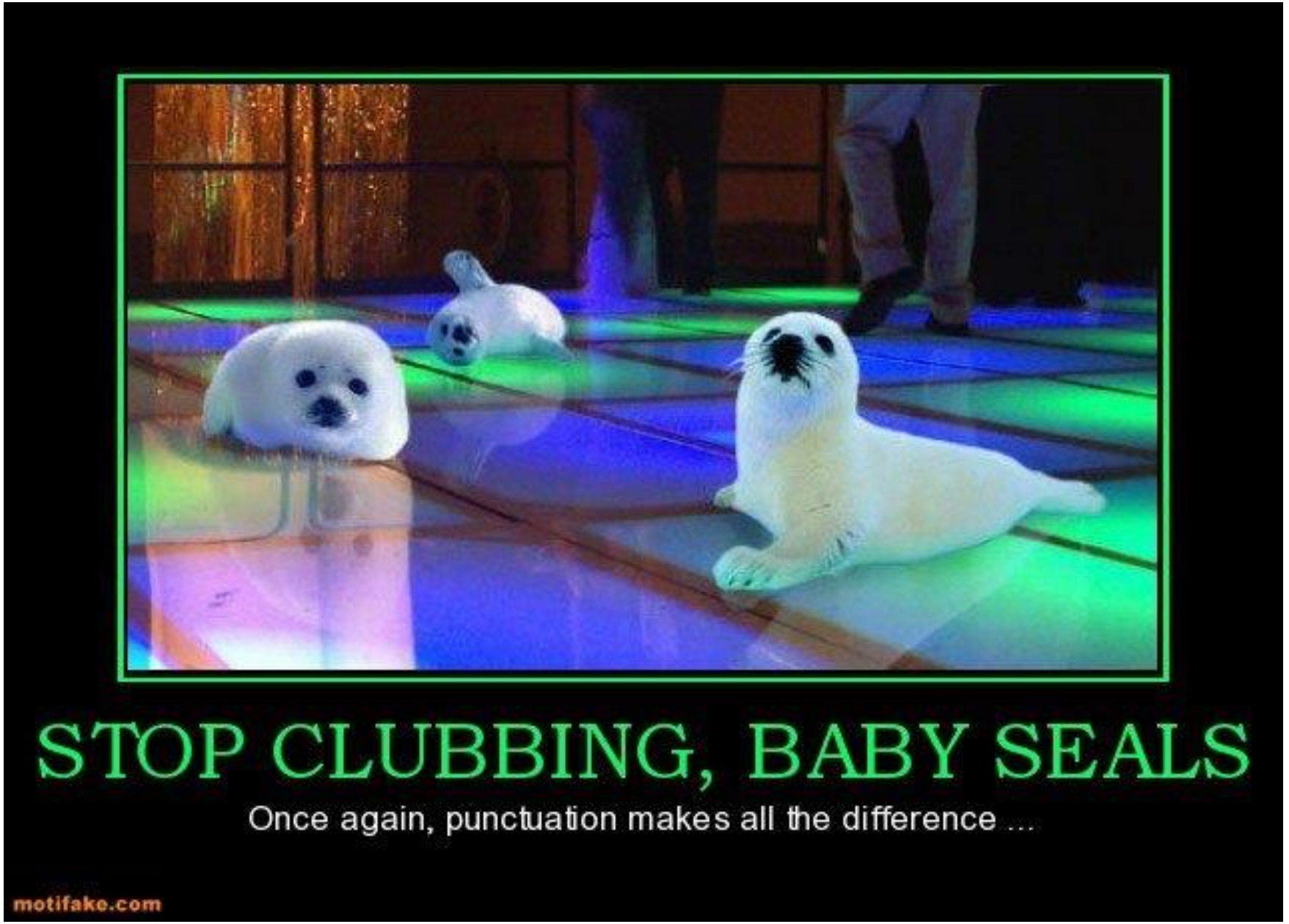
String Processing

Working with Text in R

Jesse Lecy

Text is hard to analyze…

because language is fickle.


STOP CLUBBING, BABY SEALS
Once again, punctuation makes all the difference ...
motifake.com


ATTENTION
Toilet
ONLY
for
DISABLED
ELDERLY
PREGNANT
CHILDREN
THANK YOU for shopping with us!!!
JOIN OUR GRAMMAR PARTY AT FB.COM/GRAMMARBASE


HUNTERS
PLEASE USE
CAUTION
WHEN HUNTING
PEDESTRIANS
USING
WALK TRAILS

strings

STRINGS

functions

Function	Use
grep()	Find a word or phrase (returns the proper string).
grepl()	Find a word or phrase (returns a logical vector).
regexpr()	Find a part of a word or phrase - very flexible.
agrep()	Find an approximate match.
sub()	Replace the first occurence of a word or phrase.
gsub()	Replace ALL occurences of a word or phrase.
———	———————————————
paste()	Combine multiple strings into a single string.
strsplit()	Split one string into multiple strings.
substr()	Extract part of a string.

**GREP: Globally search for a Regular Expression and Print*

REGULAR EXPRESSIONS

Operator	Use
.	matches any single character (wild card for single character)
*	matches 0 or more characters (wild card for any number of characters)
^	start of a string
$	end of a string
?	match any time a character appears 0 or 1 times
+	match any time a character appears 1 or more times
|	OR statement - match either statement given
[ ]	OR statement - match any of the characters given
[^]	match any characters EXCEPT those given in the list
\	escape character - turns an operator into plain text

```r
strings <- c("^ab", "ab", "abc", "abd", "abe", "ab 12", "ab$")
```

```r
# match anything that starts with ab followed by any character

grep("ab.", strings, value = TRUE)
```

```
## [1] "abc"   "abd"   "abe"   "ab 12" "ab$"
```

```r
# match abc OR abd OR abe

grep("ab[c-e]", strings, value = TRUE)
```

```
## [1] "abc" "abd" "abe"
```

```r
# match anything that is NOT abc

grep("ab[^c]", strings, value = TRUE)
```

```
## [1] "abd"   "abe"   "ab 12" "ab$"
```

Operator	Use
.	matches any single character (wild card for single character)
*	matches 0 or more characters (wild card for any number of characters)
^	start of a string
$	end of a string
?	match any time a character appears 0 or 1 times
+	match any time a character appears 1 or more times
|	OR statement - match either statement given
[ ]	OR statement - match any of the characters given
[^]	match any characters EXCEPT those given in the list
\	escape character - turns an operator into plain text

QUANTIFIERS

Operator	Use
*	matches at least 0 times.
.	matches only one time
+	matches at least 1 times.
?	matches at most 1 times.
{n}	matches exactly n times.
{n,}	matches at least n times.
{n,m}	matches between n and m times.

```r
strings <- c("ht","hot","hoot","hooot")
```

```r
# match at least zero times

grep("h*t", strings, value = TRUE)
```

```
## [1] "ht"    "hot"   "hoot"  "hooot"
```

```r
# match ONLY one time

grep("h.t", strings, value = TRUE)
```

```
## [1] "hot"
```

```r
# match exactly n times

grep("ho{2}t", strings, value = TRUE)
```

```
## [1] "hoot"
```

Operator	Use
*	matches at least 0 times.
.	matches only one time
+	matches at least 1 times.
?	matches at most 1 times.
{n}	matches exactly n times.
{n,}	matches at least n times.
{n,m}	matches between n and m times.

