Text Mining in R

1 Introduction to Text Cleaning

Here, we shall use some texts from `Gutenbergr` package to demonstrate the idea of tidy text.

```
library(stringr)
library(tidytext)
library(dplyr)
library(gutenbergr)
library(ggplot2)
library(plotly)
```

Here, the `tidytext` package is mainly used for all the text mining works, the package `dplyr` provides the piping operator `%>%` which can readily be used to determine the course of action to be taken with the data, and package `gutenbergr` is used to download relevant text files from Project Gutenberg (https://www.gutenberg.org/). We shall be using `ggplot2` and `plotly` for making interactive plots.

2 Exploring Project Gutenberg

The `gutenbergr` package provides the data `gutenberg_metadata` that contains all the information about all books available in Project Gutenberg, as well as their author, title and relevant details.

```
gutenberg_metadata[1:1000, ]
```

gutenberg_id <int>
0
1
2
3
4
5
6
7
8
9

1-10 of 1,000 rows | 1-1 of 8 columns Previous 1 2 3 4 5 6 ... 100 Next

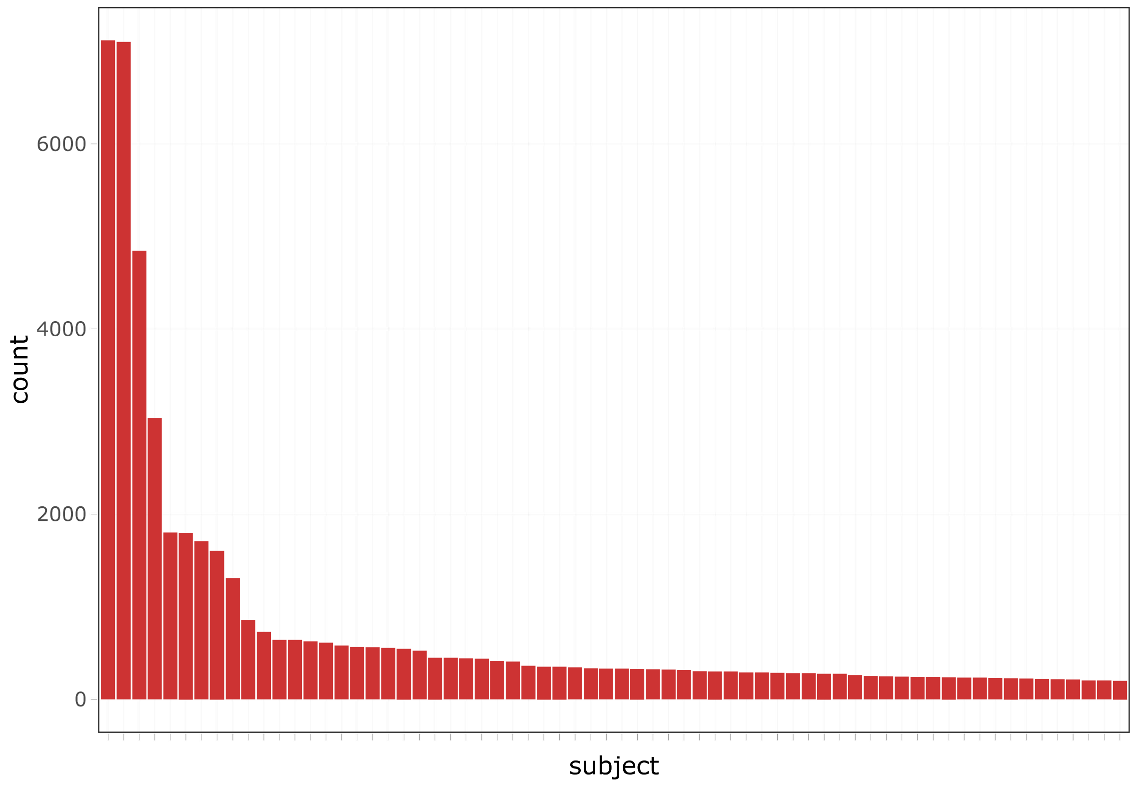
However, since this contains many books which are written in languages other than English, we first filter out only those which are written in English using `gutenberg_works()` function.

