You should all go watch Branagh's Hamlet (1996)

Earlier this year I watched Kenneth Branagh's Hamlet (1996) and wow, I cannot recommend this movie enough. Not only is it by far the best Hamlet I have ever seen (on stage or screen), it has a fair claim to being the best Shakespeare 'full stop' (or 'period' as our American cousins say), and makes it to my list of favourite films of any sort. Gorgeously filmed on 70 mm film (including the spectacular setting of Blenheim Palace), 242 minutes long, brilliant direction, amazing atmosphere and ability to surface all the subplots and character quirks. Despite knowing the story backwards – having seen so many renderings of it, and every second line feeling like a quotation built into my cultural DNA – this film moved me in a new ways and depths that I rarely get from any work of art.

And yes, it does have a star-studded cast (perhaps excessively star-studded – Robin Williams and Gerard Depardieu seem wasted, and imagine using John Gielgud and Judi Dench for walk-on non-speaking parts), but the most important thing is that these people can really act. Branagh himself is a faultless tour-de-force, Kate Winslet is perfect as Ophelia, Julie Christie a wonderful Gertrude, and Derek Jacobi's Claudius is spot-on. (Well noticed, observers of quality theatrical film and TV – Derek Jacobi became famous for the title role in I Claudius – but this is unrelated). Richard Briers seems to be made to play Polonius, with the perfect combination of humour, pomposity and pathos.

Some of the greatness of the movie comes from seeing famous stars in what are sometimes seen as also-ran parts in ways that make you see both the greatness of both the role and the actor – Billy Crystal as a gravedigger, and Charlton Heston bringing dignity and depth to the leader of the players.

This movie's interpretation of the story plays up the role and dark aspects of Hamlet senior, whose statue broods over Elsinore and whose relationship to 'our' Hamlet is mirrored in Fortinbras senior and junior… ok I don't have time to explain that here, but well, it's complicated, right? Which is the point.

Anyway, "do yourself a favour" as Molly Meldrum would have said, and see this movie.

Now, I need something data-related if I am going to justify having a blog on this. So I thought I would tie my refreshed enthusiasm for the full text of Hamlet with a familiarisation project that has been on my to-do list for months (understand biterm topic modelling). As it happened, events (in the form of a blog post on another topic) overtook the topic model focus. I think the biggest interest here turns out to actually be the start of a generalisable, efficient way of representing text from a play in a tidy data model. So let's have a go.

Tidying Hamlet

Basic structure of the raw data

When I first download one of the five or more versions of Hamlet from the Gutenberg project, it looks like this:

```
 [1] "HAMLET, PRINCE OF DENMARK"
 [2] ""
 [3] "by William Shakespeare"
 [4] ""
 [5] ""
 [6] ""
 [7] ""
 [8] "PERSONS REPRESENTED."
 [9] ""
[10] "Claudius, King of Denmark."
[11] "Hamlet, Son to the former, and Nephew to the present King."
[12] "Polonius, Lord Chamberlain."
[13] "Horatio, Friend to Hamlet."
[14] "Laertes, Son to Polonius."
[15] "Voltimand, Courtier."
[16] "Cornelius, Courtier."
[17] "Rosencrantz, Courtier."
[18] "Guildenstern, Courtier."
[19] "Osric, Courtier."
[20] "A Gentleman, Courtier."
[21] "A Priest."
[22] "Marcellus, Officer."
[23] "Bernardo, Officer."
[24] "Francisco, a Soldier"
[25] "Reynaldo, Servant to Polonius."
[26] "Players."
[27] "Two Clowns, Grave-diggers."
[28] "Fortinbras, Prince of Norway."
[29] "A Captain."
[30] "English Ambassadors."
[31] "Ghost of Hamlet's Father."
[32] ""
[33] "Gertrude, Queen of Denmark, and Mother of Hamlet."
[34] "Ophelia, Daughter to Polonius."
[35] ""
[36] "Lords, Ladies, Officers, Soldiers, Sailors, Messengers, and other"
[37] "Attendants."
[38] ""
```

```
 [39] "SCENE. Elsinore."
 [40] ""
 [41] ""
 [42] ""
 [43] "ACT I."
 [44] ""
 [45] "Scene I. Elsinore. A platform before the Castle."
 [46] ""
 [47] "[Francisco at his post. Enter to him Bernardo.]"
 [48] ""
 [49] "Ber."
 [50] "Who's there?"
 [51] ""
 [52] "Fran."
 [53] "Nay, answer me: stand, and unfold yourself."
 [54] ""
 [55] "Ber."
 [56] "Long live the king!"
 [57] ""
 [58] "Fran."
 [59] "Bernardo?"
 [60] ""
 [61] "Ber."
 [62] "He."
 [63] ""
 [64] "Fran."
 [65] "You come most carefully upon your hour."
 [66] ""
 [67] "Ber."
 [68] "'Tis now struck twelve. Get thee to bed, Francisco."
 [69] ""
 [70] "Fran."
 [71] "For this relief much thanks: 'tis bitter cold,"
 [72] "And I am sick at heart."
 [73] ""
 [74] "Ber."
 [75] "Have you had quiet guard?"
 [76] ""
 [77] "Fran."
 [78] "Not a mouse stirring."
 [79] ""
 [80] "Ber."
 [81] "Well, good night."
 [82] "If you do meet Horatio and Marcellus,"
 [83] "The rivals of my watch, bid them make haste."
 [84] ""
 [85] "Fran."
 [86] "I think I hear them.--Stand, ho! Who is there?"
 [87] ""
 [88] "[Enter Horatio and Marcellus.]"
 [89] ""
 [90] "Hor."
 [91] "Friends to this ground."
 [92] ""
 [93] "Mar."
 [94] "And liegemen to the Dane."
 [95] ""
 [96] "Fran."
 [97] "Give you good-night."
 [98] ""
 [99] "Mar."
[100] "O, farewell, honest soldier;"
```

