practicum_2

Narges Yarahmadi Gharaei

2023-06-04

```
knitr::opts_chunk$set(echo = TRUE)
library(readr)
library(dplyr)
```

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

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

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

```
library(tidyr)
library(ggplot2)
library(scales)
```

```
##
## Attaching package: 'scales'

## The following object is masked from 'package:readr':
##
##     col_factor
```

```
library(tidytext)
library(textstem)
```

```
## Loading required package: koRpus.lang.en

## Loading required package: koRpus

## Loading required package: sylly

## For information on available language packages for 'koRpus', run
##
##   available.koRpus.lang()
##
## and see ?install.koRpus.lang()
```

```
##
## Attaching package: 'koRpus'

## The following object is masked from 'package:readr':
##
##     tokenize
```

```
library(clinspacy)
```

```
## Welcome to clinspacy.

## By default, this package will install and use miniconda and create a "clinspacy" conda environment.

## If you want to override this behavior, use clinspacy_init(miniconda = FALSE) and specify an alternati
```

```
library(topicmodels)
library('reshape2')
```

```
##
## Attaching package: 'reshape2'

## The following object is masked from 'package:tidyr':
##
##     smiths
```

```
library(stringr)
```

This practical is based on exploratory data analysis, named entity recognition, and topic modelling of unstructured medical note free-text data derived from electronic medical records (EMR). Real EMR data is very difficult to access without a specific need/request so this data set is derived from medical transcription data instead. I'll also caveat that the options of natural language processing (NLP) in R are far inferior to those available in Python.

First, install the packages in the setup block (`install.packages(c("readr", "dplyr", "tidyr", "ggplot2", "tidtext", "textstem", "clinspacy", "topicmodels", "reshape2"))`).

Note: To try and make it clearer which library certain functions are coming from clearer, I'll try to do explicit imports throughout this notebook.

Data Parsing

After that we can grab the dataset directly from the `clinspacy` library.

```
raw.data <- clinspacy::dataset_mtsamples()
dplyr::glimpse(raw.data)
```

```
## Rows: 4,999
## Columns: 6
## $ note_id           <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 1~
## $ description       <chr> "A 23-year-old white female presents with complaint ~
## $ medical_specialty <chr> "Allergy / Immunology", "Bariatrics", "Bariatrics", ~
## $ sample_name       <chr> "Allergic Rhinitis", "Laparoscopic Gastric Bypass Co~
## $ transcription     <chr> "SUBJECTIVE:,  This 23-year-old white female present~
## $ keywords          <chr> "allergy / immunology, allergic rhinitis, allergies,~
```

```
raw.data$transcription[[175]]
```

```
## [1] "PREOPERATIVE DIAGNOSES:,1.  Hallux rigidus, left foot.,2.  Elevated first metatarsal, left foot
```

There is no explanation or data dictionary with this dataset, which is a surprisingly common and frustrating turn of events!

1 Using the output of dplyr's `glimpse` command (or rstudio's data viewer by clicking on `raw.data` in the Environment pane) provide a description of what you think each variable in this dataset contains.

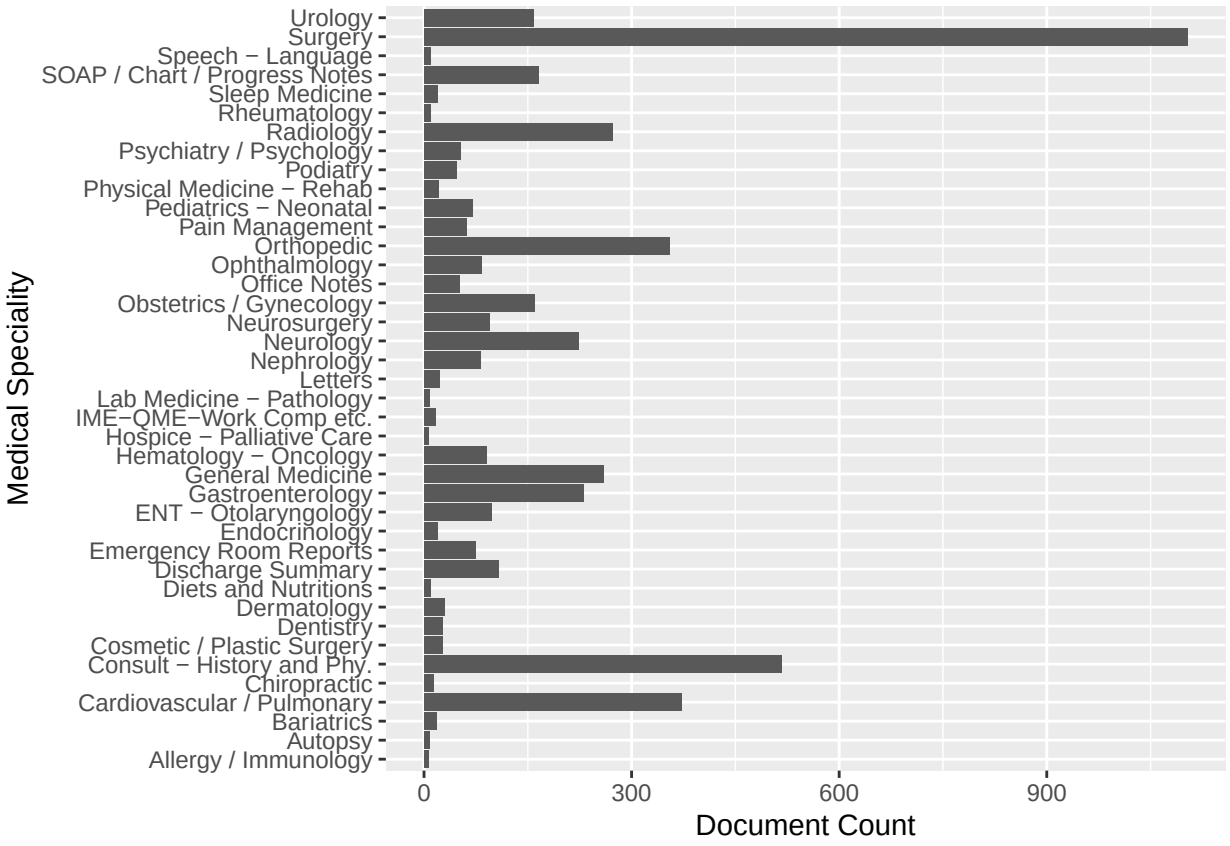
Let's see how many different medical specialties are featured in these notes:

```
raw.data %>% dplyr::select(medical_specialty) %>% dplyr::n_distinct()
```

```
## [1] 40
```

So, how many transcripts are there from each specialty:

```
ggplot2::ggplot(raw.data, ggplot2::aes(y=medical_specialty)) + ggplot2::geom_bar() + labs(x="Document C
```



Let's make our life easier and filter down to 3 specialties: a diagonstic/lab, a medical, and a surgical specialty

```
filtered.data <- raw.data %>% dplyr::filter(medical_specialty %in% c("Orthopedic", "Radiology", "Surgery
```

Text Processing

Let's now apply our standard pre-processing to the transcripts from these specialties.
We are going to use the `tidytext` package to tokenise the transcript free-text.
Let's remove stop words first. e.g., "the", "of", "to", and so forth. These are known as stop words and we can remove them relative easily using a list from `tidytext::stop_words` and `dplyr::anti_join()`

```
analysis.data <- filtered.data %>%
  unnest_tokens(word, transcription) %>%
  mutate(word = str_replace_all(word, "[^[:alnum:]]", "")) %>%
  filter(!str_detect(word, "[0-9]")) %>%
  anti_join(stop_words) %>%
  group_by(note_id) %>%
  summarise(transcription = paste(word, collapse = " ")) %>%
  left_join(select(filtered.data, -transcription), by = "note_id")
```

```
## Joining with `by = join_by(word)`
```

Now let's tokenize the `transcription` to words (unigram) By default this tokenises to words but other options include characters, n-grams, sentences, lines, paragraphs, or separation around a regular expression.

```
tokenized.data.unigram <- analysis.data %>% tidytext::unnest_tokens(word, transcription, to_lower=TRUE)
tokenized.data.unigram
```

```
## # A tibble: 401,835 x 6
##    note_id description              medical_specialty sample_name keywords word
##      <int> <chr>                    <chr>             <chr>       <chr>    <chr>
##  1     175 Austin & Youngswick bun~ Surgery           Youngswick~ surgery~ preo~
##  2     175 Austin & Youngswick bun~ Surgery           Youngswick~ surgery~ diag~
##  3     175 Austin & Youngswick bun~ Surgery           Youngswick~ surgery~ hall~
##  4     175 Austin & Youngswick bun~ Surgery           Youngswick~ surgery~ rigi~
##  5     175 Austin & Youngswick bun~ Surgery           Youngswick~ surgery~ left
##  6     175 Austin & Youngswick bun~ Surgery           Youngswick~ surgery~ foot
##  7     175 Austin & Youngswick bun~ Surgery           Youngswick~ surgery~ elev~
##  8     175 Austin & Youngswick bun~ Surgery           Youngswick~ surgery~ meta~
##  9     175 Austin & Youngswick bun~ Surgery           Youngswick~ surgery~ left
## 10     175 Austin & Youngswick bun~ Surgery           Youngswick~ surgery~ foot
## # i 401,825 more rows
```

