' button

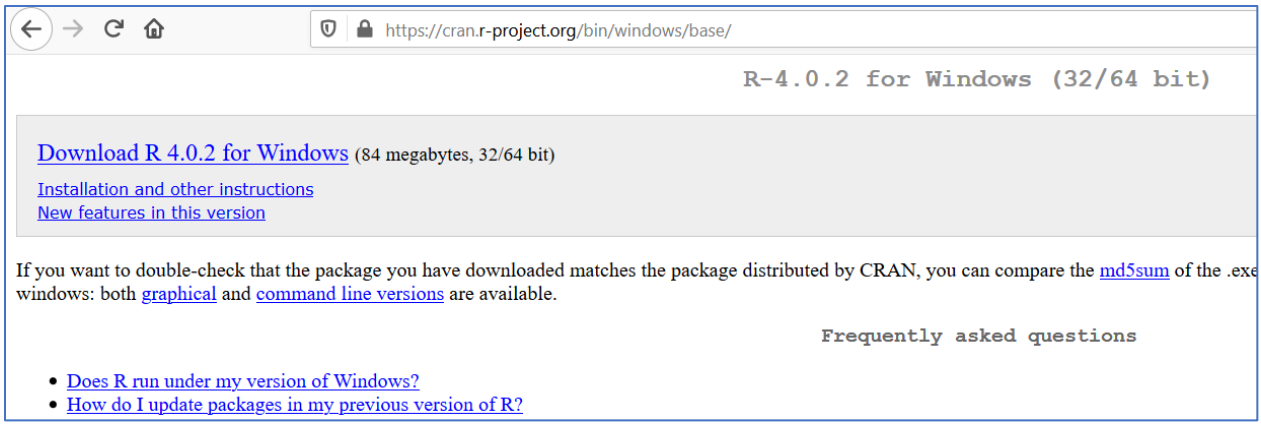

Figure 36: Downloading R for Windows

3. Click 'Save File'

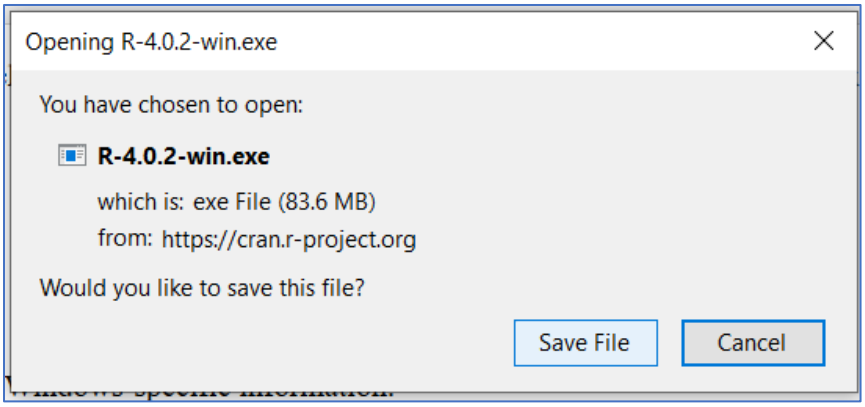

Figure 37: Saving R download file

4. Select the language

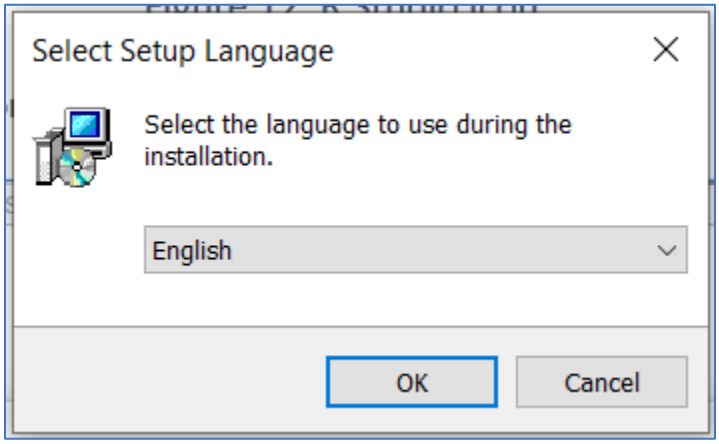

Figure 38: Selecting the language for R

5. Read the information sheet and click next

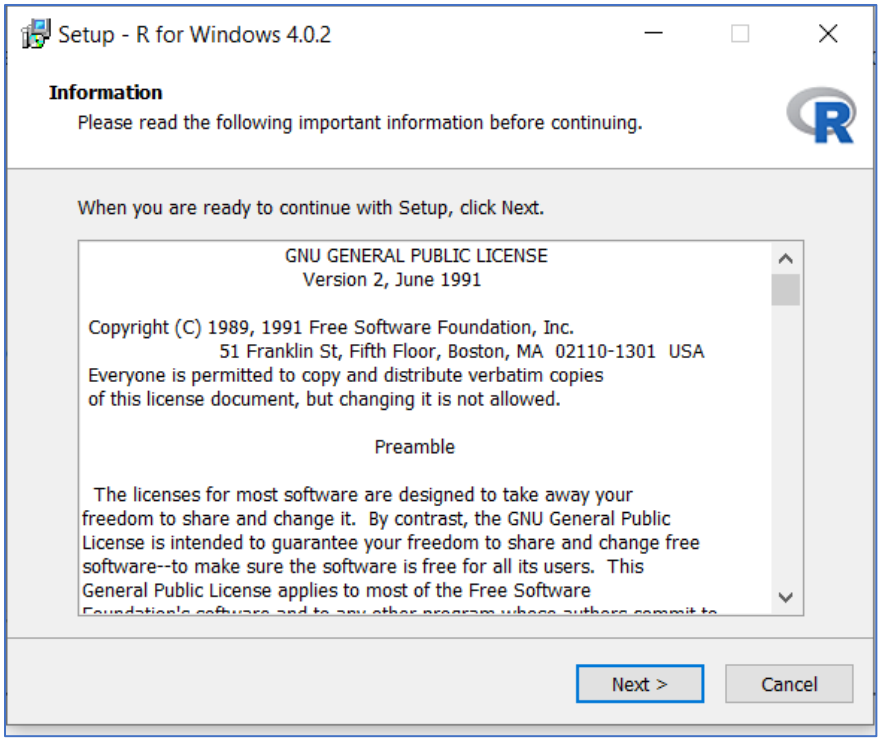

Figure 39: R the information sheet

6. Choose the location where you want to save the file then click ‘Next’

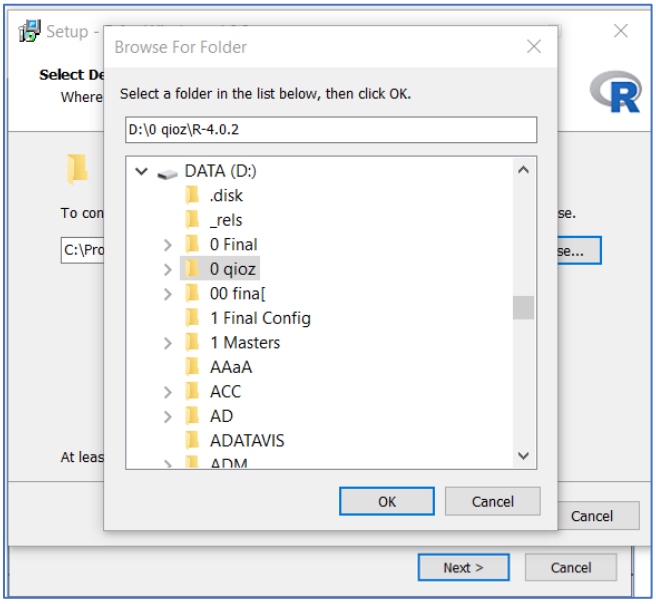

Figure 40: Choosing location of folder to save R

7. Choose the 32-bit or 64-bit files depending on your computer, then click next

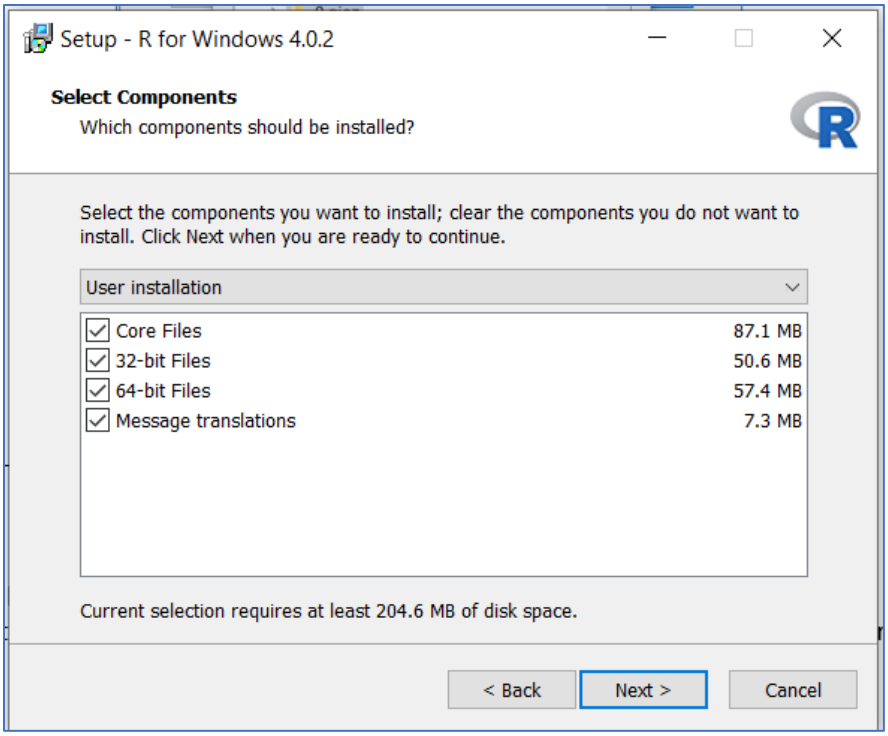

Figure 41: Choosing R file type

8. Choose ‘No (accept defaults)’ then click ‘Next’

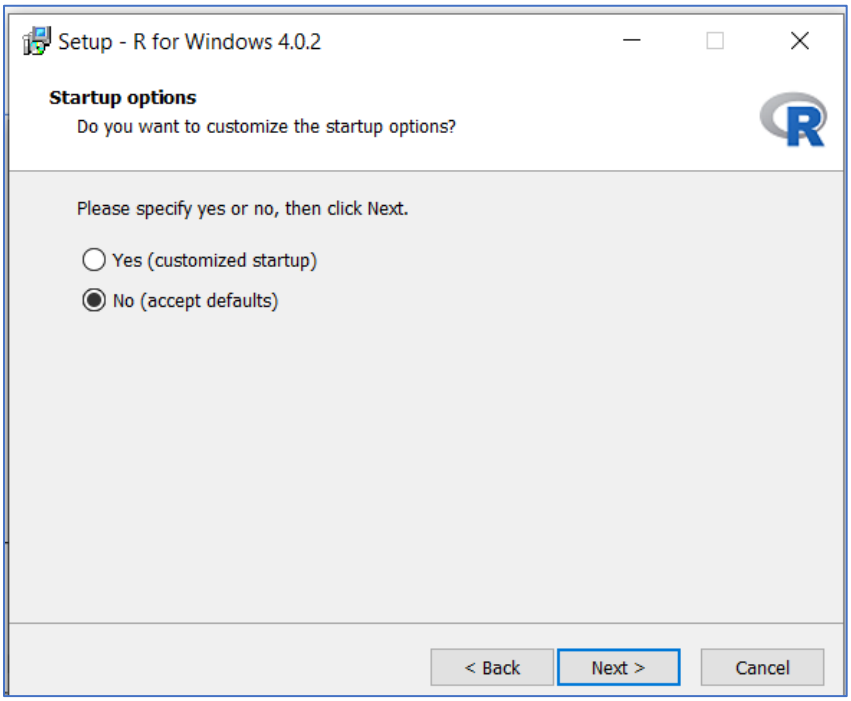

Figure 42: Accepting defaults in R for Windows

9. Select your start menu folder

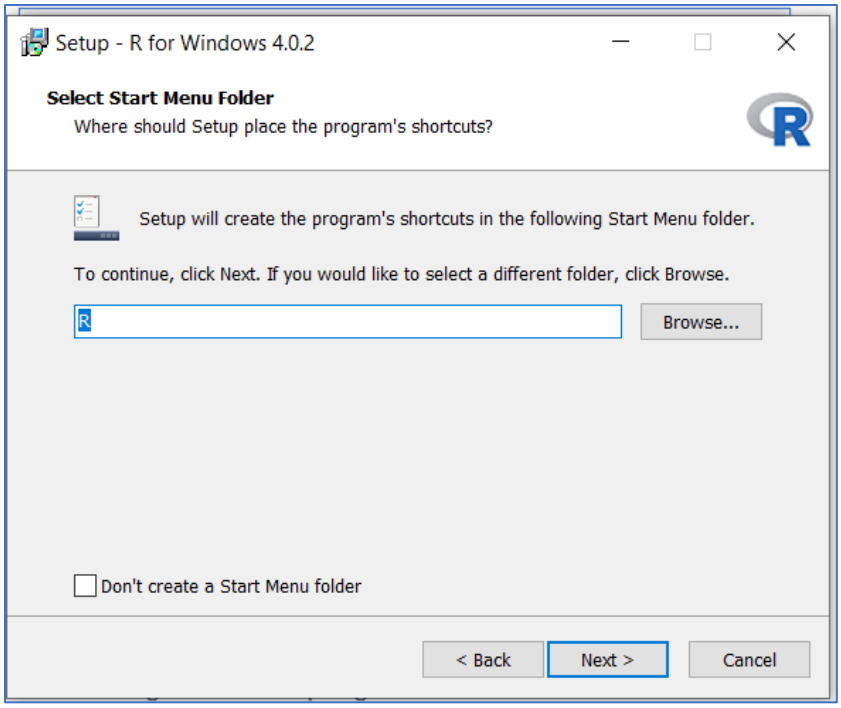

Figure 43: Selecting start menu folder

10. Select additional tasks (add a desktop shortcut if you want)

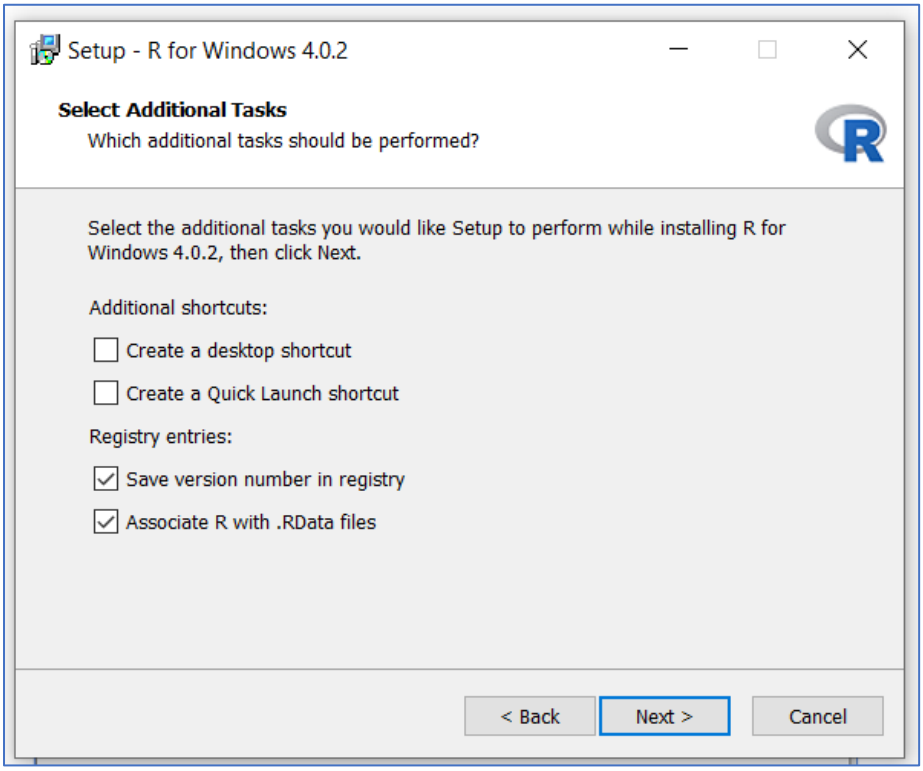

Figure 44: Selecting additional tasks

11. Click ‘Finish’

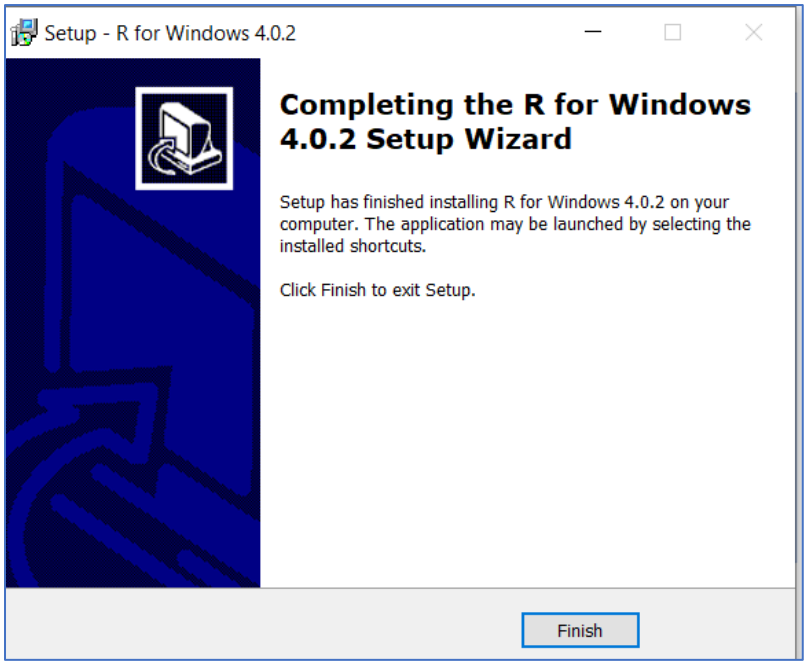

Figure 45: Finishing R setup on Windows

Downloading R Studio

For Mac

1. Download R Studio from https://rstudio.com/products/rstudio/download/
2. Select the ‘download’ button under the ‘RStudio Desktop, Open Source License, Free’ text

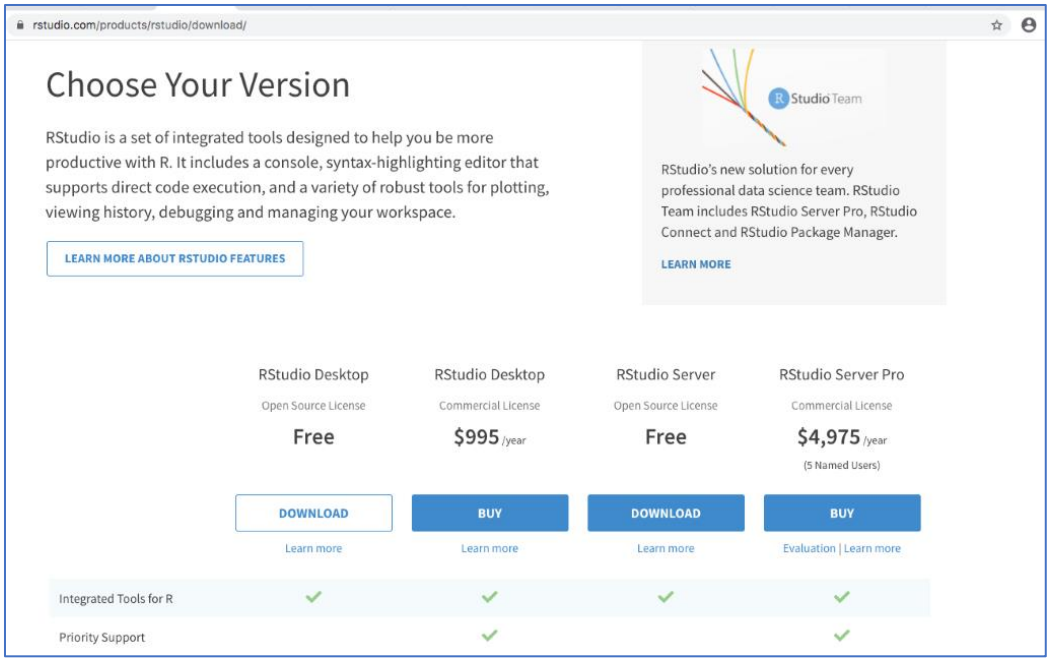

Figure 46: Choosing the free R studio version

3. Click 'Download RStudio for Mac'

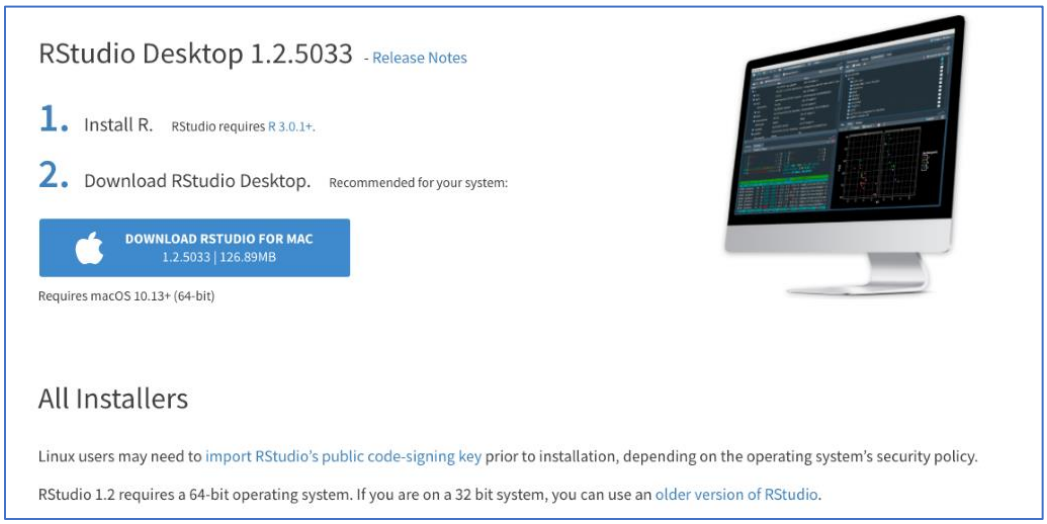

Figure 47: Downloading R studio desktop for Mac

4. Click on the downloaded .dmg file

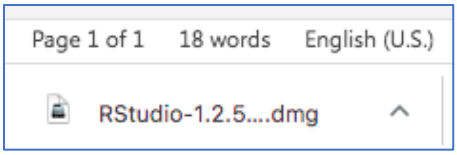

Figure 48: R Studio .dmg file

5. The R Studio icon should appear wherever you downloaded it

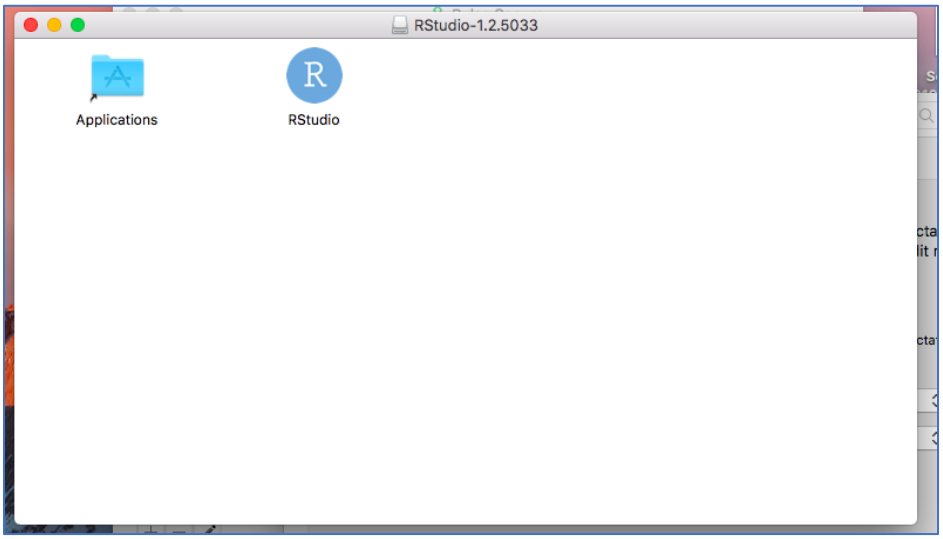

Figure 49: R Studio icon

6. Click the R Studio icon then click open (the file is safe)

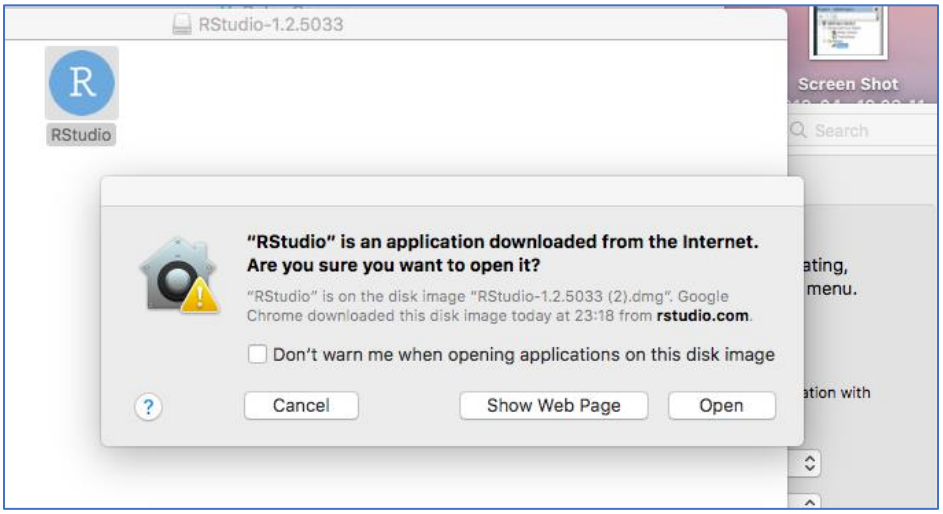

Figure 50: Opening R Studio on Mac

For Windows

1. Go to https://rstudio.com/products/rstudio/download/
2. Click on 'Download' under 'R Studio desktop FREE'

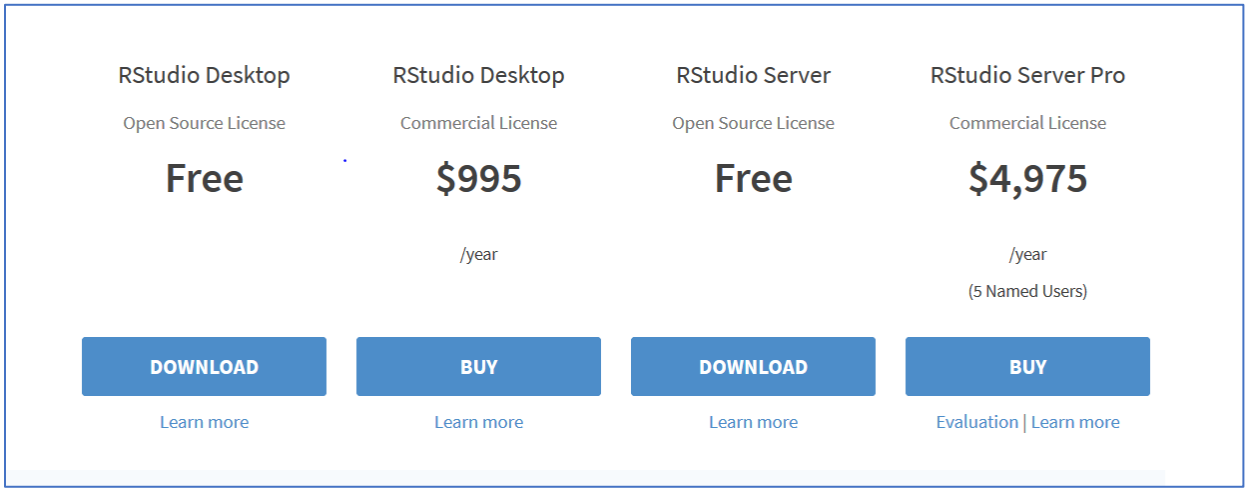

Figure 51: Downloading the free version of R Studio desktop

3. Click 'Download R Studio for Windows'

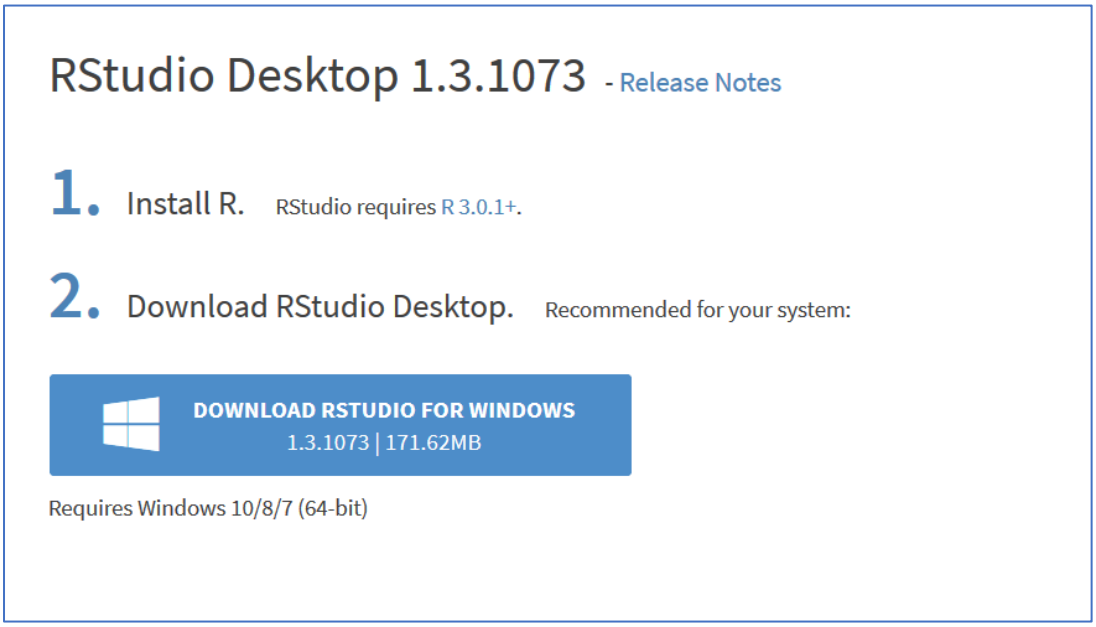

Figure 52: R Studio for Windows download

4. Click 'Save File'

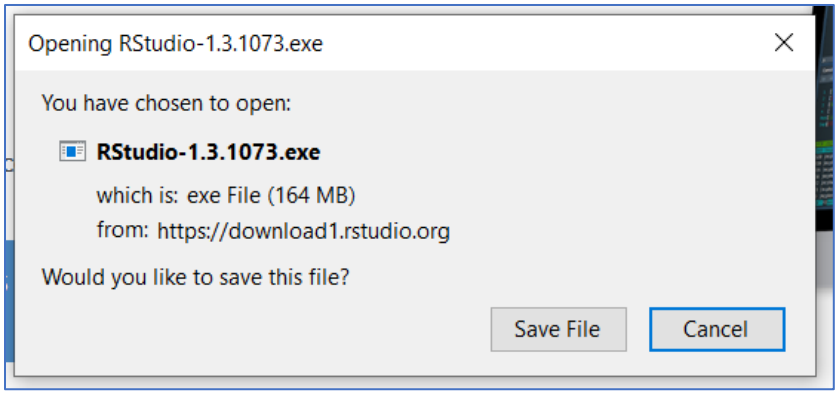

Figure 53: Saving R studio file in Windows

5. Open the downloaded file
6. Click ‘Next’

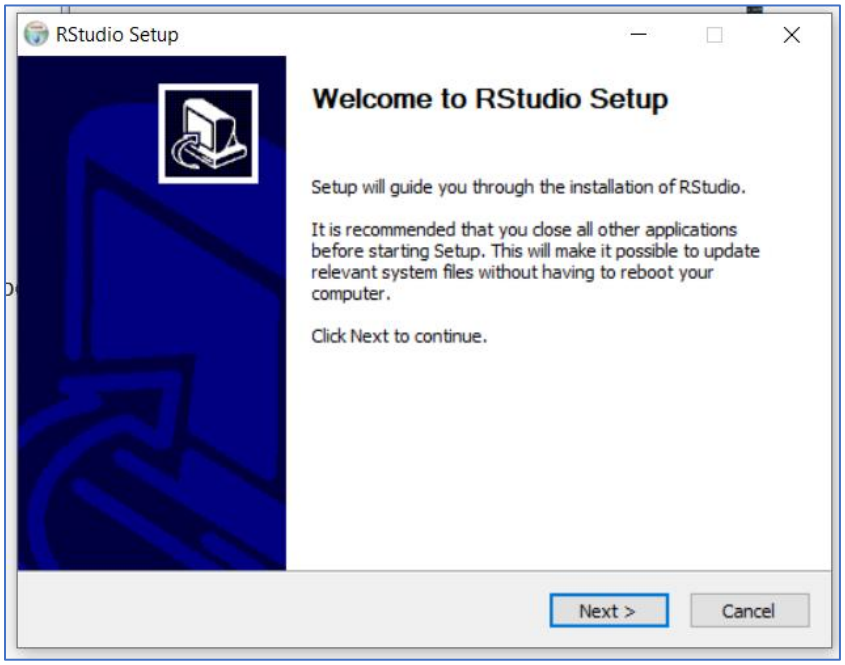

Figure 54: R Studio setup welcome screen

7. Choose the folder in which to download R Studio, then click ‘Ok’ then ‘Next’

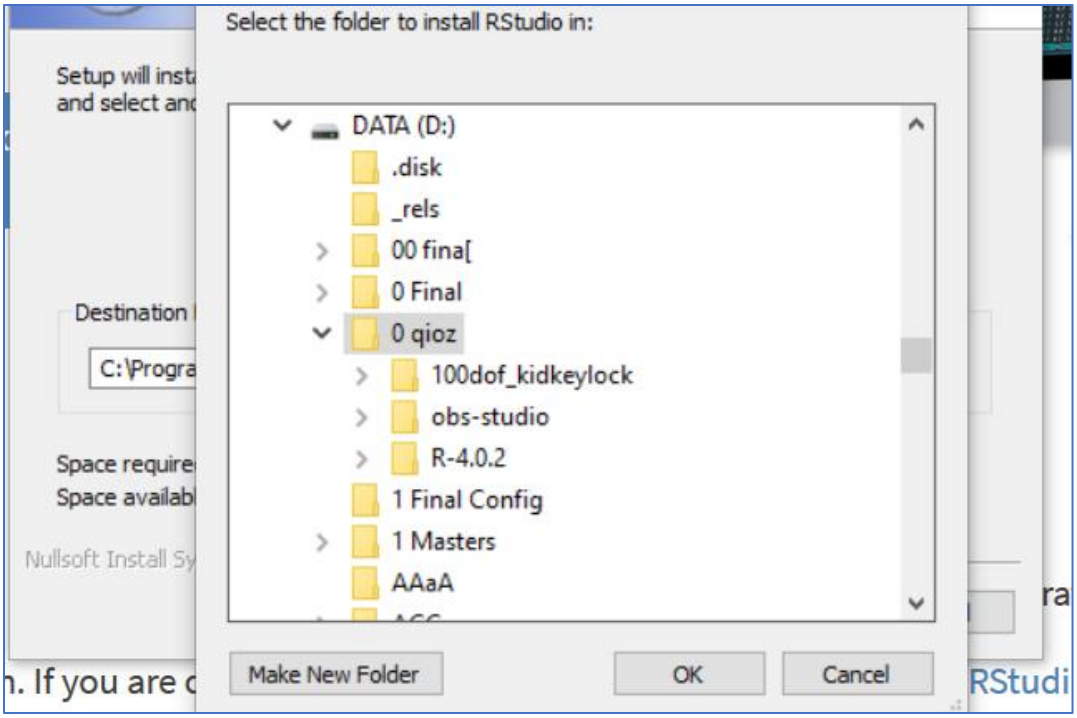

Figure 55: Choosing where to download r studio

8. Choose your start menu folder then click ‘Install’

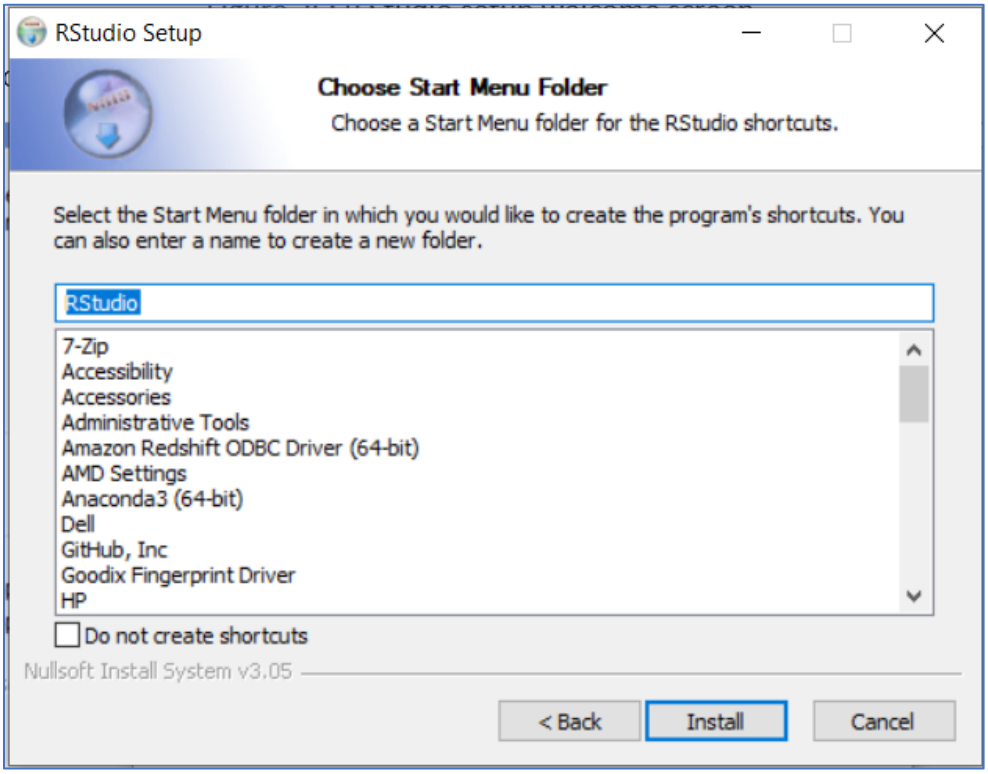

Figure 56: Installing R Studio

R Code

Twitter Authentication

1. Go to https://developer.twitter.com/en

2. Sign into (or if necessary click 'Sign Up' and follow the instructions to sign up for an account) your Twitter account

3. Click on 'Apply for a developer account'

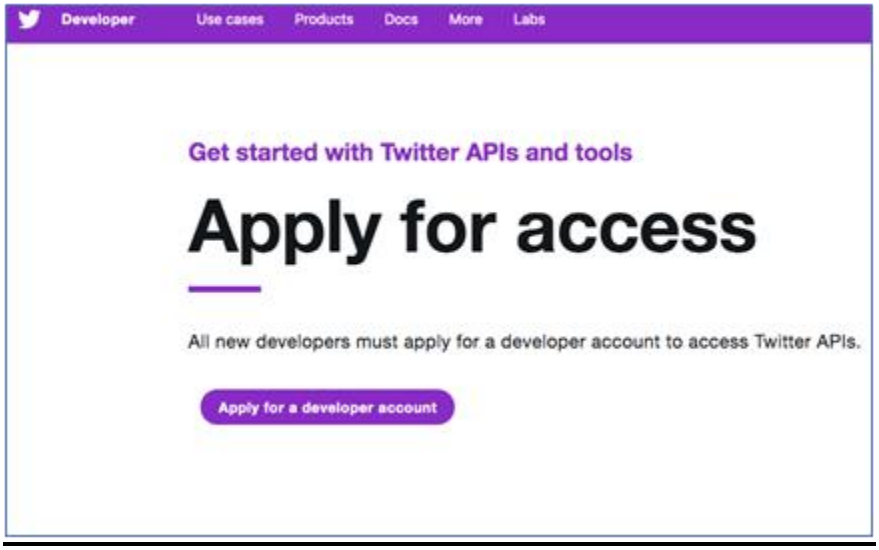

Figure 57: Applying for Twitter access

4. Click on the use case boxes (in this case 'Student' was chosen') then click 'Next'

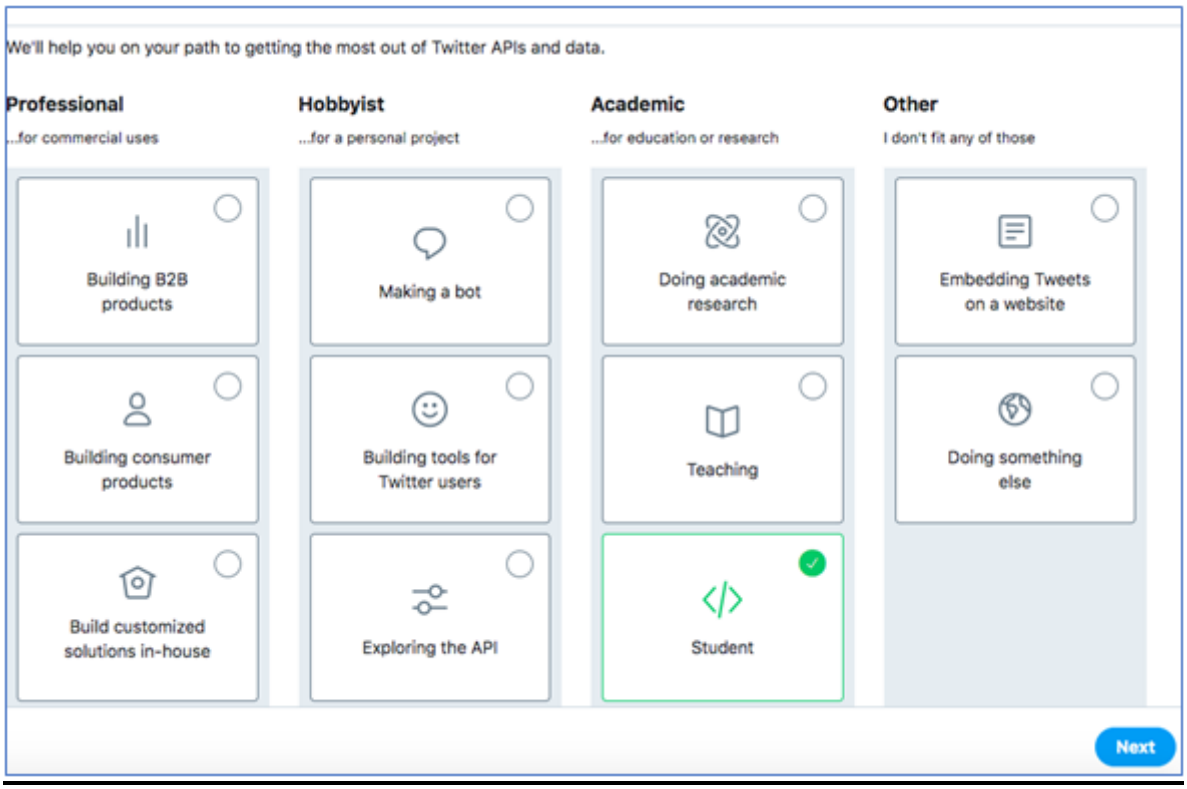

Figure 58: Twitter use case

5. Enter in the required details then scroll to the bottom and click 'Next'

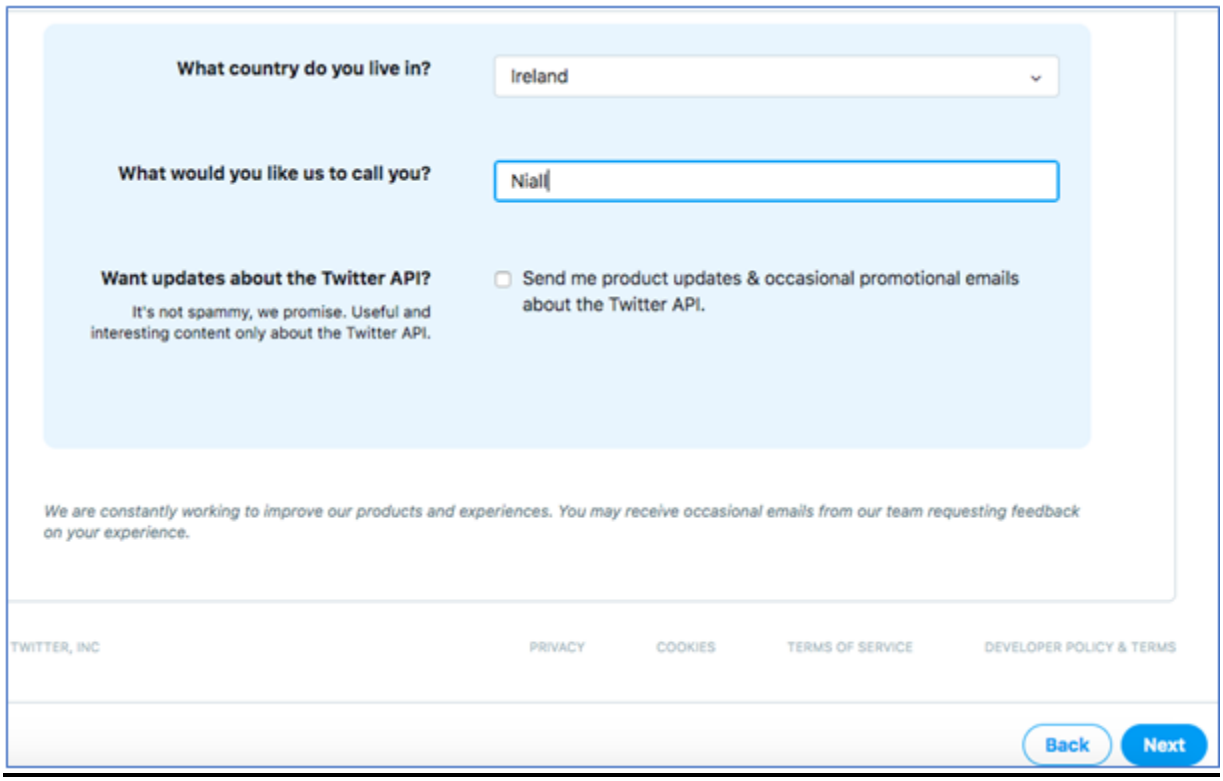

Figure 59: Entering details

6. Enter an explanation for how you will use the API. Here is a sample that you may use: 'I will us it for research purposes. The project I am currently doing involves scraping tweets from Twitter and analysing these tweets using sentiment analysis and token analysis. This is with the aim of studying attitudes towards vaccines.'

