# Sentiment Analysis

*Dean Hope Robertson*

*9/18/2018*

# Section 4: Sentiment Analysis

**This section looks various sentiment analysis techniques and how to adapt them to the historical South African presidential SONA speeches. The markdown will cover the following sentiment analysis techniques:**

**The first step is to load the data ('speeches.csv' file) from the shared GitHub repository, which contains the pre-processed text which has been divided into separated sentences per president.**

```
raw_data <- read.csv("https://raw.githubusercontent.com/James-Leslie/president-speech-classifier/master/data/speeches.csv")
raw_data = as.tibble(raw_data)
```

# Tokenization

**Once the data has been read in, it is possible to apply tokenization. Tokenization is a technique, which allows text to be broken up into 'tokens' or any unit of text, which is necessary for analysis. Tokens range from words, sentences, paragraphs or even chapters depending on the user's preferences. This is done using the unnest_tokens() function in the tidytext package. If we tokenize the text into words, it is then possible to determine the average word count per president. As seen below, Mbeki has the highest number of words per speech out of the 6 presidents in this assignment.**

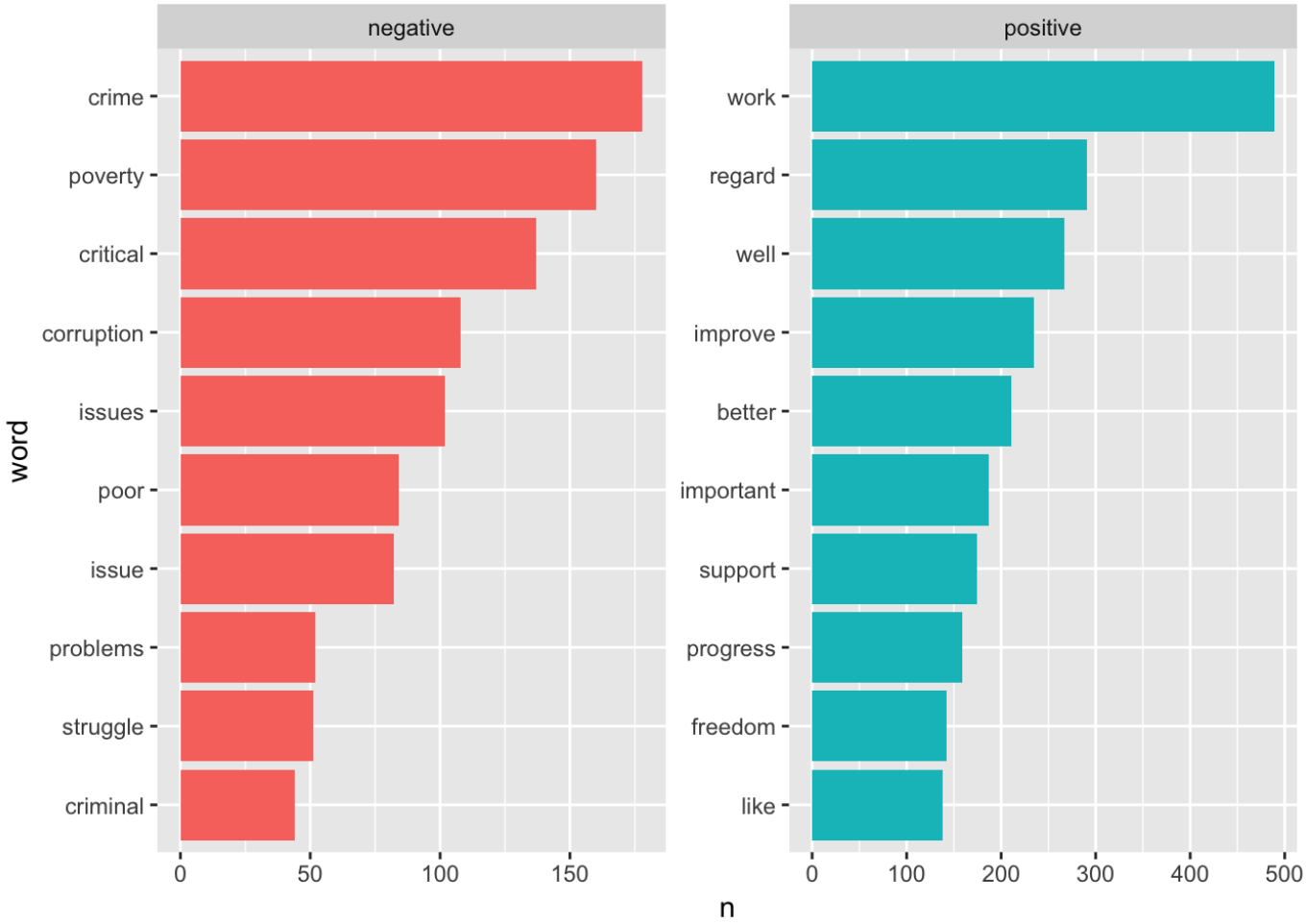
# Sentiment Analyis

# Lexicons

**A sentiment lexicon is a dictionary of words, which have been scored in various ways to provide some insight into the emotional content of a word. Depending on the lexicon, words can be scored in a binary fashion (positive/negative), numerical or even branch up to a higher nuanced emotional category. There are three lexicons covered in this assignment namely:**

**We are able to map the 'Bing' lexicon using an inner_join to the SONA speeches to review what the 'top 10' most frequently used positive and negative words in SONA history are:**

```
#apply sentiment 'bing' lexicon
  data %>%inner_join(get_sentiments('bing')) %>%
  count(word, sentiment) %>%
  group_by(sentiment) %>%
  top_n(10) %>%
  ungroup() %>%
  mutate(word = reorder(word, n)) %>% # Make word a factor in order of n

#Plot the Top 10 Positive and Negative words Across ALL Speeches
ggplot(aes(word,n, fill = sentiment)) +
  # Make a bar chart with geom_col()
  geom_col(show.legend = FALSE) +
  facet_wrap(~sentiment, scales = "free") +
  coord_flip()
```




**As seen it above, there is no need to manually remove *'stop words'* seeing as the lexicons do not contain any *neutral* words and the inner_join removes the stop words from the original text. The speeches are viewed as a combination of individual words and each word has a sentiment or emotional connotation attached to it. Here we can see that negative words such as *'crime, poverty and corruption'* and *'work, regard and improve'* are the most frequently used negative and positive words, respectively.**

