Analyzing Phrases

As always, we set the working directory and then load the packages we want.

```
library(readr)
library(tidyr)
library(tidytext)
library(okcupiddata)
library(ggplot2)
library(dplyr)
library(stringr)
library(scales) # scale functions for visualization
library(igraph)
library(ggraph)
library(reshape2)
library(magrittr)
```

Next let's follow the examples from chapter 4 and extract bigrams from the very same data. In the next chunk, we extract bigrams and inspect them as lists. Again, we continue to look only at the essays prepared by smokers.

For clarity, let's break the process down into steps. First, subset the okcupid `profiles` to include just the essay responses and the user's response about smoking. The `data_frame` function creates a tibble. In creating the tibble, we rename `essay0` as `text` -- not necessary, but follows the example in Silge and Robinson.

Then we create a new binary variable, `smoker` that identifies non-smokers as "no" and everyone else as "yes", dropping the original `smokes` variable (notice the command `select(-smokes)`, which selects all columns *except for* `smokes`.

```
tidy_okcupid <- select_(profiles,"essay0","smokes")

tidy_okcupid <- data_frame(smokes=profiles$smokes,text=profiles$essay0)
tidy_okcupid <- tidy_okcupid %>%
     mutate(smoker = ifelse(smokes=="no","no","yes")) %>%
     select(-smokes)
```

The next few lines of code *remove* all `NA` rows, because the `unnest_tokens` function requires complete cases. The last line in this chunk unnests 2-word phrases from the . You may also want to experiment with longer phrases.

```
tidy_smoker <- tidy_okcupid %>%
     select(smoker, text) %>%
     na.omit()

smoker_bigrams <- unnest_tokens(tidy_smoker, bigram, text, token="ngrams",
n=2)
```

```
smoker_bigrams
```

```
## # A tibble: 1,226,066 × 2
##    smoker     bigram
##     <chr>      <chr>
## 1      no       i am
## 2      no       am a
## 3      no     a chef
## 4      no  chef this
## 5      no    this is
## 6      no    is what
## 7      no  what that
## 8      no that means
## 9      no    means 1
## 10     no        1 i
## # ... with 1,226,056 more rows
```

At this point, the tibble `smoker_bigrams` consists of more than 1,226,000 pairs of adjacent words. The next code chunk counts and sorts the pairs to find the most common ones. After that, it `separates` the pairs into `word1` and `word2` to search for and remove stopwords, and then `unites` the pairs that remain, finally recounting and sorting the shorter list of pairs.

Consult Silge & Robinson to learn how to customize the list of stopwords.

```
smoker_bigrams %>%
     count(bigram, sort = TRUE)
```

```
## # A tibble: 323,132 × 2
##           bigram     n
##            <chr> <int>
## 1           i am 14243
## 2          i'm a  6571
## 3         i love  6445
## 4         in the  5275
## 5         i like  4480
## 6         i have  4346
## 7           am a  4215
## 8          and i  3382
## 9  san francisco  3269
## 10       like to  3128
## # ... with 323,122 more rows
```

```
bigrams_separated <- smoker_bigrams %>%
     separate(bigram, c("word1", "word2"), sep = " ")

bigrams_filtered <- bigrams_separated %>%
     filter(!word1 %in% stop_words$word) %>%
     filter(!word2 %in% stop_words$word)

# new bigram counts:
```