You can also do bi-grams

```
tokenized.data <- analysis.data %>% tidytext::unnest_tokens(ngram, transcription, token = "ngrams", n=2
tokenized.data
```

```
## # A tibble: 400,121 x 6
##    note_id description              medical_specialty sample_name keywords ngram
##      <int> <chr>                    <chr>             <chr>       <chr>    <chr>
##  1     175 Austin & Youngswick bun~ Surgery           Youngswick~ surgery~ preo~
##  2     175 Austin & Youngswick bun~ Surgery           Youngswick~ surgery~ diag~
##  3     175 Austin & Youngswick bun~ Surgery           Youngswick~ surgery~ hall~
##  4     175 Austin & Youngswick bun~ Surgery           Youngswick~ surgery~ rigi~
##  5     175 Austin & Youngswick bun~ Surgery           Youngswick~ surgery~ left~
```

```
##  6     175 Austin & Youngswick bun~ Surgery           Youngswick~ surgery~ foot~
##  7     175 Austin & Youngswick bun~ Surgery           Youngswick~ surgery~ elev~
##  8     175 Austin & Youngswick bun~ Surgery           Youngswick~ surgery~ meta~
##  9     175 Austin & Youngswick bun~ Surgery           Youngswick~ surgery~ left~
## 10     175 Austin & Youngswick bun~ Surgery           Youngswick~ surgery~ foot~
## # i 400,111 more rows
```

2 How many unique unigrams are there in the transcripts from each specialty:

```
tokenized.data.unigram %>%
  dplyr::group_by(medical_specialty) %>%
  dplyr::distinct(word) %>%
  dplyr::summarise(n=dplyr::n())
```

```
## # A tibble: 3 x 2
##   medical_specialty     n
##   <chr>             <int>
## 1 Orthopedic         7682
## 2 Radiology          5935
## 3 Surgery           11977
```

For bigram

```
tokenized.data %>% dplyr::group_by(medical_specialty) %>% dplyr::distinct(ngram) %>% dplyr::summarise(n=
```

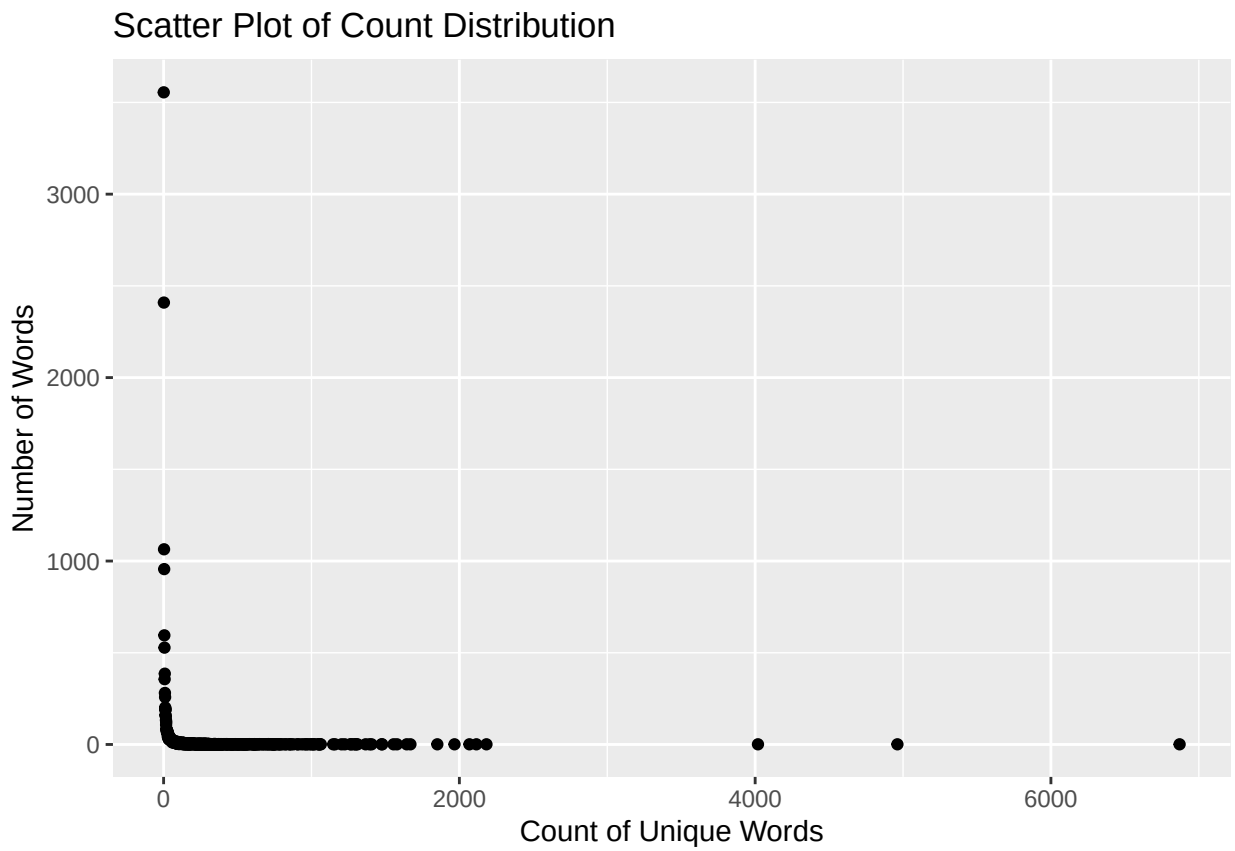
```
## # A tibble: 3 x 2
##   medical_specialty      n
##   <chr>              <int>
## 1 Orthopedic         55732
## 2 Radiology          28297
## 3 Surgery           130404
```

Let's plot some distribution of unigram tokens (words)

```
word_counts <- tokenized.data.unigram %>%
    group_by(word) %>%
    summarise(count = n()) %>%
    ungroup() %>%
    arrange(desc(count))

count_distribution <- word_counts %>%
  group_by(count) %>%
  summarise(num_words = n()) %>%
  ungroup()

 ggplot2::ggplot(count_distribution, aes(x = count, y = num_words)) +
  geom_point() +
  labs(title = "Scatter Plot of Count Distribution",
       x = "Count of Unique Words",
       y = "Number of Words")
```

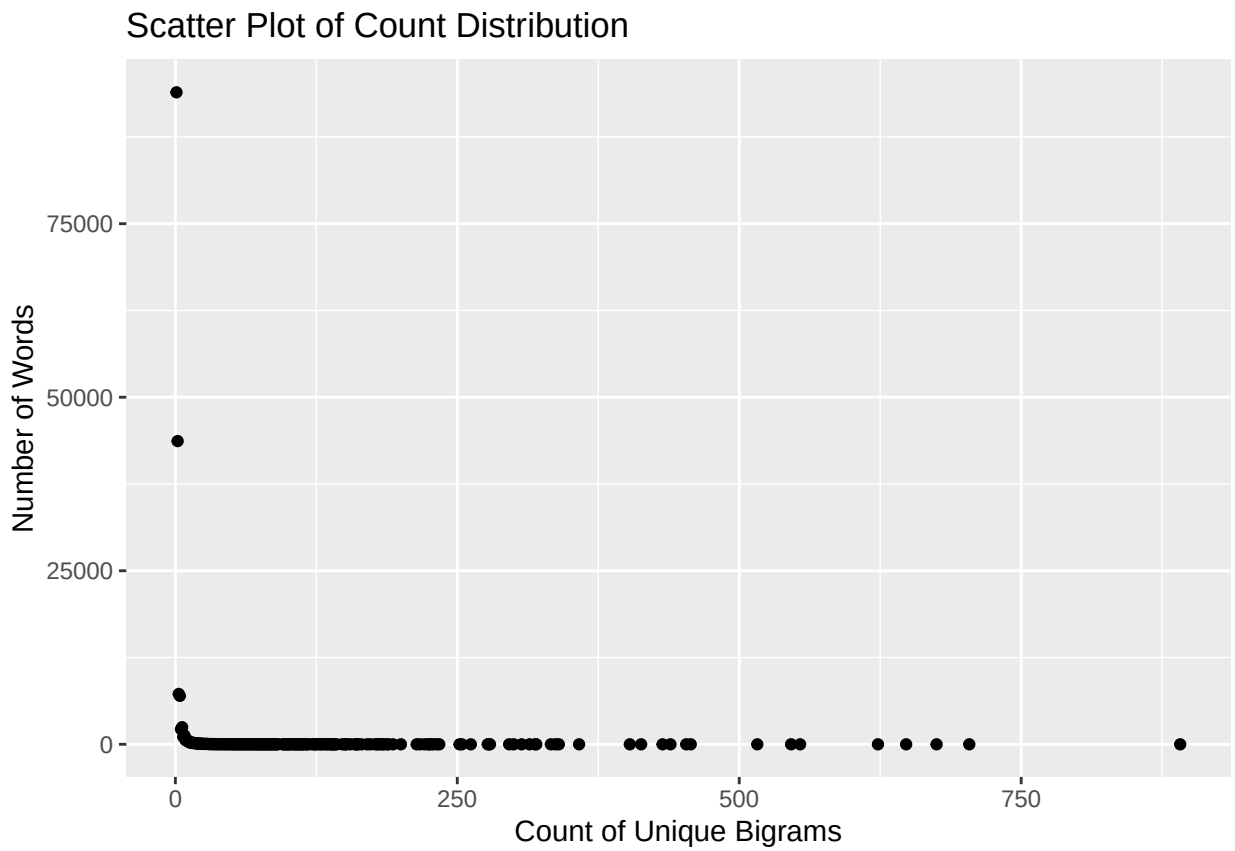


Let's plot some distribution of bigram tokens (words)

```
word_counts <- tokenized.data %>%
    group_by(ngram) %>%
    summarise(count = n()) %>%
    ungroup() %>%
    arrange(desc(count))

count_distribution <- word_counts %>%
  group_by(count) %>%
  summarise(num_words = n()) %>%
  ungroup()

 ggplot2::ggplot(count_distribution, aes(x = count, y = num_words)) +
  geom_point() +
  labs(title = "Scatter Plot of Count Distribution",
       x = "Count of Unique Bigrams",
       y = "Number of Words")
```