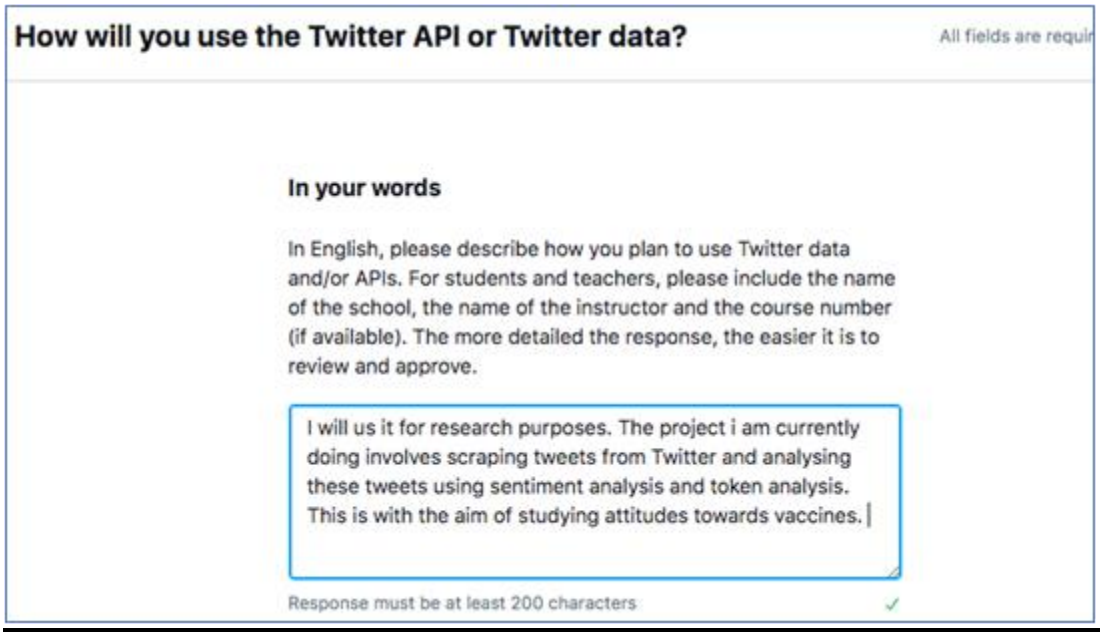

Figure 60: Explaining use case

7. Fill in the section for how you will analyze the Twitter data. Here is an example that may be used: 'The twitter data will be analysed using sentiment analysis and unnesting of tokens, which will be used for analysis and for the generation of visual reports. This is with a view to analysing vaccine related data from Twitter'

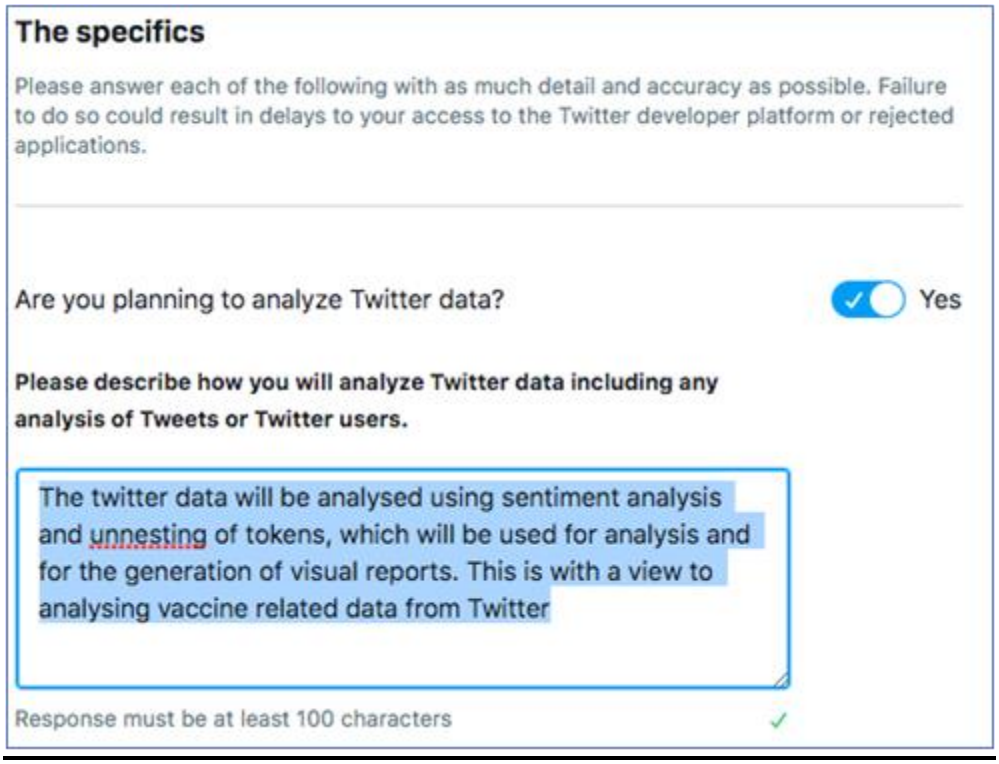

Figure 61: Explaining analysis process

8. Fill in the section for how you plan to use these features. Here is what may be used: 'The platform will use some of this data, such as the text of the tweets, and will extract data related to the tweets such as the number of likes, followers and retweets'

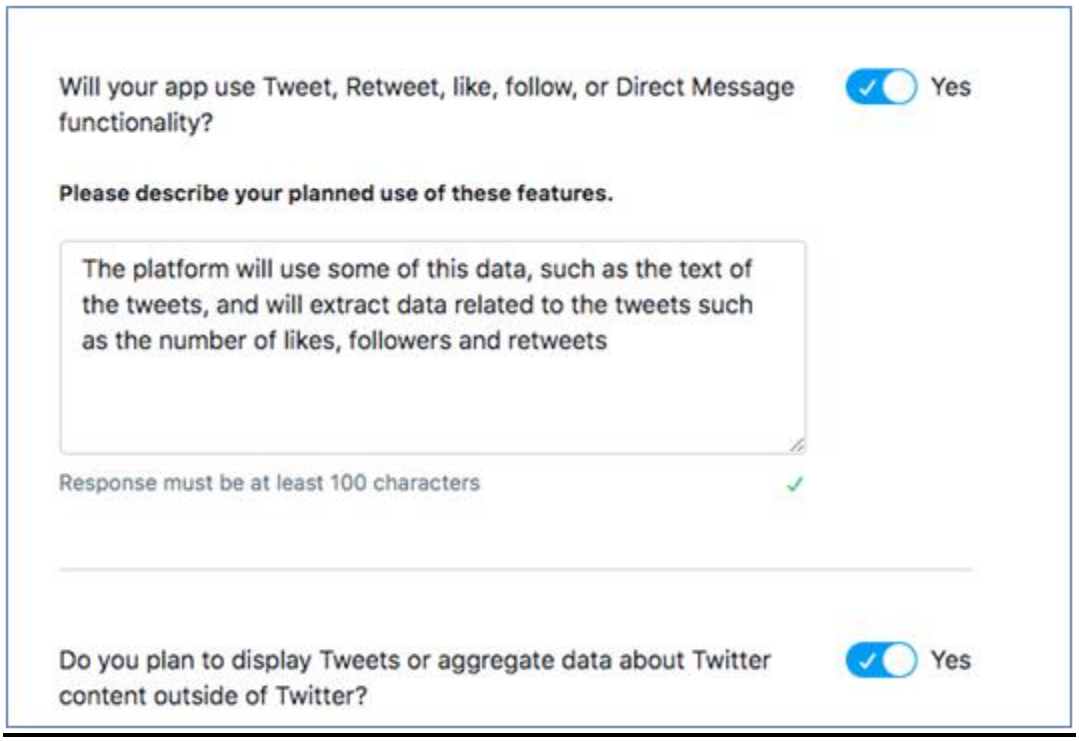

Figure 62: Explaining how features will be used

9. Fill in the box for how the Twitter data will be displayed. For example: 'The data will be used for a project that will be available on the internet. However, the Twitter data will not be publicly available'

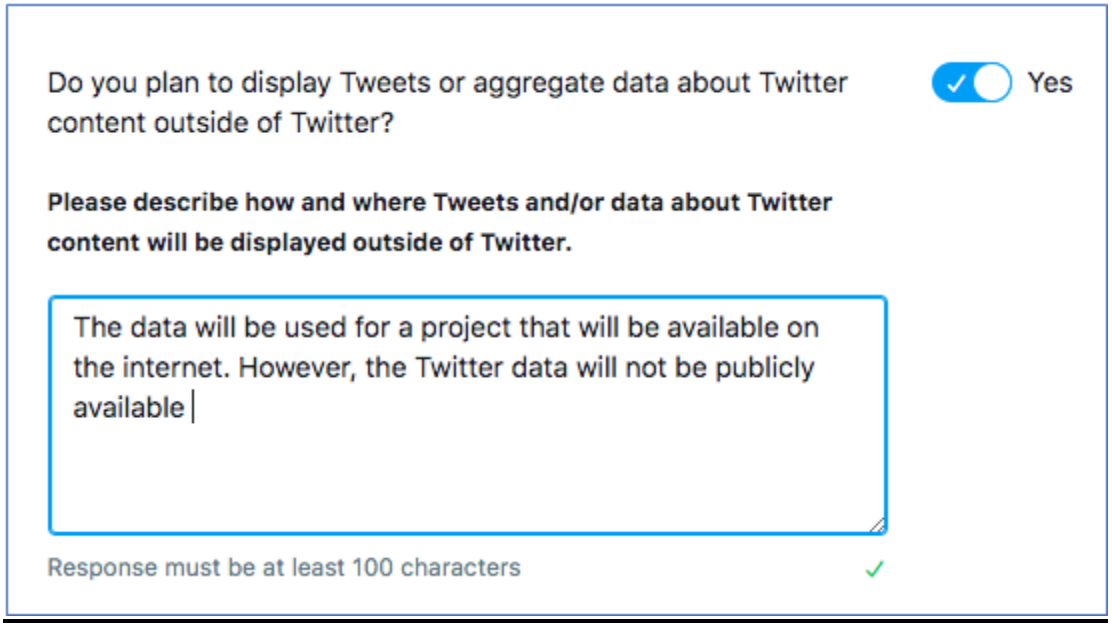

Figure 63: How Twitter content will be displayed outside of Twitter

10. Fill in the box for the government entities that will be provided with the data then click 'Next'

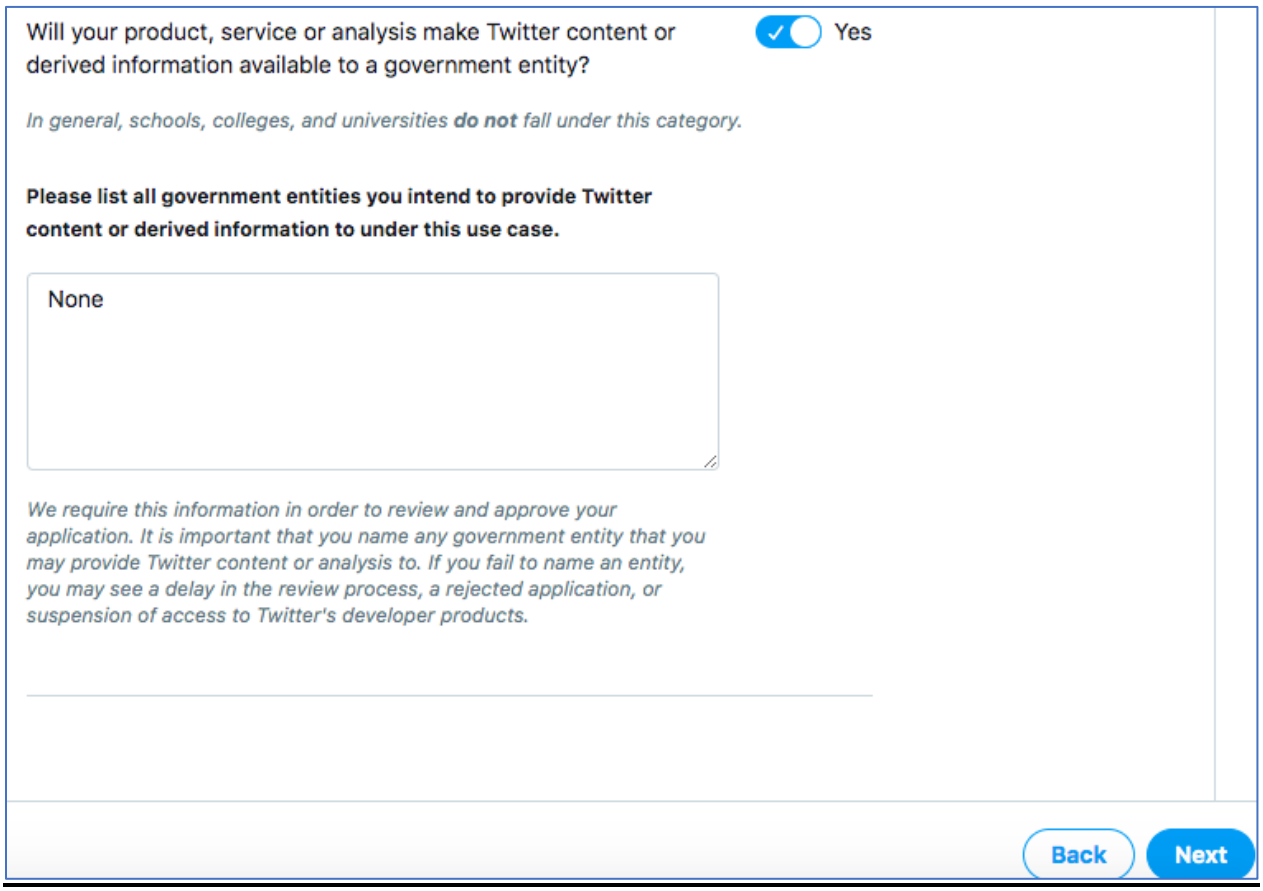

Figure 64: If the content will be made available to a government entity

11. Verify that all of the details are correct, then click 'Looks Good!'
12. Review the terms and conditions and then click to agree to them

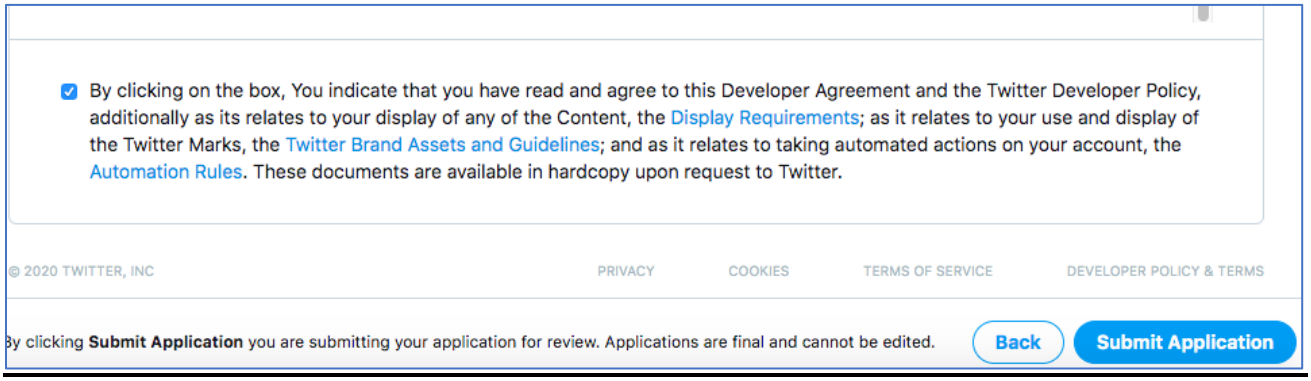

Figure 65: Terms and conditions

13. Click 'Submit Application'
14. Go to the email address that you registered with Twitter, and get the email about confirming your email. Click 'Confirm your email'

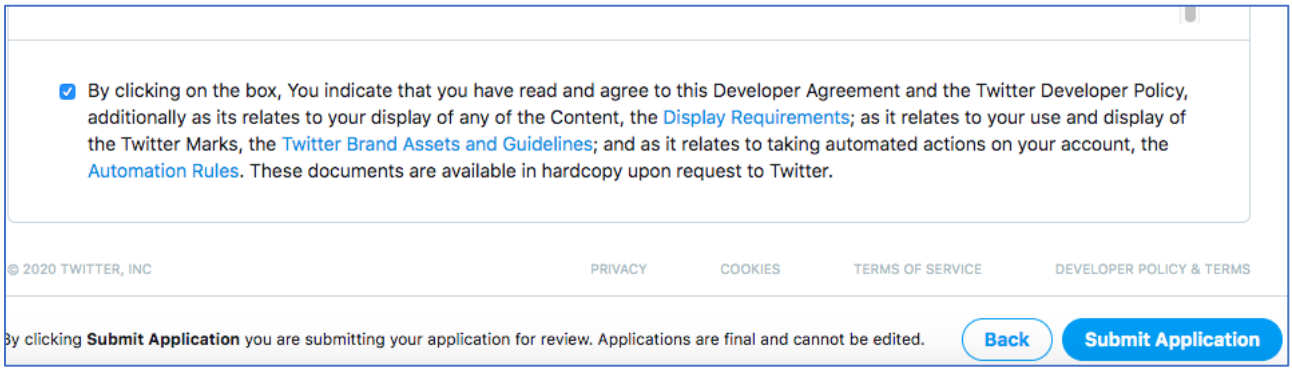

Figure 66: Submitting application

15. Twitter will send you a confirmation email if you have been authorized a developer account
16. Once you are authorized, go to https://developer.twitter.com/en
17. Hover over the name of your project. The name is 'Programming Project' and click on 'Apps'

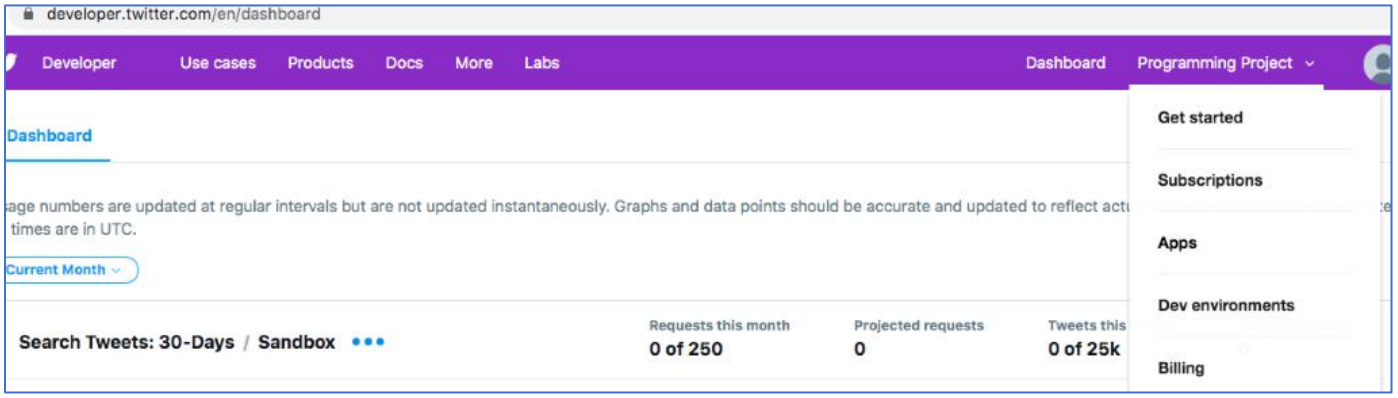

Figure 67: Selecting your project

18. Click 'Create an app'

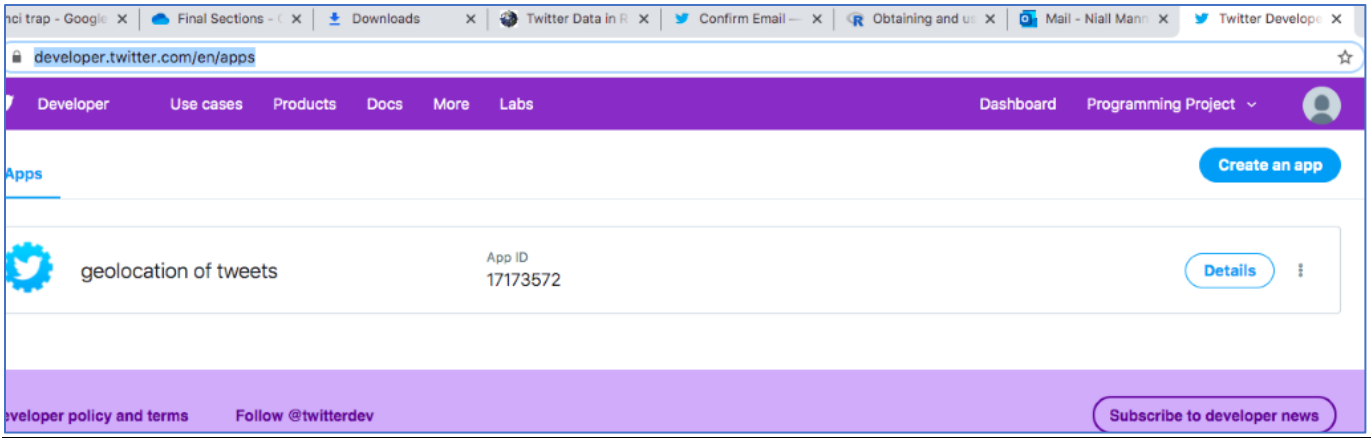

Figure 68: Creating Twitter app

19. Enter in an app name such as 'Vaccine Research' and a description such as 'The app will be used to scrape vaccine related tweets for analysis'

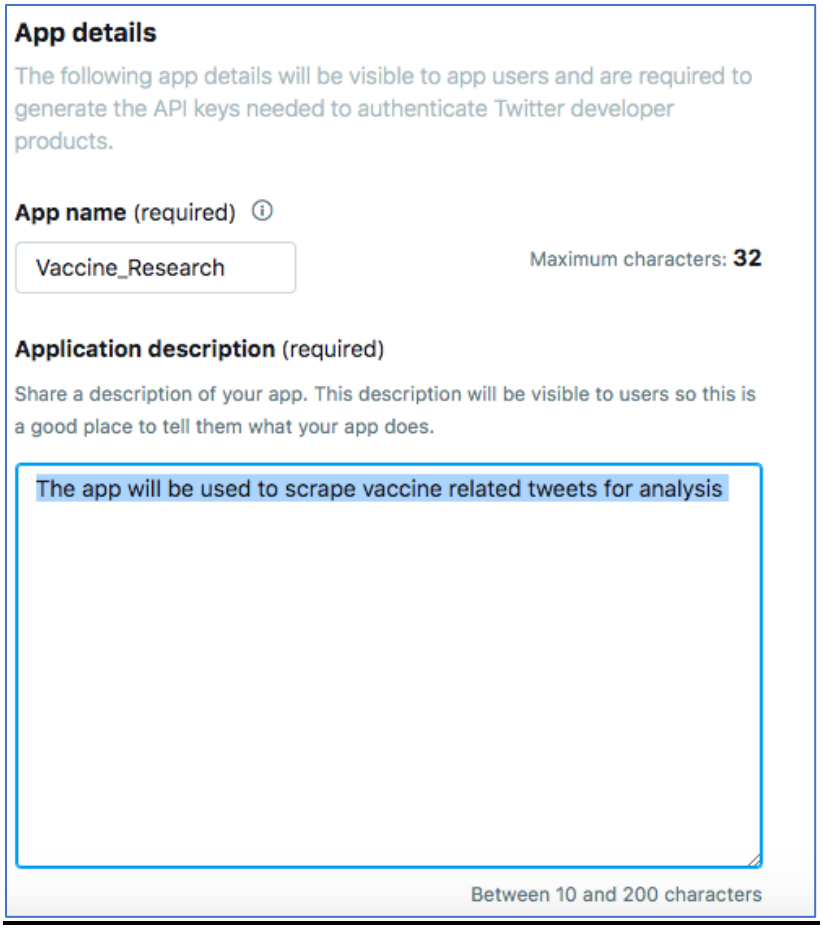

Figure 69: Describing your application

20. Enter a website url (go to your Twitter profile page and copy and paste the url after the 'http://'). Fill in the 'Callback url' box with 'http://127.0.0.1:1410'

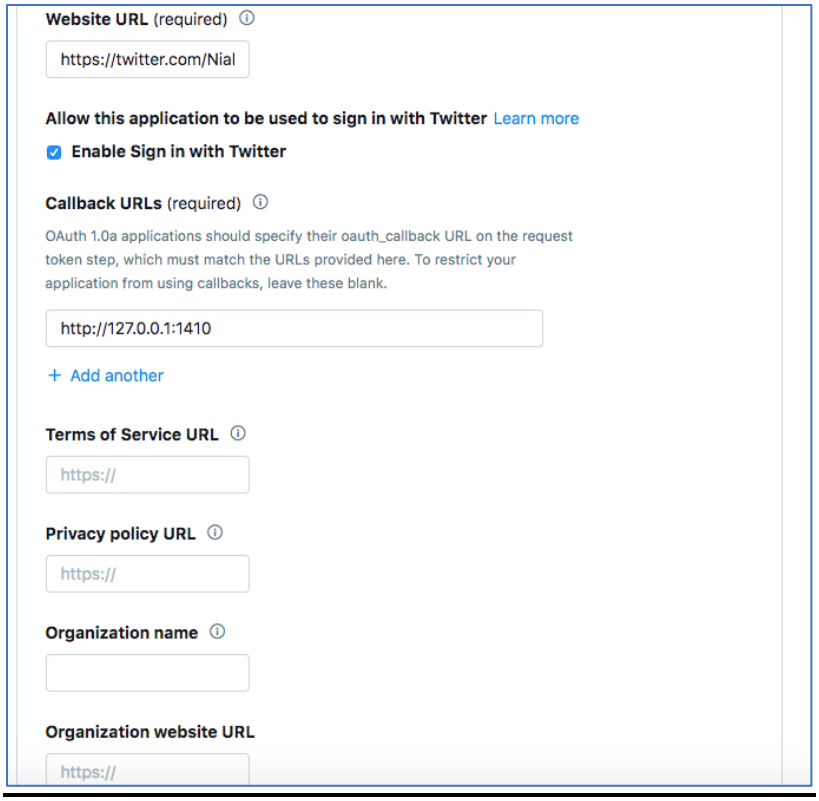

Figure 70: Entering a website url

21. Enter in how the app will be used. For example: 'The app will be used for scraping tweets related to vaccines for analysis and visualisation. This is in order to analyse user attitudes towards vaccines' Then click 'Create'

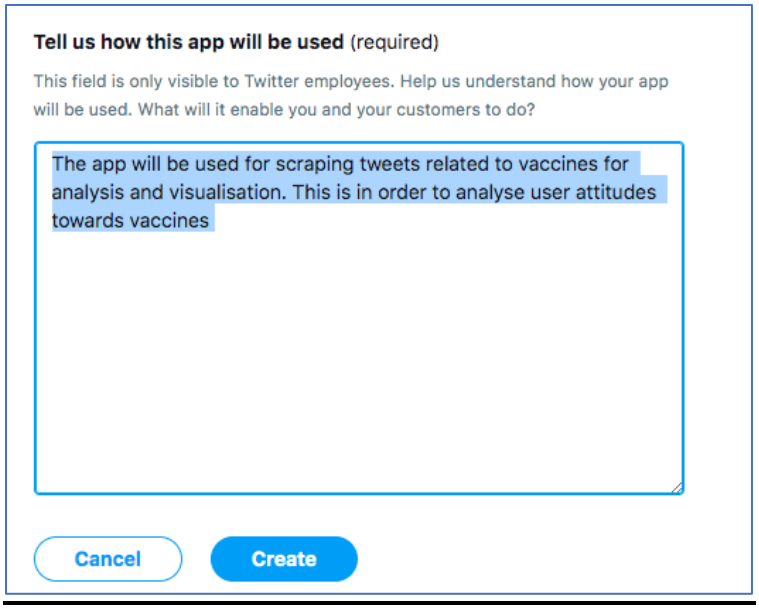

Figure 71: Explaining how to Twitter app will be used

22. Review the terms and conditions, and then click 'Create'

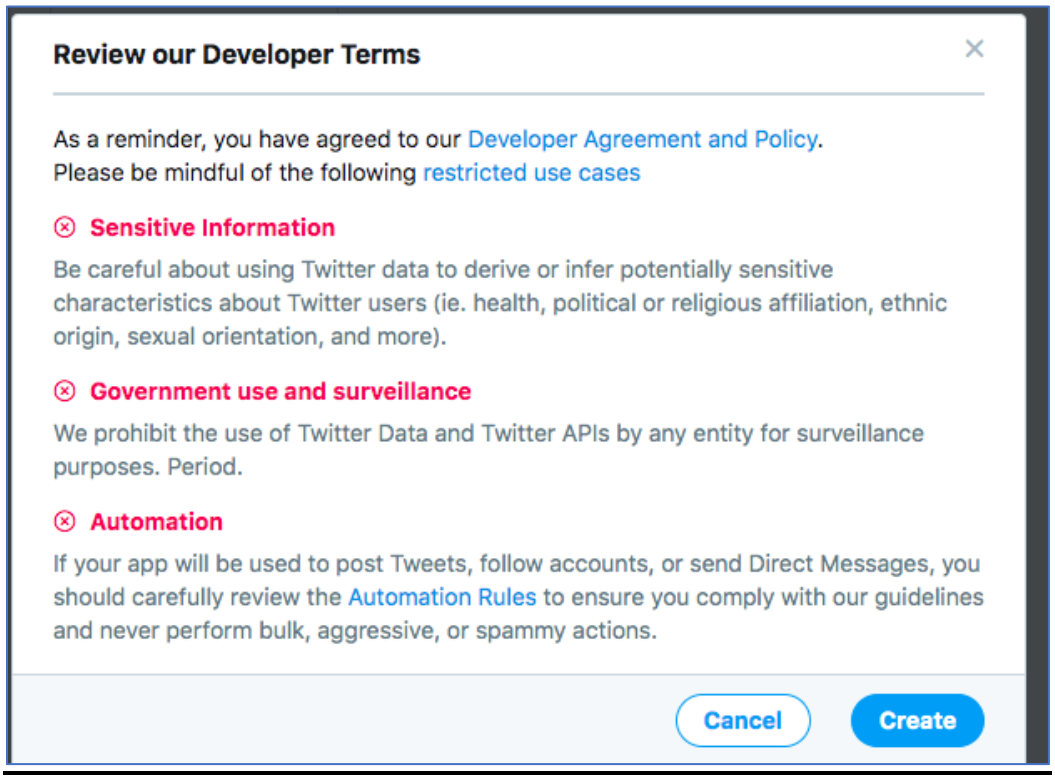

Figure 72: Reviewing developer terms

23. Copy and paste the API key and API secret key into a new word document

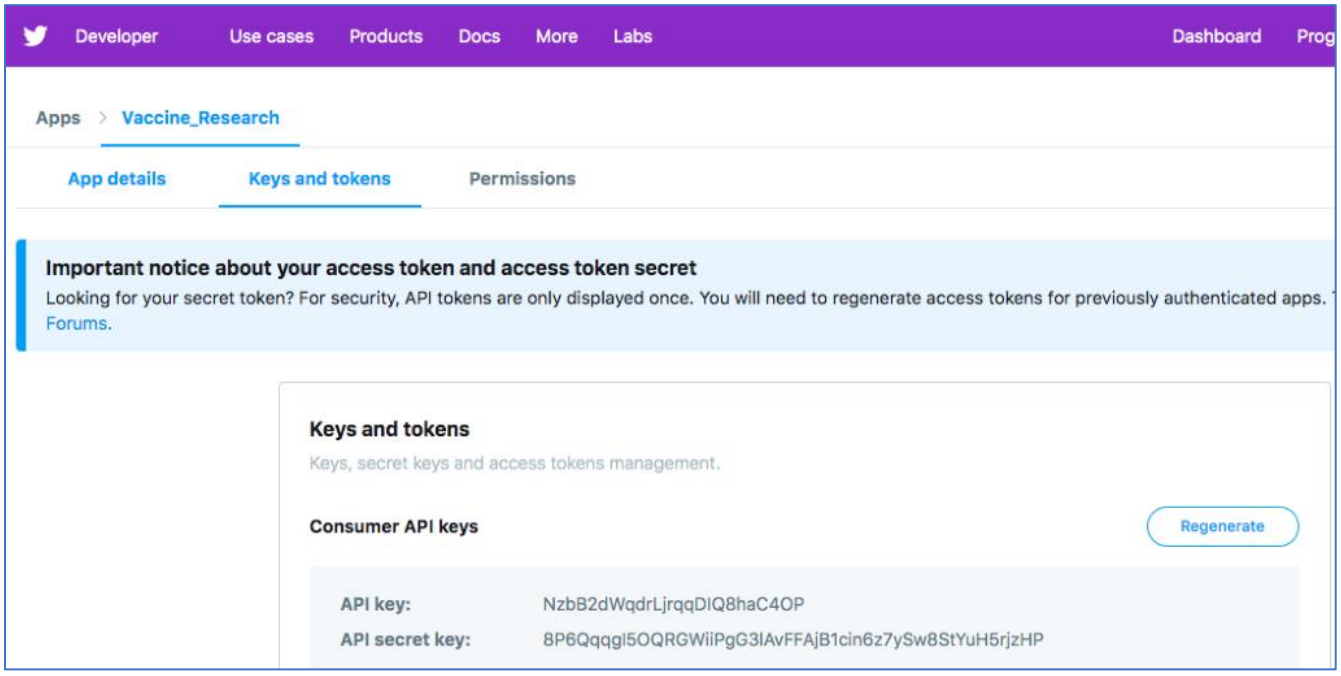

Figure 73: Copying keys

24. Click on 'Regenerate'

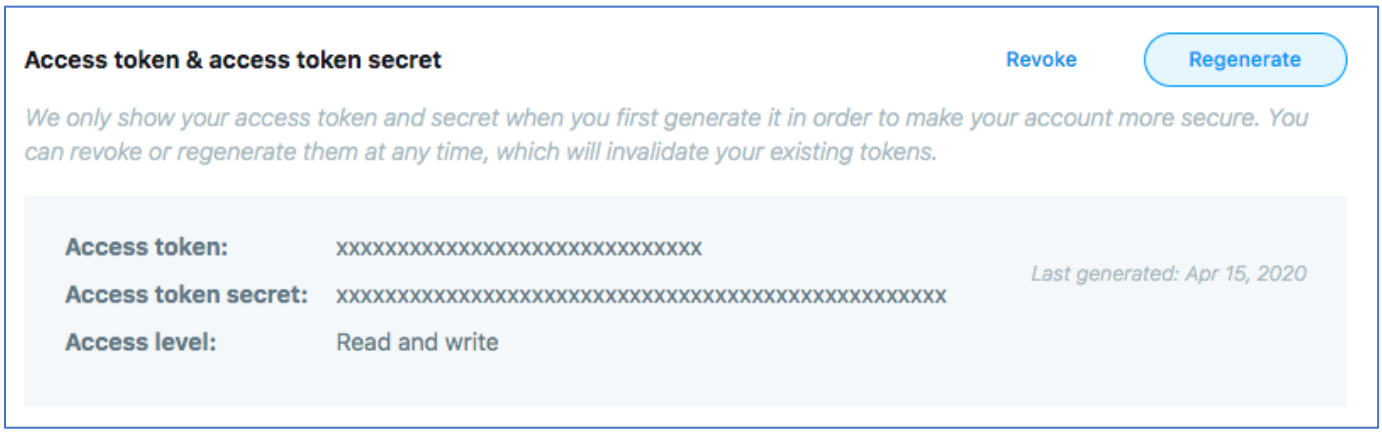

Figure 74: Regenerating keys

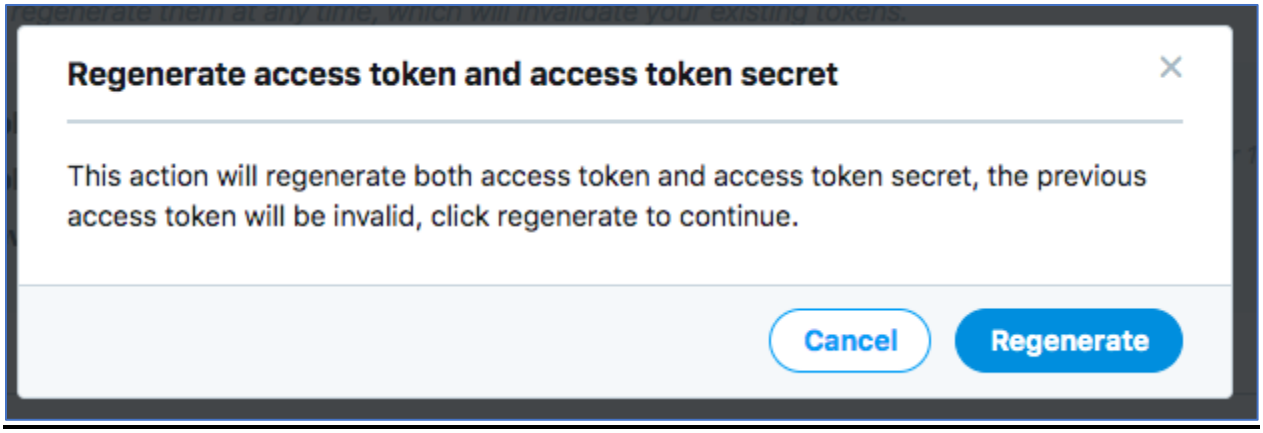

Figure 75: C*onfirming regeneration of keys*

25. Copy and paste the access token and access token secret to the document

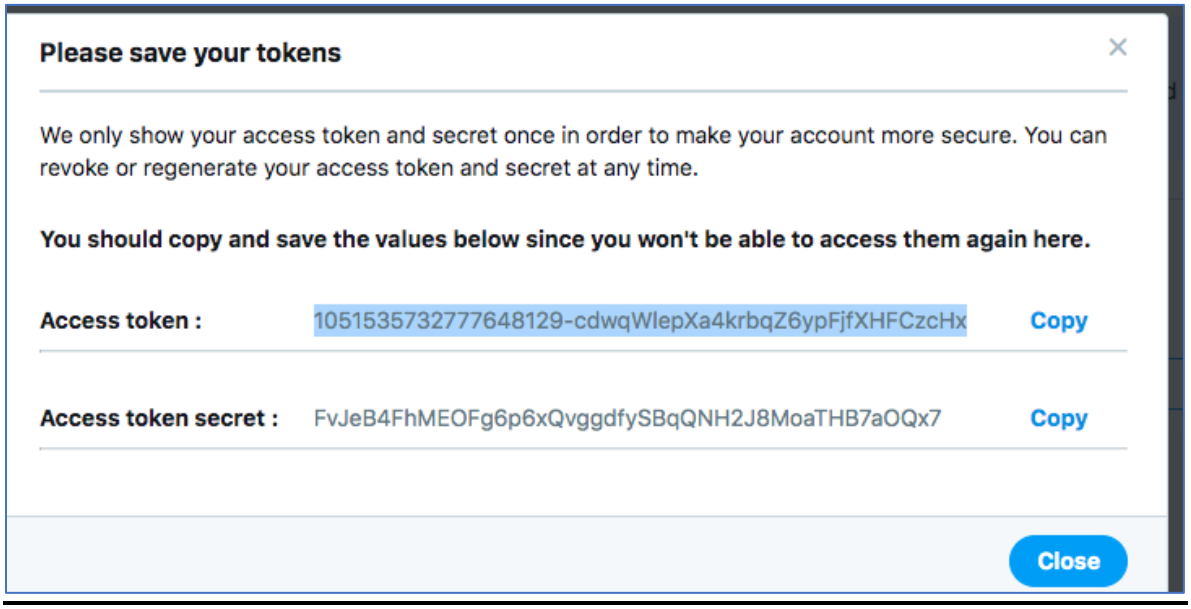

Figure 76: Copying access token

Google Translate API Keys

1. Go to https://console.cloud.google.com/
2. sign into google
3. Click 'Ireland' as the country, and agree to the terms and conditions
4. Click 'AGREE AND CONTINUE'
5. Click on 'Activate' to activate a free trial subscription to Google Cloud

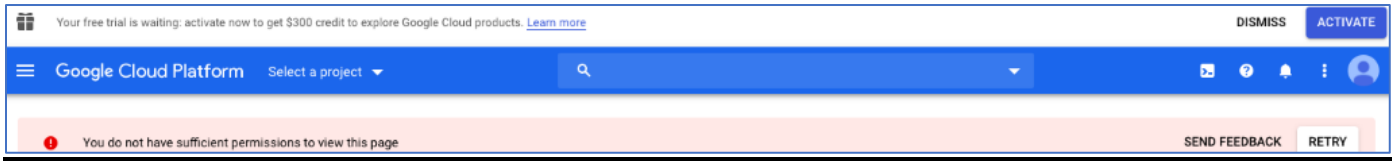

Figure 77: Activating Google Cloud free trial

6. Enter the correct email and country, and agree to the terms and conditions, then click 'Continue'
7. Enter each of the required details, then enter a credit card number, then click 'Start My Free Trial'
8. On the left hand side, hover on the 'APIs and Services' then click 'Dashboard'

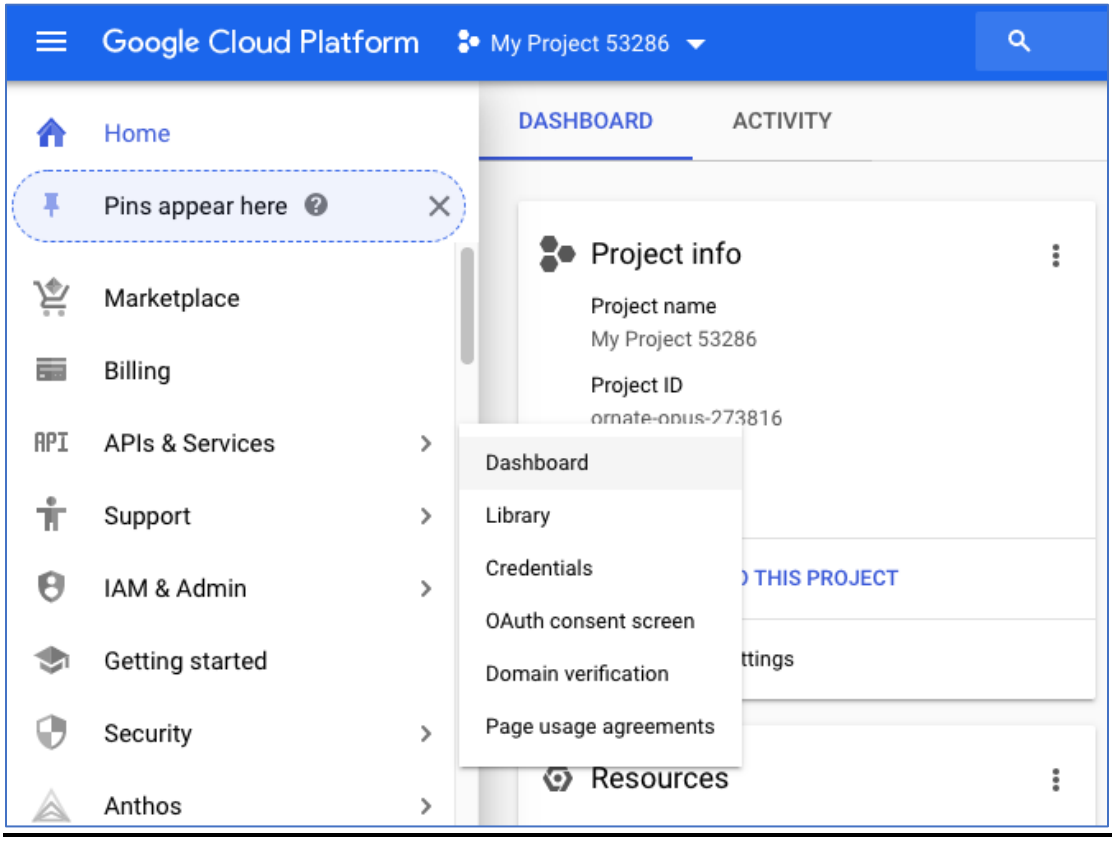

Figure 77: Clicking on dashboard

9. Click on 'Credentials' in the left hand side

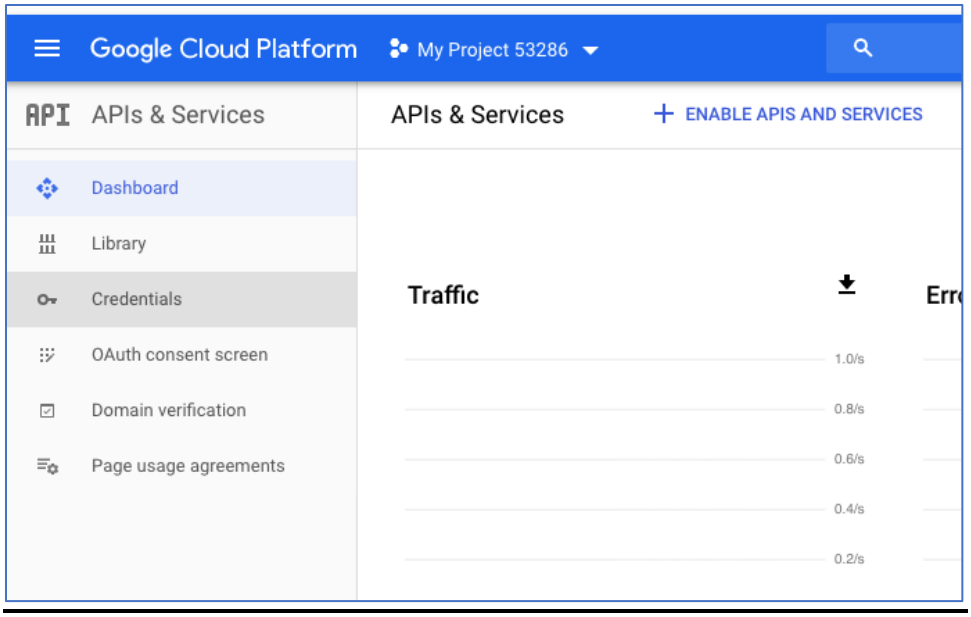

Figure 78: Getting cloud credentials

10. Click on 'Create Credentials'- 'API key'

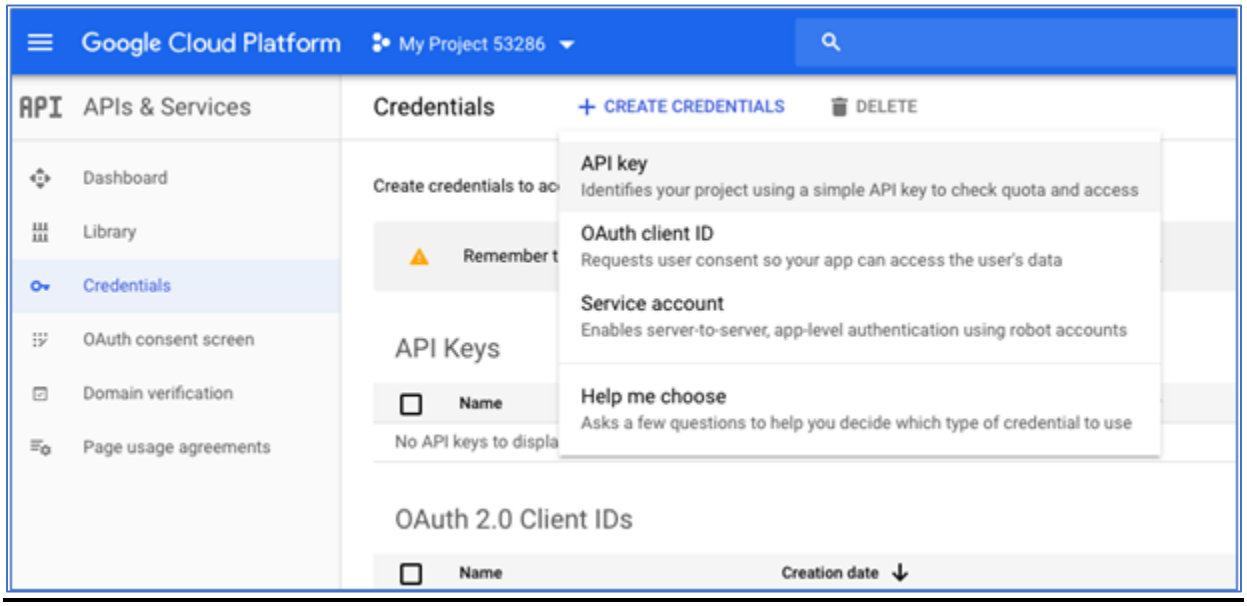

Figure 79: Creating Google Cloud credentials

11. Copy the key

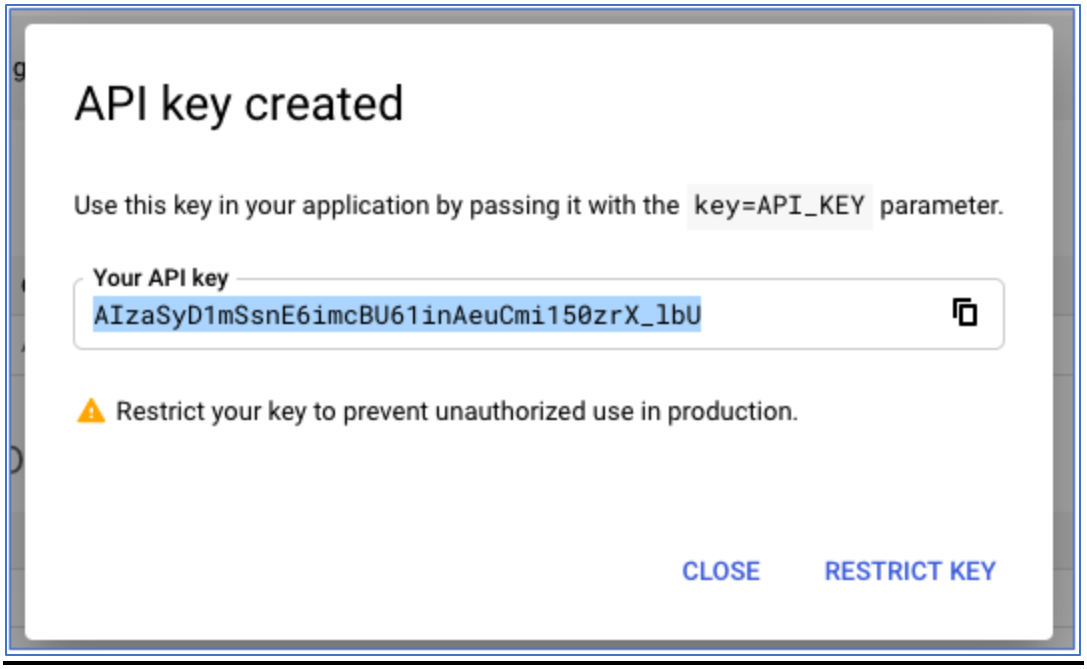

Figure 80: Google API key

R Code

1. Install and load the required packages

```r
# Install and load require packages
install.packages(c("SuperLearner", "ipred", "adabag", "randomForest", "dplyr", "e1071", "nnet",
                   "tidytext", "rtweet", "ggplot2", "tidyverse", "gridExtra", "tidyr", "plyr",
                   "randomForest", "kernlab", "neuralnet", "ggrepel", "translateR", "gtrendsR",
                   "MASS", "DMwR", "C50", "fUnitRoots", "radarchart", "DT", "tsoutliers", "tseries",
                   "MTS", "vars", "syuzhet", "forecastML", "twitteR", "wordcloud2", "plotly",
                   "BBmisc", "grid", "ggplotify", "UBL", "reshape2", "ROAuth", "caret", "kernlab",
                   "cowplot", "tidyverse", "mice", "mlr", "ranger", "ROSE", "C50", "gmodels",
                   "Boruta", "tidyverse", "ROAuth", "gtrendsR", "plyr", "maps", "ggrepel", "gbm",
                   "translateR", "tidyr", "tm", "wordcloud2", "qdap", "radarchart", "MASS", "shiny",
                   "caret", "class", "cowplot", "ggmap", "forecast", "partykit", "party",
                   "flexdashboard", "textdata"))
# Load libraries
loading <- c("SuperLearner", "ipred", "adabag", "randomForest", "e1071", "nnet",
             "tidytext", "rtweet", "ggplot2", "tidyverse", "gridExtra", "tidyr",
             "randomForest", "kernlab", "neuralnet", "ggrepel", "translateR", "gtrendsR",
             "MASS", "DMwR", "C50", "fUnitRoots", "radarchart", "DT", "tsoutliers", "tseries",
             "MTS", "vars", "syuzhet", "forecastML", "twitteR", "wordcloud2", "plotly",
             "BBmisc", "grid", "ggplotify", "UBL", "reshape2", "ROAuth", "caret", "kernlab",
             "cowplot", "tidyverse", "mice", "mlr", "ranger", "ROSE", "C50", "gmodels", "textdata",
             "Boruta", "tidyverse", "ROAuth", "gtrendsR", "plyr", "dplyr", "maps", "ggrepel", "gbm",
             "translateR", "tidyr", "tm", "wordcloud2", "qdap", "radarchart", "MASS", "shiny",
             "caret", "class", "cowplot", "ggmap", "forecast", "partykit", "party", "flexdashboard")
lapply(loading, require, character.only = TRUE)
```

