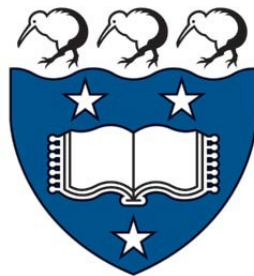


# Text Analytics

Jason Peter Cairns

Supervised by Chris Wild



Bachelor of Science (Honours)  
Department of Statistics  
The University of Auckland  
New Zealand

# Todo list

should this be an abstract? . . . . .	9
common functions: sentiment, summarisation, scoring . . . . .	9
Existing Systems . . . . .	9
current issues . . . . .	9
What iNZight is - capabilities, popularity, etc. . . . .	9
how our program fits in - shiny, inzicht lite etc. . . . .	9
Copy over from notes, flesh out a bit . . . . .	9
Praise tidytext book, complain about the package . . . . .	9
types of text that we can work with: novels, free response data etc. . . . .	9
discuss limitations placed: not going into linguistic territory etc. . . . .	9
Explain broadness of term . . . . .	10
compile glossary from terms here . . . . .	10
Areas of text analytics in a data science framework . . . . .	10
what we have done . . . . .	10
what we haven't done . . . . .	10
terms and their centrality . . . . .	10
generalisation: n-grams, sentences etc. . . . .	10
computer science vs statistics - reflection in data science . . . . .	10
why process . . . . .	10
stopwords, lemmatisation etc. . . . .	10
modelling vs db joins - more info in notes . . . . .	10
why compute scores & statistics . . . . .	10
scoring - tf-idf, word count . . . . .	10
Suggestions for further research - more on the statistics of words . . . . .	10
recount the book of John text analysis . . . . .	10
why sentiment . . . . .	10
Process of sentiment . . . . .	10
sentiment modelling vs db joins . . . . .	10
our implementation and why . . . . .	10
reviews . . . . .	10
issues . . . . .	10
why compute summarisations . . . . .	10
lexrank, textrank - include notes on lexrank . . . . .	10

other methods . . . . .	10
reddit bot example . . . . .	10
topic modelling . . . . .	10
Term correlation . . . . .	10
modelling based on linguistic features . . . . .	10
talk about score vs structure . . . . .	10
complain about tag clouds . . . . .	10
talk about ggpage . . . . .	10
discuss our experimentations with some alternative visualisations . . .	10
why R . . . . .	11
Why Shiny . . . . .	11
why tidyverse . . . . .	11
possible future: datatables, futures, etc. . . . .	11
why functional . . . . .	11
Why lossless data . . . . .	11
Why structure it like it has been . . . . .	11
make graph of architecture . . . . .	11
Describe package and package creation . . . . .	11
following three sections copy and paste from the notes - buffing up as necessary . . . . .	11
include screenshots . . . . .	11
summarise successes . . . . .	12
summarise failures . . . . .	12
general thoughts on the topic . . . . .	12
educational potential of text analytics . . . . .	12

# Acknowledgements

# Contents

<b>Listings</b>	<b>5</b>
<b>Tables</b>	<b>6</b>
<b>Figures</b>	<b>7</b>
<b>1 Introduction</b>	<b>9</b>
1.1 Intention . . . . .	9
1.2 Background: Text Analytics (incl. examples) . . . . .	9
1.3 Background: inZight . . . . .	9
1.4 Literature Review (existing packages in R) . . . . .	9
1.5 Scope of work . . . . .	9
<b>2 Text Analytics Prolusion</b>	<b>10</b>
2.1 overview . . . . .	10
2.2 terms . . . . .	10
2.3 Historical Background . . . . .	10
2.4 Processing . . . . .	10
2.5 scores & statistics . . . . .	10
2.6 Sentiment . . . . .	10
2.7 Summarisation . . . . .	10
2.8 what we didn't do (yet) . . . . .	10
2.9 Visualisation . . . . .	10
<b>3 Program Structure</b>	<b>11</b>
3.1 Program Architecture . . . . .	11
3.2 Import . . . . .	11
3.3 Insight . . . . .	11
3.4 Visualisation . . . . .	11
3.5 User Interface . . . . .	11
<b>4 Conclusion</b>	<b>12</b>
4.1 Summary . . . . .	12
4.2 Recommendations . . . . .	12

