Exploring Differences in Online Communites of Programming Languages in Data Science

Josue Rodriguez

12/1/2019

Introduction

This project set out to explore whether there were differences in the online communities corresponding to the `R`, `Python`, and `Julia` programming languages. To do so, I used `Python` to scrape the `#rstats`, `#pydata`, and `#julialang` Twitter hashtags. I examined these tweets using word-frequncies, tf-idf, sentiment analysis, and structural topic modeling. I hypothesized that the `R` community would use more positive language overall. All code and data used in this project are available on my GitHub at `https://github.com/josue-rodriguez/analyzing-twitter-hashtags`.

Load in data and libraries

```
library(tidyverse)
library(tidytext)
library(drlib)
library(scales)
library(ggpubr)
library(cowplot)

data("stop_words")

tweets <- read_csv('tweet_results.csv')
tweets
```

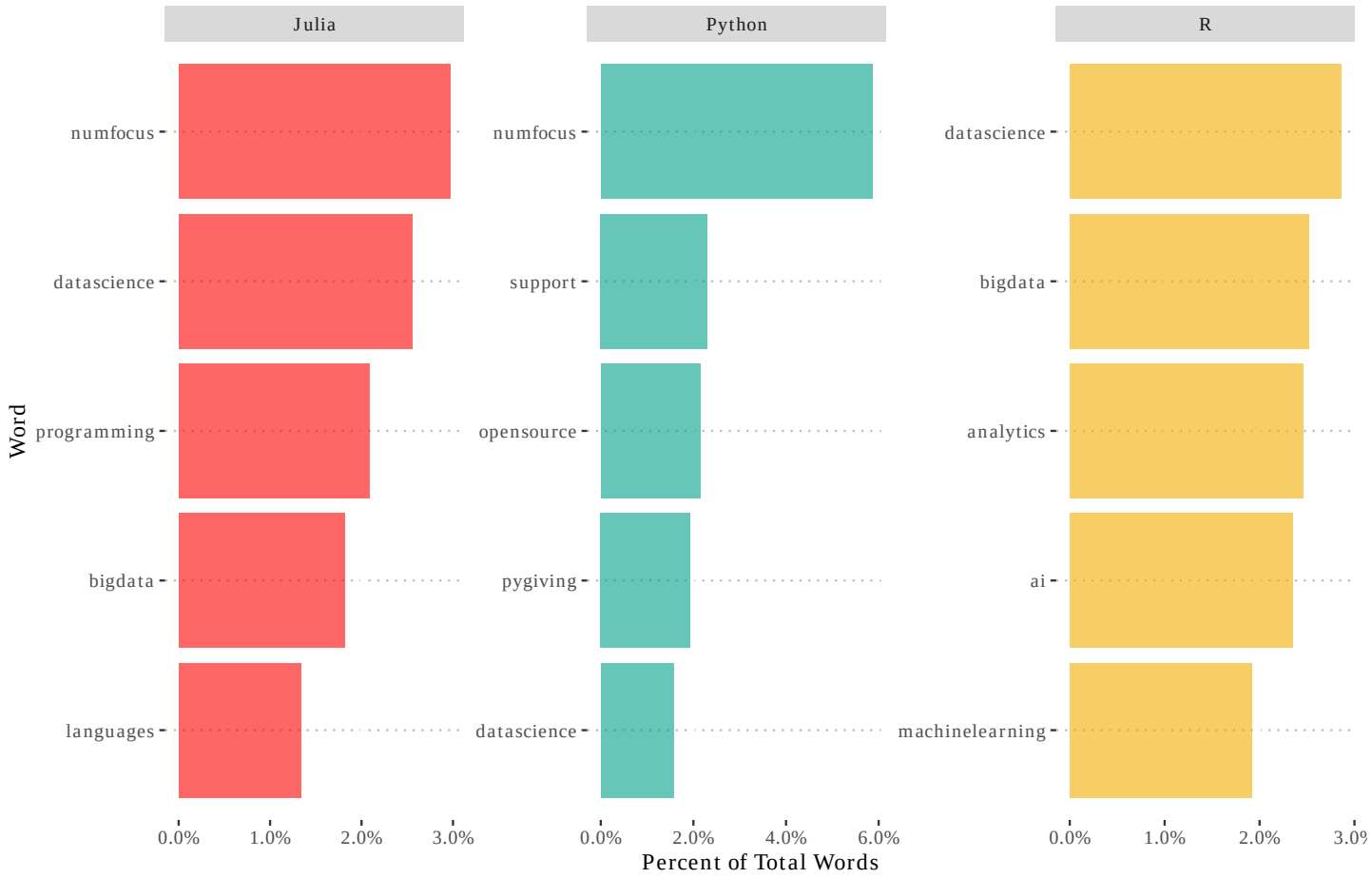
```
## # A tibble: 49,735 x 2
##    language tweet
##    <chr>    <chr>
##  1 R        "b'RT @LearnRinaDay: Debugging in R: How to Easily and Efficie~
##  2 R        "b'#30DayMapChallenge #\\xe2\\x83\\xa31\\xe2\\x83\\xa39\\xe2\\~
##  3 R        "b'RT @gp_pulipaka: Data Science on Steroids with #Kubeflow. #~
##  4 R        b'RT @CRANberriesFeed: CRAN updates: corpustools enveomics.R O~
##  5 R        "b'RT @leonawicz: I found the cause of this grave issue that w~
##  6 R        "b'RT @tomkXY: @dr_piv I think @thecarpentries is great for al~
##  7 R        "b'RT @IsabellaGhement: What I really wanted is a prediction i~
##  8 R        "b'RT @IsabellaGhement: I am trying to use the Gls() function ~
##  9 R        b'RT @Rbloggers: Hangman game with R  {https://t.co/178DPBsU4t~
## 10 R        "b'RT @daniellequinn88: Register now for the next course, happ~
## # ... with 49,725 more rows
```