3 How many unique bi-grams are there in each category without stop words and numbers?

```
unique_bigrams <- tokenized.data %>%
  filter(!str_detect(ngram, "\\b(?:stopword1|stopword2|stopword3|...)\\b")) %>%
  filter(!str_detect(ngram, "[0-9]")) %>%
  group_by(medical_specialty) %>%
  distinct(ngram) %>%
  summarise(unique_bigrams = n())

print(unique_bigrams)
```

```
## # A tibble: 3 x 2
##   medical_specialty unique_bigrams
##   <chr>                      <int>
## 1 Orthopedic                 50401
## 2 Radiology                  24116
## 3 Surgery                   119664
```

```
unique_bigrams <- tokenized.data %>%
  dplyr::group_by(medical_specialty) %>%
  group_by(medical_specialty) %>%
  distinct(ngram) %>%
  summarise(unique_bigrams = n())

print(unique_bigrams)
```

```
## # A tibble: 3 x 2
##   medical_specialty unique_bigrams
##   <chr>                      <int>
## 1 Orthopedic                 55732
## 2 Radiology                  28297
## 3 Surgery                   130404
```

Sometimes we are interested in tokenising/segmenting things other than words like whole sentences or paragraphs.
4 How many unique sentences are there in each category? Hint: use `?tidytext::unnest_tokens` to see the documentation for this function.

```
unique_sentences <- analysis.data %>%
  group_by(medical_specialty) %>%
  mutate(sentence = strsplit(transcription, "\\.")) %>%
  tidyr::unnest(sentence) %>%
  distinct(sentence) %>%
  summarise(unique_sentences = n())

print(unique_sentences)
```

```
## # A tibble: 3 x 2
##   medical_specialty unique_sentences
##   <chr>                        <int>
## 1 Orthopedic                     354
## 2 Radiology                      273
## 3 Surgery                       1087
```

Now that we've tokenized to words and removed stop words, we can find the most commonly word used within each category:

```
tokenized.data %>%
  dplyr::group_by(medical_specialty) %>%
  dplyr::count(ngram, sort = TRUE) %>%
  dplyr::top_n(5)
```

```
## Selecting by n
```

```
## # A tibble: 16 x 3
## # Groups:   medical_specialty [3]
##    medical_specialty ngram                       n
##    <chr>             <chr>                   <int>
##  1 Surgery           prepped draped            696
##  2 Surgery           preoperative diagnosis    555
##  3 Surgery           procedure patient         551
##  4 Surgery           postoperative diagnosis   518
##  5 Surgery           tolerated procedure       515
##  6 Orthopedic        prepped draped            183
##  7 Orthopedic        preoperative diagnosis    141
##  8 Orthopedic        lower extremity           139
##  9 Orthopedic        range motion              139
## 10 Orthopedic        postoperative diagnosis   124
```

```
## 11 Radiology         carotid artery            59
## 12 Radiology         heart rate                52
## 13 Radiology         reason exam               51
## 14 Radiology         left ventricular          50
## 15 Radiology         coronary artery           43
## 16 Radiology         exam unremarkable         43
```

We should lemmatize the tokenized words to prevent over counting of similar words before further analyses. Annoyingly, `tidytext` doesn't have a built-in lemmatizer.

5 Do you think a general purpose lemmatizer will work well for medical data? Why might it not?

Using a general-purpose lemmatizer for medical data may have limitations and might not work as effectively as a specialized lemmatizer designed for medical terminology. Here are a few reasons why a general-purpose lemmatizer may not perform optimally for medical data:

Specialized Medical Terminology: Medical data often contains domain-specific terms, abbreviations, and acronyms that may not be present in a general-purpose lemmatizer's dictionary. These specialized medical terms require a domain-specific understanding to accurately lemmatize them.

Contextual Ambiguity: Medical texts can include terms that have different meanings depending on the context. For example, the word "heart" can refer to an organ, a cardiac condition, or as a metaphorical expression. A general-purpose lemmatizer may not consider the context and could produce incorrect lemmatization results.

Rare or Uncommon Words: Medical data often contains rare or uncommon words that are specific to certain medical conditions or procedures. These words might not be included in a general-purpose lemmatizer's dictionary, leading to incorrect lemmatization or the omission of certain terms.

Language Variations: Medical texts may include variations in language, such as medical abbreviations, misspellings, or regional variations. A general-purpose lemmatizer may struggle to handle these variations and may not accurately lemmatize the terms.

To address these challenges, specialized lemmatizers or natural language processing tools specifically designed for medical data, such as clinical NLP (Natural Language Processing) libraries or medical ontologies, are often used. These tools consider the specific vocabulary, context, and intricacies of medical language, providing more accurate lemmatization results for medical texts.

While a general-purpose lemmatizer can still be useful to some extent, it's important to be aware of its limitations when working with medical data and consider utilizing specialized tools tailored to the medical domain for better accuracy and performance.

Unfortunately, a specialised lemmatizer like in `clinspacy` is going to be very painful to install so we will just use a simple lemmatizer for now:

```
lemmatized.data <- tokenized.data %>% dplyr::mutate(lemma=textstem::lemmatize_words(ngram))
```

```
lemmatized.data
```

```
## # A tibble: 400,121 x 7
##    note_id description        medical_specialty sample_name keywords ngram lemma
##      <int> <chr>              <chr>             <chr>       <chr>    <chr> <chr>
##  1     175 Austin & Youngswi~ Surgery           Youngswick~ surgery~ preo~ preo~
##  2     175 Austin & Youngswi~ Surgery           Youngswick~ surgery~ diag~ diag~
##  3     175 Austin & Youngswi~ Surgery           Youngswick~ surgery~ hall~ hall~
##  4     175 Austin & Youngswi~ Surgery           Youngswick~ surgery~ rigi~ rigi~
##  5     175 Austin & Youngswi~ Surgery           Youngswick~ surgery~ left~ left~
##  6     175 Austin & Youngswi~ Surgery           Youngswick~ surgery~ foot~ foot~
```

```
##  7     175 Austin & Youngswi~ Surgery           Youngswick~ surgery~ elev~ elev~
##  8     175 Austin & Youngswi~ Surgery           Youngswick~ surgery~ meta~ meta~
##  9     175 Austin & Youngswi~ Surgery           Youngswick~ surgery~ left~ left~
## 10     175 Austin & Youngswi~ Surgery           Youngswick~ surgery~ foot~ foot~
## # i 400,111 more rows
```

We can now calculate the frequency of lemmas within each specialty and note.

```
lemma.freq <- lemmatized.data %>%
  dplyr::count(medical_specialty, lemma) %>%
  dplyr::group_by(medical_specialty) %>%
  dplyr::mutate(proportion = n / sum(n)) %>%
  tidyr::pivot_wider(names_from = medical_specialty, values_from = proportion) %>%
  tidyr::pivot_longer(`Surgery`:`Radiology`,
               names_to = "medical_specialty", values_to = "proportion")
```

```
sorted_lemma.freq <- lemma.freq %>%
  arrange(desc(proportion))

top_20_lemmas <- sorted_lemma.freq %>%
  top_n(20)
```

```
## Selecting by proportion
```

```
print(top_20_lemmas)
```