<b>5 Appendix</b>	<b>13</b>
Glossary	30
Index	30
Bibliography	31

## List of source codes

## List of Tables



## List of Figures

# Chapter 1

## Introduction

### 1.1 Intention

Text Analytics serves to glean insight from a body of text. Within the broad category of text analytics, we seek to answer questions about what the text is communicating, what is felt about it, and how this information is structured. In this dissertation, we demonstrate the creation of a user-friendly program to perform text analytics functions using modern R with the Shiny web application framework. In a literate style, we illustrate top-down the structure of such a program, as well as the data structures and computational processes that have established their value for such a program.

### 1.2 Background: Text Analytics (incl. examples)

### 1.3 Background: inZight

### 1.4 Literature Review (existing packages in R)

### 1.5 Scope of work

should  
this be  
an ab-  
stract?

common  
func-  
tions:  
senti-  
ment,  
sum-  
mar-  
isation,  
scoring

Existing  
Systems

current  
issues

What  
inZight  
is - cap-  
abilities,  
popular-  
ity, etc.

how our  
program  
fits in

## Chapter 2

# Text Analytics Prolusion

### 2.1 overview

### 2.2 terms

term

### 2.3 Historical Background

### 2.4 Processing

### 2.5 scores & statistics

### 2.6 Sentiment

### 2.7 Summarisation

### 2.8 what we didn't do (yet)

### 2.9 Visualisation

Explain  
broad-  
ness of  
term

compile  
glossary  
from  
terms  
here

Areas  
of text  
analyt-  
ics in  
a data  
science  
frame-  
work

what  
we have  
done

what we  
haven't  
done

terms  
and  
their  
central-  
ity

generalisation:  
n-  
grams,  
sen-  
tences  
etc.

## Chapter 3

# Program Structure

### 3.1 Program Architecture

### 3.2 Import

### 3.3 Insight

### 3.4 Visualisation

### 3.5 User Interface

why R

Why  
Shiny

why  
tidy-  
verse

possible  
future:  
data-  
tables,  
futures,  
etc.

why  
func-  
tional

Why  
lossless  
data

Why  
struc-  
ture  
it like  
it has  
been

make  
graph of  
architec-  
ture

Describe  
pack-  
age and  
package  
creation

following  
three

## Chapter 4

# Conclusion

### 4.1 Summary

### 4.2 Recommendations

summarise  
suc-  
cesses

summarise  
failures

general  
thoughts  
on the  
topic

educational  
poten-  
tial of  
text  
analyt-  
ics

## Chapter 5

# Appendix

The following pages are a copy of the documentation for the R package created as a part of this dissertation. They were automatically generated through the Roxygen2 system.

# Package ‘inzightta’

August 16, 2019

**Title** iNZight Text Analytics

**Version** 0.0.0.9000

**Description** Provides text analytics functions for the importation, analysis, and visualisation of text. This package is designed specifically for output in shiny, with the analytical functions all working well with dplyr tools.

**License** GPL-3

**Encoding** UTF-8

**LazyData** true

**Imports** readr,  
tibble,  
stringr,  
dplyr,  
readxl,  
purrr,  
tidytext,  
textstem,  
magrittr,  
stats,  
textrank,  
lexRankr