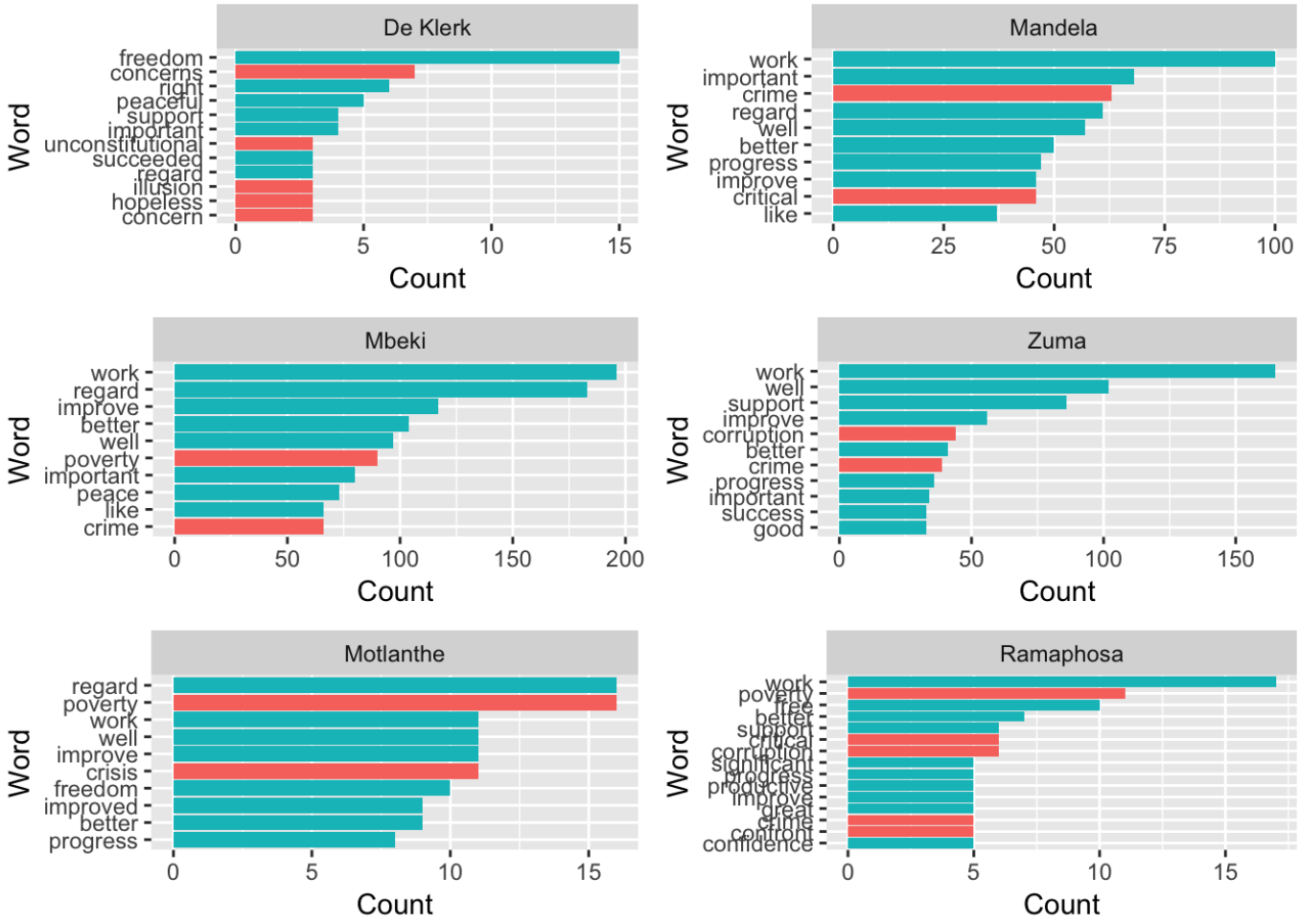
**The sentiment of the entire speech can be calculated in a number of ways; the most common being *net sentiment* of the sum of individual words in the text. This is the most basic and consequently the most commonly use approach to sentiment analysis. All the data manipulation is conducted using the tidy tool ecosystem which allows the user to apply quick calculations to the *tibble* data effeciently and effectively.**

**As seen below the overall sentiment for each speech was calculated in terms of a positivity ratio, which is the ratio of positive words to negative words used throughout as president's speech. Ironically, President Jacob Zuma has the highest positive ratio where as Mandela is towards the lower end of the spectrum.**

```
#Calculating Net Sentiment
  data%>%inner_join(get_sentiments('bing')) %>%
  count(labels, sentiment) %>%
  spread(sentiment,n, fill = 0) %>%
  mutate(net_sent = positive-negative) %>%
  mutate(pos_ratio = positive/negative) %>%
  arrange(desc(pos_ratio))
```

**Using the tidy tool and sentiment analysis techniques mentioned above, we are now able to adapt the 'bing' lexicon in order to define what the most commonly used words per president are and what sentiment is attached to each word (positive or negative). We are able to vizualize this with the help of *ggplot2* package in R. The *blue* and *red* horizontal bars represent positive and negative senitments respectively.**

```
#Type of Sentiment
sent_data = data %>%inner_join(get_sentiments('bing'))
#Plottng fucntion using GGplot
plotevent <- function(string){
    sent_data %>%
    count(word,labels,sentiment) %>%
    group_by(labels) %>%
    top_n(10) %>% #only the top 10 selected
    ungroup() %>% filter(labels == string) %>%
    mutate(word = reorder(word, n)) %>% #order the words in terms of frequency (n)
    ggplot(aes(word,n, fill=sentiment)) +
    geom_col(show.legend = FALSE) + #ggtitle(as.character(no_speech$labels[i])) +
    facet_wrap(~labels, scales = "free") + ylab('Count') + xlab('Word') + coord_fl
ip()
}
#Call the plots onto one pages (grid)
p = list()
for(i in 1:6) {
  event = no_speech$labels[i]
  p[[i]] = plotevent(as.character(event))
}
do.call(grid.arrange,p)
```