Figure 81: Installing and loading the required packages

Section 1: Online Content

Gtrends

Line Chart

Based on work by Tang (2018)

1. Extract Google trends data from the US for all times based on the keywords "vaccine", "vaccination", "swine" and "covid-19"

```
> vaccine_line <- gtrends(c("vaccine", "vaccination",
+     "swine", "covid-19"), geo = "US", time = "all")
```

Figure 82: Scraping vaccine related Google trends data from the US

2. Create a format for the line graph

```
> plot.gtrends.silent <- function(x, ...) {df <- x$interest_over_time
+ df$date <- as.Date(df$date)
+ df$hits <- if(typeof(df$hits) == 'character'){
+   as.numeric(gsub('<','',df$hits)) } else { df$hits}
+ df$legend <- paste(df$keyword, " (", df$geo, ")", sep = "")
+ p <- ggplot(df, aes_string(x = "date", y = "hits", color = "legend")) +
+   geom_line(size = 2) +
+   xlab("Date") +
+   ylab("Search hits") +
+   ggtitle("") +
+   theme_bw()
+ invisible(p)}
```

Figure 83: Creating format for Google trends line chart

3. Create a theme for the line chart

```
> gtrends_Theme = theme(axis.title.x = element_text(size = 19),
+ axis.text.x = element_text(size = 12), axis.title.y = element_text(size = 20),
+ axis.text.y = element_text(size = 22), legend.text = element_text(color = "black", size = 15),
+ plot.title = element_text(hjust = 0.5, size = 20, face = 'bold'))
```

Figure 84: Theme for Google trends line chart

4. Generate the Google trends line chart

```
> my_plot <- plot.gtrends.silent(vaccine_line)
```

Figure 85: Generating Google trends line chart

5. Plot the final Google trends line chart

```
> my_plot + scale_x_date(date_breaks = "1 year", date_labels = "%Y") + Project_Theme +
+ ggtitle("Google Search Interest (US)")
```

Figure 86: Plotting the Google trends line chart

6. Scrape the tweets for interest in vaccine related side effects

```
> side_effects_line <- gtrends(c("vaccine side effects", "thimerosal",
+     "vaccines autism", "vaccine autism link"), geo = "US", time = "all")
```

Figure 87: Scraping Google trends related to vaccine side effects

7. Generate the plot for interest over time

```
> side_effects_plot <- plot.gtrends.silent(side_effects_line)
```

Figure 88: Generating plot for Google trends related to vaccine side effects

8. Plot the interest over time for the side effects

```
> side_effects_plot + scale_x_date(date_breaks = "1 year", date_labels = "%Y") + Project_Theme +
+   ggtitle("Google Search Interest in Vaccine Side Effects Over TIme (US)")
```

Figure 89: Plotting Google search interest in vaccine side effects over time

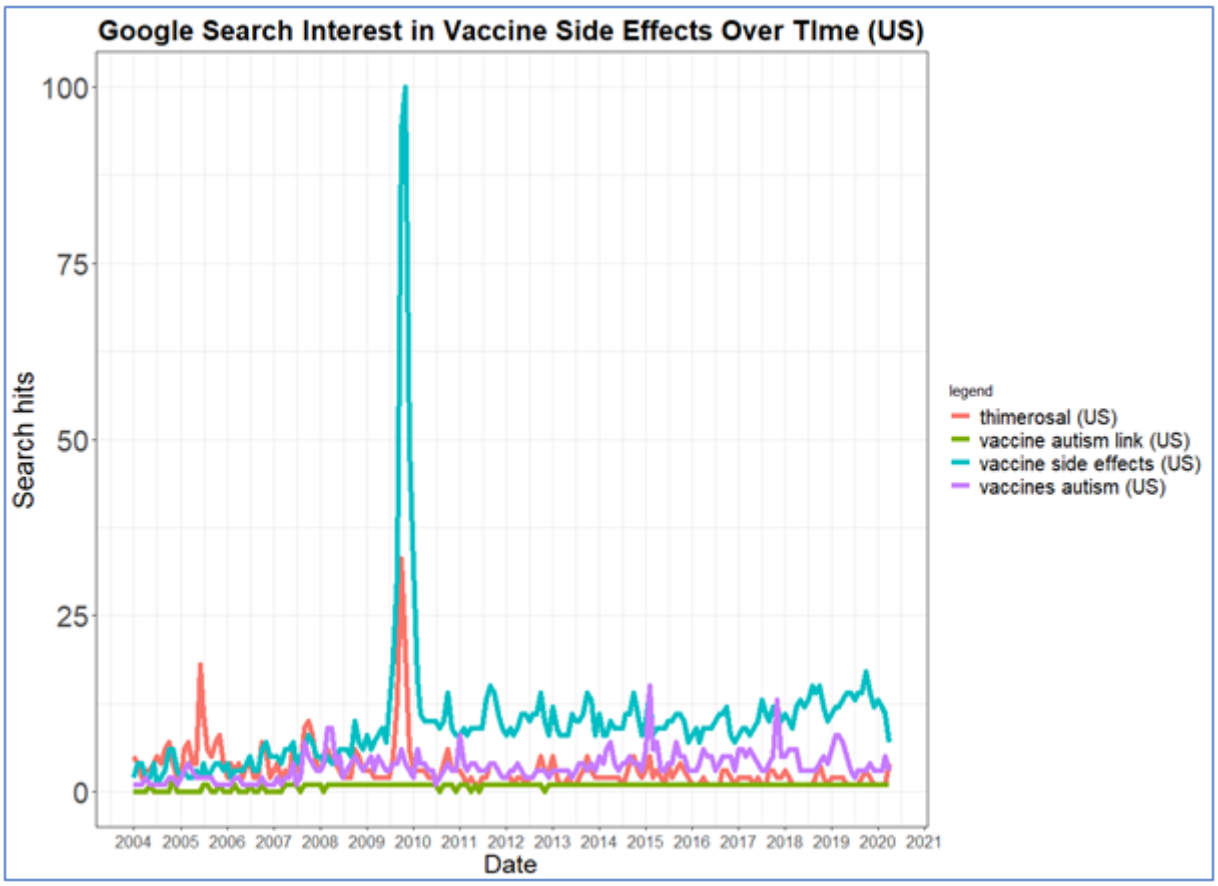

Figure 9: Vaccine side effects Google search interest over time

```
other_line <- gtrends(c("immunization", "mainstream media",
  "pharmaceutical industry", "vaccine"), geo = "US", time = "all")
other_plot <- plot.gtrends.silent(other_line)
other_plot +
  scale_x_date(date_breaks = "1 year", date_labels = "%Y") + Project_Theme +
  ggtitle("Google Search Interest in Vaccine Related Topics Over Time (US)")
```

Figure 90: Plotting Google search interest in vaccine related topics over time

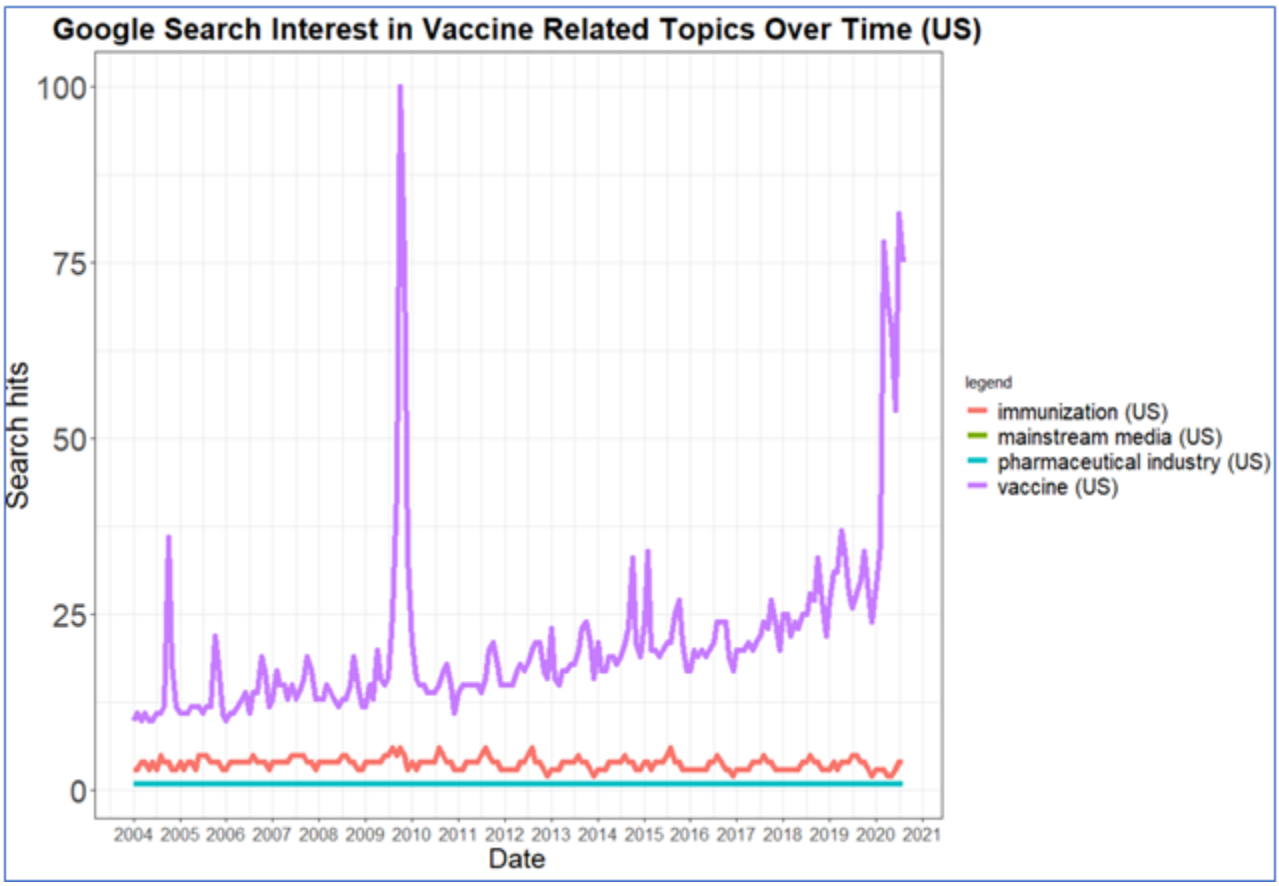

Figure 91: Vaccine related topics Google search interest over time

Map of Google Trends Data

Based on work by https://peerchristensen.netlify.app/post/mapping-hurricane-search-data-from-google-trends/

1. Create the theme for the map (remove all elements such as title and legend)

```
> my_theme <- function() {theme_bw() + theme(panel.background  = element_blank(),
+ plot.background = element_rect(fill = "seashell"),
+ panel.border = element_blank(), strip.background  = element_blank(),
+ plot.margin = unit(c(.5, .5, .5, .5), "cm"),
+ panel.spacing = unit(3, "lines"), panel.grid.major  = element_blank(),
+ panel.grid.minor = element_blank(), legend.background = element_blank(),
+ legend.key = element_blank(), legend.title      = element_blank())}
```

Figure 92: Creating theme for Google trends map

2. Create the second theme for the map

```
> my_theme2 <- function() {my_theme() + theme(axis.title = element_blank(),
+ axis.text = element_blank(), axis.ticks = element_blank())}
```

Figure 93: Creating second theme for Google trends map

3. Scrape the Google trends data for the last year for the keyword “vaccine” in the US

```
> vaccine_gtrends <- gtrends(c("vaccine"), time = "today 12-m", geo = "US")
```

Figure 94: Scraping vaccine related Google trends data

4. Extract the values for interest over time from the dataset

```
> vaccine_gtrends <- vaccine_gtrends$interest_by_region
```

Figure 95: Extracting interest over time

5. Conver the region column to lower, merge this data with geographical data from the ‘statesMap’ dataset and create a set of labels for the map

```
> gtrends_vaccine$region  <- sapply(gtrends_vaccine$location,tolower)
> final_gtrends    <- merge(statesMap ,gtrends_vaccine,by="region")
> regionLabels <- aggregate(cbind(long, lat) ~ region, data=final_gtrends,
+                           FUN=function(x) mean(range(x)))
```

Figure 96: Preparing geographical information for Google trends map

6. Generate the map for gtrends map

```
> final_gtrends %>% ggplot() + geom_polygon(aes(x=long,y=lat,group=group,fill=log(hits)),
+ colour="white") + scale_fill_continuous(low="ivory",high="midnightblue") + guides(fill = "colorbar") +
+ geom_text_repel(data=regionLabels, aes(long, lat, label = region), size=5) +
+ scale_fill_distiller(palette = "Reds") + my_theme2() + theme(legend.position = "none") +
+ theme(plot.title = element_text(hjust = 0.5)) +
+ coord_fixed(1.3) + ggtitle("Google search interest for 'vaccine' in the US Over the Last 12 Months")
```

Figure 97: Generating Google search interest map

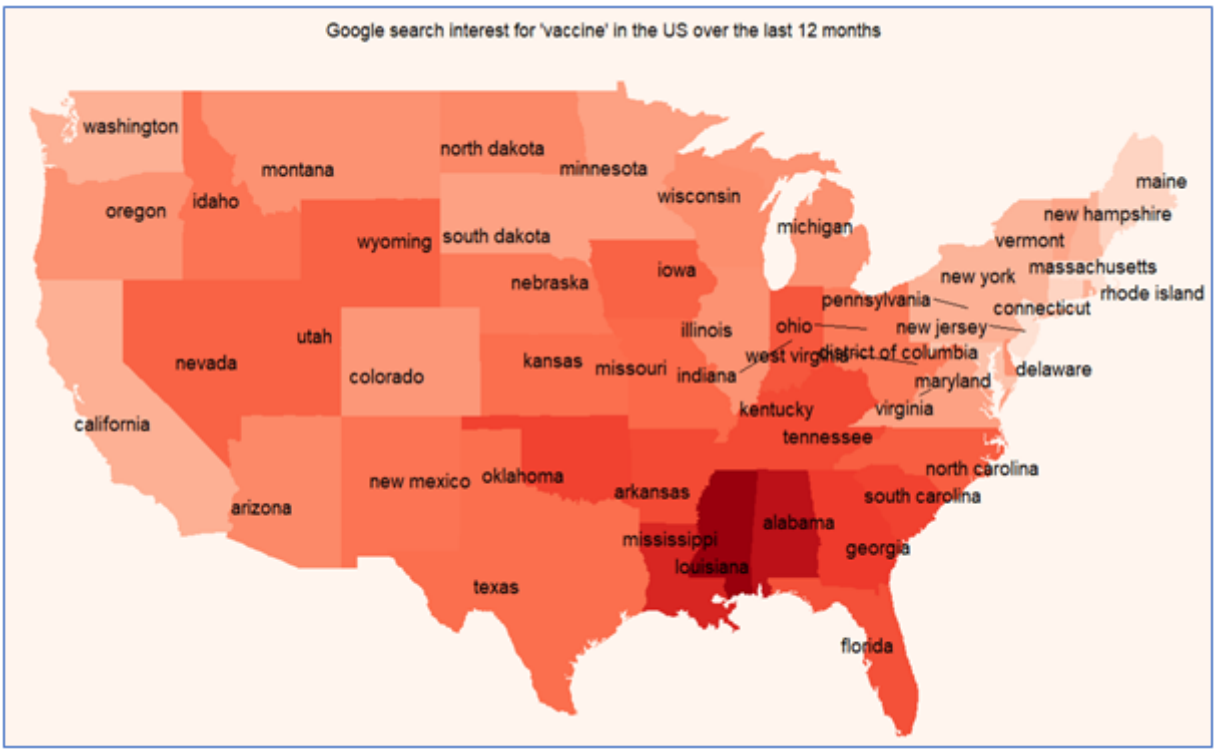

Figure 98: Map for Google search interest in 'vaccine' in the US

Extracting and Analysing Twitter Data

1. Set up Twitter authentication

```
> setup_twitter_oauth('jNjVkpRdmRceYEiZyO33t8CXu',
+                     'HdIKY3YgNRxNLVwEK4Qz4gX0YdmpoWKUUrnICXcfxhmPCFEJl1',
+                     '1051535732777648129-YUDQFUzojiCNl2iQ3xPpyAIOmDM3bB',
+                     'r2u3LMszcp2fWu7NtDBIFsJim19i10MvzHC8CX9HQp9Bv')
```

Figure 99: Setting up Twitter authentication

2. Scrape vaccine related tweets from seven locations in Texas (the geocode and radius can be obtained using https://www.mapdevelopers.com/draw-circle-tool.php). The steps for scraping tweets were from https://rtweet.info/

```
sanangelo <- search_tweets('vax OR vaccine OR vaccinated OR vaccination OR immunization
OR immunized', geocode = "31.4638,-100.4370,127mi", n=20000, include_rts = FALSE)
sanangelo$place <- 'San Angelo'
amarillo <- search_tweets('vax OR vaccine OR vaccinated OR vaccination OR immunization
OR immunized', geocode = "35.2220,-101.8313,73mi", n=20000, include_rts = FALSE)
amarillo$place <- 'Amarillo'
Lubbock <- search_tweets('vax OR vaccine OR vaccinated OR vaccination OR immunization
OR immunized', geocode = "33.5779,-101.8552,49mi", n=20000, include_rts = FALSE)
Lubbock$place <- 'Lubbock'
Austin <- search_tweets('vax OR vaccine OR vaccinated OR vaccination OR immunization
OR immunized', geocode = "30.2672,-97.7431,57mi", n=20000, include_rts = FALSE)
Austin$place <- 'Austin'
Houston <- search_tweets('vax OR vaccine OR vaccinated OR vaccination OR immunization
OR immunized', geocode = "29.7604,-95.3698,62mi", n=20000, include_rts = FALSE)
Houston$place <- 'Houston'
SevenSisters <- search_tweets('vax OR vaccine OR vaccinated OR vaccination OR immunization
OR immunized', geocode = "28.0106,-98.5392,104mi", n=20000, include_rts = FALSE)
SevenSisters$place <- 'Seven Sisters'
Athens <- search_tweets('vax OR vaccine OR vaccinated OR vaccination OR immunization
OR immunized', geocode = "32.2049,-95.8555,104mi", n=20000, include_rts = FALSE)
Athens$place <- 'Athens'
```

Figure 100: Scraping vaccine related tweets from Texas

3. Combine these tweets

```
combineddataframe <- rbind(Athens,SevenSisters,Houston,Austin,Lubbock,amarillo,sanangelo)
```

Figure 101: Combining tweets

4. Remove duplicate tweets (from the 'text' column) an view the columns of the twitter dataframe

```
combineddataframe = combineddataframe[!duplicated(combineddataframe$text),]
colnames(combineddataframe)
```

Figure 102: Removing duplicate tweets and viewing column names

```
 [1] "user_id"                "status_id"              "created_at"             "screen_name"
 [5] "text"                   "source"                 "display_text_width"     "reply_to_status_id"
 [9] "reply_to_user_id"       "reply_to_screen_name"   "is_quote"               "is_retweet"
[13] "favorite_count"         "retweet_count"          "quote_count"            "reply_count"
[17] "hashtags"               "symbols"                "urls_url"               "urls_t.co"
[21] "urls_expanded_url"      "media_url"              "media_t.co"             "media_expanded_url"
[25] "media_type"             "ext_media_url"          "ext_media_t.co"         "ext_media_expanded_url"
[29] "ext_media_type"         "mentions_user_id"       "mentions_screen_name"   "lang"
[33] "quoted_status_id"       "quoted_text"            "quoted_created_at"      "quoted_source"
[37] "quoted_favorite_count"  "quoted_retweet_count"   "quoted_user_id"         "quoted_screen_name"
[41] "quoted_name"            "quoted_followers_count" "quoted_friends_count"   "quoted_statuses_count"
[45] "quoted_location"        "quoted_description"     "quoted_verified"        "retweet_status_id"
[49] "retweet_text"           "retweet_created_at"     "retweet_source"         "retweet_favorite_count"
[53] "retweet_retweet_count"  "retweet_user_id"        "retweet_screen_name"    "retweet_name"
[57] "retweet_followers_count" "retweet_friends_count" "retweet_statuses_count" "retweet_location"
[61] "retweet_description"    "retweet_verified"       "place_url"              "place_name"
[65] "place_full_name"        "place_type"             "country"                "country_code"
[69] "geo_coords"             "coords_coords"          "bbox_coords"            "status_url"
[73] "name"                   "location"               "description"            "url"
[77] "protected"              "followers_count"        "friends_count"          "listed_count"
[81] "statuses_count"         "favourites_count"       "account_created_at"     "verified"
[85] "profile_url"            "profile_expanded_url"   "account_lang"           "profile_banner_url"
[89] "profile_background_url" "profile_image_url"      "place"
```

Figure 103: Column names of vaccine related tweets

5. Scrape the Spanish tweets by using the keyword “vacuna” (meaning vaccine in Spanish)

```
sp_sanangelo <- search_tweets('vacuna', geocode = "31.4638,-100.4370,127mi",
n=20000, include_rts = FALSE)
sp_sanangelo$place <- 'San Angelo'
sp_amarillo <- search_tweets('vacuna', geocode = "35.2220,-101.8313,73mi",
n=20000, include_rts = FALSE)
sp_amarillo$place <- 'Amarillo'
sp_Lubbock <- search_tweets('vacuna', geocode = "33.5779,-101.8552,49mi",
n=20000, include_rts = FALSE)
sp_Lubbock$place <- 'Lubbock'
sp_Austin <- search_tweets('vacuna', geocode = "30.2672,-97.7431,57mi",
n=20000, include_rts = FALSE)
sp_Austin$place <- 'Austin'
sp_Houston <- search_tweets('vacuna', geocode = "29.7604,-95.3698,62mi",
n=20000, include_rts = FALSE)
sp_Houston$place <- 'Houston'
sp_SevenSisters <- search_tweets('vacuna', geocode = "28.0106,-98.5392,104mi",
n=20000, include_rts = FALSE)
sp_SevenSisters$place <- 'Seven Sisters'
sp_Athens <- search_tweets('vacuna', geocode = "32.2049,-95.8555,104mi",
n=20000, include_rts = FALSE)
sp_Athens$place <- 'Athens'
```

Figure 24: Scraping Spanish tweets

6. Bind the Spanish tweets, remove duplicate tweets and view the fifth column

```
spanish_tweets <- rbind(sp_Athens, sp_SevenSisters,
sp_Houston, sp_Austin, sp_Lubbock, sp_amarillo, sp_sanangelo)
spanish_tweets = spanish_tweets[!duplicated(spanish_tweets$text),]
spanish_tweets[,5, drop = FALSE]
```

Figure 104: Preprocessing and viewing Spanish tweets

```
# A tibble: 165 x 1
  text
  <chr>
1 "@GiuseppeNoc Nada de ponerse esa vacuna"
2 "@HallaquitasM @observadorpapal @RobertoCarlo14 @alias877 @pintacapf Publica un video de las 2 personas que se pus~
3 "@HallaquitasM @observadorpapal @RobertoCarlo14 @alias877 @pintacapf Ya hay 2 personas en Inglaterra que se pusier~
4 "@observadorpapal una vez puesta la vacuna hay algún efecto?? Cual seria el control a que estarian las 2 personas ~
5 "Mas Claro Vacuna contra el Covid19 sera Larga la espera  https://t.co/tdpfSL1Ddw"
```

Figure 105: Content of fifth row of Spanish tweets

7. Translate the Spanish tweets fro Spanish to English using the previously obtained Google API key

```
translated <- translate(dataset = spanish_tweets, content.field = 'text',
google.api.key = "AIzaSyAGMlOFArvFwjFJOcbRY5JVNdq8jAoQLj0",
source.lang = 'es', target.lang = 'en')
```

Figure 106: Translating Spanish tweets

8. Extract the desired columns from the English ("comb") and Spanish ("tTweets") tweets

```
comb <- combineddataframe[ ,c(3, 5, 14, 32, 91)]
tTweets <- translated[ ,c(3, 14, 32, 91, 92)]
```

Figure 107: Extracting desired columns from twitter dataframes

9. View the extracted columns

```
> colnames(tTweets)
[1] "created_at"       "retweet_count"    "lang"             "place"            "translatedContent"
> colnames(comb)
[1] "created_at"    "text"          "retweet_count" "lang"          "place"
```

Figure 108: Extracted columns from Twitter dataframes

10. View the translated fifth row of the Spanish tweets

```
tTweets[ ,7, drop = FALSE]
```

Figure 109: Viewing translated fifth row of Spanish tweets

```
# A tibble: 165 x 1
  text
  <chr>
1 @GiuseppeNoc No getting that shot
2 @HallaquitasM @observadorpapal @ RobertoCarlo14 @ alias877 @pintacapf Post a video of the 2 people who got the vac~
3 @HallaquitasM @observadorpapal @ RobertoCarlo14 @ alias877 @pintacapf There are already 2 people in England who go~
4 @papal observer once the vaccine is in place, is there any effect? What would be the control that would be the 2 p~
5 Clearer Vaccine against Covid19 will be Long waiting https://t.co/tdpfSL1Ddw
6 In Colombia a pseudo scientist was promoted in the 90s, Dr. Patarroyo called himself and the State financed his la~
7 Who wins the race to make the coronavirus vaccine? @PanAmPost_en https://t.co/wER6eHT6z9
```

Figure 110: Translated content of fifth row of Spanish tweets

11. Change the name of the tTweets column "translatedContent" to "text", bind tTweets and comb and view the final dataset

```
colnames(tTweets)[5] <- "text"
final_tweets <- rbind(tTweets, comb)
head(final_tweets)
```

Figure 111: Creating combined Twitter dataframe

```
# A tibble: 6 x 5
  created_at          retweet_count lang  place  text
  <dttm>                      <int> <chr> <chr>  <chr>
1 2020-05-02 17:31:27             0 es    Athens @GiuseppeNoc No getting that shot
2 2020-04-29 23:12:13             0 es    Athens @HallaquitasM @observadorpapal @ RobertoCarlo14 (
3 2020-04-29 23:11:32             1 es    Athens @HallaquitasM @observadorpapal @ RobertoCarlo14 (
4 2020-04-29 23:02:45             1 es    Athens @papal observer once the vaccine is in place, is
5 2020-05-02 12:55:08             0 es    Athens Clearer Vaccine against Covid19 will be Long wai
6 2020-05-02 10:39:27             0 es    Athens In Colombia a pseudo scientist was promoted in t
```

Figure 112: View of final Twitter dataframe

12. Save the file with todays date

```
setwd("D:\\1 Masters\\Dataset\\Project Datasets")
todays_date <- Sys.Date()
output_name <- paste("Twitter ", todays_date,".csv", sep="")
write.csv(final_tweets, file= output_name)
```

Figure 113: Saving Twitter dataframe with todays date

Analysis of Twitter Data

1. Bind the dataframes, extract the desired columns and view the combined dataframe

```
Texas_Tweets <- rbind(Texas_09th, Texas_16th, Texas_23rd, Texas_30th)
final_texas_tweets <- Texas_Tweets[ ,c(3, 5, 13, 14, 17, 32, 91)]
head(final_texas_tweets)
```

Figure 114: Processing and viewing Twitter dataset

```
          created_at favorite_count retweet_count hashtags lang  place
1 2020-03-09 18:38:56              0             0     <NA>   en Athens
2 2020-03-02 05:05:21              4             3     <NA>   en Athens
3 2020-03-02 05:15:24              1             0     <NA>   en Athens
4 2020-03-02 07:34:08             10             6     <NA>   en Athens
5 2020-03-02 05:02:08             19            18     <NA>   en Athens
6 2020-03-09 18:38:04              0             0     <NA>   en Athens


                                    text
1
                          @GeenaJagger @realDonaldTrump and if the vaccine ends up coming from Israel, all the drooli
ng anti semitic leftists can step to the side
```

Figure 115: View of Twitter dataframe

2. Plot the volume of tweets per day

```
p <- final_texas_tweets %>%
  ts_plot("3 hours") +
  ggplot2::theme_minimal() +
  ggplot2::theme(plot.title = ggplot2::element_text(face = "bold")) +
  ggplot2::labs(
    x = "Date", y = "Number of Tweets",
    title = "Volume of Vaccine Related Tweets By Date")
w <- p + theme(axis.text=element_text(size=18),
               axis.title=element_text(size=18))
q <- w + theme(plot.title = element_text(size=20, face="bold"))
q + theme(plot.title = element_text(hjust = 0.5))
```

Figure 116: Plotting number of tweets per day

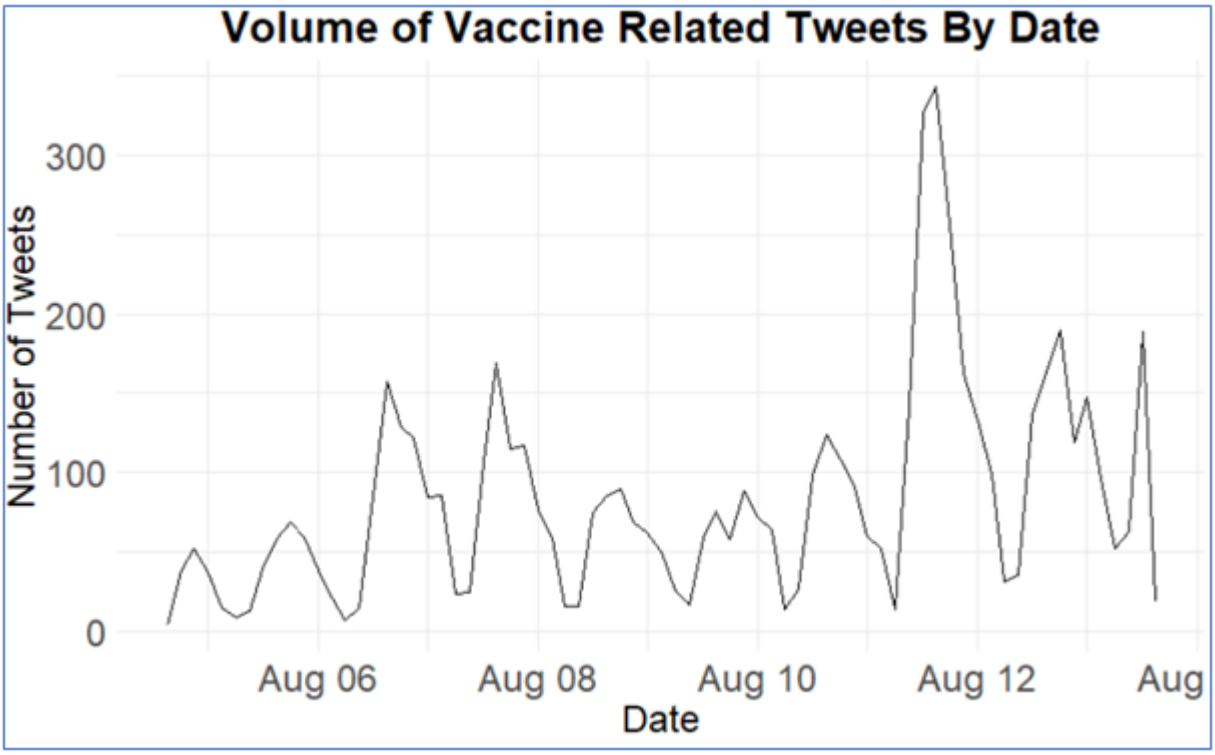

Figure 117: Plot of number of tweets per day

3. View the column names in the twitter dataframe

```
> colnames(final_texas_tweets)

[1] "created_at"     "favorite_count" "retweet_count"  "hashtags"       "lang"           "place"
[7] "text"
```

Figure 118: Viewing column naes of twitter dataframe

4. Change the column names

```
texasdf <- final_texas_tweets %>% select(Date = created_at, Text = text, Favourite_Count = favorite_count,
Number_of_Retweets = retweet_count, Language = lang, Hashtags = hashtags, Location = place)
```

Figure 119: Changing column names of Twitter dataframe

5. View the class of the dataframe columns

```
> selected <- c("Date", "Text", "Location", "Language", "Hashtags")
> texasdf[selected] <- lapply(texasdf[selected], as.character)
> sapply(texasdf, class)

          Date             Text    Favourite_Count Number_of_Retweets         Language         Hashtags
   "character"      "character"          "integer"          "integer"      "character"      "character"
      Location
   "character"
```

Figure 120: Viewing the class of the twitter columns

Preprocessing

The preprocessing stage was based on the methodology by Liske (2018(a)) and Liske, D. (2018(b))

1. Prepare the text column (which contains the tweets) by removing special characters, fixing contractions and converting the date to the date format and converting the letters to lower case, then view the prepared data

```
removeSpecialChars <- function(x) gsub("[^a-zA-Z0-9 ]", " ", x)
texasdf$Text <- sapply(texasdf$Text, removeSpecialChars)
texasdf$Date <- as.Date(texasdf$Date)
texasdf$Text <- sapply(texasdf$Text, tolower)
fix.contractions <- function(doc) {
  doc <- gsub("won't", "will not", doc)
  doc <- gsub("can't", "can not", doc)
  doc <- gsub("n't", " not", doc)
  doc <- gsub("'ll", " will", doc)
  doc <- gsub("'re", " are", doc)
  doc <- gsub("'ve", " have", doc)
  doc <- gsub("'m", " am", doc)
  doc <- gsub("'d", " would", doc)
  doc <- gsub("'s", "", doc)
  return(doc)}
texasddf$Text <- sapply(texasdf$Text, fix.contractions)
str(texasdf[50, ]$Text, nchar.max = 300)
```

Figure 121: Preprocessing of tweets column

```
 chr "Big Pharma and our government used to spread viruses by chemtrails until people started understanding it  Then they
 want to releasing it in public and then they watch it spread and travel  Next year  we will have a vaccine for it just l
ike the avian flu  the swine flu  etc "
```

Figure 122: Example of a cleaned tweet

Analyse and Visualise Twitter Data

The Twitter data was prepared and analysed based on work by Liske, D. (2018(a)) and Liske, D. (2018(b))

1. Create a theme for the colors of the visuals and for the format of the ggplots

```
my_colors <- c("#E69F00", "#56B4E9", "#009E73", "#CC79A7", "#D55E00")
theme_lyrics <- function()
{theme(plot.title = element_text(hjust = 0.5),
       axis.text.x = element_blank(),
       axis.ticks = element_blank(),
       panel.grid.major = element_blank(),
       panel.grid.minor = element_blank(),
       legend.position = "none")}
```

Figure 123: Setting colors and theme for visuals

2. Unnest the tokens of the tweets (breaks them into individual words)

```
final_tweets_filtered <- texasdf %>%
  unnest_tokens(word, Text) %>%
  anti_join(stop_words) %>%
  distinct() %>%
  filter(nchar(word) > 3)
```

Figure 124: Unnesting tokens

3. Plot the most frequently occurring words in the tweets

```
raw_tweets <- final_tweets_filtered %>%
  count(word, sort = TRUE) %>%
  top_n(30) %>%
  ungroup() %>%
  mutate(word = reorder(word, n)) %>%
  ggplot() +
  geom_col(aes(word, n), fill = my_colors[4]) +
  theme(legend.position = "none",
        plot.title = element_text(hjust = 0.5),
        panel.grid.major = element_blank()) +
  xlab("Word") +
  ylab("Word Count") +
  ggtitle("Most Frequently Used Words In Texas Tweets") +
  coord_flip()
raw_tweets <- raw_tweets + theme(axis.text=element_text(size=20),
                                 axis.title=element_text(size=20))
raw_tweets <- raw_tweets + theme(plot.title = element_text(size=20, face="bold"))
raw_tweets
```

Figure 125: Plotting most frequently occuring words in tweets

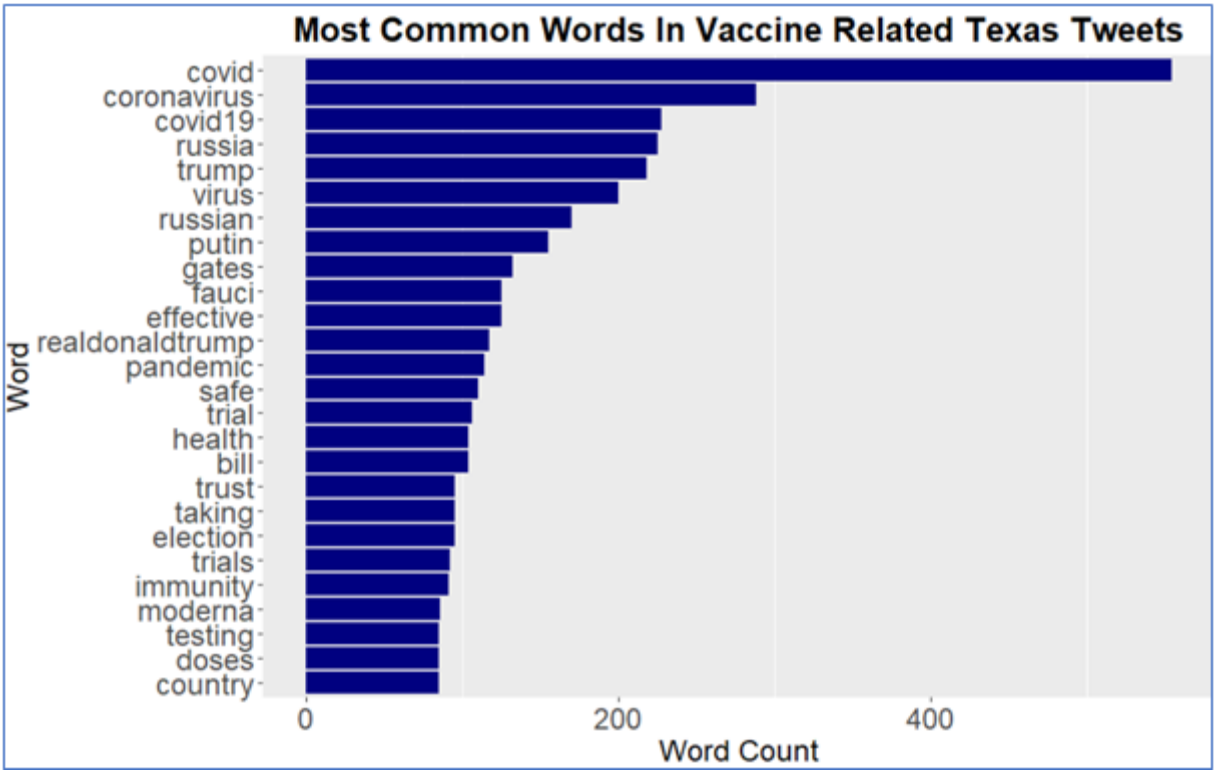

Figure 126: Plot of most common words in tweets

4. Define the words from the first plot that were undesirable (this is at your discretion)

```
undesirable_words <- c("https", "vaccine", "time",
                       "news", "vaccinated", "vaccines",
                       "vaccination", "doesn", "months", "anti",
                       "stop", "ready", "world", "people")
```

Figure 127: Defining undesirable words

5. Unnest the tokens again and plot the most frequently occurring words again, this time with the undesirable words removed (this plot is in the technical report)

```
final_tweet_filtered <- texasdf %>%
  unnest_tokens(word, Text) %>%
  anti_join(stop_words) %>%
  distinct() %>%
  filter(!word %in% undesirable_words) %>%
  filter(nchar(word) > 3)

r <- final_tweet_filtered %>%
  count(word, sort = TRUE) %>%
  top_n(25) %>%
  ungroup() %>%
  mutate(word = reorder(word, n)) %>%
  ggplot() +
  geom_col(aes(word, n), fill = my_colors[4]) +
  theme(legend.position = "none",
        plot.title = element_text(hjust = 0.5),
        panel.grid.major = element_blank()) +
  xlab("Word") +
  ylab("Word Count") +
  ggtitle("Most Common Words In Vaccine Related Texas Tweets")
  coord_flip()
z <- r + theme(axis.text=element_text(size=18),
               axis.title=element_text(size=18))
z + theme(plot.title = element_text(size=20, face="bold"))
```

Figure 128: Unnesting and plotting word frequency without undesirable words

6. Obtain the "afinn" sentiment lexicon (contains list of words and their sentiment score, obtain the score for the sentiments and convert these to either "positive" or "negative"

```
new_sentiments <- get_sentiments("afinn")
names(new_sentiments)[names(new_sentiments) == 'value'] <- 'score'
new_sentiments <- new_sentiments %>% mutate(lexicon = "afinn",
sentiment = ifelse(score >= 0, "positive", "negative"),
words_in_lexicon = n_distinct((word)))
```

Figure 129: Obtaining and preparing the afinn lexicon

7. Count the frequency of each word in the tweets and plot the wordcloud

```
tweets_words_counts <- final_tweet_filtered %>% dplyr::count(word, sort = TRUE)
wordcloud2(tweets_words_counts[1:13000, ], size = .5)
```

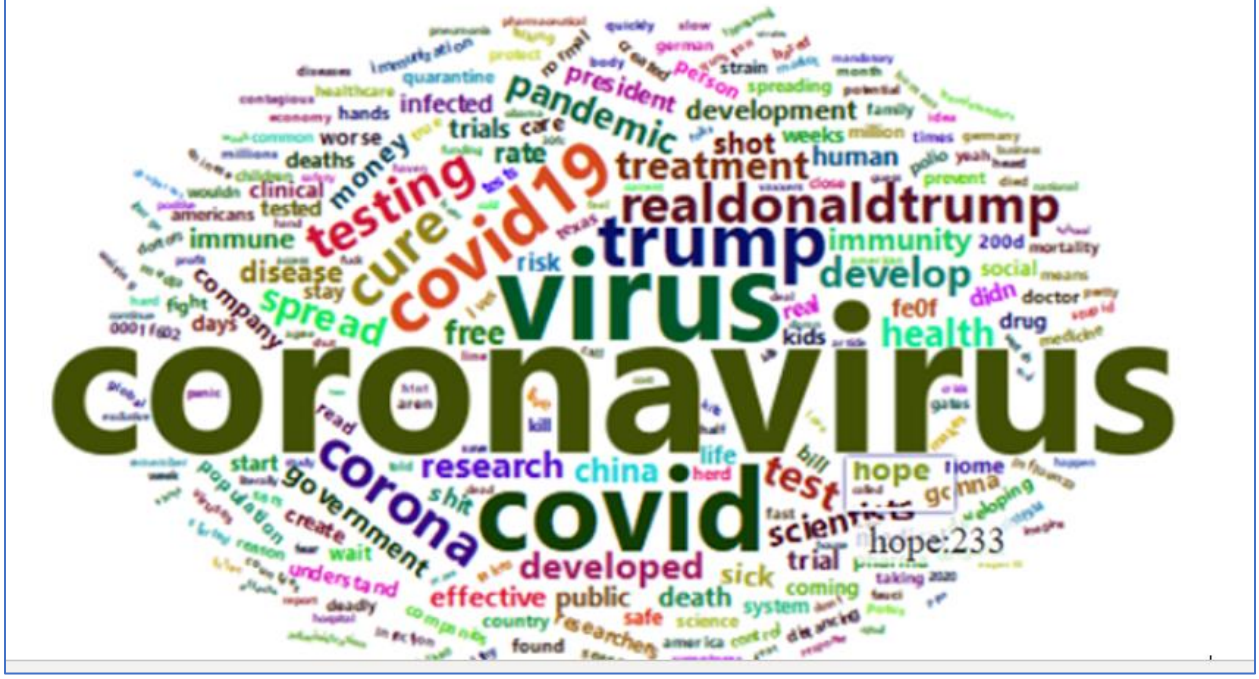

Figure 130: Word cloud of most common words in tweets

8. Join the unnested tokens from the twitter data with the “nrc” sentiment lexicon and remove the words that are “positive” or “negative”

```
texas_nrc <- final_tweet_filtered %>% inner_join(get_sentiments("nrc")) %>%
filter(!sentiment %in% c("positive", "negative"))
```