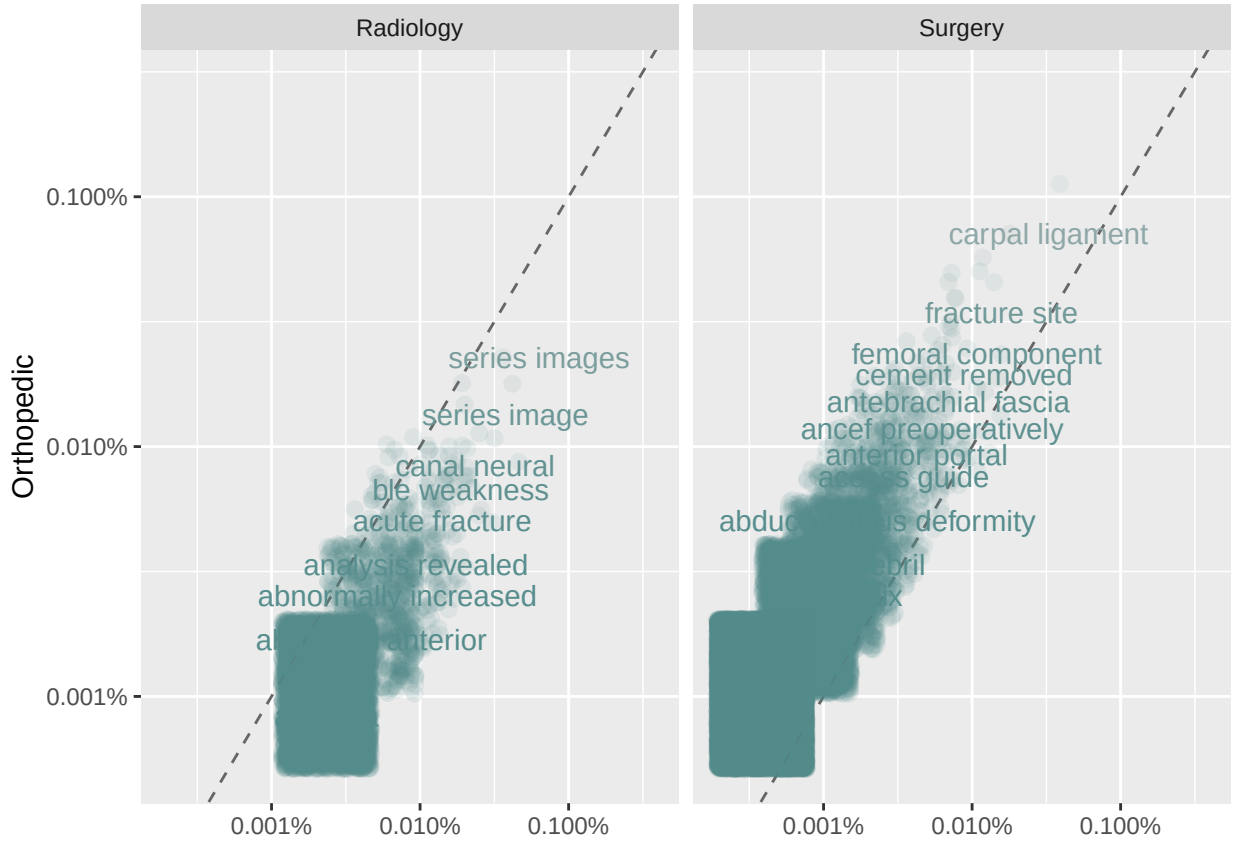
```
## # A tibble: 20 x 5
##    lemma                      n Orthopedic medical_specialty proportion
##    <chr>                  <int>      <dbl> <chr>                   <dbl>
##  1 prepped draped           696         NA Surgery               0.00268
##  2 preoperative diagnosis   555         NA Surgery               0.00214
##  3 procedure patient        551         NA Surgery               0.00212
##  4 postoperative diagnosis  518         NA Surgery               0.00199
##  5 tolerated procedure      515         NA Surgery               0.00198
##  6 patient tolerated        457         NA Surgery               0.00176
##  7 blood loss               439         NA Surgery               0.00169
##  8 draped usual             392         NA Surgery               0.00151
##  9 stable condition         370         NA Surgery               0.00142
## 10 carotid artery            59         NA Radiology             0.00139
## 11 estimated blood          353         NA Surgery               0.00136
## 12 supine position          350         NA Surgery               0.00135
## 13 procedure performed      338         NA Surgery               0.00130
## 14 sterile fashion          334         NA Surgery               0.00129
## 15 patient brought          332         NA Surgery               0.00128
## 16 heart rate                52         NA Radiology             0.00122
## 17 reason exam               51         NA Radiology             0.00120
## 18 left ventricular          50         NA Radiology             0.00117
## 19 informed consent         304         NA Surgery               0.00117
## 20 coronary artery          295         NA Surgery               0.00114
```

And plot the relative proportions

```
ggplot2::ggplot(lemma.freq, ggplot2::aes(x=proportion,
                                         y=`Orthopedic`,
                                         color=abs(`Orthopedic` - proportion))) +
  ggplot2::geom_abline(color="gray40", lty=2) +
  ggplot2::geom_jitter(alpha=0.1, size=2.5, width=0.3, height=0.3) +
  ggplot2::geom_text(ggplot2::aes(label=lemma), check_overlap=TRUE, vjust=1.5) +
  ggplot2::scale_x_log10(labels=scales::percent_format()) +
  ggplot2::scale_y_log10(labels=scales::percent_format()) +
  ggplot2::scale_color_gradient(limits=c(0, 0.001), low="darkslategray4", high="gray75") +
  ggplot2::facet_wrap(~medical_specialty, ncol = 2) +
  ggplot2::theme(legend.position="none") +
  ggplot2:: labs(y="Orthopedic", x = NULL)
```

```
## Warning: Removed 314404 rows containing missing values (`geom_point()`).
```

```
## Warning: Removed 314404 rows containing missing values (`geom_text()`).
```



6__A What does this plot tell you about the relative similarity of lemma frequencies between Surgery and Orthopedic and between radiology and Surgery?

The plot represents the relative proportions of lemmas in Radiology and Surgery specialties. Here's a closer examination of the plot and its implications:

Scatterplot Distribution: The scattered points on the plot illustrate the distribution of lemma frequencies in both Radiology and Surgery. Each point represents a lemma, and its position corresponds to its proportion

within the respective specialty. The x-axis represents the proportion of each lemma, while the y-axis focuses specifically on the Surgery specialty.

Similarities: Points that cluster around the diagonal line indicate lemmas with similar frequencies in both specialties. These overlapping lemmas suggest shared terminology or concepts that are relevant to both Radiology and Surgery. The closer the points are to the diagonal line, the more similar the lemma frequencies between the two specialties.

Differences: Lemmas that deviate from the diagonal line signify discrepancies in their frequencies between Radiology and Surgery. These points represent terms that are more prevalent in one specialty compared to the other. The further away the points are from the diagonal, the greater the difference in lemma frequencies between the specialties.

Interpretation: By analyzing the plot, we can gain insights into the relative similarity of lemma frequencies between Radiology and Surgery. The distribution of points indicates that while there are some shared lemmas, there are also significant differences in lemma frequencies between the two specialties. This suggests that Radiology and Surgery have distinct terminologies and linguistic characteristics.

Considering the nature of Radiology, which focuses on medical imaging and diagnostics, and Surgery, which encompasses various surgical procedures, it is expected to have both common and distinct terminologies. Radiology may have specific lemmas related to imaging techniques, anatomical structures, or diagnostic findings, while Surgery may have lemmas associated with surgical procedures, instruments, or post-operative care.

In conclusion, the plot provides insights into the relative similarity of lemma frequencies between Radiology and Surgery. It showcases the overlap and discrepancies in lemma usage, reflecting the distinct linguistic characteristics and terminology of each specialty.

6__B Based on what these specialties involve, is this what you would expect?

Based on the general understanding of the specialties involved, it is somewhat expected to observe both similarities and differences in lemma frequencies between Radiology and Surgery, as indicated by the plot.

Similarities: The presence of shared lemmas with similar frequencies suggests that there are common terminologies and concepts that are relevant to both Radiology and Surgery. This can be attributed to their interconnected nature in the medical field, where radiological imaging plays a role in preoperative planning and postoperative evaluation. For example, terms related to anatomical structures or general medical terminology may be shared between the two specialties.

Differences: The discrepancies in lemma frequencies between Radiology and Surgery are also expected due to the specific focus and practices of each specialty. Radiology primarily deals with imaging techniques, interpretations, and diagnostic findings. Therefore, it may have lemmas specific to radiological modalities, such as CT scans, MRI, or X-rays, as well as terms related to specific imaging features or abnormalities. On the other hand, Surgery involves a broader range of procedures, surgical techniques, instruments, and postoperative care. Thus, it may have lemmas that are unique to surgical interventions and perioperative management.

Overall, the plot reflects the expected pattern of lemma frequencies between Radiology and Surgery. It demonstrates both the shared terminology and concepts relevant to both specialties, as well as the distinct linguistic characteristics and domain-specific terms associated with each specialty's focus.

7 Modify the above plotting code to do a direct comparison of Surgery and Radiology (i.e., have Surgery or Radiology on the Y-axis and the other 2 specialties as the X facets)