What words are used most often among these online communities?

To answer this question, word frequencies were computed for each language, after removing stop words (i.e., overly common words, or words not of interest). Frequencies were computed as the percentage of tweets which included a word or phrase. Julia and Python both used numfocus the most, indicating that there might be some overlap between those two communities. Interestingly, Python commonly used "pygiving" and "opensource", while only R had the phrase "machinelearning" in it's top 5 most frequent words. This is a counterintutive finding as I anticipated a larger focus on machine learning within the Python community.

```
# additional stop words (i.e., words of no value to us)
additional_stop_words <- c("xe2", "x80", "b'rt", "pydata", "rstats", "t.co",
                           "x94", "https", "hnwuqaqrcx", "xa6", "x9f", "xf0", "rstats",
                           "julialang", "pystats", "pydata", "python", "r", "julia",
```

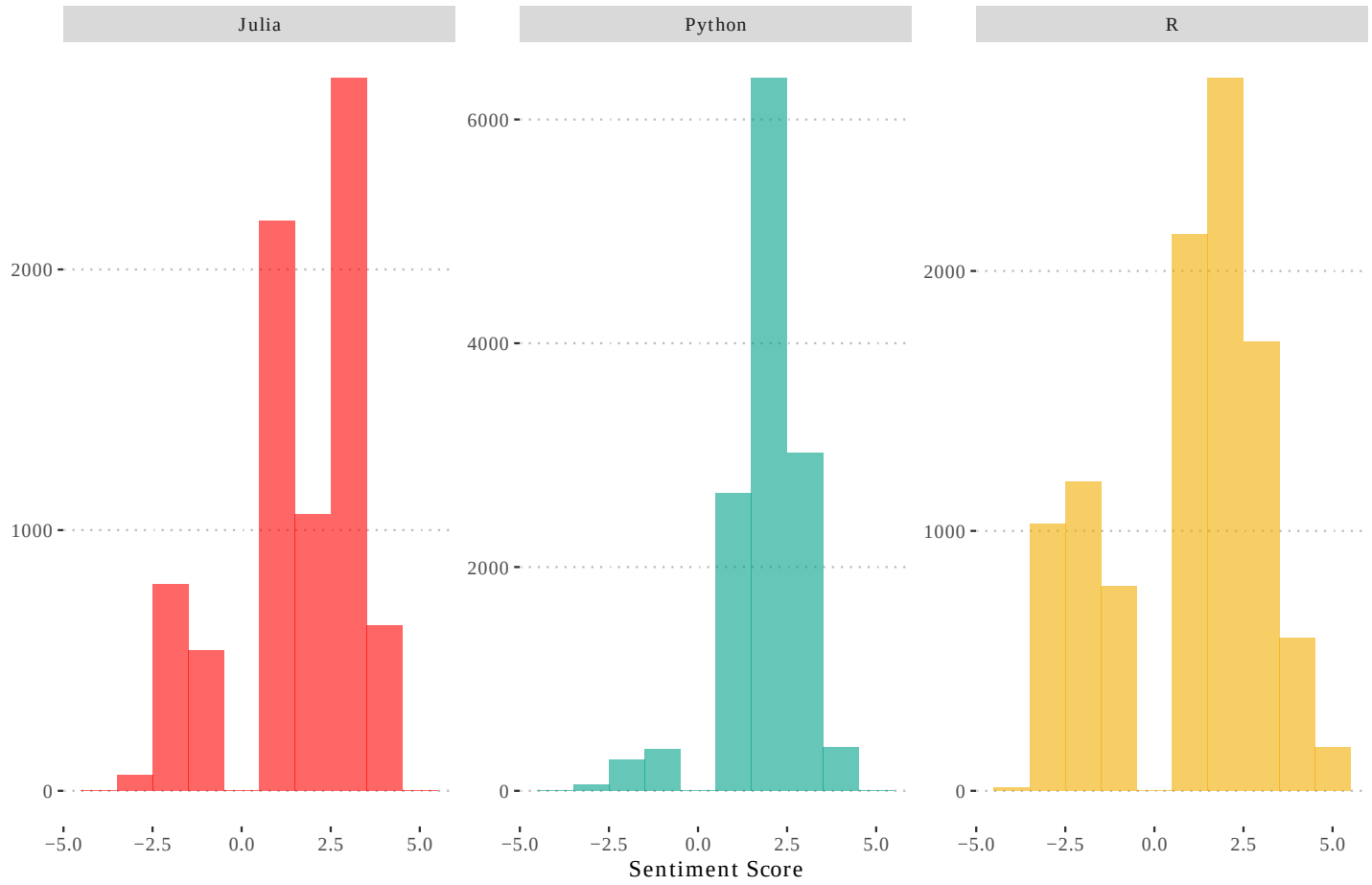
```
                          "nhttps", "rt", "iiot", "b'julia", "1.3", "y5w2ntksxt", "gp_pulipaka")

tweets_cleaned <-
  tweets %>%
  unnest_tokens(word, tweet) %>%
  anti_join(stop_words) %>%
  filter(!word %in% additional_stop_words)

top_five <-
  tweets_cleaned %>%
  count(language, word) %>%
  group_by(language) %>%
  mutate(perc = n / sum(n)) %>%
  top_n(5) %>%
  ungroup

# set options for plot
windowsFonts(Century = windowsFont("Century Gothic"))
pal <- wesanderson::wes_palette("Darjeeling1")

# plot top words by count
top_five %>%
  mutate(word = reorder_within(word, perc, language)) %>%
  ggplot(aes(word, perc, fill = language)) +
  geom_col(alpha = 0.6, show.legend = FALSE) +
  scale_x_reordered() +