POSITION

Operator	Use
^	matches the start of the STRING.
$	matches the end of the STRING.
\\b	matches the empty string at either edge of a WORD.
\\B	matches the string provided it is NOT at an edge of a word.

```r
strings <- c("abcd", "cdab", "cabd", "c abd")
```

```r
# anywhere in the text
```

```r
grep("ab", strings, value = TRUE)
```

```
## [1] "abcd"  "cdab"  "cabd"  "c abd"
```

```r
# at the beginning of a STRING

grep("^ab", strings, value = TRUE)
```

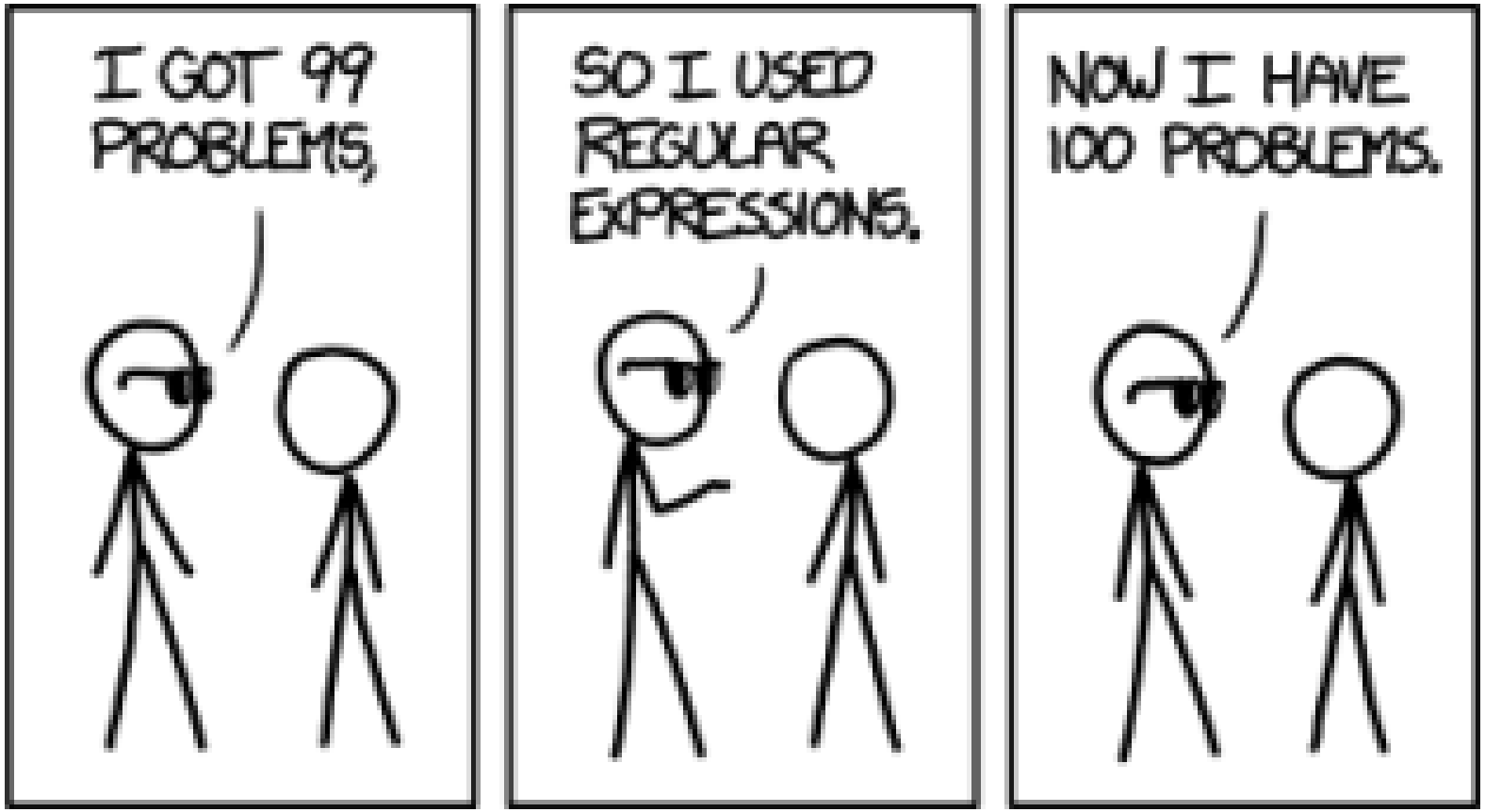
```
## [1] "abcd"
```

```r
# at the end of a STRING

grep("ab$", strings, value = TRUE)
```