So this is interesting. Here are some things we need to take into account in putting this into a data model that loses no or minimal information from its structure and content:

- We have about 40 lines of preamble, then we are on to a combination of dialogue and stage directions.
- The play is structured into Acts and Scenes. Scenes have numbers and descriptions (eg `Scene I. Elsinore. A platform before the Castle.`) but Acts just have numbers. Scenes are strictly hierarchically ordered under Acts (ie there is no Scene that continues over the boundary between two Acts).
- The stage directions are in square brackets `[Enter Horatio and Marcellus]`. A stage direction can be seen as applying until it is countermanded (for example, once Francisco and Bernardo have entered, they should be considered as present on stage until we are told otherwise)
- A continuous line of speech can go over several lines, such as Francisco in lines 71 and 72 of the above. The line breaks may be

significant (and sometimes certainly are when the line is in verse), or not.
- The character speaking is referred to be abbrevaition (eg `Fran.`), which needs decoding to be fully human-readable (an actor learning the play, or a reader familiar with it would know that `Hor.` is Horatio, but only by mental decoding – something that should be done in the data model)

Here is the R code that sets up my session, including preparation for things I haven't talked about yet, downloads the play and gets us to this point of looking at the first 100 lines:

Post continues below R code

```
library(tidyverse)
library(tidytext)
library(gutenbergr)
library(SnowballC)
library(tidygraph)
library(ggraph)
library(Cairo)
library(BTM)
library(textplot)
library(concaveman)

# TODO - a problem with lines 6380:6381 which has a stage direction across two lines.

#' Replace NA values in a vector with counting from the previous value
#'
#' @param x a vector character
#' @examples
#' replace_na_with_num(c(1,1,1,1, NA, NA, 1, 1, 1, NA))
replace_na_with_num <- function(x){
  # I can't think of a way to do this next operation without a loop because there is iteration
  for(i in 2:length(x)){
    if(is.na(x[i])){
      x[i] <- x[i-1] + 1
    }
  }
  return(x)
}
stopifnot(replace_na_with_num(c(1,1,1,NA, NA, 1, NA)) == c(1,1,1,2,3,1,2))

# There seem to be two Shakespeares, definitely duplicates:
filter(gutenberg_authors, grepl("Shakespeare", author))

# At least five versions of Hamlet
filter(gutenberg_metadata, grepl("Hamlet", title) &
         author == "Shakespeare, William" &
         grepl("Public domain", rights) &
         language == "en") %>%
  select(gutenberg_id, title, gutenberg_bookshelf)

# I'll download the first mentioned. May not be the best.
hamlet <- gutenberg_download(1524) %>%
  mutate(original_line_number = 1:n()) %>%
  select(-gutenberg_id)

hamlet %>%
  select(text) %>%
  slice(1:100) %>%
  pull(text)
```

Stage directions

With one annoying exception, the stage directions are all single lines so we can pick them up easily with a regular expression that looks for lines that begin and finish in square brackets.

```
> # Number of times each stage direction used
> hamlet %>%
+   filter(grepl("^\\[.*\\]$", text)) %>%
+   count(text, name = "times_used", sort = TRUE)
# A tibble: 129 x 2
   text                                  times_used

 1 [Exit.]                                       19
 2 [Exeunt.]                                     13
 3 [Sings.]                                      11
 4 [Enter Polonius.]                              5
```

```
 5 [Exit Polonius.]                              5
 6 [Enter Hamlet.]                               4
 7 [Exeunt Rosencrantz and Guildenstern.]        4
 8 [Dies.]                                       3
 9 [Enter Rosencrantz and Guildenstern.]         3
10 [Exit Ghost.]                                 3
```

Here is the exception – it comes from near the end, when a whole bunch of people (too many to list in a single line) come in with weapons to be used in the duel between Hamlet and Laertes:

```
> hamlet[6380:6381, ]
# A tibble: 2 x 2
  text                                                    original_line_number

1 [Enter King, Queen, Laertes, Lords, Osric, and Attendants with                6380
2 foils &c.]                                                                    6381
```

Scenes and Acts

Similarly, we can use regular expression to find with nearly 100% accuracy the beginnings of Scenes and Acts. Note that the capitalisation of `Act` is inconsistent:

```
> # Scenes and acts. Sometimes upper case sometimes not
> hamlet %>%
+   filter(grepl("^Scene\\s", text, ignore.case = TRUE) | grepl("^Act\\s", text, ignore.case =
TRUE))
# A tibble: 26 x 2
   text                                          original_line_number

 1 ACT I.                                                          43
 2 Scene I. Elsinore. A platform before the Castle.                45
 3 Scene II. Elsinore. A room of state in the Castle.             378
 4 Scene III. A room in Polonius's house.                         822
 5 Scene IV. The platform.                                       1029
 6 Scene V. A more remote part of the Castle.                    1211
 7 Act II.                                                       1560
 8 Scene I. A room in Polonius's house.                          1562
 9 Scene II. A room in the Castle.                               1779
10 ACT III.                                                      2744
# ... with 16 more rows
```