  scale_y_continuous(labels = percent_format(accuracy = .1)) +
  scale_fill_manual(values = pal) +
  facet_wrap(~ language, scale = "free") +
  coord_flip() +
  labs(title = "5 Most Frequently used Words by Programming Language",
       subtitle = "Julia and Python are almost identical. R has more of a focus on machine learning.",
       x = "Word",
       y = "Percent of Total Words") +
  theme_pubclean(base_family = "Century")
```



Do different online communities display different emotions in their tweets?

Sentiment analysis was employed to answer this question. Sentiment scores were given a rating between -5 and 5 using the AFINN lexicon, and their distributions were plotted. Negative scores indicate negative sentiment and positive scores indicate positive sentiment. Positive language (scores of about 2.5) was mostly used across all three online communities. The R online community had the highest density of negative language, contrary to my expectation, but in line with my expectation, also had the highest density of positive language. Overall, Julia, Python, and R were very similar in their distribution of sentiment scores.

```
# calculate sentiment scores
tweet_sentiments <-
  tweets_cleaned %>%
  mutate(tweet_num = row_number()) %>%
  inner_join(get_sentiments("afinn")) %>%
  rename(sentiment = value)

ggplot(tweet_sentiments, aes(sentiment, fill = language)) +
  geom_histogram(bins = 10, alpha = 0.6, show.legend = FALSE) +
  facet_wrap(~ language, scales = "free_y") +
  scale_fill_manual(values = pal) +
  labs(title = "Distribution of Sentiment Scores Across Languages",
       subtitle = "Tweets across all hashtags used mainly positive language.",
       x = "Sentiment Score",
       y = NULL) +
  theme_pubclean(base_family = "Century")
```