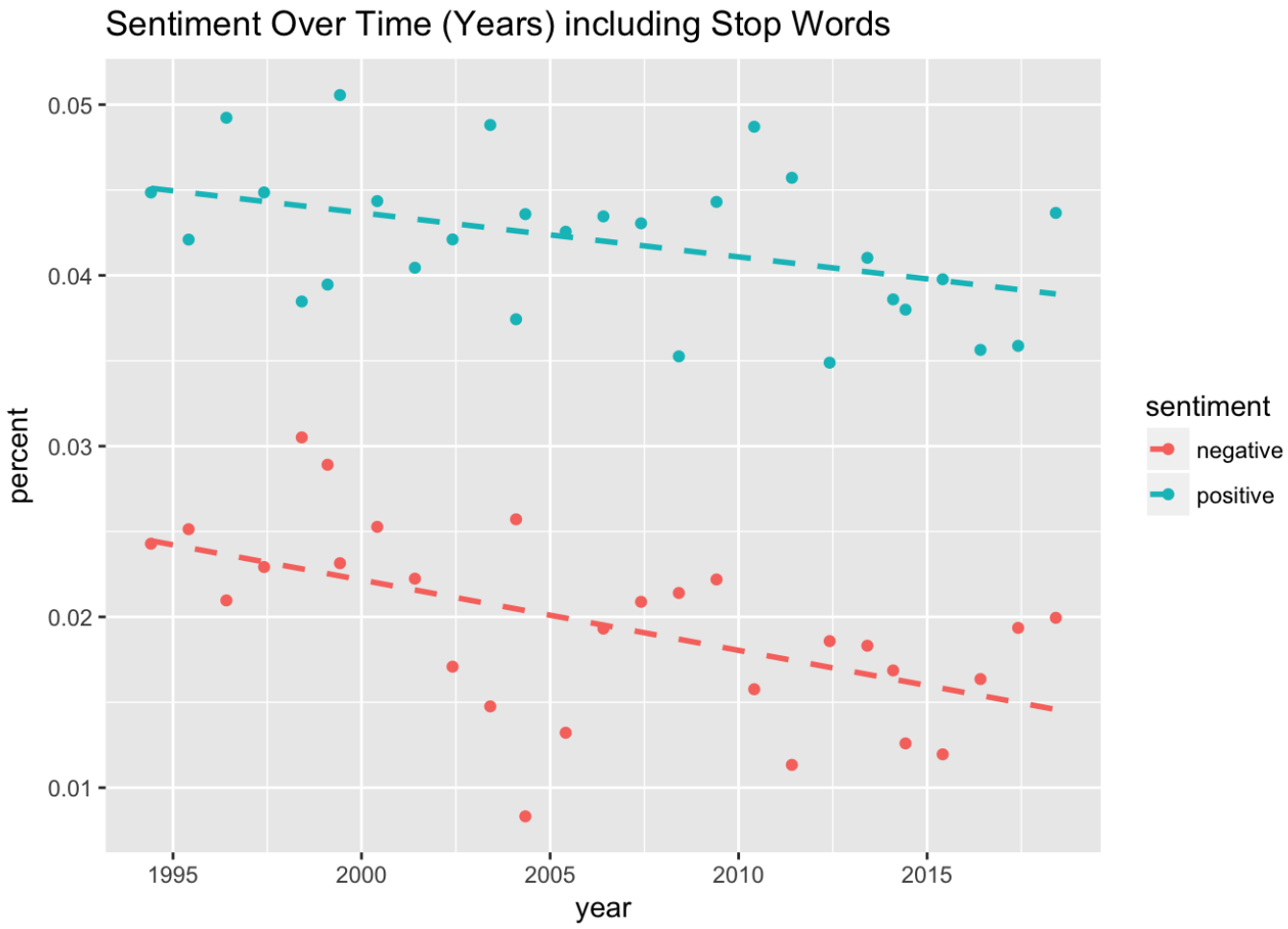



# Sentiment Over Time

**One of the more intriguing insights to derive is the how sentiment has changed over time, whether that would be due to the changes in presidential speech style or perhaps due to the transformation of the country as a whole. We can adapt the 'bing' sentiment lexicon and convert the years into dates as to be able to plot time on a continuous scale.**

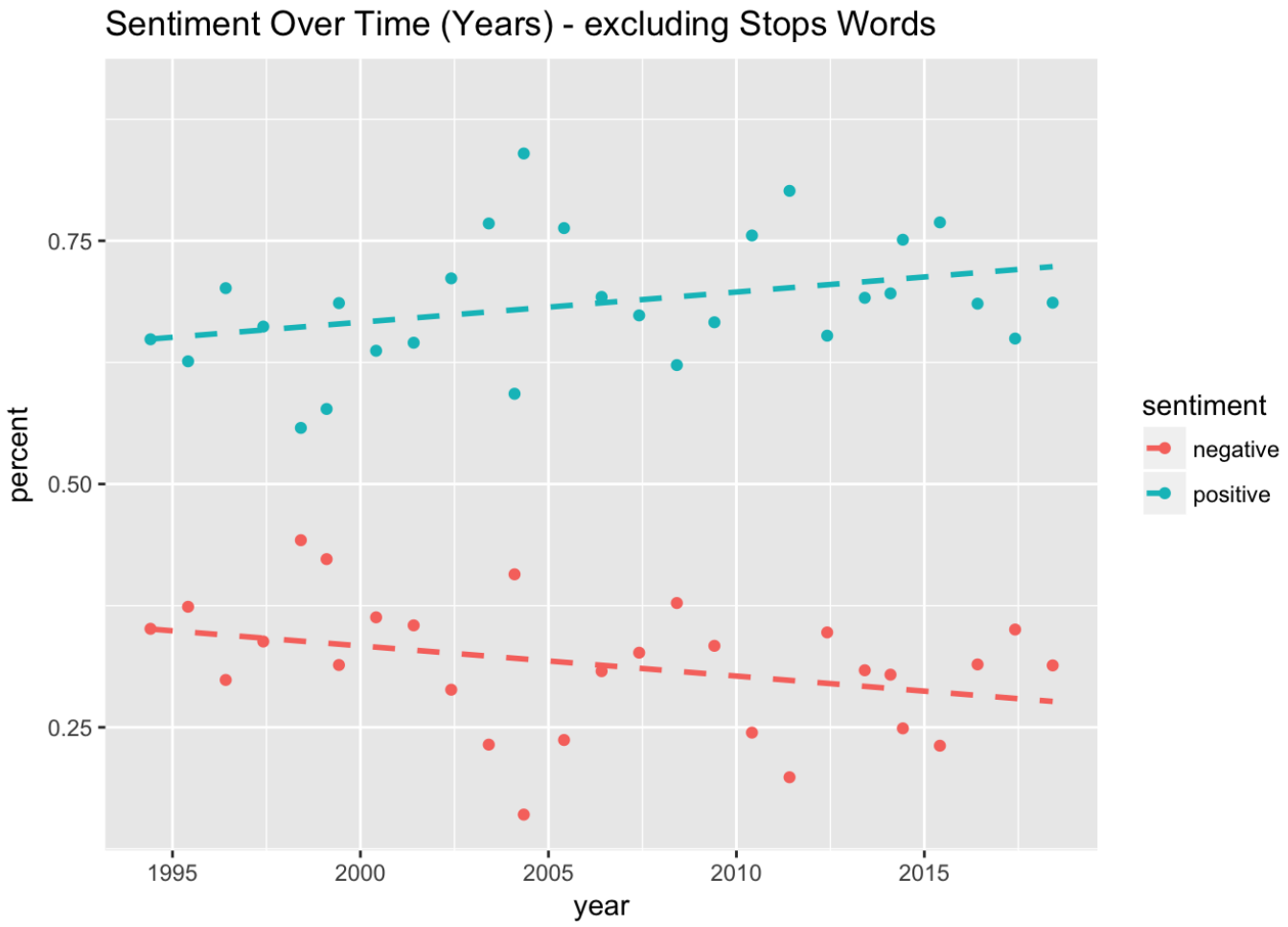
**The first plot calculates sentiment as a percentage of the time total number of words given as:**