Character names

Generally, a line indicating a new speaker can be identified by the pattern of “capital letter – lower case letters – full stop – end of line”. So it is straightforward to pull out lines that meet this pattern with a regular expression and count them:

```
> hamlet %>%
+   filter(grepl("^[A-Z][a-z]+\\.$", text)) %>%
+   count(text, name= "number_speeches", sort = TRUE) %>%
+   slice(1:20)
# A tibble: 20 x 2
   text   number_speeches

 1 Ham.               358
 2 Hor.               108
 3 King.              102
 4 Pol.                86
 5 Queen.              69
 6 Laer.               62
 7 Oph.                57
 8 Ros.                45
 9 Mar.                31
10 Guil.               29
11 Osr.                25
12 Ber.                19
13 Ghost.              14
14 Rey.                13
15 Fran.                8
16 Capt.                7
17 Fort.                4
18 All.                 3
19 Danes.               3
20 Gent.                3
```

358 distinct speeches to learn if you want to play Hamlet, versus (for example) 29 for Guildenstern. Basically that simple search-and-count works, although there are annoyingly some context-specific uses of `Both.` and `All.`; and there are a few spoken lines (`Farewell.` and

`Good.`) that get mixed up into this pattern. It's very difficult to identify an algorithm that can tell these apart, particularly because (for example) `Good.` is a plausible abbreviation of a character name. I think only a human could be sure of the difference in a specific context, so we will need a bit of careful manual coding of some parts.

To decode abbreviations into character names and add some useful metadata for later, I created by hand a vector of the eight main characters (using my human interpretation to decide on what constituted "main"), and table that relates the abbreviations (`King.`, etc) to full names of the character ("Claudius, King of Denmark.") and human-friendly abbreviations for plots and the like using the term more commonly used today ("Claudius").

I also made a vector of words that looked like character names and feature as single word lines in the play but aren't referring to characters: `No.`, `Dead.`, `He.`, etc.

```
#==================Processing=======================
main_chars <- c("Hamlet", "Horatio", "Claudius", "Gertrude", "Ophelia", "Polonius", "Laertes",
"Ghost")
# These aren't the top 8 characters - First gravedigger and Rosencrantz both have more words
than Ghost -
# but seem the 8 most fully-rounded real people.

personae <- tribble(~speaker, ~speaker_abb, ~speaker_sh,
                    "Claudius, King of Denmark.", "King.", "Claudius",
                    "Hamlet, Son to the former, and Nephew to the present King.", "Ham.",
"Hamlet",
                    "Polonius, Lord Chamberlain.", "Pol.", "Polonius",
                    "Horatio, Friend to Hamlet.", "Hor.",             "Horatio",
                    "Laertes, Son to Polonius.", "Laer.", "Laertes",
                    "Voltimand, Courtier.", "Volt.", "Voltimand",
                    "Cornelius, Courtier.", "???",               "Cornelius",
                    "Rosencrantz, Courtier.", "Ros.",                        "Rosencrantz",
                    "Guildenstern, Courtier.", "Guil.",
"Guildenstern",
                    "Osric, Courtier.", "Osr.",                                "Osric",
                    "A Gentleman, Courtier.", "Gent.", "Gentleman",
                    "Marcellus, Officer.", "Mar.", "Marcellus",
                    "Bernardo, Officer.", "Ber.",              "Bernardo",
                    "Francisco, a Soldier", "Fran.",                      "Francisco",
                    "Reynaldo, Servant to Polonius.", "Rey.","Reynaldo",
                    "Players", "Players.",                             "Players",
                    "Fortinbras, Prince of Norway.", "For.", "Fortinbras",
                    "Fortinbras, Prince of Norway.", "Fort.", "Fortinbras",
                    "A Captain.", "Capt.",                                 "Captain",
                    "English Ambassador 1.", "1 Ambassador.", "English Ambassador",
                    "Ghost of Hamlet's Father.", "Ghost.", "Ghost",
                    "Gertrude, Queen of Denmark, and Mother of Hamlet.", "Queen.", "Gertrude",
                    "Ophelia, Daughter to Polonius.", "Oph.", "Ophelia",
                    "Prologue to The Murder of Gonzago, a play within a play", "Pro.", "Player
prologue",
                     "King in The Murder of Gonzago, a play within a play", "P. King.",
"Player King",
                   "Queen in The Murder of Gonzago, a play within a play", "P. Queen.",
"Player Queen",
                     "Lucianus, nephew to the King in play within a play", "Luc.", "Lucianus",
                   "Danes", "Danes.", "Danes",
                   "Servant", "Servant.", "Servant",
                   "Sailor", "Sailor.", "Sailor",
                   "Messenger", "Mess.", "Messenger",
                   "First gravedigger clown", "1 Clown.", "First gravedigger",
                   "Second gravedigger clown", "2 Clown.", "Second gravedigger",
                   "First priest", "1 Priest.", "First Priest",
                   "Second priest", "2 Priest.", "Second Priest",
                   "Danish courtier lord", "Lord.", "Lord",
                   # There are 3 uses of 'All.', in each case mean different groups
                   "All present on stage", "All.", "All",
                   # only two uses of 'Both.', near the beginning:
                   "Marcellus and Bernardo together", "Both.", "Marcellus and Bernardo") %>%
  mutate(main_character = speaker_sh %in% main_chars)

not_people <- c("He.", "Say.", "Farewell.", "Perpend.", "Nothing.", "Good.", "Swear.", "No.",
"Dead.", "One.")


# This chunk of code was how I identified the abbreviations that weren't in the original list
of personae
# (eg "1 Clown."), and also refined the list above of not_people.
hamlet %>%
```

```
  slice(-(1:38)) %>%
  mutate(speaker_abb = case_when(
    # Most people picked up by this
    grepl("^[A-Z][a-z]+\\.$", text)         ~ text,
    # "1 Ambassador", "2 Clown" and similar are picked up by this:
    grepl("^[1-9]\\s[A-Z][a-z]+\\.$", text) ~ text,
    # P. King. etc picked up by this:
    grepl("^[A-Z]\\.\\s[A-Z][a-z]+\\.$", text) ~ text
    )) %>%
  filter(text != "") %>%
  mutate(speaker_abb = if_else(speaker_abb %in% not_people, NA_character_, speaker_abb)) %>%
  fill(speaker_abb) %>%
  filter(!is.na(speaker_abb)) %>%
  anti_join(personae, by = "speaker_abb")
```

