Class 7 - Text Analysis & Potpourri

Text Analysis & R Potpourri

This class will be a bit more advanced because it's so specific. We'll end with some fun other things that R can do. Here we go!

regex, word stems, & working with text

`regex` stands for "regular expression." It's a standard way to parse text. It is also SUPER COMPLICATED. I have to Google it every time, hands-down. This is where you copy-and-paste a solution from StackOverflow!

```
#install.packages("stringr")
library(stringr)
library(titanic)
titanic_data <- titanic_test


##  Example 1 - which names had "Thomas" in it?
full_names <- titanic_data$Name
grep("Thomas", full_names, value = T)
```

```
##  [1] "Myles, Mr. Thomas Francis"
##  [2] "Jefferys, Mr. Clifford Thomas"
##  [3] "Franklin, Mr. Thomas Parham"
##  [4] "Davison, Mr. Thomas Henry"
##  [5] "Thomas, Mrs. Alexander (Thamine Thelma\")\""
##  [6] "Thomas, Mr. John"
##  [7] "Thomas, Mr. Charles P"
##  [8] "Storey, Mr. Thomas"
##  [9] "Andrew, Mr. Frank Thomas"
## [10] "Everett, Mr. Thomas James"
## [11] "McCaffry, Mr. Thomas Francis"
## [12] "Oxenham, Mr. Percy Thomas"
## [13] "Thomas, Mr. Tannous"
## [14] "Conlon, Mr. Thomas Henry"
```

```
## Example 2 - how would we just get the passenger's last names?
## the pattern is last name + comma + everything else -> so we'll take everything up to the comma
## https://stackoverflow.com/questions/2013124/regex-matching-up-to-the-first-occurrence-of-a-character
## https://stackoverflow.com/questions/2192316/extract-a-regular-expression-match
full_names[1]
```

```
## [1] "Kelly, Mr. James"
```

```
gsub(",.*","", full_names[1]) ## replace everything after the , with a blank string
```

```
## [1] "Kelly"
```

```
## Example 3 - a conditional column that flags if it has the names Thomas or James
titanic_data$T_J <- 0
titanic_data$T_J[grep("Thomas", titanic_data$Name)] <- 1
titanic_data$T_J[grep("James", titanic_data$Name)] <- 1
```

tf-idf

tf-idf stands for "term frequency, inverse document frequency" and it is a great tool to find words that differentiate groups of text. In short, it measures how frequent a term (word) is in each group and compares that to the frequency of that word across all groups. A word is ranked highly if it is common in one group but uncommon across all groups.

I have used this to find keywords from surveys/evaluations - what are Promoters saying that Detractors are not, and vice versa?