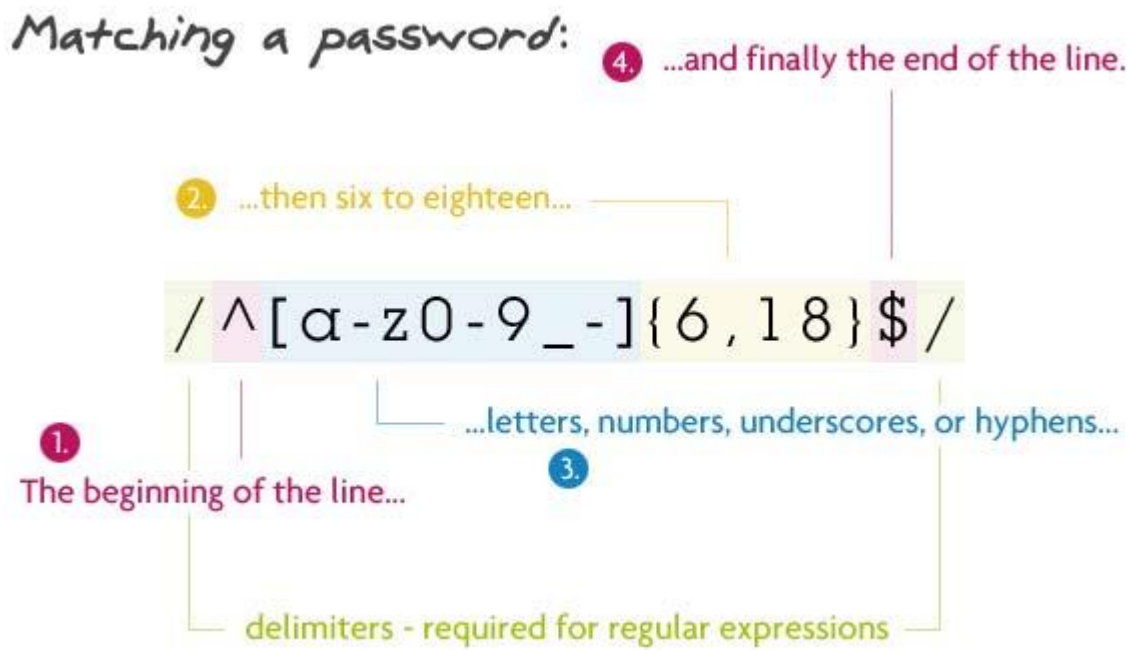
```
## [1] "cdab"
```

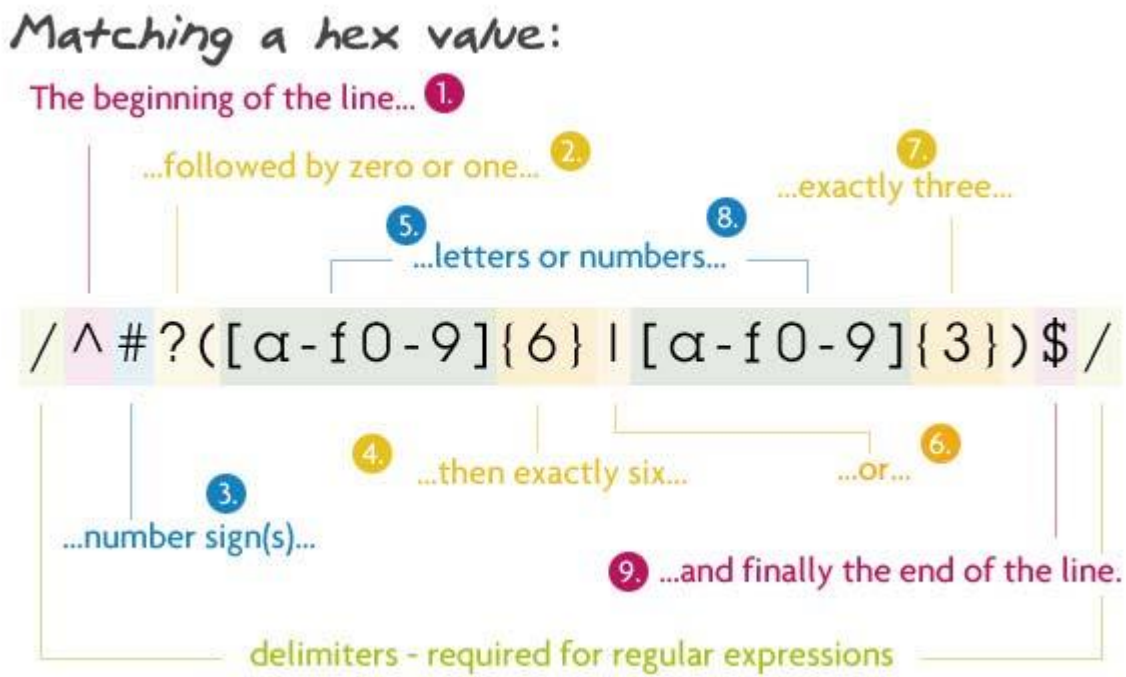
Operator	Use
^	matches the start of the STRING.
$	matches the end of the STRING.
\\b	matches the empty string at either edge of a WORD.
\\B	matches the string provided it is NOT at an edge of a word.


I GOT 99 PROBLEMS,
SO I USED REGULAR EXPRESSIONS.
NOW I HAVE 100 PROBLEMS.

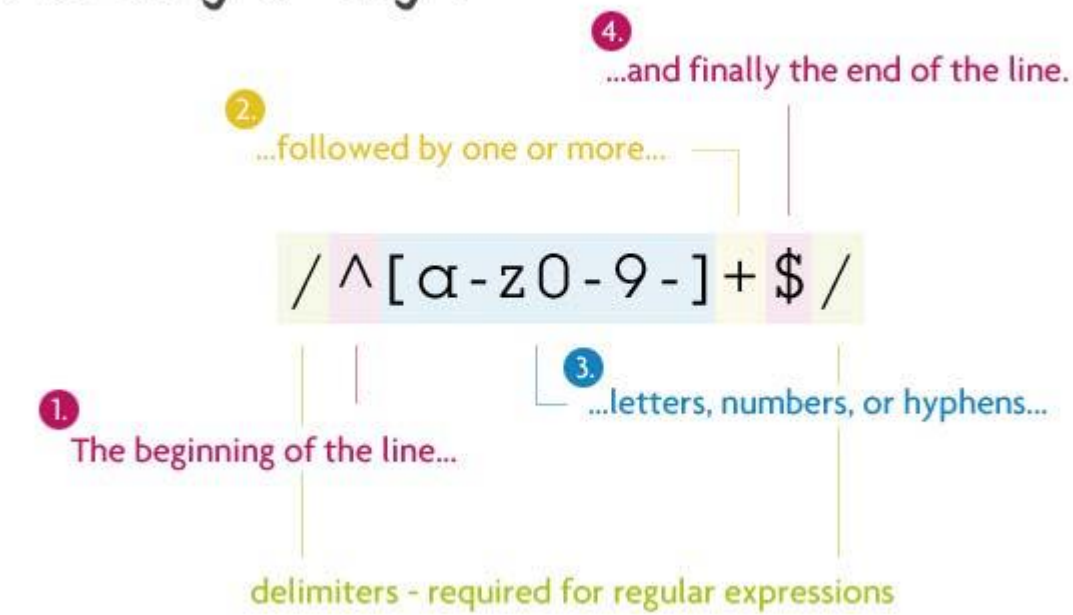
Regex in the wild:
advanced examples for inspiration

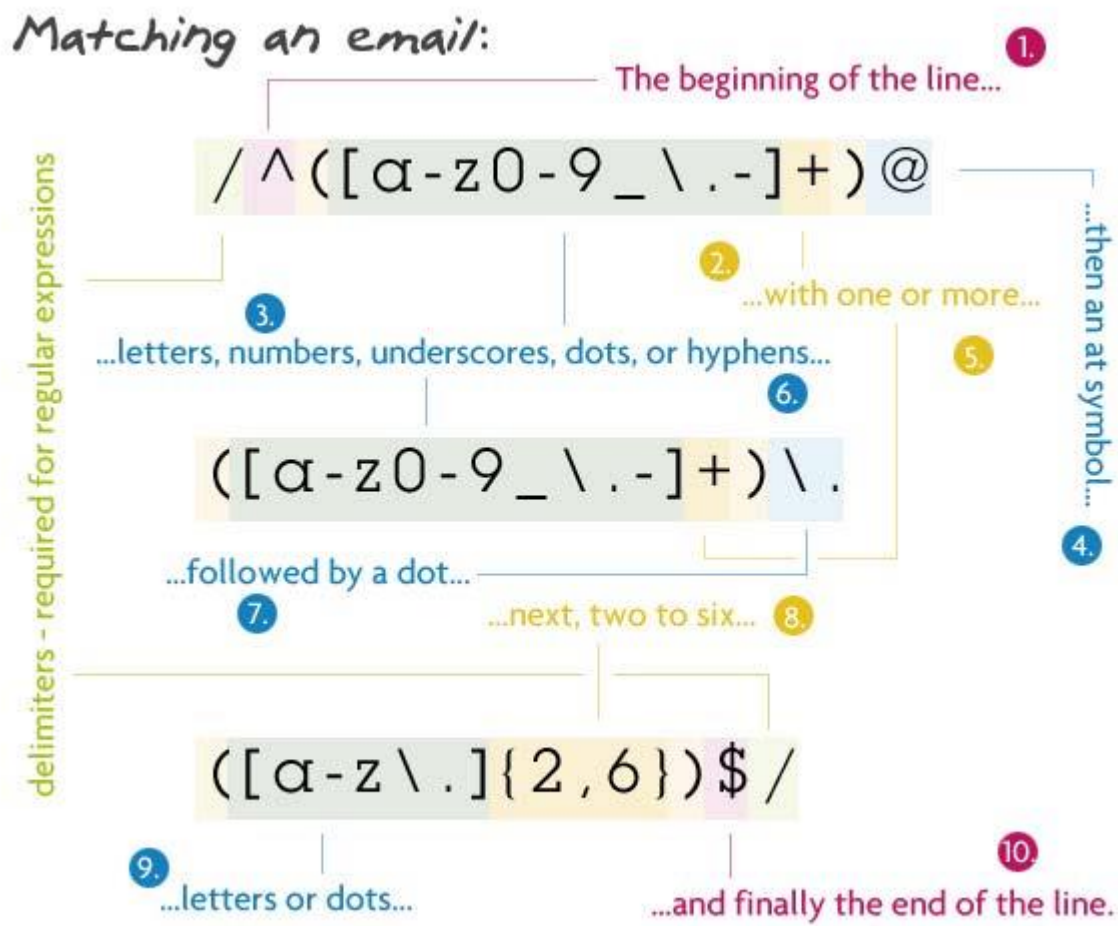
http://code.tutsplus.com/tutorials/8-regular-expressions-you-should-know--net-6149

