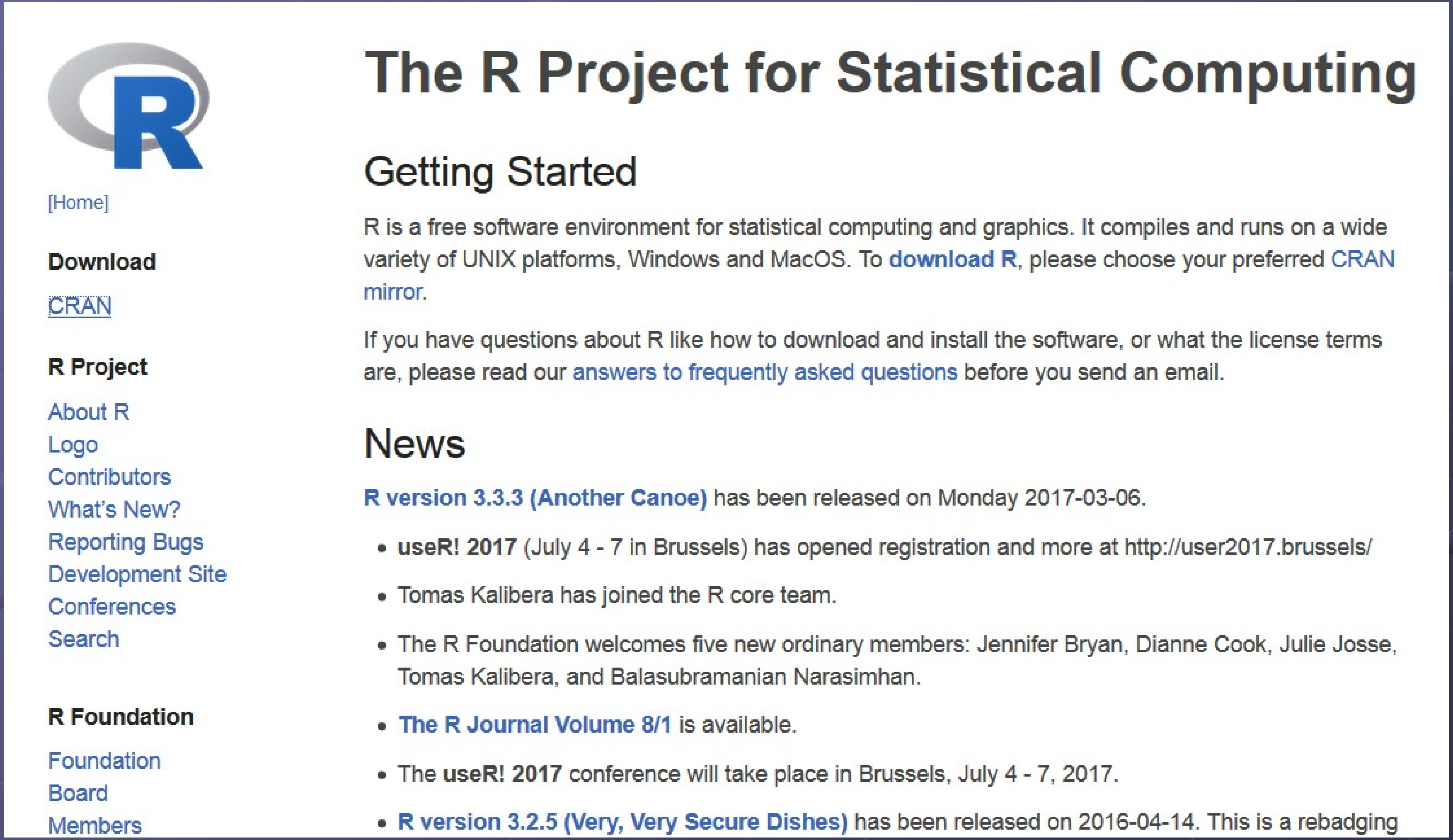
Using R for Text Mining



[Home]

Download

CRAN

R Project

About R
Logo
Contributors
What's New?
Reporting Bugs
Development Site
Conferences
Search

R Foundation

Foundation
Board
Members

The R Project for Statistical Computing

Getting Started

R is a free software environment for statistical computing and graphics. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS. To **download R**, please choose your preferred CRAN mirror.

If you have questions about R like how to download and install the software, or what the license terms are, please read our answers to frequently asked questions before you send an email.