Figure 131: Finding sentiment of Texas tweets

9. Plot the most common emotions in the tweets (figure 4 in report)

```
syuzhet_vector <- get_sentiment(final_tweet_filtered$word, method="nrc")
nrc_plot <- texas_nrc %>%
  group_by(sentiment) %>%
  summarise(word_count = n()) %>%
  ungroup() %>%
  mutate(sentiment = reorder(sentiment, word_count)) %>%
  ggplot(aes(sentiment, word_count, fill = -word_count)) +
  geom_col() +
  guides(fill = FALSE) +
  theme_project() +
  labs(x = "Emotion", y = "Word Count") +
  scale_y_continuous(limits = c(0, 25000)) +
  ggtitle("Sentiment Of Vaccine Related Tweets In Texas") +
  coord_flip()
nrc_plot <- nrc_plot + theme(axis.text=element_text(size=18),
                             axis.title=element_text(size=18))
nrc_plot + theme(plot.title = element_text(size=20, face="bold"))
```

Figure 132: Plotting most common emotions in tweets

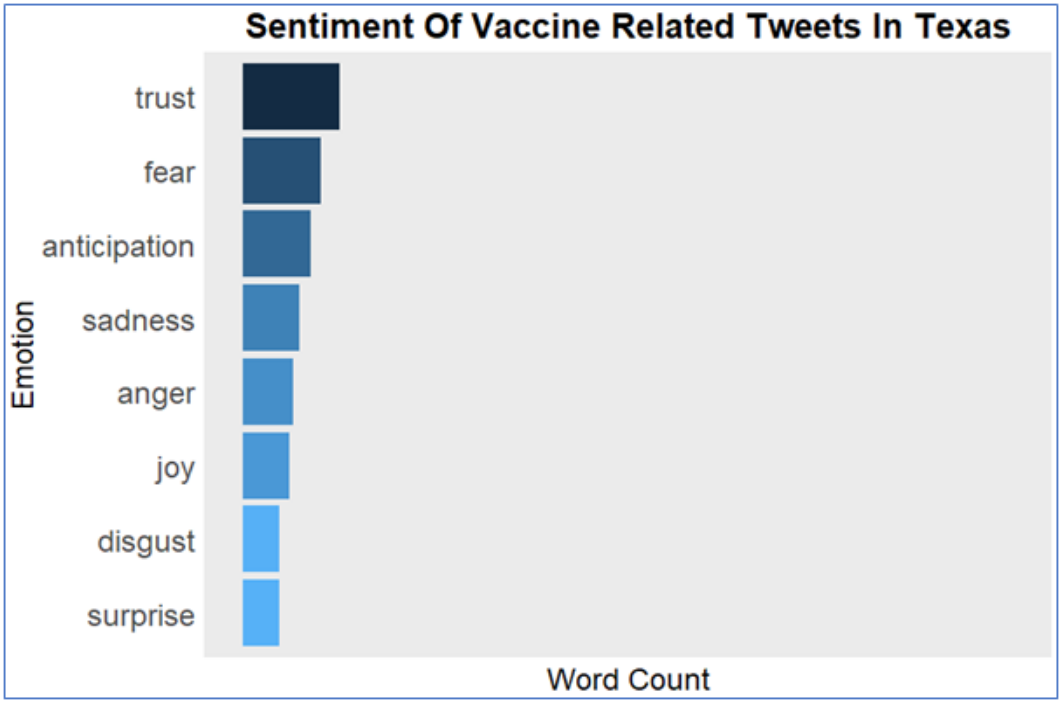

Figure 133: Plot of most common emotions in tweets

10. Plot the count of positive and negative words in the dataframe by joining the unnested tokens with the “bing” lexicon

```
vaccine_bing <- final_tweet_filtered %>% inner_join(get_sentiments("bing"))
sent_texas <- vaccine_bing %>% group_by(sentiment) %>%
  dplyr::summarise(word_count = dplyr::n()) %>%
  ungroup() %>% mutate(sentiment = reorder(sentiment, word_count)) %>%
  ggplot(aes(sentiment, word_count, fill = sentiment)) +
  geom_col() + guides(fill = FALSE) + theme_project() +
  labs(x = NULL, y = "Word Count") + scale_y_continuous(limits = c(0, 15000)) +
  ggtitle("Sentiment of Vaccine Related Tweets") + coord_flip()
sent_texas
```

Figure 134: Plotting frequency of positive/negative sentiments in tweets

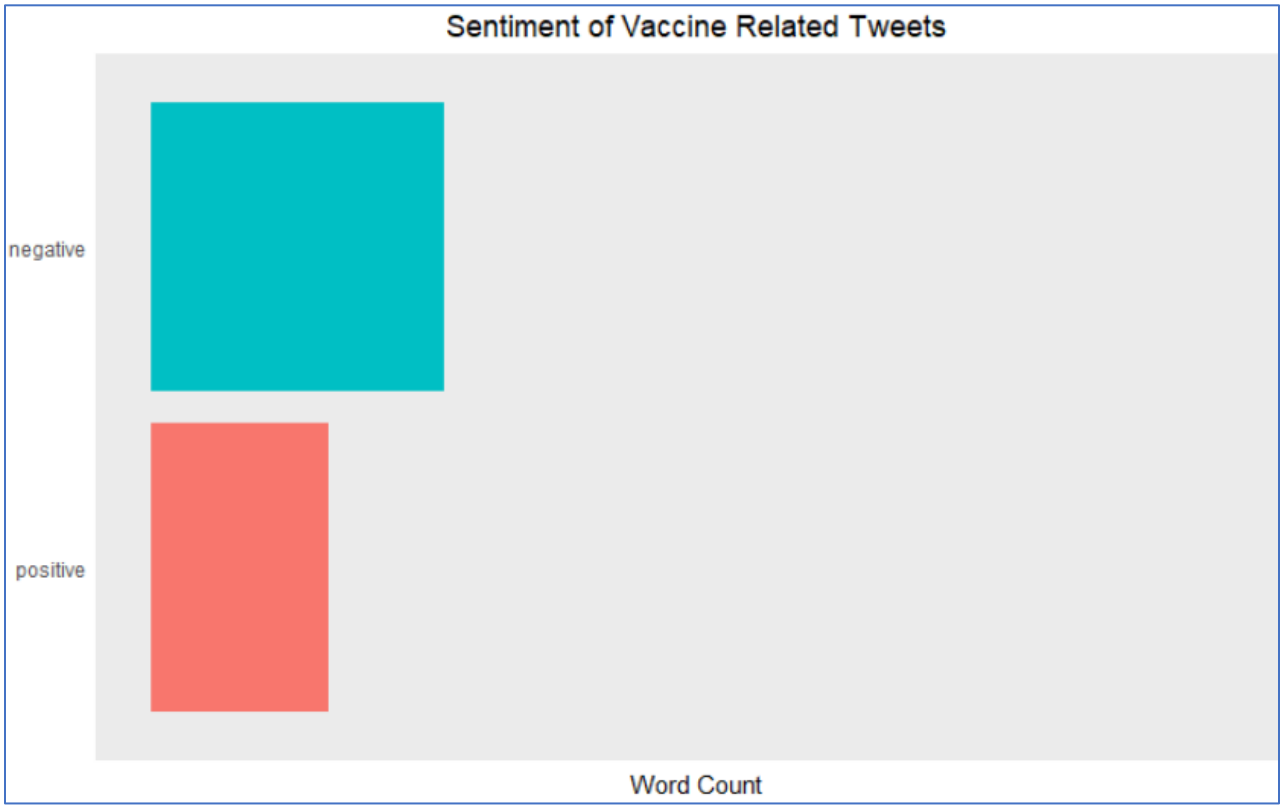

Figure 135: Plot of frequency of positive/negative sentiments in tweets

11. Generate a radar chart of the most common emotions in three selected locations

```
city_radar_chart <- year_sentiment_nrc %>%
  inner_join(total_sentiment_year, by = "Location") %>%
  mutate(percent = sentiment_year_count / year_total * 100 ) %>%
  filter(Location %in% c("Austin","Houston","San Angelo")) %>%
  select(-sentiment_year_count, -year_total) %>%
  spread(Location, percent) %>%
  chartJSRadar(showToolTipLabel = TRUE,
               main = "Vaccine Related Texas Tweets Radar")
city_radar_chart
```

Figure 136: Generating radar chart of most common emotions per location

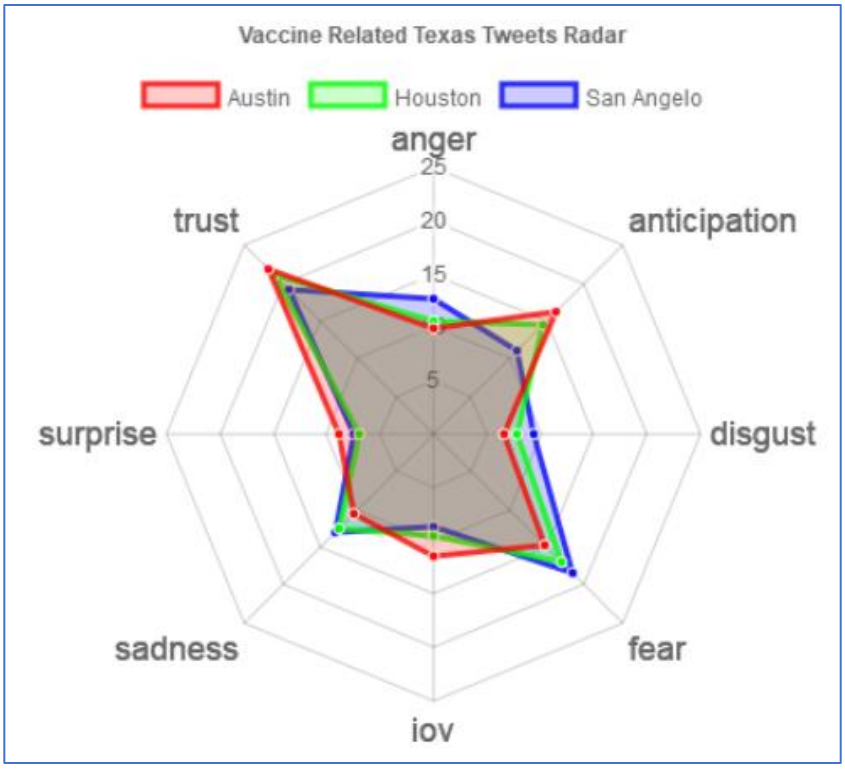

Figure 137: Radar chart of most common emotions per location

12. Unnest the tokens of the original tweets, but this time into bigrams (two word combinations) and prepare the bigrams by removing stopwords (such as "the") and undesirable words

```
vaccine_bigrams <- texasdf %>%
  unnest_tokens(bigram, Text, token = "ngrams", n = 2)
bigrams_separated <- vaccine_bigrams %>%
  separate(bigram, c("word1", "word2"), sep = " ")
bigrams_filtered <- bigrams_separated %>%
  filter(!word1 %in% stop_words$word) %>%
  filter(!word2 %in% stop_words$word) %>%
  filter(!word1 %in% undesirable_words) %>%
  filter(!word2 %in% undesirable_words)
```

Figure 138: Generating and preparing bigrams from tweets

13. Generate a plot of the most common bigrams in the tweets

```
bigram_texas <- bigrams_filtered %>% filter(word1 != word2) %>%
filter(Location != "NA") %>% unite(bigram, word1, word2, sep = " ") %>%
inner_join(texasdf) %>% count(bigram, Location, sort = TRUE) %>%
group_by(Location) %>% slice(seq_len(7)) %>% ungroup() %>%
arrange(Location, n) %>% mutate(row = row_number())

bigram_texas %>% ggplot(aes(row, n, fill = Location)) +
geom_col(show.legend = FALSE) + facet_wrap(~Location, scales = "free_y") +
xlab(NULL) + ylab(NULL) + scale_x_continuous(breaks = bigram_texas$row,
labels = bigram_texas$bigram) + theme_lyrics() + theme(panel.grid.major.x =
element_blank()) + ggtitle("Bigrams Per Region of Texas") + coord_flip()
```

Figure 139: Generating bigram plot

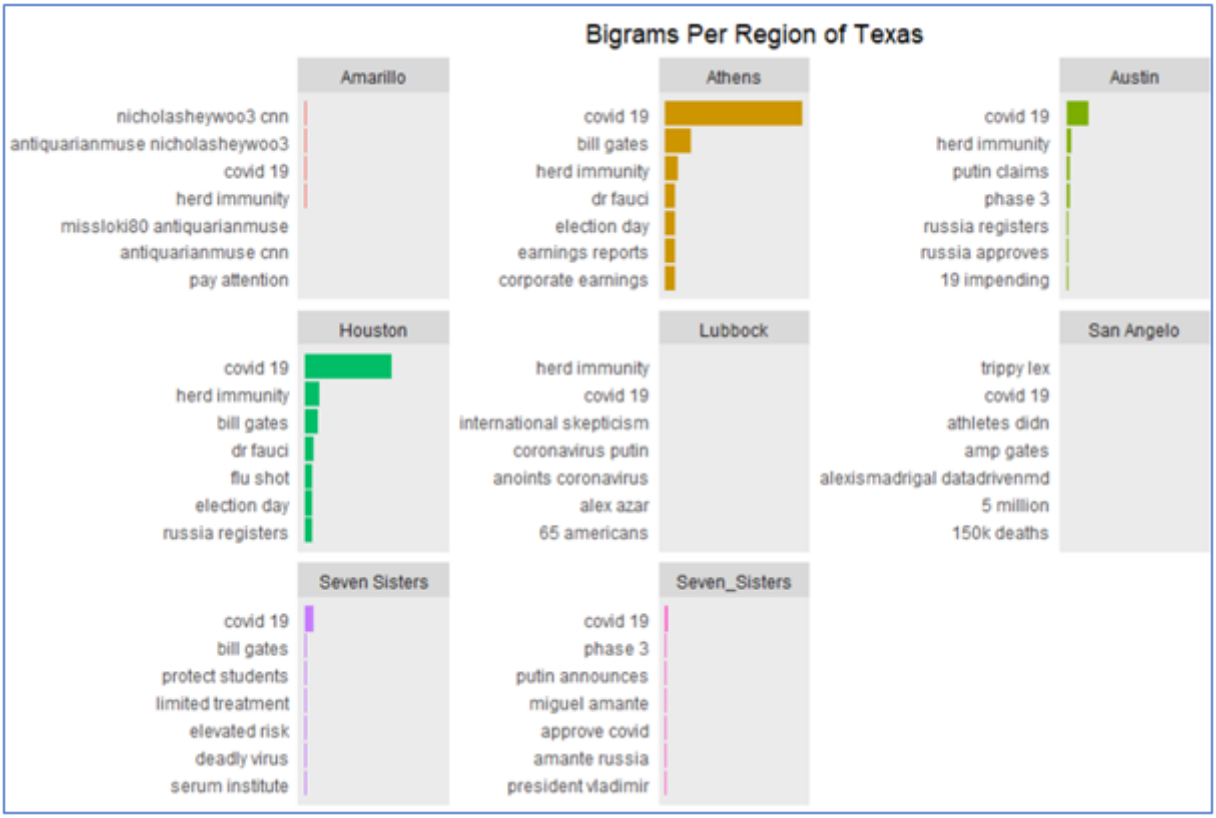

Figure 140: Plot of most common bigrams in tweets

14. Generate a plot for the frequency of tweets per location

```
tweets_by_location <- texas_location %>%
  count(Location) %>%
  select(Location, year_total = n)
tweets_by_location$Location <- factor(total_sentiment_year$Location,
                                      levels=c("Lubbock", "San Angelo", "Amarillo",
                                               "Seven Sisters", "Austin", "Houston", "Athens"))
p <- ggplot(data = tweets_by_location, aes(x = Location, y = year_total, fill = Location)) +
  geom_bar(stat="identity") + xlab("Tweet Count") + ggtitle("Number of Vaccine Related Tweets By Location")
  labs(x = "Location", y = "Number of Tweets")
q <- p + coord_flip()
q <- q + theme(legend.position = "none")
q <- q + theme(axis.text=element_text(size=18), axis.title=element_text(size=18))
q + theme(plot.title = element_text(size=20, face="bold"))
```

Figure 141: Generating plot for number of tweets per location

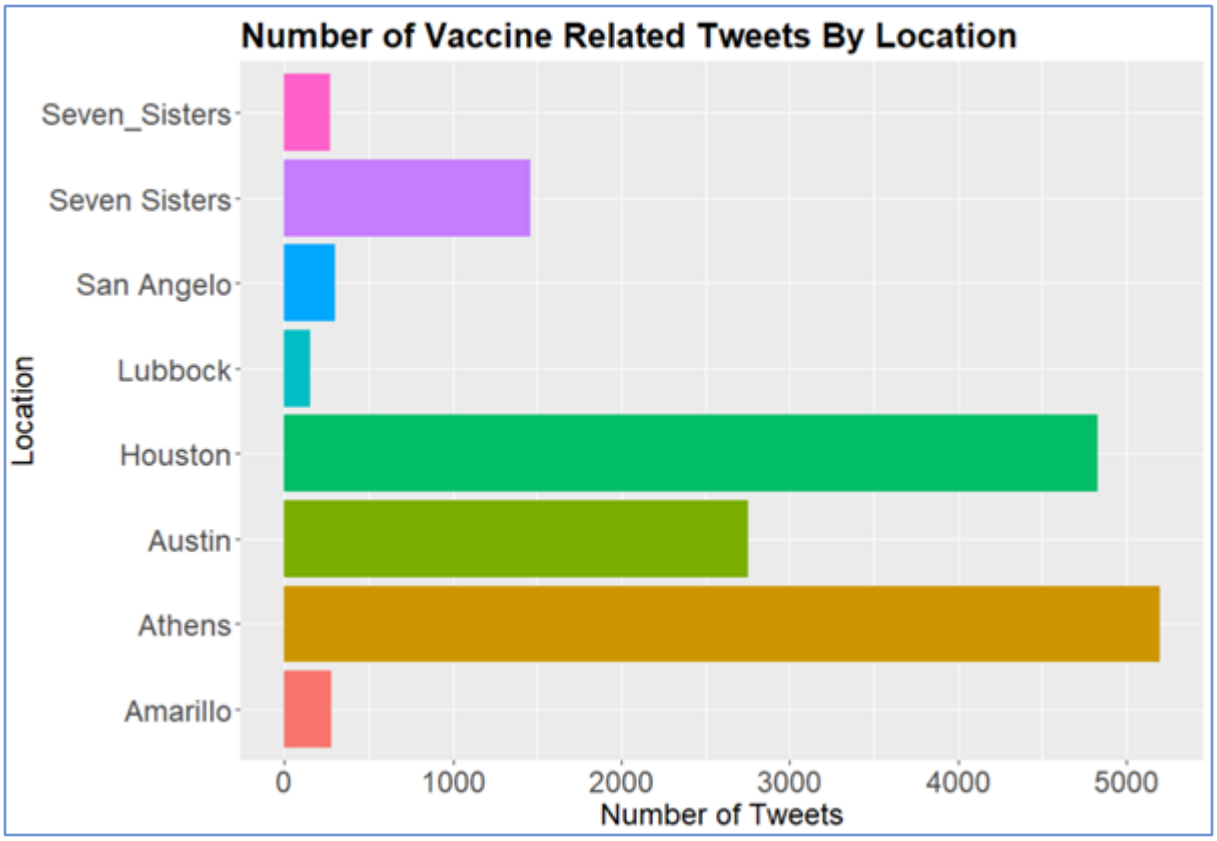

Figure 14: Plot of number of tweets per location

15. Get the sentiment of each word in the tweets, and count the frequecny of each word for each sentiment

```
texas_nrc <- final_tweet_filtered %>% inner_join(get_sentiments("nrc"))
texas_sentiment <- texas_nrc %>% group_by(sentiment) %>%
count(word, sort = TRUE) %>% arrange(desc(n)) %>%
slice(seq_len(10)) %>% ungroup()
```

Figure 143: Finding sentiment of most common words

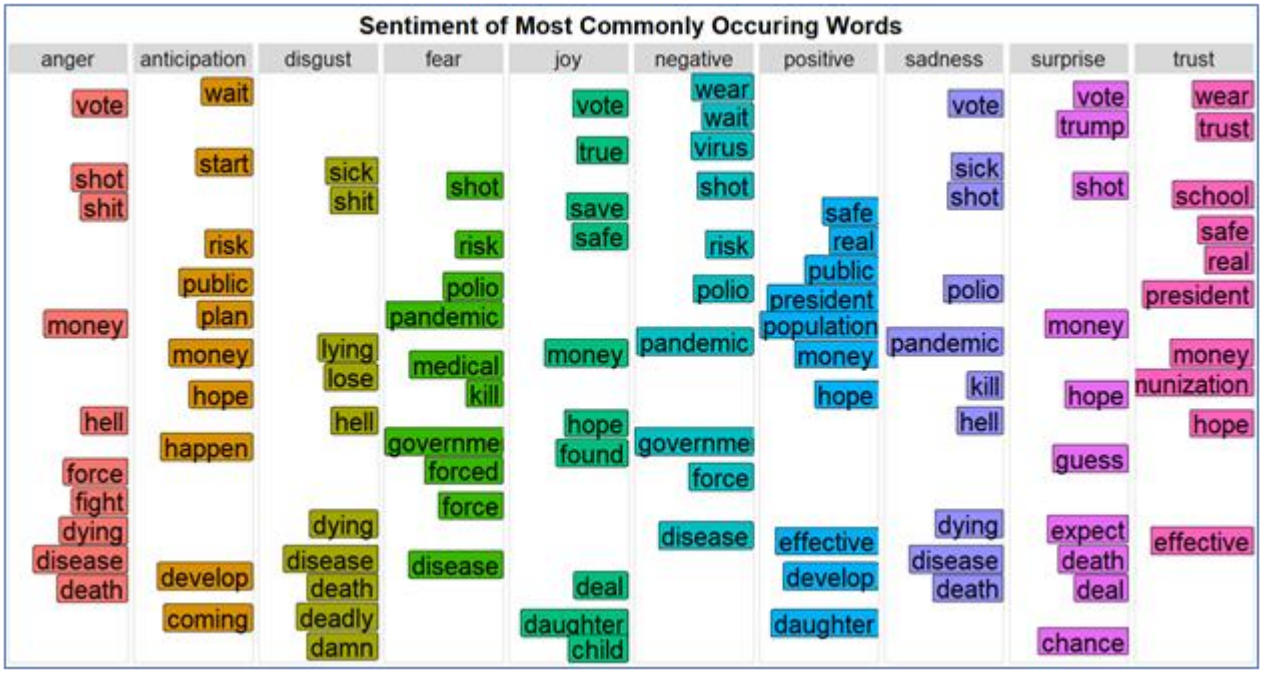

Figure 144: Plot of sentiment of most common words in tweets

Section 2: NIS Data Mining

NIS Data Mining

1. The working directory was set to where the NIS datasets were

```
getwd()
setwd("/Users/cianmannion/documents")
rawdf <- read.csv("raw.csv", header=T, na.strings = c(""), stringsAsFactors = T)
rawd <- read.csv("rawwer.csv", header=T, na.strings = c(""), stringsAsFactors = T)
```

Figure 145: Reading in raw NIS datasets

2. The columns containing the files chosen for the study were extracted

```
imbalanceddata <- rawdf[ ,c(1, 2, 4, 5, 7, 8, 10, 11, 14, 18, 19, 21, 22, 29, 32, 33, 38, 42)]
imbalanceddat <- rawd[ ,c(4, 11, 14, 16, 18, 19, 21, 23, 25, 29, 30, 36, 37, 45, 72, 176, 181, 183)]
```

Figure 146: Extracting columns of interest

3. The names column of the two datasets were changed to more descriptive names

```
colnames(imbalanceddat) = c("Adequate_Data", "Duration", "Household_Size", "Was_Child_Breastfed", "Child_Number", "WIC",
"Education_Status", "Firstborn", "Income_Group", "Mother_Age_Group", "Marital_Status", "Race", "House_Ownership_Status",
"Provider_Facility", "Vaccination_Status", "Insurance_Type", "Number_Providers", "Region")
colnames(imbalanceddata) = c("Adequate_Data", "Duration", "Household_Size", "Was_Child_Breastfed", "Child_Number", "WIC",
"Education_Status", "Firstborn", "Income_Group", "Mother_Age_Group", "Marital_Status", "Race", "House_Ownership_Status",
"Provider_Facility", "Vaccination_Status", "Insurance_Type", "Number_Providers", "Region")
```

Figure 147: Renaming NIS columns

4. The datasets were combined (in the form they are present in the upload file), missing values removed and written as a csv

```
imbalanceddata <- rbind(imbalanceddat, imbalanceddata)
imbalanceddata<-na.omit(imbalanceddata)
write.csv(imbalanceddata, file= "NIS_dataset.csv")
```

Figure 148: Merging, cleaning and writing the NIS dataset

5. View the factor level for each of the categorical variables

```
> table(imbalanceddata$Provider_Facility)

     ALL HOSPITAL FACILITIES ALL MILITARY/OTHER FACILITIES        ALL PRIVATE FACILITIES         ALL PUBLIC FACILITIES
                        4904                         810                          16539                          3296
                       MIXED                          NA      TYPE OF PROVIDER UNKNOWN
                        4526                         590                              1
> table(imbalanceddata$Education_Status)

                < 12 YEARS > 12 YEARS, NON-COLLEGE GRAD                    12 YEARS                 COLLEGE GRAD
                        3166                        7413                        5088                        14999
> table(imbalanceddata$House_Ownership_Status)

          DON'T KNOW      OTHER ARRANGMENT OWNED OR BEING BOUGHT             REFUSED              RENTED
                  41                        1091                18335              66               11133

> table(imbalanceddata$Firstborn)

   NO   YES
18620 12046
> table(imbalanceddata$Marital_Status)

                                                                 MARRIED
                                                                   22372
NEVER MARRIED/WIDOWED/DIVORCED/SEPARATED/DECEASED/LIVING WITH PARTNER
                                                                    8294
> table(imbalanceddata$Mother_Age_Group)

<= 29 YEARS >= 30 YEARS
       9999       20667
> table(imbalanceddata$Race)

                          HISPANIC            NON-HISPANIC BLACK ONLY NON-HISPANIC OTHER + MULTIPLE RACE
                              6567                               2183                                4163
          NON-HISPANIC WHITE ONLY
                             17753
> table(imbalanceddata$Number_Providers)

    0     1     2    3+
  304 21021  7683  1658
> table(imbalanceddata$Household_Size)

    2     3     4     5     6     7    8+
  680  6953 11226  6391  3002  1262  1152
> table(imbalanceddata$Insurance_Type)

                                                                  ANY MEDICAID
                                                                         11414
                                                                OTHER INSURANCE
                                                                          1337
OTHER INSURANCE (CHIP, IHS, MILITARY, OR OTHER, ALONE OR IN COMB. WITH PRIVATE INSURANCE)
                                                                          1299
                                                           PRIVATE INSURANCE ONLY
                                                                         15674
                                                                     UNINSURED
                                                                           942

> table(imbalanceddata$Was_Child_Breastfed)

DON'T KNOW         NO    REFUSED        YES
        82       3806          3      26775
> table(imbalanceddata$Child_Number)

FOUR OR MORE          ONE TWO OR THREE
        3882         8350        18434
> table(imbalanceddata$WIC)

        DON'T KNOW NEVER HEARD OF WIC                 NO            REFUSED                YES
                91                  61              17859                  7              12648
```

Figure 149: Factor levels for categorical variables

3. Remove undesired factor levels

```
> imbalanceddata <- imbalanceddata[!grepl("REFUSED|NEVER HEARD OF WIC|DON'T KNOW",
+                  imbalanceddata$WIC),]
> imbalanceddata <- imbalanceddata[!grepl("OTHER ARRANGMENT|REFUSED|DON'T KNOW",
+                  imbalanceddata$House_Ownership_Status),]
> imbalanceddata <- imbalanceddata[!grepl("DON'T KNOW|REFUSED", imbalanceddata$Was_Child_Breastfed),]
> imbalanceddata <- imbalanceddata[!grepl("NA|TYPE OF PROVIDER UNKNOWN|REFUSED",
+                  imbalanceddata$Provider_Facility),]
> imbalanceddata <- imbalanceddata[!grepl("NA", imbalanceddata$Firstborn),]
```

Figure 150: Removing undesired factor levels

4. View the class of each variable

```
> sapply(imbalanceddata, class)
                         X           Adequate_Data                Duration          Household_Size
                 "integer"                "factor"               "integer"                "factor"
   Was_Child_Breastfed               Child_Number                     WIC          Education_Status
                 "factor"                "factor"                "factor"                "factor"
              Firstborn               Income_Group       Mother_Age_Group           Marital_Status
                 "factor"                "numeric"                "factor"                "factor"
                    Race  House_Ownership_Status       Provider_Facility       Vaccination_Status
                 "factor"                "factor"                "factor"                "factor"
        Insurance_Type         Number_Providers                  Region
                 "factor"                "factor"                "factor"
```

Figure 151: Viewing classes of NIS variables

5. Remove ID column and column with whether or not the child had adequate provider data and change the "Duration" variable to numeric datatype

```
imbalanceddata <- imbalanceddata[ ,c(3:19)]
imbalanceddata$Duration <- as.numeric(imbalanceddata$Duration)
```

Figure 152: Removing unwanted columns and converting duration to numeric

6. Remove the empty factor levels and view the cleaned data

```
> imbalanceddata$Provider_Facility <- factor(imbalanceddata$Provider_Facility)
> imbalanceddata$Was_Child_Breastfed <- factor(imbalanceddata$Was_Child_Breastfed)
> imbalanceddata$House_Ownership_Status <- factor(imbalanceddata$House_Ownership_Status)
> imbalanceddata$WIC <- factor(imbalanceddata$WIC)
> imbalanceddata$Number_Providers <- factor(imbalanceddata$Number_Providers)
> imbalanceddata$Insurance_Type <- factor(imbalanceddata$Insurance_Type)
> imbalanceddata$Firstborn <- factor(imbalanceddata$Firstborn)
```

Figure 153: Removing empty factor levels

```
> table(imbalanceddata$Provider_Facility)

      ALL HOSPITAL FACILITIES ALL MILITARY/OTHER FACILITIES        ALL PRIVATE FACILITIES
                         4703                          777                         15864
        ALL PUBLIC FACILITIES                        MIXED
                         3073                         4282
> table(imbalanceddata$WIC)

   NO   YES
17043 11656
> table(imbalanceddata$House_Ownership_Status)

OWNED OR BEING BOUGHT                RENTED
                17831                 10868
> table(imbalanceddata$Number_Providers)

    1     2    3+
19909  7251  1539
> table(imbalanceddata$Was_Child_Breastfed)

   NO   YES
 3531 25168
```

Figure 154: Viewing cleaned factors

7. Convert the target column factor levels to “Vaccinated” and “Unvaccinated”

```
levels(imbalanceddata$Vaccination_Status) <- c("Unvaccinated", "Vaccinated")
```

Figure 155: Renaming target column factor levels

8. Find the variance, range and mean of the “Duration” and “Income Group” columns

```
> range(imbalanceddata$Duration)
[1]  1 72
> range(imbalanceddata$Income_Group)
[1] 0.5 3.0
> mean(imbalanceddata$Duration)
[1] 34.71494
> mean(imbalanceddata$Income_Group)
[1] 2.119357
> var(imbalanceddata$Duration)
[1] 329.9301
> var(imbalanceddata$Income_Group)
[1] 0.9693978
```

Figure 156: Exploratory analysis of numerical variables

Visualisation of NIS Data

1. Convert income into a factor with three levels (“low”, “medium”, “high”)

```
imbalanceddata$Income_Level <- cut(imbalanceddata$Income_Group, 3,
include.lowest=TRUE, labels=c("Low", "Medium", "High"))
```

Figure 157: Converting income into a factor column

2. Clean the variables by changing the factor levels to shorter names that are easier to visualise, and reorder factors to desired order

```
imbalanceddata$Insurance_Type <- revalue(imbalanceddata$Insurance_Type,
c("Medicaid"="Medicaid", "Other Insurance"="Other", "Private Insurance" =
"Private", "Uninsured"="Uninsured",  "Public"="Public"))
imbalanceddata$Race <- revalue(imbalanceddata$Race, c("HISPANIC" = "Hispanic",
"NON-HISPANIC OTHER + MULTIPLE RACE" = "Mixed", "NON-HISPANIC BLACK ONLY" = "Black",
"NON-HISPANIC WHITE ONLY" = "White"))
imbalanceddata$Race <- factor(imbalanceddata$Race,
levels = c("White", "Mixed", "Hispanic", "Black"))
imbalanceddata$Education_Status <- revalue(imbalanceddata$Education_Status,
c("< 12 YEARS"="<12y", "> 12 YEARS, NON-COLLEGE GRAD"=">12y NG",
"12 YEARS" = "12y", "COLLEGE GRAD"="Graduate"))
imbalanceddata$Education_Status <- factor(imbalanceddata$Education_Status,
levels=c("Graduate", ">12y NG", "12y", "<12y"))
imbalanceddata$Child_Number <- factor(imbalanceddata$Child_Number,
levels=c("ONE", "TWO OR THREE", "FOUR OR MORE"))
imbalanceddata$Child_Number <- revalue(imbalanceddata$Child_Number,
c("ONE"= "1", "TWO OR THREE" = "2 or 3", "FOUR OR MORE" = ">=4"))
imbalanceddata$Marital_Status <- revalue(imbalanceddata$Marital_Status,
c("NEVER MARRIED/WIDOWED/DIVORCED/SEPARATED/DECEASED/LIVING WITH PARTNER" = "Unmarried",
"MARRIED"="Married"))
imbalanceddata$Provider_Facility <- revalue(imbalanceddata$Provider_Facility,
c("ALL HOSPITAL FACILITIES" = "All", "ALL MILITARY/OTHER FACILITIES"="Military",
"ALL PRIVATE FACILITIES" = "Private", "ALL PUBLIC FACILITIES" = "Public", "MIXED"="Mixed"))
imbalanceddata$Provider_Facility <- factor(imbalanceddata$Provider_Facility,
levels=c("Private", "Mixed", "All", "Military", "Public"))
```

Figure 158: Renaming and reordering factor levels

3. Generate plots for each factor and plot these together using the plot_grid function (from “cowplot” package). These are found in figure 5 in the technical report

```
plot16 <-ggplot(data = imbalanceddata[!is.na(imbalanceddata$Income_Level), ],
aes(x = Income_Level,fill = Vaccination_Status)) +
geom_bar(position="fill") + ylab("count") + my_theme + theme(legend.position = "none")
plot10 <-ggplot(data = imbalanceddata[!is.na(imbalanceddata$Insurance_Type), ],
aes(x = Insurance_Type,fill = Vaccination_Status)) +
geom_bar(position="fill") + ylab("Count") + theme(legend.position = "none") + smaller_theme
plot15 <-ggplot(data = imbalanceddata[!is.na(imbalanceddata$Race), ],
aes(x = Race, fill = Vaccination_Status)) +
geom_bar(position="fill") + ylab("count") + smaller_theme + theme(legend.position = "none")
plot3 <-ggplot(data = imbalanceddata[!is.na(imbalanceddata$Child_Number), ],
aes(x = Child_Number,fill = Vaccination_Status)) +
geom_bar(position="fill") + theme(legend.position = "none") + my_theme
plot1 <-ggplot(data = imbalanceddata[!is.na(imbalanceddata$Household_Size), ],
aes(x = Household_Size, fill = Vaccination_Status)) +
geom_bar(position="fill") + theme(legend.position = "none") + my_theme
plot5 <-ggplot(data = imbalanceddata[!is.na(imbalanceddata$Education_Status), ],
aes(x = Education_Status,fill = Vaccination_Status)) +
geom_bar(position="fill") + theme(legend.position = "none") + smaller_theme
plot_grid(plot1, plot3, plot5, plot16, plot10, plot15, nrow = 3, ncol = 2,
labels = "AUTO", label_size = 18, align = "v")
```

Figure 159: Plotting variables relationship with vaccination status

4. Generate the other barplots and plot them together using plot_grid

```
plot16 <-ggplot(data = imbalanceddata[!is.na(imbalanceddata$Income_Level), ],
aes(x = Income_Level,fill = Vaccination_Status)) +
geom_bar(position="fill") + ylab("count") + my_theme + theme(legend.position = "none")
plot10 <-ggplot(data = imbalanceddata[!is.na(imbalanceddata$Insurance_Type), ],
aes(x = Insurance_Type,fill = Vaccination_Status)) +
geom_bar(position="fill") + ylab("Count") + theme(legend.position = "none") + smaller_theme
plot15 <-ggplot(data = imbalanceddata[!is.na(imbalanceddata$Race), ],
aes(x = Race, fill = Vaccination_Status)) +
geom_bar(position="fill") + ylab("count") + smaller_theme + theme(legend.position = "none")
plot3 <-ggplot(data = imbalanceddata[!is.na(imbalanceddata$Child_Number), ],
aes(x = Child_Number,fill = Vaccination_Status)) +
geom_bar(position="fill") + theme(legend.position = "none") + my_theme
plot1 <-ggplot(data = imbalanceddata[!is.na(imbalanceddata$Household_Size), ],
aes(x = Household_Size, fill = Vaccination_Status)) +
geom_bar(position="fill") + theme(legend.position = "none") + my_theme
plot5 <-ggplot(data = imbalanceddata[!is.na(imbalanceddata$Education_Status), ],
aes(x = Education_Status,fill = Vaccination_Status)) +
geom_bar(position="fill") + theme(legend.position = "none") + smaller_theme
plot_grid(plot1, plot3, plot5, plot16, plot10, plot15, nrow = 3, ncol = 2,
labels = "AUTO", label_size = 18, align = "v")
```

Figure 160: Plotting relationship between variables with vaccination status

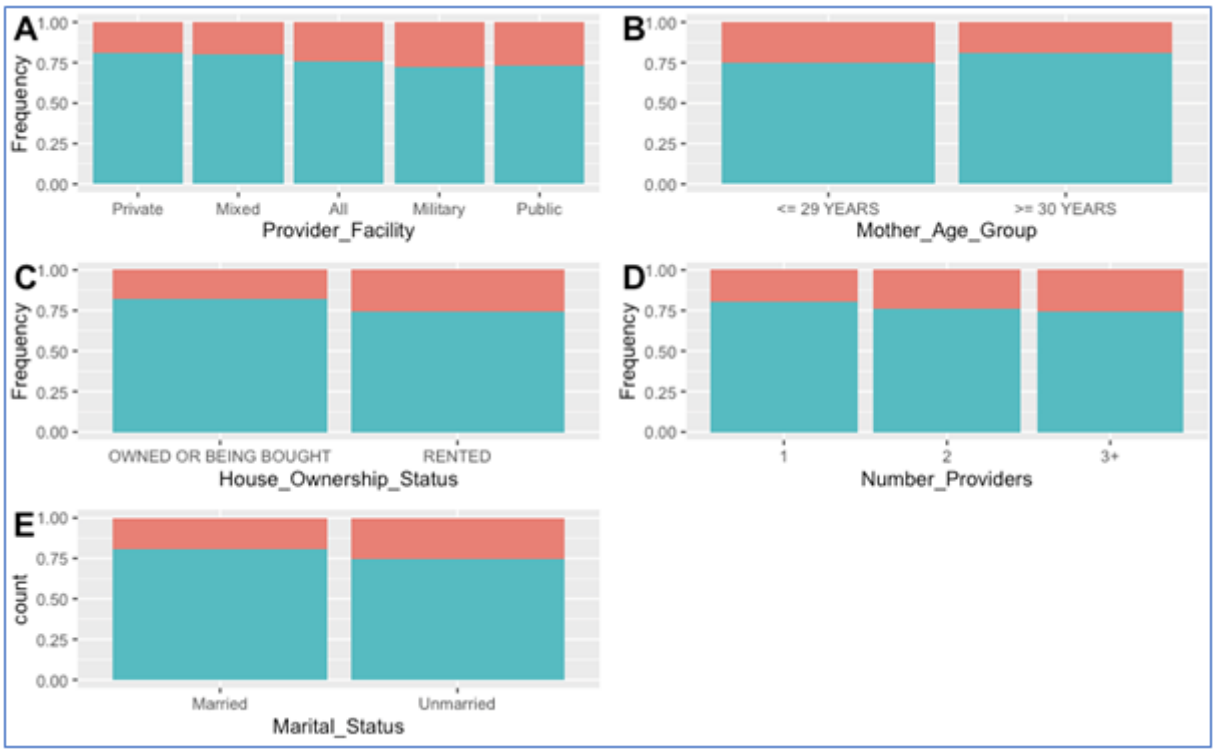

Figure 161: Plots of variables relationship with vaccination status

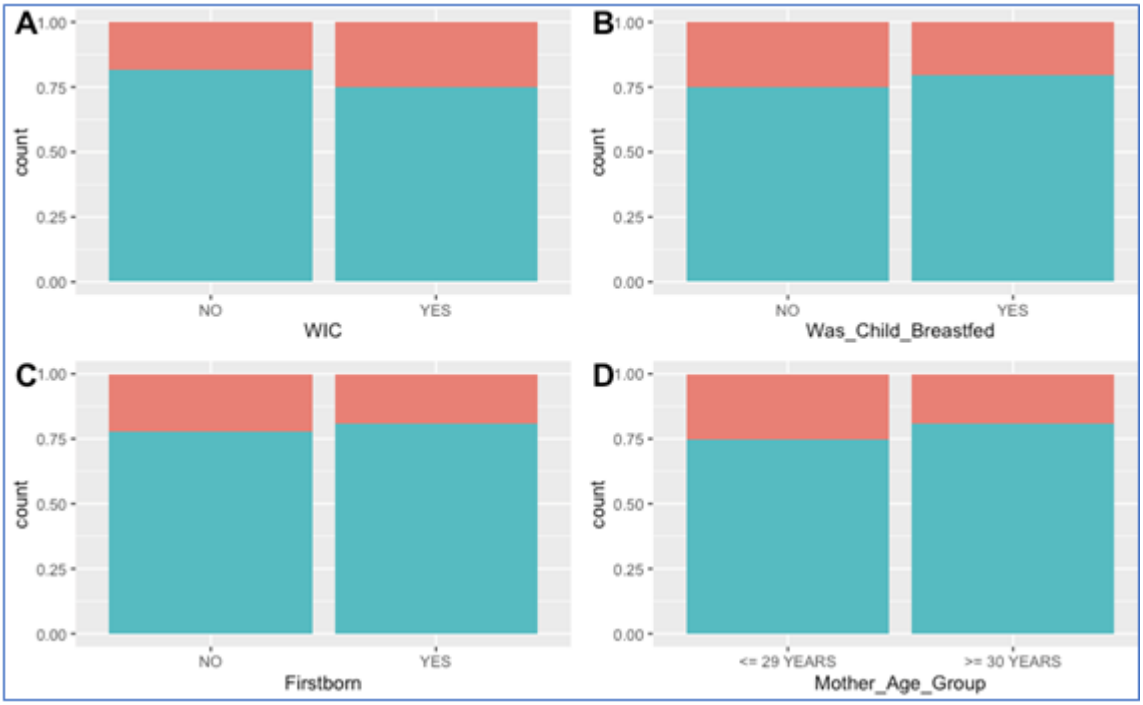

Figure 163: Plots of variables relationship with vaccination status

5. Clean the data by removing unwanted answers from the survey

```
imbalanceddata <- imbalanceddata[!grepl("DON'T KNOW|REFUSED|OTHER ARRANGMENT",
imbalanceddata$House_Ownership_Status),]
imbalanceddata <- imbalanceddata[!grepl("NA", imbalanceddata$Insurance_Type),]
imbalanceddata <- imbalanceddata[!grepl("DON'T KNOW|REFUSED",
imbalanceddata$Was_Child_Breastfed),]
imbalanceddata <- imbalanceddata[!grepl("REFUSED|DON'T KNOW|NEVER HEARD OF WIC",
imbalanceddata$WIC),]
imbalanceddata <- imbalanceddata[!grepl("0", imbalanceddata$Number_Providers),]
imbalanceddata <- imbalanceddata[!grepl("TYPE OF PROVIDER UNKNOWN",
imbalanceddata$Provider_Facility),]
imbalanceddata <- imbalanceddata[!grepl("NA", imbalanceddata$Provider_Facility),]
```

Figure 164: Removing rows from NIS data

6. Generate a boxplot and histogram of continuous variables (shown as figure

```
bf_histogram <- qplot(imbalanceddata$Duration,
geom="histogram", main = "Histogram for Breastfeeding Duration",  xlab =
"Breastfeeding Duration", fill = 'red') + theme(legend.position = "none") + my_theme
income_histogram <- qplot(imbalanceddata$Income_Group,
geom="histogram", main = "Histogram for Income",
xlab = "Income", fill = 'red') + theme(legend.position = "none") + my_theme
boxplot_bf <- ggplot(imbalanceddata, aes(x = Duration)) + geom_boxplot(fill = 'red') +
coord_flip() +
ggtitle("Boxplot of Breastfeeding Duration") + my_theme
boxplot_income <- ggplot(imbalanceddata, aes(x = Income_Group)) +
geom_boxplot(fill = 'red') + coord_flip() +
ggtitle("Boxplot of Income")  + my_theme
plot_grid(boxplot_bf, boxplot_income, bf_histogram, income_histogram, nrow = 2, ncol = 2,
labels = "AUTO", label_size = 20, align = "v")
```

Figure 165: Generating histograms and boxplots of continuous variables

7. Perform chi-squared tests to measure the relationship between the categorical variables and Vaccination Status. $P < 0.05$ means the relationship is significant

```
> chisq.test(imbalanceddata$Vaccination_Status, imbalanceddata$Household_Size)

        Pearson's Chi-squared test

data:  imbalanceddata$Vaccination_Status and imbalanceddata$Household_Size
X-squared = 320.25, df = 6, p-value < 2.2e-16

> chisq.test(imbalanceddata$Vaccination_Status, imbalanceddata$Was_Child_Breastfed)

        Pearson's Chi-squared test with Yates' continuity correction

data:  imbalanceddata$Vaccination_Status and imbalanceddata$Was_Child_Breastfed
X-squared = 41.502, df = 1, p-value = 1.178e-10

> chisq.test(imbalanceddata$Vaccination_Status, imbalanceddata$Child_Number)

        Pearson's Chi-squared test

data:  imbalanceddata$Vaccination_Status and imbalanceddata$Child_Number
X-squared = 213.04, df = 2, p-value < 2.2e-16

> chisq.test(imbalanceddata$Vaccination_Status, imbalanceddata$WIC)

        Pearson's Chi-squared test with Yates' continuity correction

data:  imbalanceddata$Vaccination_Status and imbalanceddata$WIC
X-squared = 185.46, df = 1, p-value < 2.2e-16

> chisq.test(imbalanceddata$Vaccination_Status, imbalanceddata$Education_Status)

        Pearson's Chi-squared test

data:  imbalanceddata$Vaccination_Status and imbalanceddata$Education_Status
X-squared = 359.6, df = 3, p-value < 2.2e-16


> chisq.test(imbalanceddata$Vaccination_Status, imbalanceddata$Firstborn)

        Pearson's Chi-squared test with Yates' continuity correction

data:  imbalanceddata$Vaccination_Status and imbalanceddata$Firstborn
X-squared = 49.48, df = 1, p-value = 2.004e-12

> chisq.test(imbalanceddata$Vaccination_Status, imbalanceddata$Mother_Age_Group)

        Pearson's Chi-squared test with Yates' continuity correction

data:  imbalanceddata$Vaccination_Status and imbalanceddata$Mother_Age_Group
X-squared = 161, df = 1, p-value < 2.2e-16

> chisq.test(imbalanceddata$Vaccination_Status, imbalanceddata$Marital_Status)

        Pearson's Chi-squared test with Yates' continuity correction

data:  imbalanceddata$Vaccination_Status and imbalanceddata$Marital_Status
X-squared = 133.66, df = 1, p-value < 2.2e-16

> chisq.test(imbalanceddata$Vaccination_Status, imbalanceddata$Race)

        Pearson's Chi-squared test

data:  imbalanceddata$Vaccination_Status and imbalanceddata$Race
X-squared = 77.789, df = 3, p-value < 2.2e-16
```