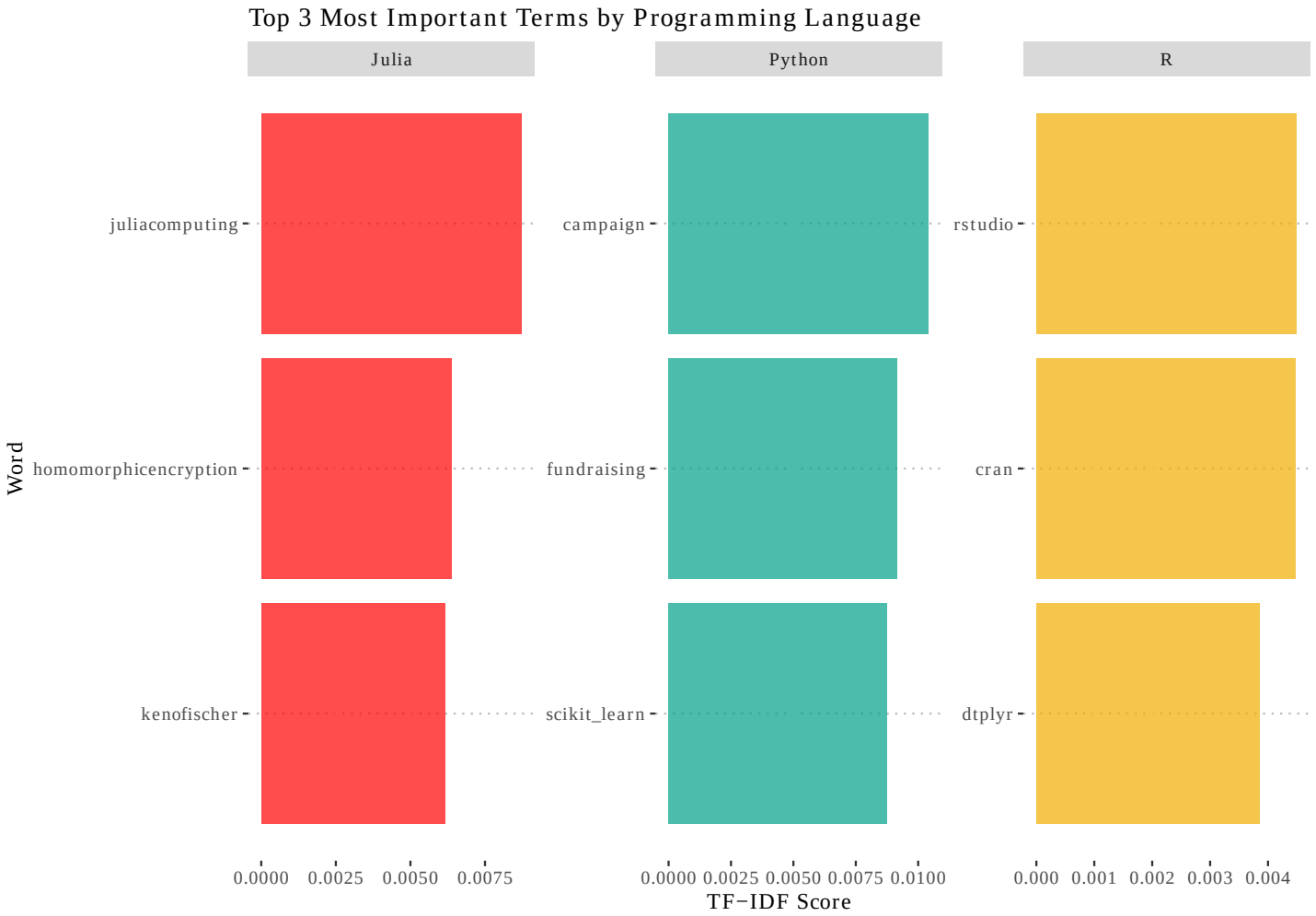


Are some words more important in certain online communities?