News

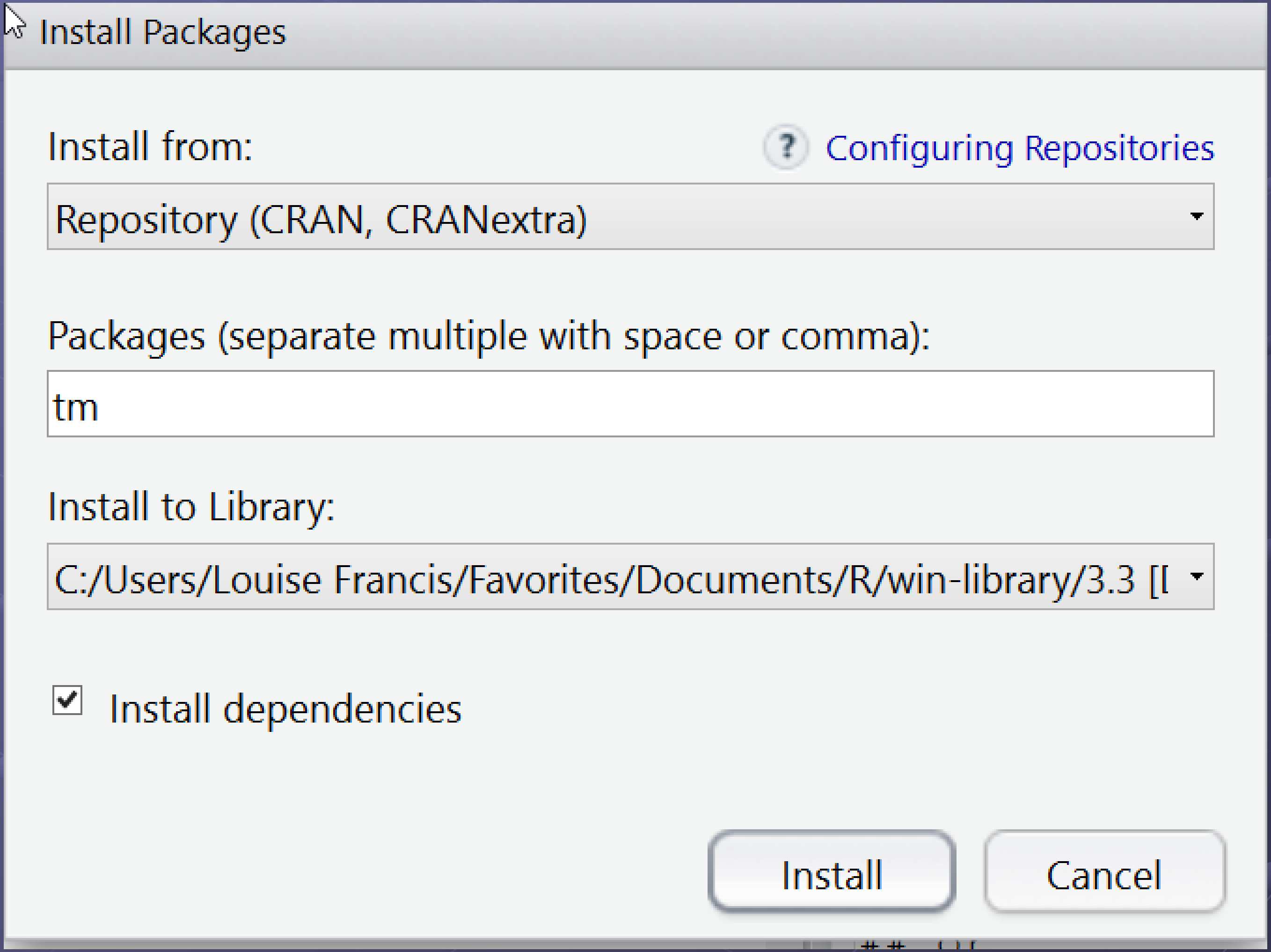
R version 3.3.3 (Another Canoe) has been released on Monday 2017-03-06.

- **useR! 2017** (July 4 - 7 in Brussels) has opened registration and more at http://user2017.brussels/
- Tomas Kalibera has joined the R core team.
- The R Foundation welcomes five new ordinary members: Jennifer Bryan, Dianne Cook, Julie Josse, Tomas Kalibera, and Balasubramanian Narasimhan.
- **The R Journal Volume 8/1** is available.
- The **useR! 2017** conference will take place in Brussels, July 4 - 7, 2017.
- **R version 3.2.5 (Very, Very Secure Dishes)** has been released on 2016-04-14. This is a rebadging

R Text Mining

Assumes knowledge of R but not how to text mine in R

Install the tm package



Install Packages

Install from:

Repository (CRAN, CRANextra)

Packages (separate multiple with space or comma):

tm

Install to Library:

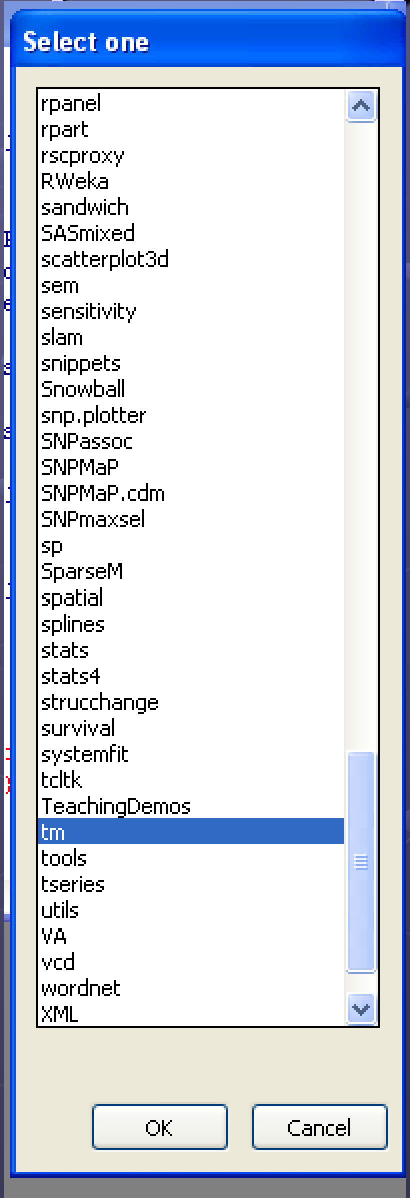
C:/Users/Louise Francis/Favorites/Documents/R/win-library/3.3 [I

☑ Install dependencies

Load the tm package

- The "library" command loads the bundled functions of the desired library. Help describing those functions is a short command away.
- >library(tm) #loads library

- >help(package=tm)



What we do with **tm**

- Apply functions similar to Perl regular expression and parsing methods
- Statistically analyze the data to derive content from unstructured text data

TM – data sources & readers

```
> #**************************************************************************
> # for an up-to-date list of available data sources and readers
```