Figure 165: First set of Chi-Squared tests

```
> chisq.test(imbalanceddata$Vaccination_Status, imbalanceddata$Firstborn)

        Pearson's Chi-squared test with Yates' continuity correction

data:  imbalanceddata$Vaccination_Status and imbalanceddata$Firstborn
X-squared = 49.48, df = 1, p-value = 2.004e-12

> chisq.test(imbalanceddata$Vaccination_Status, imbalanceddata$Mother_Age_Group)

        Pearson's Chi-squared test with Yates' continuity correction

data:  imbalanceddata$Vaccination_Status and imbalanceddata$Mother_Age_Group
X-squared = 161, df = 1, p-value < 2.2e-16

> chisq.test(imbalanceddata$Vaccination_Status, imbalanceddata$Marital_Status)

        Pearson's Chi-squared test with Yates' continuity correction

data:  imbalanceddata$Vaccination_Status and imbalanceddata$Marital_Status
X-squared = 133.66, df = 1, p-value < 2.2e-16

> chisq.test(imbalanceddata$Vaccination_Status, imbalanceddata$Race)

        Pearson's Chi-squared test

data:  imbalanceddata$Vaccination_Status and imbalanceddata$Race
X-squared = 77.789, df = 3, p-value < 2.2e-16

> chisq.test(imbalanceddata$Vaccination_Status, imbalanceddata$House_Ownership_Status)

        Pearson's Chi-squared test with Yates' continuity correction

data:  imbalanceddata$Vaccination_Status and imbalanceddata$House_Ownership_Status
X-squared = 227.9, df = 1, p-value < 2.2e-16

> chisq.test(imbalanceddata$Vaccination_Status, imbalanceddata$Insurance_Type)

        Pearson's Chi-squared test

data:  imbalanceddata$Vaccination_Status and imbalanceddata$Insurance_Type
X-squared = 502.46, df = 4, p-value < 2.2e-16

> chisq.test(imbalanceddata$Vaccination_Status, imbalanceddata$Number_Providers)

        Pearson's Chi-squared test

data:  imbalanceddata$Vaccination_Status and imbalanceddata$Number_Providers
X-squared = 68.556, df = 2, p-value = 1.298e-15

> chisq.test(imbalanceddata$Vaccination_Status, imbalanceddata$Region)

        Pearson's Chi-squared test

data:  imbalanceddata$Vaccination_Status and imbalanceddata$Region
X-squared = 179.93, df = 50, p-value < 2.2e-16

> chisq.test(imbalanceddata$Vaccination_Status, imbalanceddata$Provider_Facility)

        Pearson's Chi-squared test

data:  imbalanceddata$Vaccination_Status and imbalanceddata$Provider_Facility
X-squared = 1112.6, df = 6, p-value < 2.2e-16
```

Figure 166: Second set of Chi-Squared tests

8. Perform T-Testing to measure the relationship between income/duration with vaccination status. P < 0.05 means the relationship is significant

```
> t.test(imbalanceddata$Duration ~ imbalanceddata$Vaccination_Status,var.equal=TRUE)

        Two Sample t-test

data:  imbalanceddata$Duration by imbalanceddata$Vaccination_Status
t = 6.1381, df = 28962, p-value = 8.458e-10
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 0.05937092 0.11507575
sample estimates:
mean in group Unvaccinated   mean in group Vaccinated
                0.06816393                -0.01905940

> t.test(imbalanceddata$Income_Group ~ imbalanceddata$Vaccination_Status,var.equal=TRUE)

        Two Sample t-test

data:  imbalanceddata$Income_Group by imbalanceddata$Vaccination_Status
t = -20.636, df = 28962, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.3189577 -0.2636219
sample estimates:
mean in group Unvaccinated   mean in group Vaccinated
                -0.2276393                 0.0636505
```

Figure 167: T-Test of continuous variables with Vaccination Status

9. Perform ANOVA tests to measure the relationship between income with variables of interest

```
> grouped_education <- group_by(imbalanceddata, Education_Status)
> summarise(grouped_education, group_mean = mean(Income_Group, na.rm=TRUE))
# A tibble: 4 x 2
  Education_Status             group_mean
  <fct>                             <dbl>
1 < 12 YEARS                       -1.14
2 > 12 YEARS, NON-COLLEGE GRAD     -0.243
3 12 YEARS                         -0.735
4 COLLEGE GRAD                      0.588
> education_ANOVA <- lm(Income_Group ~ Education_Status, data =imbalanceddata)
> anova(education_ANOVA)
Analysis of Variance Table

Response: Income_Group
                    Df Sum Sq Mean Sq F value    Pr(>F)
Education_Status     3  11724  3908.1  6565.4 < 2.2e-16 ***
Residuals        28960  17239     0.6
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Figure 168: ANOVA test between income and education

```
> grouped_household <- group_by(imbalanceddata, Household_Size)
> summarise(grouped_household, group_mean = mean(Income_Group, na.rm=TRUE))
# A tibble: 7 x 2
  Household_Size group_mean
  <fct>               <dbl>
1 2                  -0.537
2 3                   0.285
3 4                   0.203
4 5                  -0.166
5 6                  -0.403
6 7                  -0.663
7 8+                 -0.836
> household_ANOVA <- lm(Income_Group ~ Household_Size, data =imbalanceddata)
> anova(household_ANOVA)
Analysis of Variance Table

Response: Income_Group
                Df  Sum Sq Mean Sq F value    Pr(>F)
Household_Size   6  2991.9  498.64  555.97 < 2.2e-16 ***
Residuals    28957 25971.1    0.90
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Figure 169: ANOVA test between income and household size

```
> grouped_race <- group_by(imbalanceddata, Race)
> summarise(grouped_race, group_mean = mean(Income_Group, na.rm=TRUE))
# A tibble: 4 x 2
  Race                                group_mean
  <fct>                                    <dbl>
1 HISPANIC                                -0.609
2 NON-HISPANIC BLACK ONLY                 -0.539
3 NON-HISPANIC OTHER + MULTIPLE RACE       0.0346
4 NON-HISPANIC WHITE ONLY                  0.281
> race_ANOVA <- lm(Income_Group ~ Race, data =imbalanceddata)
> anova(race_ANOVA)
Analysis of Variance Table

Response: Income_Group
             Df Sum Sq Mean Sq F value    Pr(>F)
Race          3   4221 1407.02  1646.9 < 2.2e-16 ***
Residuals 28960  24742    0.85
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Figure 170: ANOVA test between income and race

```
> grouped_insurance <- group_by(imbalanceddata, Insurance_Type)
> summarise(grouped_insurance, group_mean = mean(Income_Group, na.rm=TRUE))
# A tibble: 5 x 2
  Insurance_Type                                                                group_mean
  <fct>                                                                          <dbl>
1 ANY MEDICAID                                                                  -0.912
2 OTHER INSURANCE                                                                0.0951
3 OTHER INSURANCE (CHIP, IHS, MILITARY, OR OTHER, ALONE OR IN COMB. WITH PRIVATE INSURANCE)     0.0905
4 PRIVATE INSURANCE ONLY                                                         0.642
5 UNINSURED                                                                     -0.432
> insurance_ANOVA <- lm(Income_Group ~ Insurance_Type, data =imbalanceddata)
> anova(insurance_ANOVA)
Analysis of Variance Table

Response: Income_Group
                Df Sum Sq Mean Sq F value    Pr(>F)
Insurance_Type    4  15150  3787.4  7940.2 < 2.2e-16 ***
Residuals     28959  13813     0.5
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Figure 171: ANOVA test between income and insurance type

9. Normalize the numerical variables

```
imbalanceddata$Duration <- normalize(imbalanceddata$Duration,
method = "standardize", range = c(0, 1),
margin = 1L, on.constant = "quiet")
imbalanceddata$Income_Group <- normalize(imbalanceddata$Income_Group,
method = "standardize", range = c(0, 1),
margin = 1L, on.constant = "quiet")
```

Figure 172: Normalizing duration and income variables

10. Remove any unwanted columns

```
imbalanceddata <- imbalanceddata[ ,c(3:19)]
```

Figure 173: Removing unwanted columns

11. Convert the data into training and test data

```
sample <- floor(0.80 * nrow(imbalanceddata))
set.seed(567)
train_ind <- sample(seq_len(nrow(imbalanceddata)), size = sample)
training <- imbalanceddata[train_ind, ]
test <- imbalanceddata[-train_ind, ]
```

Figure 174: Creating training and test datasets

Class Imbalance

1. Use four different methods to correct the class imbalance of the target (Vaccination Status) using the "ROSE" package

```
> under_trains <- ovun.sample(Vaccination_Status ~ .,
+ data = training, method = "under", N = 9700, seed = 1)$data
> table(under_trains$Vaccination_Status)

  Vaccinated Unvaccinated
        4671         5029
> both_trains <- ovun.sample(Vaccination_Status ~ .,
+ data = training, method = "both", p=0.5, N=11400, seed = 1)$data
> table(both_trains$Vaccination_Status)

  Vaccinated Unvaccinated
        5739         5661
> synth_trains <- ROSE(Vaccination_Status ~ .,
+ data = training, seed = 1)$data
> table(synth_trains$Vaccination_Status)

  Vaccinated Unvaccinated
       11666        11505
> over_trains <- ovun.sample(Vaccination_Status ~ .,
+ data = training, method = "over",N = 36200)$data
> table(over_trains$Vaccination_Status)

  Vaccinated Unvaccinated
       18142        18058
```

Figure 175: Correcting class imbalance using ROSE package

2. Generate random forest models with each of the class corrected datasets

```
> set.seed(322)
> under_model <- randomForest(Vaccination_Status~., data = under_trains)
> under_pred <- predict(under_model, test)
> suppressWarnings(under_cm <- confusionMatrix(test$Vaccination_Status, under_pred))
> set.seed(2323)
> both_model <- randomForest(Vaccination_Status~., data = both_trains)
> both_pred <- predict(both_model, test)
> suppressWarnings(both_cm <- confusionMatrix(test$Vaccination_Status, both_pred))
> set.seed(323)
> synth_model <- randomForest(Vaccination_Status~., data = synth_trains)
> synth_pred <- predict(synth_model, test)
> suppressWarnings(synth_cm <- confusionMatrix(test$Vaccination_Status, synth_pred))
> set.seed(3232)
> over_model <- randomForest(Vaccination_Status~., data = over_trains)
> over_pred <- predict(over_model, test)
> suppressWarnings(over_cm <- confusionMatrix(test$Vaccination_Status, over_pred))
```

Figure 176: Building Random Forest models to compare sampling methods

3. Generate barplots to compare the sampling methods

```
compare_samples <- rbind(over_cm$overall, synth_cm$overall,
both_cm$overall, under_cm$overall)
compare_samples <- as.data.frame(compare_samples)
compare_samples$Method <- c('Oversampled', 'Synthetic Data',
'Under/Oversampled', 'Undersampled')
Project_Theme = theme(
  axis.title.x = element_text(size = 19),
  axis.text.x = element_text(size = 19),
  axis.title.y = element_text(size = 20),
  plot.title = element_text(hjust = 0.5, size = 20, face = 'bold'))
p <- ggplot(compare_samples, aes(x= reorder(Method, -Accuracy),
weight=Accuracy, fill = Method))
p <- p + geom_bar() + ggtitle("Comparison of Accuracy of Sampling Methods") +
Project_Theme
p <- p + xlab("Sampling Method") + ylab("Accuracy")
p + scale_fill_discrete(name = "Sampling Method")
```

Figure 177: Building barplot to compare accuracy of sampling methods

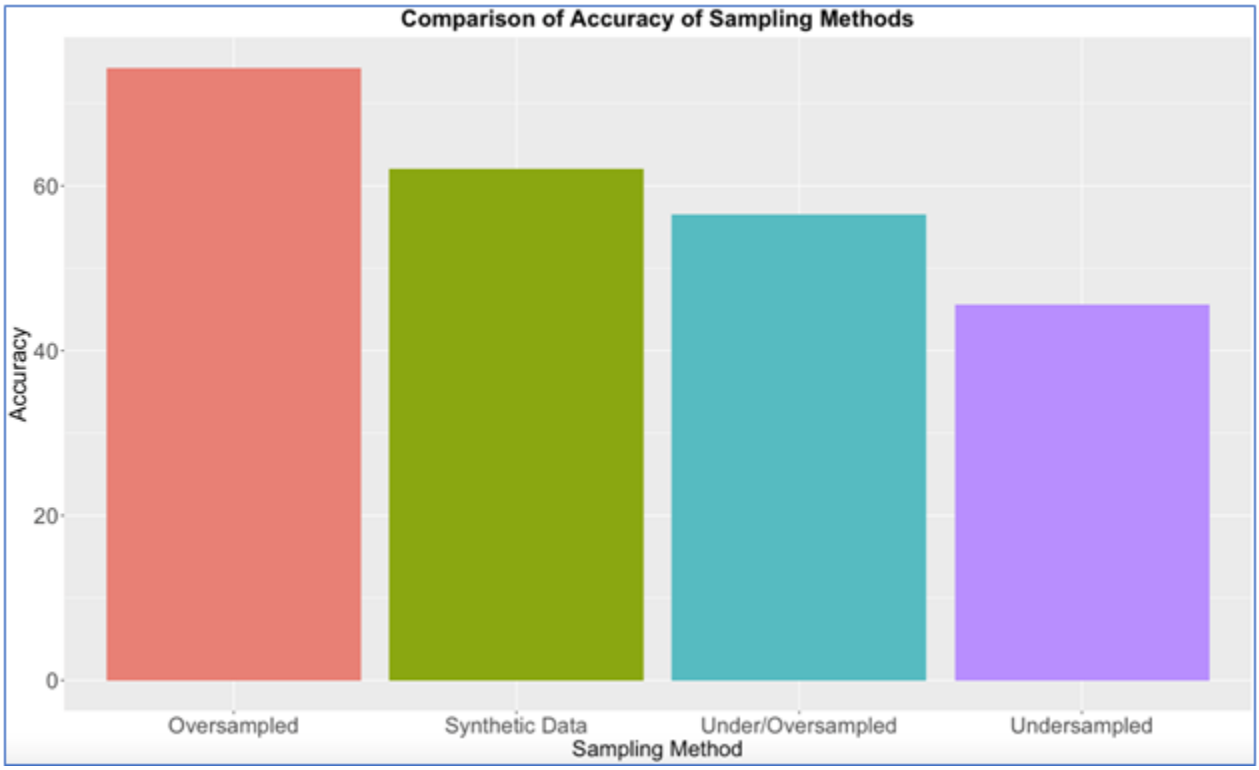

Figure 178: Accuracy of Sampling Methods

4. Create the final training dataset with the oversampled data

```
trains <- over_trains
```

Figure 179: Creating final training data

5. Generate a Random Forest model, obtain the variable importance and plot this using a 'ggplot2' barplot

```
set.seed(232)
imp_rf <- randomForest(Vaccination_Status ~ ., data=trains,  importance=TRUE)
var_imp <- varImpPlot(imp_rf)
var_imp <- as.data.frame(var_imp)
var_imp$varnames <- rownames(var_imp)
rownames(var_imp) <- NULL

ggplot(var_imp, aes(x=reorder(varnames, MeanDecreaseAccuracy),
                    weight=MeanDecreaseAccuracy, fill=varnames)) +
  geom_bar() + scale_fill_discrete(name="Variable Group") +
  ylab("Importance") + xlab("Variable") +
  coord_flip() + theme(legend.position = "none") +
  ggtitle("Random Forest Variable Importance") +
  theme(plot.title = element_text(hjust = 0.5))
```

Figure 180: Variable importance plot using Random Forest

Random Forest

Based on Khanh (2018)

1. Tune the Random Forrest mtry valaue (from http://math.furman.edu/~dcs/courses/math47/R/library/randomForest/html/tuneRF.html)

```
set.seed(3232)
tuning_rf <- tuneRF(trains[,-14], trains[,14], stepFactor = 2.5, plot =TRUE)
```

Figure 181: Tuning Random Forest

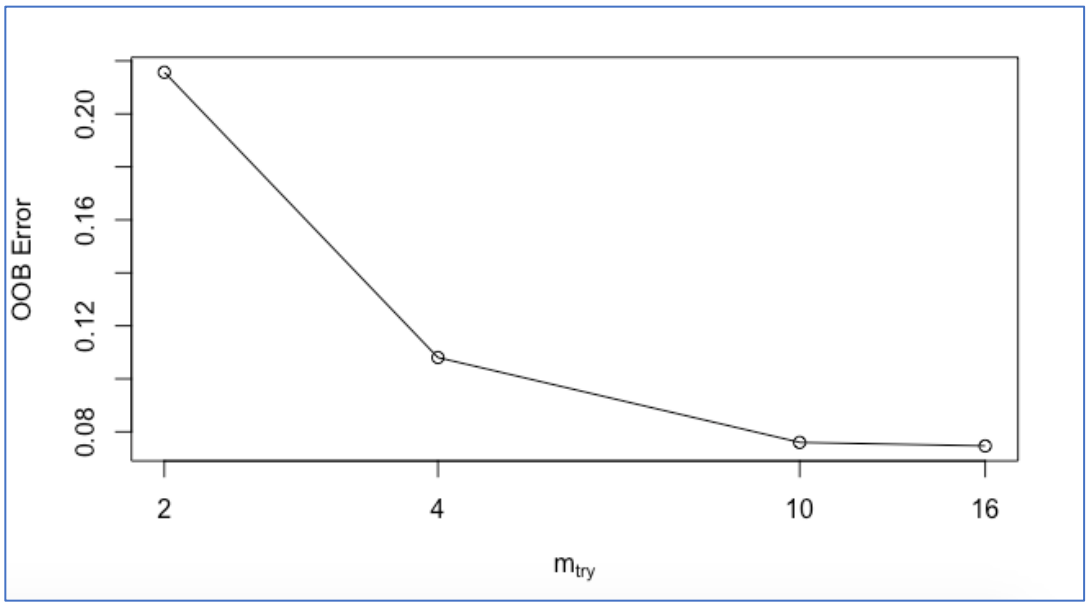

Figure 182: Plot of optimal mtry in Random Forest

2. Find the optimal number of trees for the Random Forest model

```
control <- trainControl(method = 'repeatedcv',
                        number = 10,
                        repeats = 3,
                        search = 'grid')
tunegrid <- expand.grid(.mtry=c(16))
modellist <- list()
for(ntree in c(100, 250, 500, 1000)) {set.seed(3233)
  fit <- caret::train(Vaccination_Status~., data = trained,
  method = "rf", tuneGrid = tunegrid, trControl = control, ntree = ntree)
key <- toString(ntree)
modellist[[key]] <- fit}
results <- resamples(modellist)
summary(results)
```

Figure 183: Finding optimal number of trees in Random Forest

```
> summary(results)

Call:
summary.resamples(object = results)

Models: 100, 250, 500, 1000
Number of resamples: 30

Accuracy
      Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
100  0.759 0.7811954 0.7860000 0.7873685 0.7930000 0.8168168    0
250  0.761 0.7808356 0.7875000 0.7880341 0.7970000 0.8128128    0
500  0.759 0.7795000 0.7873939 0.7882343 0.7949487 0.8191808    0
1000 0.761 0.7810856 0.7870000 0.7879682 0.7947500 0.8148148    0

Kappa
          Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
100  0.5181157 0.5624731 0.5720000 0.5747673 0.5860147 0.6336692    0
250  0.5220994 0.5617097 0.5750341 0.5760997 0.5940185 0.6256605    0
500  0.5181002 0.5590749 0.5748157 0.5764990 0.5899172 0.6383764    0
1000 0.5221070 0.5622158 0.5740290 0.5759678 0.5895590 0.6296641    0
```

Figure 184: Results of Random Forest tuning for number of trees

3. Build untuned and tuned models

```
set.seed(1234)
untuned_rf_model <- randomForest(Vaccination_Status ~., data = trains)
untuned_rf_pred <- predict(untuned_rf_model, test)
set.seed(323)
tuned_rf_model <- randomForest(Vaccination_Status ~., data = trains, ntree = 250, mtry = 16)
tuned_rf_pred <- predict(untuned_rf_model, test)
```

Figure 185: Building untuned and tuned Random Forest

4. Generate confusion matrices

```
> confusionMatrix(untuned_rf_pred,
+                 test$Vaccination_Status)
Confusion Matrix and Statistics

             Reference
Prediction     Unvaccinated Vaccinated
  Unvaccinated          140        387
  Vaccinated            645       2610

               Accuracy : 0.7271
                 95% CI : (0.7126, 0.7413)
    No Information Rate : 0.7924
    P-Value [Acc > NIR] : 1

                  Kappa : 0.056

 Mcnemar's Test P-Value : 1.244e-15

            Sensitivity : 0.17834
            Specificity : 0.87087
         Pos Pred Value : 0.26565
         Neg Pred Value : 0.80184
             Prevalence : 0.20756
         Detection Rate : 0.03702
   Detection Prevalence : 0.13934
      Balanced Accuracy : 0.52461

       'Positive' Class : Unvaccinated
```

```
> confusionMatrix(tuned_rf_pred,
+                 test$Vaccination_Status)
Confusion Matrix and Statistics

             Reference
Prediction     Unvaccinated Vaccinated
  Unvaccinated          141        388
  Vaccinated            644       2609

               Accuracy : 0.7271
                 95% CI : (0.7126, 0.7413)
    No Information Rate : 0.7924
    P-Value [Acc > NIR] : 1

                  Kappa : 0.057

 Mcnemar's Test P-Value : 2.058e-15

            Sensitivity : 0.17962
            Specificity : 0.87054
         Pos Pred Value : 0.26654
         Neg Pred Value : 0.80203
             Prevalence : 0.20756
         Detection Rate : 0.03728
   Detection Prevalence : 0.13987
      Balanced Accuracy : 0.52508

       'Positive' Class : Unvaccinated
```

Figure 186: Confusion matrices of untuned and tuned Random Forest

Bagging

1. Build the bagging models

```
set.seed(453)
untuned_bag <- bagging(formula = Vaccination_Status ~ ., data = trains)
untuned_bagging_pred <- predict(untuned_bag, test)
untuned_target <- as.factor(untuned_bagging_pred$class)
set.seed(432)
bagging_model <- bagging(formula = Vaccination_Status ~ ., data = trains, nbagg = 100)
bagging_pred <- predict(bagging_model, test)
bagging_target <- as.factor(bagging_pred$class)
```

Figure 187: Building the bagging models

2. Create a confusion matrix for the models

```
> confusionMatrix(untuned_target,
+                 test$Vaccination_Status)
Confusion Matrix and Statistics

               Reference
Prediction      Unvaccinated Vaccinated
  Unvaccinated           707       2110
  Vaccinated             492       2431

               Accuracy : 0.5467
                 95% CI : (0.5337, 0.5596)
    No Information Rate : 0.7911
    P-Value [Acc > NIR] : 1

                  Kappa : 0.0835

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.5897
            Specificity : 0.5353
         Pos Pred Value : 0.2510
         Neg Pred Value : 0.8317
             Prevalence : 0.2089
         Detection Rate : 0.1232
   Detection Prevalence : 0.4908
      Balanced Accuracy : 0.5625

       'Positive' Class : Unvaccinated
```

```
> confusionMatrix(tuned_bagging_target,
+                 test$Vaccination_Status)
Confusion Matrix and Statistics

               Reference
Prediction      Unvaccinated Vaccinated
  Unvaccinated           707       2110
  Vaccinated             492       2431

               Accuracy : 0.5467
                 95% CI : (0.5337, 0.5596)
    No Information Rate : 0.7911
    P-Value [Acc > NIR] : 1

                  Kappa : 0.0835

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.5897
            Specificity : 0.5353
         Pos Pred Value : 0.2510
         Neg Pred Value : 0.8317
             Prevalence : 0.2089
         Detection Rate : 0.1232
   Detection Prevalence : 0.4908
      Balanced Accuracy : 0.5625

       'Positive' Class : Unvaccinated
```

Figure 188: Confusion matrices of bagging models

Superlearner

The SuperLearner was Kennedy (2017)

1. Use 3-fold cross v

```
test <- na.omit(test)
y <- as.numeric(trains[,14])-1
ytest <- as.numeric(test[,14])-1
x <- trains[ ,c(1:13, 15:17)]
xtest <- test[ ,c(1:13, 15:17)]
set.seed(323)
CV_sl <- CV.SuperLearner(y, x, V =3,
            family=binomial(),
            SL.library=list("SL.ranger",
            "SL.xgboost", "SL.ipredbagg"))
plot(CV_sl)
```

Figure 189: Superlearner cross validation

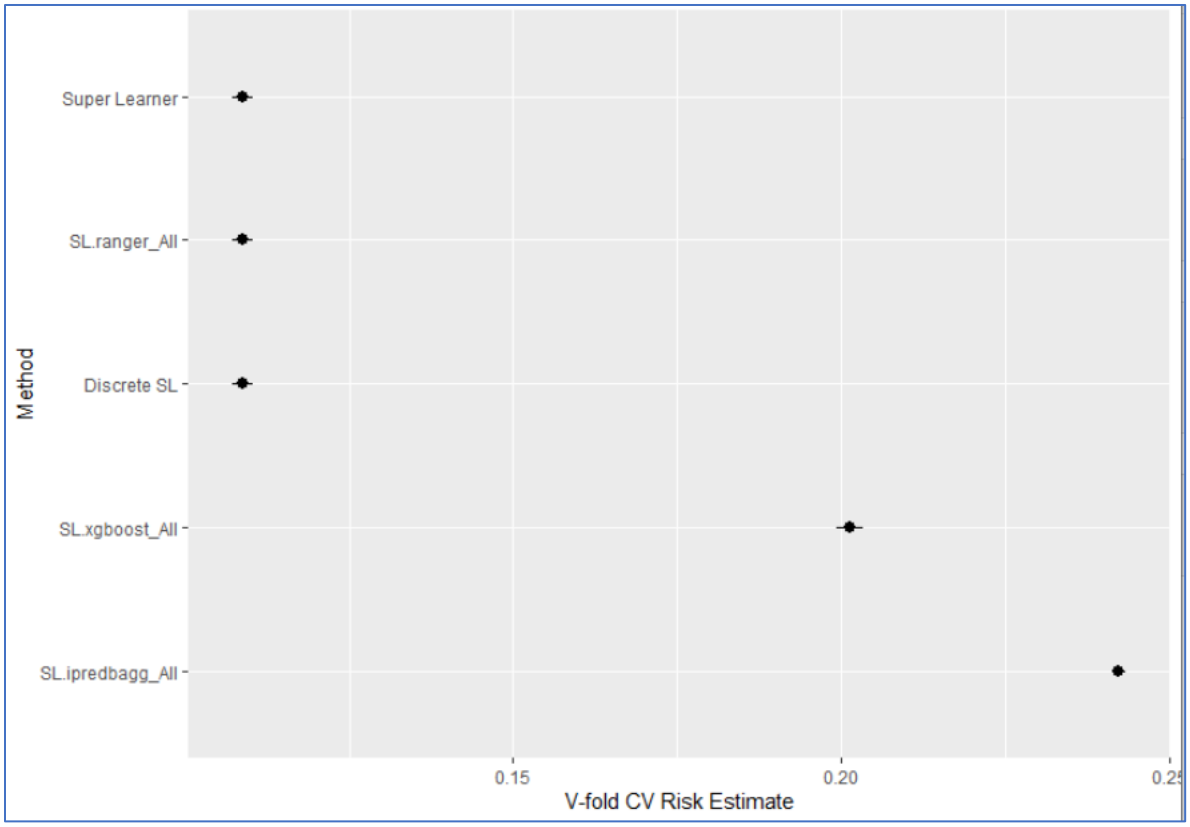

Figure 190: Plot of Superlearner cross validation

2. Tune the SuperLearner using Ranger (a faster version of Random Forest), ipredbag and XGBoost. These were chosen because of their light weight

```
set.seed(323)
three_sl <- SuperLearner(y,
                         x,
                         family=binomial(),
                         SL.library=list("SL.ranger",
                                         "SL.xgboost",
                                         "SL.ipredbagg"))
```

Figure 191: Building three model Superlearner

3. Generate predictions and a confusion matrix for the model

```
three__sl_pred <- predict.SuperLearner(three_sl, newdata=xtest)
conv.preds <- ifelse(three__sl_pred$pred>=0.5,0,1)
conv.pred <- ytest
conv.preds <- as.factor(conv.preds)
conv.pred <- as.factor(conv.pred)
three_sl_cm <- confusionMatrix(conv.pred, conv.preds)
three_sl_cm
```

Figure 192: Generating predictions and confusion matrix for three model Superlearner

```
> three_sl_cm
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0   83  702
         1  212 2785

               Accuracy : 0.7583
                 95% CI : (0.7444, 0.7719)
    No Information Rate : 0.922
    P-Value [Acc > NIR] : 1

                  Kappa : 0.0455

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.28136
            Specificity : 0.79868
         Pos Pred Value : 0.10573
         Neg Pred Value : 0.92926
             Prevalence : 0.07800
         Detection Rate : 0.02195
   Detection Prevalence : 0.20756
      Balanced Accuracy : 0.54002

       'Positive' Class : 0
```

Figure 193: Confusion matrix for first Superlearner

4. Build the second SuperLearner with just boosting and Ranger

```
set.seed(323)
two_sl <- SuperLearner(y,
                       x,
                       family=binomial(),
                       SL.library=list("SL.ranger",
                                       "SL.xgboost"))
```

Figure 194: Building three model Superlearner

5. Generate predictions and the confusion matrix

```
set.seed(323)
two_sl <- SuperLearner(y,
                       x,
                       family=binomial(),
                       SL.library=list("SL.ranger",
                                       "SL.xgboost"))
two_sl_pred <- predict.SuperLearner(two_sl, newdata=xtest)
conv.preds <- ifelse(two_sl_pred$pred>=0.5,0,1)
conv.pred <- ytest
conv.preds <- as.factor(conv.preds)
conv.pred <- as.factor(conv.pred)
two_sl_cm <- confusionMatrix(conv.pred, conv.preds)
two_sl_cm

# one model- ranger
set.seed(323)
single_sl <- SuperLearner(y,
                          x,
                          family=binomial(),
                          SL.library=list("SL.ranger"))
single_sl_pred <- predict.SuperLearner(single_sl, newdata=xtest)
conv.preds <- ifelse(single_sl_pred$pred>=0.5,0,1)
conv.pred <- ytest
conv.preds <- as.factor(conv.preds)
conv.pred <- as.factor(conv.pred)
single_sl_cm <- confusionMatrix(conv.pred, conv.preds)
single_sl_cm
```

Figure 195: Generating predictions and confusion matrix for double and single Superlearner

```
> two_sl_cm
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0   85  700
         1  202 2795

               Accuracy : 0.7615
                 95% CI : (0.7476, 0.775)
    No Information Rate : 0.9241
    P-Value [Acc > NIR] : 1

                  Kappa : 0.0534

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.29617
            Specificity : 0.79971
         Pos Pred Value : 0.10828
         Neg Pred Value : 0.93260
             Prevalence : 0.07589
         Detection Rate : 0.02247
   Detection Prevalence : 0.20756
      Balanced Accuracy : 0.54794

       'Positive' Class : 0
```

```
> single_sl_cm
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0   89  696
         1  207 2790

               Accuracy : 0.7612
                 95% CI : (0.7473, 0.7748)
    No Information Rate : 0.9217
    P-Value [Acc > NIR] : 1

                  Kappa : 0.0575

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.30068
            Specificity : 0.80034
         Pos Pred Value : 0.11338
         Neg Pred Value : 0.93093
             Prevalence : 0.07827
         Detection Rate : 0.02353
   Detection Prevalence : 0.20756
      Balanced Accuracy : 0.55051

       'Positive' Class : 0
```

Figure 196: Confusion matrix of two model SuperLearner

6. Create tuned ranger model, run the model with xgboost, predict the results and generate the confusion matrix

```
SL.tuning.ranger <- function(...){
  SL.ranger(..., num.trees=100, mtry=16)}
set.seed(323)
tuned_sl <- SuperLearner(y,
                         x,
                         family=binomial(),
                         SL.library=list("SL.tuning.ranger",
                                         "SL.xgboost"))
tuned_sl <- predict.SuperLearner(tuned_sl, newdata=xtest)
conv.preds <- ifelse(tuned_sl$pred>=0.5,0,1)
conv.pred <- ytest
conv.preds <- as.factor(conv.preds)
conv.pred <- as.factor(conv.pred)
tuned_sl_cm <- confusionMatrix(conv.pred, conv.preds)
```

Figure 197: Implementing tuned SuperLearner

```
> tuned_sl_cm
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0  115  670
         1  338 2659

               Accuracy : 0.7335
                 95% CI : (0.7191, 0.7475)
    No Information Rate : 0.8802
    P-Value [Acc > NIR] : 1

                  Kappa : 0.04

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.25386
            Specificity : 0.79874
         Pos Pred Value : 0.14650
         Neg Pred Value : 0.88722
             Prevalence : 0.11978
         Detection Rate : 0.03041
   Detection Prevalence : 0.20756
      Balanced Accuracy : 0.52630

       'Positive' Class : 0
```

Figure 198: Tuned SuperLearner confusion matrix

SVM

1. Train four SVM models with different kernels, generate a confusion matrix for each model and subset the accuracy measure from the confusion matrix

```
set.seed(1234)
svm_vanilla <- ksvm(Vaccination_Status ~ ., data = trains, kernel = "vanilladot")
vanilla_pred <- predict(svm_vanilla, test)
vanillaconfusion <- confusionMatrix(vanilla_pred, test$Vaccination_Status)
vanillaconfusion$overall
vanilla <- vanillaconfusion$overall
vanilla
vanilla$kernel <- 'vanilla'
vanilla <- data.frame(as.list(vanilla))

set.seed(2313)
svm_rbfdot <- ksvm(Vaccination_Status ~ ., data = trains, kernel = "rbfdot")
rbfdot_pred <- predict(svm_rbfdot, test)
rbfdotconfusion <- confusionMatrix(rbfdot_pred, test$Vaccination_Status)
rbfdotconfusion$overall
rbf <- rbfdotconfusion$overall
rbf
rbf$kernel <- 'rbf_dot'
rbf <- data.frame(as.list(rbf))

set.seed(1234)
svm_laplacedot <- ksvm(Vaccination_Status ~ ., data = trains, kernel = "laplacedot")
laplacedot_pred <- predict(svm_laplacedot, test)
laplacedot_confusion <- confusionMatrix(laplacedot_pred, test$Vaccination_Status)
laplacedot_confusion$overall
laplacedot <- laplacedot_confusion$overall
laplacedot
laplacedot$kernel <- 'laplace_dot'
laplacedot <- data.frame(as.list(laplacedot))

set.seed(1234)
svm_classifier <- ksvm(Vaccination_Status ~ ., data = trains, kernel = "besseldot")
besseldot_pred <- predict(svm_classifier, test)
besseldot_confusion <- confusionMatrix(besseldot_pred, test$Vaccination_Status)
besseldot_confusion$overall
besseldot <- besseldot_confusion$overall
besseldot
besseldot$kernel <- 'bessel_dot'
besseldot <- data.frame(as.list(besseldot))
svmkernelcombined <- rbind(rbf, vanilla, laplacedot, besseldot)
```

Figure 199: Training svm models with different kernels

2. Generate a barplot for the accuracy of each kernel

```
svmkernelcombined <- rbind(rbf, vanilla, laplacedot, besseldot)
ggplot(data=svmkernelcombined, aes(x=reorder(kernel, -Accuracy),
y=Accuracy, fill = kernel)) + geom_bar(stat="identity") +
coord_cartesian(ylim=c(0.5, 0.62)) + ggtitle("Accuracy of Each SVM Kernel") +
theme(plot.title = element_text(hjust = 0.5)) + xlab("Kernel")
```

Figure 200: Generating barplot comparing accuracy of each svm kernel

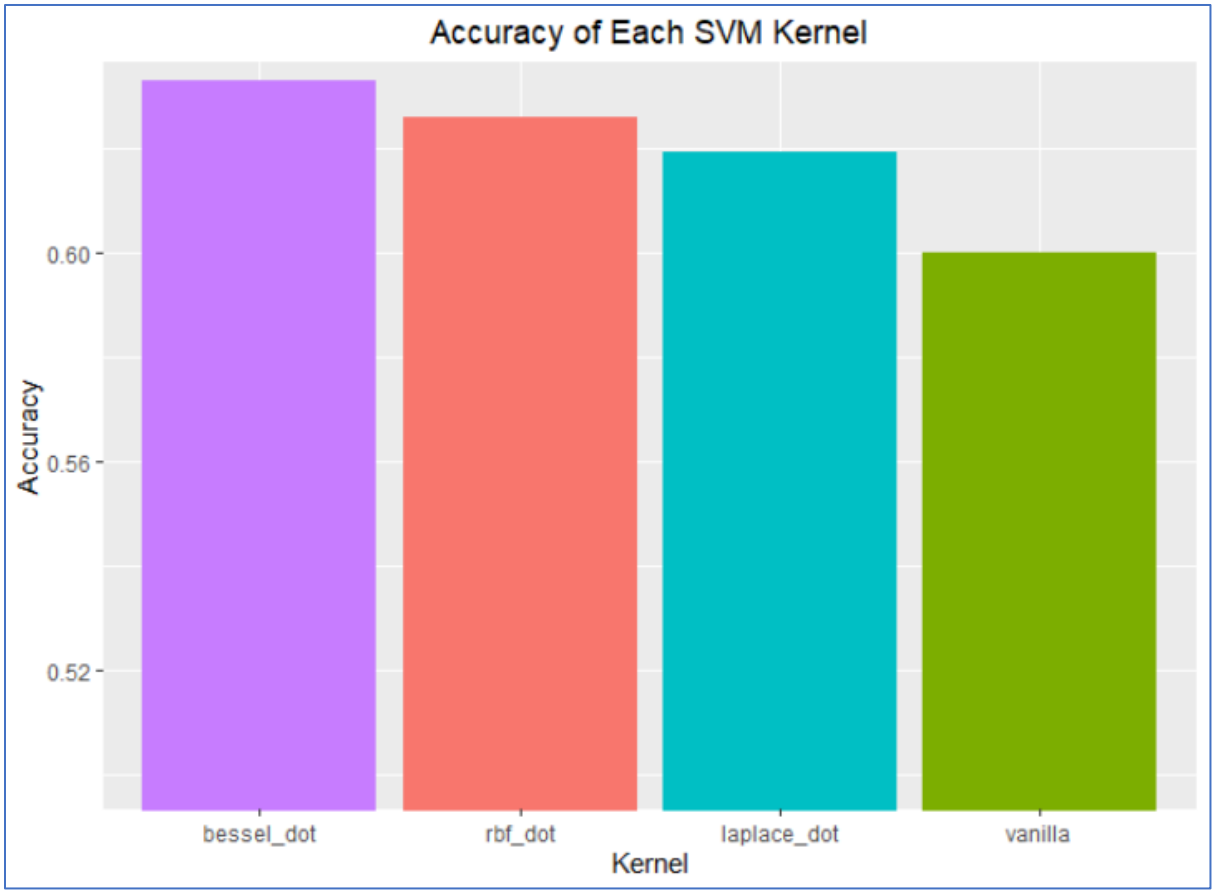

Figure 201: Barplot of kernel accuracy

3. Tune the cost parameter

```
set.seed(142)
svmtune_cost <- tune(svm, vaccination_status ~ ., data = trained,
kernel='linear', ranges=list(cost=c(0.0001, 0.001, 0.01, 0.1, 1, 10)))
plot(svmtune_cost, main = "Tuning SVM Cost")
```

Figure 203: Tuning the cost in SVM

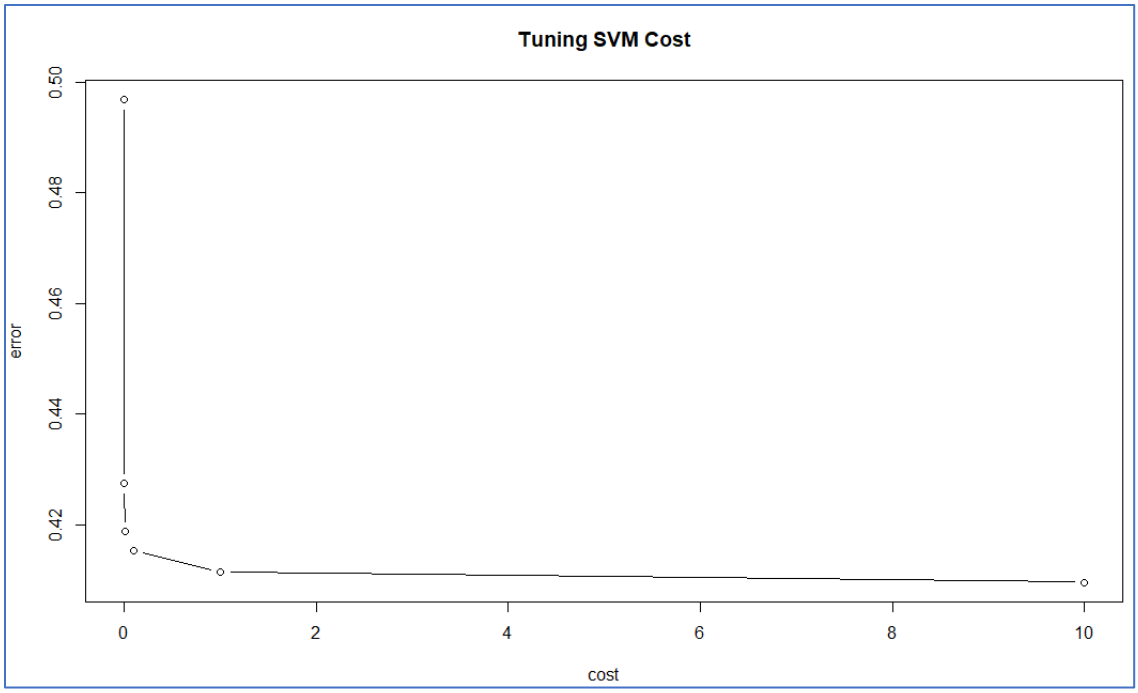

Figure 204: Plot of SVM cost tuning

4. Tune the sigma parameter

```
set.seed(1424)
svmtune <- tune(svm, Vaccination_Status ~ ., data = trained,
kernel='linear', ranges=list(sigma = c(0.001,0.003,0.006,0.009)))
plot(svmtune, main="Tuning SVM Model: Error vs Sigma")
```

Figure 205: Tuning sigma in SVM

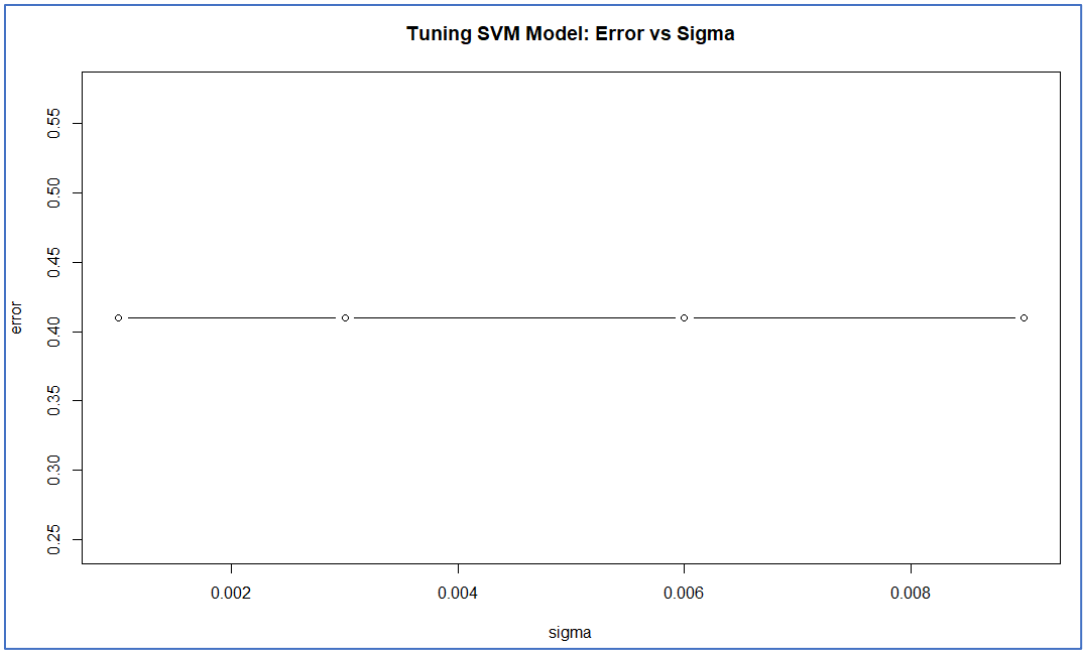

Figure 206: Plot of optimal sigma value for SVM

5. Build the untuned and tuned models

```
set.seed(434)
untuned_svm_model <- ksvm(Vaccination_Status ~., data = trains)
untuned_svm_pred <- predict(untuned_svm_model, test)
set.seed(343)
tuned_ksvm_model = ksvm(Vaccination_Status~., data=trains, kpar=list(sigma = .001),
C = 10, kernel = "besseldot")
tuned_svm_pred <- predict(tuned_ksvm_model, test)
```

Figure 207: Building untuned and tuned SVM

6. Generate the confusion matrices for the SVM models

```
> confusionMatrix(untuned_svm_pred,
+                 test$Vaccination_Status)
Confusion Matrix and Statistics

             Reference
Prediction     Unvaccinated Vaccinated
  Unvaccinated          366        996
  Vaccinated            419       2001

               Accuracy : 0.6259
                 95% CI : (0.6102, 0.6413)
    No Information Rate : 0.7924
    P-Value [Acc > NIR] : 1

                  Kappa : 0.1053

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.46624
            Specificity : 0.66767
         Pos Pred Value : 0.26872
         Neg Pred Value : 0.82686
             Prevalence : 0.20756
         Detection Rate : 0.09677
   Detection Prevalence : 0.36013
      Balanced Accuracy : 0.56695

       'Positive' Class : Unvaccinated
```

```
> confusionMatrix(tuned_svm_pred,
+       test$Vaccination_Status)
Confusion Matrix and Statistics

             Reference
Prediction     Unvaccinated Vaccinated
  Unvaccinated            0          0
  Vaccinated            785       2997

               Accuracy : 0.7924
                 95% CI : (0.7792, 0.8053)
    No Information Rate : 0.7924
    P-Value [Acc > NIR] : 0.5096

                  Kappa : 0

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.0000
            Specificity : 1.0000
         Pos Pred Value :    NaN
         Neg Pred Value : 0.7924
             Prevalence : 0.2076
         Detection Rate : 0.0000
   Detection Prevalence : 0.0000
      Balanced Accuracy : 0.5000

       'Positive' Class : Unvaccinated
```

Figure 208: Confusion matrices of untuned and tuned SVM

Naïve Bayes

1. Perform paarameter tuning on the Naïve Bayes

```
features <- setdiff(names(trained), "Vaccination_Status")
x <- trained[, features]
y <- trained$Vaccination_Status
train_control <- trainControl(method = "cv", number = 10)
search_grid <- expand.grid(usekernel = c(TRUE, FALSE),
                           fL = 0:5, adjust = seq(0, 5, by = 1))
set.seed(1234)
tuning_nb_model <- caret::train(
  x = x,
  y = y,
  method = "nb",   trControl = train_control,
  tuneGrid = search_grid,
  preProc = c("BoxCox", "center", "scale"))
plot(tuning_nb_model)
```