**RoxygenNote** 6.1.1

## R topics documented:

aggregate_sentiment . . . . .	2
determine_stopwords . . . . .	3
format_data . . . . .	3
get_bigram . . . . .	4
get_chapters . . . . .	4
get_filetype . . . . .	5
get_parts . . . . .	5
get_search . . . . .	6

get_sections . . . . .	6
get_sw . . . . .	7
get_valid_input . . . . .	7
ifexp . . . . .	8
import_base_file . . . . .	8
import_csv . . . . .	9
import_excel . . . . .	9
import_files . . . . .	10
import_txt . . . . .	10
index_bigram . . . . .	11
keywords_tr . . . . .	11
key_sentences . . . . .	12
table_textcol . . . . .	12
term_count . . . . .	13
term_freq . . . . .	13
text_prep . . . . .	14
ungroup_by . . . . .	14
word_sentiment . . . . .	15
<b>Index</b>	<b>16</b>

---

aggregate_sentiment	<i>Get statistics for sentiment over some group, such as sentence.</i>
---------------------	------------------------------------------------------------------------

---

## Description

Get statistics for sentiment over some group, such as sentence.

## Usage

```
aggregate_sentiment(.data, aggregate_on, statistic)
```

## Arguments

.data	character vector of words
aggregate_on	vector to aggregate .data over; ideally, sentence_id, but could be chapter, document, etc.
statistic	function that accepts na.rm argument; e.g. mean, median, sd.



---

determine_stopwords	<i>determine stopword status</i>
---------------------	----------------------------------

---

**Description**

determine stopword status

**Usage**

```
determine_stopwords(.data, ...)
```

**Arguments**

.data	vector of words
...	arguments of get_sw

**Value**

a [tibble][tibble::tibble-package] equivalent to the input dataframe, with an additional stopword column

---

format_data	<i>formats imported data into an analysis-ready format</i>
-------------	------------------------------------------------------------

---

**Description**

formats imported data into an analysis-ready format

**Usage**

```
format_data(data)
```

**Arguments**

data	a tibble formatted with a text and (optional) group column
------	------------------------------------------------------------

**Value**

a [tibble][tibble::tibble-package] formatted such that columns correspond to identifiers of group, line, sentence, word (groups ignored)

---

get_bigram	<i>Determine bigrams</i>
------------	--------------------------

---

**Description**

Determine bigrams

**Usage**

```
get_bigram(.data)
```

**Arguments**

.data	character vector of words
-------	---------------------------

**Value**

character vector of bigrams

---

get_chapters	<i>sections text based on chapters</i>
--------------	----------------------------------------

---

**Description**

sections text based on chapters

**Usage**

```
get_chapters(.data)
```

**Arguments**

.data	vector to section
-------	-------------------

**Value**

vector of same length as .data with chapter numbers

---

get_filetype	<i>Get filetype</i>
--------------	---------------------

---

**Description**

Get filetype

**Usage**

```
get_filetype(filepath)
```

**Arguments**

filepath	string filepath of document
----------	-----------------------------

**Value**

filetype (string) - NA if no extension

---

get_parts	<i>sections text based on parts</i>
-----------	-------------------------------------

---

**Description**

sections text based on parts

**Usage**

```
get_parts(.data)
```

**Arguments**

.data	vector to section
-------	-------------------

**Value**

vector of same length as .data with part numbers

---

get_search	<i>creates a search closure to section text</i>
------------	-------------------------------------------------

---

**Description**

creates a search closure to section text

**Usage**

```
get_search(search)
```

**Arguments**

search	a string regexp for the term to seperate on, e.g. "Chapter"
--------	-------------------------------------------------------------

**Value**

closure over search expression

---

get_sections	<i>sections text based on sections</i>
--------------	----------------------------------------

---

**Description**

sections text based on sections

**Usage**

```
get_sections(.data)
```

**Arguments**

.data	vector to section
-------	-------------------

**Value**

vector of same length as .data with section numbers

---

get_sw	<i>Gets stopwords from a default list and user-provided list</i>
--------	------------------------------------------------------------------

---

**Description**

Gets stopwords from a default list and user-provided list

**Usage**

```
get_sw(lexicon = "snowball", addl = NA)
```

**Arguments**

lexicon	a string name of a stopwords list, one of "smart", "snowball", or "onix"
addl	user defined character vector of additional stopwords, each element being a stop-word