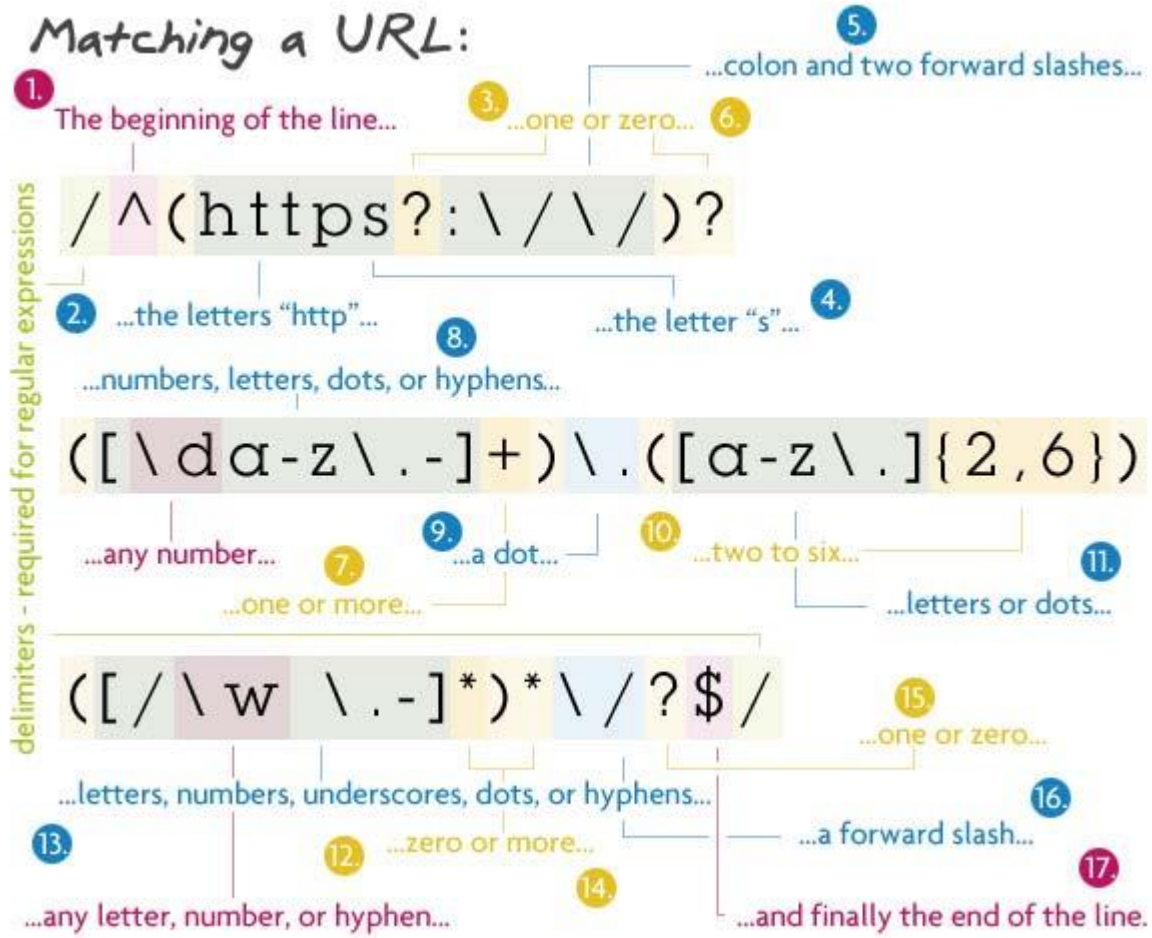

Matching a password:
4. ...and finally the end of the line.
2. ...then six to eighteen...
/^[a-z0-9_-]{6,18}$/
...letters, numbers, underscores, or hyphens...
3.
1.
The beginning of the line...
delimiters - required for regular expressions

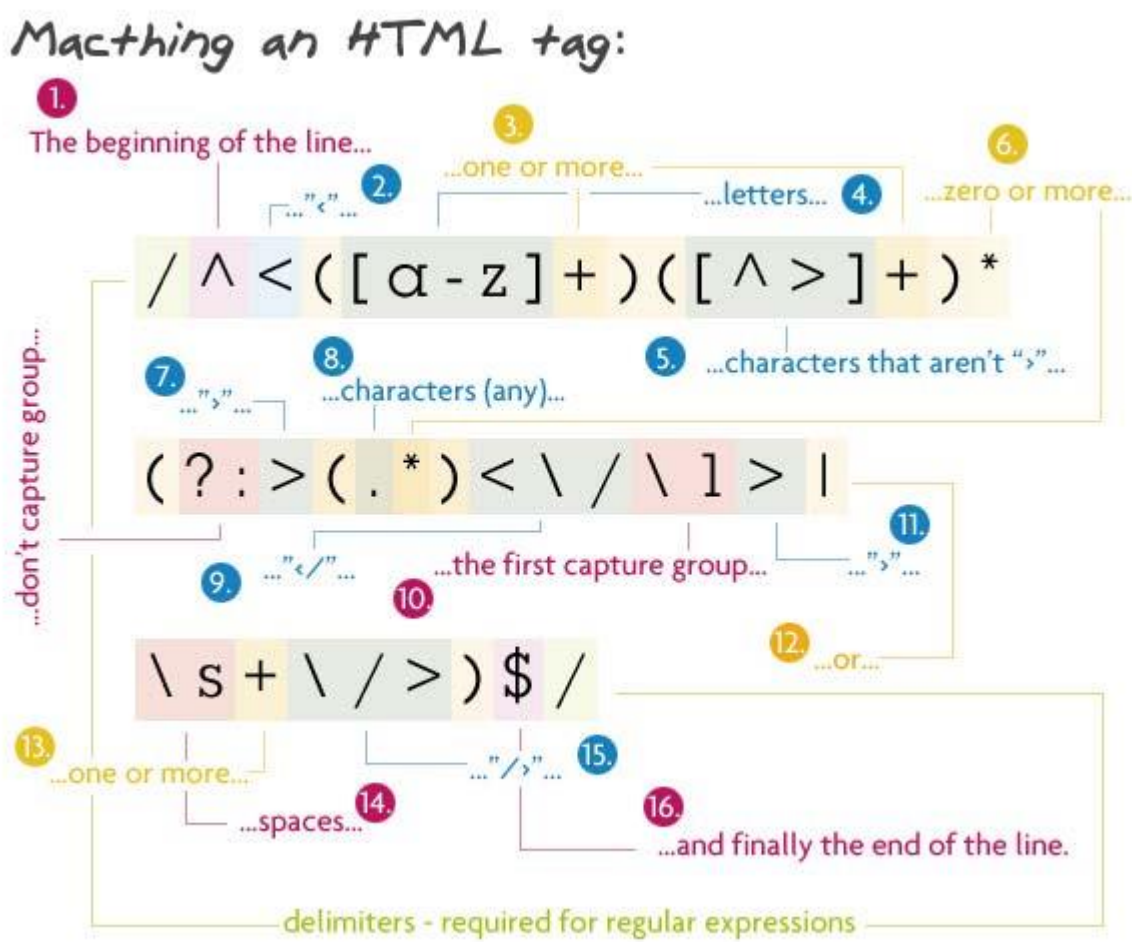

Matching a hex value:
The beginning of the line... 1.
...followed by zero or one... 2.
...number sign(s)... 3.
...then exactly six... 4.
...letters or numbers... 5.
...or... 6.
...exactly three... 7.
8.
9. ...and finally the end of the line.
/^#?([a-f0-9]{6}|[a-f0-9]{3})$/
delimiters - required for regular expressions