Now, let us see which genre's these works fall into.

```
genre_dat <- gutenberg_subjects %>%
    group_by(subject) %>%
    summarise(count = n()) %>%
    dplyr::filter(count > 200) %>%
    mutate(subject = reorder(subject, desc(count)))

p <- ggplot(genre_dat, aes(x = subject, y = count)) +
    geom_col(fill = "brown3") +
    theme_bw() +
    theme(axis.text.x = element_blank())

ggplotly(p)
```



Now, we shall consider downloading some of the Sherlock Holmes stories, which can be filtered by using regular expressions.

```
df <- gutenberg_subjects %>% filter(grepl("Holmes, Sherlock", subject))
df
```

gutenberg_id <int>	subject_type <chr>	subject <chr>
108	lcsh	Holmes, Sherlock (Fictitious character) -- Fiction
221	lcsh	Holmes, Sherlock (Fictitious character) -- Fiction
244	lcsh	Holmes, Sherlock (Fictitious character) -- Fiction
834	lcsh	Holmes, Sherlock (Fictitious character) -- Fiction
1661	lcsh	Holmes, Sherlock (Fictitious character) -- Fiction
2097	lcsh	Holmes, Sherlock (Fictitious character) -- Fiction
2343	lcsh	Holmes, Sherlock (Fictitious character) -- Fiction
2344	lcsh	Holmes, Sherlock (Fictitious character) -- Fiction
2345	lcsh	Holmes, Sherlock (Fictitious character) -- Fiction

gutenberg_id	subject_type	subject
<int>	<chr>	<chr>
2346	lcsh	Holmes, Sherlock (Fictitious character) -- Fiction

1-10 of 47 rows Previous 1 2 3 4 5 Next

Let us download first five texts of the sherlock holmes stories corresponding to the above ids.

```
sherlockdata <- gutenberg_download(df$gutenberg_id[1:5], meta_fields = "title")
head(sherlockdata, 100)   # just a few rows to give you a feel about it
```

gutenberg_id	text
<int>	<chr>
108	THE RETURN OF SHERLOCK HOLMES,
108	
108	A Collection of Holmes Adventures
108	
108	
108	by Sir Arthur Conan Doyle
108	
108	
108	
108	

1-10 of 100 rows | 1-2 of 3 columns Previous 1 2 3 4 5 6 ... 10 Next

3 Counting Words

3.1 Splitting into Words

Once we have our data containing the texts of the stories, the next thing is to split the story into paragraphs, paragraphs into sentences, sentences into words. We shall treat words as the most basic foundation of text processing.

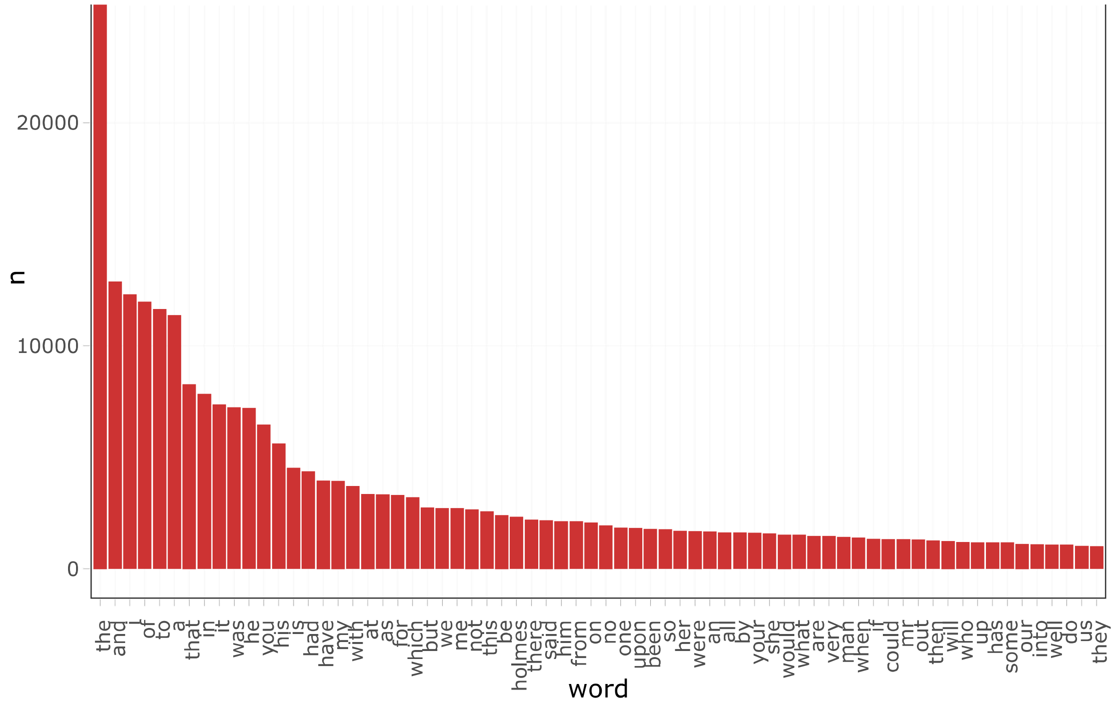
Now, the `gutenberg_download` function already splits our story into sentences, hence, we just need to split them up into words. For this, `unnest_token` function is used.

```
tidy_books <- sherlockdata %>% unnest_tokens(word, text) %>%
  mutate(word = str_remove_all(word, "[^a-zA-Z]"))
```