A table of data with “spoken line” as grain

I use these lookup tables and vectors to create my first tidier version of the data, `hamlet_lines`. This object is shown below. In this version, each row of the table is a complete line of dialogue. Rows of the original data that are not dialogue but refer to the Act, Scene, stage direction or who is speaking have been pivoted wider to appear as columns such as `speaker`, `last_stage_direction`, `act` and `scene`.

Here is how that object looks:

```
> hamlet_lines
# A tibble: 3,945 x 11
   text        original_line_nu~ speaker_abb speaker   speaker_sh main_character
last_stage_direction act    scene      new_speaker_thi~ line_number_thi~

 1 Who's ther~                50 Ber.        Bernardo~ Bernardo   FALSE          Francisco at
his po~ ACT I. Scene I. ~ TRUE                           1
 2 Nay, answe~                53 Fran.       Francisc~ Francisco  FALSE          Francisco at
his po~ ACT I. Scene I. ~ TRUE                           1
 3 Long live ~                56 Ber.        Bernardo~ Bernardo   FALSE          Francisco at
his po~ ACT I. Scene I. ~ TRUE                           1
 4 Bernardo?                  59 Fran.       Francisc~ Francisco  FALSE          Francisco at
his po~ ACT I. Scene I. ~ TRUE                           1
 5 He.                        62 Ber.        Bernardo~ Bernardo   FALSE          Francisco at
his po~ ACT I. Scene I. ~ TRUE                           1
 6 You come m~                65 Fran.       Francisc~ Francisco  FALSE          Francisco at
his po~ ACT I. Scene I. ~ TRUE                           1
 7 'Tis now s~                68 Ber.        Bernardo~ Bernardo   FALSE          Francisco at
his po~ ACT I. Scene I. ~ TRUE                           1
 8 For this r~                71 Fran.       Francisc~ Francisco  FALSE          Francisco at
his po~ ACT I. Scene I. ~ TRUE                           1
 9 And I am s~                72 Fran.       Francisc~ Francisco  FALSE          Francisco at
his po~ ACT I. Scene I. ~ FALSE                          2
10 Have you h~                75 Ber.        Bernardo~ Bernardo   FALSE          Francisco at
his po~ ACT I. Scene I. ~ TRUE                           1
# ... with 3,935 more rows
```

Note that we have kept the successive lines of text by a single character as per the original – for instance rows 71 and 72 of the original data, spoken by Fancisco, are now rows 8 and 9 but are still two separate rows. A logical flag `new_speaker_this_line` helps us identify such cases for future.

Here’s the chunk of code that creates this table, with “line” as its granularity, from the original messy data. Most of this code is dealing with various quirks in the data, such as `2 Clown` (for “second clown”, or second gravedigger) having a different structure of character abbreviation from most of the roles.

```
hamlet_lines <- hamlet %>%
  slice(-(1:39)) %>%
  # Identify the speaker:
  mutate(speaker_abb = case_when(
    # Most people picked up by this
    grepl("^[A-Z][a-z]+\\.$", text)         ~ text,
    # "1 Ambassador", "2 Clown" and similar are picked up by this:
    grepl("^[1-9]\\s[A-Z][a-z]+\\.$", text) ~ text,
    # P. King. etc picked up by this:
    grepl("^[A-Z]\\.\\s[A-Z][a-z]+\\.$", text) ~ text
  )) %>%
  filter(text != "") %>%
  mutate(speaker_abb = if_else(speaker_abb %in% not_people, NA_character_, speaker_abb)) %>%
  fill(speaker_abb) %>%
  filter(is.na(speaker_abb) | text != speaker_abb) %>%
  left_join(personae, by = "speaker_abb") %>%
  # Identify stage directions:
```

```
        mutate(last_stage_direction = case_when(
               grepl("^\\[.*\\]$", text) ~ text
        )) %>%
        fill(last_stage_direction) %>%
        filter(is.na(last_stage_direction) | text != last_stage_direction) %>%
        mutate(last_stage_direction = gsub("[", "", last_stage_direction, fixed = TRUE),
               last_stage_direction = gsub("]", "", last_stage_direction, fixed = TRUE)) %>%
        # Identify Act:
        mutate(act = case_when(
          grepl("^A[Cc][Tt]\\s", text) ~ text
        )) %>%
        fill(act) %>%
        filter(is.na(act) | text != act) %>%
        # Identify Scene
        mutate(scene = case_when(
          grepl("^S[Cc][Ee][Nn][Ee]\\s", text) ~ text
        )) %>%
        fill(scene) %>%
        filter(is.na(scene) | text != scene) %>%
        # regularise some spelling:
        mutate(scene = gsub("Castle", "castle", scene)) %>%
        # identify the actual speeches and count the lines in each speech by a continuing speaker
        mutate(new_speaker_this_line = is.na(lag(speaker)) | speaker != lag(speaker),
               line_number_this_speech = ifelse(new_speaker_this_line, 1, NA),
               line_number_this_speech = replace_na_with_num(line_number_this_speech))
```