Matching a "slug":




Matching an email:
1.
The beginning of the line...
/^([a-z0-9_\.-]+)@
...then an at symbol...
4.
2.
...with one or more...
5.
3.
...letters, numbers, underscores, dots, or hyphens...
6.
([a-z0-9_\.-]+)\.
...followed by a dot...
7.
...next, two to six...
8.
delimiters - required for regular expressions
([a-z\.]{2,6})$/
9.
...letters or dots...
10.
...and finally the end of the line.


Matching a URL:
delimiters - required for regular expressions
/^(https?:\/\/)?
([\da-z\.-]+)\.([a-z\.]{2,6})
([/\w \.-]*)*\/?$/
1. The beginning of the line...
2. ...the letters "http"...
3. ...one or zero...
4. ...the letter "s"...
5. ...colon and two forward slashes...
6.
7. ...one or more...
8. ...numbers, letters, dots, or hyphens...
9. ...a dot...
10. ...two to six...
11. ...letters or dots...
12. ...zero or more...
13. ...any letter, number, or hyphen...
14.
15. ...one or zero...
16. ...a forward slash...
17. ...and finally the end of the line.


Macthing an HTML tag:
1.
The beginning of the line...
2.
..."<"...
3.
...one or more...
4.
...letters...
5.
...characters that aren't ">"...
6.
...zero or more...
/ ^ < ([a - z] +) ([^ >] +) *
7.
..."<"...
8.
...characters (any)...
(? : > (. *) < \ / \ 1 > |
...don't capture group...
9.
..."</"...
10.
...the first capture group...
11.
..."<"...
12.
...or...
\ s + \ / >) $ /
13.
...one or more...
14.
...spaces...
15.
..."/>"...
16.
...and finally the end of the line.
delimiters - required for regular expressions

fuzzy matching

`agrep()`

ARE TWO WORDS THE SAME?

GEORGE BUSH.....GEORGE W. BUSH

PLANE.....PLAIN

BUREAUCRACY.....BUREACRACY

INTENTION VS. EXECUTION

Dan Jurafsky

Minimum Edit Distance

- Two strings and their **alignment**:

```
I N T E * N T I O N
| | | | | | | | | |
* E X E C U T I O N
```

Dan Jurafsky

Minimum Edit Distance

```
I N T E * N T I O N
| | | | | | | | | |
* E X E C U T I O N
d s s   i s
```

- If each operation has cost of 1
  - Distance between these is 5
- If substitutions cost 2 (Levenshtein)
  - Distance between them is 8

CALCULATE EDIT DISTANCE

```
> adist( "intention", "execution" )
     [,1]
[1,]    5

> adist( "intention", "execution",
         costs=c(insertions=1, deletions=1, substitutions=2) )
     [,1]
[1,]    8
```

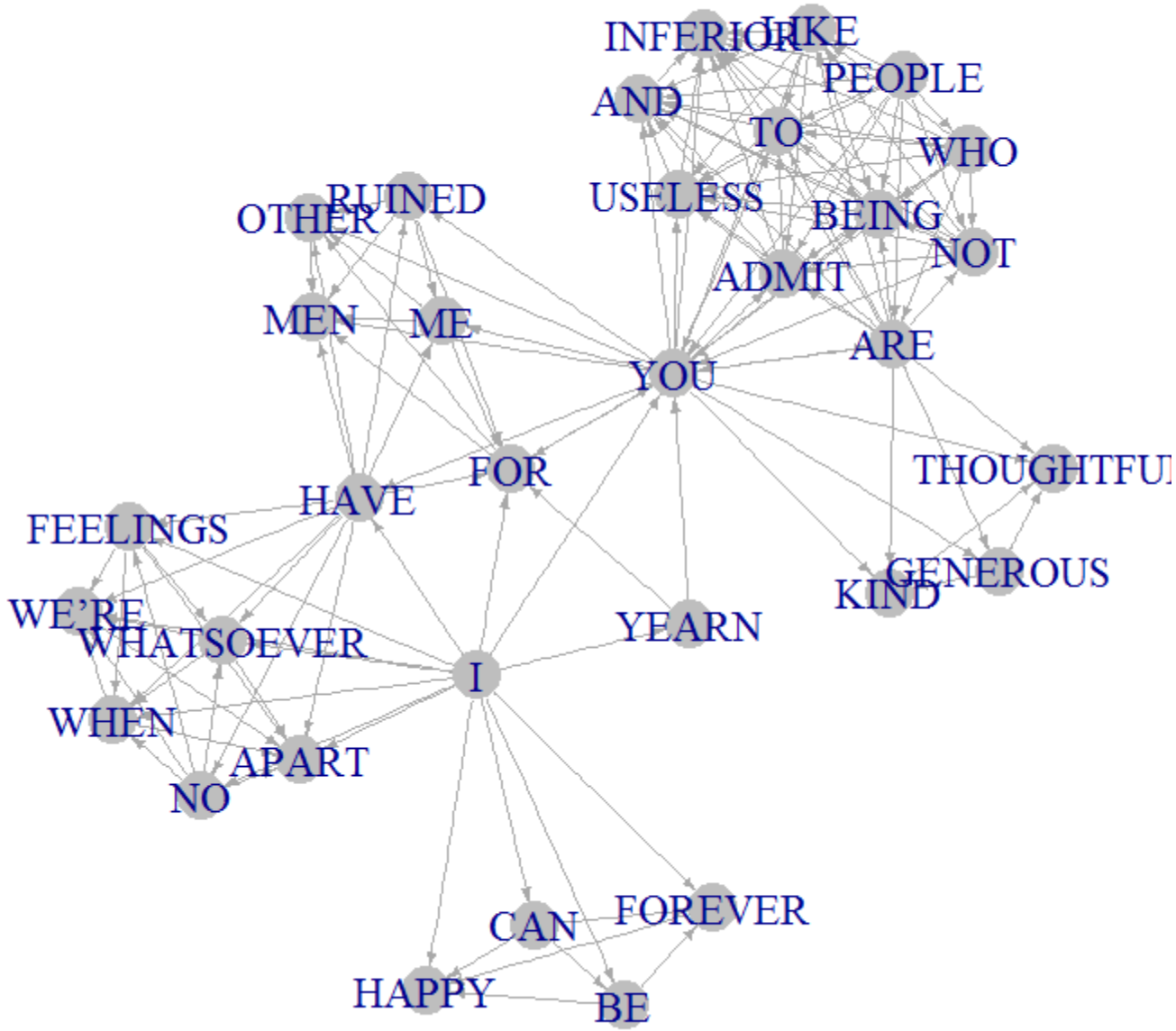
RETURN WORD WITH EDIT DISTANCE < 10%

```
> agrep("lazy", c("daisy", "lasy", "fazed"), value = TRUE )
[1] "lasy"
```

"grep" stands for "**g**lobally search a **r**egular **e**xpression and **p**rint"

"a" stands for approximate matching

practice


INFERIOR
LIKE
PEOPLE
AND
TO
WHO
USELESS
BEING
RUINED
OTHER
NOT
ADMIT
MEN
ME
ARE
YOU
FOR
THOUGHTFU
HAVE
FEELINGS
GENEROUS
KIND
WE'RE
WHATSOEVER
YEARN
I
WHEN
APART
NO
FOREVER
CAN
HAPPY
BE

Dear John:

I want a man who knows what love is all about. You are generous, kind, thoughtful. People who are not like you admit to being useless and inferior. You have ruined me for other men. I yearn for you. I have no feelings whatsoever when we're apart. I can be forever happy.