Term frequency—inverse document frequency (tf-idf) is a statistic which aims to represent the importance of a word to a document. Tweets from each online community were given tf-idf scores in order to answer whether some words were more important in some online communities than others. In this project, tweets belonging to a specific online community made up a document. Thus, tf-idf was used to determine which words were most important to there respective online communities. The most important terms were "juliacomputing", "campaign", and "rstudio", for the Julia, Python, and R Twitter communities, respectively. Sci-kit learn was an important term to the Python communinity, lending support to the idea that machine learning is central to their community.

```
# create tf-idf, sort by top language and get top ten
tweets_tf_idf <-
  tweets_cleaned %>%
  count(language, word, sort = TRUE) %>%
  bind_tf_idf(term = word, document = language, n = n) %>%
  arrange(-tf_idf) %>%
  group_by(language) %>%
  top_n(3, wt = tf_idf) %>%
  ungroup

# plot tf-idf
tweets_tf_idf %>%
  mutate(word = reorder_within(word, tf_idf, within = language)) %>%
  ggplot(aes(word, tf_idf, fill = language)) +
  geom_col(alpha = 0.7, show.legend = FALSE) +
  facet_wrap(~ language, scales = "free") +
  scale_x_reordered() +
```

```
  coord_flip() +
  scale_fill_manual(values = pal) +
  labs(title = "Top 3 Most Important Terms by Programming Language",
       x = "Word",
       y = "TF-IDF Score") +
  theme_pubclean(base_family = "Century")
```



Are there distinct topics of conversaton within online communities?

Structural topic modeling was used to discover whether the R, Python, and Julia online communities engaged in distinct topics of tweets. Structural topic modeling is a technique used to identify latent topics within a text documents. It is useful for identifying what words belong to a topic, and the probability the topic belongs to one or more online communities. The user is required to specify the number of topics to identify, and later subjective judgments are made on the appropriate number of topics to use. Models were firt with 3, 4, and 5 topics. The model with 3 topics fit best, as there were 3 topics identified, and each topic had probability of 1 of belonging to at least 1 community. When evaluating each topic, they each align closely to the most frequent words used by each online community. This finding lends support that the R, Python, and Julia online data science communities each tweet about distinct topics. However, these topics are similar.

```
# create sparse matrix with one term per document, per row
tweets_dfm <-
  tweets_cleaned %>%
  count(language, word, sort = TRUE) %>%
  cast_dfm(document = language, term = word, value = n)
```

```
# create topic models
library(stm)
```