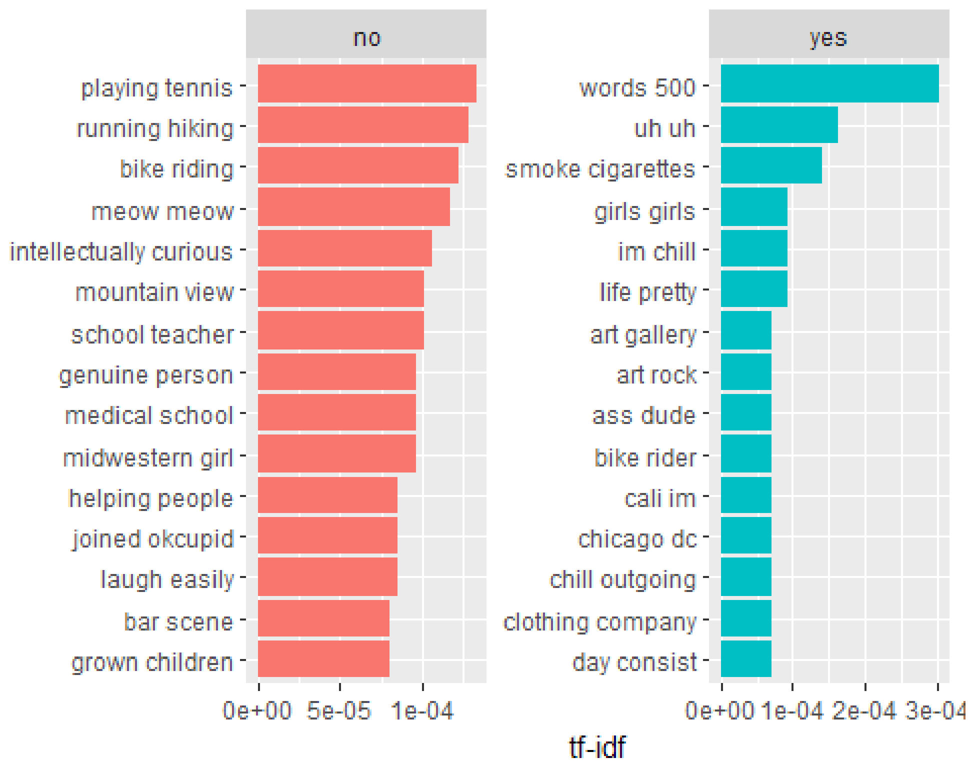
```
bigram_counts <- bigrams_filtered %>%
     count(word1, word2, smoker, sort = TRUE)


bigram_counts

## Source: local data frame [105,832 x 4]
## Groups: word1, word2 [99,982]
##
##        word1     word2 smoker     n
##        <chr>     <chr>  <chr> <int>
## 1        san francisco     no  2669
## 2       east     coast     no   750
## 3        san francisco    yes   600
## 4        fun    loving     no   537
## 5   recently     moved     no   449
## 6       east       bay     no   254
## 7     online    dating     no   229
## 8      enjoy      life     no   214
## 9       grad    school     no   208
## 10       san     diego     no   187
## # ... with 105,822 more rows

bigrams_united <- bigrams_filtered %>%
     unite(bigram, word1, word2, sep = " ")

bigrams_united

## # A tibble: 159,694 × 2
##    smoker                   bigram
## *   <chr>                    <chr>
## 1      no                  means 1
## 2      no             workaholic 2
## 3      no           writing public
## 4      no              public text
## 5      no            online dating
## 6      no              dating site
## 7      no               site makes
## 8      no pleasantly uncomfortable
## 9      no               school hey
## 10     no         australian living
## # ... with 159,684 more rows
```

As noted in class and in the readings, a simple frequency count can be misleading. The next code chunk weights term frequency by the inverse frequency within the documents (smokers vs non-smokers.

```
     bigram_tf_idf <- bigrams_united %>%
          count(smoker, bigram) %>%
          bind_tf_idf(bigram, smoker, n) %>%
```

```r
        arrange(desc(tf_idf))

    bigram_tf_idf

## Source: local data frame [105,832 x 6]
## Groups: smoker [2]
##
##    smoker                 bigram     n           tf       idf       tf_idf
##     <chr>                  <chr> <int>        <dbl>     <dbl>        <dbl>
## 1     yes              words 500    13 0.0004362270 0.6931472 0.0003023695
## 2     yes                  uh uh     7 0.0002348914 0.6931472 0.0001628143
## 3     yes       smoke cigarettes     6 0.0002013355 0.6931472 0.0001395552
## 4      no         playing tennis    25 0.0001924661 0.6931472 0.0001334073
## 5      no         running hiking    24 0.0001847675 0.6931472 0.0001280710
## 6      no            bike riding    23 0.0001770688 0.6931472 0.0001227348
## 7      no              meow meow    22 0.0001693702 0.6931472 0.0001173985
## 8      no intellectually curious    20 0.0001539729 0.6931472 0.0001067259
## 9      no          mountain view    19 0.0001462742 0.6931472 0.0001013896
## 10     no         school teacher    19 0.0001462742 0.6931472 0.0001013896
## # ... with 105,822 more rows

    bigrams_separated %>%
        filter(word1 == "not") %>%
        count(word1, word2, sort = TRUE)

## Source: local data frame [995 x 3]
## Groups: word1 [1]
##
##    word1   word2     n
##    <chr>   <chr> <int>
## 1    not       a   365
## 2    not      to   227
## 3    not  really   213
## 4    not    sure   196
## 5    not looking   165
## 6    not     the   146
## 7    not      so   111
## 8    not      in    97
## 9    not    very    96
## 10   not     too    84
## # ... with 985 more rows

plot_pairs <- bigram_tf_idf %>%
  arrange(desc(tf_idf)) %>%
  mutate(pair = factor(bigram, levels = rev(unique(bigram))))

plot_pairs %>%
  group_by(smoker) %>%
  top_n(15) %>%
  ungroup %>%
  ggplot(aes(pair, tf_idf, fill = smoker)) +
```

```
  geom_col(show.legend = FALSE) +
  labs(x = NULL, y = "tf-idf") +
  facet_wrap(~smoker, ncol = 2, scales = "free") +
  coord_flip()

## Selecting by pair
```