I followed this example - https://www.tidytextmining.com/tfidf.html

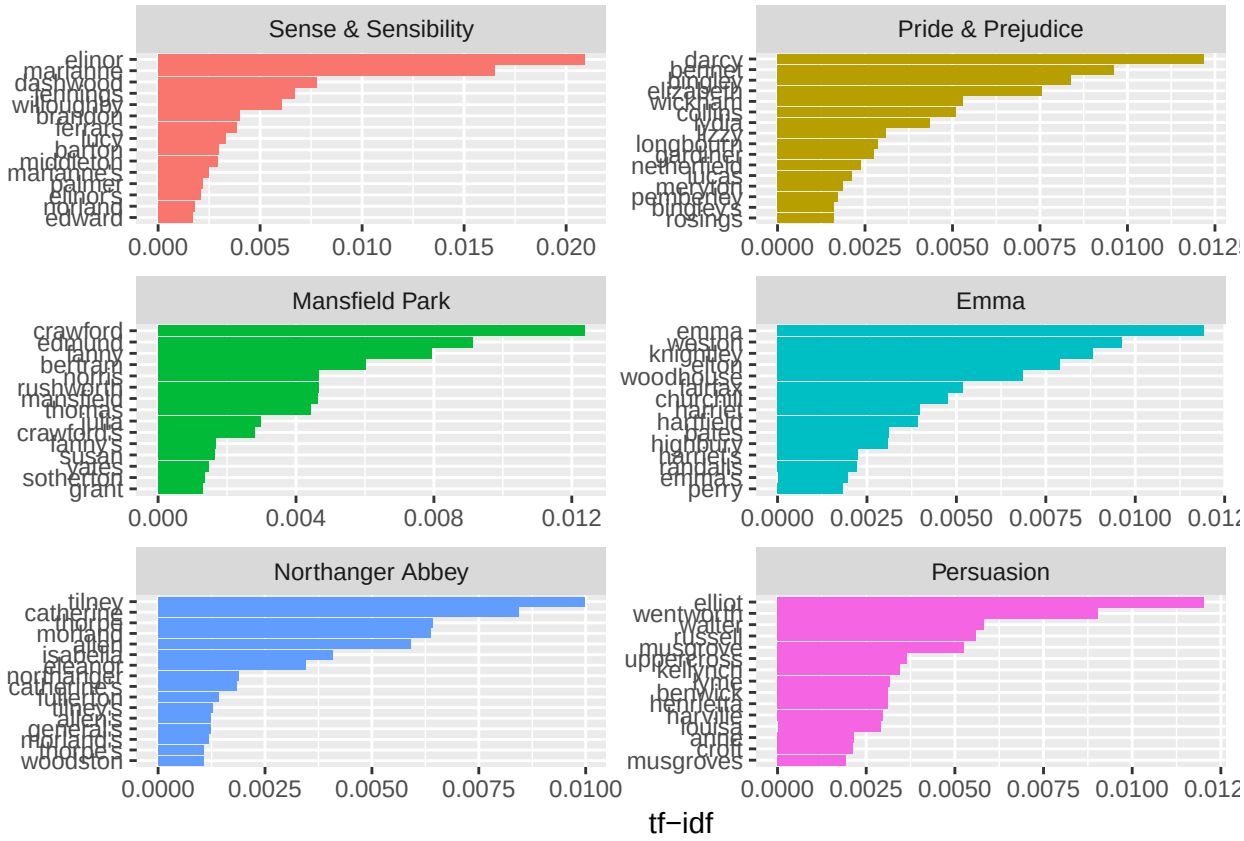
```
#install.packages("janeaustenr")
#install.packages("tidytext")
#install.packages("stopwords")
library(janeaustenr)
suppressMessages(library(dplyr))
library(tidytext)
library(stopwords)
library(ggplot2)
```

```
## Registered S3 methods overwritten by 'ggplot2':
##   method         from
##   [.quosures     rlang
##   c.quosures     rlang
##   print.quosures rlang
```

```
book_words <- austen_books() %>%
  unnest_tokens(word, text) %>%
  count(book, word, sort = TRUE)

## let's start by removing stop words from this list
book_words <- book_words[-which(book_words$word %in% stopwords()), ]

total_words <- book_words %>%
  group_by(book) %>%
  summarize(total = sum(n))

book_words <- left_join(book_words, total_words, by = "book")

## this is where the math happens!
book_words <- book_words %>%
  bind_tf_idf(word, book, n)

## let's plot the top words by each book
book_words %>%
  arrange(desc(tf_idf)) %>%
  mutate(word = factor(word, levels = rev(unique(word)))) %>%
```

```
  group_by(book) %>%
  top_n(15) %>%
  ungroup() %>%
  ggplot(aes(word, tf_idf, fill = book)) +
  geom_col(show.legend = FALSE) +
  labs(x = NULL, y = "tf-idf") +
  facet_wrap(~book, ncol = 2, scales = "free") +
  coord_flip()
```

```
## Selecting by tf_idf
```



Other cool things that R does!

Since this is our last class, here are some other fun things that R can do to inspire you.