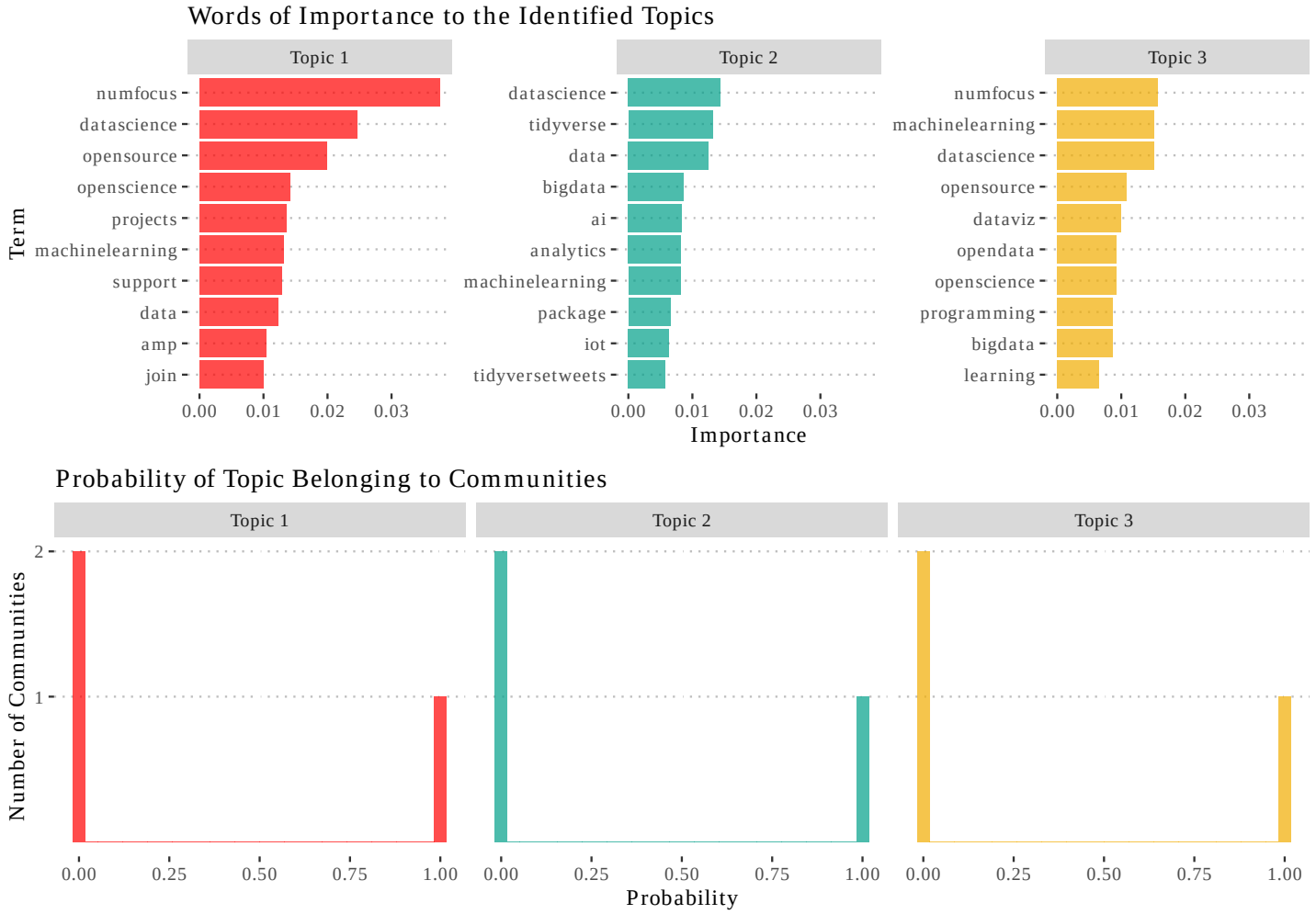
```r
# recommended topics for small corpora is 3 - 10
# models were ran and written out to files to avoid re-running the code
stm.3 <- stm(tweets_dfm, K = 3, verbose = FALSE, init.type = "Spectral")
stm.4 <- stm(tweets_dfm, K = 4, verbose = FALSE, init.type = "Spectral")
stm.5 <- stm(tweets_dfm, K = 5, verbose = FALSE, init.type = "Spectral")
save(stm.3, stm.4, stm.5, file = "02-stm-data.RData")
```

```r
load("02-stm-data.RData")

# create functions to tidy stm output and plot

plot_stm_beta <- function(topic_model) {
  # suppress messages from following code block
  suppressMessages({
    # create dataframe with topic and beta weight
    td_beta <- broom::tidy(topic_model)

    # grab top 10 words associated with each topic
    plotting_data <-
      td_beta%>%
      group_by(topic) %>%
      top_n(10) %>%
      ungroup %>%
      mutate(topic = factor(paste("Topic", topic)),
             term  = reorder_within(term, beta, within = topic))
    })

  # create plot of each topic and associated words
  beta_plot <-
    ggplot(plotting_data, aes(term, beta, fill = topic)) +
    geom_col(alpha = 0.7, show.legend = FALSE) +
    facet_wrap(~ topic, scales = "free_y") +
    scale_fill_manual(values = pal) +
    scale_x_reordered() +
    coord_flip() +
    ylab("Importance") +
    xlab("Term") +
    theme_pubclean(base_family = "Century")

  # print plot out
  return(beta_plot)
}

plot_stm_gamma <- function(topic_model) {
  # create dataframe with topic and gamma weight
  td_gamma <- broom::tidy(topic_model,
                          matrix = "gamma",
                          document_names = rownames(tweets_dfm))

  # grab probabilites that each topic is
  # associated with X number of languages
  plotting_data <-
    td_gamma %>%
    mutate(topic = factor(paste("Topic", topic)))

  # create plot
  gamma_plot <-
    ggplot(plotting_data, aes(gamma, fill = topic)) +
    geom_histogram(alpha = 0.7, show.legend = FALSE, bins = 30) +
    facet_wrap(~ topic) +
```

```
    scale_y_continuous(breaks = c(1,2, 3)) +
    scale_fill_manual(values = pal) +
    xlab("Probability") +
    ylab("Number of Communities") +
    theme_pubclean(base_family = "Century") +
    theme(axis.text.x = element_text())
  # print plot
  return(gamma_plot)
}
```