Now that we have collected the words, let us visualize the frequency of the words.

```
p <- tidy_books %>% count(word, sort = TRUE) %>% dplyr::filter(n > 1000) %>% mutate(word = reorder(wo
  rd, desc(n))) %>%
    ggplot(aes(x = word, y = n)) +
    geom_col(fill = "brown3") +
    theme_bw() +
    theme(axis.text.x = element_text(angle = 90))

ggplotly(p)
```



We see that the most appearing words are *the, and, of, a, an* etc. which should appear in every text very often, and clearly, they does not give a meaningful feature for the stories of Sherlock Holmes. Therefore, we need to remove these.

3.2 Removing Stopwords

`tidytext` package comes with a built-in dataset for stopwords in English.

```
data("stop_words")  # some data containing the stop words
head(stop_words, 100)
```

word <chr>	lexicon <chr>
a	SMART
a's	SMART
able	SMART
about	SMART
above	SMART
according	SMART
accordingly	SMART
across	SMART
actually	SMART
after	SMART

1-10 of 100 rows Previous 1 2 3 4 5 6 ... 10 Next

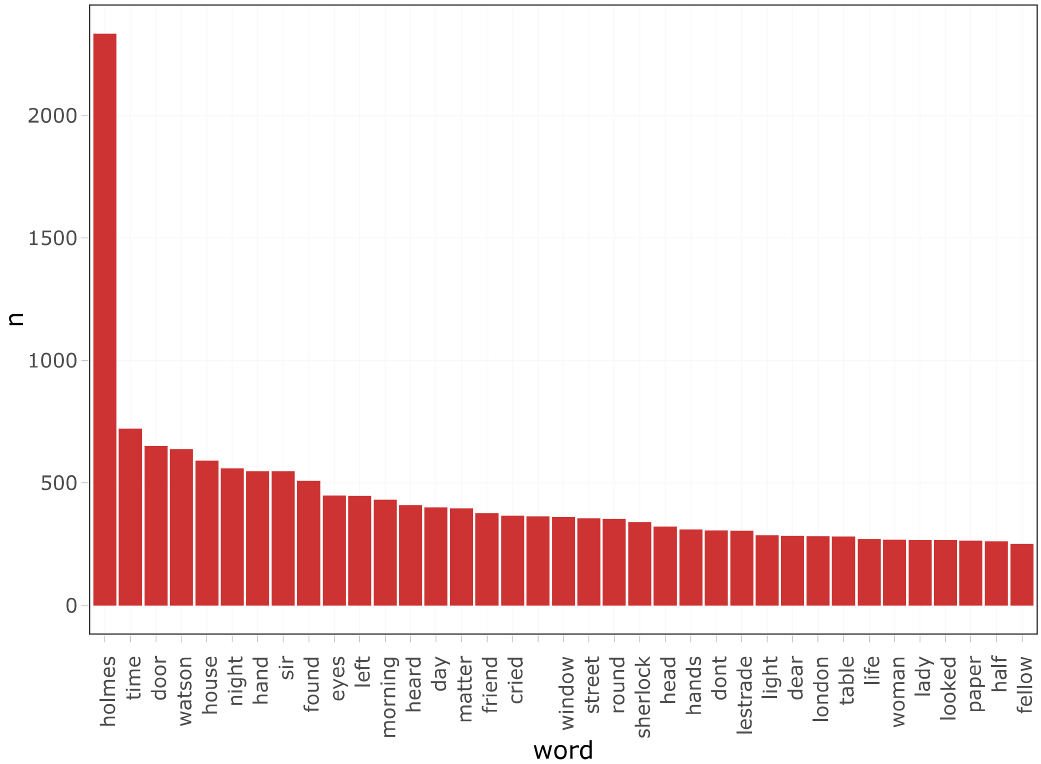
Now that you have an idea of what these stopwords are, we need to remove these words from `tidy_books` dataframe. This is done by removing the words which are common to both `tidy_books` and `stop_words`, which is basically performing an operation opposite of join. Hence, we are going to use `anti_join` function for this.

```
tidy_books <- tidy_books %>% anti_join(stop_words)
```