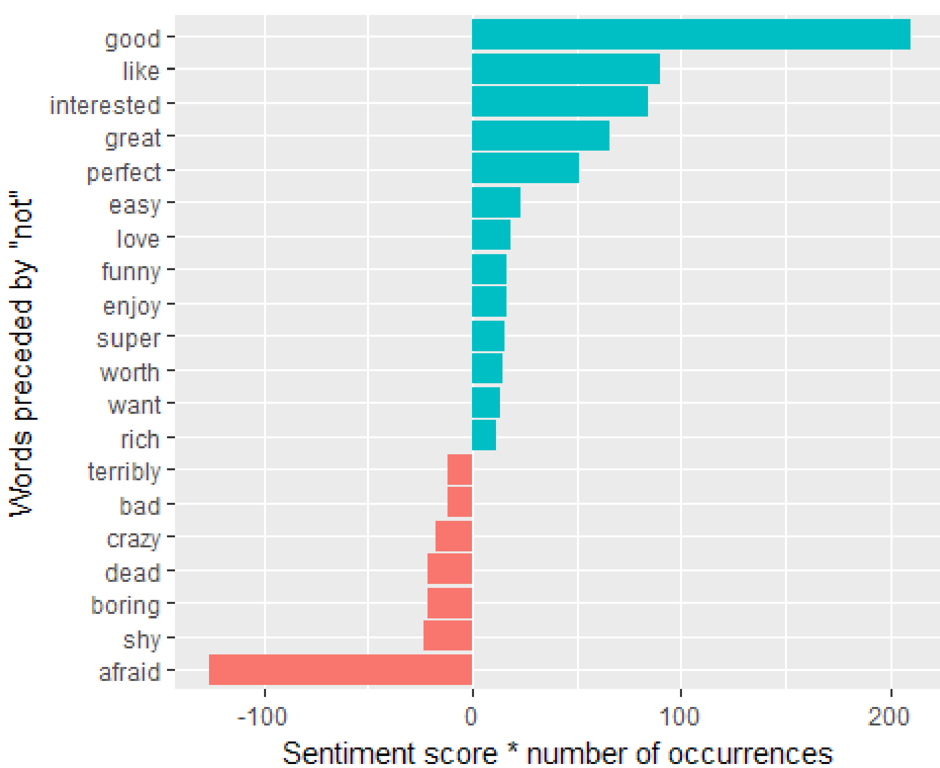


This next chunk is not required for your Project, but follows the Silge & Robinson example. We specifically look for word pairs beginning that include negation (like "not"), since that can be a common style.

```
     AFINN <- get_sentiments("afinn")

     not_words <- bigrams_separated %>%
          filter(word1 == "not") %>%
          inner_join(AFINN, by = c(word2 = "word")) %>%
          count(word2, score, sort = TRUE) %>%
          ungroup()

     not_words

## # A tibble: 168 × 3
##         word2 score     n
##         <chr> <int> <int>
## 1        good     3    70
## 2      afraid    -2    63
## 3        like     2    45
```

```
## 4  interested      2     42
## 5        easy      1     23
## 6         shy     -1     23
## 7       great      3     22
## 8     perfect      3     17
## 9        want      1     13
## 10      crazy     -2      9
## # ... with 158 more rows
```