This tidy table of data, at the granularity of line of dialogue, makes it easy to count lines per speaker, Act, Scene, etc. For example, we can see that Act II, Scene II has the most spoken lines (590):

```
> count(hamlet_lines, act, scene)
# A tibble: 20 x 3
   act      scene                                              n

 1 ACT I.   Scene I. Elsinore. A platform before the castle.     190
 2 ACT I.   Scene II. Elsinore. A room of state in the castle.   280
 3 ACT I.   Scene III. A room in Polonius's house.               141
 4 ACT I.   Scene IV. The platform.                              101
 5 ACT I.   Scene V. A more remote part of the castle.          212
 6 Act II.  Scene I. A room in Polonius's house.                130
 7 Act II.  Scene II. A room in the castle.                     590
 8 ACT III. Scene I. A room in the castle.                      196
 9 ACT III. Scene II. A hall in the castle.                     383
10 ACT III. Scene III. A room in the castle.                    102
11 ACT III. Scene IV. Another room in the castle.               236
12 ACT IV.  Scene I. A room in the castle.                       46
13 ACT IV.  Scene II. Another room in the castle.                28
14 ACT IV.  Scene III. Another room in the castle.               71
15 ACT IV.  Scene IV. A plain in Denmark.                        68
16 ACT IV.  Scene V. Elsinore. A room in the castle.            228
17 ACT IV.  Scene VI. Another room in the castle.                29
18 ACT IV.  Scene VII. Another room in the castle.              212
19 ACT V.   Scene I. A churchyard.                              283
20 ACT V.   Scene II. A hall in the castle.                     419
```

A table of data with “spoken word” as grain

My next step is to make a table or data frame with *spoken word* as the grain. Here I want not just bags of words, but the original sequence preserved, and whether a word was originally at the beginning of a line, as well as who is speaking (and all the different abbreviations of their name and groups that person belongs to), the last stage direction, the Act and Scene, etc. For future use I also want a stemmed version of each word (eg so “answered”, “answering” and “answers” all reduce to their stem “answer”), and a flag of whether each word is a stop word or not.

Here’s how that table is going to look, capturing all 30,000 or so words spoken in the uncut version of Hamlet”

```
> hamlet_words %>% select(contains("word"), everything())
# A tibble: 30,022 x 20
   word  stopword word_stem new_speaker_thi~ word_number_thi~ word_number word_number_thi~
word_number_thi~ original_line_n~ speaker_abb

 1 who's TRUE     who'      TRUE                           1           1                1
1               50 Ber.
 2 there TRUE     there     FALSE                          2           2                2
2               50 Ber.
 3 nay   FALSE    nai       TRUE                           1           3                3
3               53 Fran.
 4 answ~ FALSE    answer    FALSE                          2           4                4
```

```
4               53 Fran.
 5 me    TRUE     me        FALSE                          3           5                5
5               53 Fran.
 6 stand FALSE    stand     FALSE                          4           6                6
6               53 Fran.
 7 and   TRUE     and       FALSE                          5           7                7
7               53 Fran.
 8 unfo~ FALSE    unfold    FALSE                          6           8                8
8               53 Fran.
 9 your~ TRUE     yourself  FALSE                          7           9                9
9               53 Fran.
10 long  FALSE    long      TRUE                           1          10               10
10               56 Ber.
# ... with 30,012 more rows, and 10 more variables: speaker , speaker_sh , main_character ,
last_stage_direction ,
#   act , scene , new_speaker_this_line , line_number_this_speech , speech_number , top_20_char
```

Note that the `hamlet_words` data frame has about 30,000 words – one for each word in Hamlet, excluding stage directions.