**\[\begin{equation} Sentiment = \frac{PositiveNegative \ words}{Total\ number\ of \ Words} \end{equation}\]**




**The above graphic shows both average positive and negative sentiment *decreasing* between the years 1994 and 2018. However this is the second plot we change the calculation for sentiment by removing the neutral words and displaying the sentiment given by the following equation:**

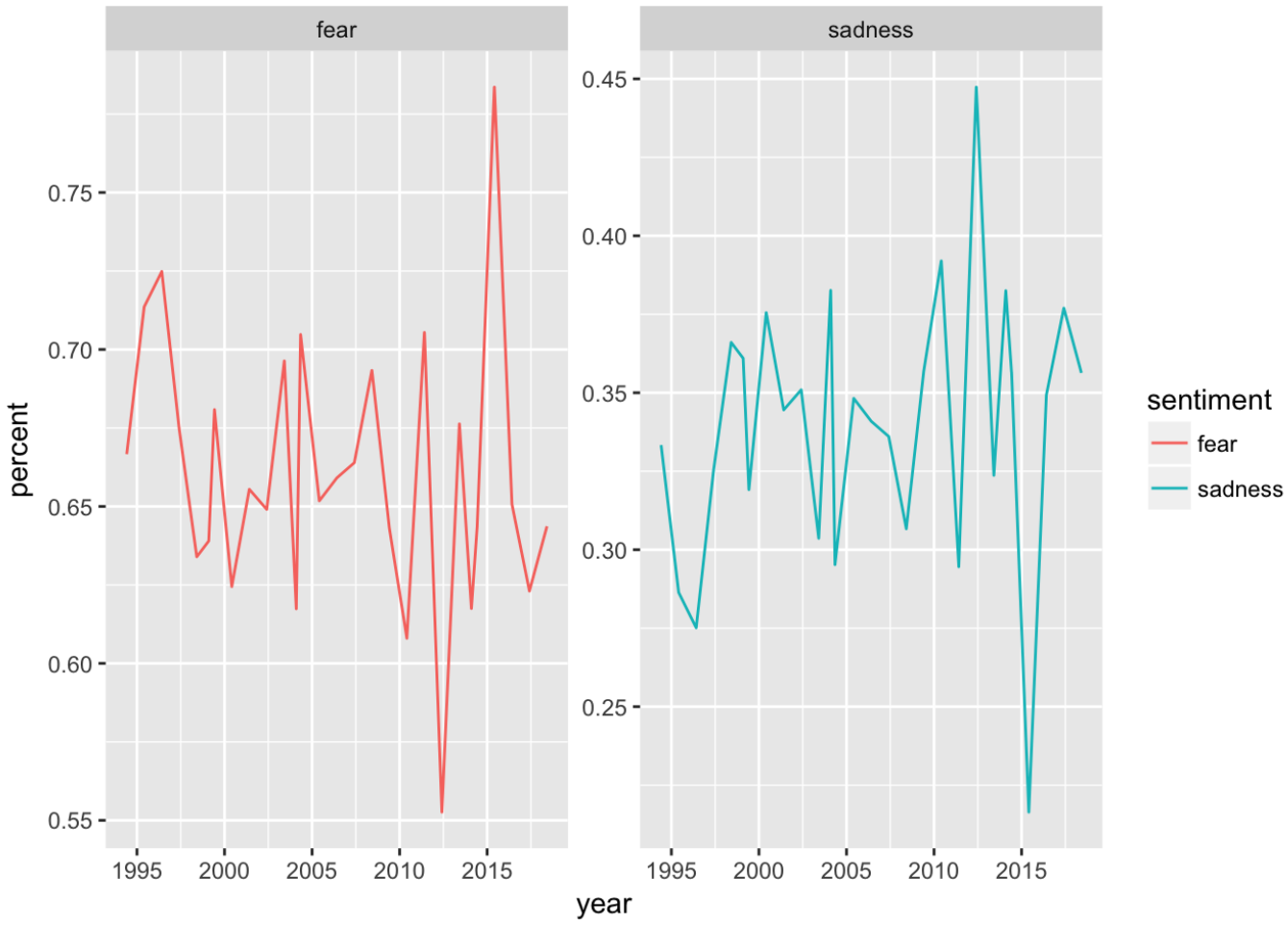
**\[\begin{equation} Sentiment = \frac{PositiveINegative \ words}{Positive + Negative \ words} \end{equation}\]**
**The result can be seen below:**




**Individually the two plots above tell a conflicting story of how sentiment has changed over time, however, when viewed together express an interesting trend in the SONA speeches over time. The first plot has both positive and negative sentiment decreasing over time, due to the fact that there seem to be less meaningful words (positive/negative) being used in the speeches when compared to the amount of neutral words (which includes stop words). From this statement one could argue that presidents have become more accustomed to waffling in the SONA speeches and the majority of the speech in recent years has become more neutral in sentiment. The second plot depicts the SONA speech over time without any neutral words and instead shows that speeches have become *more positive*, and subsequently *less negative* over time, as the two alternating sentiments diverse from the 1994 onwards.**

# Using an Alternative Lexicon (nrc)

**It is possible to adapt the 'nrc' lexicon in order to show a broader range of emotional sentiment. The 'nrc' lexicon contains 10 emotions in which word may be categorized. To call the new lexicon we run the get_sentiment() function and request the 'nrc' lexicon. The following plots show how sentiment 'sadness' and 'fear' fluctuate over time.**

