

## Bigram Network Graphs

1. Create network graph of bigrams in Wells' novels. Do not include stop words. Make the links darker the more common the bigram is. Use arrows at the end of the line toward the second word. Colorize the central node. Show a chart and your code with line-by-line comments.

```
#Load packages and download data
#Load required Libraries
library(tidytext)
library(gutenbergr)
library(ggplot2)
library(dplyr)
library(tidyr)
library(tidyverse)
library(igraph)
library(ggraph)
library(stringr)

#Download HG Wells text data with book title as meta field----
hgwells <- gutenberg_download(c(35, 36, 5230, 159), meta_fields = "title")
```

### 1.1 Bigram counts

```
#Extract the bigrams with stop words
bigrams <- hgwells %>%
  unnest_tokens(bigram, text, token = "ngrams", n = 2)

#Separate bigrams for excluding stop words
separatedBigrams <- bigrams %>%
  separate(bigram, c("word1", "word2"), sep = " ")

#Filter and remove stop words
filteredBigrams <- separatedBigrams %>%
  filter(!word1 %in% stop_words$word) %>%
  filter(!word2 %in% stop_words$word)

# Count clean bigrams excluding NA:
counts <- filteredBigrams %>%
  na.omit() %>%
  count(word1, word2, sort = TRUE)
#print the top 10 rows
head(counts, 10)

## # A tibble: 10 x 3
##   word1 word2      n
##   <chr> <chr>   <int>
```

```
## 1 time traveller 54
## 2 beast people 44
## 3 heat ray 35
## 4 time machine 33
## 5 red weed 25
## 6 beast folk 24
## 7 black smoke 23
## 8 ulla ulla 20
## 9 hyena swine 17
## 10 thomas marvel 17
```

## 1.2 Bigram network graph

*# Create the igraph object from the counts data frame*

```
bigramNetwork <- counts %>%
  #Filter bigrams with counts greater than 5
  filter(n > 5) %>%
  graph_from_data_frame()
```

```
bigramNetwork
```

```
## IGRAPH d1cdd64 DN-- 113 71 --
## + attr: name (v/c), n (e/n)
## + edges from d1cdd64 (vertex names):
## [1] time ->traveller beast ->people heat ->ray
## [4] time ->machine red ->weed beast ->folk
## [7] black ->smoke ulla ->ulla hyena ->swine
## [10] thomas ->marvel dr ->kemp handling->machine
## [13] hundred ->yards looked ->round white ->haired
## [16] blue ->sky front ->door pine ->trees
## [19] blood ->stained dressing->gown sand ->pits
## [22] teddy ->henfrey fighting->machine half ->past
## + ... omitted several edges
```

*#Convert to plot using ggraph*

```
set.seed(2021)
```

*#set the Directionality element*

```
pointer <- grid::arrow(type = "closed", length = unit(.1, "inches"))
```

```
ggraph(bigramNetwork, layout = "fr") +
```

*#Use edge\_alpha aesthetic to make links transparent based on how common or rare the bigram is.*

*#end\_cap tells the arrow where to stop*

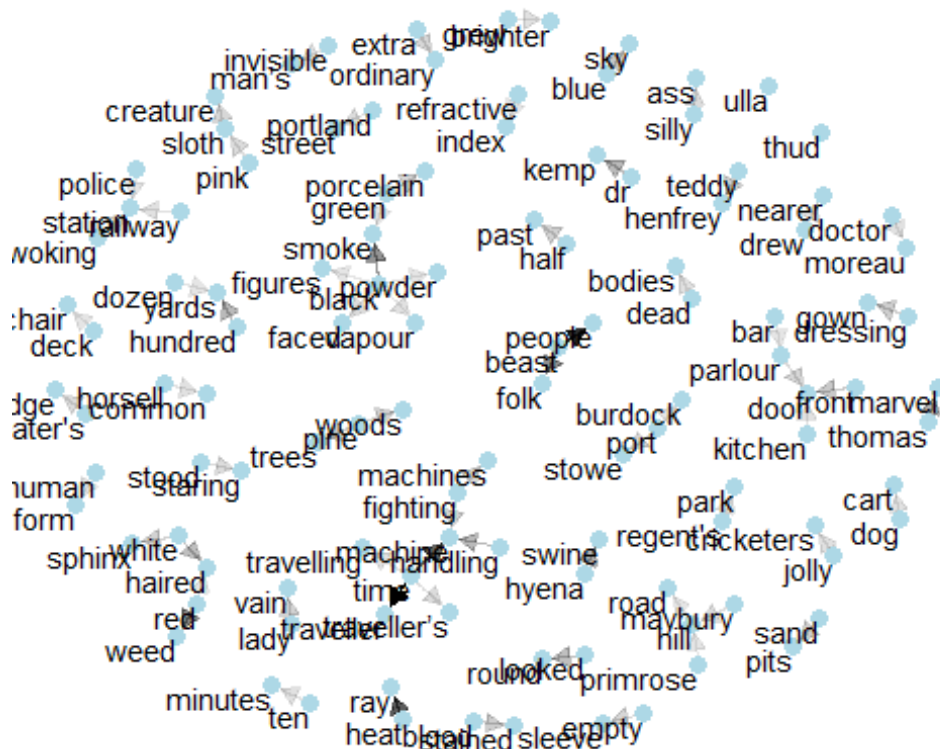
```
geom_edge_link(aes(edge_alpha = n), show.legend = FALSE,
  arrow = pointer, end_cap = circle(.05, 'inches')) +
```

*#Colorize nodes and set node size*

```
geom_node_point(color = "lightblue", size = 3) +
```

*#Anotate the nodes with the names*

```
geom_node_text(aes(label = name), vjust = 1, hjust = 1) +  
#Add theme to the plot for better visualization  
theme_void()
```



*Commentary:*

The graph shows a network of most common bigrams in Well's Novels text. Each arrow connects an origin node to a destination node for the bigrams present in the text. A darker arrow means that the bigrams connected are most common. In our example, *Time traveller*, *Time machine*, *Beast people* are some of the most common bigrams in the text.