And here is how I made that table out of the `hamlet_lines`

```
# Make up a set of stopwords.
# The concept of stopwords seems a bit arbitrary. Why aren't "shall", "go", "us" and "one"
# already stopwords in the snowball lexicon whereas "i", "our" are?
our_stopwords <- tibble(word = c(
  "thou", "thine", "o", "tis", "thee", "thy", "sir", "hath", "lord", "us", "one"
)) %>%
  mutate(lexicon = "made up") %>%
  rbind(get_stopwords())

# First iteration of data frame with word as grain
hamlet_words <- hamlet_lines %>%
  tidytext::unnest_tokens(output ="word", input = "text") %>%
  left_join(our_stopwords, by = "word") %>%
  mutate(stopword = !is.na(lexicon)) %>%
  select(-lexicon) %>%
  mutate(word_stem = wordStem(word)) %>%
  # identify the actual speeches and count the words in each speech by a continuing speaker
  mutate(new_speaker_this_word = is.na(lag(speaker)) | speaker != lag(speaker),
         word_number_this_speech = ifelse(new_speaker_this_word, 1, NA),
         word_number_this_speech = replace_na_with_num(word_number_this_speech)) %>%
  mutate(word_number = 1:n()) %>%
  group_by(act) %>%
  mutate(word_number_this_act = 1:n()) %>%
  group_by(act, scene) %>%
  mutate(word_number_this_scene = 1:n()) %>%
  ungroup()

# Identify if starting a speech or not...
words_starting_speeches <- hamlet_words %>%
  filter(new_speaker_this_word) %>%
  select(word_number) %>%
  mutate(speech_number = 1:n())

# ... and rejoin to the original:
hamlet_words <- hamlet_words %>%
  left_join(words_starting_speeches, by = "word_number") %>%
  fill(speech_number)

# words per speaker:
char_summary <- hamlet_words %>%
  group_by(speaker_sh) %>%
  summarise(words = n(),
            speeches = sum(new_speaker_this_word),
            words_per_speech = words / speeches,
            stopwords = sum(stopword),
            prop_non_stop = 1 - stopwords / words,
            most_words_single_speech = max(word_number_this_speech)) %>%
  arrange(desc(words))

# are you one of the top 20 speakers?...
top_20_chars <- char_summary[1:20, ]$speaker_sh


# ... and rejoin to the original:
```

```
hamlet_words <- hamlet_words %>%
  mutate(top_20_char = speaker_sh %in% top_20_chars,
         speaker_sh = factor(speaker_sh, levels = char_summary$speaker_sh))
```

Analysis

How many words each?

OK, it's time to do something fun now that we have tidied the data. Let's start with just counting words. Out of the 30,000 words in Hamlet, who says how many?

You might have noticed in the code that `char_summary` data frame I make along the way there. It has some summary information on each character such as the number of words and speeches, words per speech, stop words, proportion of words that aren't stopwords, etc. Here's what that looks like. We see that (of course), Hamlet himself has the most words to say; in fact, just over a third of all the words in Hamlet are spoken by the titular character:

speaker_sh	words	speeches	words_per_speech	stopwords	prop_non_stop	most_words_single_speech
Hamlet	11907	346	34.413295	6412	0.4614932	473
Claudius	4116	101	40.752475	2189	0.4681730	392
Polonius	3017	84	35.916667	1661	0.4494531	241
Horatio	2034	106	19.188679	1128	0.4454277	225
Laertes	1463	62	23.596774	764	0.4777854	274
Ophelia	1163	57	20.403509	661	0.4316423	125
Gertrude	1084	68	15.941176	561	0.4824723	137
First gravedigger	741	33	22.454545	421	0.4318489	76
Rosencrantz	695	45	15.444444	411	0.4086331	103
Ghost	682	14	48.714286	332	0.5131965	375
Marcellus	429	31	13.838710	228	0.4685315	80
Osric	358	25	14.320000	200	0.4413408	64
Player King	337	4	84.250000	154	0.5430267	233
Guildenstern	323	29	11.137931	206	0.3622291	43
Player Queen	238	5	47.600000	112	0.5294118	102
Fortinbras	222	6	37.000000	113	0.4909910	98
Bernardo	205	19	10.789474	109	0.4682927	35
Gentleman	192	3	64.000000	86	0.5520833	104
Voltimand	149	1	149.000000	73	0.5100671	149
Second gravedigger	98	12	8.166667	55	0.4387755	22
First Priest	91	2	45.500000	49	0.4615385	65
Captain	84	7	12.000000	48	0.4285714	53
Lord	67	3	22.333333	39	0.4179104	42
Reynaldo	65	13	5.000000	49	0.2461538	11
Francisco	55	8	6.875000	25	0.5454545	14
English Ambassador	40	1	40.000000	23	0.4250000	40
Lucianus	40	1	40.000000	8	0.8000000	40
Messenger	36	2	18.000000	25	0.3055556	23
Sailor	35	1	35.000000	23	0.3428571	35
Player prologue	16	1	16.000000	10	0.3750000	16
Danes	13	3	4.333333	9	0.3076923	5
All	9	3	3.000000	3	0.6666667	5
Marcellus and Bernardo	9	2	4.500000	6	0.3333333	6
Servant	9	1	9.000000	6	0.3333333	9

Hang on, didn't we see earlier that Hamlet had 358 speeches, and now we find only 346? That's an anomaly related to how the data tidying worked. The 358 came from counting likes that appeared as `Ham.`; the 346 is more complex and looks for when Hamlet starts speaking after someone else had. My first guess at the cause of the anomaly is scene breaks where Hamlet was the last person speaking in the previous scene, and also first in the new one. But this would need checking. I don't have the inclination to fix that now, so let's just note this as an example of the sort of detail that needs fixing.

Distinctive words

It's interesting to look at the most distinctive words spoken by each character. The plot below shows the proportion of each character's words that are a particular wordstem, divided by that proportion for the whole play. Effectively, this is the term frequency – inverse document frequency (TFIDF).Some of this is interesting (Polonius talks about his 'daughter' Ophelia; the Ghost talks about 'blood', 'foul', and being 'beneath'), some not so much (Hamlet's distinctive words in particular seem to have little message for me).