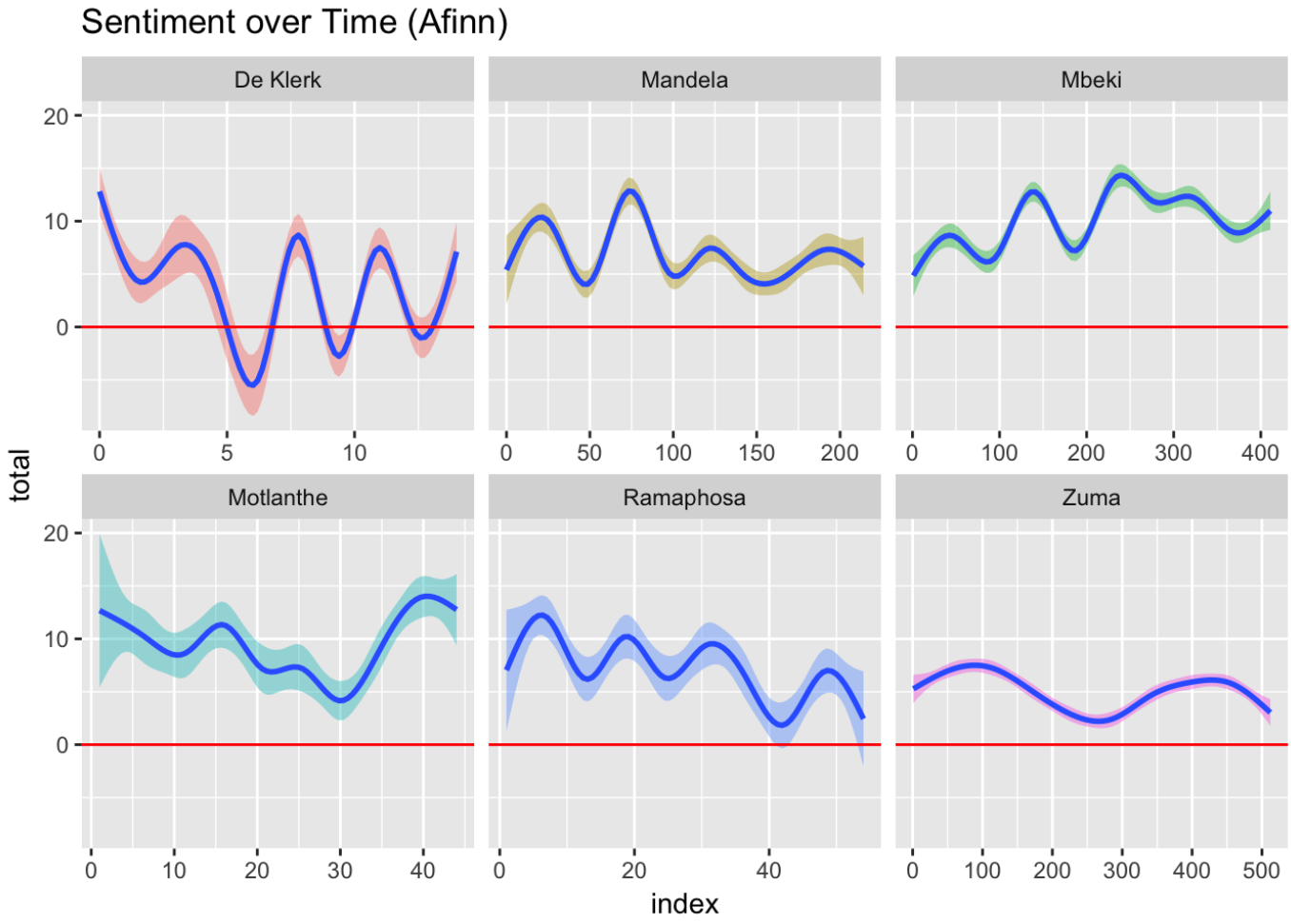



# Within Speech Fluctuations

**Sentiment analysis is not limited to a specific timeframe and one can dive further into the speeches themselves allowing us to see how sentiment may change throughout the delivery of the speech. In this section, the '*Afinn*' lexicon has been utilized. This lexicon maps an associated integer between -5 and 5 to a set list of words. The value itself indicates whether the word is negative or positive and to what extent to which this words has effective on the overall sentiment.**

**The index below is used to not only indicate a timeframe within the speech, but also used to sum sentiment scores around, creating a net-sentiment score for every 5 sentences within in the speech. The senitment over time is shown for each president below:**

```
#AFINN
raw_data %>% group_by(labels) %>%
  mutate(linenumber = row_number()) %>% #assign each sentence a linenumber
  unnest_tokens(word, text) %>%
  ungroup() %>%
  inner_join(get_sentiments("afinn")) %>%
  count(word, labels, index = linenumber %/%5, score) %>% #clump every 5 sentences
into a section/paragraph
  group_by(index,labels)  %>%
  mutate(total = sum(score)) %>%
    # Put index on x-axis, sentiment on y-axis, and map comedy/tragedy to fill
  ggplot(aes(index, total, fill=labels)) +
  # Make a bar chart with geom_col()
  geom_smooth(show.legend = FALSE) +
  geom_hline(yintercept=0, col='red') +  ggtitle('Sentiment over Time (Afinn)') +
  # Separate panels for each title with facet_wrap()
  facet_wrap(~labels,scales = "free_x" )
```




**As seen above, almost every president other than De Klerk manages to remain above a neutral sentiment (red line), and keeps the sentiment positive. These plots let the user gain insight into each president speaking styles and see the difference between one another. In contrasting styles, Mbeki seems to slowly build up the net-sentiment throughout his speech, but Ramaphosa starts off strong and slowly cools down close to neutral by the end.**

# N-grams and Bi-Grams

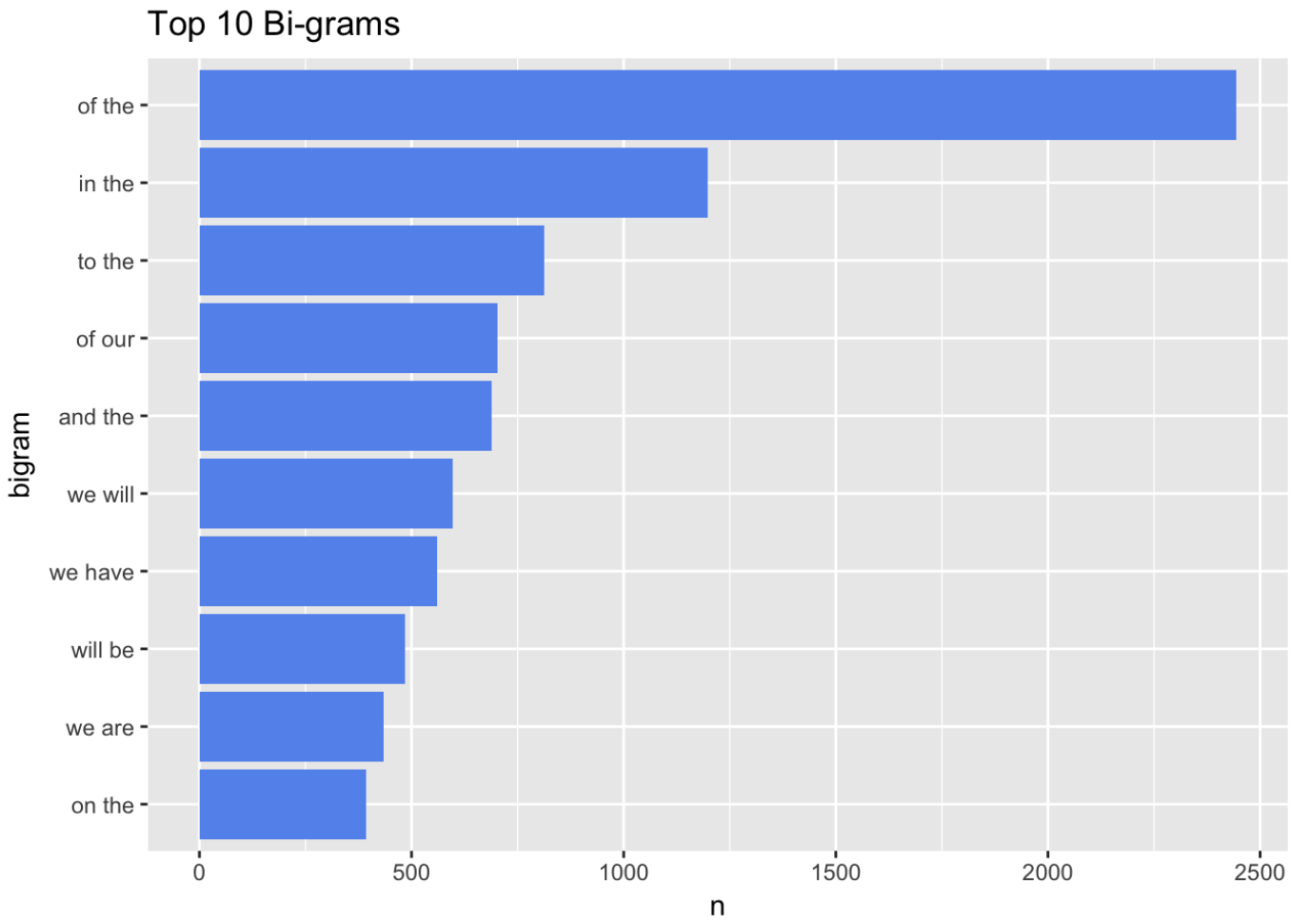
**This next section moves away from words as individual units and their single relation to the**