2. Create a `count_bigrams` function to reuse for counting bigrams in other texts. Comment your code line by line.

```
#Count bigrams function
count_bigrams <- function(dataset){
  #Extract the bigrams with stop words
  bigrams <- dataset %>%
    unnest_tokens(bigram, text, token = "ngrams", n = 2)
  #Separate bigrams for excluding stop words
  bigrams %>%
    separate(bigram, c("word1", "word2"), sep = " ") %>%
  #Filter and remove stop words
    filter(!word1 %in% stop_words$word) %>%
    filter(!word2 %in% stop_words$word)%>%
  # Count clean bigrams excluding NA:
```

```

na.omit() %>%
count(word1, word2, sort = TRUE)
}

```

3. Create a `visualize_bigrams` function to reuse for visualizing network graphs of other texts. Comment your code line by line. Show an example network graph.

```

#Visualize bigrams function
visualize_bigrams <- function(datasetBigrams){
  set.seed(2021)
  #set the Directionality element
  pointer <- grid::arrow(type = "closed", length = unit(.1, "inches"))

  datasetBigrams %>%
    graph_from_data_frame() %>%
    ggraph(layout = "fr") +
      #Use edge_alpha aesthetic to make links transparent based on how common
      #or rare the bigram is.
      #end_cap tells the arrow where to stop
      geom_edge_link(aes(edge_alpha = n), show.legend = FALSE,
                     arrow = pointer, end_cap = circle(.05, 'inches')) +
      #Colorize nodes and set node size
      geom_node_point(color = "lightblue", size = 3) +
      #Anotate the nodes with the names
      geom_node_text(aes(label = name), vjust = 1, hjust = 1) +
      #Add theme to the plot for better visualization
      theme_void()
}

```

Example network graph

```

#Testing the functions using kjv bible text

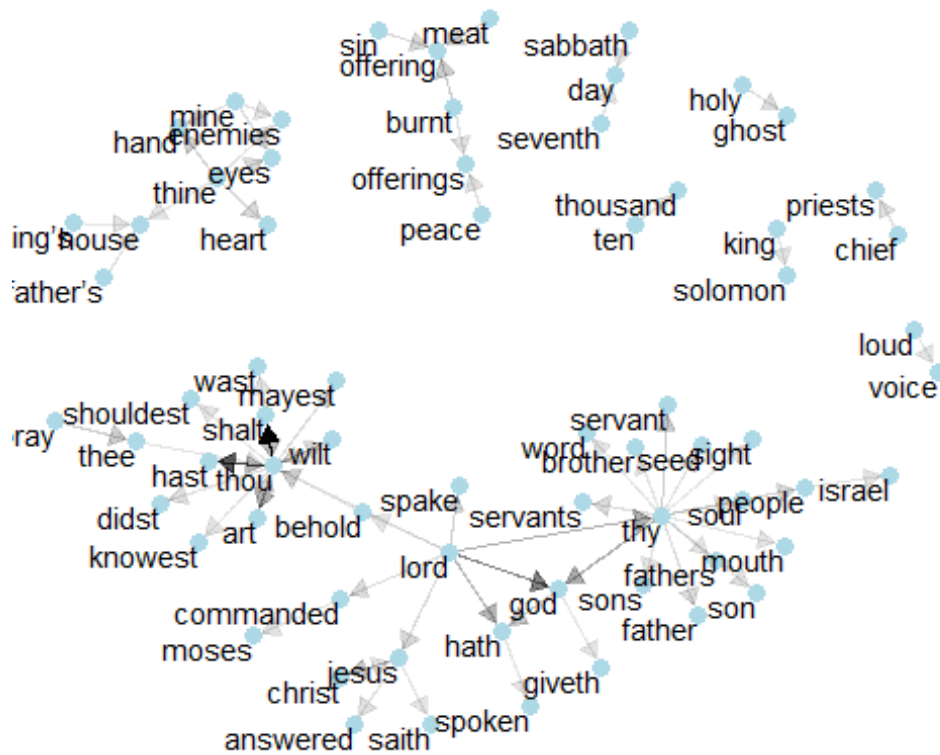
# the King James version is book 10 on Project Gutenberg:
library(gutenbergr)
kjbv <- gutenbergr::download(10)

#kjbv bigrams
kjbvBigrams <- kjbv %>%
  count_bigrams()

# filter out rare combinations, as well as digits
kjbvBigrams <- kjbv %>%
  count_bigrams()

```

```
kjvBigrams %>%
  filter(n > 40,
    !str_detect(word1, "\\d"),
    !str_detect(word2, "\\d")) %>%
  visualize_bigrams()
```



#### Commentary:

The graph shows a network of most common bigrams in kjv Bible text. Each arrow connects an origin node to a destination node for the bigrams present in the text. A darker arrow means that the bigrams connected are most common. In our example, *thou shalt* and *Thou hast* are some of the most common bigrams in the text. We can also observe that *Thou* and *Thy* are the most common central nodes in the text.