Will you let me be yours?

Gloria

Dear John:

I want a man who knows what love is. All about you are generous, kind, thoughtful people, who are not like you. Admit to being useless and inferior. You have ruined me. For other men, I yearn. For you, I have no feelings whatsoever. When we're apart, I can be forever happy. Will you let me be?

Yours,

Gloria

INSTRUCTIONS

1. Import text
2. Standardize letter case
3. Remove commas, quote marks, special characters
4. Delete empty lines
5. Split the text into sentences
6. Build a network based upon all words in each sentence

FUNCTIONS

Import Text File:

```
readLines( "filename.txt" )
```

Uppercase / Lowercase

```
toupper( x= ), tolower( x= )          # where x is a character vector
```

Find and Replace All:

```
gsub( pattern=, replacement=, x= )   # pattern = what to replace
                                     # replacement = new text
                                     # x = character vector
```

Split Text by Delimiter:

```
strsplit( x=, split= )               # x = character vector
                                     # split = delimiter where splits occur
```

LOAD TEXT FILE

```
setwd( “…" )

x <- readLines("Dear John 1.txt", warn=FALSE)
```

STANDARDIZE CASE

```
x <- toupper( x )
```

REMOVE SPECIAL CHARACTERS

```
x <- gsub( ",", "", x )
x <- gsub( "we’re", "we are", x )
x <- gsub( "\\:", "", x )
x <- gsub( "\\?", "", x )
```

