

Text Analysis in R

Lecture 1: Introduction

Helge Liebert

Introduction

- Text data is now commonly used as a modeling input in both research and industry.
- Often involves substantial amounts of preparation.
- Text processing is a common task in IT - many helpful tools available.
- Working with text requires understanding regular expressions.

Basics

Which language to choose?

- *It depends.* Choice is use-case- and taste-specific.
- *Anything* can be done in *any* language. Convenience varies.
- Concepts and toolkits transfer easily most of the time.
- Prior knowledge vs. task suitability.
- Ease of exploratory analysis vs. ease of deployment in production.

Possible options

- Specialized languages: R, Julia, MATLAB/Octave, Stata, Gauss, ...
- General-purpose languages: Python, Perl, Ruby, C, ...
- Choose a high-level, dynamic, interpreted language unless you are sure you require the extra speed of a compiled language.
- Ideally free and open source. Popular is typically better.
- Research ex ante which libraries are mature and best for solving your specific problem.

- R is the major statistical programming language.
- It is free, used in many sciences and in industry. Good documentation.
- New models are frequently published and implemented first in R.
- Having data processing and analysis in the same language is nice.
- Good library support for common tools (e.g. databases, regular expressions).
- Specific tasks for which high-level wrapper functions are not available may be very cumbersome.
- In recent years, R development has been very active and libraries exist for almost anything.

- General-purpose programming language, supports object-oriented programming.
- Reads like english. Explicit and clear. Whitespace matters, no braces. (*“There should be one obvious way to do it”.*)
- Used extensively in industry and sciences. Good documentation.
- Libraries for almost anything.
- Many science-related libraries exist for other languages, but rarely are they as mature.
- Good and growing support for statistical modeling.
- A bit less suited for interactive data work (but more so for deployment in production).

This lecture

- The lab sessions utilize R.
- Any task covered by this lecture can be accomplished using R or Python (augmented by shell programs).
- R, Python, SQL and knowing your way around a terminal are highly valued skills on the job market.
- Rule-of-thumb recommendation:
 - Simple data analysis/small text corpora:
Stick with R. Augment with other tools where required.
 - More involved data processing/larger text corpora:
Go with Python. You can still analyze data in R.
 - ... and whatever program your colleagues are using.

- Operating systems - Windows, MacOS, Linux?
- Scientific programming - Stata, R, Python, Matlab/Octave, Julia, ...?
- General purpose programming languages?
- CLI, servers, shell scripting?
- (Relational) databases - SQL?
- Version control - Git?

A note on text editing

- A script is a set of *plain text* instructions, fed to an interpreter.
- Editing is independent from running code.
- R scripts usually have the suffix `.r`, Python `.py`, Shell `.sh`.
- Proficiency in a text editor makes working with text easier and faster.
- Too many options to list. All are better than Notepad.
- A few options: VS Code, Sublime Text, Atom, Notepad++, ...
- Learning Vim or Emacs requires you to invest some time.

A possible setup

```
emacs@helge-x250 ~
17 \begin{frame}[A note on operating systems]
18 \begin{itemize}
19 \item MacOS or Linux offer built-in access to a Unix shell (Bash).
20 \item Further software is managed via a package management system and
21 distributed via software repositories.
22 \item On Linux, use your package manager to install anything you require.
23 \item On MacOS, familiarize yourself with Homebrew. Install itern2 if you
24 want a function terminal.
25 \end{itemize}
26 \begin{itemize}
27 \item For Windows, many tools are not available or cumbersome to use.
28 Dependency resolution can be a nightmare.
29 \item Windows does not provide proper access to a Unix shell.
30 \item Even reliably installing Python was a chore until recently (now use
31 Anaconda).
32 \end{itemize}
33 \end{frame}
34
35 \begin{frame}[fragile][Command line interpreters and shells]
36 \begin{itemize}
37 \item An interface that lets you interact with your computer.
38 \item A CLI using a programming language that allows
39 you to execute programs and scripts.
40 \item Unix-based operating systems (Linux, MacOS) have Bash pre-installed.
41 \item Windows has cmd (or PowerShell). These are not a viable
42 replacement. Cygwin or WSL may be. Git Bash is incomplete.
43 \end{itemize}
44 \begin{itemize}
45 \item Some examples:
46 \begin{minted}[fontsize=\footnotesize]{bash}
47 cd somedir/subdir # navigate to a folder
48 cd .. # navigate to parent directory
49 cd # return to your home folder
50 ls # list directory contents
51 A # start the # console
52 \end{minted}
53 \item \textit{CTRL + c} aborts a process, \textit{CTRL + d} quits.
54 \end{itemize}
55 \end{frame}
56
57 \begin{frame}[fragile][Examples]
58 \begin{itemize}
59 \item Some examples:
60 \begin{minted}[fontsize=\footnotesize]{bash}
61 via myscript.r # edit your R script with vim
62 R -f myscript.r # execute your R script
63 python myscript.py # execute your python script
64 git add myscript.py # stage file for version control
65 git commit myscript.py # stage file for version control
66 man ssh # display manual pages for the ssh program
67 ssh myusername@13.438.14.673 # secure shell login to your remote server
68 \end{minted}
69 \end{itemize}
70 \begin{itemize}
71 \item Sounds tedious? It is. But it can also be extremely powerful.
72 \end{itemize}
73 \end{frame}
```