**sentiments, and now looks at the relationship between words. In this section, we will look at words and their relation to each other; and if certain words tend to occur next to each other frequently. In order to do this, the text is first tokenized into ngrams. An ngram is pair of adjacent words, and to do this we call the following function below:**

```
#retokenize the sentences to retrive bigrams (two word combinations)
bi_data = raw_data %>% unnest_tokens(bigram, text,token = "ngrams", n = 2)
```

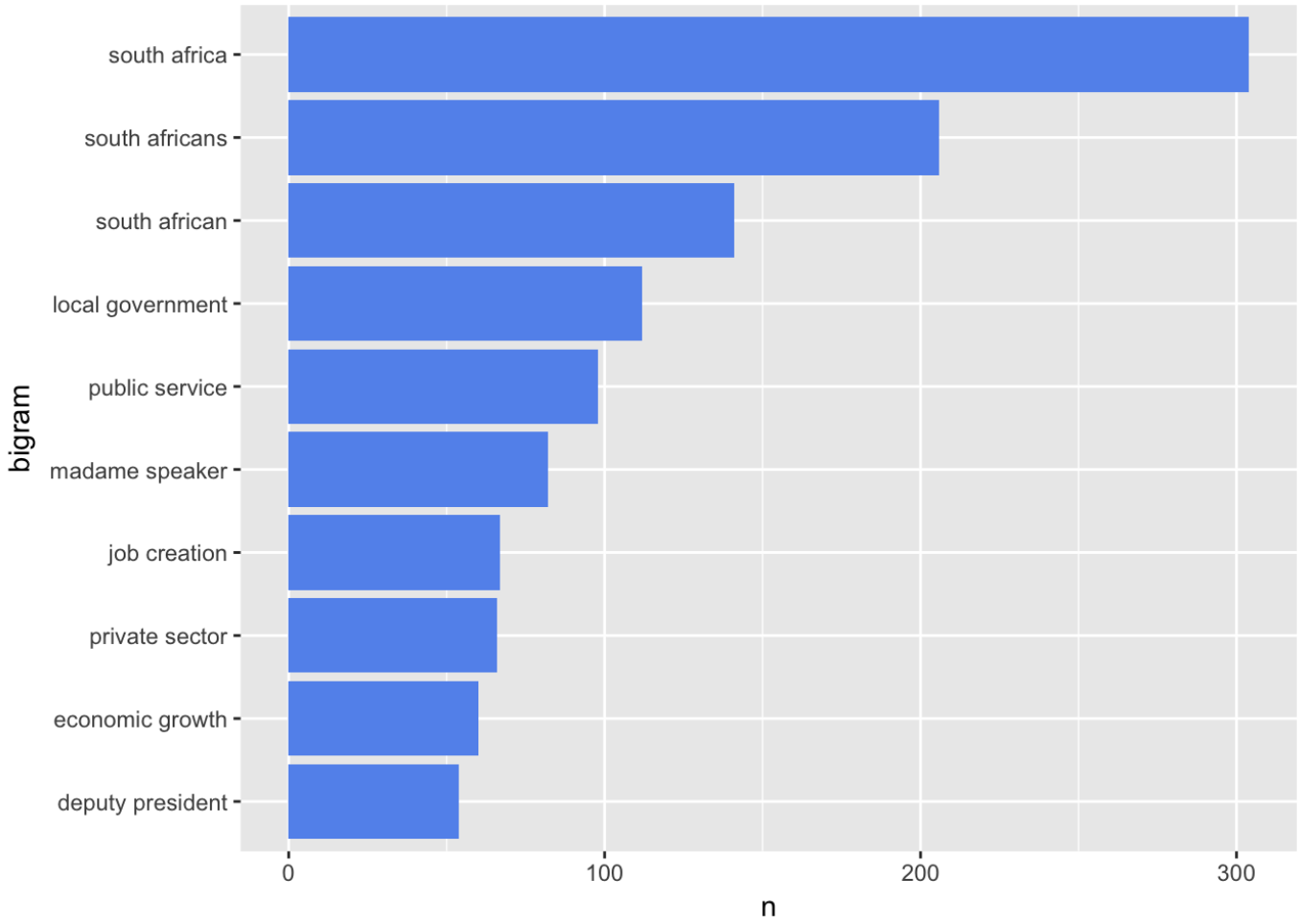
**Above the number of words is specified to equal 2, and thus onyl looking for two word combinations caleld bi-grams. It is now possible to use the tidy data principles to determine the top 10 bi-grams. As to be expected, the top 10 bi-grams are usually combinations of 'stop words' such as :**

```
#Count the most populalr bi-grams
bi_data %>% count(bigram, sort = TRUE) %>%
  top_n(10) %>%
  ungroup() %>%
  mutate(bigram = reorder(bigram, n)) %>%
  ggplot(aes(x=bigram,y=n)) +
  geom_col( fill = "cornflowerblue") +
  coord_flip() +
  ggtitle("Top 10 Bi-grams")
```




**As can be seen above, the most common bi-grams are made up on *stop words* which need to removed. The stop_words dataframe found in the *tidytext* package is utilized as can be seen below:**

```
#Split up bigrams into two words
sep_bi_data=bi_data %>% separate(bigram, c("word1", "word2"), sep = " ")
#Remove stop words
bigrams_filtered <- sep_bi_data %>%
  filter(!word1 %in% stop_words$word) %>%
  filter(!word2 %in% stop_words$word)
#Unite the bigrams once again
bigrams_united <- bigrams_filtered %>%
  unite(bigram, word1, word2, sep = " ")

bigrams_united %>% count(bigram, sort = TRUE) %>%
  top_n(10) %>%
  ungroup() %>%
  mutate(bigram = reorder(bigram, n)) %>%
  ggplot(aes(bigram, n)) +
  geom_col(fill = "cornflowerblue") +
  coord_flip()
```