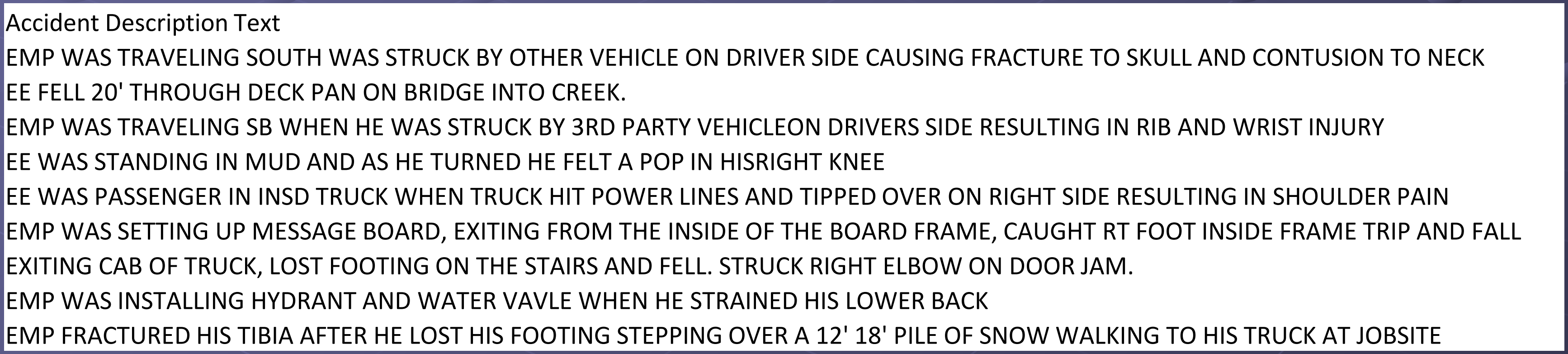
- #**
- **getSources()**
- "DataframeSource" "DirSource"   "URISource"

```
                  "VectorSource"    "XMLSource"       "ZipSource"
```

- **getReaders()**

```
                  "DataframeSource" "DirSource"        "URISource"
                  "VectorSource"    "XMLSource"       "ZipSource"
```

The Data
WC Accident Description



Accident Description Text
EMP WAS TRAVELING SOUTH WAS STRUCK BY OTHER VEHICLE ON DRIVER SIDE CAUSING FRACTURE TO SKULL AND CONTUSION TO NECK
EE FELL 20' THROUGH DECK PAN ON BRIDGE INTO CREEK.
EMP WAS TRAVELING SB WHEN HE WAS STRUCK BY 3RD PARTY VEHICLEON DRIVERS SIDE RESULTING IN RIB AND WRIST INJURY
EE WAS STANDING IN MUD AND AS HE TURNED HE FELT A POP IN HISRIGHT KNEE
EE WAS PASSENGER IN INSD TRUCK WHEN TRUCK HIT POWER LINES AND TIPPED OVER ON RIGHT SIDE RESULTING IN SHOULDER PAIN
EMP WAS SETTING UP MESSAGE BOARD, EXITING FROM THE INSIDE OF THE BOARD FRAME, CAUGHT RT FOOT INSIDE FRAME TRIP AND FALL
EXITING CAB OF TRUCK, LOST FOOTING ON THE STAIRS AND FELL. STRUCK RIGHT ELBOW ON DOOR JAM.
EMP WAS INSTALLING HYDRANT AND WATER VAVLE WHEN HE STRAINED HIS LOWER BACK
EMP FRACTURED HIS TIBIA AFTER HE LOST HIS FOOTING STEPPING OVER A 12' 18' PILE OF SNOW WALKING TO HIS TRUCK AT JOBSITE

<<VCorpus>>Metadata: corpus specific: 0, document level (indexed): 0Content: documents: 2[[1]]<<PlainTextDocument>>Met

TM – source documents

```
#*****************************************************************
# read in source document collection
#*****************************************************************
>txt.csv <- read.csv(file="("WC Acc Description.csv"))
#******Inspect first few rows
>head(txt.csv)

# First do the preprocessing
# turn accident description into corpus for package tm
txt <-VCorpus(VectorSource(desc))
inspect(txt[1:2])
```

```
<<VCorpus>>
Metadata:  corpus specific: 0, document level (indexed): 0
Content:  documents: 2

[[1]]
<<PlainTextDocument>>
Metadata:  7
Content:  chars: 58

[[2]]
<<PlainTextDocument>>
Metadata:  7
Content:  chars: 128
```

TM – preprocess: use lowercase

> #**

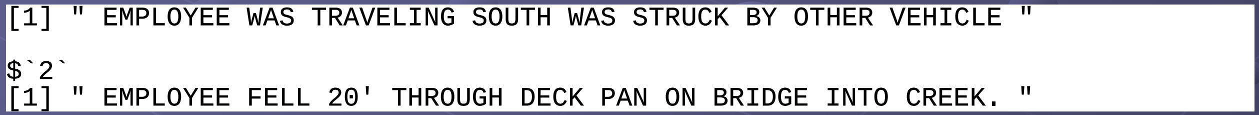
- # a little pre-processing to prep the data for TM
- # strip extra white spaces

> # convert to lower case

> # tmTolower is one of several available text transformations.