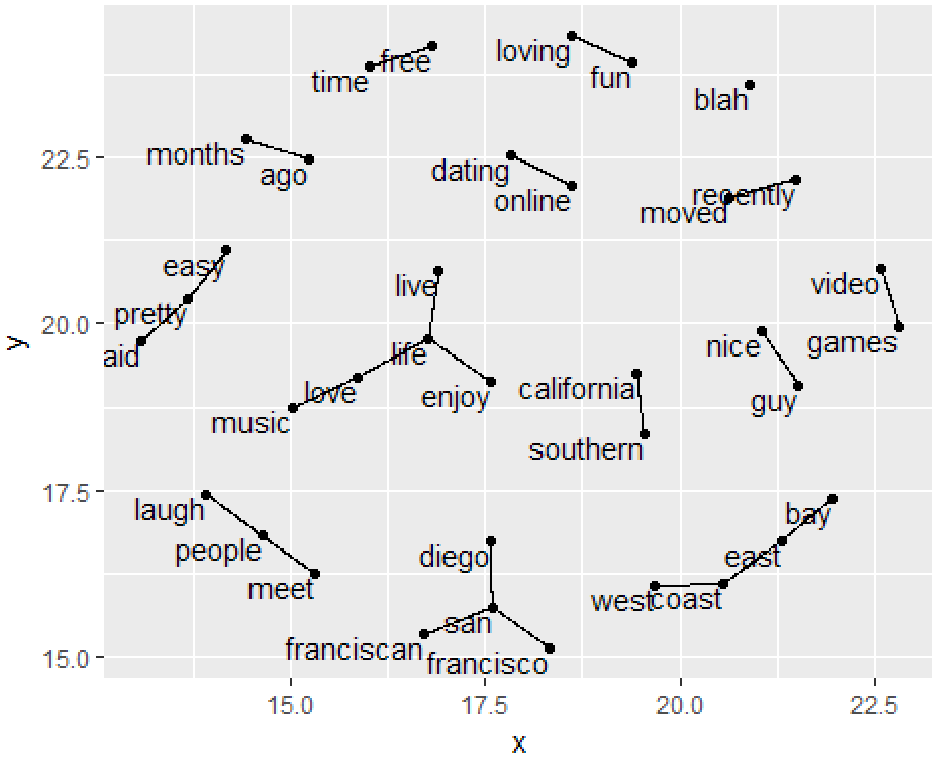
```
not_words %>%
    mutate(contribution = n * score) %>%
    arrange(desc(abs(contribution))) %>%
    head(20) %>%
    mutate(word2 = reorder(word2, contribution)) %>%
    ggplot(aes(word2, n * score, fill = n * score > 0)) +
    geom_col(show.legend = FALSE) +
    xlab("Words preceded by \"not\"") +
    ylab("Sentiment score * number of occurrences") +
    coord_flip()
```




```
negation_words <- c("not", "no", "never", "without")

negated_words <- bigrams_separated %>%
    filter(word1 %in% negation_words) %>%
    inner_join(AFINN, by = c(word2 = "word")) %>%
    count(word1, word2, score, sort = TRUE) %>%
    ungroup()
```

At this point we begin to make network graphs, first showing the cost common word combinations for *non-smokers. You could then repeat and modify the chunk for smokers*.

```
     # original counts
     bigram_counts

## Source: local data frame [105,832 x 4]
## Groups: word1, word2 [99,982]
##
##        word1     word2 smoker     n
##        <chr>     <chr>  <chr> <int>
## 1        san francisco     no  2669
## 2       east     coast     no   750
## 3        san francisco    yes   600
## 4        fun    loving     no   537
## 5   recently     moved     no   449
## 6       east       bay     no   254
## 7     online    dating     no   229
## 8      enjoy      life     no   214
## 9       grad    school     no   208
## 10       san     diego     no   187
## # ... with 105,822 more rows

# filter for only relatively common combinations
# might want to experiment with values other than 25 in filter line
     bigram_graph <- bigram_counts %>%
          group_by(smoker) %>%
          filter(n > 25) %>%
          filter(smoker=="yes") %>%
          graph_from_data_frame()

     bigram_graph

## IGRAPH DN-- 36 23 --
## + attr: name (v/c), smoker (e/c), n (e/n)
## + edges (vertex names):
##  [1] san     ->francisco  recently->moved      fun     ->loving
##  [4] east    ->coast      love    ->music      east    ->bay
##  [7] online  ->dating     san     ->diego      months  ->ago
## [10] blah    ->blah       people  ->laugh      pretty  ->laid
## [13] meet    ->people     nice    ->guy        pretty  ->easy
## [16] video   ->games      enjoy   ->life       live    ->life
## [19] free    ->time       west    ->coast      love    ->life
## [22] san     ->franciscan southern->california

     set.seed(2017)

     ggraph(bigram_graph, layout = "fr") +
          geom_edge_link() +
          geom_node_point() +
          geom_node_text(aes(label = name),  vjust = 1, hjust = 1)
```



Finally, as in Chapter 4 of Silge and Robinson, here's an alternative and more attractive network graph. This will use the same subset of participants as the one created above.

```
set.seed(2016)

a <- grid::arrow(type = "closed", length = unit(.05, "inches"))

ggraph(bigram_graph, layout = "fr") +
     geom_edge_link(aes(edge_alpha = n), show.legend = FALSE,
                    arrow = a, end_cap = circle(.05, 'inches')) +
     geom_node_point(color = "lightblue", size = 5) +
     geom_node_text(aes(label = name), vjust = 1, hjust = 1) +
     theme_void()
```


music
love
live
life
enjoy
games
video
guy
nice
meet
people
laugh
recently
moved
ego
franciscan
san
cisco
blah
bay
east
coast
west
laid
pretty
easy
months
ago
online
dating
southern
california
loving
fun
time
free