**Value**

a [tibble][tibble::tibble-package] with one column named "word"

---

get_valid_input	<i>helper function to get valid input (recursively)</i>
-----------------	---------------------------------------------------------

---

**Description**

helper function to get valid input (recursively)

**Usage**

```
get_valid_input(options, init = TRUE)
```

**Arguments**

options	vector of options that valid input should be drawn from
init	whether this is the initial attempt, used only as recursive information

**Value**

readline output that exists in the vector of options

---

ifexp	<i>scheme-like if expression, without restriction of returning same-size table of .test, as ifelse() does</i>
-------	---------------------------------------------------------------------------------------------------------------

---

**Description**

scheme-like if expression, without restriction of returning same-size table of .test, as ifelse() does

**Usage**

```
ifexp(.test, true, false)
```

**Arguments**

.test	predicate to test
true	expression to return if .test evals to TRUE
false	expression to return if .test evals to TRUE

**Value**

either true or false

---

import_base_file	<i>Base case for file import</i>
------------------	----------------------------------

---

**Description**

Base case for file import

**Usage**

```
import_base_file(filepath)
```

**Arguments**

filepath	string filepath of file for import
----------	------------------------------------

**Value**

imported file with document id

---

import_csv	<i>Import csv file</i>
------------	------------------------

---

**Description**

Import csv file

**Usage**

```
import_csv(filepath)
```

**Arguments**

filepath            a string indicating the relative or absolute filepath of the file to import

**Value**

a [tibble][tibble::tibble-package] of each row corresponding to a line of the text file, with the column named "text"

---

import_excel	<i>Import excel file</i>
--------------	--------------------------

---

**Description**

Import excel file

**Usage**

```
import_excel(filepath)
```

**Arguments**

filepath            a string indicating the relative or absolute filepath of the file to import

**Value**

a [tibble][tibble::tibble-package] of each row corresponding to a line of the text file, with the column named "text"

---

import_files	<i>Import any number of files</i>
--------------	-----------------------------------

---

**Description**

Import any number of files

**Usage**

```
import_files(filepaths)
```

**Arguments**

filepaths	char vector of filepaths
-----------	--------------------------

**Value**

a [tibble][tibble::tibble-package] imported files with document id

---

import_txt	<i>Import text file</i>
------------	-------------------------

---

**Description**

Import text file

**Usage**

```
import_txt(filepath)
```

**Arguments**

filepath	a string indicating the relative or absolute filepath of the file to import
----------	-----------------------------------------------------------------------------

**Value**

a [tibble][tibble::tibble-package] of each row corresponding to a line of the text file, with the column named "text"



---

index_bigram	<i>get bigram at index i of list1 &amp; 2</i>
--------------	-----------------------------------------------

---

**Description**

get bigram at index i of list1 & 2

**Usage**

```
index_bigram(i, list1, list2)
```

**Arguments**

i	numeric index to attain bigram at
list1	list or vector for first bigram token
list2	list or vector for second bigram token

**Value**

bigram of list1 and list2 at index i, skipping NA's

---

keywords_tr	<i>Determine textrank score for vector of words</i>
-------------	-----------------------------------------------------

---

**Description**

Determine textrank score for vector of words

**Usage**

```
keywords_tr(.data)
```

**Arguments**

.data	character vector of words
-------	---------------------------

**Value**

vector of scores for each word

---

key_sentences	<i>get score for key sentences as per Lexrank</i>
---------------	---------------------------------------------------

---

**Description**

get score for key sentences as per Lexrank

**Usage**

```
key_sentences(.data, aggregate_on)
```

**Arguments**

.data	character vector of words
aggregate_on	vector to aggregate .data over; ideally, sentence_id

---

table_textcol	<i>Interactively determine and automatically mark the text column of a table</i>
---------------	----------------------------------------------------------------------------------