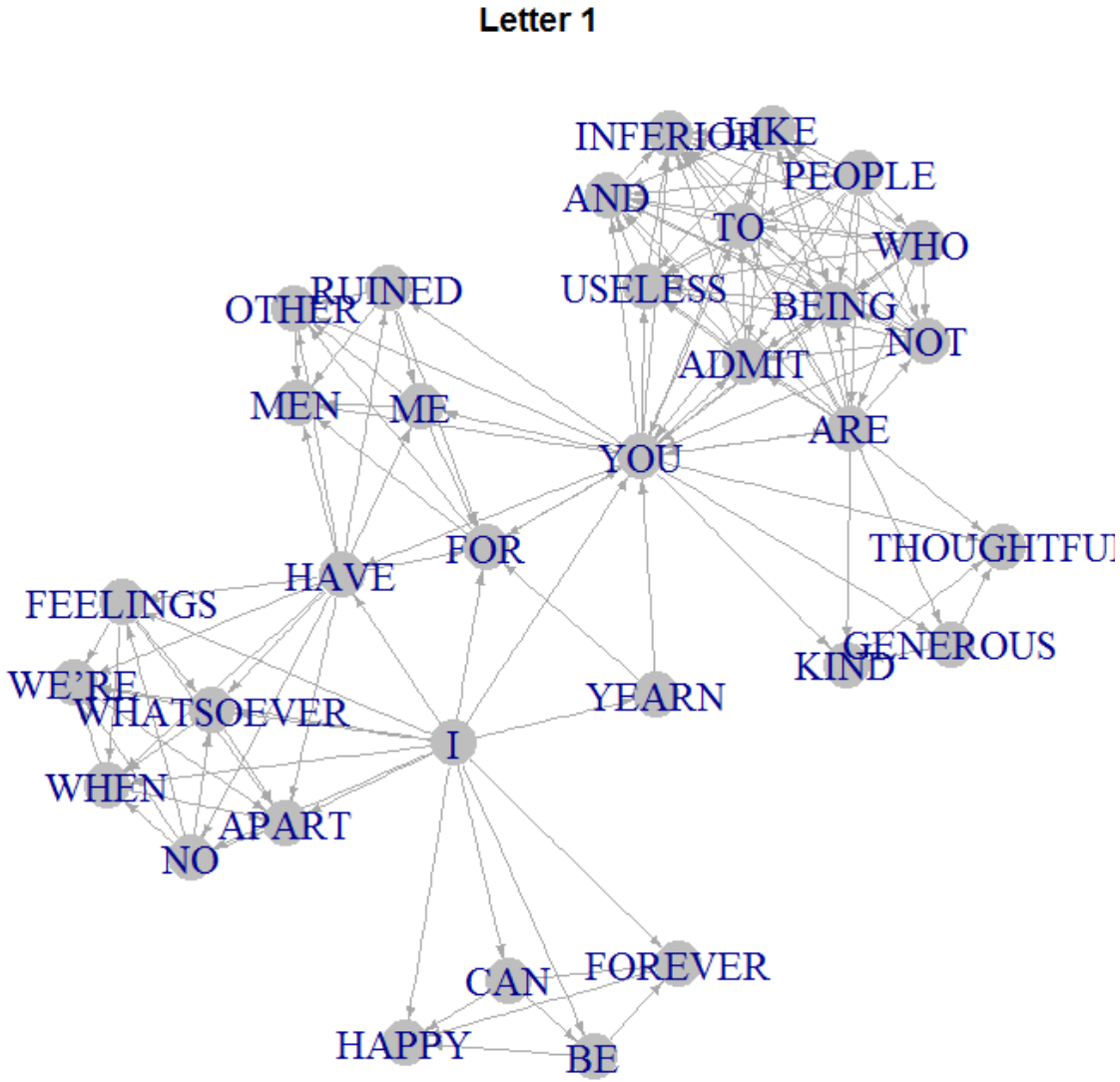
REMOVE BLANK LINES

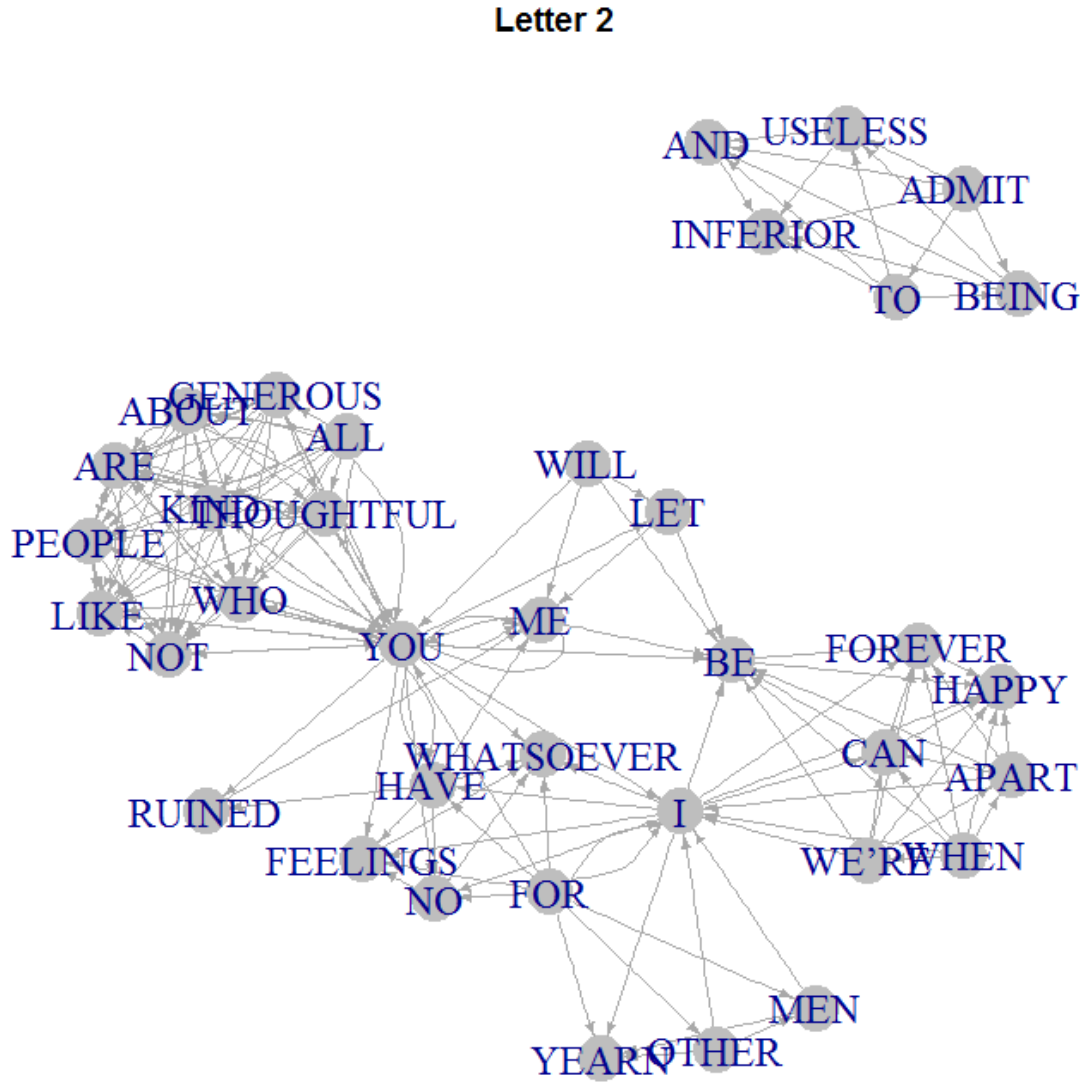
```
grep( "^$", x )

x <- x[ - grep( "^$", x ) ]
```