Shiny

Shiny is R's web application framework. It takes some getting used to, but is very fun! Here's one example I did in the real world last year: https://kiratebbe.shinyapps.io/markov/

Here are some more examples: https://gallery.shinyapps.io/082-word-cloud/ https://github.com/rstudio/shiny-examples

```
library(ggplot2)

#install.packages("shiny")
library(shiny)

ui <- basicPage(
  plotOutput("plot1",
    click = "plot_click",
    dblclick = "plot_dblclick",
    hover = "plot_hover",
    brush = "plot_brush"
  ),
  verbatimTextOutput("info")
)

server <- function(input, output) {
  output$plot1 <- renderPlot({
    plot(mtcars$wt, mtcars$mpg)
  })

  output$info <- renderText({
    xy_str <- function(e) {
      if(is.null(e)) return("NULL\n")
      paste0("x=", round(e$x, 1), " y=", round(e$y, 1), "\n")
    }
    xy_range_str <- function(e) {
      if(is.null(e)) return("NULL\n")
      paste0("xmin=", round(e$xmin, 1), " xmax=", round(e$xmax, 1),
             " ymin=", round(e$ymin, 1), " ymax=", round(e$ymax, 1))
    }

    paste0(
      "click: ", xy_str(input$plot_click),
      "dblclick: ", xy_str(input$plot_dblclick),
      "hover: ", xy_str(input$plot_hover),
      "brush: ", xy_range_str(input$plot_brush)
    )
  })
}

shinyApp(ui, server)
```

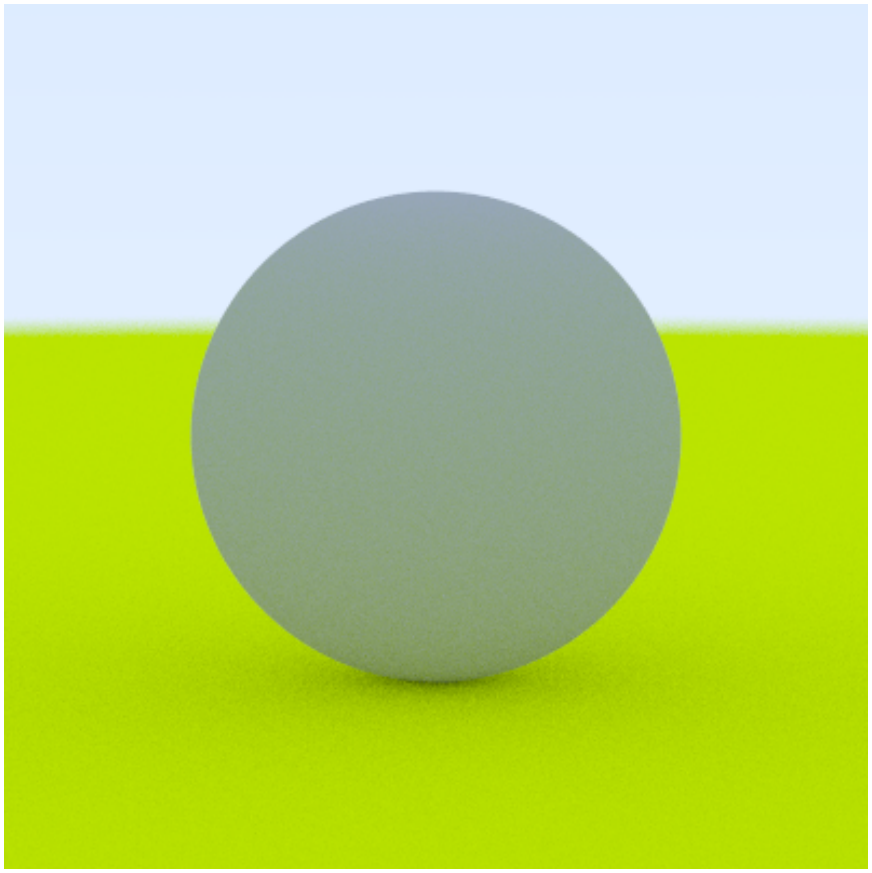
```
## PhantomJS not found. You can install it with webshot::install_phantomjs(). If it is installed, pleas
```