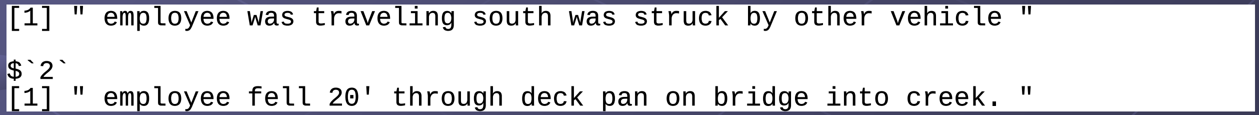
> # To see all currently available use: **getTransformations()**

- #**
- # Extra whitespace is eliminated by:
- txt2 <- tm_map(txt, stripWhitespace)
- lapply(txt2[1:5], as.character)



```
[1] " EMPLOYEE WAS TRAVELING SOUTH WAS STRUCK BY OTHER VEHICLE "

$`2`
[1] " EMPLOYEE FELL 20' THROUGH DECK PAN ON BRIDGE INTO CREEK. "
```

> txt2 <- tm_map(txt2, content_transformer(tolower))
lapply(txt2[1:5], as.character)

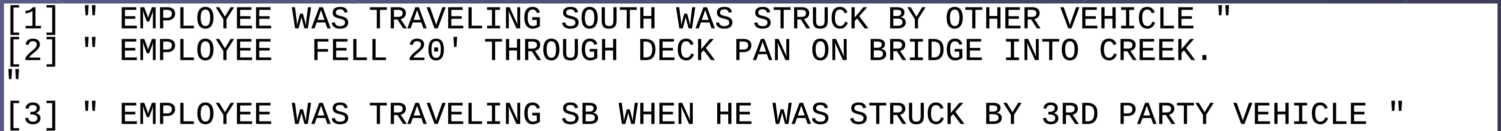


```
[1] " employee was traveling south was struck by other vehicle "

$`2`
[1] " employee fell 20' through deck pan on bridge into creek. "
```

TM – search & replace

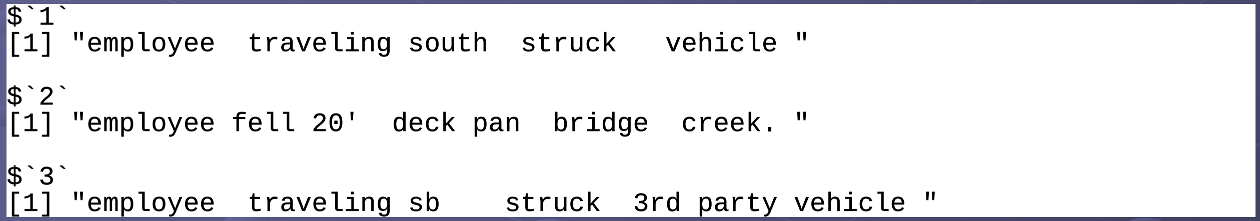
- #**
- # split word HISRIGHT into two pieces
- # Replace all variations of Employee (EE, EMP) with Employee
- # Use Regular Expressions and gsub. Test with grep
- # Apply regular expressions before converting to Corpus in tm
- #**
- # gsub(pattern, replacement, character data)
- desc<-gsub("HISRIGHT","his right",desc)

- desc<-gsub("\\bEMP(\\w*) |\\bEE|^EMP(\\b)"," EMPLOYEE ",desc)
- head(desc)



```
[1] " EMPLOYEE WAS TRAVELING SOUTH WAS STRUCK BY OTHER VEHICLE "
[2] " EMPLOYEE  FELL 20' THROUGH DECK PAN ON BRIDGE INTO CREEK.
"
[3] " EMPLOYEE WAS TRAVELING SB WHEN HE WAS STRUCK BY 3RD PARTY VEHICLE "
```

TM – search & replace con’t

- > #**
- > # remove **stopwords**
- **> text mining functions contain a dictionary of stopwords for removal**
- **> The dictionary is specific to the labguage**
- > #**
- ># Removal of stopwords by:
- >txt2 <- tm_map(txt2, removeWords, stopwords("english"))
- >lapply(txt2[1:10], as.character)
- 

```
$`1`
[1] "employee  traveling south  struck   vehicle "

$`2`
[1] "employee fell 20'  deck pan  bridge  creek. "

$`3`
[1] "employee  traveling sb    struck  3rd party vehicle "
```

- # One can also remove particular words:
- txt2 <- tm_map(txt2, removeWords, c("ees","employee"))
- class(txt2)

TM – search & replace con’t

- > #**
- > # remove **numbers & punctuation**
- > #**
- > txt <- tm_map(txt, **removeNumbers**)
- > txt <- tm_map(txt, **removePunctuation**)
- > lapply(txt2[1:10], as.character)

TM – search & replace con’t

- **# Get a list of the possible transformations**
- **getTransformations()**

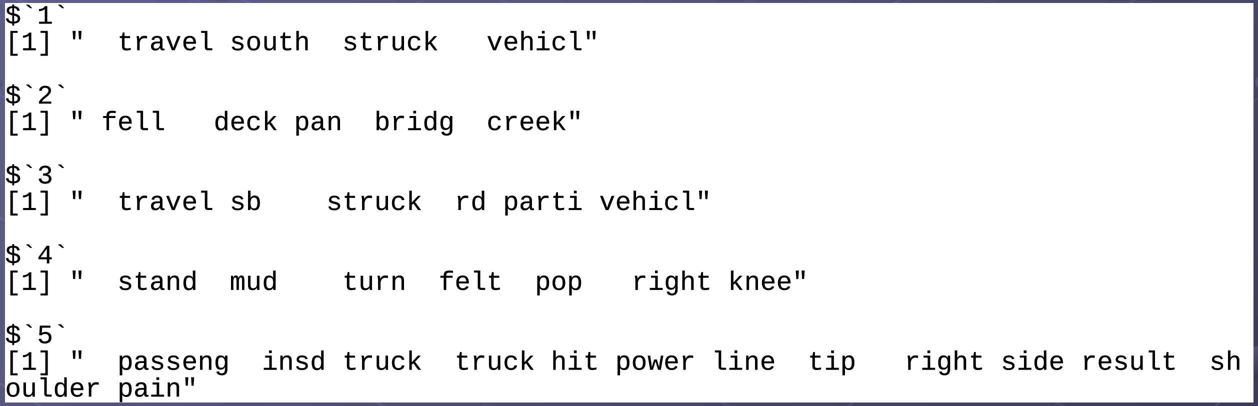
```
[1] "removeNumbers"     "removePunctuation" "removeWords"
[4] "stemDocument"      "stripWhitespace"
```