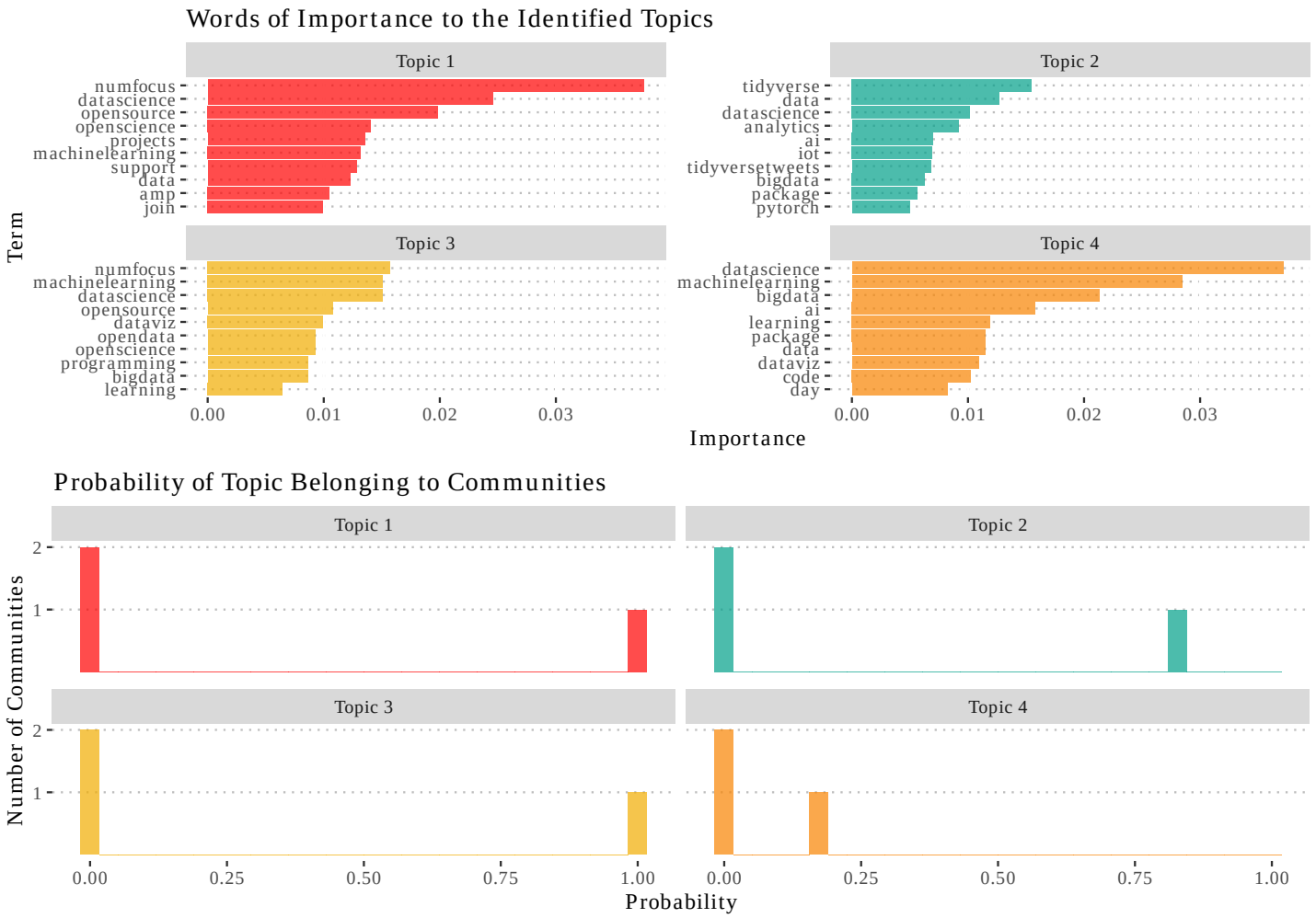
3 Topics

```
pb3 <- plot_stm_beta(stm.3) + labs(title = "Words of Importance to the Identified Topics")
pg3 <- plot_stm_gamma(stm.3) + labs(title = "Probability of Topic Belonging to Communities")
plot_grid(pb3, pg3, nrow = 2)
```