Shiny applications not supported in static R Markdown documents

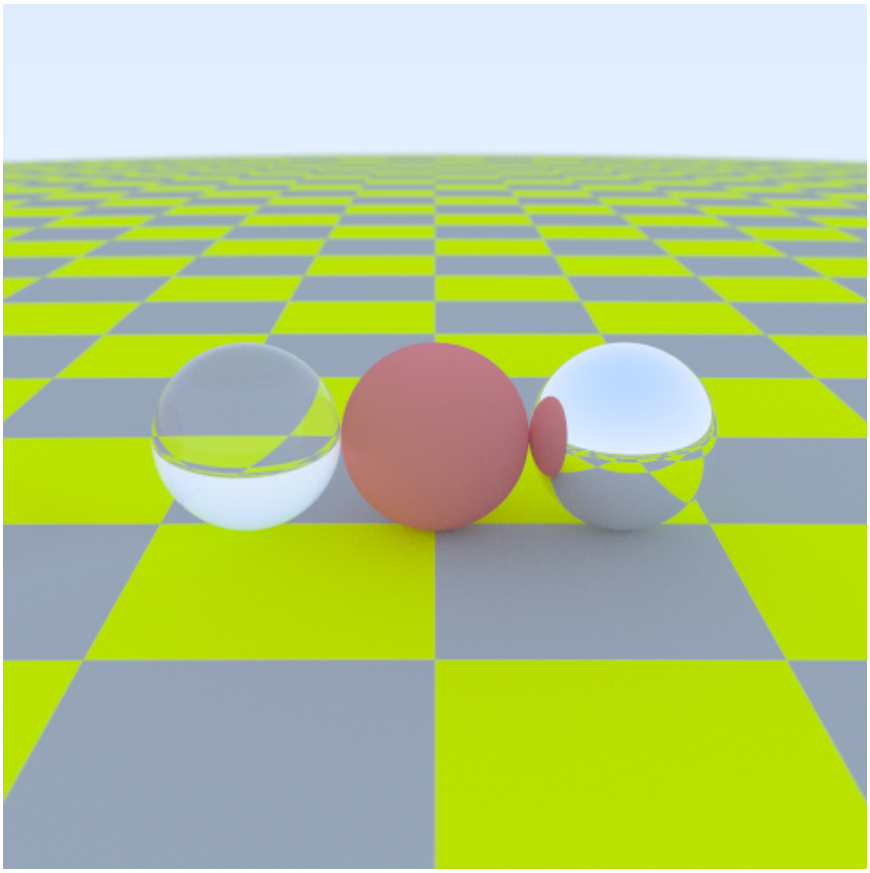
Rayrender

This package is new and renders 3D objects. I'm very new to it but am always impressed when I see examples!

```
#remotes::install_github("tylermorganwall/rayrender")
library(rayrender)

## a sphere
scene = sphere(y = -1001, radius = 1000,material = lambertian(color = "#ccff00")) %>%
  add_object(sphere(material = lambertian(color="grey50")))
render_scene(scene)
```



```
## three spheres! you can change the colors!
scene2 = sphere(y = -1001,radius = 1000,
                material = lambertian(color = "#ccff00",
                                      checkercolor="grey50")) %>%
  add_object(sphere(material = lambertian(color="#dd4444"))) %>%
  add_object(sphere(z=-2,material=metal())) %>%
  add_object(sphere(z=2,material=dielectric()))
render_scene(scene2, width=500, height=500, samples=500, fov=40, lookfrom=c(12, 4, 0))
```