Stem words

- Many words have several forms such as singular and plural, future and past tense
- For the purposes of text mining the different forms typically convey the same meaning
- Text mining software contains list of stemmed words which is a single representation of all the forms of the word

Stem the Text Data

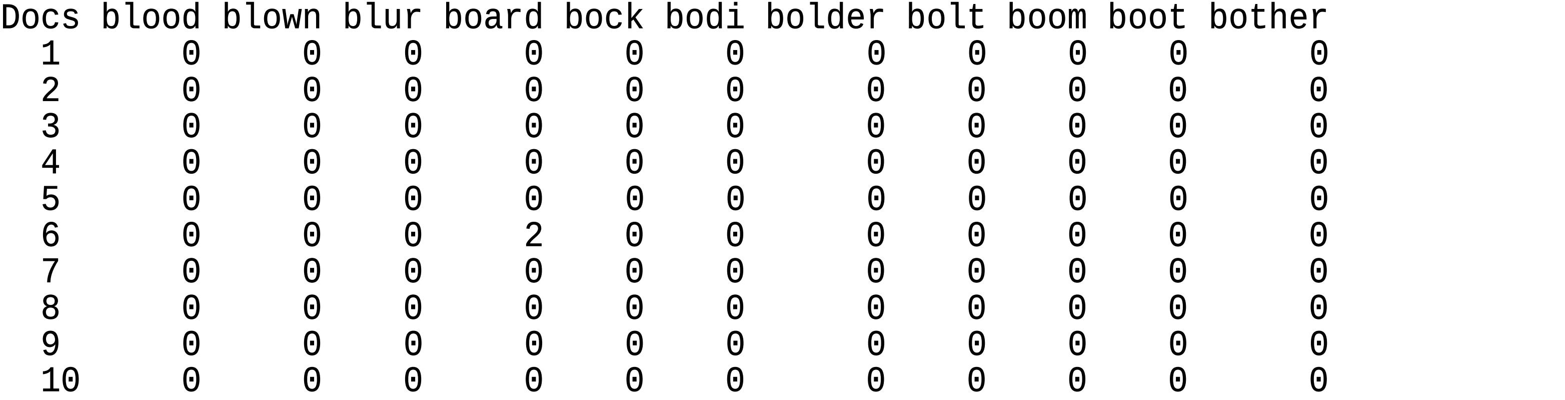
- txt2 <- tm_map(txt2, stemDocument)
- lapply(txt2[1:5], as.character)



```
$`1`
[1] "  travel south  struck   vehicl"

$`2`
[1] " fell   deck pan  bridg  creek"

$`3`
[1] "  travel sb    struck  rd parti vehicl"

$`4`
[1] "  stand  mud    turn  felt  pop   right knee"

$`5`
[1] "  passeng  insd truck  truck hit power line   tip   right side result  sh
oulder pain"
```

TM – Document by Term Matrix

```
> #******************************************************************
> # create a document_by_term matrix
> #******************************************************************
dtm <- DocumentTermMatrix(txt2)
# get number of rows and columns
nrow(dtm)
450
ncol(dtm)
1077

# Take a look at what is in the document term matrix
inspect(dtm[1:10, 100:120])
```



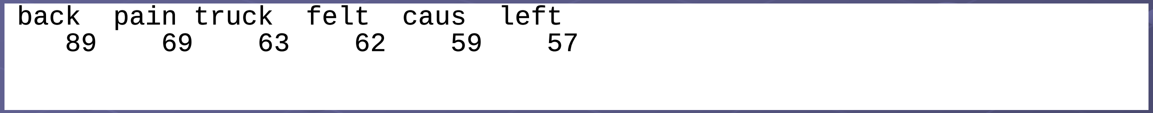
```
Docs blood blown blur board bock bodi bolder bolt boom boot bother
  1      0     0    0     0    0    0      0    0    0    0      0
  2      0     0    0     0    0    0      0    0    0    0      0
  3      0     0    0     0    0    0      0    0    0    0      0
  4      0     0    0     0    0    0      0    0    0    0      0
  5      0     0    0     0    0    0      0    0    0    0      0
  6      0     0    0     2    0    0      0    0    0    0      0
  7      0     0    0     0    0    0      0    0    0    0      0
  8      0     0    0     0    0    0      0    0    0    0      0
  9      0     0    0     0    0    0      0    0    0    0      0
  10     0     0    0     0    0    0      0    0    0    0      0
```

TM – ID Frequent Occuring Words

> #***

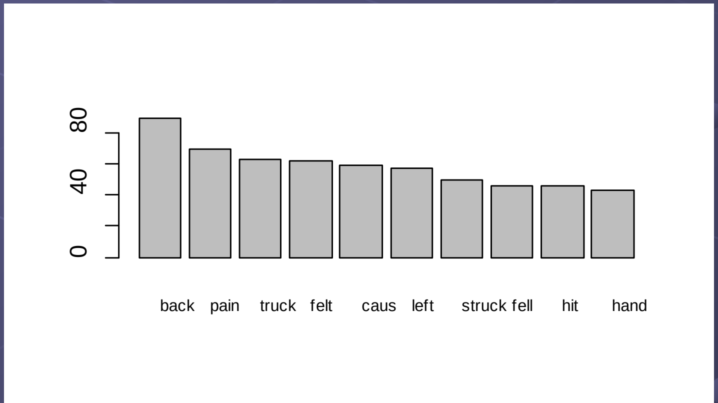
> # **ID frequently occuring words**

- #***
- # Explore our data
- # Organize terms by their frequency:
- freq <- colSums(as.matrix(dtm))
- length(freq)
- freq <- freq[order(freq, decreasing=TRUE)]
- head(freq)