Figure 209: Tuning Naive Bayes

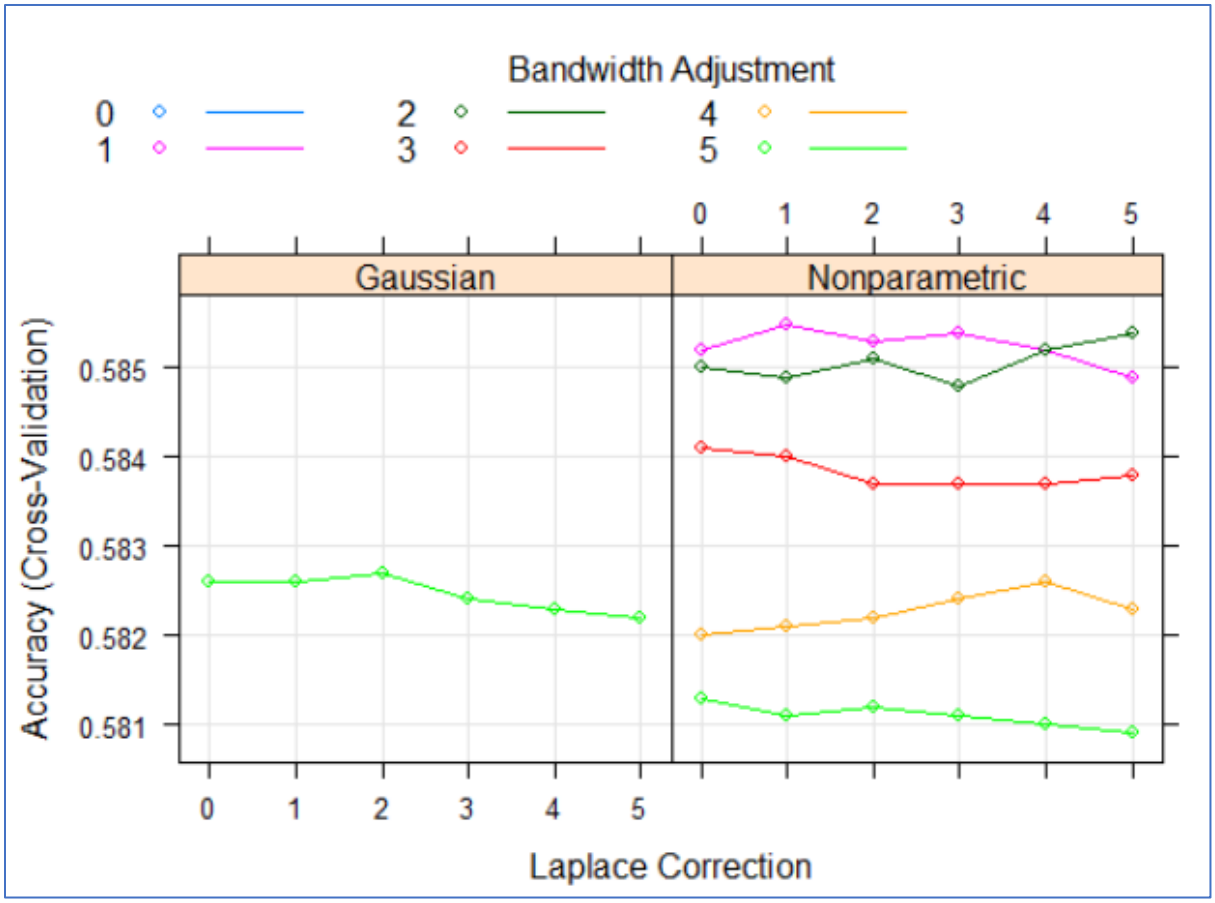

Figure 210: Plot of Naive Bayes model tuning

2. Build the tuned and untuned models

```
set.seed(543)
untuned_naive_bayes_model <- naiveBayes(Vaccination_Status ~ ., data = trains)
untuned_naive_bayes <- predict(untuned_naive_bayes_model, test)
search_grid <- expand.grid(usekernel = c(TRUE), fL = 1, adjust = 2)
set.seed(237)
tuned_nb_model <- caret::train(x = x, y = y, method = "nb",  tuneGrid = search_grid,
preProc = c("BoxCox", "center", "scale", "pca"))
tuned_naive_bayes <- predict(tuned_nb_model, test, type = "raw")
```

Figure 211: Building Naive Bayes models

3. Generate confusion matrices for the untuned and tuned models

```
> confusionMatrix(untuned_naive_bayes,
+                 test$Vaccination_Status)
Confusion Matrix and Statistics

             Reference
Prediction     Unvaccinated Vaccinated
  Unvaccinated          434       1182
  Vaccinated            351       1815

               Accuracy : 0.5947
                 95% CI : (0.5788, 0.6104)
    No Information Rate : 0.7924
    P-Value [Acc > NIR] : 1

                  Kappa : 0.114

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.5529
            Specificity : 0.6056
         Pos Pred Value : 0.2686
         Neg Pred Value : 0.8380
             Prevalence : 0.2076
         Detection Rate : 0.1148
   Detection Prevalence : 0.4273
      Balanced Accuracy : 0.5792

       'Positive' Class : Unvaccinated
```

```
> confusionMatrix(tuned_naive_bayes,
+                 test$Vaccination_Status)
Confusion Matrix and Statistics

             Reference
Prediction     Unvaccinated Vaccinated
  Unvaccinated          441       1208
  Vaccinated            344       1789

               Accuracy : 0.5896
                 95% CI : (0.5738, 0.6054)
    No Information Rate : 0.7924
    P-Value [Acc > NIR] : 1

                  Kappa : 0.1129

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.5618
            Specificity : 0.5969
         Pos Pred Value : 0.2674
         Neg Pred Value : 0.8387
             Prevalence : 0.2076
         Detection Rate : 0.1166
   Detection Prevalence : 0.4360
      Balanced Accuracy : 0.5794

       'Positive' Class : Unvaccinated
```

Figure 212: Confusion matrices for untuned and tuned Naive Bayes

C5.0

1. Tune the trials parameter for C5.0

```
fitControl <- trainControl(method = "repeatedcv",
     number = 5, repeats = 3, returnResamp="all")
grid <- expand.grid( .winnow = c(TRUE), .trials=c(1, 25, 50, 75, 100), .model="tree" )
set.seed(1231)
C50_tune <- caret::train(x = trained[-14],
                         y = trained[,14],
                         tuneGrid=grid,
                         trControl=fitControl,
                         method="C5.0",VERBOSE=FALSE)
plot(C50_tune, main = "Plot of Optimal Trials for C5.0",
     xlab = "Trials", ylab = "accuracy")
```

Figure 213: C5.0 parameter tuning for number of trials

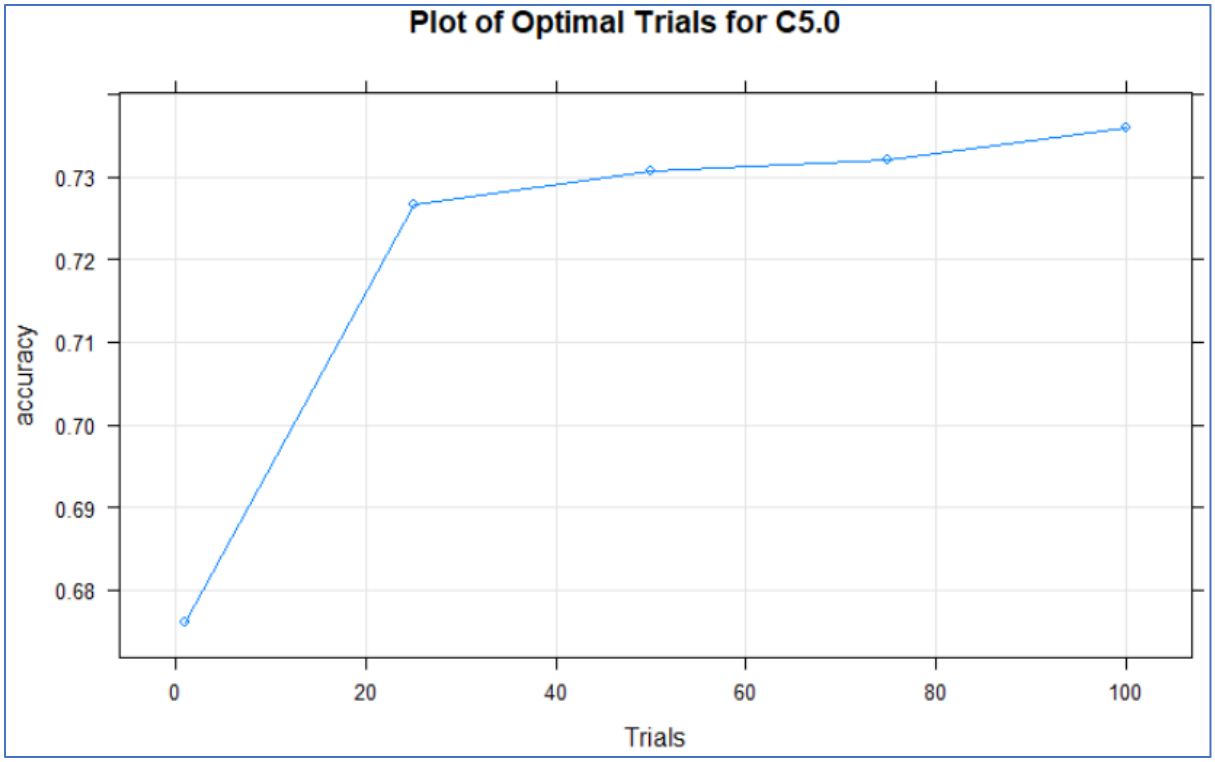

Figure 214: Plot of optimal trials for C5.0

2. Find the optimal winnow value for C5.0

```
grid <- expand.grid( .winnow = c(TRUE,FALSE), .trials=c(100), .model="tree" )
set.seed(1231)
C50_tune_winnow <- caret::train(x=trained[-14] ,
                                y=trained[,14],
                                tuneGrid=grid,
                                trControl=fitControl,
                                method="C5.0",VERBOSE=FALSE)
winnowplot <- C50_tune_winnow$results
p <- ggplot(data=winnowplot, aes(x = winnow, y = Accuracy, fill = winnow)) +
  geom_bar(stat="identity") +
  coord_cartesian(ylim=c(0.67, 0.685)) +
  ggtitle("Accuracy of C50 Model Based on Winnow Status") +
  theme(plot.title = element_text(hjust = 0.5)) +
  xlab("Winnow")
p
```

Figure 215: Finding the optimal value for winnow in C5.0

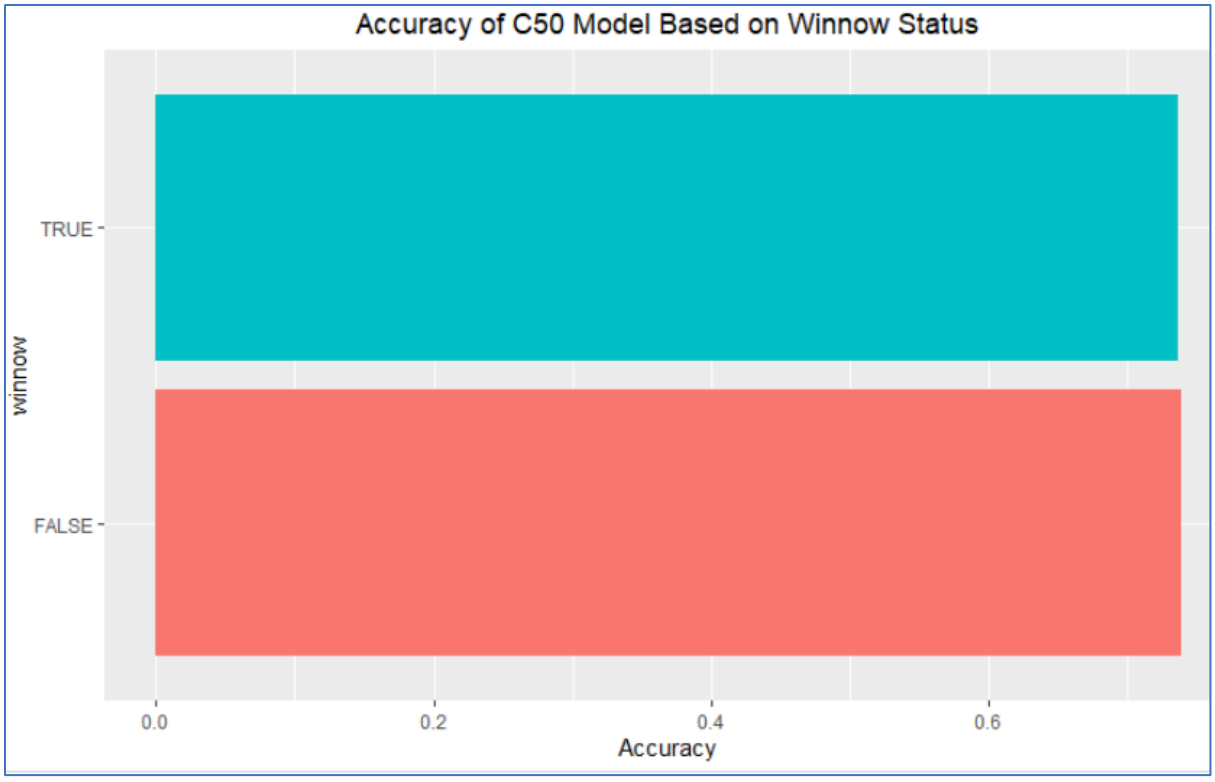

Figure 216: Finding optimal winnow value for C5.0

3. Build the untuned and tuned C5.0 models

```
set.seed(1244)
C50_default <- C5.0(trains[-14], trains$Vaccination_Status)
untuned_C50_pred <- predict(C50_default, test)
grid <- expand.grid( .winnow = c(FALSE), .trials=c(100), .model="tree" )
set.seed(1231)
C50_tuned <- caret::train(x=trains[-14] ,
                          y=trains[,14],
                          tuneGrid=grid,
                          method="C5.0",VERBOSE=FALSE)
tuned_C50_pred <- predict(C50_tune_winnow, test)
```

Figure 217: Building untuned and tuned C5.0 models

4. Generate the confusion matrices for the C5.0 models

```
> confusionMatrix(untuned_C50_pred,
+                 test$Vaccination_Status)
Confusion Matrix and Statistics

             Reference
Prediction     Unvaccinated Vaccinated
  Unvaccinated          370       1195
  Vaccinated            829       3346

               Accuracy : 0.6474
                 95% CI : (0.6349, 0.6598)
    No Information Rate : 0.7911
    P-Value [Acc > NIR] : 1

                  Kappa : 0.0408

 Mcnemar's Test P-Value : 4.934e-16

            Sensitivity : 0.30859
            Specificity : 0.73684
         Pos Pred Value : 0.23642
         Neg Pred Value : 0.80144
             Prevalence : 0.20889
         Detection Rate : 0.06446
   Detection Prevalence : 0.27265
      Balanced Accuracy : 0.52272

       'Positive' Class : Unvaccinated
```

```
> confusionMatrix(tuned_C50_pred,
+                 test$Vaccination_Status)
Confusion Matrix and Statistics

             Reference
Prediction     Unvaccinated Vaccinated
  Unvaccinated          203        572
  Vaccinated            996       3969

               Accuracy : 0.7268
                 95% CI : (0.7151, 0.7383)
    No Information Rate : 0.7911
    P-Value [Acc > NIR] : 1

                  Kappa : 0.0498

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.16931
            Specificity : 0.87404
         Pos Pred Value : 0.26194
         Neg Pred Value : 0.79940
             Prevalence : 0.20889
         Detection Rate : 0.03537
   Detection Prevalence : 0.13502
      Balanced Accuracy : 0.52167

       'Positive' Class : Unvaccinated
```

Figure 218: Confusion matrices for untuned and tuned C5.0 models

GBM

1. Tune the number of trees (boosting iterations) and interaction depth (max depth) of the GBM model

```
gbmGrid <- expand.grid(interaction.depth = seq(1,10, by = 2),
           n.trees = c(25, 50, 100, 200), shrinkage = c(0, 1),
           n.minobsinnode = 10)
boostctrl <- trainControl(method = "repeatedcv", number = 10,
           repeats = 1, classProbs = FALSE, savePredictions = T)
set.seed(323)
tuning_gbm <- caret::train(Vaccination_Status ~ .,
           data = trained, method = "gbm", trControl = boostctrl,
           verbose = FALSE, tuneGrid = gbmGrid)
plot(tuning_gbm, main = "Tuning GBM")
```

Figure 219: Tuning GBM

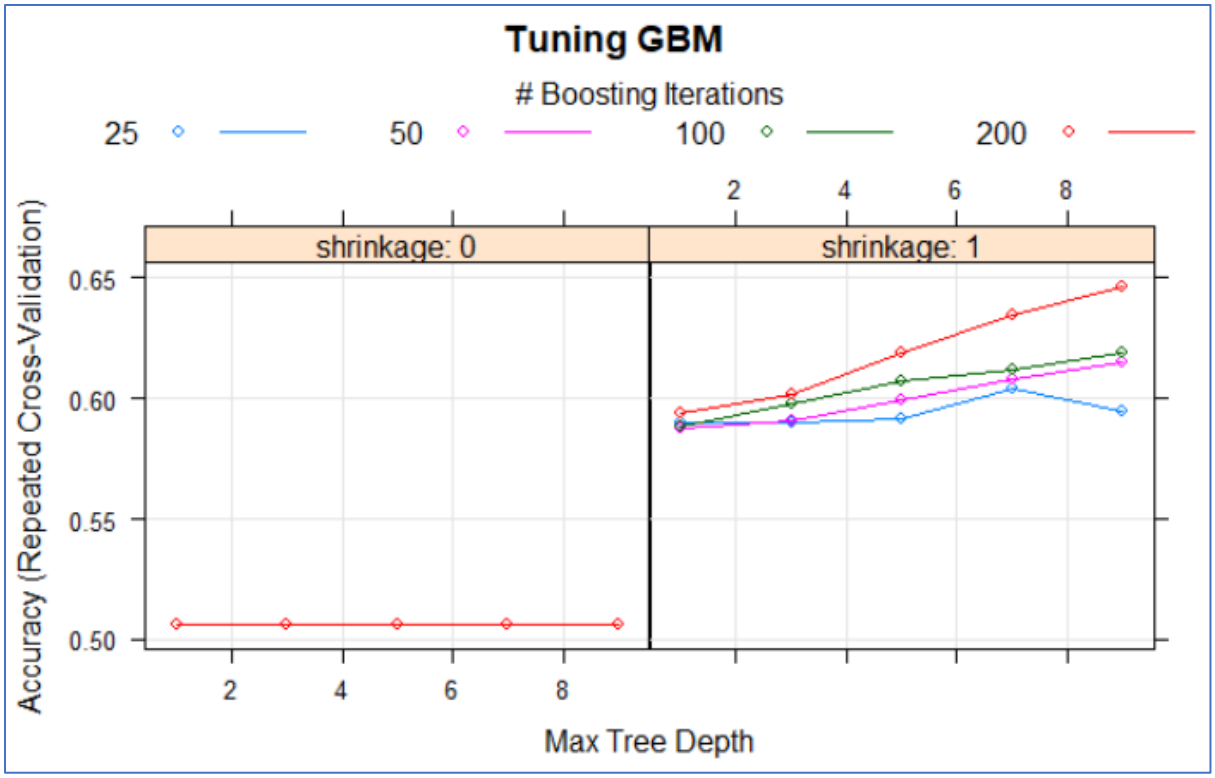

Figure 220: Finding optimal parameters for GBM

2. Build the untuned and tuned models

```
training <- trains
training$Vaccination_Status <- as.numeric(training$Vaccination_Status)-1
set.seed(232)
untuned_gbm <- gbm(Vaccination_Status ~ ., data = training, n.trees = 100)
untuned_gbm_pred <- predict(untuned_gbm, test, n.trees =  100)
untuned_gbm_pred <- ifelse(untuned_gbm_pred >= 0.5, 0, 1)
target <- as.numeric(test[,14])-1
untuned_gbm_pred <- as.factor(target)

set.seed(3233)
tuned_gbm <- gbm(Vaccination_Status ~ ., data = training,
                 n.trees = 200, interaction.depth = 20)
tuned__gbm_pred <- predict(tuned_gbm, test, n.trees = 200)
tuned__gbm_pred <- ifelse(tuned__gbm_pred >= 0, 0, 1)
tuned__gbm_pred <- as.factor(tuned__gbm_pred)

testing <- test
testing$Vaccination_Status <- as.numeric(testing$Vaccination_Status)-1
testing$Vaccination_Status <- as.factor(testing$Vaccination_Status)
```

Figure 221: Building and predicting untuned and tuned C-Forest

3. Generate confusion matrices for the models

```
> confusionMatrix(untuned_gbm_pred,
+                 testing$Vaccination_Status)
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0 1199    0
         1    0 4541

               Accuracy : 1
                 95% CI : (0.9994, 1)
    No Information Rate : 0.7911
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 1

 Mcnemar's Test P-Value : NA

            Sensitivity : 1.0000
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 1.0000
             Prevalence : 0.2089
         Detection Rate : 0.2089
   Detection Prevalence : 0.2089
      Balanced Accuracy : 1.0000

       'Positive' Class : 0
```

```
> confusionMatrix(tuned__gbm_pred,
+                 testing$Vaccination_Status)
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0  490 1326
         1  709 3215

               Accuracy : 0.6455
                 95% CI : (0.6329, 0.6579)
    No Information Rate : 0.7911
    P-Value [Acc > NIR] : 1

                  Kappa : 0.0981

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.40867
            Specificity : 0.70799
         Pos Pred Value : 0.26982
         Neg Pred Value : 0.81932
             Prevalence : 0.20889
         Detection Rate : 0.08537
   Detection Prevalence : 0.31638
      Balanced Accuracy : 0.55833

       'Positive' Class : 0
```

Figure 222: Tuned and untuned confusion matrices for gbm

C-Forest

1. Tune the mincriterion and max depth of the C-Forest

```
ctree_control <- trainControl(method = 'cv', number=5,summaryFunction=defaultSummary)
ctree_tune_grid <- expand.grid(mincriterion=c(0.010, 0.255, 0.50, 0.745, 0.990),
                    maxdepth = seq(15, 50, 5))
ctree_formula <- Vaccination_Status ~ Education_Status + Income_Group +
  Number_Providers + Household_Size + Child_Number + Insurance_Type +
  Provider_Facility + Marital_Status + Duration + Race + House_Ownership_Status +
  Mother_Age_Group + Firstborn
set.seed(1234)
ctree_tune <- caret::train(ctree_formula, data=trained, method = 'ctree2',
                    trControl = ctree_control,tuneGrid = ctree_tune_grid)
plot(ctree_tune, main = "Tuning C-Forest", ylab = "Accuracy")
```

Figure 223: Tuning the C-Forest

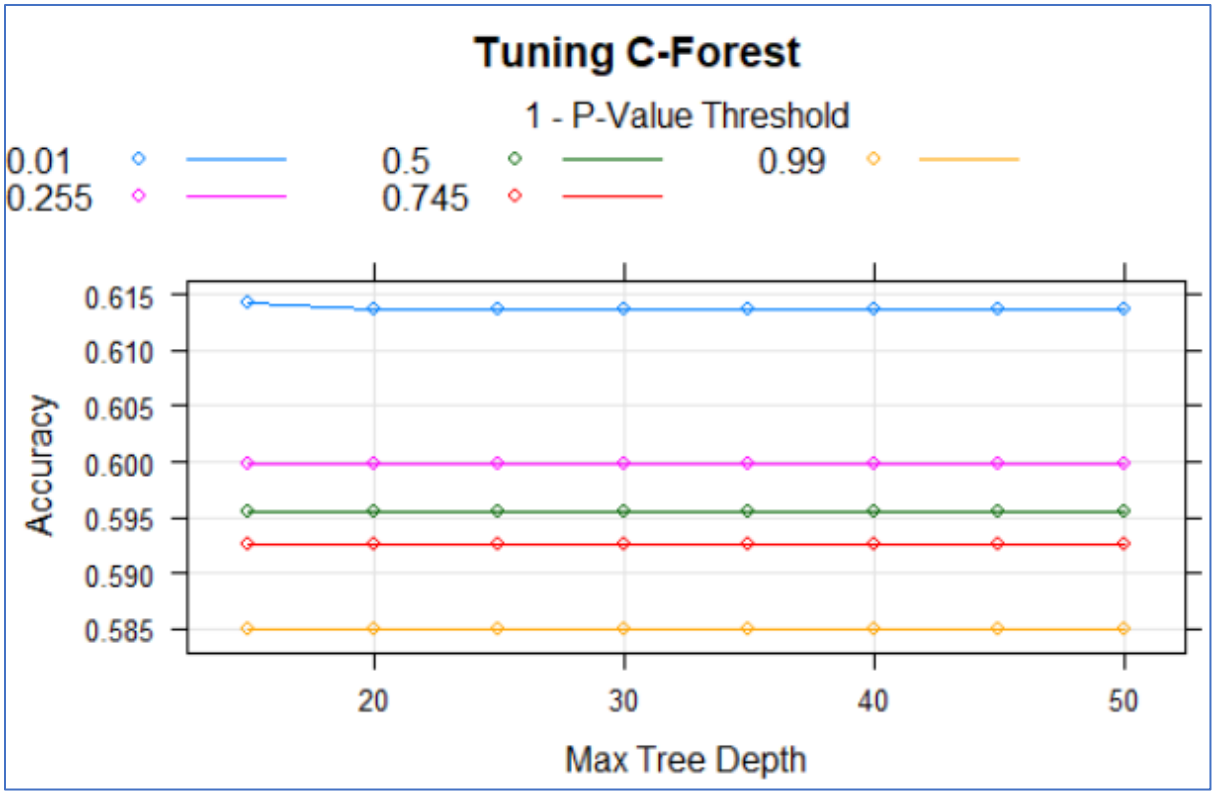

Figure 234: Plot of tuning parameters for C-Forest

2. Build the untuned and tuned models

```
set.seed(343)
untuned_cforest <- cforest(ctree_formula, data = trains)
untuned_ctree_pred <- predict(untuned_cforest, test)

Grid <- expand.grid(mincriterion=c(0.01),
                    maxdepth = c(15))
set.seed(1234)
ctree_tune <- caret::train(ctree_formula, data=trains, method = 'ctree2',
                           trControl=fitControl,tuneGrid=Grid)
tuned_ctree_pred <- predict(ctree_tune, testing)
```

Figure 235: Building and predicting untuned and tuned C-Forest

```
> confusionMatrix(untuned_ctree_pred,
+                 test$Vaccination_Status)
Confusion Matrix and Statistics

              Reference
Prediction     Unvaccinated Vaccinated
  Unvaccinated          301        718
  Vaccinated            484       2279

               Accuracy : 0.6822
                 95% CI : (0.6671, 0.697)
    No Information Rate : 0.7924
    P-Value [Acc > NIR] : 1

                  Kappa : 0.1296

 Mcnemar's Test P-Value : 1.811e-11

            Sensitivity : 0.38344
            Specificity : 0.76043
         Pos Pred Value : 0.29539
         Neg Pred Value : 0.82483
             Prevalence : 0.20756
         Detection Rate : 0.07959
   Detection Prevalence : 0.26943
      Balanced Accuracy : 0.57193

       'Positive' Class : Unvaccinated
```

```
> confusionMatrix(tuned_ctree_pred,
+                 test$Vaccination_Status)
Confusion Matrix and Statistics

              Reference
Prediction     Unvaccinated Vaccinated
  Unvaccinated          295        834
  Vaccinated            490       2163

               Accuracy : 0.6499
                 95% CI : (0.6345, 0.6651)
    No Information Rate : 0.7924
    P-Value [Acc > NIR] : 1

                  Kappa : 0.0839

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.3758
            Specificity : 0.7217
         Pos Pred Value : 0.2613
         Neg Pred Value : 0.8153
             Prevalence : 0.2076
         Detection Rate : 0.0780
   Detection Prevalence : 0.2985
      Balanced Accuracy : 0.5488

       'Positive' Class : Unvaccinated
```

Figure 236: Tuned and untuned confusion matrices for C-Forest

Neural Network

1. Tune the size of the neural network (numbers higher than 10 produced an error)

```
set.seed(965)
tune_nnet <- tune.nnet(Vaccination_Status ~., data=trained, size = 1:10)
plot(tune_nnet, main = "Tuning Neural Network")
```

Figure 267: Tuning size of neural network

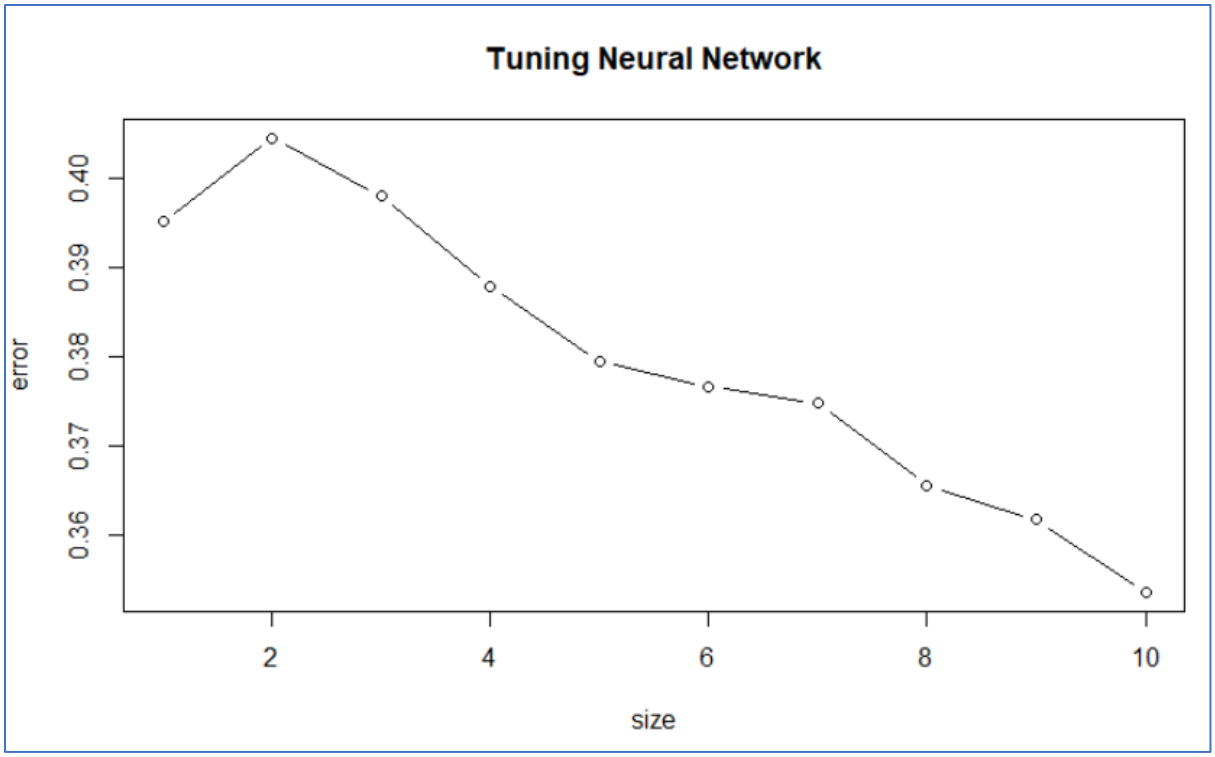

Figure 238: Plot of optimal size for neural network

2. Tune the decay of the neural network based on https://stackoverflow.com/questions/42417948/how-to-use-size-and-decay-in-nnet

```
fitControl <- trainControl(method = "repeatedcv",  number = 10,
repeats = 1, classProbs = TRUE, summaryFunction = twoClassSummary)
nnetGrid <-  expand.grid(size = 2,
decay = c(0.1, 0.5, 0.9, 1.3))
set.seed(545)
nnetFit <- caret::train(Vaccination_Status ~ ., data = trained, method = "nnet",
metric = "ROC", trControl = fitControl, tuneGrid = nnetGrid, verbose = FALSE)
plot(nnetFit, main = "Neural Network Accuracy By Weight Decay")
```

Figure 239: Tuning weight decay for Neural Network

```
> nnetFit
Neural Network

10000 samples
   16 predictor
    2 classes: 'Vaccinated', 'Unvaccinated'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 1 times)
Summary of sample sizes: 9000, 9000, 9000, 9000, 9000, 9000, ...
Resampling results across tuning parameters:

  decay  ROC        Sens       Spec
  0.1    0.6770363  0.6281377  0.6371542
  0.5    0.6679646  0.6263158  0.6233202
  0.9    0.6621798  0.6194332  0.6233202
  1.3    0.6672409  0.6218623  0.6171937

Tuning parameter 'size' was held constant at a value of 10
ROC was used to select the optimal model using the largest value.
The final values used for the model were size = 10 and decay = 0.1.
```

Figure 240: Performance of neural net at different decays

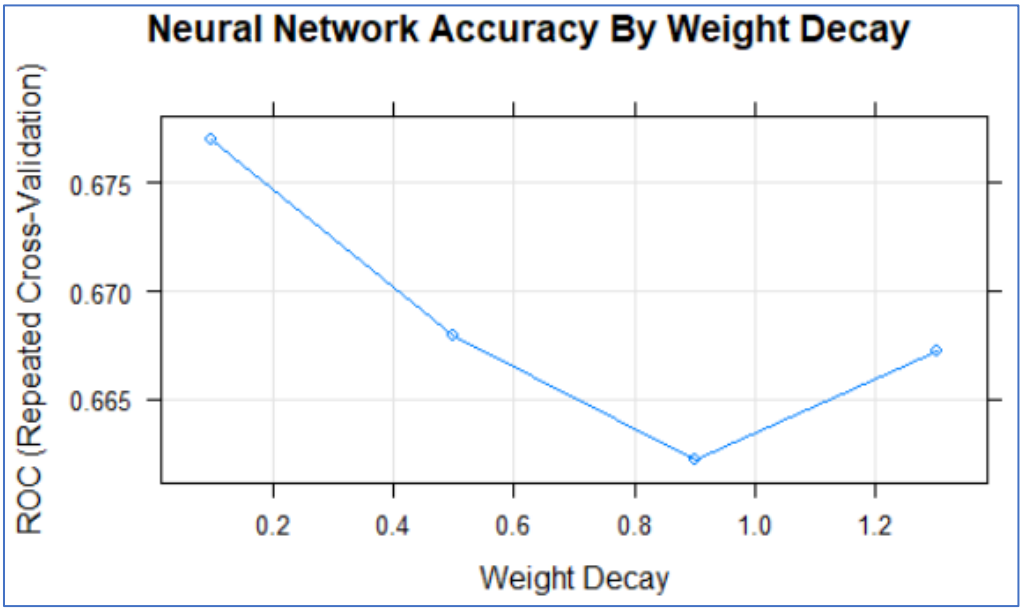

Figure 241: Optimal weight decay for neural network

3. Build the untuned and tuned models

```
set.seed(232)
untuned_nnet <- nnet(Vaccination_Status ~., data=trains, size = 10)
untuned_nnet_pred <- predict(untuned_nnet, test)
untuned_nnet_pred <- ifelse(untuned_nnet_pred >= 0.5,0,1)
untuned_nnet_pred <- as.factor(untuned_nnet_pred)
set.seed(434)
tuned_nnet <- nnet(Vaccination_Status ~., data=trains, size = 10, decay = 0.1)
tuned_nnet_pred <- predict(tuned_nnet, test)
tuned_nnet_pred <- ifelse(tuned_nnet_pred >= 0.5,0,1)
tuned_nnet_pred <- as.factor(tuned_nnet_pred)
```

Figure 242: Building untuned and tuned neural networks

4. Generate a confusion matrix for the untuned and tuned models

```
confusiobMatrix(untuned_nnet_pred, nnet_test)
confusionMatrix(tuned_nnet_pred, nnet_test)
```

Figure 243: Generate confusion matrices

```
> confusionMatrix(untuned_nnet_pred,
+                 nnet_test)
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0  338  914
         1  447 2083

               Accuracy : 0.6401
                 95% CI : (0.6246, 0.6555)
    No Information Rate : 0.7924
    P-Value [Acc > NIR] : 1

                  Kappa : 0.103

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.43057
            Specificity : 0.69503
         Pos Pred Value : 0.26997
         Neg Pred Value : 0.82332
             Prevalence : 0.20756
         Detection Rate : 0.08937
   Detection Prevalence : 0.33104
      Balanced Accuracy : 0.56280

       'Positive' Class : 0
```

```
> confusionMatrix(tuned_nnet_pred,
+                 nnet_test)
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0  343 1015
         1  442 1982

               Accuracy : 0.6148
                 95% CI : (0.599, 0.6303)
    No Information Rate : 0.7924
    P-Value [Acc > NIR] : 1

                  Kappa : 0.0774

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.43694
            Specificity : 0.66133
         Pos Pred Value : 0.25258
         Neg Pred Value : 0.81766
             Prevalence : 0.20756
         Detection Rate : 0.09069
   Detection Prevalence : 0.35907
      Balanced Accuracy : 0.54914

       'Positive' Class : 0
```

Figure 244: Confusion matrices of untuned and tuned Neural Networks

KNN

1. Tune the KNN model to find the optimal parameters using 1o-fold cross validation

```
selected <-c("Was_Child_Breastfed", "Household_Size", "Child_Number", "Firstborn",
             "Insurance_Type", "Education_Status", "Was_Child_Breastfed",
             "Race", "Mother_Age_Group", "Marital_Status", "House_Ownership_Status",
             "Provider_Facility", "Number_Providers", "WIC", "Region")
for (i in selected){trains[,i] <- as.numeric(trains[,i])}
for (i in selected){trains[,i] <- as.factor(trains[,i])}
for (i in selected){test[,i] <- as.numeric(test[,i])}
for (i in selected){test[,i] <- as.factor(test[,i])}

trained <- na.omit(trained)
test <- na.omit(test)
# From https://rpubs.com/njvijay/16251
set.seed(232)
knn.cross <- tune.knn(x = trained[,-14], y = trained[,14], k = 1:25,
                      tunecontrol = tune.control(sampling = "cross"), cross=10)
plot(knn.cross, main = "Optimal Value for K in KNN")
```

Figure 245: Tuning the KNN model

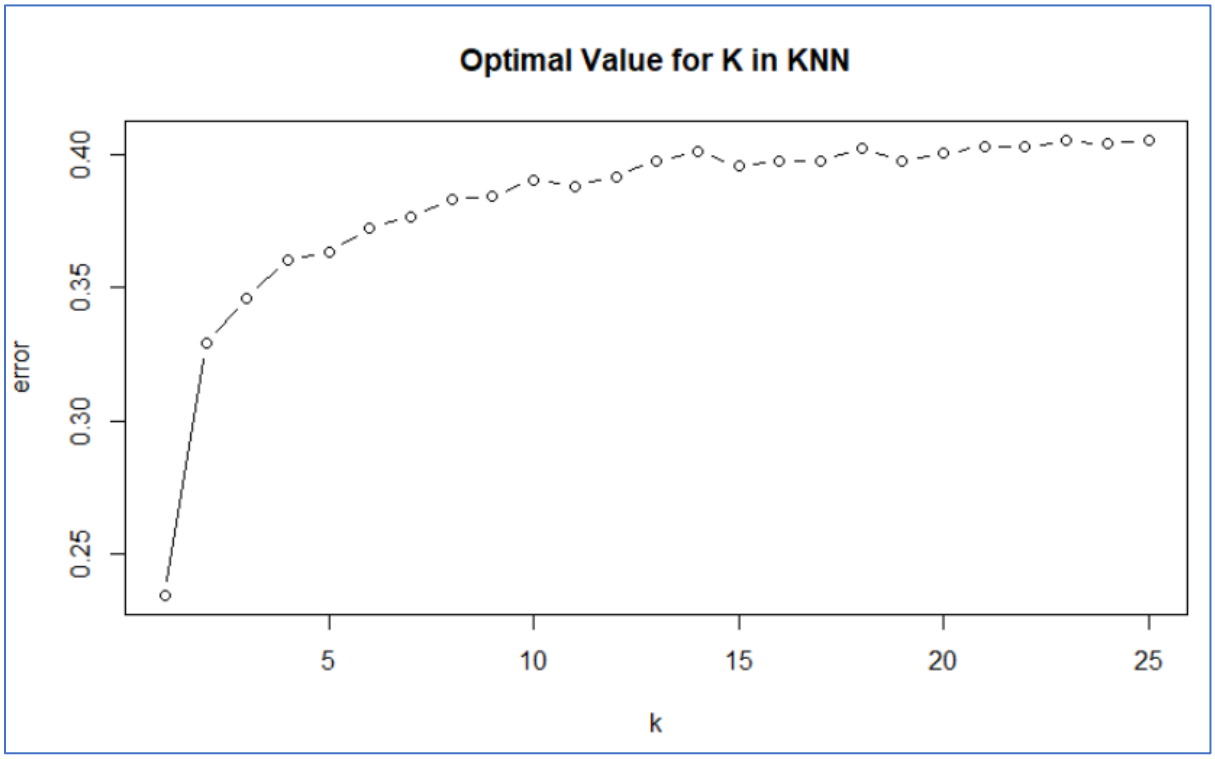

Figure 246: Tuning k value for KNN

2. Build the untuned and tuned KNN models

```
trains <- na.omit(trains)
set.seed(324)
knn_untuned <- kNN(Vaccination_Status ~ .,trains, test, norm=FALSE)
set.seed(232)
knn_tuned <- kNN(Vaccination_Status ~ .,trains, test, norm=FALSE, k = 1)
```

Figure 247: Building tuned and untuned KNN models

3. Create a confusion matrix for the untuned and tuned models

```
> confusionMatrix(knn_untuned,
+                 test$Vaccination_Status)
Confusion Matrix and Statistics

              Reference
Prediction     Unvaccinated Vaccinated
  Unvaccinated          206        609
  Vaccinated            579       2388

               Accuracy : 0.6859
                 95% CI : (0.6708, 0.7007)
    No Information Rate : 0.7924
    P-Value [Acc > NIR] : 1.0000

                  Kappa : 0.0584

 Mcnemar's Test P-Value : 0.4001

            Sensitivity : 0.26242
            Specificity : 0.79680
         Pos Pred Value : 0.25276
         Neg Pred Value : 0.80485
             Prevalence : 0.20756
         Detection Rate : 0.05447
   Detection Prevalence : 0.21549
      Balanced Accuracy : 0.52961

       'Positive' Class : Unvaccinated
```

```
> confusionMatrix(knn_tuned,
+                 test$Vaccination_Status)
Confusion Matrix and Statistics

              Reference
Prediction     Unvaccinated Vaccinated
  Unvaccinated          205        613
  Vaccinated            580       2384

               Accuracy : 0.6846
                 95% CI : (0.6695, 0.6994)
    No Information Rate : 0.7924
    P-Value [Acc > NIR] : 1.0000

                  Kappa : 0.0557

 Mcnemar's Test P-Value : 0.3542

            Sensitivity : 0.2611
            Specificity : 0.7955
         Pos Pred Value : 0.2506
         Neg Pred Value : 0.8043
             Prevalence : 0.2076
         Detection Rate : 0.0542
   Detection Prevalence : 0.2163
      Balanced Accuracy : 0.5283

       'Positive' Class : Unvaccinated
```

Figure 248: Confusion matrices of untuned and tuned KNN

Comparison of Model Performance

1. Generate a dataframe of the model performance and create multiple bar plots of these metrics. Combine the plots using the "plot_grid" function

```
Accuracy  <- c(72.7, 54.67, 64.55, 68.22, 68.59, 64.01, 62.59, 76.12, 59.47, 71.66)
Sensitivity <- c(17.8, 58.97, 40.86, 38.34, 26.24, 43.06, 46.62, 30.01, 55.29, 19.62)
Specificity <- c(87.09, 53.53, 70.78, 76.04, 79.68, 69.5, 66.77, 80, 60.56, 85.3)
models <- c('Random Forest', 'Bagging', 'Boosting', 'C-Forest', 'KNN', 'Neural Network',
            'SVM', 'Superlearner', 'Naïve Bayes', 'C5.0')
combined <- cbind(Accuracy, Sensitivity, Specificity)
combined <- as.data.frame(combined)
combined <- cbind(combined, models)
plot_acc <- ggplot(data = combined, aes(x = reorder(models, Accuracy), y = Accuracy)) +
  geom_bar(stat="identity", fill = "steelblue") + xlab("Model") + coord_flip()
plot_sens <- ggplot(data = combined, aes(x = reorder(models, Sensitivity), y = Sensitivity)) +
  geom_bar(stat="identity", fill = "purple") + xlab("Model") + coord_flip()
plot_spec <- ggplot(data = combined, aes(x = reorder(models, Specificity), y = Specificity)) +
  geom_bar(stat="identity", fill = "red") + xlab("Model") + coord_flip()
plot_grid(plot_acc, plot_spec, plot_sens,
          nrow = 2, ncol = 2, labels = "AUTO", label_size = 18, align = "v")
```

Figure 249: Generating a plot to compare model performance

Time Series Forecasting

This chapter uses techniques based on work by Tejendra, S. (No Date)

1. Set the working directory to the folder with the timeseries file

```
# For windows
setwd("D:\\Niall Mannion\\Documents")
# For Mac
setwd("/Users/Niall Mannion/Documents")
```

Figure 250: Setting working directory

2. Read the file into R and extract the columns for the states of the US

```
timeseries <- read.csv("Timeseries.csv", header=T,
skip = 1, na.strings=c(""), stringsAsFactors = T)
tsdf <- timeseries[ ,c(1,2,8,15,22,29,36,43,50,57,
64,71,78,85,92,99,106,112,119,126,132,139,146,152)]
```

Figure 251: Reading timeseries file and extracting desired columns

3. Reshape the time series data

```
news <- melt(tsdf, id.vars=c("Names"))
newsa <- spread(news, Names, value)
newsat <- newsa[ ,c(7, 91, 112)]
```

Figure 252: Reshaping timeseries data

4. View the timeseries dataframe

```
> newsat
   Arkansas Oklahoma Texas
1      55.1     42.8  52.4
2      62.1     58.9  61.8
3      74.8     66.0  65.6
4      65.9     70.9  63.9
5      69.5     70.5  64.5
6      66.5     68.4  62.0
7      68.1     70.3  70.0
8      71.0     65.3  67.9
9      76.5     70.5  74.8
10     82.4     72.1  72.5
11     67.8     75.7  78.4
12     74.9     80.4  76.7
13     75.0     80.1  78.2
14     78.0     73.6  78.6
15     38.5     60.1  43.7
16     59.4     55.3  62.8
17     70.5     71.0  74.9
18     70.0     65.0  67.7
19     60.5     69.4  74.6
20     70.3     76.7  67.6
21     69.7     78.8  74.0
22     72.1     70.9  73.1
23     72.1     71.4  72.3
```

Figure 253: Timeseries dataframe

5. Convert the dataframe to a timeseries dataset and plot the timeseries

```
mymts = ts(newsat, frequency = 1, start = c(1995, 1))
plot(mymts, main = "Time Series of Vaccination Rates", xlab="Year")
```