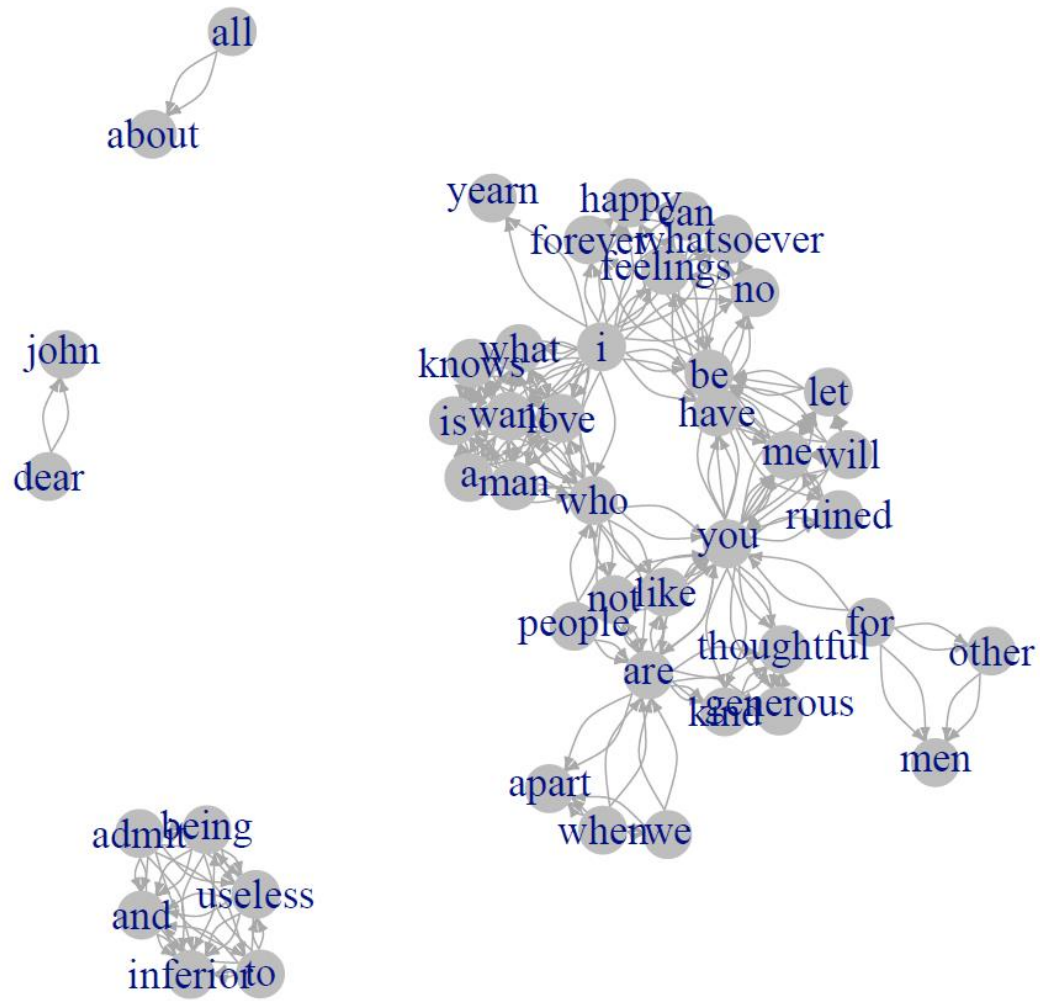
SPLIT INTO SENTENCES

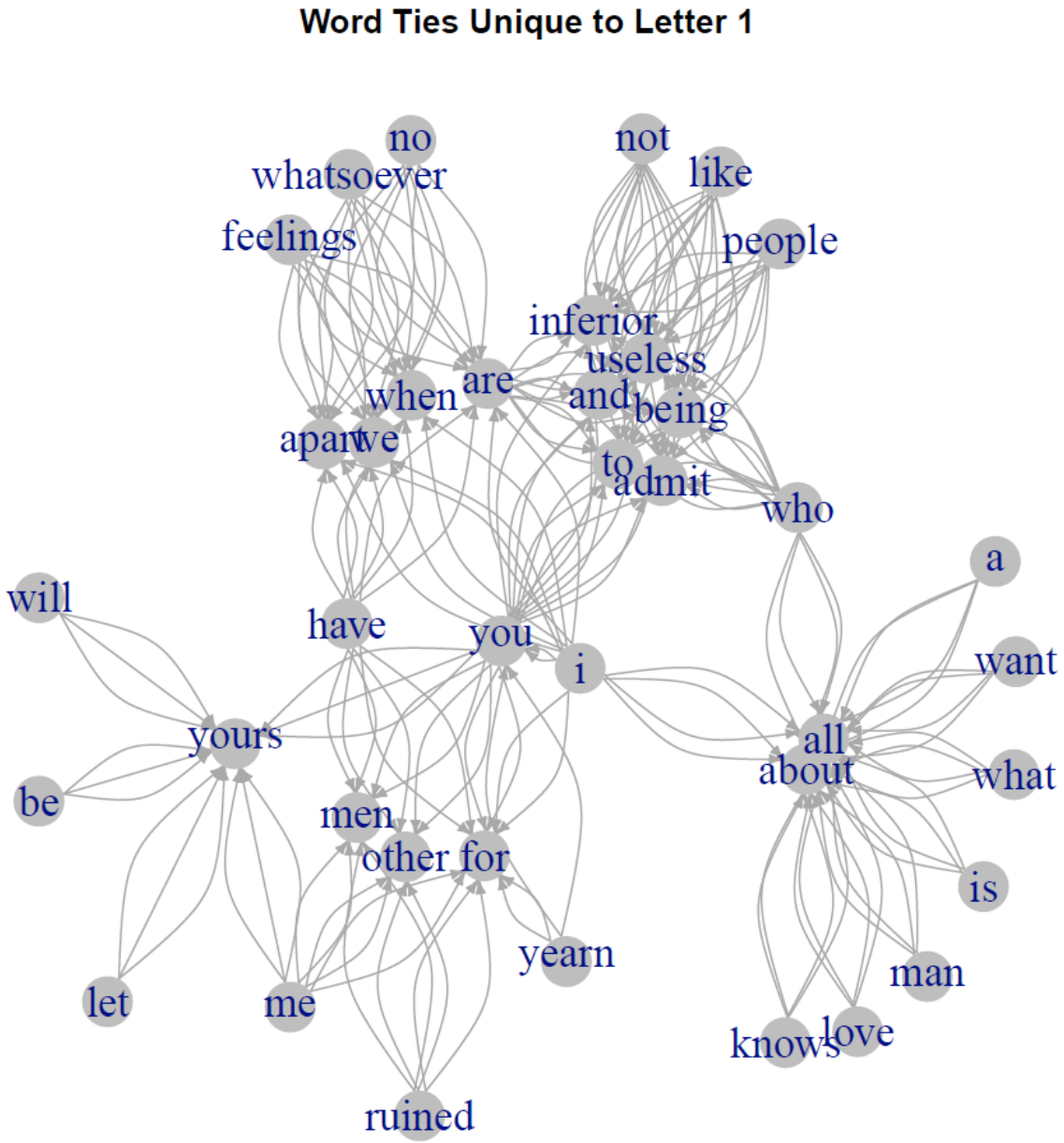
```
strsplit( x, split="\\.")
```

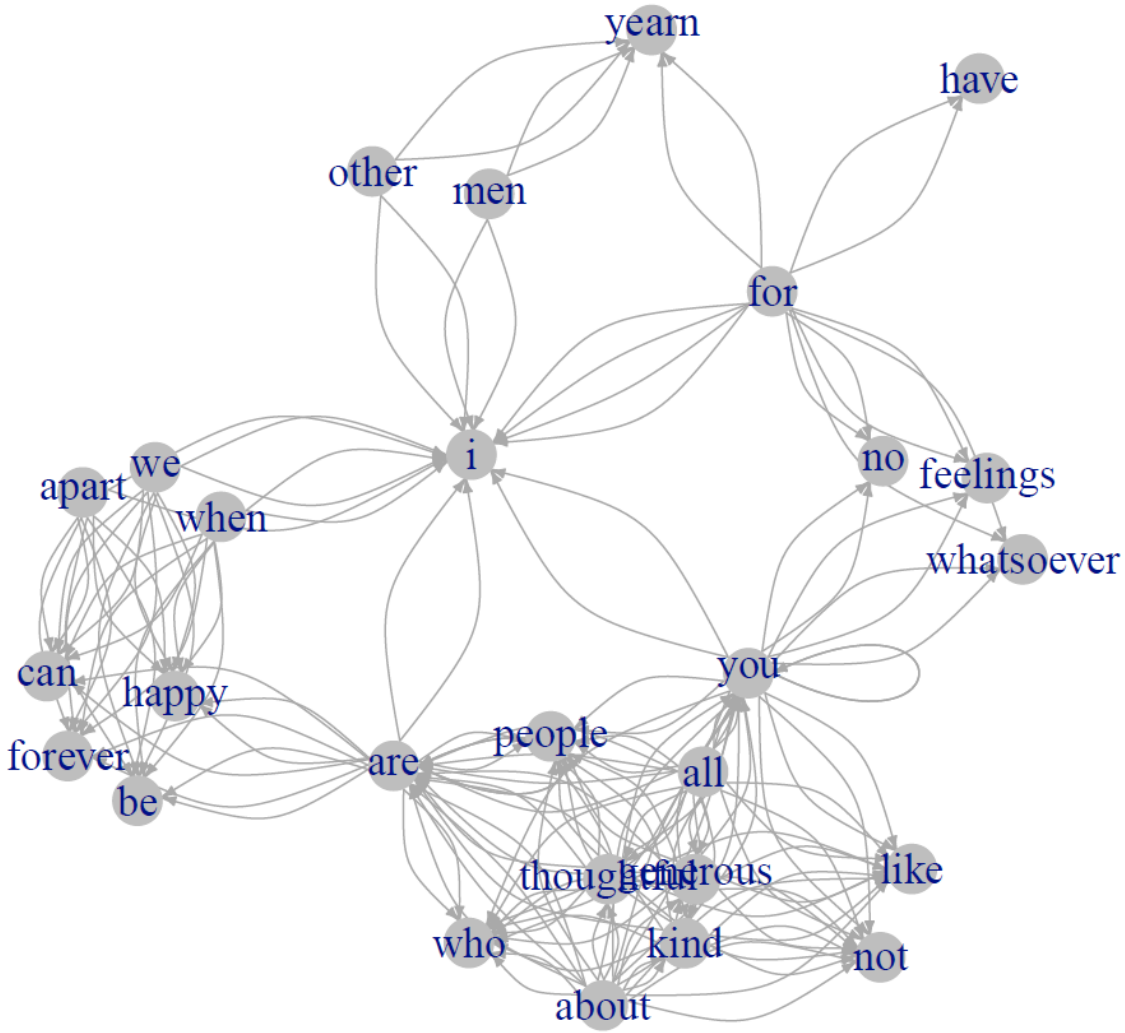

Letter 1
INFERIOR
LIKE
PEOPLE
AND
TO
WHO
USELESS
BEING
RUINED
OTHER
NOT
ADMIT
MEN
ME
ARE
YOU
FOR
THOUGHTFUL
HAVE
FEELINGS
GENEROUS
KIND
WE'RE
WHATSOEVER
YEARN
I
WHEN
APART
NO
CAN
FOREVER
HAPPY
BE


Letter 2
AND
USELESS
ADMIT
INFERIOR
TO
BEING
GENEROUS
ABOUT
ALL
ARE
WILL
KIND
THOUGHTFUL
LET
PEOPLE
LIKE
WHO
ME
NOT
YOU
BE
FOREVER
HAPPY
WHATSOEVER
CAN
APART
HAVE
RUINED
I
FEELINGS
WE'RE
WHEN
NO
FOR
MEN
YEARN
OTHER

Word Ties Shared in Both Letters




Word Ties Unique to Letter 1
no
not
whatsoever
like
feelings
people
inferior
useless
are
when
and
being
apart
we
to
admit
who
a
will
have
you
i
want
yours
all
about
what
be
men
other
for
is
yearn
man
let
me
love
knows
ruined


Word Ties Unique to Letter 2
yearn
have
other
men
for
i
we
no
feelings
apart
when
whatsoever
you
can
happy
people
forever
are
all
be
like
who
kind
not
about