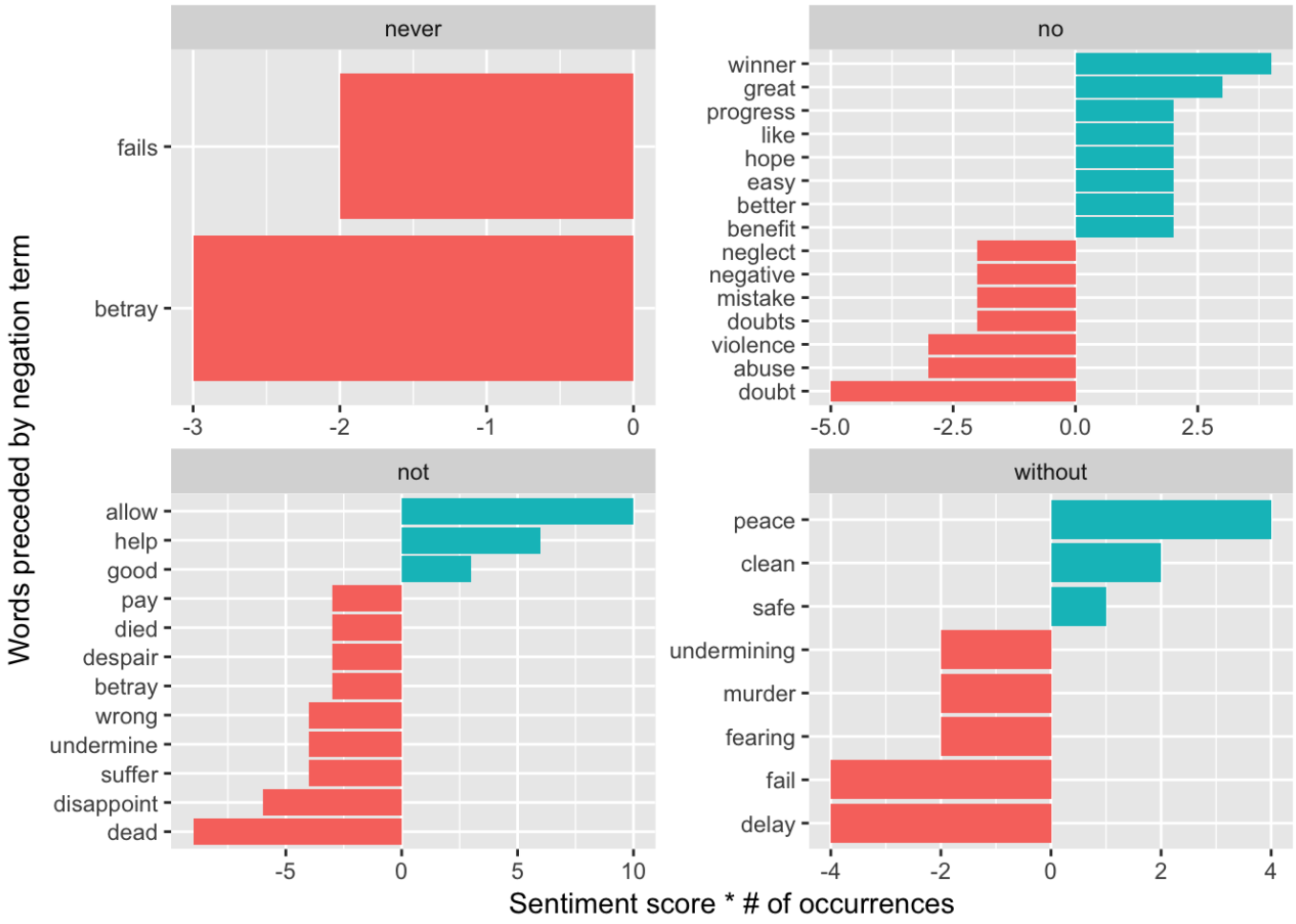

# Negators and Amplifiers

**One of the most common problems in text mining is dealing with the presence of negators and amplifiers within text. Negators are words that invert polarized meaning e.g. “not good”; whilst amplifiers are words that increase the emtional intent e.g “very good”. Carrying on with the seciton above, we can not check for how many times a ‘not’ negator has come up on the SONA speeches by filtering for the ‘not’ negator in the first word of the bi-gram and then inner_joining a lexicon to calcuate an score for prescense of the negator.**

```
#left join the AFINN sentiment lexicon and count the number of time the 'not' negat
or has been used with followw up word which now has a different meaning due to the
presence of the negator
  sep_bi_data %>%
  filter(word1 == "not") %>%
  inner_join(get_sentiments('afinn'), by = c(word2 = "word")) %>%
  count(word2, score, sort = TRUE) %>%
  ungroup()%>%
  mutate(contribution = n * score) %>%
  arrange(desc(abs(contribution))) %>%
  head(15) %>%
  mutate(word2 = reorder(word2, contribution)) %>%
  ggplot(aes(word2, n * score, fill = n * score > 0)) +
  geom_col(show.legend = FALSE) +
  xlab("Words preceded by \"not\"") + ggtitle('The "not" negator in SONA speeches')
+
  ylab("Sentiment score * number of occurrences") +
  coord_flip()
```


**The plot above shows the top 15 bi-grams with all are preceeded by the 'not' negator. In the case above, the negator seem to invert polarized meaning on majority negative words (red), and thus inverting the negative sentiment to a positive sentiment. This is then conducted for other negators as can be seen below:**

```
#Other Negators
negation_words <- c("not", "no", "never", "without")
negated_words <- sep_bi_data %>%
  filter(word1 %in% negation_words) %>%
  inner_join(get_sentiments('afinn'), by = c(word2 = "word")) %>%
  count(word1, word2, score, sort = TRUE) %>%
  ungroup()
#Graphically view the top 10 contributing follow-up
negated_words %>%
  mutate(contribution = n * score,
         word2 = reorder(paste(word2, word1, sep = "__"), contribution)) %>%
  group_by(word1) %>%
  top_n(10, abs(contribution)) %>%
  ggplot(aes(word2, contribution, fill = n * score > 0)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~ word1, scales = "free") +
  scale_x_discrete(labels = function(x) gsub("__.+$", "", x)) +
  xlab("Words preceded by negation term") +
  ylab("Sentiment score * # of occurrences") +
  coord_flip()
```