Now we look into the same bar diagram for most appearing words.

```
p <- tidy_books %>% count(word, sort = TRUE) %>% dplyr::filter(n > 250) %>% mutate(word = reorder(wor
  d, desc(n))) %>%
    ggplot(aes(x = word, y = n)) +
    geom_col(fill = "brown3") +
    theme_bw() +
    theme(axis.text.x = element_text(angle = 90))

ggplotly(p)
```



Now we find that the most appearing word is `holmes` , which not a stopword since it is not very common in English, however, it should be very common in all texts of Sherlock Holmes. Hence, there are some very common words, which is common for the corpus we are investigating, and we need a proper way to remove these words, based on some automatic thresholding. Tf-idf (https://en.wikipedia.org/wiki/Tf%E2%80%93idf) comes to the rescue!

3.3 Counting tf-idf

Firstly, we shall create term document matrix. For this, we create the grouping by `title` and `word` , and then count the number of such appearences in this combination.

```
term_doc <- tidy_books %>% group_by(title, word) %>% summarise(count = n())
```

Now that we have the term document matrix, we can create the tf-idf using `bind_tf_idf` function.

```
term_doc <- term_doc %>% bind_tf_idf(word, title, count)
term_doc[1:1000, ]
```

title <chr>	word <chr>	count <int>	tf <dbl>	idf <dbl>	tf_idf <dbl>
A Study in Scarlet		87	0.0059568641	NA	NA
A Study in Scarlet	abandon	3	0.0002054091	0.6931472	1.423787e-04
A Study in Scarlet	abandoned	3	0.0002054091	0.2876821	5.909252e-05
A Study in Scarlet	aboard	1	0.0000684697	0.6931472	4.745958e-05

title <chr>	word <chr>	count <int>	tf <dbl>	idf <dbl>	tf_idf <dbl>
A Study in Scarlet	abroad	1	0.0000684697	0.2876821	1.969751e-05
A Study in Scarlet	abruptly	1	0.0000684697	0.2876821	1.969751e-05
A Study in Scarlet	absence	3	0.0002054091	0.0000000	0.000000e+00
A Study in Scarlet	absent	5	0.0003423485	0.0000000	0.000000e+00
A Study in Scarlet	absentee	1	0.0000684697	1.3862944	9.491916e-05
A Study in Scarlet	absolute	2	0.0001369394	0.0000000	0.000000e+00

1-10 of 1,000 rows Previous **1** 2 3 4 5 6 ... 100 Next

Let us see some words for which `tf-idf` is 0, these words are those which appears in almost all the documents under consideration.

```
common_words <- term_doc %>%
  filter(tf_idf == 0) %>%
  group_by(word) %>%
  summarise(count = sum(count))%>%
  arrange(desc(count))
common_words[1:100, ]
```

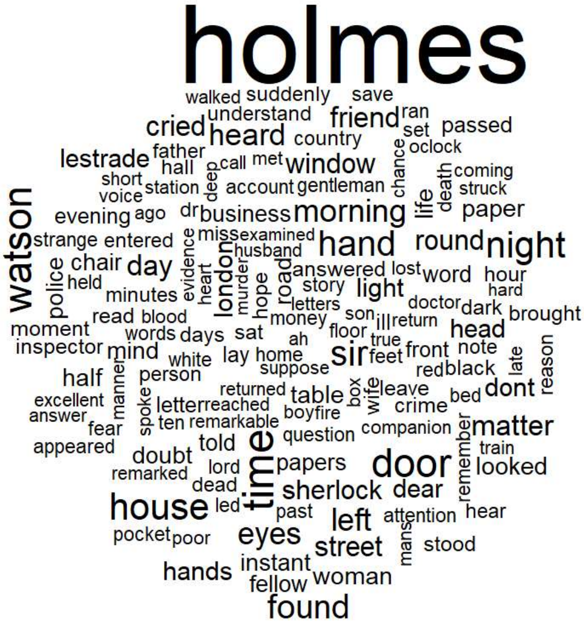
word <chr>	count <int>
holmes	2334
time	722
door	651
watson	638
house	591
night	559
hand	548
sir	547
found	508
eyes	448