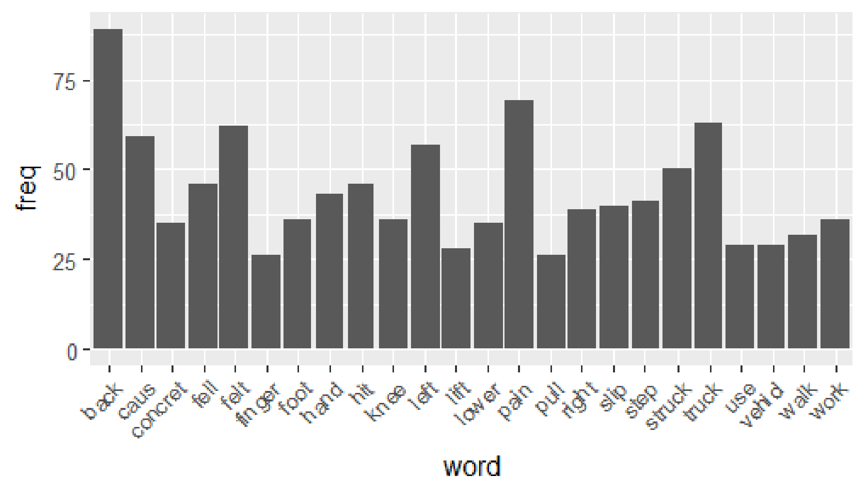
```
back  pain truck  felt  caus  left
   89    69    63    62    59    57
```

# This will identify all terms that appear frequently (in this case, 25 or more times).
findFreqTerms(dtm, 25); # lowfreq =25



Graph of Frequent Terms Using ggplot2

```
# Plot words that appear at least 25 times.
wf <- data.frame(word=names(freq), freq=freq)   # create data set with words and its frequency
library(ggplot2)
p <- ggplot(subset(wf, freq>25), aes(word, freq))
p <- p + geom_bar(stat="identity")
p <- p + theme(axis.text.x=element_text(angle=45, hjust=1))
```



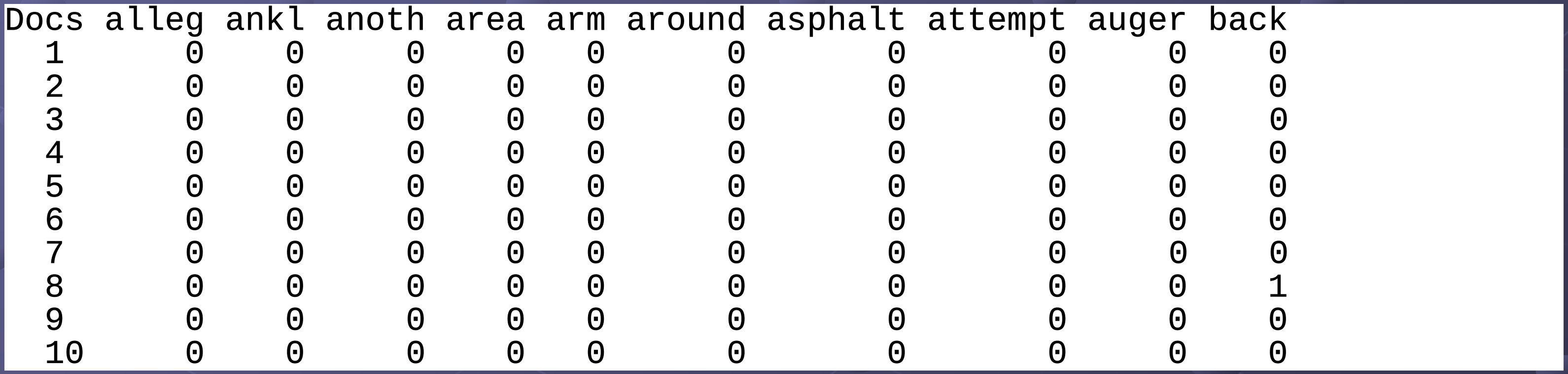
Tighten up DTM and remove sparse terms

> #***

> # tighten up dtm matrix by **removing sparse terms**

- #***
- dtm3 <- removeSparseTerms(dtm, 0.99); # removes terms with 97.5% zeros;
- ncol(dtm3)
- 210

- inspect(dtm3[1:10, 1:15])