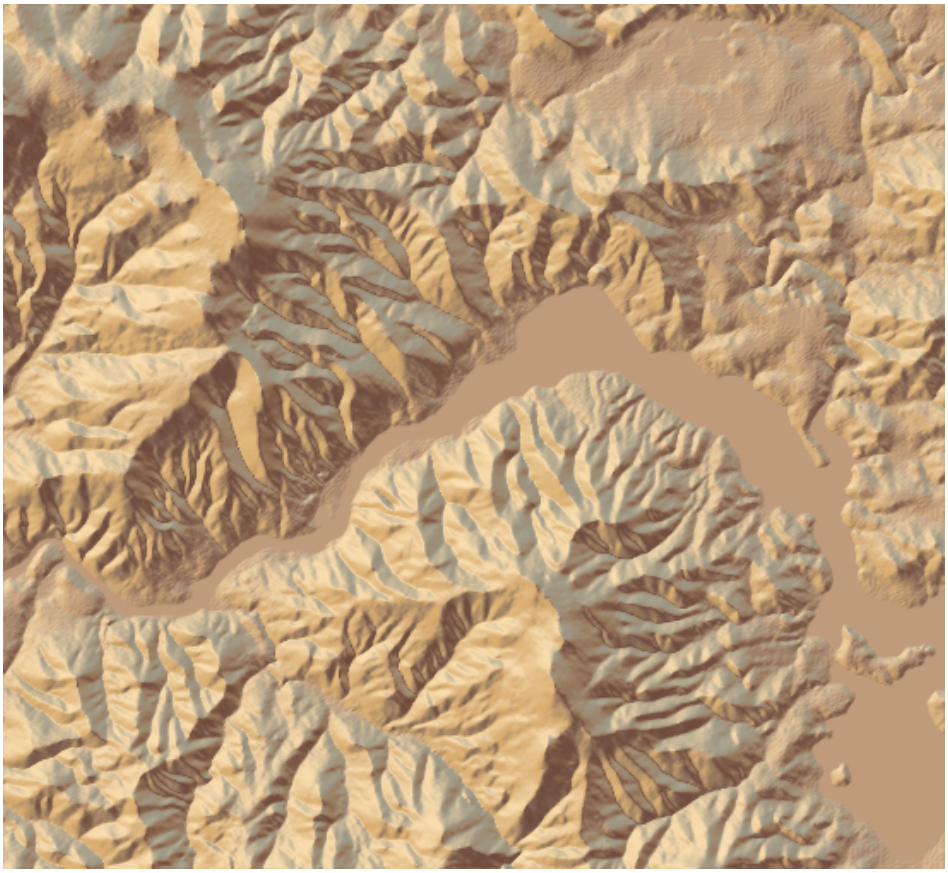
Rayshader

Even MORE crazy - you can make shaded, 3D, geographical maps!

Check out more information and examples here: https://github.com/tylermorganwall/rayshader

```
# install.packages("devtools")
# devtools::install_github("tylermorganwall/rayshader") ## takes a long time to install
# install.packages("rgdal")
suppressMessages(library(rgdal))
library(rayshader)

# loading in the data
loadzip = tempfile()
download.file("https://tylermw.com/data/dem_01.tif.zip", loadzip)
localtif = raster::raster(unzip(loadzip, "dem_01.tif"))
unlink(loadzip)

# conversion to matrix & adding shadows
elmat = matrix(raster::extract(localtif,raster::extent(localtif),buffer=1000),
               nrow=ncol(localtif),ncol=nrow(localtif))
raymat = ray_shade(elmat)
ambmat = ambient_shade(elmat)

# basic desert render
```

```
elmat %>%
  sphere_shade(texture = "desert") %>%
  plot_map()
```



```
# an interactive 3D map!
# try out different color palettes! "imhof1","imhof2","imhof3",imhof4","desert","bw", or "unicorn"
elmat %>%
  sphere_shade(texture = "desert") %>%
  add_water(detect_water(elmat), color="desert") %>%
  add_shadow(raymat,0.5) %>%
  add_shadow(ambmat,0.5) %>%
  plot_3d(elmat,zscale=10,fov=30,theta=-225,phi=25,windowsize=c(800,800),zoom=0.3)
```