# Polarity (Negation Handling)

**This section will cover how to handle negators and amplifiers in sentiment analysis in R. Any sentence is Polarity goes hand in hand with sentiment analysis an is used interchangeably with sentiment. The *SentimentR* package is able to read the text and assign a polarity score for each sentence by calling the sentiment_by function. The polarity score is calculated as follows for the**

**example sentence:**

- **“This course is very good”**

**The *amplifier* in this sentence is ‘very’ and is given a weight score of 0.6 while the polarize word ‘good’ is given a value of 0.75. The polarity score is the determined by:**

**$$\begin{equation} Polarity\ Score = \frac{Amplier|Negator + polarized\ word}{\sqrt{Total\ number\ of\ Words}} \end{equation}$$**

**The example sentence would have a score of 0.603. This method is able to accurately take negators and amplifiers into account when determining overall sentiment. Now it is possible to apply this technique to all the sentences across a variety of president to determine a more accurate sentiment trend over time.**

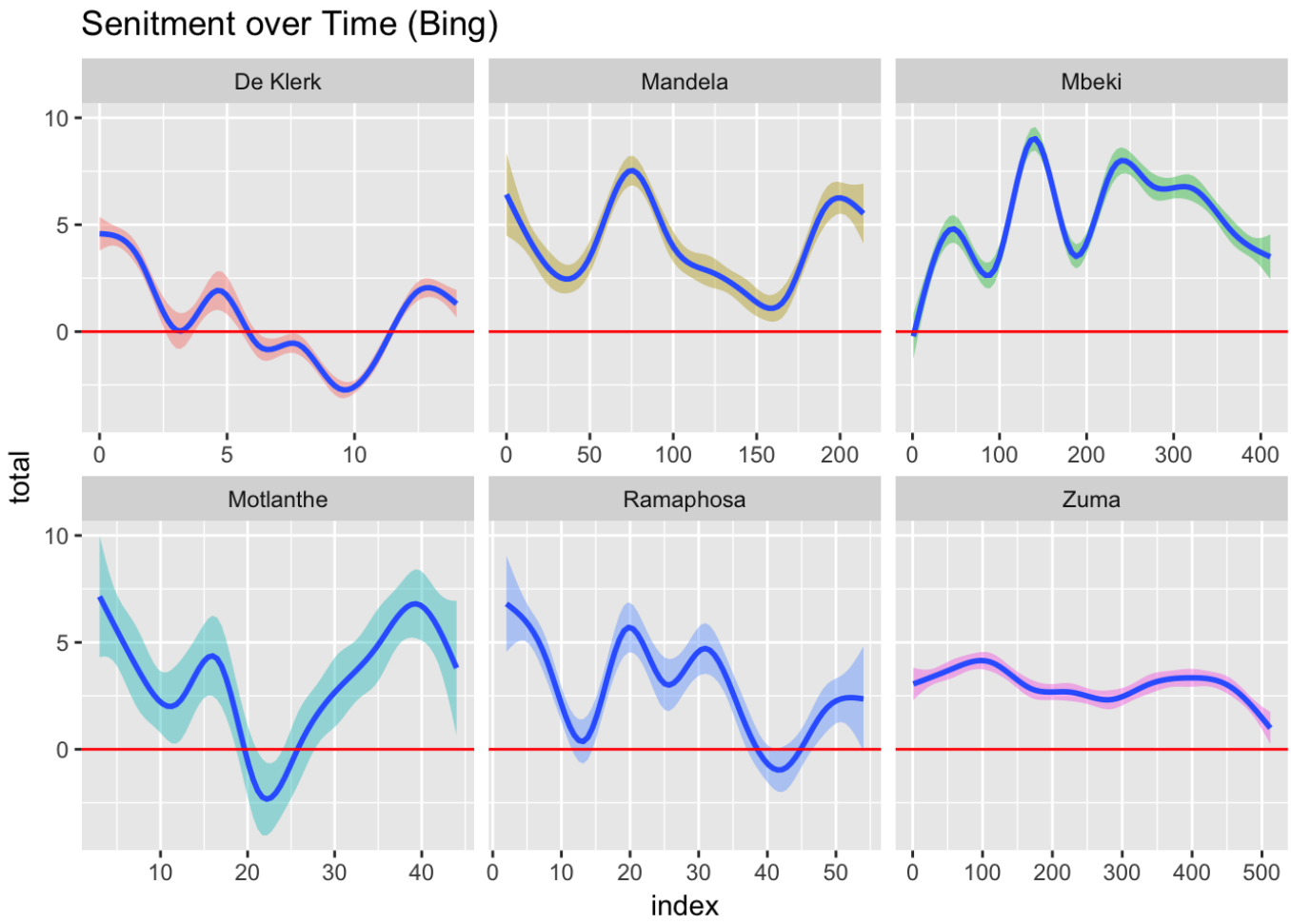
```
pol_data=raw_data %>% group_by(labels) %>%
  mutate(linenumber = row_number()) %>%
  ungroup()
  #get polarity by applying Sentiment_by() fucntion
  ratings = sentiment_by(get_sentences(pol_data$text))
  pol_data$polarity = ratings$ave_sentiment

  pol_data %>%
  count(text, labels, index = linenumber %/%5, polarity) %>%
  ungroup() %>%
    # Put index on x-axis, sentiment on y-axis, and map comedy/tragedy to fill
  ggplot(aes(index, polarity, fill=labels)) +
  # Make a bar chart with geom_col()
  geom_smooth(show.legend = FALSE) +
  geom_hline(yintercept=0, col='red') +
  # Separate panels for each title with facet_wrap()
  facet_wrap(~labels,scales = "free_x" )
```


# Appendix

```
#BING LEXICON
#reprocess Data to create sentence numbers
raw_data %>% group_by(labels) %>%
  mutate(linenumber = row_number()) %>%
  unnest_tokens(word, text) %>%
  ungroup() %>%
  inner_join(get_sentiments("bing")) %>%
  count(word, labels, index = linenumber %/%5, sentiment) %>%
  spread(sentiment, n, fill = 0) %>%
  mutate(sent = positive - negative) %>%
  group_by(index,labels) %>%
  mutate(total = sum(sent)) %>%
  ungroup() %>%
  # Put index on x-axis, sentiment on y-axis, and map comedy/tragedy to fill
  ggplot(aes(index, total, fill=labels)) +
  # Make a bar chart with geom_col()
  geom_smooth(show.legend = FALSE) +
  geom_hline(yintercept=0, col='red') + ggtitle('Senitment over Time (Bing)') +
  # Separate panels for each title with facet_wrap()
  facet_wrap(~labels,scales = "free_x" )
```

```
## Joining, by = "word"
```

```
## `geom_smooth()` using method = 'gam'
```


Senitment over Time (Bing)
De Klerk
Mandela
Mbeki
Motlanthe
Ramaphosa
Zuma
total
index
10
5
0
0
5
10
0
50
100
150
200
0
100
200
300
400
10
20
30
40
0
10
20
30
40
50
0
100
200
300
400
500