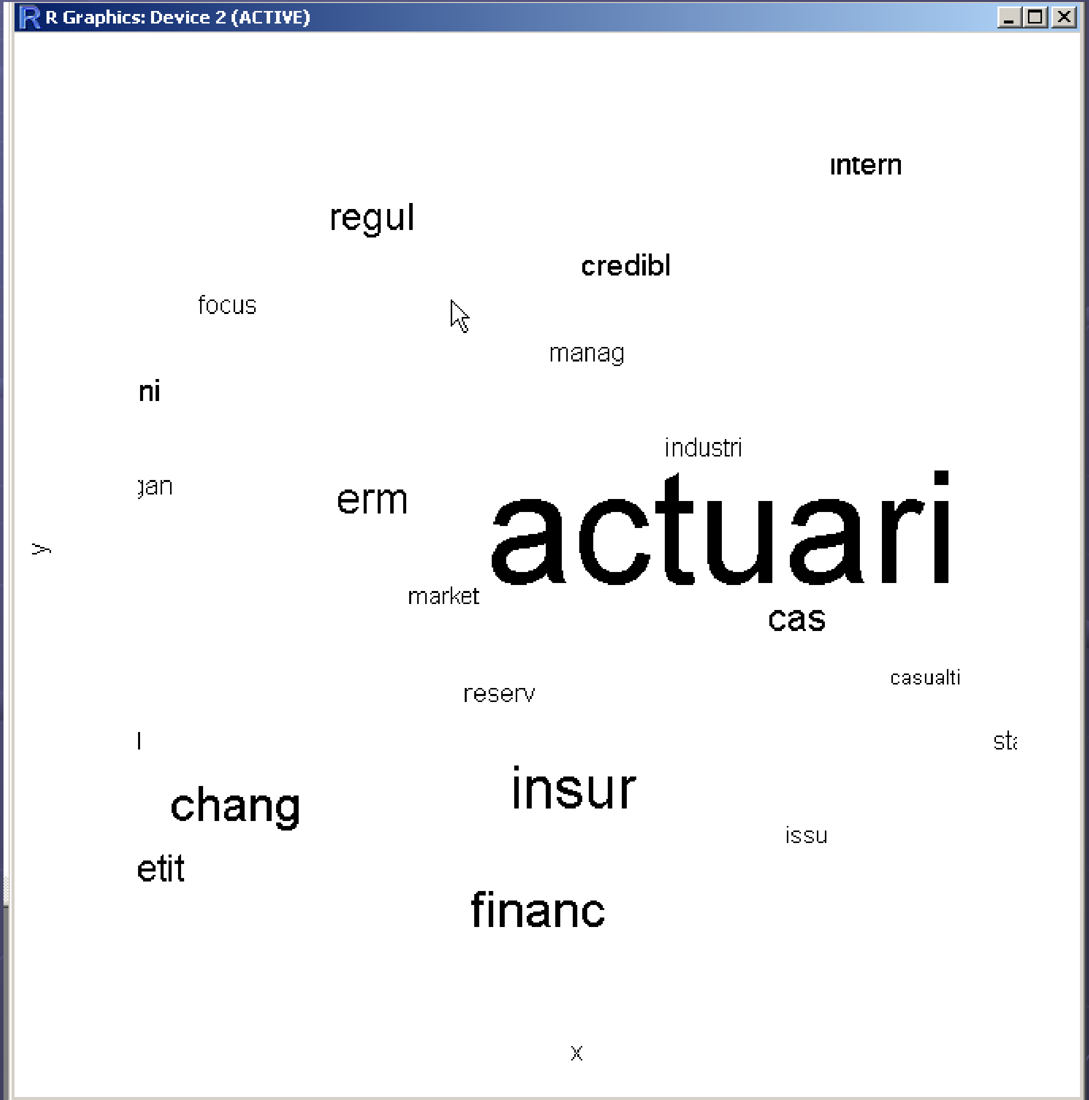
```
Docs alleg ankl anoth area arm around asphalt attempt auger back
  1      0    0     0    0   0      0       0       0     0    0
  2      0    0     0    0   0      0       0       0     0    0
  3      0    0     0    0   0      0       0       0     0    0
  4      0    0     0    0   0      0       0       0     0    0
  5      0    0     0    0   0      0       0       0     0    0
  6      0    0     0    0   0      0       0       0     0    0
  7      0    0     0    0   0      0       0       0     0    0
  8      0    0     0    0   0      0       0       0     0    1
  9      0    0     0    0   0      0       0       0     0    0
  10     0    0     0    0   0      0       0       0     0    0
```

Three Variations on Tag Clouds

```
#********************************************************************
# tag Cloud as a list
2/06/Tag-cloud-for-the-R-Graph-Gallery
#********************************************************************
# as a list
require(snippets)
v <- sapply(top2, sum);v
cloud(v, col=col.bbr(v, fit=TRUE))
```

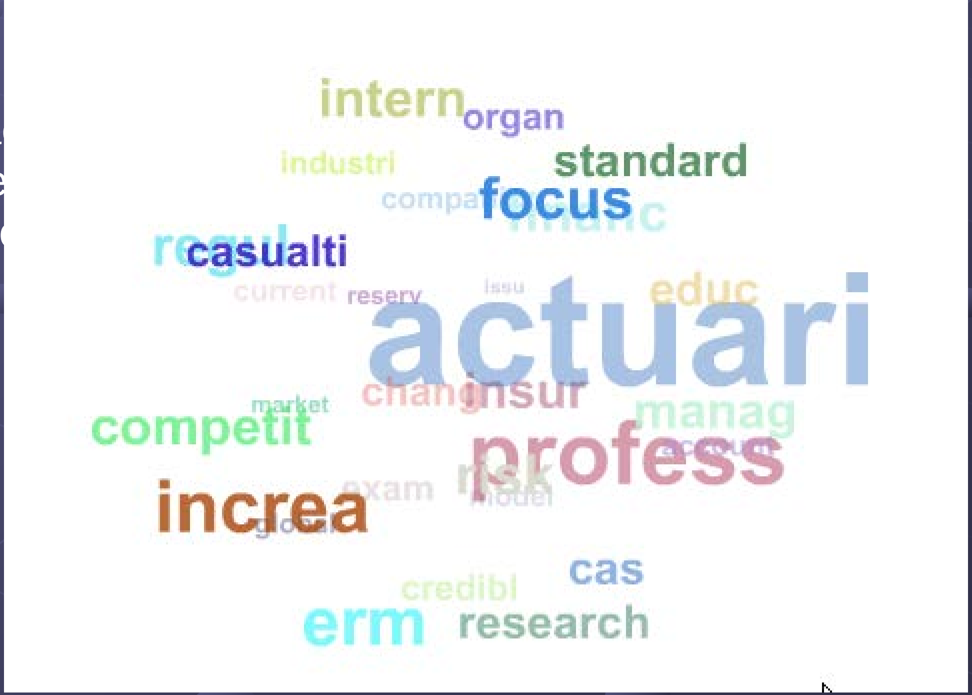
Three Variations on Tag Clouds

```
#******************************************
# Tag Cloud with random placement
#******************************************
set.seed(221);
factor=0.8;
x=runif(length(w))*factor;
y=runif(length(w))*factor;
plot(x, y, type="n", axes=FALSE);
# font size=relative term frequency
pointLabel(x, y, w, 0.10*v);
```