Examples

- Some examples:

```
vim myscript.r # edit your R script with vim
R -f myscript.r # execute your R script
python myscript.py # execute your python script
git add myscript.py # stage file for version control
git commit myscript.py # stage file for version control
man ssh # display manual pages for the ssh program
ssh myusername@13.438.14.673 # secure shell login to your remote server
```
- Sounds tedious? It is. But it can also be extremely powerful.
- Convert all your pdf files in a folder to text and search them.

```
for file in *.pdf; do pdftotext "$file"; done
grep -icr "keyword" *.txt
```

A note on text editors

- A script is a set of *plain* text instructions, fed to an interpreter.
- R scripts usually have the suffix .r, Python .py, Shell .sh.
- Much of our work involves working with text files.
- Some text editor is required. A good text editor makes working with text much easier and faster.
- Too many options to list. All are better than Notepad.
- A few suggestions: VS Code, Sublime Text, Atom, Notepad++.
- Learning Vim or Emacs requires you to invest some time.
- Text editors allow you to integrate your work.
- Sometimes IDEs with GUI may be more convenient.
- Features: Efficient text editing, syntax checking, completion, ...

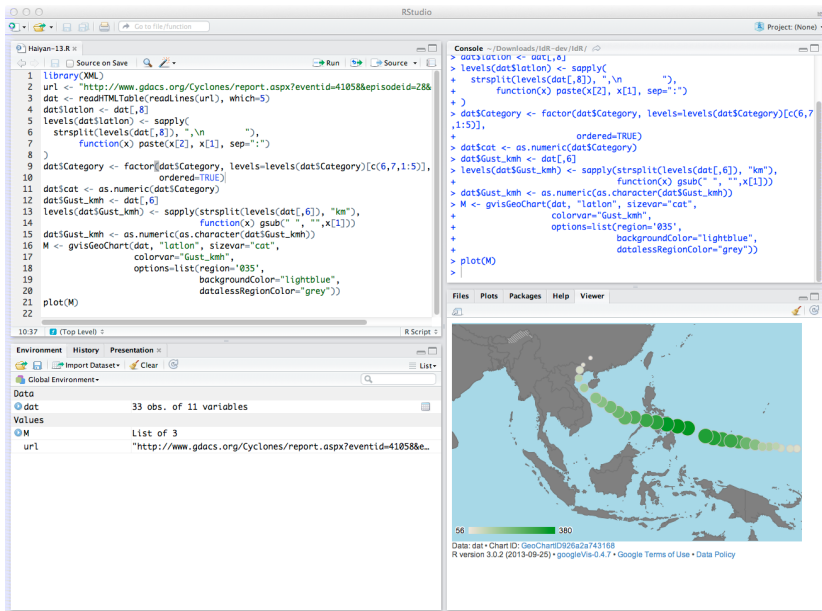
... that is universal

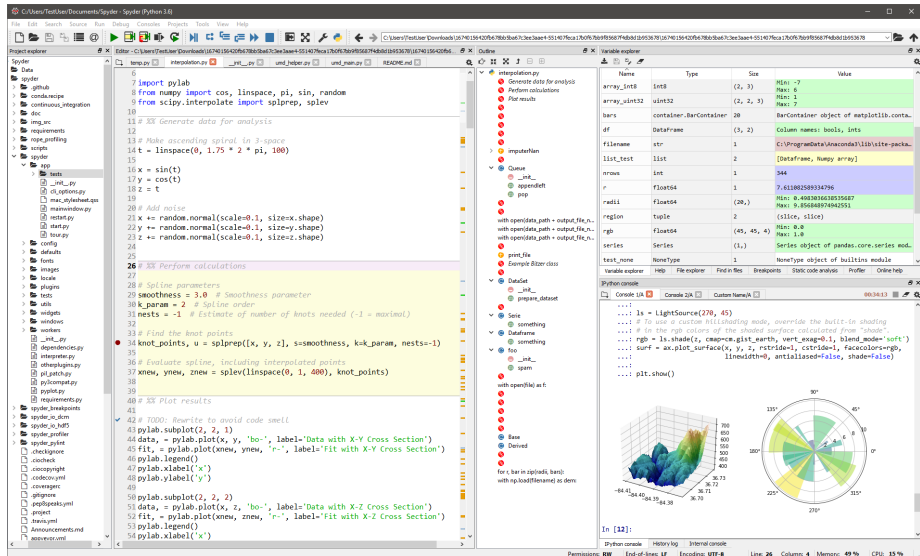
```
emacs@helge-x250 ~
10 ## Get maxpages and other information for iteration
11 response <- fromJSON(url, flatten = TRUE)
12 response$pages
13 maxpages <- response$pages$pages
14 records <- response$pages$total
15 columns <- ncol(response$loans)
16
17 ## Open csv, write header
18 header <- names(response$loans)
19 write.table(t(header), file = "data/kliva.csv", sep = ";",
20            col.names = FALSE, row.names = FALSE)
21
22 ## Or collect in data frame (don't do this for large jobs)
23 data <- data.frame(matrix(nrow = 0, ncol = columns))
24 names(data) <- header
25
26 ## Simple helper function to flatten columns
27 unnest <- function(col) paste(unlist(col), collapse = ", ")
28
29 ## Iterate over pages, limit to first three
30 for (p in seq(1, maxpages, by = 1)[1:3]) {
31   ## Info
32   print(pasted(p, "/", maxpages))
33
34   ## Append page to url
35   pquery <- paste0(url, "&page=", p)
36
37   ## Get data, assert completeness
38   loans <- fromJSON(pquery, flatten = TRUE)$loans
39   stopifnot(nrow(loans) == pagelength)
40   stopifnot(ncol(loans) == columns)
41
42   ## Fix nested list columns ... or just use data.table::fwrite()
43   ## str(loans)
44   loans$tags <- sapply(loans$tags, unnest)