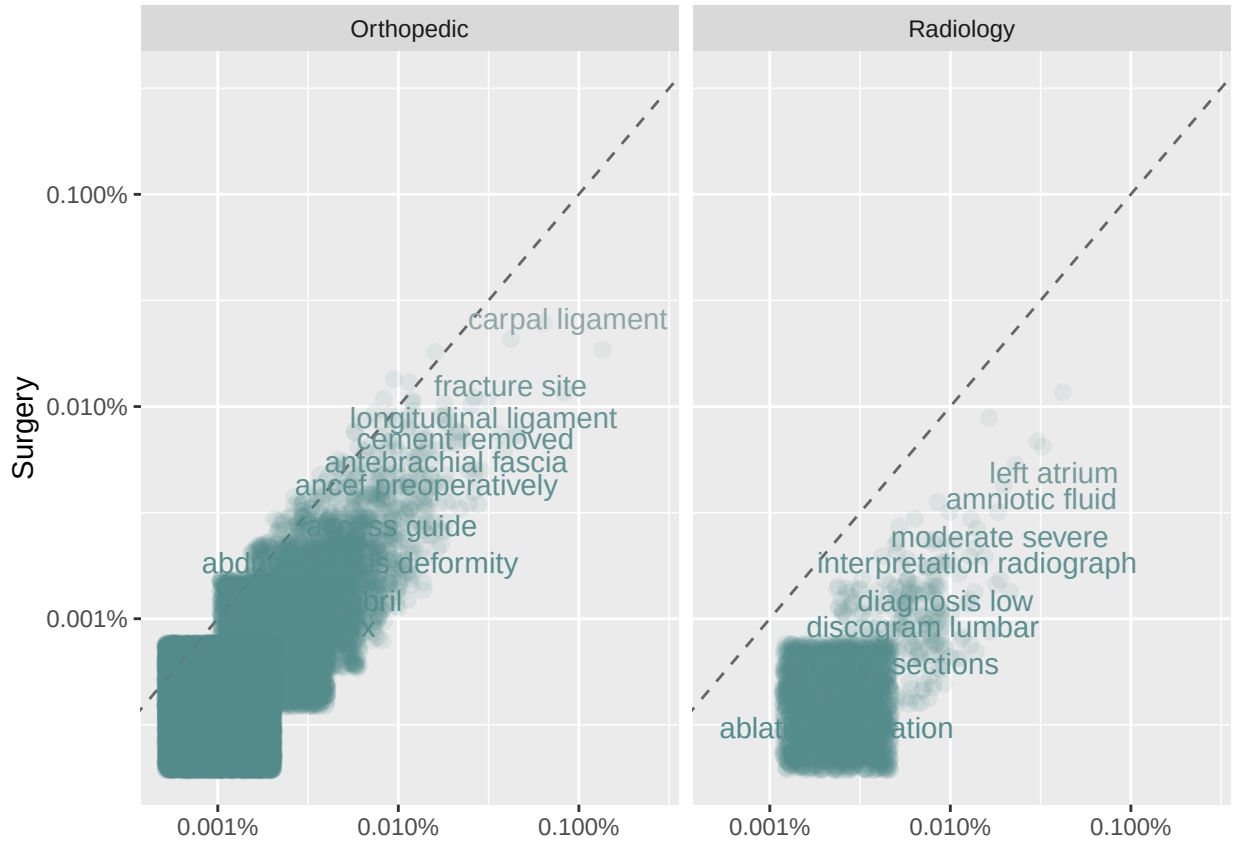
```
lemma.freq <- lemmatized.data %>%
  dplyr::count(medical_specialty, lemma) %>%
  dplyr::group_by(medical_specialty) %>%
  dplyr::mutate(proportion = n / sum(n)) %>%
  tidyr::pivot_wider(names_from = medical_specialty, values_from = proportion) %>%
```

```
  tidyr::pivot_longer(`Orthopedic`:`Radiology`,
               names_to = "medical_specialty", values_to = "proportion")

ggplot2::ggplot(lemma.freq, ggplot2::aes(x=proportion,
                                         y=`Surgery`,
                                         color=abs(`Surgery` - proportion))) +
  ggplot2::geom_abline(color="gray40", lty=2) +
  ggplot2::geom_jitter(alpha=0.1, size=2.5, width=0.3, height=0.3) +
  ggplot2::geom_text(ggplot2::aes(label=lemma), check_overlap=TRUE, vjust=1.5) +
  ggplot2::scale_x_log10(labels=scales::percent_format()) +
  ggplot2::scale_y_log10(labels=scales::percent_format()) +
  ggplot2::scale_color_gradient(limits=c(0, 0.001), low="darkslategray4", high="gray75") +
  ggplot2::facet_wrap(~medical_specialty, ncol = 2) +
  ggplot2::theme(legend.position="none") +
  ggplot2:: labs(y="Surgery", x = NULL)
```

```
## Warning: Removed 317819 rows containing missing values (‘geom_point()‘).
```

```
## Warning: Removed 317820 rows containing missing values (‘geom_text()‘).
```




```
lemma.freq
```

```
## # A tibble: 351,352 x 5
##    lemma                n     Surgery medical_specialty proportion
```

```
##    <chr>            <int>       <dbl> <chr>                <dbl>
##  1 aa body              1  0.00000385 Orthopedic         0.0000102
##  2 aa body              1  0.00000385 Radiology         NA
##  3 aa femoral           1  0.00000385 Orthopedic         0.0000102
##  4 aa femoral           1  0.00000385 Radiology         NA
##  5 aa trial             1  0.00000385 Orthopedic         0.0000102
##  6 aa trial             1  0.00000385 Radiology         NA
##  7 abandoned device     1  0.00000385 Orthopedic         0.0000102
##  8 abandoned device     1  0.00000385 Radiology         NA
##  9 abc abc              1 NA          Orthopedic         0.0000102
## 10 abc abc              1 NA          Radiology         NA
## # i 351,342 more rows
```

TF-IDF Normalisation

Maybe looking at lemmas across all notes in a specialty is misleading, what if we look at lemma frequencies across a specialty.

```
lemma.counts <- lemmatized.data %>% dplyr::count(medical_specialty, lemma)
total.counts <- lemma.counts %>%
                      dplyr::group_by(medical_specialty) %>%
                      dplyr::summarise(total=sum(n))

all.counts <- dplyr::left_join(lemma.counts, total.counts)
```

```
## Joining with `by = join_by(medical_specialty)`
```

Now we can calculate the term frequency / invariant document frequency (tf-idf):

```
all.counts.tfidf <- tidytext::bind_tf_idf(all.counts, lemma, medical_specialty, n)
```

We can then look at the top 10 lemma by tf-idf within each specialty:

```
all.counts.tfidf %>% dplyr::group_by(medical_specialty) %>% dplyr::slice_max(order_by=tf_idf, n=10)
```

```
## # A tibble: 30 x 7
## # Groups:   medical_specialty [3]
##    medical_specialty lemma                    n total       tf   idf   tf_idf
##    <chr>             <chr>                <int> <int>    <dbl> <dbl>    <dbl>
##  1 Orthopedic        range motion           139 97774 0.00142  0.405 0.000576
##  2 Orthopedic        carpal ligament         85 97774 0.000869 0.405 0.000352
##  3 Orthopedic        transverse carpal       81 97774 0.000828 0.405 0.000336
##  4 Orthopedic        extremity prepped       79 97774 0.000808 0.405 0.000328
##  5 Orthopedic        proximal phalanx        75 97774 0.000767 0.405 0.000311
##  6 Orthopedic        department anesthesia   63 97774 0.000644 0.405 0.000261
##  7 Orthopedic        dissection carried      63 97774 0.000644 0.405 0.000261
##  8 Orthopedic        steri strips            59 97774 0.000603 0.405 0.000245
##  9 Orthopedic        closed vicryl           58 97774 0.000593 0.405 0.000241
## 10 Orthopedic        dressing applied        58 97774 0.000593 0.405 0.000241
## # i 20 more rows
```

8 Are there any lemmas that stand out in these lists? Why? Orthopedic: range motion , carpal ligament

Radiology: myocardial perfusion, motor units

Surgery: anterior chamber , lithotomy position

These lemmas stand out because they have higher tf-idf scores within their respective specialties, indicating their relative importance and uniqueness within the specialty. Each specialty has its specific terminology and procedures, and these lemmas reflect the specific language and concepts associated with each field.

For example, in Orthopedic, terms like "range motion," "carpal ligament," and "proximal phalanx" are relevant to orthopedic procedures and conditions. In Radiology, terms like "myocardial perfusion," "calcific plaque," and "perfusion imaging" are related to imaging techniques and findings. In Surgery, terms like "anterior chamber," "lithotomy position," and "suture ligated" are associated with surgical procedures and anatomical structures.

Overall, the identified lemmas align with the specialized knowledge and focus of each specialty. They represent key concepts, anatomical structures, procedures, or conditions that are commonly encountered within the respective medical fields.

We can look at transcriptions using these unusual lemmas to check how they are used with `stringr::str_detect`

```
analysis.data %>% dplyr::select(medical_specialty, transcription) %>% dplyr::filter(stringr::str_detect
```

```
## # A tibble: 1 x 2
##   medical_specialty transcription
##   <chr>             <chr>
## 1 Surgery           preoperative diagnoses thrombosed left forearm loop fistula~
```

9 Extract an example of one of the other unusual "top lemmas" by modifying the above code

first i sort the lemmas decending to see the unusual lemmas

```
all.counts.tfidf %>% dplyr::group_by(medical_specialty) %>% dplyr::slice_max(order_by=desc(tf_idf), n=3
```

```
## # A tibble: 7,872 x 7
## # Groups:   medical_specialty [3]
##    medical_specialty lemma                      n total        tf   idf tf_idf
##    <chr>             <chr>                  <int> <int>     <dbl> <dbl>  <dbl>
##  1 Orthopedic        abdomen evidence           1 97774 0.0000102     0      0
##  2 Orthopedic        abdominal aorta            4 97774 0.0000409     0      0
##  3 Orthopedic        abdominal pain             4 97774 0.0000409     0      0
##  4 Orthopedic        abdominal pelvic           1 97774 0.0000102     0      0
##  5 Orthopedic        abductor hallucis          4 97774 0.0000409     0      0
##  6 Orthopedic        abductovalgus deformity    6 97774 0.0000614     0      0
##  7 Orthopedic        ablation indication        1 97774 0.0000102     0      0
##  8 Orthopedic        abnormalities noted        1 97774 0.0000102     0      0
##  9 Orthopedic        abnormality identified     2 97774 0.0000205     0      0
## 10 Orthopedic        access dye                 1 97774 0.0000102     0      0
## # i 7,862 more rows
```

```
analysis.data %>% dplyr::select(medical_specialty, transcription) %>% dplyr::filter(stringr::str_detect
```

```
## # A tibble: 1 x 2
##   medical_specialty transcription
##   <chr>             <chr>
## 1 Surgery           preoperative diagnosis low syndrome low pain left lower ext~
```

or we can use the following code to find some more unusual lemmas.

```
top_lemmas <- all.counts.tfidf %>%
  dplyr::group_by(medical_specialty) %>%
  dplyr::slice_max(order_by = desc(tf_idf), n = 10)

example_lemma <- top_lemmas %>%
  dplyr::group_by(medical_specialty) %>%
  dplyr::sample_n(size = 1)

example_lemma
```

```
## # A tibble: 3 x 7
## # Groups:   medical_specialty [3]
##   medical_specialty lemma                 n  total        tf   idf tf_idf
##   <chr>             <chr>             <int>  <int>     <dbl> <dbl>  <dbl>
## 1 Orthopedic        patient positioned    11  97774 0.000113      0      0
## 2 Radiology         pain procedure         4  42585 0.0000939     0      0
## 3 Surgery           noted tolerated        4 259762 0.0000154     0      0
```

```
analysis.data %>% dplyr::select(medical_specialty, transcription) %>% dplyr::filter(stringr::str_detect
```

```
## # A tibble: 1 x 2
##   medical_specialty transcription
##   <chr>             <chr>
## 1 Surgery           preoperative diagnosis foraminal disc herniation left posto~
```

Topic Modelling

In NLP, we often have collections of documents (in our case EMR transcriptions) that we'd like to divide into groups so that we can understand them separately. Topic modeling is a method for unsupervised classification of such documents, similar to clustering on numeric data.