1-10 of 100 rows Previous **1** 2 3 4 5 6 ... 10 Next

Now we see that holmes is really a very common word, which appears in all of the texts, hence does not have much meaning to understand the differences between the stories of Sherlock holmes.

Let us make an wordcloud out of these most appearing words, which possibly describes the generality of stories of Sherlock Holmes.

```
library(wordcloud)
common_words %>% with(wordcloud(word, count, max.words = 150))
```



So, now we remove these common words from the term document dataframe, which will only retain the meaningful words.

```
term_doc <- term_doc %>% anti_join(common_words, by = c("word" = "word"))
```

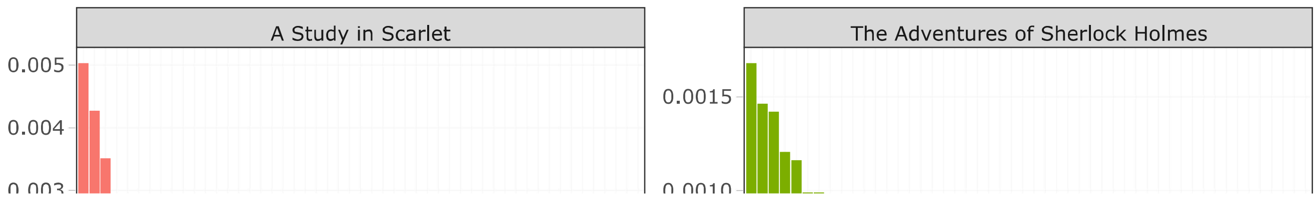
Note that, there are 4 books available in our data. Now, corresponding to each book, we rearrange the rows in decreasing tf-idf, obtaining the important words for each book, so that we can visualize them in a suitable plot.