45   ## loans$themes <- sapply(loans$themes, unnest) # missing for older records
46   loans$description_languages <- sapply(loans$description_languages, unnest)
47   ## str(loans)
48
49   ## Collect loans in data frame
50   data <- rbind(data, loans)
51
52   ## Append to file
53   write.table(loans, "data/kliva.csv", sep = ";", append = TRUE,
54             col.names = FALSE, row.names = FALSE)
55 }
56
57 ## head(data)
58 ## dim(data)
59
60 ## They work for you, can Example
61 apikey <- "C3WtBTkKAbdQVqdBKJa"
62 base <- "https://www.theworkforyou.com/api/"
63 format <- "%s"
64
65 ## 5.0 rest-kliva.r ESS[1] [R] pLyf1m4keHwT[0 0]eHmcKIL
66 loading line: apikey <- "C3WtBTkKAbdQVqdBKJa"
67
68 6 Vulnerable Groups
69 location.country_code location.country location.town location.geo_level
70 1 KH Cambodia Kampong Chan town
71 2 VN Vietnam Thanh Hoà town
72 3 VN Vietnam Thanh Hoà town
73 4 VN Vietnam Q1 Hwz Thanh town
74 5 VN Vietnam Thanh Hoà town
75 6 VN Vietnam Thanh Hoà town
76
77 location.geo.pairs location.geo.type
78 1 12 105.5 point
79 2 19.806692 105.785182 point
80 3 19.806692 105.785182 point
81 4 19.436971 105.374762 point
82 5 19.806692 105.785182 point
83 6 19.806692 105.785182 point
84
85 > [1] 20 25
86
87 >>>>>>> query <- paste0("country_code=", country, "&",
88 + "sectors=", sector, "&",
89 + "borrower_type=", type, "&",
90 + "status=", status, "&",
91 + "sort_by=", sortby)
92 url <- paste0(baseurl, method, query)
93 response <- fromJSON(url, flatten = TRUE)
94 response$pages
95 maxpages <- response$pages$pages
96 records <- response$pages$total
97 columns <- ncol(response$loans)
98 > $page
99 [1] 1
100
101 $total
102 [1] 3361
103
104 $page_size
105 [1] 20
106
107 $pages
108 [1] 169
109
110 >>>> header <- names(response$loans)
111 write.table(t(header), file = "data/kliva.csv", sep = ";",
112            col.names = FALSE, row.names = FALSE)
113 unnest <- function(col) paste(unlist(col), collapse = ", ")
114 for (p in seq(1, maxpages, by = 1)[1:3]) {
115   print(pasted(p, "/", maxpages))
116   pquery <- paste0(url, "&page=", p)
117   loans <- fromJSON(pquery, flatten = TRUE)$loans
118   stopifnot(nrow(loans) == pagelength)
119   stopifnot(ncol(loans) == columns)
120   loans$tags <- sapply(loans$tags, unnest)
121   loans$description_languages <- sapply(loans$description_languages, unnest)
122   write.table(loans, "data/kliva.csv", sep = ";", append = TRUE,
123             col.names = FALSE, row.names = FALSE)
124 }
125
126 apikey <- "C3WtBTkKAbdQVqdBKJa"
127 [1] "1/169"
128 [1] "2/169"
129 [1] "3/169"
130 >>>