Here's the code to calculate those TFIDF values (I am using `dplyr` to do this explicitly so I can be sure of what is going on, rather than using specialist functions) and draw that chart of distinctive words:

```
#--------------------Distinctive words-------------------
```

```
# most distinctive word stem for each of the main cast
dist_words <- hamlet_words %>%
  filter(!stopword) %>%
  count(word_stem, speaker_sh) %>%
  group_by(speaker_sh) %>%
  mutate(prop_this_speaker = n / sum(n)) %>%
  ungroup() %>%
  mutate(prop_overall = n / sum(n),
         relative = prop_this_speaker / prop_overall) %>%
  filter(n > 2) %>%
  group_by(speaker_sh) %>%
  arrange(desc(relative), desc(n)) %>%
  slice(1:5) %>%
  filter(speaker_sh %in% top_20_chars)
# note that "mai" is the word stem for "may", and "joi" for "joy"

# plot of disinctive word stems
dist_words %>%
  group_by(speaker_sh) %>%
  mutate(rank = 1:n()) %>%
  ggplot(aes(x = rank, y = reorder(speaker_sh, desc(speaker_sh)), label = word_stem)) +
  geom_text(size = 3) +
  theme(panel.grid.major.x = element_blank(),
        panel.grid.minor.x = element_blank(),
        axis.text.x = element_blank()) +
  labs(x = "", y = "",
       title = "Most distinctive word stems for the top 20 characters in Hamlet")
```

How long are characters' speeches

Beyond simple counts and averages of words, lines and speeches, it might be interesting to see how the *distribution* of how long the various speeches different characters deliver. Here's how that looks for nine of the main characters:

The main thing I spot here is that the Ghost of Hamlet's father gets only a few speeches but they tend to be long and uninterrupted compared to the other characters. Gertrude, Ophelia and Horatio tend to have short speeches less than 100 words, acting as foils for their interlocutors (normally Hamlet himself). Hamlet of course has the full range – many short bits of snappy dialogue, but also three or four lengthy 400+ word speeches.

The longest speech is Hamlet's in Act III Scene I. In this speech he reflects on the impromptu play just put on by the Players; contrasts their acted emotion and passion with his own indecision; considers he must still be unsure whether the Ghost is misleading him about the villainy of Caludius; and explains his plan to use a play (within a play) to get to test Caludius' conscience (paragraph breaks added by hand):

"Ay, so, God b' wi' ye!

Now I am alone. O, what a rogue and peasant slave am I! Is it not monstrous that this player here, But in a fiction, in a dream of passion, Could force his soul so to his own conceit That from her working all his visage wan'd; Tears in his eyes, distraction in's aspect, A broken voice, and his whole function suiting With forms to his conceit? And all for nothing! For Hecuba? What's Hecuba to him, or he to Hecuba, That he should weep for her? What would he do, Had he the motive and the cue for passion That I have? He would drown the stage with tears And cleave the general ear with horrid speech; Make mad the guilty, and appal the free; Confound the ignorant, and amaze, indeed, The very faculties of eyes and ears.

"Yet I, A dull and muddy-mettled rascal, peak, Like John-a-dreams, unpregnant of my cause, And can say nothing; no, not for a king Upon whose property and most dear life A damn'd defeat was made. Am I a coward? Who calls me villain? breaks my pate across? Plucks off my beard and blows it in my face? Tweaks me by the nose? gives me the lie i' the throat As deep as to the lungs? who does me this, ha? 'Swounds, I should take it: for it cannot be But I am pigeon-liver'd, and lack gall To make oppression bitter; or ere this I should have fatted all the region kites With this slave's offal: bloody, bawdy villain! Remorseless, treacherous, lecherous, kindless villain! O, vengeance!

"Why, what an ass am I! This is most brave, That I, the son of a dear father murder'd, Prompted to my revenge by heaven and hell, Must, like a whore, unpack my heart with words And fall a-cursing like a very drab, A scullion! Fie upon't! foh!–About, my brain!

"I have heard That guilty creatures, sitting at a play, Have by the very cunning of the scene Been struck so to the soul that presently They have proclaim'd their malefactions; For murder, though it have no tongue, will speak With most miraculous organ, I'll have these players Play something like the murder of my father Before mine uncle: I'll observe his looks; I'll tent him to the quick: if he but blench, I know my course.

"The spirit that I have seen May be the devil: and the devil hath power To assume a pleasing shape; yea, and perhaps Out of my weakness and my melancholy,– As he is very potent with such spirits,– Abuses me to damn me: I'll have grounds More relative than this.–the play's the thing Wherein I'll catch the conscience of the king. [Enter King, Queen, Polonius, Ophelia, Rosencrantz, and Guildenstern.]"