---

**Description**

Interactively determine and automatically mark the text column of a table

**Usage**

```
table_textcol(data)
```

**Arguments**

data	dataframe with column requiring marking
------	-----------------------------------------

**Value**

same dataframe with text column renamed to "text"

---

term_count	<i>Determine the number of terms at each aggregate level</i>
------------	--------------------------------------------------------------

---

**Description**

Determine the number of terms at each aggregate level

**Usage**

```
term_count(.data, aggregate_on)
```

**Arguments**

.data	character vector of terms
aggregate_on	vector to split .data on for insight

**Value**

vector of number of terms for each aggregate level, same length as .data

---

term_freq	<i>Determine term frequency</i>
-----------	---------------------------------

---

**Description**

Determine term frequency

**Usage**

```
term_freq(.data)
```

**Arguments**

.data	character vector of terms
-------	---------------------------

**Value**

numeric vector of term frequencies

---

text_prep	<i>takes imported one-line-per-row data and prepares it for later analysis</i>
-----------	--------------------------------------------------------------------------------

---

**Description**

takes imported one-line-per-row data and prepares it for later analysis

**Usage**

```
text_prep(.data, lemmatize = TRUE, stopwords = TRUE,
          sw_lexicon = "snowball", addl_stopwords = NA)
```

**Arguments**

.data	tibble with one line of text per row
lemmatize	boolean, whether to lemmatize or not
stopwords	boolean, whether to remove stopwords or not
sw_lexicon	string, lexicon with which to remove stopwords
addl_stopwords	char vector of user-supplied stopwords

**Value**

a [tibble][tibble::tibble-package] with one token per line, stopwords removed leaving NA values, column for analysis named "text"

---

ungroup_by	<i>helper function to ungroup for dplyr. functions equivalently to group_by() but with standard (string) evaluation</i>
------------	-------------------------------------------------------------------------------------------------------------------------

---

**Description**

helper function to ungroup for dplyr. functions equivalently to group\_by() but with standard (string) evaluation

**Usage**

```
ungroup_by(x, ...)
```

**Arguments**

x	tibble to perform function on
...	string of groups to ungroup on

**Value**

x with ... no longer grouped upon

---

word_sentiment	<i>Determine sentiment of words</i>
----------------	-------------------------------------

---

**Description**

Determine sentiment of words

**Usage**

```
word_sentiment(.data, lexicon = "afinn")
```

**Arguments**

.data	vector of words
lexicon	sentiment lexicon to use, based on the corpus provided by tidytext

**Value**

vector with sentiment score of each word in the vector

# Index

`aggregate_sentiment`, [2](#)

`determine_stopwords`, [3](#)

`format_data`, [3](#)

`get_bigram`, [4](#)

`get_chapters`, [4](#)

`get_filetype`, [5](#)

`get_parts`, [5](#)

`get_search`, [6](#)

`get_sections`, [6](#)

`get_sw`, [7](#)

`get_valid_input`, [7](#)

`ifexp`, [8](#)

`import_base_file`, [8](#)

`import_csv`, [9](#)

`import_excel`, [9](#)

`import_files`, [10](#)

`import_txt`, [10](#)

`index_bigram`, [11](#)

`key_sentences`, [12](#)

`keywords_tr`, [11](#)

`table_textcol`, [12](#)

`term_count`, [13](#)

`term_freq`, [13](#)

`text_prep`, [14](#)

`ungroup_by`, [14](#)

`word_sentiment`, [15](#)

# Glossary

**term** “a word or expression that has a precise meaning in some uses or is peculiar to a science, art, profession, or subject”[1] — here text analysts have capitalised on the generalisation of “term” to include subcomponents or aggregations of words. 9

# Bibliography

- [1] Merriam-Webster Dictionary, ed. *Term — Definition of Term*. 17th Aug. 2019. URL: <https://www.merriam-webster.com/dictionary/term> (cit. on p. 30).