Three Variations on Tag Clouds

```
#*************************************************************
# install.packages("fun", repos="http://r-forge.r-project.org").
# with style
#*************************************************************
require(fun)
data(tagData)
v <- as.matrix(sort(sapply(top2,
doit),decreasing=TRUE)[1:numwords],
colnames=count);v[1:numwords];v
x <-data.frame(rownames(v), "http://www.casa
tagData$color[1:length(v)],tagData$hicolor[1:le
                                        colnames(x) <- c
'count','color','hicolor');x
htmlFile=paste(tempfile(), ".html", sep="")
#htmlFile=paste("tagData", ".html", sep="")
if (file.create(htmlFile)) {
   tagCloud(x, htmlFile)
   browseURL(htmlFile)
}
```

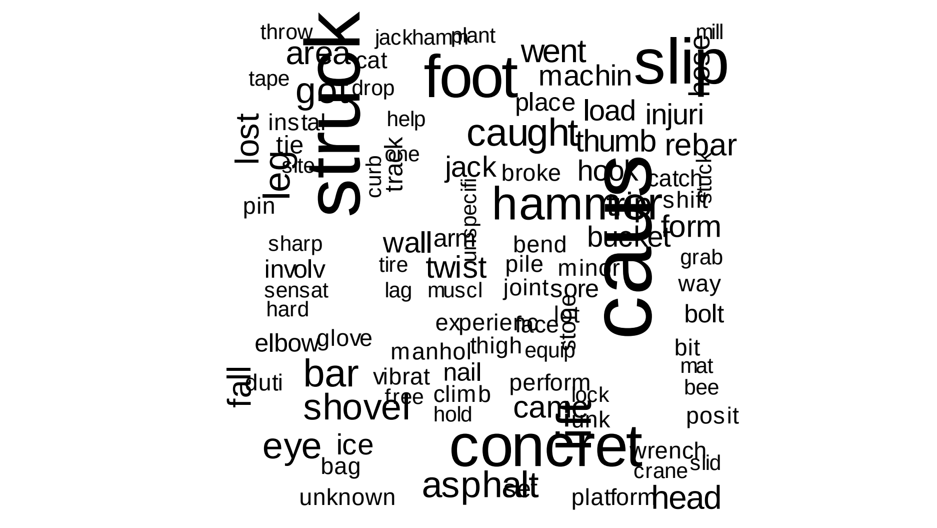


Word Clouds

- An interesting way to graphically present words is through word clouds
- The size of the word is proportional to its frequency of occurrence
- Web sites such as wordle have popularized the method

Word Cloud Example

- # fun with wordclouds
- library(wordcloud)
- # Plot words that occur at least 25 times.
- set.seed(142)
- wordcloud(names(freq), freq, min.freq=5)

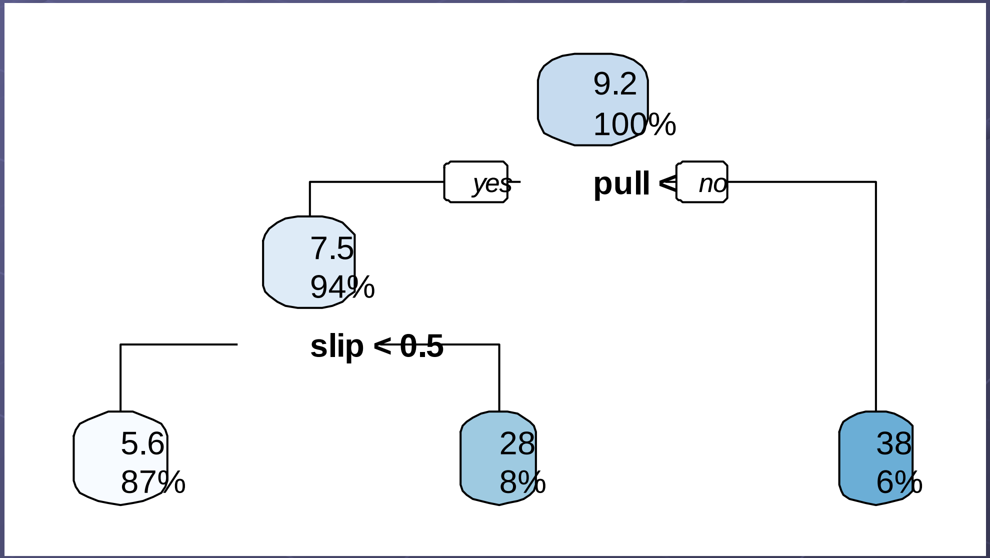


Use Document Term Matrix in Prediction

- Combine DTM with other data
- In this case it is combined with data tha has the claim duration
- Use terms as predictors
- dtm4<-as.matrix(dtm3)
- wcdata<-data.frame(wcdata,dtm4)
- names(wcdata)

Fitted Tree

```
Tree.dtm2<-
rpart(DisabiilityDays~alleg+back+caus+con
cret+fell+felt+finger+foot + hand+ hit
+knee+left+lift+lower+pain+pull+right+slip+
step+struck+truck+use+vehicl+walk+work,
data=wcdata1)
```


9.2
100%
yes
pull <
no
7.5
94%
slip < 0.5
5.6
87%
28
8%
38
6%