Here's the R code to plot the distribution of speech lengths, and extract in full an original speech that has been identified from our tidied row-grain data frame:

```
#--------------------Distribution of length of speeches--------------------

# chart of distribution of length of speeches
hamlet_words %>%
  filter(main_character | speaker_sh == "First gravedigger") %>%
  group_by(speech_number) %>%
```

```
  filter(word_number_this_speech == max(word_number_this_speech)) %>%
  select(speaker_sh, speech_number, word_number_this_speech) %>%
  ggplot(aes(x = word_number_this_speech)) +
  facet_wrap(~speaker_sh,scales = "free_y") +
  geom_density(fill = "steelblue", col = NA, alpha = 0.5) +
  geom_rug() +
  scale_x_sqrt() +
  labs(x = "Number of continuous words in row",
       title = "Length of uninterrupted speeches in Shakespeare's Hamlet") +
  theme(panel.spacing = unit(2, "lines"))

# Find the longest speech:
longest_speech <- hamlet_words %>%
  arrange(desc(word_number_this_speech)) %>%
  slice(1) %>%
  pull(speech_number)

# Full text of that speech:
hamlet_words %>%
  filter(speech_number == longest_speech) %>%
  distinct(original_line_number) %>%
  inner_join(hamlet, by = "original_line_number") %>%
  select(text) %>%
  pull(text) %>%
  cat()
```

Incidentally, this exercise drew my attention to another problem – the final stage direction here `[Enter King, Queen, Polonius, Ophelia, Rosencrantz, and Guildenstern.]` had not been correctly identified as such in my tidying program. So another detail to fix (I count about four of these so far…).

Who interacts with whom?

Something that's of interest that we can find out with tidied, structured data that is difficult otherwise is the interactions between different characters. A play features (mostly) a single character talking at a time, and can be analysed as a series of hand-offs from one character to another. We can count each transition and turn it into a network chart. For illustrative purposes here, I am ignoring the direction of these transitions, and just counting the absolute number. This gives us a nice illustration of who interacts (either interrupting, or simply taking turns) with whom:

What I like about this graph is how it shows the relationship between Hamlet and his wingman Horatio as core to the play; while also making clear who are the other major speaking characters (Claudius, Polonius, Laertes, Gertrude, etc) and the relationhsips between them.

Here's the code for that network graph, made simple by the wonder of Thomas Pedersen's `tidygraph` and `ggraph` R packages.

```
#=========who interacts with whom==============

hamlet_handovers <- hamlet_words %>%
  filter(new_speaker_this_word) %>%
  mutate(previous_speaker = lag(speaker_sh)) %>%
  select(new_speaker = speaker_sh, previous_speaker, act, scene, word_number)  %>%
  mutate(words_this_speech = lead(word_number, default = max(hamlet_words$word_number)) -
word_number)

d <- hamlet_handovers %>%
  filter(!is.na(previous_speaker)) %>%
  count(new_speaker, previous_speaker, sort = TRUE)  %>%
  select(from = previous_speaker, to = new_speaker, n) %>%
  mutate(from = as.character(from),
         to = as.character(to))

# we are going to count from-to and to-from as though they are the same,
# ie ignoring direction
d1 <- d %>%
  rename(new_from = to,
         to = from) %>%
  rename(from = new_from) %>%
  rbind(d) %>%
  group_by(from, to) %>%
  summarise(n = sum(n)) %>%
  filter(from < to)
# to do = tidy up Marcellus and Bernardo here

d2 <-as_tbl_graph(d1) %>%
  activate(nodes) %>%
  left_join(char_summary, by = c("name" = "speaker_sh"))
```

```
layout <- create_layout(d2, layout = 'igraph', algorithm = 'kk')

ggraph(layout) +
  geom_edge_arc(strength = 0,
                aes(edge_colour = n, edge_width = n)) +
  geom_node_label(aes(label = name), colour = "white", fill = "grey10", label.size = 0) +
  geom_node_point(aes(size = words), shape = 19, colour = "steelblue", alpha = 0.5) +
  scale_edge_colour_viridis() +
  theme(panel.background = element_rect(fill = "black")) +
  scale_size_area(max_size = 22, label = comma) +
  labs(edge_colour = "Number of transitions:",
       size = "Total words spoken:")
```

Topics of the play

Finally, some bi-term topic modelling. In the various short speeches (from 1 word to 500), which word stems tend to come together? Here's a first look at this.

Does this help? Well, not so much. I probably need some more thinking, certainly about how many topics to use for starters.

I originally intended this topic modelling to be the main point of this blog, but in the six months that this post has sat in "it's good, but it needs fixing" purgatory I have since used bi-term topic modelling elsewhere, and no longer feel great motivation to finish this idea off immediately. I *do* think there is big potential in using topic modelling to understand topics within a Shakespeare play (ie treating each speech as a document), not just between the plays (treating each play as a document). Most examples using Shakespeare for topic modelling are of the latter variety, probably because (as today's post shows) there's a fair bit of wrangling required before one can start analysing speeches as documents.

Anyway, here is the absurdly short code to fit the bi-term topic model.

```
#=====================Topic modelling================
# We could do this for each individual speech; or for all speeches.
# Here we treat each speech as a document.

hw_for_btm <- hamlet_words %>%
  filter(!stopword) %>%
  select(speech_number, word_stem)

btm_model <- BTM(hw_for_btm, k = 20)

print(plot(btm_model, top_n = 7))
```

Conclusion

Well, that's it. I've had a good go at tidying one Shakespeare play into a data model that loses minimal original information, and puts us in good shape to analyse words, topics, interactions between characters, how Acts and Scenes and stage directions work, etc. There are a handful of small details I haven't fixed, but the overall project works. It wouldn't be hard to extend this to work with other Shakespeare plays, and maybe for plays in general.

The process of doing this has produced one piece of analysis I like, which is the network graph of hand-offs (one character speaking after the other). I think that has potential, to show at a glance what the characters are doing in a play. I'd like to try that elsewhere.

In the meantime, that's all folks. Take care out there, wear a mask, and don't go out if you don't have to. Unless you're in New Zealand or Western Australia, in which case, sure, go out, see if the rest of us mind, just don't rub it in.