Figure 254: Converting to timeseries and plotting the timeseries

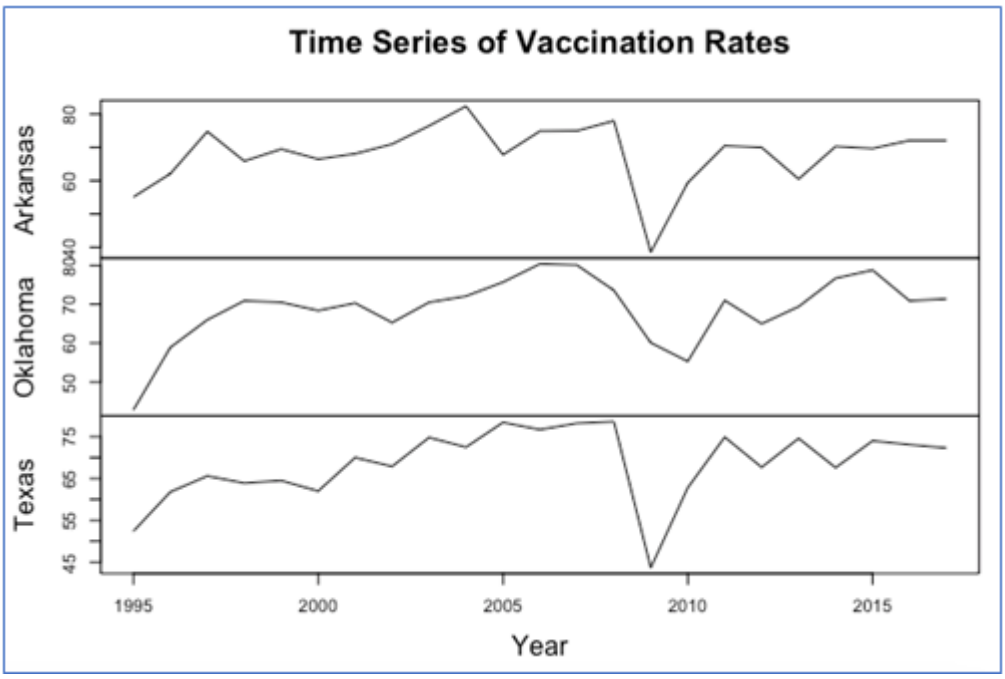

Figure 255: Timeseries plot

Treat Outliers

Texas

1. Retrieve the Texas timeseries, convert to timeseries dataframe and detect and visualise outliers in the timeseries

```
newsat <- newsa[ ,c(112)]
untreated = ts(newsat, frequency = 1, start = c(1995, 1))
outliers_excess_ts <- tso(untreated, types = c("TC", "AO", "LS", "IO", "SLS"))
outliers_excess_ts
plot(outliers_excess_ts)
title(main = "Outlier Plot of Texas Timeseries", cex.main = 1.5, font.main = 1, line = 2.25)
title(ylab = "Vaccination Rate", line = 3, cex.lab = 1.2)
title(xlab = "Year", line = 4, cex.lab = 1.2)
```

Figure 256: Detecting outliers in Texas timeseries

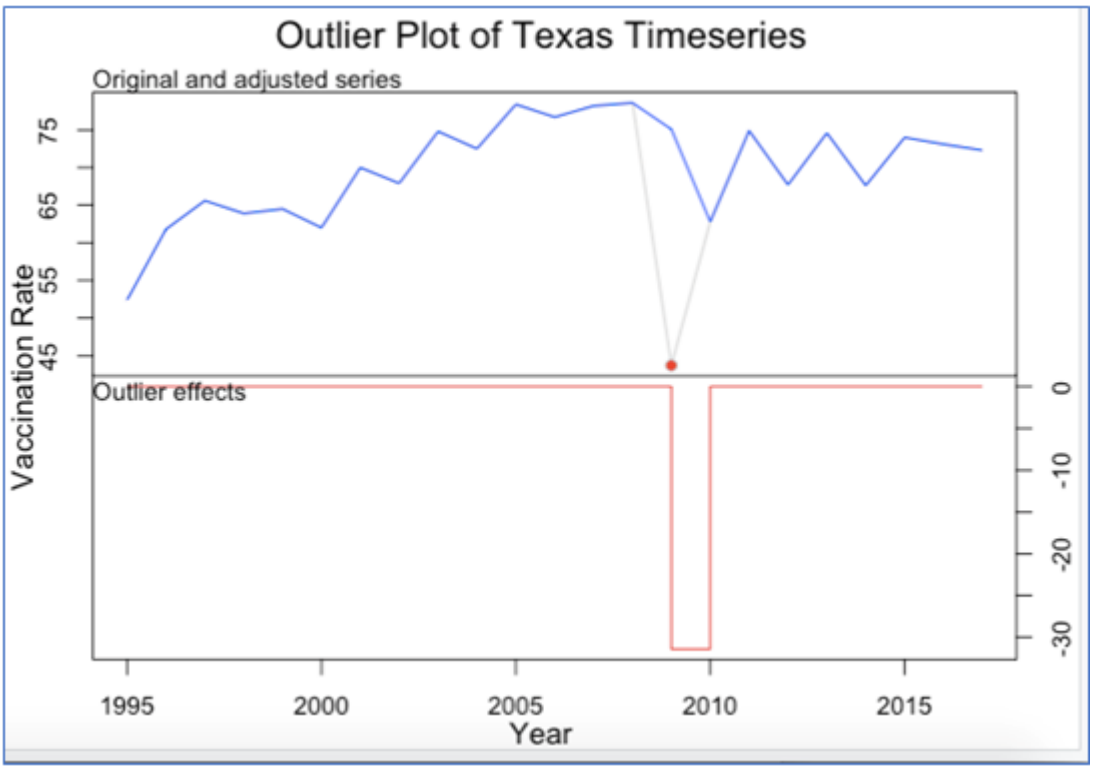

Figure 257: Texas timeseries outliers plot

```
> outliers_excess_ts$outliers
  type ind time   coefhat     tstat
1   AO  15 2009 -31.38751 -7.444673
```

Figure 258: Outliers in Texas timeseries

2. Treat the outliers in the data and plot the treated timeseries and plot the treated timeseries against the untreated data

```
n <- length(untreated)
mo_tc <- outliers("TC", outliers_idx)
tc <- outliers.effects(mo_tc, n)
coefhat <- as.numeric(outliers_excess_ts$outliers["coefhat"])
tc_effect <- coefhat*tc
Outlier_Plot <- ts(tc_effect, frequency = frequency(untreated), start = start(untreated))
Treated <- untreated - Outlier_Plot
plot(cbind(Treated, untreated, Outlier_Plot), main = "Plot of Texas
Vaccination Time Series Ouliter Treatment", xlab = "Year")
```

Figure 259: Treating outliers in Texas timeseries

3. Create the outlier treated Texas timeseries

```
Texas <- Treated
```

Figure 260: Creating outlier treated Texas timeseries

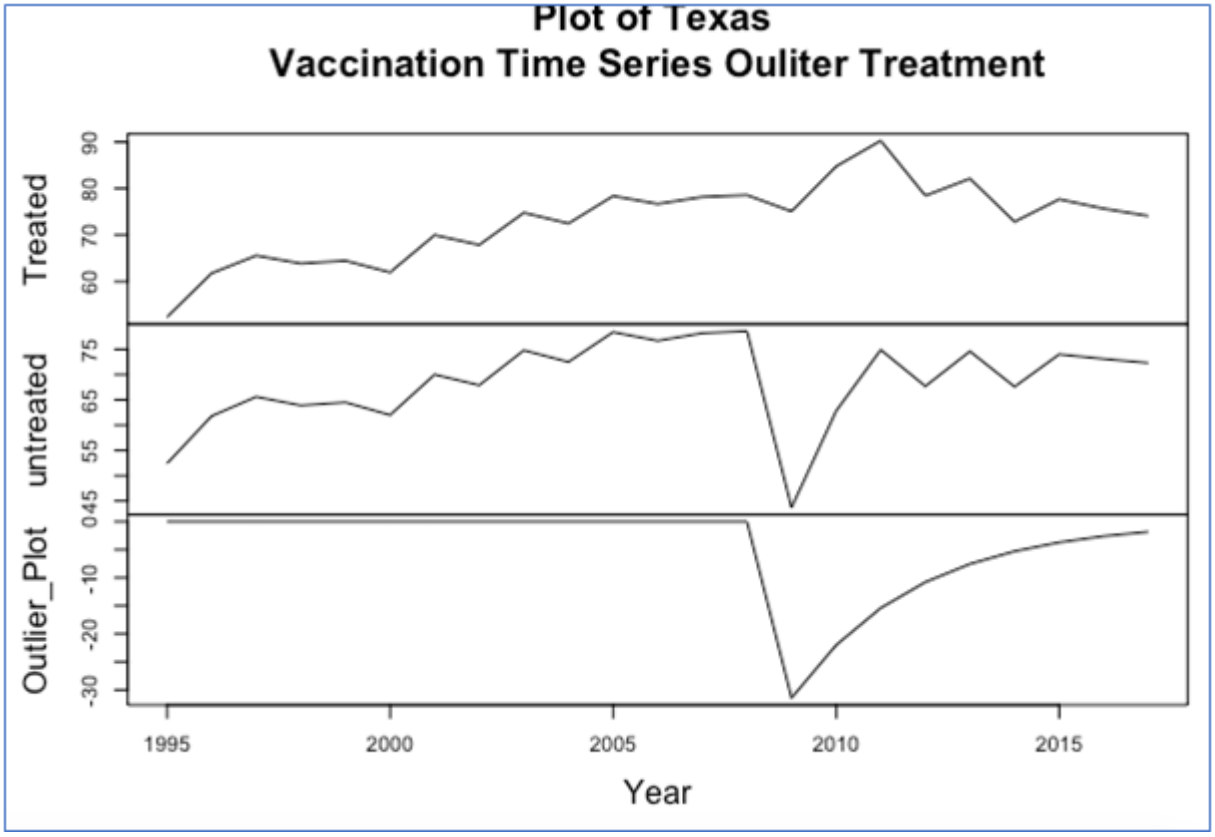

Figure 261: Plot of Texas timeseries treated for outliers versus untreated

4. Repeat the same steps for the Arkansas (column 7) and Oklahoma (column 92) data

```
newsat <- newsa[ ,c(7)]
untreated = ts(newsat, frequency = 1, start = c(1995, 1))
outliers_excess_ts <- tso(untreated, types = c("TC", "AO", "LS", "IO", "SLS"))
outliers_excess_ts
plot(outliers_excess_ts)
title(main = "Outlier Plot of Arkansas Timeseries", cex.main = 1.5, font.main = 1, line = 2.25)
title(ylab = "Vaccination Rate", line = 3, cex.lab = 1.2)
title(xlab = "Year", line = 4, cex.lab = 1.2)

outliers_excess_ts$outliers
(outliers_idx <- outliers_excess_ts$outliers$ind)
n <- length(untreated)
mo_tc <- outliers("TC", outliers_idx)
tc <- outliers.effects(mo_tc, n)
coefhat <- as.numeric(outliers_excess_ts$outliers["coefhat"])
tc_effect <- coefhat*tc
Outlier_Plot <- ts(tc_effect, frequency = frequency(untreated), start = start(untreated))
Treated <- untreated - Outlier_Plot
plot(cbind(Treated, untreated, Outlier_Plot),
     main = "Plot of Arkansas Vaccination Time Series Outlier Treatment", xlab = "Year")
Arkansas <- Treated

newsat <- newsa[ ,c(91)]
untreated = ts(newsat, frequency = 1, start = c(1995, 1))
outliers_excess_ts <- tso(untreated, types = c("TC", "AO", "LS", "IO", "SLS"))
outliers_excess_ts
plot(outliers_excess_ts)
title(main = "Outlier Plot of Oklahoma Timeseries", cex.main = 1.5, font.main = 1, line = 2.25)
title(ylab = "Vaccination Rate", line = 3, cex.lab = 1.2)
title(xlab = "Year", line = 4, cex.lab = 1.2)

outliers_excess_ts$outliers
(outliers_idx <- outliers_excess_ts$outliers$ind)
n <- length(untreated)
mo_tc <- outliers("TC", outliers_idx)
tc <- outliers.effects(mo_tc, n)
coefhat <- as.numeric(outliers_excess_ts$outliers["coefhat"])
tc_effect <- coefhat*tc
Outlier_Plot <- ts(tc_effect, frequency = frequency(untreated), start = start(untreated))
Treated <- untreated - Outlier_Plot
plot(cbind(Treated, untreated, Outlier_Plot),
     main = "Plot of Oklahoma Vaccination Time Series Outlier Treatment", xlab = "Year")
Oklahoma <- Treated
```

Figure 262: Detecting and treating outliers in Arkansas and Oklahoma timeseries

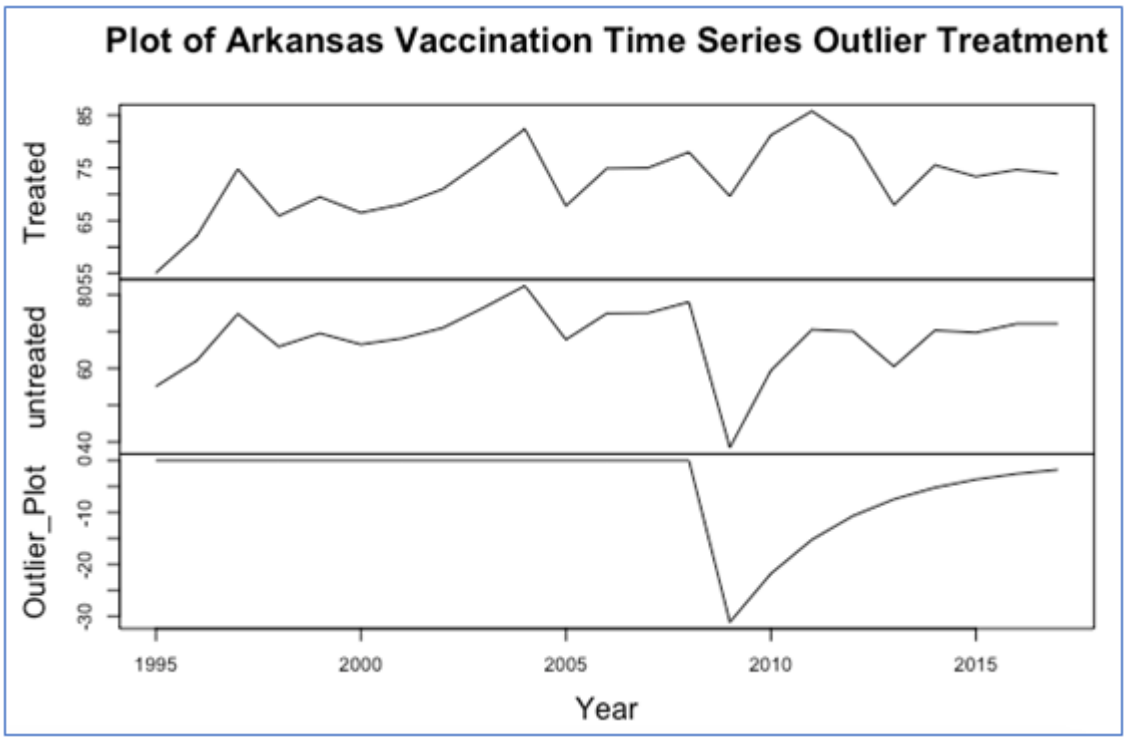

Figure 263: Plot of Arkansas timeseries treated for outliers versus untreated

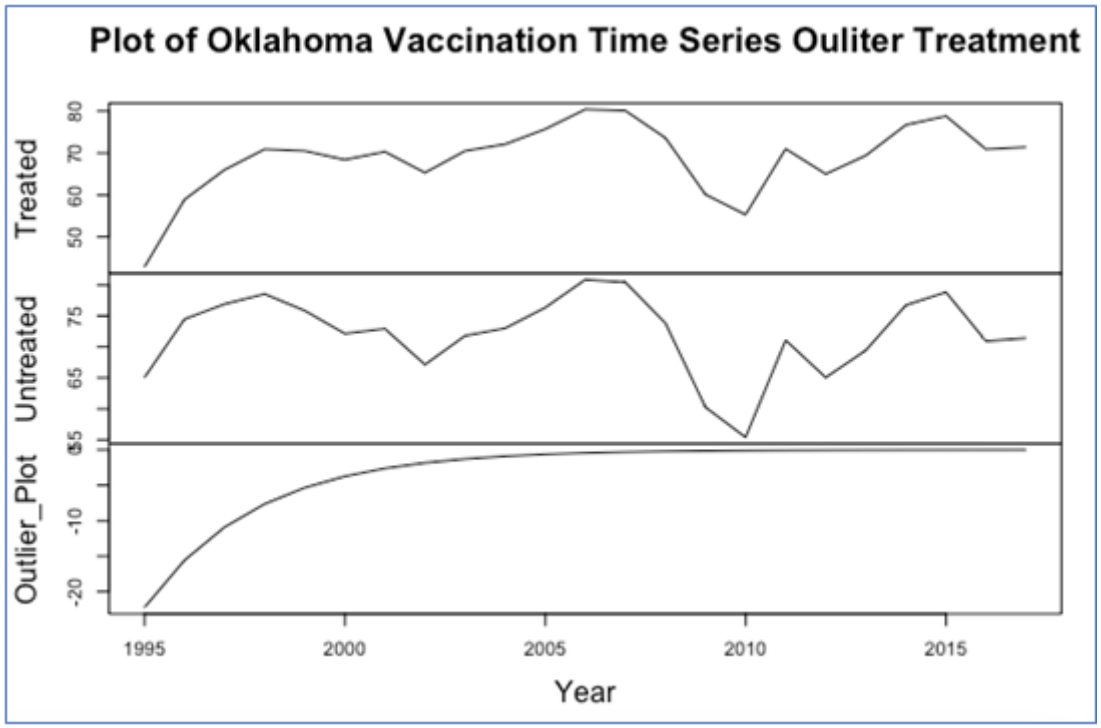

Figure 264: Plot of Oklahoma timeseries treated for outliers versus untreated

5. Combine the treated timeseries datasets

```
adjusted_ts <- cbind(Oklahoma, Texas, Arkansas)
```

Figure 265: Combining the treated timeseries data

6. Test if the data is stationary (if P<0.05 it is stationary and therefore can be used for the study)

```
> apply(adjusted_ts, 2, adfTest, lags=0, type="c",
+ title = "ADF Test for Vaccination Timeseries Data")
$Oklahoma

Title:
 ADF Test for Vaccination Timeseries Data

Test Results:
  PARAMETER:
    Lag Order: 0
  STATISTIC:
    Dickey-Fuller: -4.1323
  P VALUE:
    0.01

Description:
 Thu Apr 30 04:57:01 2020 by user:


$Texas

Title:
 ADF Test for Vaccination Timeseries Data

Test Results:
  PARAMETER:
    Lag Order: 0
  STATISTIC:
    Dickey-Fuller: -2.8309
  P VALUE:
    0.07286

Description:
 Thu Apr 30 04:57:01 2020 by user:


$Arkansas

Title:
 ADF Test for Vaccination Timeseries Data

Test Results:
  PARAMETER:
    Lag Order: 0
  STATISTIC:
    Dickey-Fuller: -3.9121
  P VALUE:
    0.01
```

Figure 266: Stationary test on timeseries data

Error of Forecasting Methods

1. Take a training samples from the year 1995-2012, extract the Texas column, convert the data to stationary, forecast the timeseries and adjust the stationary by adding the last value from the Texas timeseries (based on https://otexts.com/fpp2/accuracy.html)

```
ts_window <- window(adjusted_ts,start = 1995, end=c(2012))
texas_ts <- adjusted_ts[,2]
stnry = diffM(ts_window)
var.a <- vars::VAR(stnry, lag.max = 10, ic = "AIC", type = "none")
fcast = predict(var.a, n.ahead = 5)
Texas_fcast = fcast$fcst[2]
x = Texas_fcast$Texas[,1]
tail(ts_window)
```

Figure 267: Forecasting training data

```
> tail(ts_window)
Time Series:
Start = 2007
End = 2012
Frequency = 1
     Oklahoma    Texas Arkansas
2007     80.1 78.20000 75.00000
2008     73.6 78.60000 78.00000
2009     60.1 75.08751 69.64545
2010     55.3 84.77126 81.20182
2011     71.0 90.27988 85.76127
2012     65.0 78.46592 80.68289
```

Figure 268: End of timeseries data

2. Convert the data to its normal form by adding the 2012 value for Texas onto the data

```
x = cumsum(x) + 78.46
```

Figure 269: Converting timeseries back to original format

3. Make the x value the final prediction for Texas

```
forecast_ML <- x
```

Figure 270: Making final forecast

4. Train three other models for comparison

```
meanf_ts <- meanf(texas_ts, h = 5)
rwf_ts <- rwf(texas_ts, h = 5)
snaive_ts <- snaive(texas_ts, h = 5)
```

Figure 271: Building three time series forecasting methods

5. Create test timeseries data for between 2013-2017

```
target_ts <- window(texas_ts, start = 2013)
target_ts <- as.numeric(target_ts)
```

Figure 272: Generating test timeseries set

6. Find the Mean Absolute Error (MAE) of each model

```
> accuracy(meanf_ts, target_ts)
                      ME     RMSE      MAE       MPE     MAPE     MASE       ACF1
Training set -2.471967e-15 8.355396 6.631752 -1.427520 9.540713 1.4789427 0.6564463
Test set      3.525650e+00 4.792856 3.565148  4.441312 4.495512 0.7950613         NA
> accuracy(rwf_ts, target_ts)
                    ME     RMSE      MAE       MPE     MAPE      MASE       ACF1
Training set 0.9867922 5.554285 4.484117 1.288106 6.051274 1.0000000 -0.3898962
Test set     2.3902672 4.031705 2.883919 2.954542 3.631934 0.6431408         NA
> accuracy(snaive_ts, target_ts)
                    ME     RMSE      MAE       MPE     MAPE      MASE       ACF1
Training set 0.9867922 5.554285 4.484117 1.288106 6.051274 1.0000000 -0.3898962
Test set     2.3902672 4.031705 2.883919 2.954542 3.631934 0.6431408         NA
> accuracy(forecast_ML, target_ts)
                ME     RMSE      MAE       MPE     MAPE
Test set -6.836013 7.311107 6.836013 -8.831071 8.831071
```

Figure 273: Finding accuracy of each forecast technique

7. View the predicted values for the timeseries

```
> meanf_ts
     Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
2018       72.97405 61.44371 84.50438 54.87551 91.07258
2019       72.97405 61.44371 84.50438 54.87551 91.07258
2020       72.97405 61.44371 84.50438 54.87551 91.07258
2021       72.97405 61.44371 84.50438 54.87551 91.07258
2022       72.97405 61.44371 84.50438 54.87551 91.07258
```

Figure 274: Forecasts using mean method

```
> rwf_ts
     Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
2018       74.10943 66.99132 81.22753 63.22323 84.99563
2019       74.10943 64.04291 84.17594 58.71402 89.50484
2020       74.10943 61.78051 86.43834 55.25398 92.96488
2021       74.10943 59.87322 88.34563 52.33703 95.88182
2022       74.10943 58.19287 90.02599 49.76715 98.45171
```

Figure 275: Forecast using naive method

```
> snaive_ts
     Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
2018       74.10943 66.99132 81.22753 63.22323 84.99563
2019       74.10943 64.04291 84.17594 58.71402 89.50484
2020       74.10943 61.78051 86.43834 55.25398 92.96488
2021       74.10943 59.87322 88.34563 52.33703 95.88182
2022       74.10943 58.19287 90.02599 49.76715 98.45171
```

Figure 276: Forecast using seasonal naive method

```
> forecast_ML
[1] 91.70089 75.05027 86.42756 83.18127 80.31854
```

Figure 277: Forecast using forecastML

Forecast Using ForecastML package

1. Convert the data to stationary data and plot the stationary data

```
autoplot(ts(stnry, start = c(1990, 1), frequency = 1), lwd =  1.6) +
  ggtitle("Time Series Plot of the stationary Texas Vaccination Time-Series") +
  ylab("Vaccination Rate") + xlab("Year")
```

Figure 278: Convert timeseries to stationary and plot the data

2. Plot the timeseries data

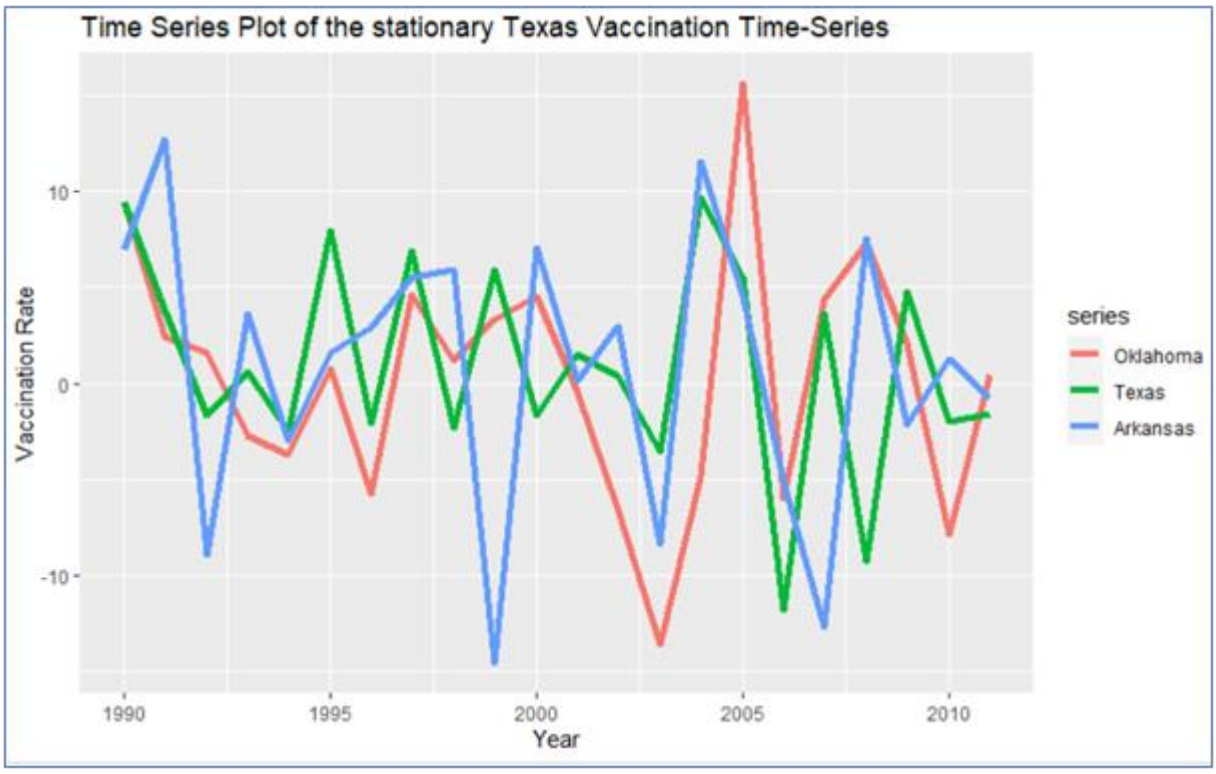

Figure 279: Plot of vaccination rates timeseries data

3. Perform forecasting five years into the future using the "predict" function

```
var.a <- vars::VAR(stnry, lag.max = 10, ic = "AIC", type = "none")
fcast = predict(var.a, n.ahead = 5)
par(mar = c(2.5,2.5,2.5,2.5))
plot(fcast)
```

Figure 280: Forecasting and plotting the forecast of the timeseries data

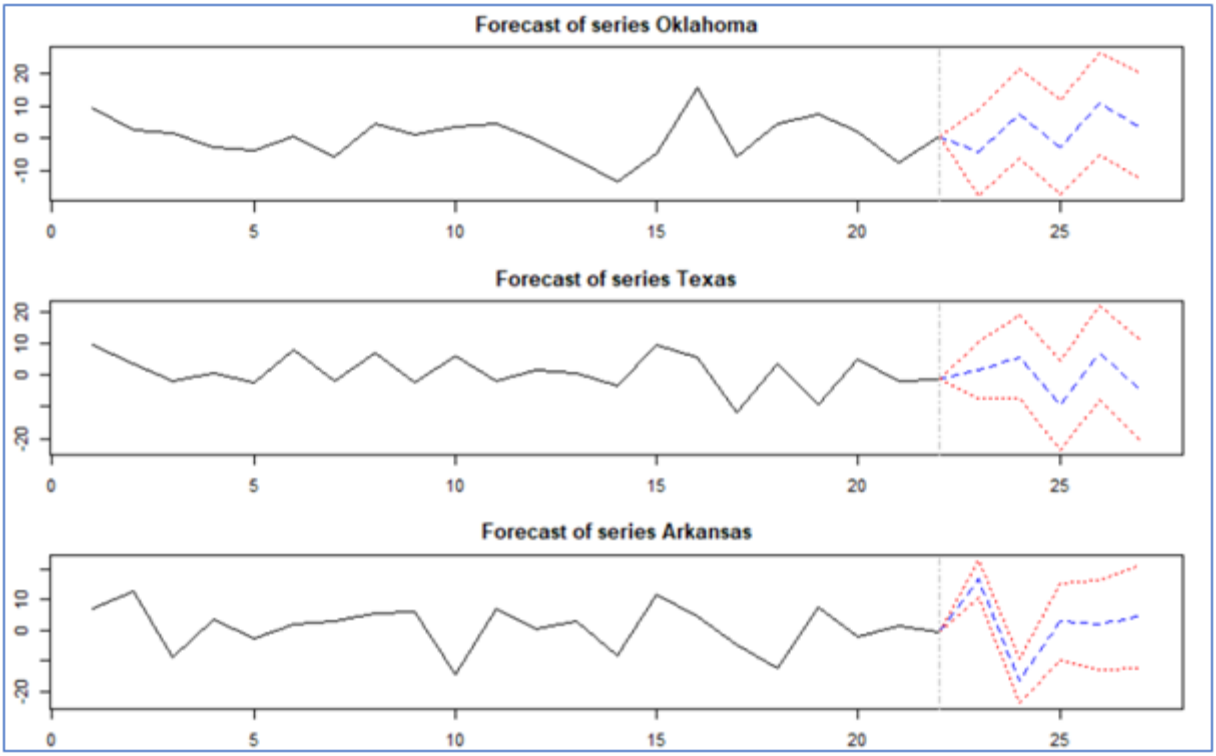

Figure 281: Plot of timeseries forecast for vaccination rates

4. Extract the forecast value for Texas, add the value from 2012 to the forecasted data and plot the forecasted timeseries

```
Texas_fcast = fcast$fcst[2]
x = Texas_fcast$Texas[,1]
tail(adjusted_ts)
x = cumsum(x) + 74.1
par(mar = c(4,4,1,4))
Forecast_Tex =ts(c(x), start = c(2013,1), frequency = 1)
plot(Forecast_Tex, main= "Forecasted Vaccination Rates for Texas", xlab = "Year",
ylab = "Vaccination Rate")
```

Figure 282: Plotting Texas forecast

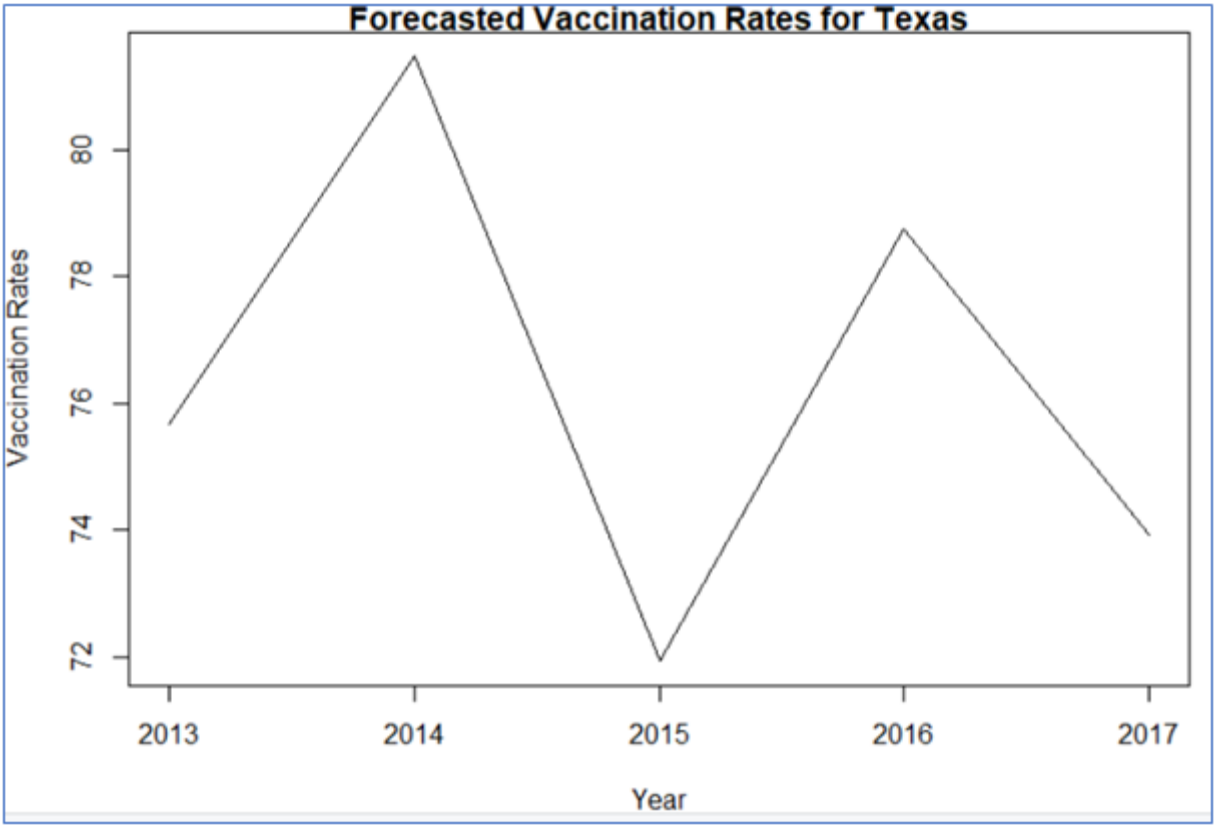

Figure 283: Forecasted vaccination rate for Texas

5. Join the forecasted values with the original timeseries data, convert the data to a dataframe and rename the column with the vaccination rates to "Texas"

```
Texas_Forecast =ts(c(adjusted_ts[,2], x), start = c(1995,1), frequency = 1)
Texas_df <- as.data.frame(Texas_Forecast[1:28])
colnames(Texas_df) <- c("Texas")
```

Figure 284: Generating final Texas forecast dataframe

6. Repeat these steps for the Arkansas and Oklahoma timeseries data

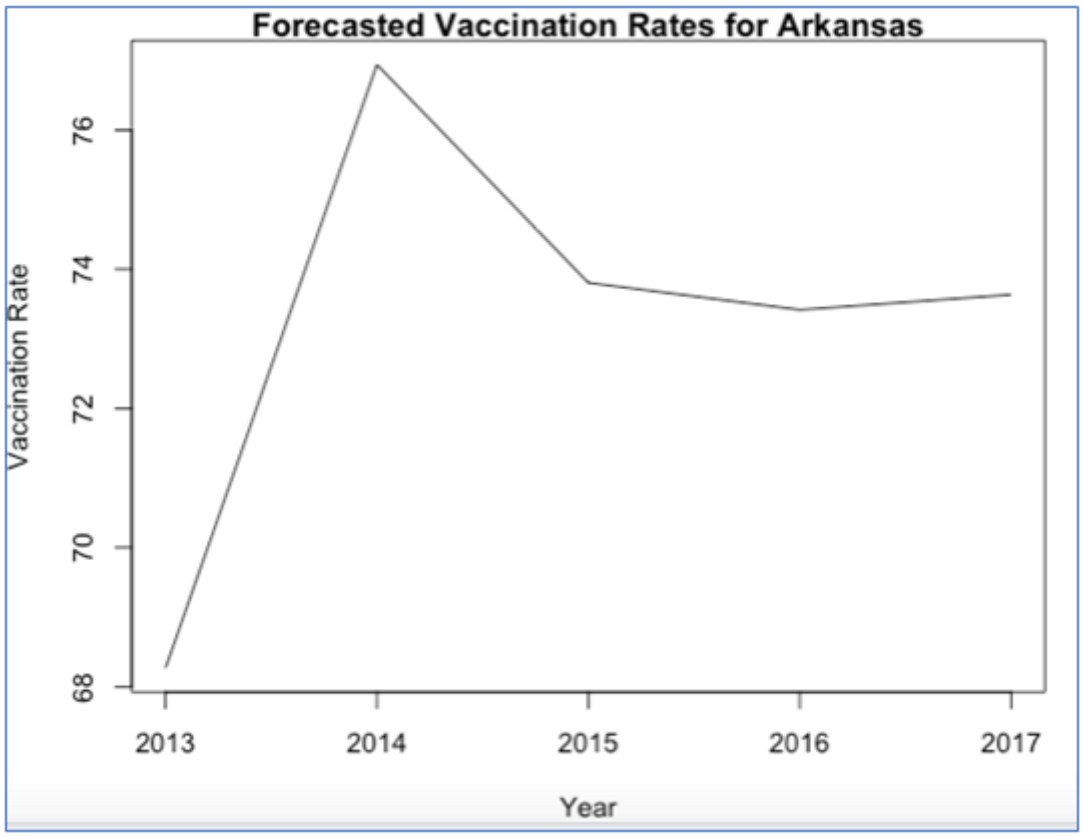

Figure 285: Forecasted vaccination rates for Arkansas

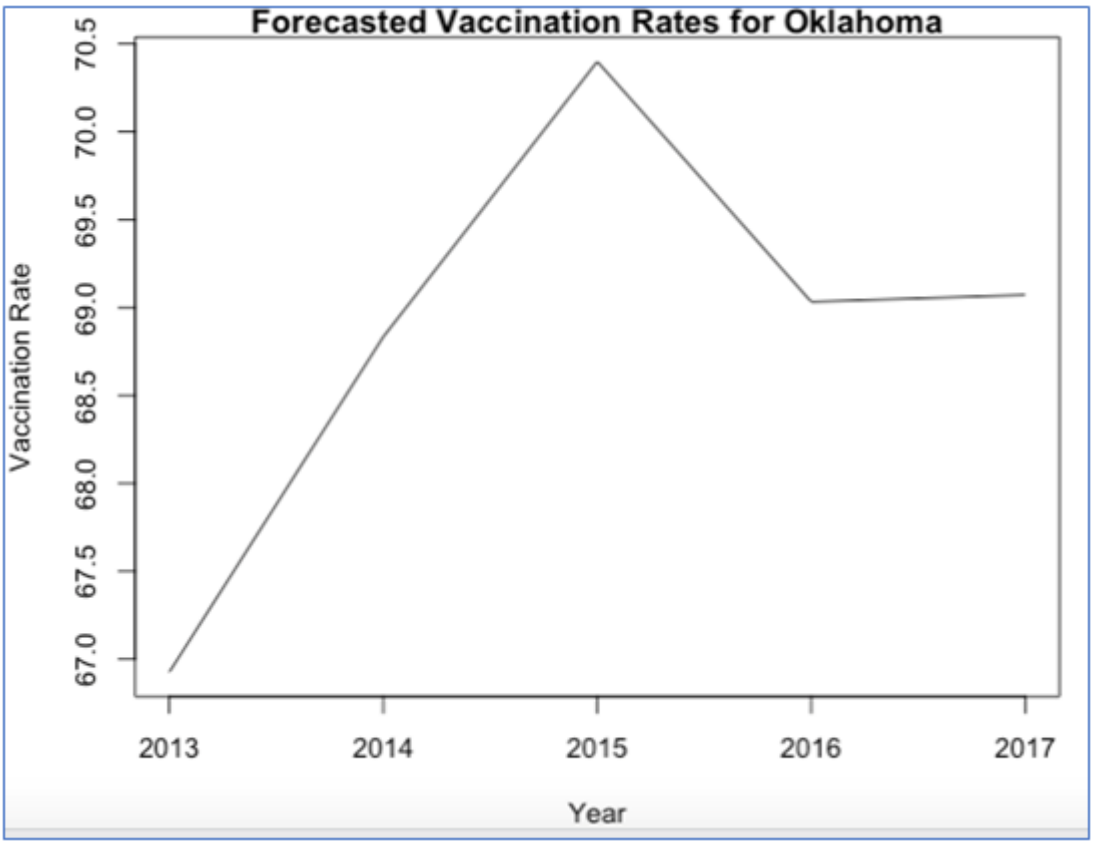

Figure 286: Forecasted vaccination rate for Oklahoma

7. Combine the forecast data, convert them to a dataframe, reshape the data and generate a line plot using "GGPlot2" of the timeseries forecasts

```
combined_ts <- cbind(Texas_Forecast, Arkansas_Forecast, Oklahoma_Forecast)
combined_df <- as.data.frame(combined_ts)
combined_df$Year <- seq(1995, 2022, by = 1)
reshaped_ts <- melt(combined_df, id="Year")
my_theme <- theme(axis.title.x = element_text(size = 15),
axis.text.x = element_text(size = 15), axis.title.y = element_text(size = 15),
plot.title = element_text(size = 17, hjust = 0.5))
ggplot(data = reshaped_ts, aes(x = Year, y = value, colour = variable)) + ylab("Vaccination Rate") +
ggtitle("Forecasted Vaccination Rates for US States") + geom_line(size = 2.6) + my_theme
```

Figure 287: Generating timeseries forecast line plot

Dashboard

1. The code regenerating the dashboard using Shiny and Flexdashboard is shown in the .rmd file
2. The code contains most of the same code that was previously shown
3. To create a new or markdown file click 'File' - 'New File'- 'R Markdown'

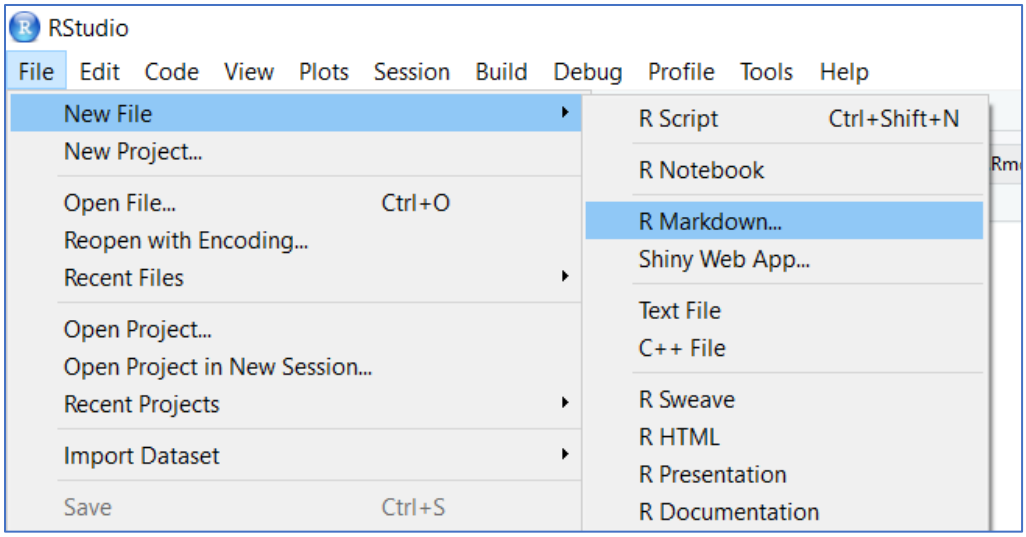

Figure 288: Creating new Rmarkdown file

4. Create the formatting for the dashboard setting the title, author, setting it to today's date, set the output to flexdashboard with rows, social to 'menu' source code 'embed'. Set the runtime to Shiny and add any files we used. Make sure these files are in the same directory as the .rmd file

```
---
title: "Vaccine Dashboard"
author: "Niall Mannion"
date: "`r format(Sys.time(), '%d %b %Y')`"
output:
  flexdashboard::flex_dashboard:
    orientation: rows
    social: menu
    source_code: embed
runtime: shiny
resource_files:
- timeseries.csv
---
```

Figure 289: Setting up Flexdashboard

5. At the start of the code that you do not want to include in your dashboard put the code shown in figure 366

```
```{r global, include=FALSE}
```

Figure 290: Setting invisible code chunks

6. Enter the code in this block that you want to appear in the final dashboard. the code itself will not appear
7. The only difference between the code in this file and the coding final or file is that only contains the most important visuals and tables, under the NIS training data and test was read directly in, and the Google Maps data was changed to make it more
```

suitable for a dashboard. For the data mining the best performing algorithm (Superlearner) was used

8. Load the training and test data, convert the outcome to vaccinated/unvaccinated and remove the ID column

```
trains <- read.csv("training_data.csv", header=T, na.strings = c(""), stringsAsFactors = T)
test <- read.csv("test_data.csv", header=T, na.strings = c(""), stringsAsFactors = T)
levels(trains$Vaccination_Status) <- c("Unvaccinated", "Vaccinated")
levels(test$Vaccination_Status) <- c("Unvaccinated", "Vaccinated")
trains <- trains[ ,c(2:18)]
test <- test[ ,c(2:18)]
```

Figure 291: Loading and preparing the NIS data

9. At the end of the code chunk leave three dashes as shown in figure 367

Figure 292: Ending code chunk

10. The structure of the first page of the dashboard as shown in figure 368. The 'ggplotly' function allows you to create interactive ggplots. enter any of the visualisations you wish to include in the dashboard into the brackets after 'ggplotly' function. Put your titles after each of the three hashtags, and set the row height do whatever you want. For more Font Awesome icons go to https://fontawesome.com/v4.7.0/icons/

````
Online Dashboard {data-icon="fa-comment-o"}
====================

Row{data-height=450}
--------

### Most Common Words In Vaccine Related Texas Tweets
```{r, echo=FALSE}
ggplotly(me)
```

### Most Common Emotions in Vaccine Related Texas Tweets
```{r, echo=FALSE}
ggplotly(d)
```

Row{data-height=500}
--------

### Google Search Hits in 'Vaccine' By State

```{r, echo=FALSE}
ggplotly(g)

```

### Google Search Interest in Vaccine Related Keywords Over Time (US)
```{r, echo=FALSE}
ggplotly(gt)
```
````

Figure 293: Creating the first page of the dashboard

11. For the second page add the visuals from the NIS study to the dashboard

```
NIS Data Mining {data-icon="fa-database"}
====================
Row
-----------------------------------------------------------------------
### Variable Importance
```{r, echo=FALSE}
ggplotly(b)
```
Model Performance
```{r, echo=FALSE}
ggplotly(f)
```

Row {.tabset}

Education Status
```{r, echo=FALSE}
ggplotly(plot5)
```
Household Size
```{r, echo=FALSE}
ggplotly(plot1)
```
Income Group
```{r, echo=FALSE}
ggplotly(plot16)
```
Insurance Type
```{r, echo=FALSE}
ggplotly(plot10)
```
Mother Age Group
```{r, echo=FALSE}
ggplotly(plot14)
```
Race
```{r, echo=FALSE}
ggplotly(plot15)
```
Number of Healthcare Providers
```{r, echo=FALSE}
ggplotly(plot11)
```
Parents' Marital Status
```{r, echo=FALSE}
ggplotly(plot8)
```
```

Figure 294: Second page of dashboard

12. On the third page use the 'datatable' function to show the predicted immunisation status of a sample of children. The ‘bound’ table contains these predicted values in this study

```
Predicted Vaccination Status {data-icon="fa-syringe"}
====================

Row
--------
### Table of Predicted Vaccination Status

```{r, echo=FALSE}
datatable(bound)
```
```

Figure 295: Third page of dashboard

13. The fourth page contains the timeseries data, with the plot of the forecast (q) and a datatable of the past immunization rates.

```
Time Series Forecasting {data-icon="fa-clock"}
====================
Row
--------

### Table of Timeseries Data
```{r, echo=FALSE}
datatable(timeseries)
```

Row

Timeseries Forecast Line Chart of Vaccination Rates