Latent Dirichlet allocation (LDA) is a particularly popular method for fitting a topic model. It treats each document as a mixture of topics, and each topic as a mixture of words. This allows documents to "overlap" each other in terms of content, rather than being separated into discrete groups, in a way that mirrors typical use of natural language.

- Every document is a mixture of topics. We imagine that each document may contain words from several topics in particular proportions. For example, in a two-topic model we could say "Document 1 is 90% topic A and 10% topic B, while Document 2 is 30% topic A and 70% topic B."
- Every topic is a mixture of words. For example, we could imagine a two-topic model of American news, with one topic for "politics" and one for "entertainment." The most common words in the politics topic might be "President", "Congress", and "government", while the entertainment topic may be made up of words such as "movies", "television", and "actor". Importantly, words can be shared between topics; a word like "budget" might appear in both equally.

LDA is a mathematical method for estimating both of these at the same time: finding the mixture of words that is associated with each topic, while also determining the mixture of topics that describes each document. There are a number of existing implementations of this algorithm, and we'll explore one of them in depth.

First lets calculate a term frequency matrix for each transcription:

```
lemma.counts <- lemmatized.data %>% dplyr::count(note_id, lemma)
total.counts <- lemma.counts %>%
                      dplyr::group_by(note_id) %>%
                      dplyr::summarise(total=sum(n))

all.counts <- dplyr::left_join(lemma.counts, total.counts)
```

```
## Joining with `by = join_by(note_id)`
```

```
emr.dcm <- all.counts %>% tidytext::cast_dtm(note_id, lemma, n)
```

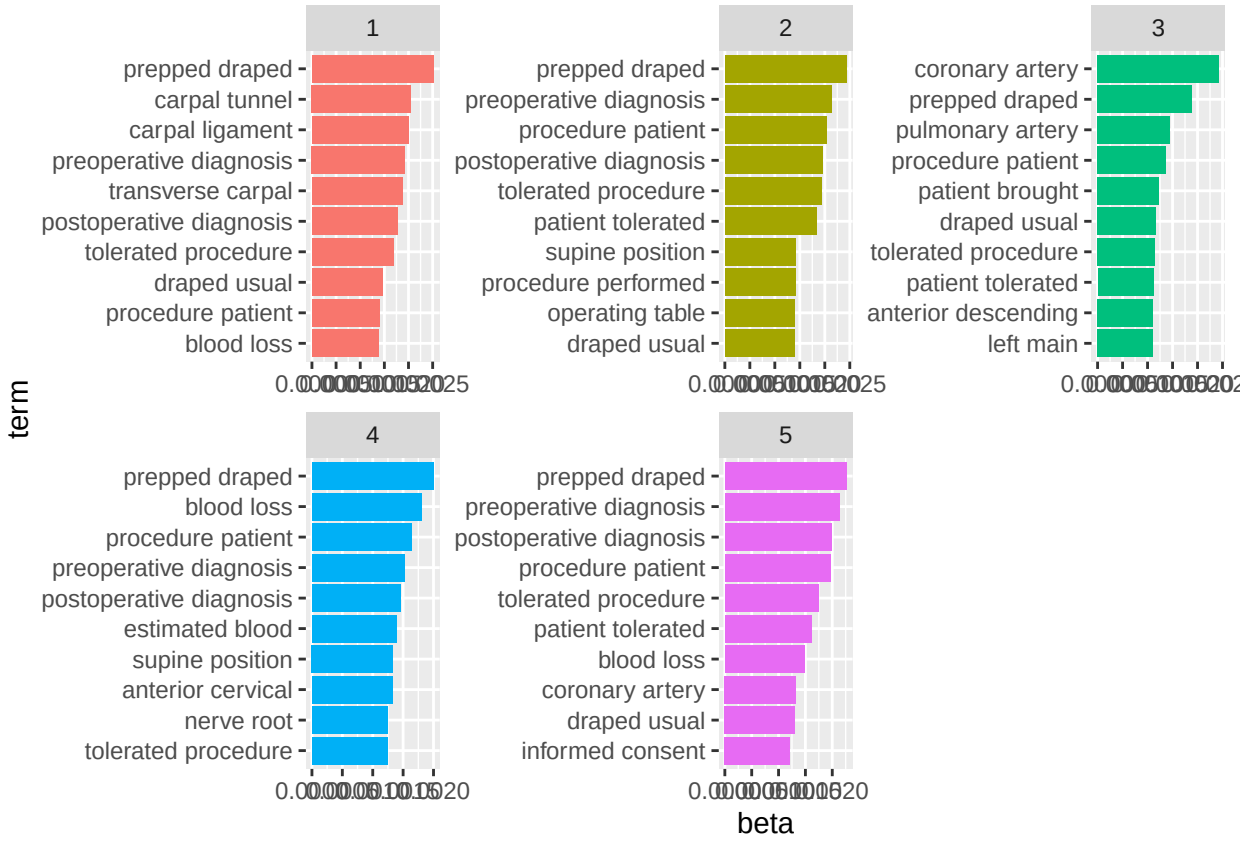
Then we can use LDA function to fit a 5 topic (`k=5`) LDA-model

```
emr.lda <- topicmodels::LDA(emr.dcm, k=5, control=list(seed=42))
emr.topics <- tidytext::tidy(emr.lda, matrix='beta')
```

Then we can extract the top terms per assigned topic:

```
top.terms <- emr.topics %>% dplyr::group_by(topic) %>%
  dplyr::slice_max(beta, n=10) %>%
  dplyr::ungroup() %>%
  dplyr::arrange(topic, -beta)


top.terms %>%
  dplyr::mutate(term=tidytext::reorder_within(term, beta, topic)) %>%
  ggplot2::ggplot(ggplot2::aes(beta, term, fill=factor(topic))) +
    ggplot2::geom_col(show.legend=FALSE) +
    ggplot2::facet_wrap(~ topic, scales='free')  +
    tidytext::scale_y_reordered()
```



Now we can ask how well do these assigned topics match up to the medical specialties from which each of these transcripts was derived.

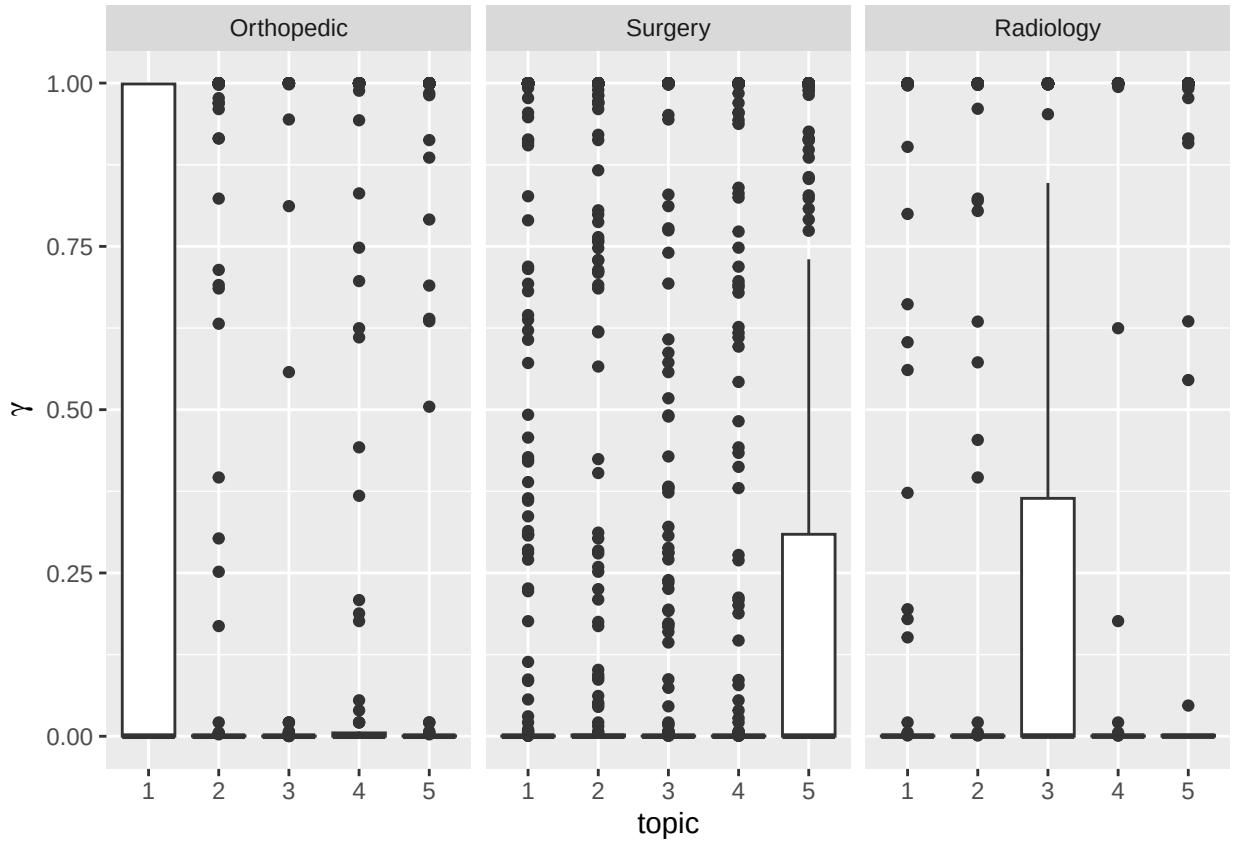
```
specialty_gamma <- tidytext::tidy(emr.lda, matrix='gamma')

# we need to join in the specialty from the note_id
note_id_specialty_mapping <- lemmatized.data %>%
  dplyr::mutate(document=as.character(note_id)) %>%
  dplyr::select(document, medical_specialty) %>%
  dplyr::distinct()

specialty_gamma <- dplyr::left_join(specialty_gamma, note_id_specialty_mapping)
```

```
## Joining with `by = join_by(document)`
```

```
specialty_gamma %>%
  dplyr::mutate(medical_specialty = reorder(medical_specialty, gamma * topic)) %>%
  ggplot2::ggplot(ggplot2::aes(factor(topic), gamma)) +
  ggplot2::geom_boxplot() +
  ggplot2::facet_wrap(~ medical_specialty) +
  ggplot2::labs(x = "topic", y = expression(gamma))
```