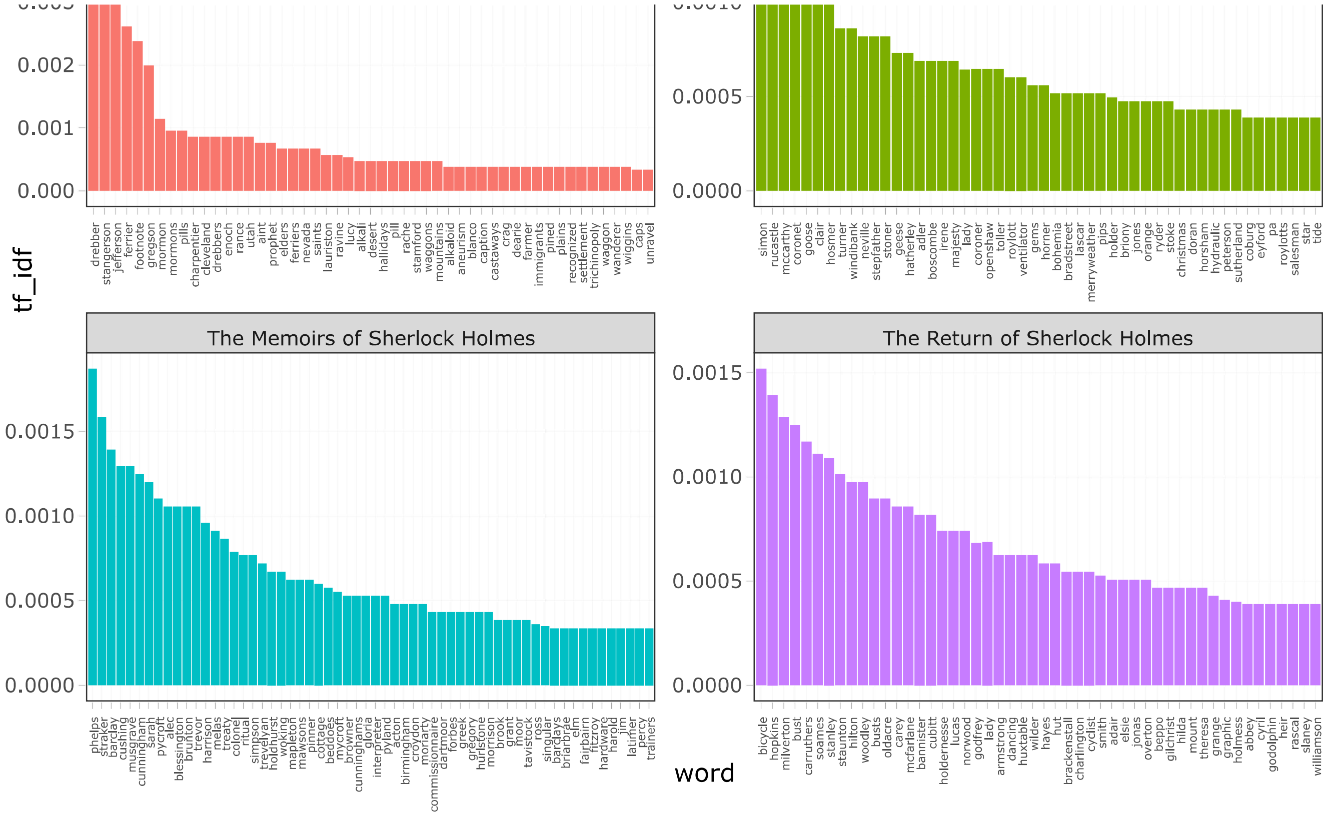
```
table(term_doc$title)
```

```
            A Study in Scarlet The Adventures of Sherlock Holmes
                          3094                              5193
   The Memoirs of Sherlock Holmes     The Return of Sherlock Holmes
                          4793                              5407
```

```
p <- term_doc %>%
  group_by(title) %>%
  arrange(desc(tf_idf)) %>%
  top_n(50) %>%
  ungroup() %>%
  mutate(word = reorder(word, desc(tf_idf) )) %>%
  ggplot(aes(x = word, y = tf_idf, fill = title)) +
  geom_col() +
  facet_wrap(~ title, scales = "free") +
  theme_bw() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1, size = 5), legend.position = "none")

ggplotly(p, width = 800, height = 600)
```




tf_idf
word
0.000
0.001
0.002
drebber stangerson jefferson ferrier footnote gregson mormon mormons pills charpentier cleveland drebbers enoch rance utah aint prophet elders ferriers nevada saints lauriston ravine lucy alkali desert hallidays pill rache stamford waggons mountains alkaloid aneurism blanco caption castaways crag deane farmer immigrants pined plains recognized settlement trichinopoly waggon wanderer wiggins caps unravel
0.0000
0.0005
simon rucastle mccarthy coronet goose clair hosmer turner windibank neville stepfather stoner geese hatherley adler boscombe irene majesty lady coroner openshaw toller roylott ventilator gems horner bohemia bradstreet lascar merryweather pips holder briony jones orange ryder stoke christmas doran horsham hydraulic peterson sutherland coburg eyford pa roylotts salesman star tide
The Memoirs of Sherlock Holmes
0.0000
0.0005
0.0010
0.0015
phelps straker barclay cushing musgrave cunningham sarah pycroft alec blessington brunton trevor harrison melas treaty colonel ritual simpson trevelyan holdhurst woking mapleton mawsons pinner cottage beddoes mycroft browner cunninghams gloria interpreter pyland acton birmingham croydon moriarty commissionaire dartmoor forbes greek gregory hurlstone morrison brook grant moor tavistock ross singular barclays briarbrae elm fairbairn fitzroy hardware harold jim latimer percy trainers
The Return of Sherlock Holmes
0.0000
0.0005
0.0010
0.0015
bicycle hopkins milverton bust carruthers soames stanley staunton hilton woodley busts oldacre carey mcfarlane bannister cubitt holdernesse lucas norwood godfrey lady armstrong dancing huxtable wilder hayes hut brackenstall charlington cyclist smith adair elsie jonas overton beppo gilchrist hilda mount theresa grange graphic holmess abbey cyril godolphin heir rascal slaney williamson