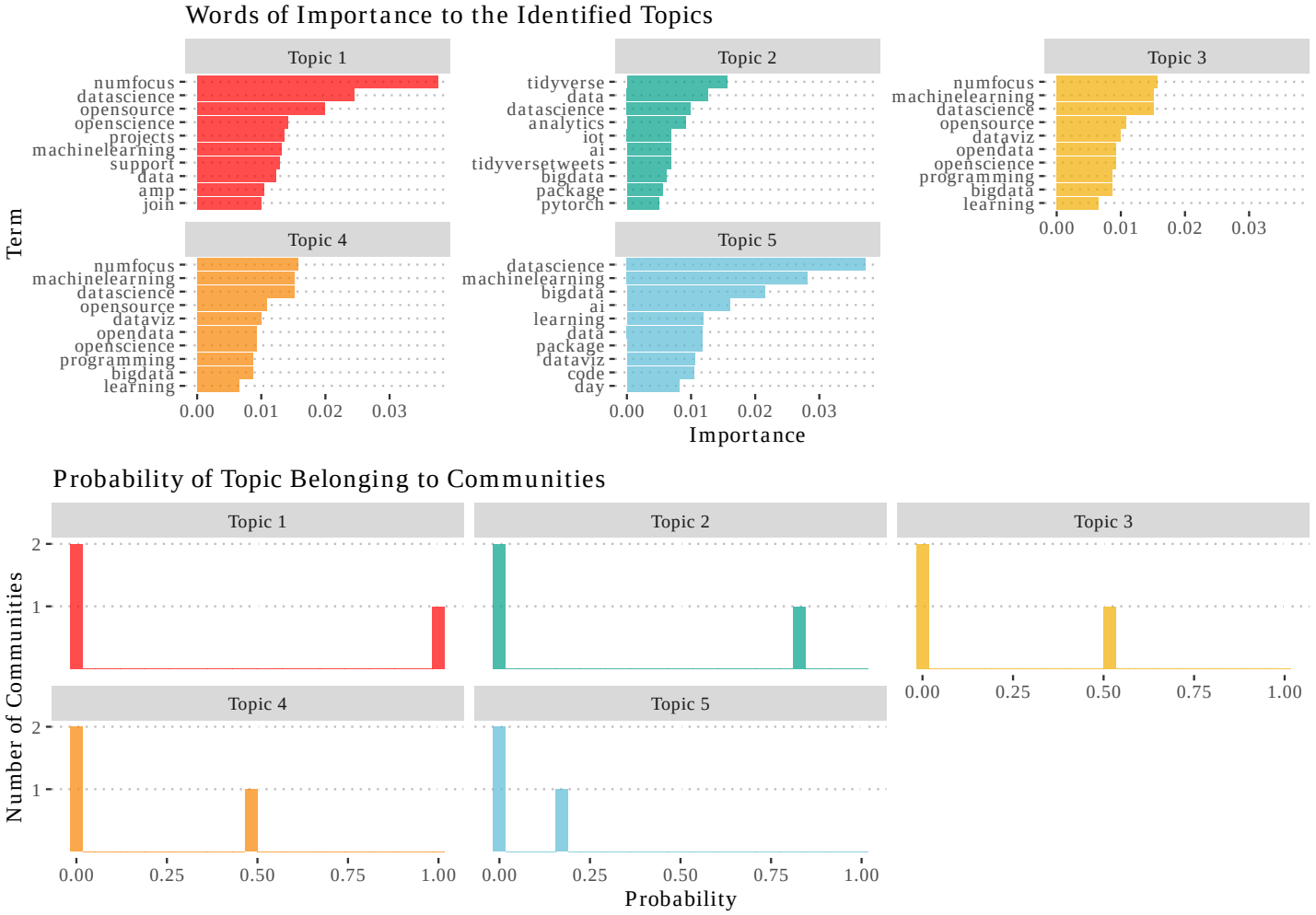
4 Topics

```
pb4 <- plot_stm_beta(stm.4) + labs(title = "Words of Importance to the Identified Topics")
pg4 <- plot_stm_gamma(stm.4) + labs(title = "Probability of Topic Belonging to Communities")

plot_grid(pb4, pg4, nrow = 2)
```



5 Topics

```
pb5 <- plot_stm_beta(stm.5) + labs(title = "Words of Importance to the Identified Topics")
pg5 <- plot_stm_gamma(stm.5) + labs(title = "Probability of Topic Belonging to Communities")

plot_grid(pb5, pg5, nrow = 2)
```



Conclusions

This project aimed to identify differences in language among online Data Science communities. Tweets were scraped from hashtags corresponding to online communities of the `R`, `Python`, and `Julia` programming languages, and their contents were analyzed. Overall, these communities seem to be slightly distinct, and share a lot of overlap in the type of language that is used, the sentiment of the language, and the topics which are discussed. Due to my subjective experience interacting with these communities online, I expected the R online community to be more positive and distinct than the Python and Julia online communities. I was surprised to find that R was much more similar to these communities across all aspects of the analysis. The Python and Julia communities used more positive language than expected. Perhaps users in these communities are all similar people to having shared interests in programming, data, and statistic, and we should all just get along.