Interestingly, Surgery assigns mostly to a single topic but radiology and Orthopedic are both more diverse in transcriptions. We'd possibly expect this from radiology due to referring to imaging for many different diagnoses/reasons. However, this may all just reflect we are using too few topics in our LDA to capture the range of possible assignments.

10 Repeat this with a 6 topic LDA, do the top terms from the 3 topic LDA still turn up? How do the specialties get split into sub-topics?

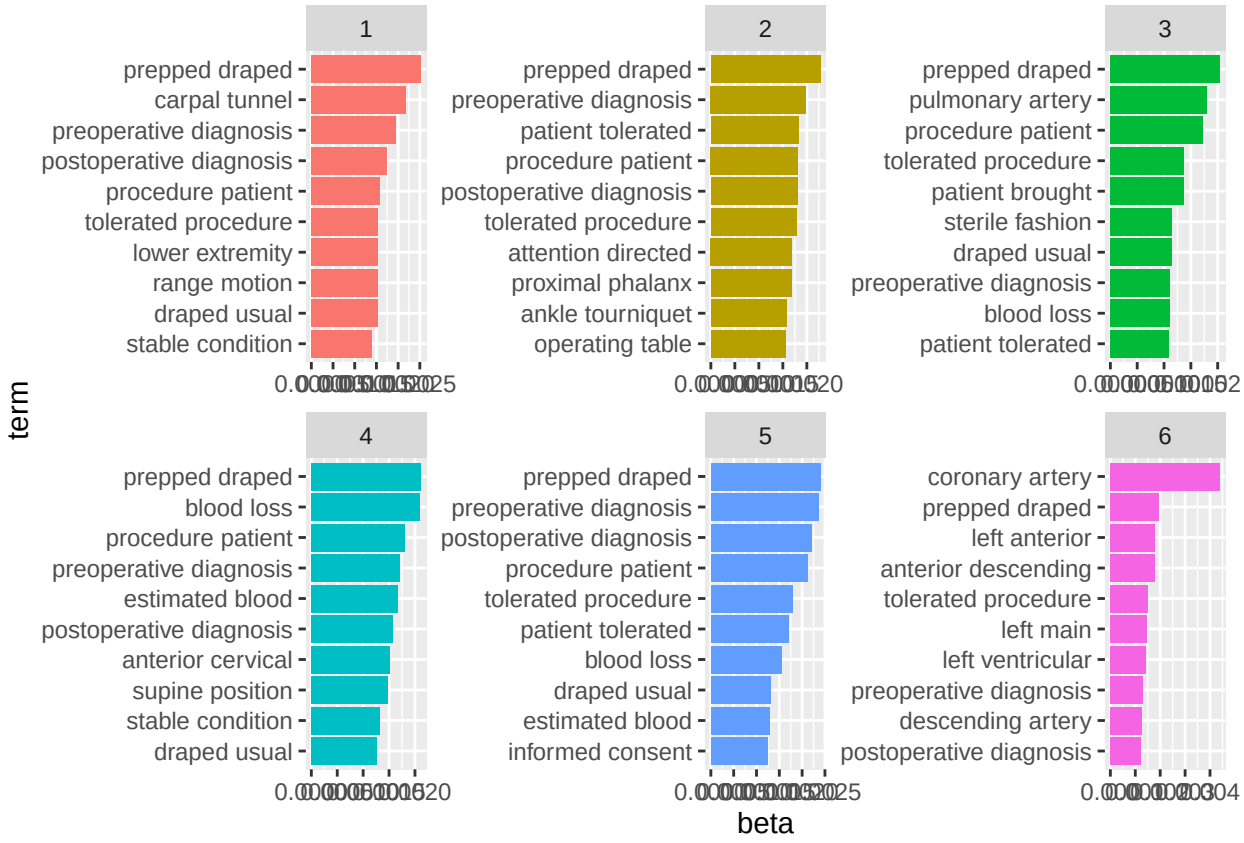
yes we can see some of the top terms of the 3-topic LDA appear among the top terms in the 6 topic LDA. This comparison will provide insights into the stability or variability of certain topics across different LDA models. Additionally, the specialties will be split into sub-topics based on the new 6-topic model. The distribution of top terms across these sub-topics can reveal how the specialties are further differentiated and grouped into more specific thematic areas. This analysis can help identify any emerging sub-topics or new patterns of topic distribution within the medical specialties. in 6-topic LDA we can see the same things of 5-topic LDA ,as the terms has not changed a lot. but by comparing 6-topic LDA with 3-topic LDA,we can find out that as the terms has changed more, the model is getting able to discover more distinct topics.

```
emr.lda_6topics <- topicmodels::LDA(emr.dcm, k=6, control=list(seed=42))
emr.topics_6topics <- tidytext::tidy(emr.lda_6topics, matrix='beta')

top.terms_6topics <- emr.topics_6topics %>% dplyr::group_by(topic) %>%
  dplyr::slice_max(beta, n=10) %>%
  dplyr::ungroup() %>%
  dplyr::arrange(topic, -beta)

top.terms_6topics %>%
  dplyr::mutate(term=tidytext::reorder_within(term, beta, topic)) %>%
```

```
  ggplot2::ggplot(ggplot2::aes(beta, term, fill=factor(topic))) +
    ggplot2::geom_col(show.legend=FALSE) +
    ggplot2::facet_wrap(~ topic, scales='free')  +
    tidytext::scale_y_reordered()
```

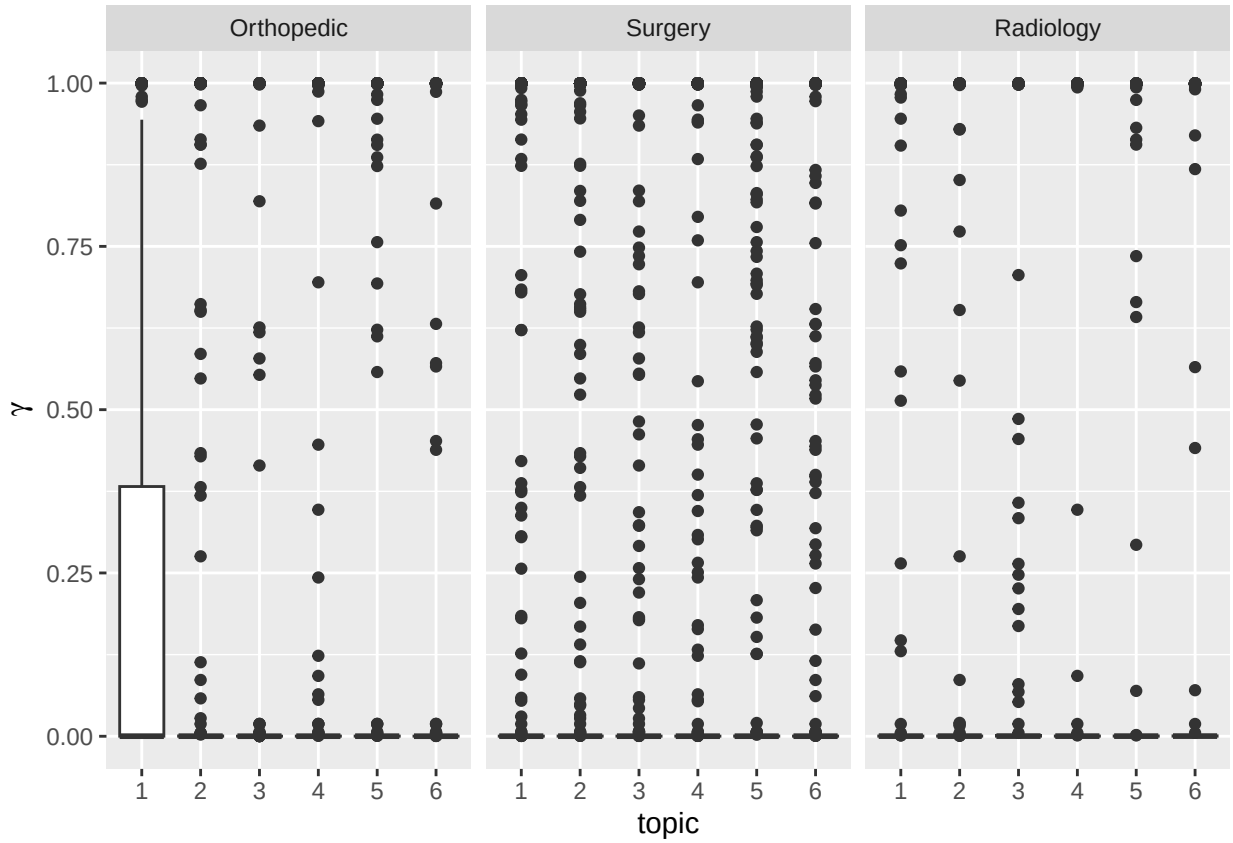



```
specialty_gamma_6topics <- tidytext::tidy(emr.lda_6topics, matrix='gamma')

# Join in the specialty from the note_id
specialty_gamma_6topics <- dplyr::left_join(specialty_gamma_6topics, note_id_specialty_mapping)
```

```
## Joining with `by = join_by(document)`
```

```
specialty_gamma_6topics %>%
  dplyr::mutate(medical_specialty = reorder(medical_specialty, gamma * topic)) %>%
  ggplot2::ggplot(ggplot2::aes(factor(topic), gamma)) +
  ggplot2::geom_boxplot() +
  ggplot2::facet_wrap(~ medical_specialty) +
  ggplot2::labs(x = "topic", y = expression(gamma))
```