```{r, echo=FALSE}
ggplotly(q)
```
```

Figure 296: Fourth page of dashboard

SQL Server Management Studio

Loading the data to SQL Database

Creating Table in SSMS

Creating Table in SSMS

1. Create the timeseries table

```
CREATE TABLE Time_Series_Destination (
State VARCHAR(100), "Year_1995" NUMERIC(10,2), "Year_1996" NUMERIC(10,2),
"Year_1997" NUMERIC(10,2), "Year_1998" NUMERIC(10,2), "Year_1999" NUMERIC(10,2),
"Year_2000" NUMERIC(10,2), "Year_2001" NUMERIC(10,2), "Year_2002" NUMERIC(10,2),
"Year_2003" NUMERIC(10,2), "Year_2004" NUMERIC(10,2), "Year_2005" NUMERIC(10,2),
"Year_2006" NUMERIC(10,2), "Year_2007" NUMERIC(10,2), "Year_2008" NUMERIC(10,2),
"Year_2009" NUMERIC(10,2), "Year_2010" NUMERIC(10,2), "Year_2011" NUMERIC(10,2),
"Year_2012" NUMERIC(10,2), "Year_2013" NUMERIC(10,2), "Year_2014" NUMERIC(10,2),
"Year_2015" NUMERIC(10,2), "Year_2016" NUMERIC(10,2), "Year_2017" NUMERIC(10,2));
```

Figure 297: Creating timeseries table

2. Create the table for storing the tweets

```
CREATE TABLE Table_of_Tweet (
Date_Created VARCHAR(50), Text_of TEXT,
Favourite_Count int, Retweet_Count int,
Hashtag TEXT, Language TEXT, Location TEXT);
```

Figure 298: Creating twitter table in SSMS

3. Create the table for the National Immunization Survey (NIS) data

```
CREATE TABLE NISurvey (
Duration_Of_Breast_or_Formula   NUMERIC(10,2),
HouseholdSize   NUMERIC,
Was_Child_Breastfed VARCHAR(255),
Region  VARCHAR(255),
Child_Number    VARCHAR(255),
Child_Currently_Receiving_WIC   VARCHAR(255),
Education_Status    VARCHAR(255),
FRSTBRN VARCHAR(255),
Income_To_Poverty_Ratio VARCHAR(255),
Mother_Age_Group    VARCHAR(255),
Marital_Status  VARCHAR(255),
Race    VARCHAR(255),
House_Ownership_Status  VARCHAR(255),
Provider_Facility VARCHAR(255),
Vaccination_Status  VARCHAR(255),
Insurance_Type  VARCHAR(255),
Number_Providers VARCHAR(255));
```

Figure 299: SQL code for NIS table

4. Create the table for Google trends by time

```
CREATE TABLE Google_Trends_By_Time (
Date_of_Search DATE, Search_Volume int,
Time_Ago VARCHAR(100), Country VARCHAR(100), Keyword VARCHAR(50),
Category int, Gprop VARCHAR(100));
```

Figure 300: Creating table for Google trends by time

SQL Server Integration Studio

1. Create a new project by clicking ‘File’- ‘New’- ‘Project’
2. Highlight the ‘Integration Services Project’ selection as shown in figure 1
3. Enter the project name and click ‘OK’

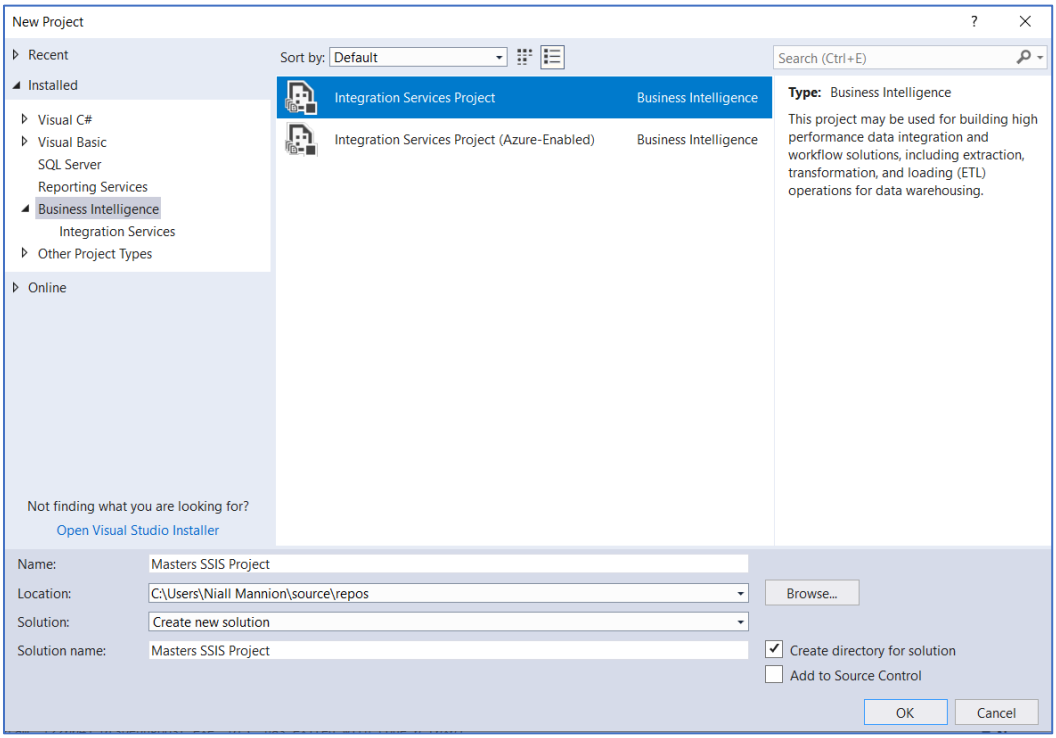

Figure 301: Creating a new SSIS project

4. Drag and drop ‘Data Flow Task’ into the blue pane. And rename it ‘Masters Data Flow

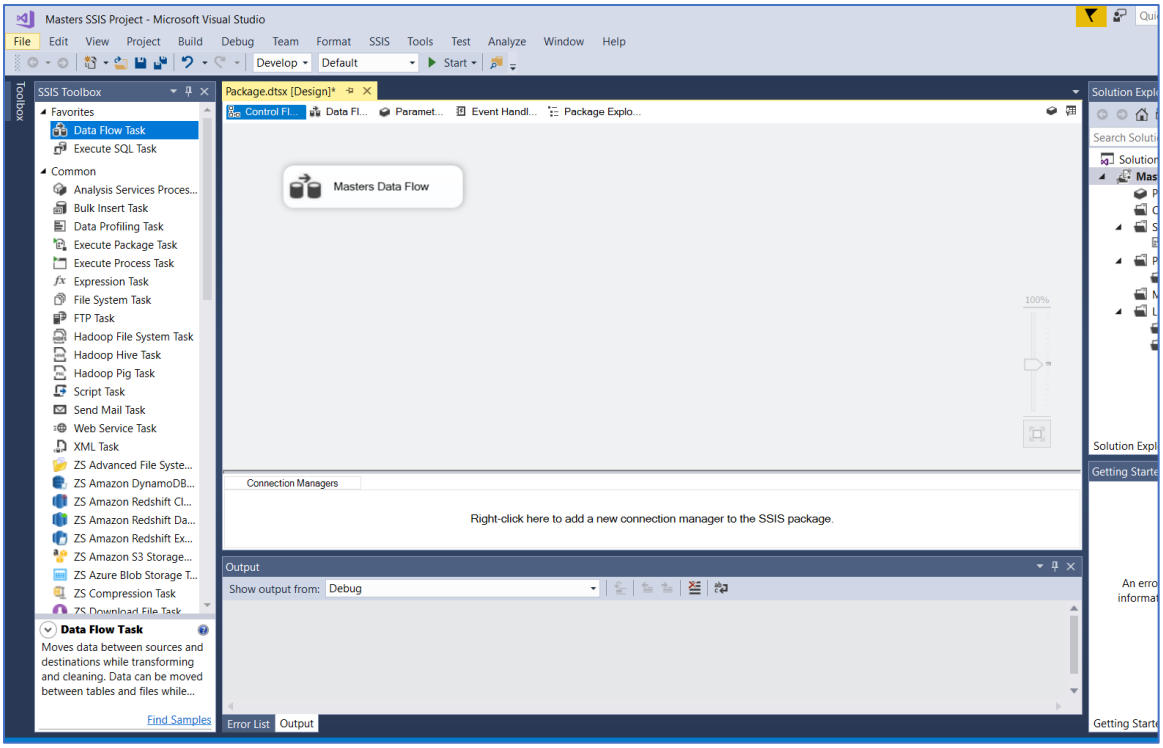

Figure 302: Creating a data flow task

5. Double click on the dataflow task
6. Drag and drop the 'Flat File Source' option into the pane

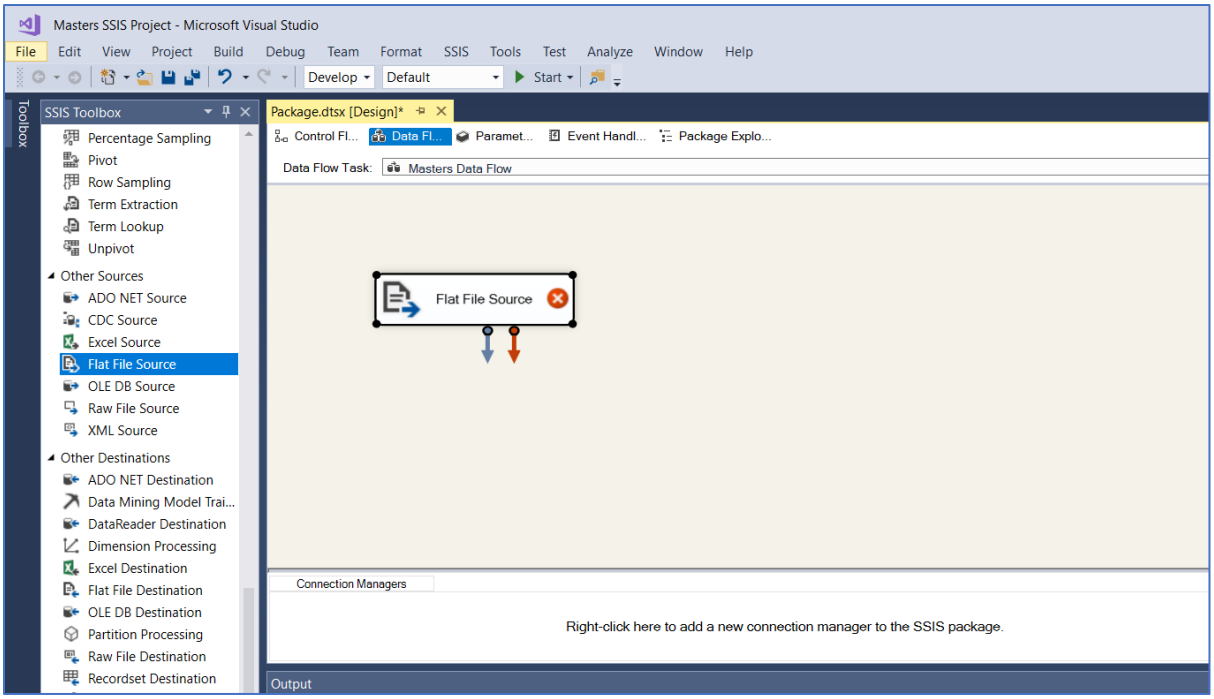

Figure 303: Creating flat file source

7. Click on the "New" box to create a new flat file connection manager

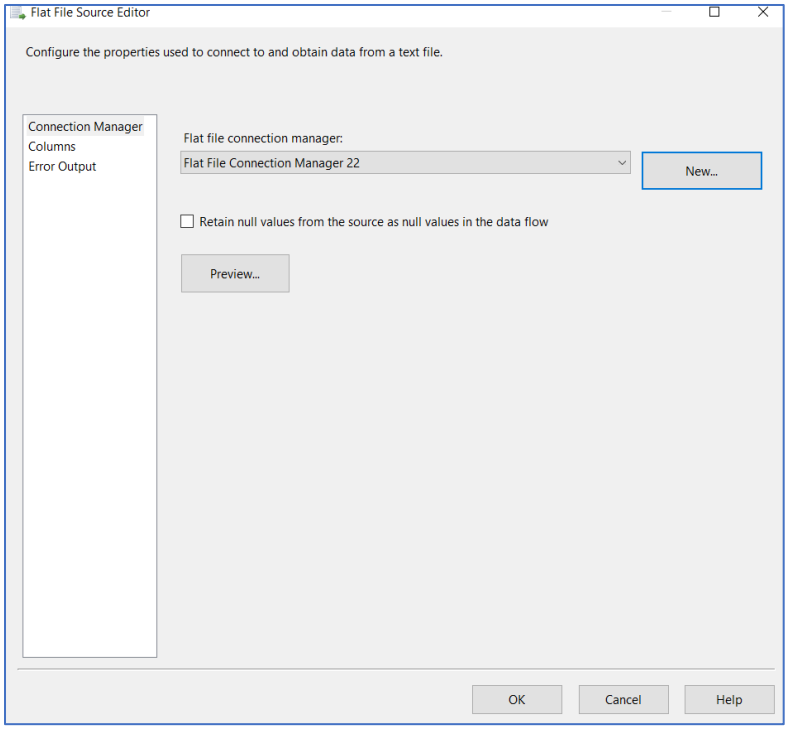

Figure 304: New flat file connection

8. Rename the ‘Connection Manager name text to ‘google trend by time connection’, and ‘google trend connection’ as the description
9. Click on the “Browse” button

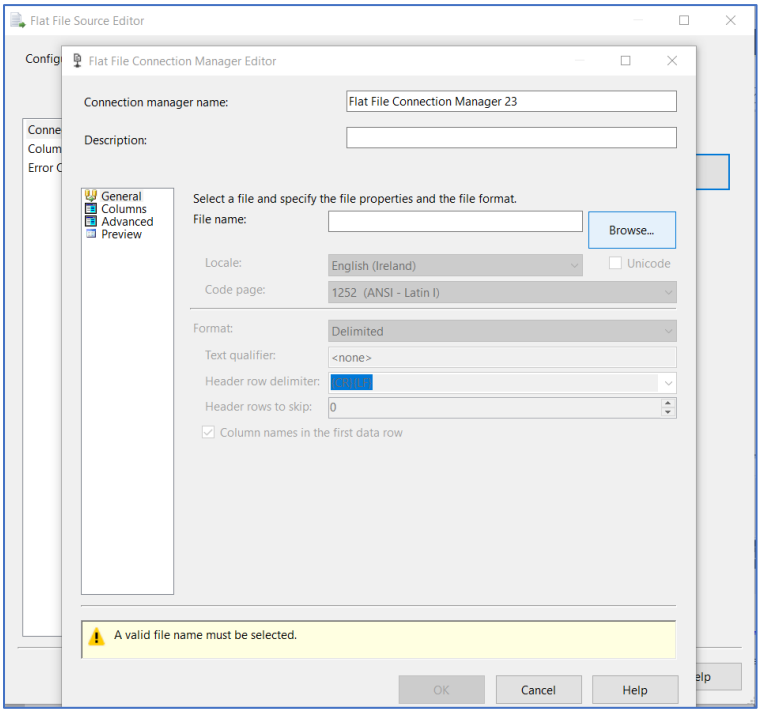

Figure 305: Finding data source

10. Click the ‘Text Files’ box and change the selection to ‘CSV files(*.csv)’
11. Double click on the target file

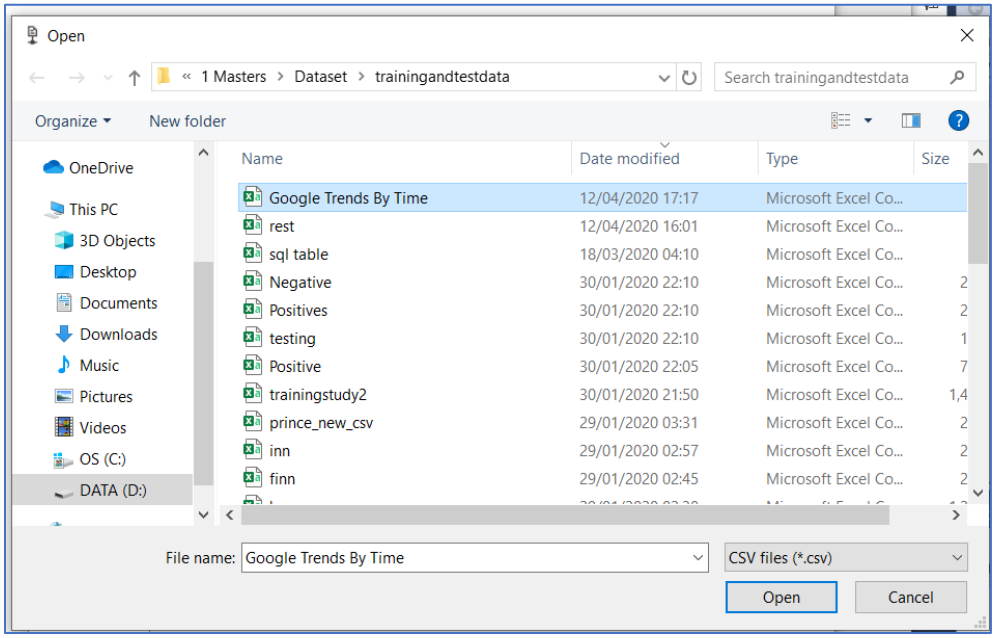

Figure 306: Importing Google Trends data

12. Leave the default parameters unchanged

13. Click on the Columns tab on the left, and ensure that the columns in the columns are correct then click ‘OK’

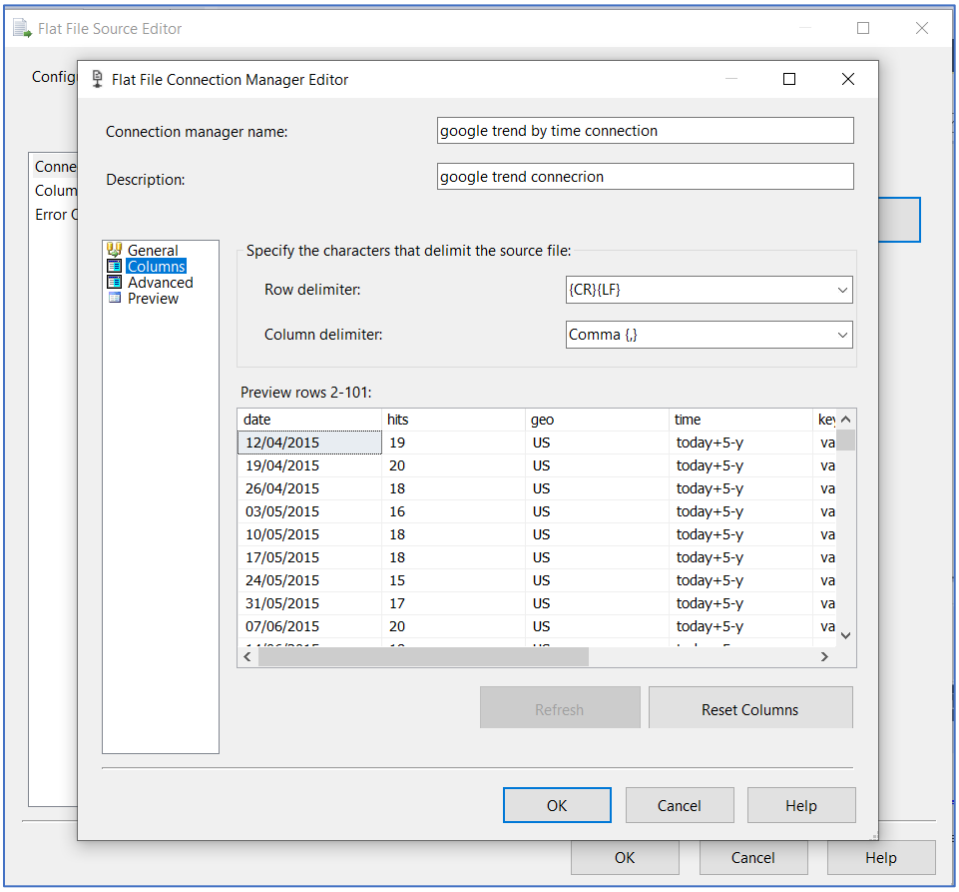

Figure 307: Columns tab

14. Click on the columns tab to choose the columns to include. In this case all the columns were included

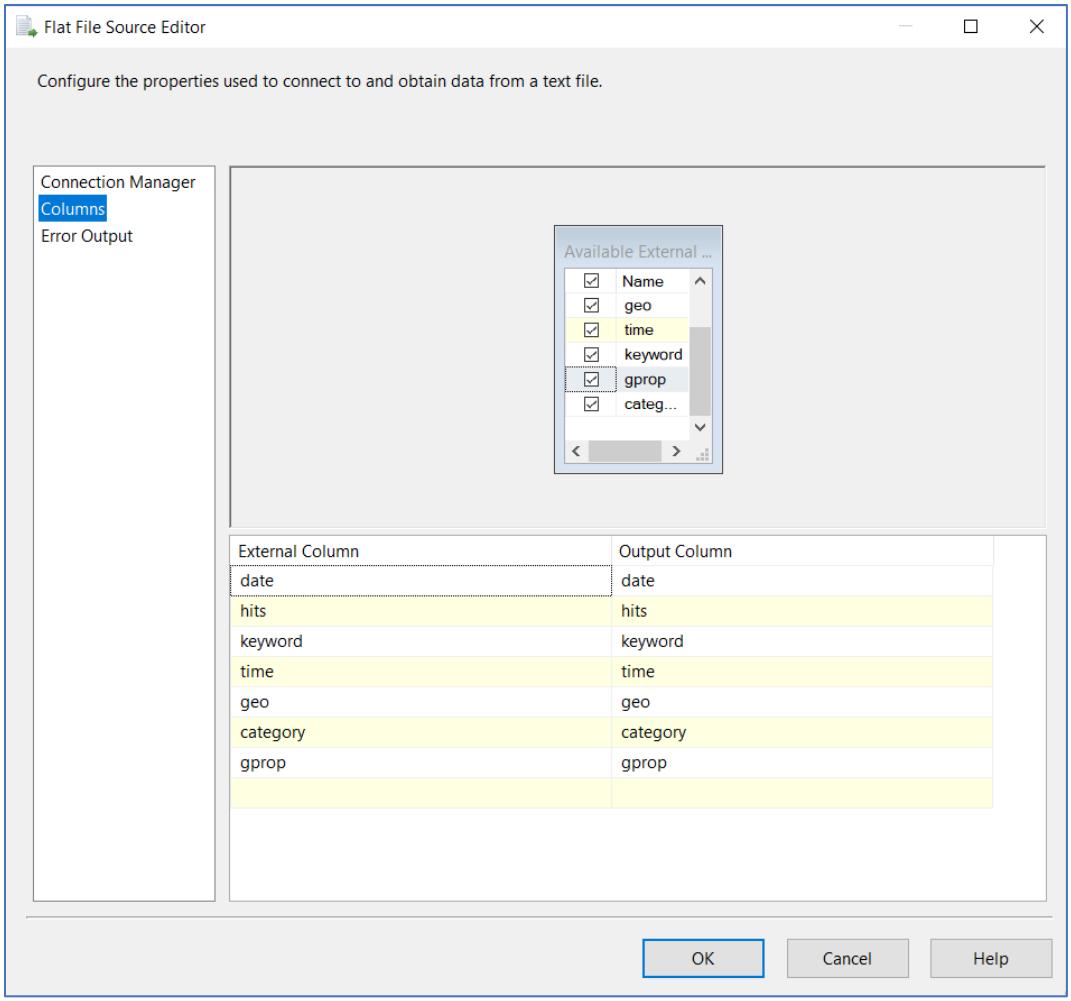

Figure 308: Select columns to include in google trends by time table

15. Click ‘OK’
16. Repeat these same steps for the timeseries, NIS and Twitter data
17. Drag and drop an ‘OLEDB’ destination to the page, rename it to a suitable name and connect the blue arrow from the Google Trends source to the destination

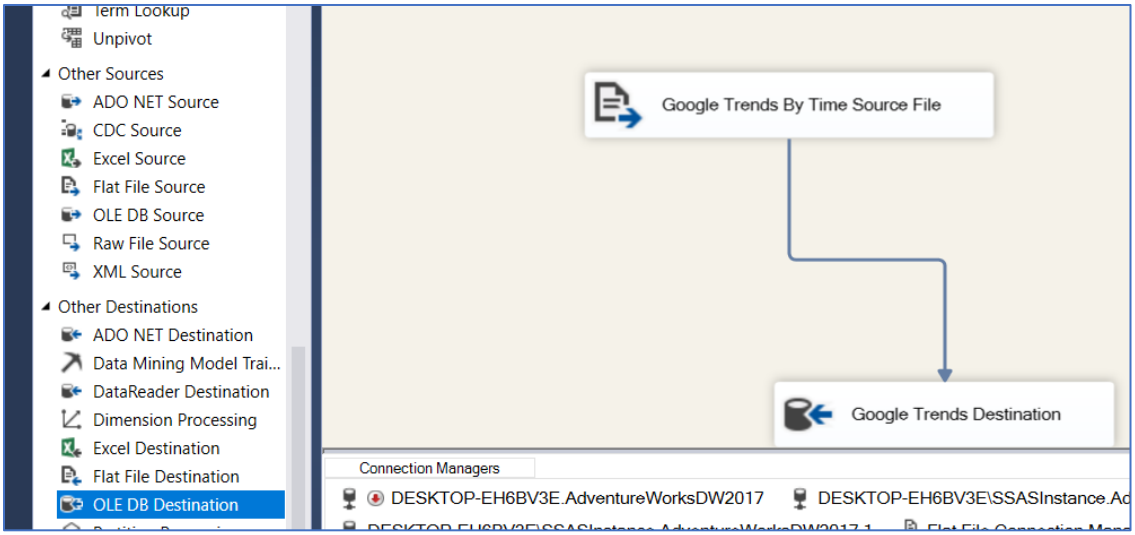

Figure 309: Adding OLDEDB destination

18. In the Connection Manager tab find your SSMS database and your table

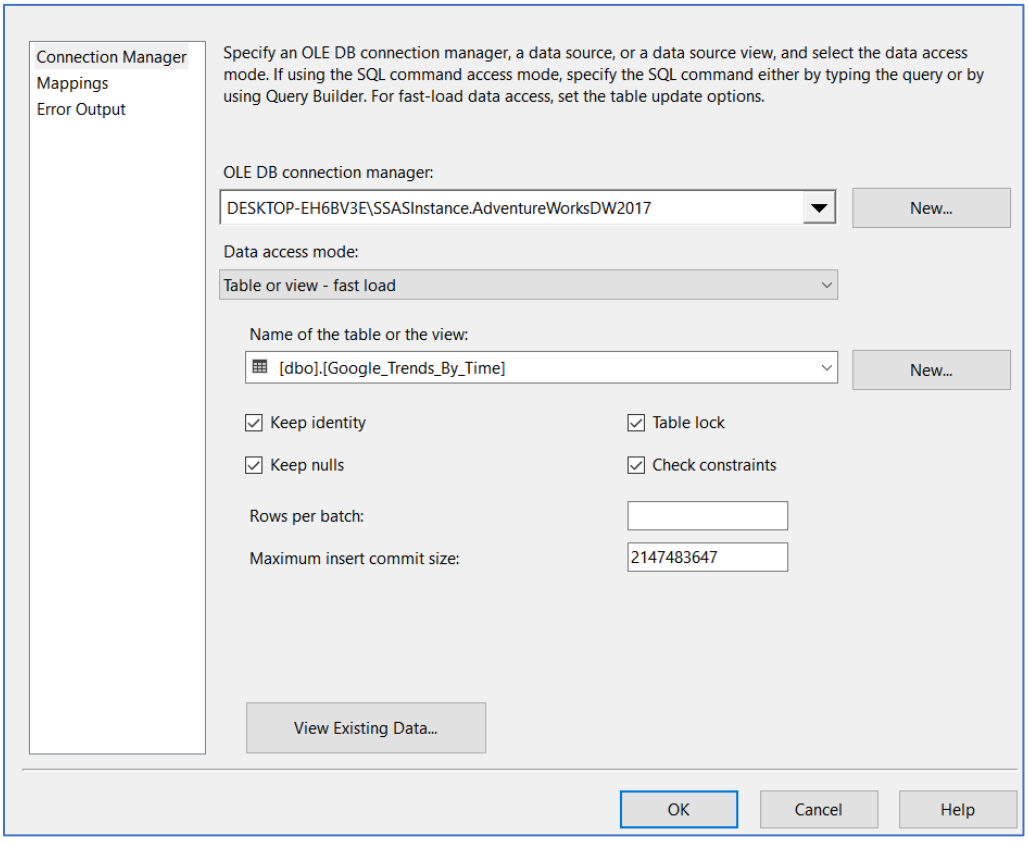

Figure 310: Choosing database and table

19. Map your columns to the appropriate columns in the table

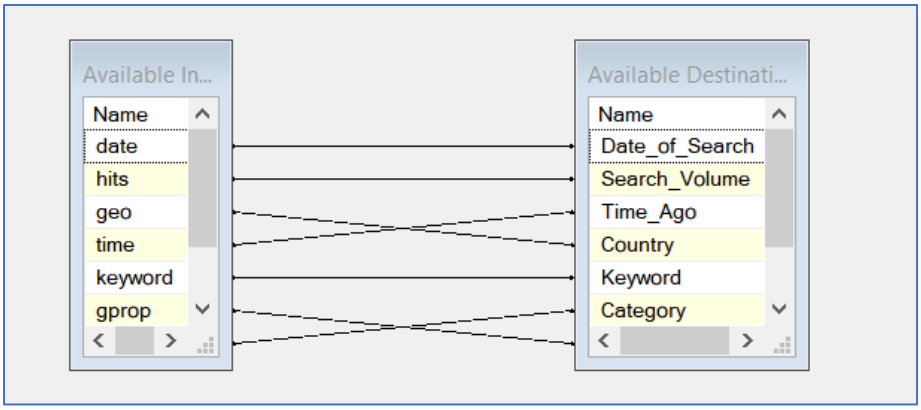

Figure 311: Mapping columns to SSMS table

20. Click the ‘Start’ button

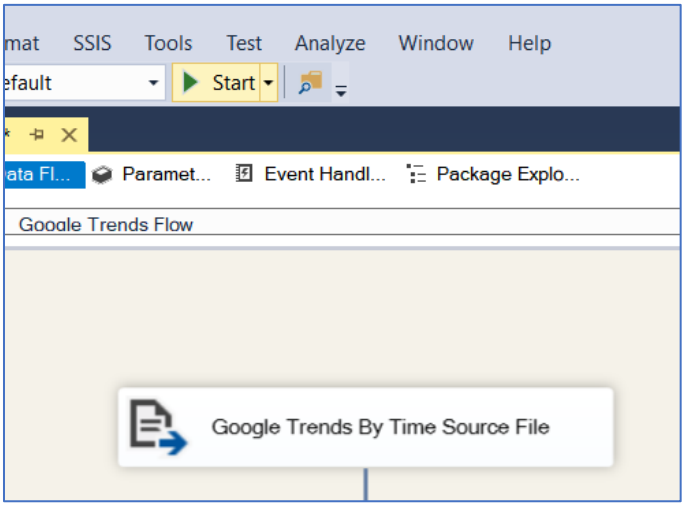

Figure 312: Starting data flow task

21. If the task is a success green ticks will appear and all of the rows of data will appear on the arrow

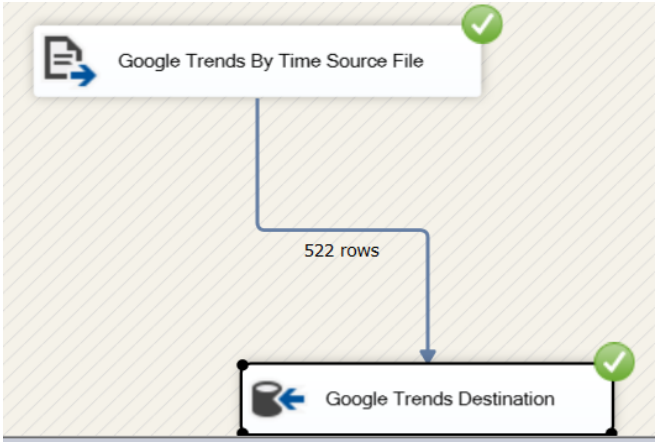

Figure 313: Successful data flow task

22. The steps are the same for the other data sources
23. One difference is in the Twitter data in the 'Configure the properties of each column' window in the File Manager select the columns and ensure that the 'OutputColumnWidth' option is high enough for your day v(this limits the number of characters that will be loaded)

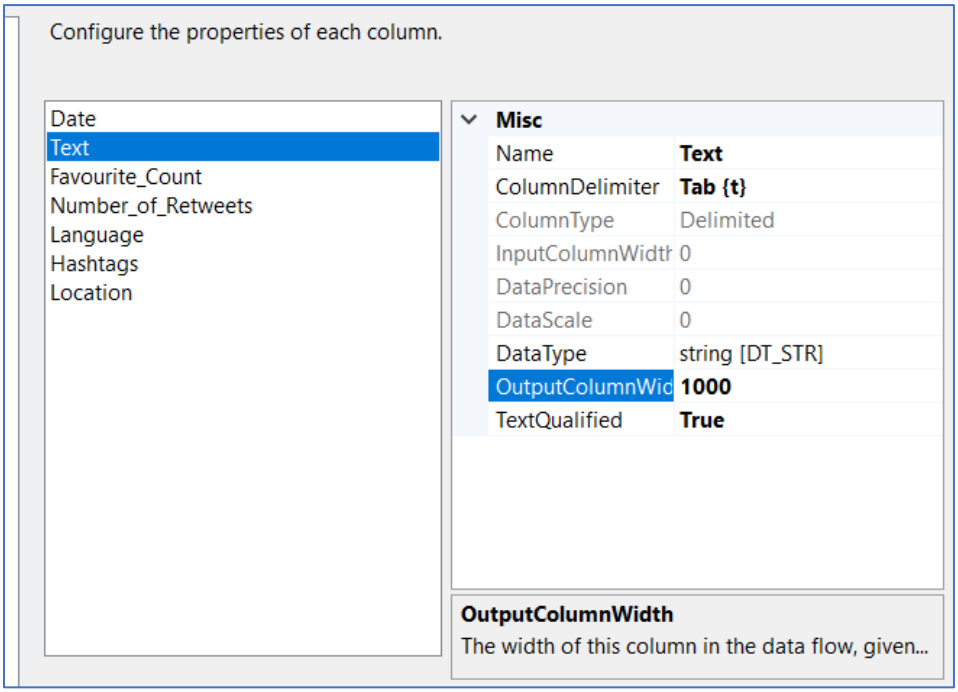

Figure 314: Output column width field

24. For the Twitter and NIS data drag and drop a 'Data Conversion' transformation task into the pane

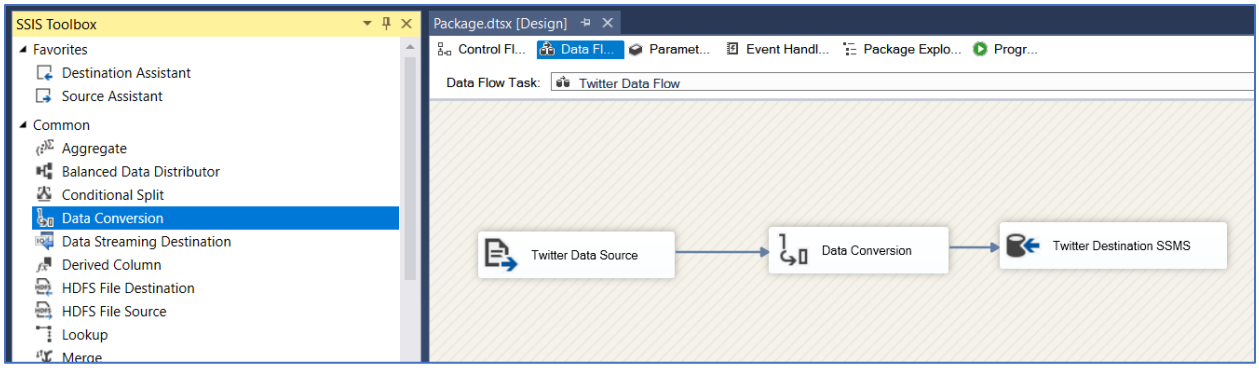

Figure 315: Data conversion task

25. Remove the column for 'Adequate Data' in the NIS data and convert the columns into the datatypes shown in figure 316

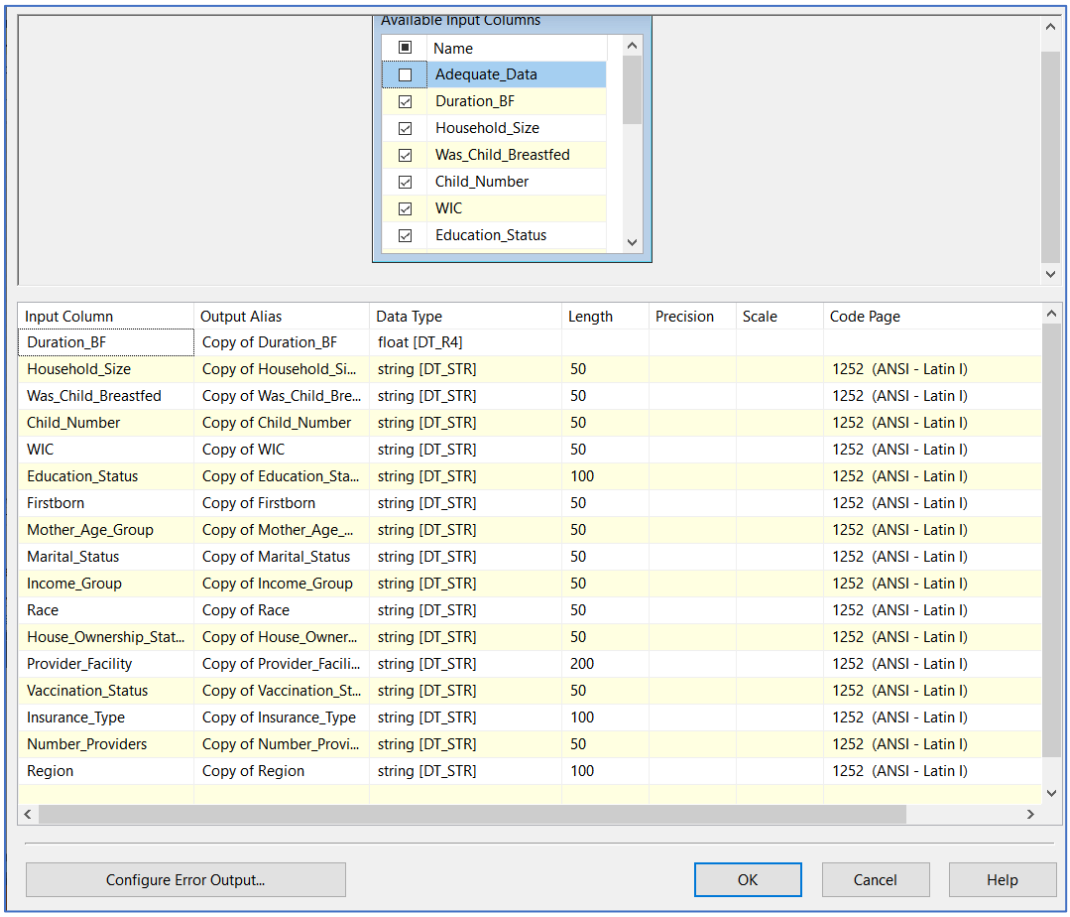

Figure 316: NIS data conversion

26. Convert the Twitter columns to the datatypes shown in figure 317

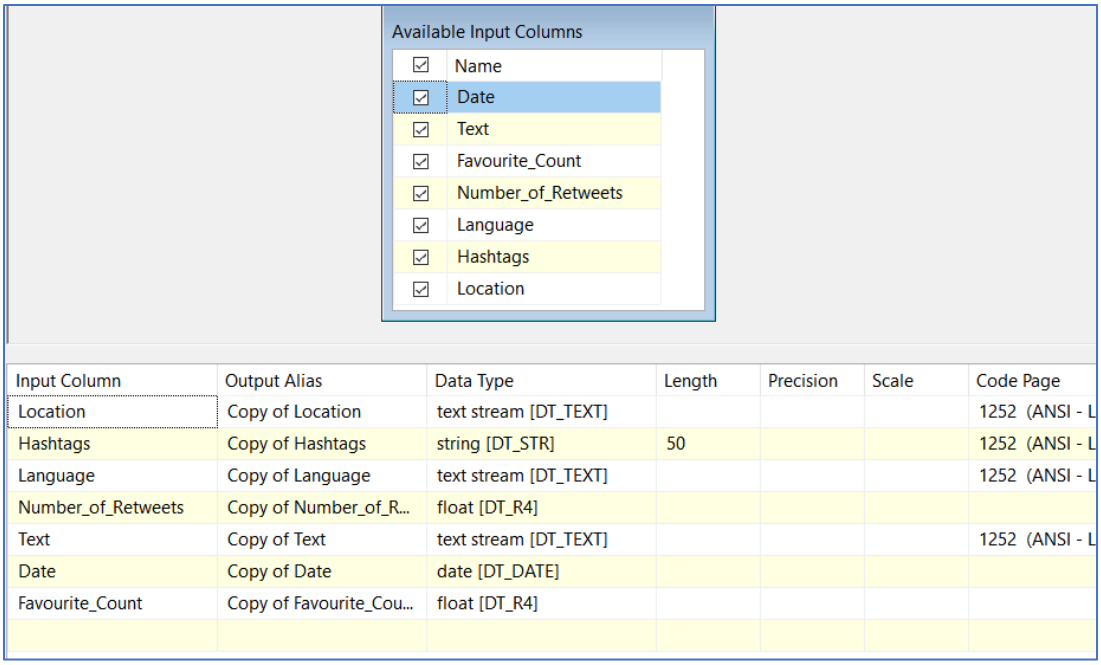

Figure 317: Twitter data conversion

Querying the SSMS Tables

1. Execute a query in SQL that generates a categorical column in the timeseries table for the rate of vaccination, with ‘high’, ‘medium’ and ‘low’ and order by this column

```
SELECT State, Year_2017 AS 'Vaccination Rate',
CASE
    WHEN Year_2017 >=80 THEN 'High'
    WHEN Year_2017 >=70 AND Year_2017 <80 THEN 'Medium'
    ELSE 'Low'
END As Rate
FROM Time_Series_Destination
ORDER BY 'Vaccination Rate' DESC;
```

Figure 318: SQl query of Timeseries table grouping vaccination rates

	State	Vaccination Rate	Rate
1	Massachusetts	84.20	High
2	North Dakota	83.10	High
3	Tennessee	82.30	High
4	New Hampshire	82.20	High
5	Virginia	81.00	High
6	Nebraska	80.70	High
7	Delaware	80.00	High
8	Connecticut	79.10	Medium
9	Florida	78.60	Medium
10	Rhode Island	78.60	Medium
11	Illinois	78.40	Medium
12	New Mexico	78.30	Medium
13	Maryland	78.30	Medium

Figure 319: SQL query putput with categorical column for vaccination rate

2. Alter the datatypes of the 'Table_of_Tweet' Location and Text_of columns to VARCHAR(MAX) datatypes

```
ALTER TABLE Table_of_Tweet
ALTER COLUMN Location VARCHAR(MAX);
```

Figure 320; Altering the data type of the Location column

```
ALTER TABLE Table_of_Tweet
ALTER COLUMN Text_of VARCHAR(MAX);
```

Figure 321: Altering the data type of the Text column

3. Query the number of tweets per location

```
select Location, COUNT(*) AS 'Number_of_Tweets'
FROM Table_of_Tweet
GROUP BY Location
ORDER BY Number_of_Tweets DESC;
```

Figure 322: Number of tweets by location

	Location	Number of Tweets
1	Amarillo	398
2	Lubbock	182
3	San Angelo	299
4	Seven Sisters	1400
5	Athens	6469
6	Houston	5507
7	Austin	3075

Figure 323: Number of tweets per location

4. Execute an SQL query to view the frequency of the word ‘autism’ grouped by the location based on code in https://stackoverflow.com/questions/881913/sql-server-function-for-displaying-word-frequency-in-a-column

```
SELECT Location, COUNT(*) Count
FROM Table_of_Tweet
CROSS APPLY (SELECT CAST('<a>'+REPLACE(Text_of,' ','</a><a>')+'</a>' AS xml) xml1 ) t1
CROSS APPLY (SELECT n.value('.','varchar(max)') AS word FROM xml1.nodes('a') x(n) ) t2
WHERE word LIKE 'autism'
GROUP BY Location
ORDER BY Count DESC
```

Figure 324: Viewing frequency of the word 'autism' by location

	Location	Count
1	Athens	22
2	Austin	9
3	Houston	8
4	Amarillo	4
5	Seven Sisters	3
6	San Angelo	2

Figure 325: Frequency of 'autism' by location in Twitter table

5. **Alter the data type of the Income to Poverty Raio column to decimal**

```
ALTER TABLE NISurvey
ALTER COLUMN Income_To_Poverty_Ratio DECIMAL(10,2);
```

Figure 326: Altering data type of income to poverty ratio

6. Query the average income to poverty ratio for each region from the NISurvey table

```
SELECT Region,
    CAST(ROUND(AVG(Income_To_Poverty_Ratio),2) AS DEC(10,2))
    Average_Income
FROM NISurvey
GROUP BY
    Region
ORDER BY
    Average_Income DESC;
```

Figure 327: Query for averge income to poverty ratio for each region

	Region	Average_Income
1	VIRGINIA	2.69
2	MASSACHUSETTS	2.58
3	MARYLAND	2.56
4	MINNESOTA	2.51
5	CALIFORNIA	2.43
6	COLORADO	2.39
7	HAWAII	2.38
8	NEW HAMPSHIRE	2.36
9	NEW JERSEY	2.35
10	WASHINGTON	2.35
11	ALASKA	2.35
12	DISTRICT OF COLUMBIA	2.33
13	CONNECTICUT	2.32

Figure 328: Average income to poverty ratio for each region

7. Execute a query for the proportion of vaccinated vs unvaccinated children grouped by education status in Texas

```
select Education_Status AS 'Education Status',
coalesce(count(case when Vaccination_Status = 'unvaccinated' then 1 end), 0) as Unvaccinated,
coalesce(count(case when Vaccination_Status = 'vaccinated' then 1 end), 0) as Vaccinated
from NISurvey
where Region = 'Texas'
group by Education_Status;
```

Figure 329: Querying the relationship between education status and vaccination status

	Education Status	Unvaccinated	Vaccinated
1	< 12 YEARS	106	335
2	12 YEARS	149	380
3	COLLEGE GRAD	216	887
4	"> 12 YEARS, NON-COLLEGE GRAD"	158	505

Figure 330: Query results of vaccination status grouped by education status

8. Query the Google trends table to view the average search volume for 'vaccine' grouped by year

```
select AVG(Search_Volume) as 'Average Search Volume',
datepart(yyyy, [Date_of_Search]) as [Year]
from Google_Trends_By_Time
group by datepart(yyyy, [Date_of_Search])
order by [Year]
```

Figure 331: Querying the average Google search volume per year

	Average Search Volume	year
1	19	2015
2	18	2016
3	19	2017
4	23	2018
5	26	2019
6	46	2020

Figure 332: Google search volume by year

References

Liske, D. (2018(a)) 'Tidy Sentiment Analysis in R'. Datacamp. Available at: https://www.datacamp.com/community/tutorials/sentiment-analysis-R

Liske, D. (2018(b)) 'Lyric Analysis with NLP & Machine Learning with R'. Datacamp. Available at: https://www.datacamp.com/community/tutorials/R-nlp-machine-learning

Tang, D. (2018) 'Visualising Google Trends results with R'. Dave Tang's Blog. Available at: https://davetang.org/muse/2018/12/31/visualising-google-trends-results-with-r/

Tejendra, S. (No Date) Multivariate TS Analysis, Available at: https://bookdown.org/singh_pratap_tejendra/intro_time_series_r/multivariate-ts-analysis.html. [Last accessed 7 August 2020]