##### (a): Counting and filtering n-grams

Our usual tidy tools apply equally well to n-gram analysis. We can examine the most common bigrams using dplyr’s count():

bigrams\_count<-count(bigrams,bigram,sort=T)  
head(bigrams\_count)

## # A tibble: 6 x 2  
## bigram n  
## <chr> <int>  
## 1 of the 5581  
## 2 in the 2743  
## 3 to the 1847  
## 4 and the 1343  
## 5 it was 1037  
## 6 from the 1036

bigrams\_EAP\_count<-count(bigrams\_EAP,bigram,sort=T)  
head(bigrams\_EAP\_count)

## # A tibble: 6 x 2  
## bigram n  
## <chr> <int>  
## 1 of the 2877  
## 2 in the 1237  
## 3 to the 823  
## 4 of a 530  
## 5 to be 431  
## 6 and the 428

bigrams\_MWS\_count<-count(bigrams\_MWS,bigram,sort=T)  
head(bigrams\_MWS\_count)

## # A tibble: 6 x 2  
## bigram n  
## <chr> <int>  
## 1 of the 1217  
## 2 in the 605  
## 3 to the 534  
## 4 and the 412  
## 5 of my 359  
## 6 on the 356

bigrams\_HPL\_count<-count(bigrams\_HPL,bigram,sort=T)  
head(bigrams\_HPL\_count)

## # A tibble: 6 x 2  
## bigram n  
## <chr> <int>  
## 1 of the 1487  
## 2 in the 901  
## 3 and the 503  
## 4 to the 490  
## 5 on the 428  
## 6 from the 350

As one might expect, a lot of the most common bigrams are pairs of common (uninteresting) words, such as of the and in the: what we call “stop-words” . This is a useful time to use tidyr’s separate(), which splits a column into multiple based on a delimiter. This lets us separate it into two columns, “word1” and “word2”, at which point we can remove cases where either is a stop-word.