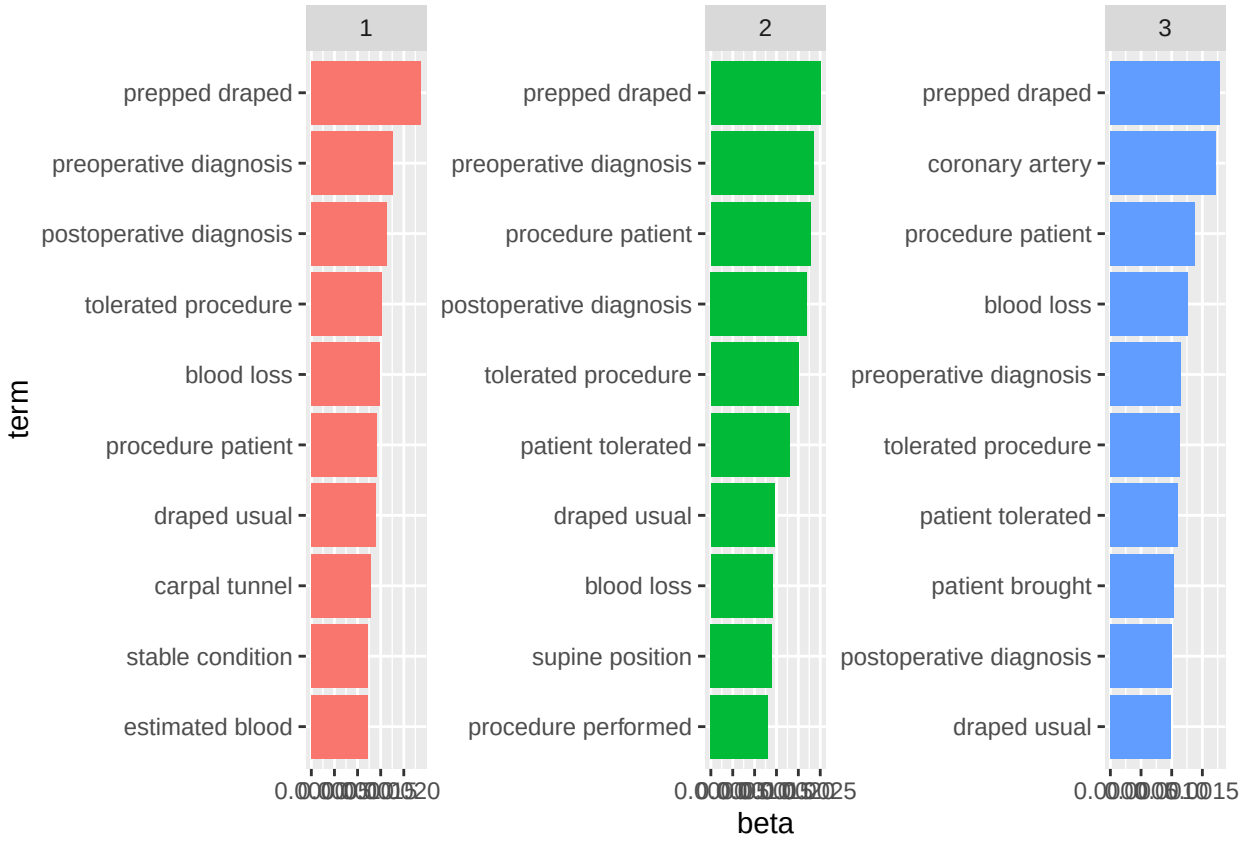


for 3 topic LDA:

```
emr.lda_3topics <- topicmodels::LDA(emr.dcm, k=3, control=list(seed=42))
emr.topics_3topics <- tidytext::tidy(emr.lda_3topics, matrix='beta')

top.terms_3topics <- emr.topics_3topics %>% dplyr::group_by(topic) %>%
  dplyr::slice_max(beta, n=10) %>%
  dplyr::ungroup() %>%
  dplyr::arrange(topic, -beta)

top.terms_3topics %>%
  dplyr::mutate(term=tidytext::reorder_within(term, beta, topic)) %>%
  ggplot2::ggplot(ggplot2::aes(beta, term, fill=factor(topic))) +
    ggplot2::geom_col(show.legend=FALSE) +
    ggplot2::facet_wrap(~ topic, scales='free')  +
    tidytext::scale_y_reordered()
```

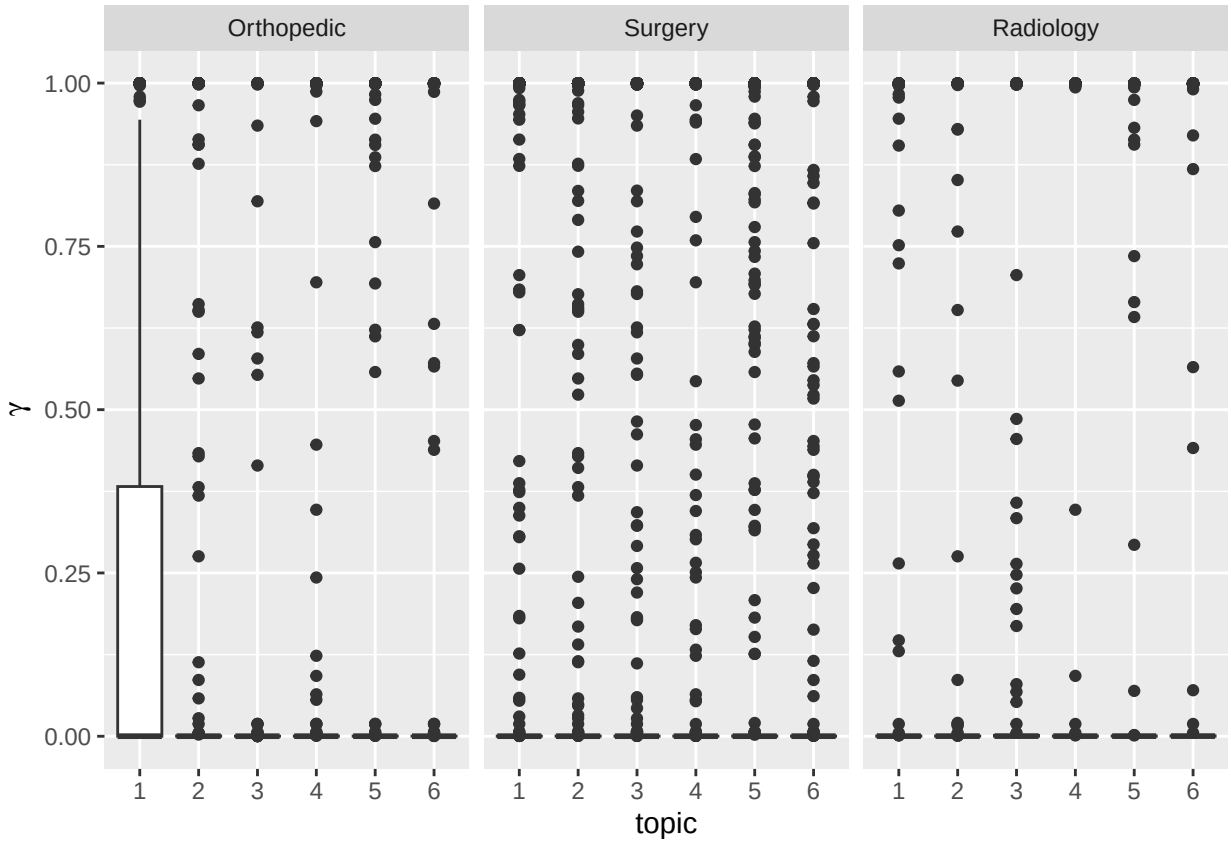



```
specialty_gamma_3topics <- tidytext::tidy(emr.lda_6topics, matrix='gamma')

# Join in the specialty from the note_id
specialty_gamma_3topics <- dplyr::left_join(specialty_gamma_3topics, note_id_specialty_mapping)
```

```
## Joining with `by = join_by(document)`
```

```
specialty_gamma_3topics %>%
  dplyr::mutate(medical_specialty = reorder(medical_specialty, gamma * topic)) %>%
  ggplot2::ggplot(ggplot2::aes(factor(topic), gamma)) +
  ggplot2::geom_boxplot() +
  ggplot2::facet_wrap(~ medical_specialty) +
  ggplot2::labs(x = "topic", y = expression(gamma))
```



Credits

Examples draw heavily on material (and directly quotes/copies text) from Julia Slige's `tidytext` textbook.