NLP - Assignment 5

In this assignment you will apply sentiment analysis on twitter messages.

1 Collect tweets

Have your twitter streamer run for an entire day to later be able to assess the average sentiment across the hours of the day.

1.1 Extract variables

Extract from the tweets's time of creation and the text.

1.1 Process time

Extract the time of day from the time variable using `str_extract_all()` from the the `stringr` package. As the pattern use `[0-9]{2}:[0-9]{2}:[0-9]{2}` or a variant of it in case the time of day is formatted differently in your case. Next parse the extracted times of day using `parse_time()` from the `readr` package. This should work without specifying the `format` argument. If not provide a specific format. `?parse_time` will tell you how. Finally, use the `hour()`-function from the lubridate package to extract the hour of day. This should in a numeric variable that contains whole number values indicating the hour of day. Make sure that your data is stored as a `tibble` and add to it the `hour` vector as a new variable.

```
# read my tweets
data = readRDS('time_and_text.txt')

# load packages
require(readr)
```

```
## Loading required package: readr
```

```
require(stringr)
```

```
## Loading required package: stringr
```

```
require(lubridate)
```

```
## Loading required package: lubridate
```

```
##
## Attaching package: 'lubridate'
```

```
## The following object is masked from 'package:base':
##
##     date
```

```
require(tibble)
```

```
## Loading required package: tibble
```

```
# extract hours
hours = hour(parse_time(unlist(str_extract_all(data[, 1],'[0-9]{2}:[0-9]{2}:[0-9]{2}'))))

# render tibble
data = as.tibble(data)
```

```
# add hours
data$hour = hours
```

2 Sentiment analysis

In this section use the very convenient `tidytext` package to run the sentiment analysis.

2.1 Tokenize

Tokenize the text (i.e., separate the tweets into words) using `unnest_tokes()` from the `tidytext` package, which yo will have to install and load. Use the pipe operator á la `data %>% unnest_tokens(word, text)` where word will be the name of the new variable containing the words (tokens) and text is (must be) the name of the variable containing the tweets.

2.2 Assign sentiments

Identify the sentiments of the different words by joining the data with the AFINN sentiment lexicon using `inner_join()` using again the pipe operator á la `data %>% inner_join(get_sentiments("afinn"))`.

2.3 Determine mean sentiment

Determine the mean sentiment using the `group_by()` and `summarize()` idiom from the `dplyr` package. E.g., `data %>% group_by(name_of_grouping_variable) %>% summarize(mean(name_of_target_variable))` - simply replace the object/variable names. Plot the result using `ggplot2` and post it on twitter. If you like you can combine all tasks of this section including plotting using the pipe (`dplyr`) and plus (`ggplot2`) operators.

```
# load packages
require(tidytext)
```

```
## Loading required package: tidytext
```

```
## Warning: package 'tidytext' was built under R version 3.4.4
```

```
require(dplyr)
```

```
## Loading required package: dplyr
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:lubridate':
##
##     intersect, setdiff, union
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
require(ggplot2)
```

```
## Loading required package: ggplot2
```

```r
# do sentiment analysis
data %>%
  unnest_tokens(word, text) %>%
  inner_join(get_sentiments("afinn")) %>%
  group_by(hour) %>%
  summarize(sentiment = mean(score)) %>%
  ggplot(aes(x = hour, y = sentiment )) +
  geom_line()
```

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