```

... that is universal

<p>each school-track, classroom formation is conditionally ignorable with respect to SN status, as students from different primary school districts are mixed and their SN status is not observed by secondary school administrators.^{\footnote{Using PISA data from secondary schools in Switzerland, \cite{Vardardottir2015} shows that track-by-school fixed effects render peer group composition conditionally uncorrelated with a large set of students' characteristics, while track fixed effects and school fixed effects do not.} Neither primary schools nor the SPS share information with secondary schools for equity reasons and to avoid stigma when transitioning between schools.}</p> <p>174 % NEU: Bei bitte lesen 175 % changed some small things. OK for me. 176 Beyond this anecdotal evidence, we formally test the validity of the identification strategy with four balancing tests, which are presented and discussed extensively in Appendix B. First, we examine whether the proportion of SN peers predicts individual baseline characteristics (gender, native speaker, and age). The aim of this test is to detect potential selection into classrooms. We also conduct this test separately for SN and non-SN students. None of the baseline characteristics are statistically significant at conventional levels, either considered individually or jointly. Second, we regress the indicator for SN status on class fixed effects, which should be jointly insignificant if assignment to classrooms is ignorable with respect to SN status \cite{chettyEyal2011}. We also conduct this test to check for ignorable assignment of SN students to teachers. We find no evidence for systematic assignment of SN students to either classes or teachers. Third, we conduct a simulation exercise in the spirit of \cite{carrell2010}. We re-sample classes and thereby assign SN students randomly to classrooms, and test whether the observed distribution of SN students differs from the simulated one. In addition, we compute the interquartile range of the proportion SN students across classes for each simulation, and compare these simulated interquartile ranges with the one we observe in the data. Neither simulation procedure uncovers any worrisome pattern in the assignment of SN students to classes. Fourth, we decompose the variation in the fraction of SN peers across and within schools. To do so, we examine the residual variation in the proportion of SN peers after partialling out the school-track-year fixed effects. We find that the residual distribution in the proportion of SN peers is consistent with variation from a random process. Overall, the balancing tests we performed indicate that the key identification assumption of (conditionally) ignorable assignment of SN students to classes is plausible.</p> <p>178 Our identification relies on variation between classes within school-track-years. Although families can potentially choose their district of residence and thereby influence schooling options for their children, possible selection into schools does not confound our results.^{\footnote{Endogenous class formation could still occur if parents request to transfer their children to a class with a lower SN fraction. To investigate this potential threat, we acquired the official education statistics from the Swiss Federal Statistical Office (SOL, \textit{Statistik der Lernenden} in German) for the years 2012-2015. Importantly, the SOL has a classroom ID which allows us to reconstruct the classes within each school-year-track. For the state of St. Gallen we find that no}</p> <p>A: ● 104k xSource/manuscript_R1_v3.tex 77:0 37% LF UTF-8 LaTeX/FPS</p>	<p>different education tracks, and classes are strictly separated between tracks. Within each school-track, classroom formation is conditionally ignorable with respect to SN status, as students from different primary school districts are mixed and their SN status is not observed by secondary school administrators.^{\footnote{Using PISA data from secondary schools in Switzerland, \cite{Vardardottir2015} shows that track-by-school fixed effects render peer group composition conditionally uncorrelated with a large set of students' characteristics, while track fixed effects and school fixed effects do not.} Neither primary schools nor the SPS share information with secondary schools for equity reasons and to avoid stigma when transitioning between schools.}</p> <p>172 % NEU: Bei bitte lesen 173 % Beyond this anecdotal evidence, we formally test the validity of the identification strategy with four balancing tests, which are presented and discussed extensively in Appendix B. First, we examine whether the proportion of SN peers predicts individual baseline characteristics (gender, native speaker, and age). The aim of this test is to detect potential selection into classrooms. We also conduct this test separately for SN and non-SN students. None of the baseline characteristics are statistically significant at conventional levels, either considered individually or jointly. Second, we regress the indicator for SN status on class fixed effects, which should be jointly insignificant if assignment to classroom is ignorable with respect to SN status \cite{chettyEyal2011}. We also conduct this test to check for ignorable assignment of SN students to specific teachers. We find no evidence for systematic assignment of SN students to either classes or teachers. Third, we conduct a simulation exercise in the spirit of \cite{carrell2010}. We re-sample classes, randomly assign SN students to classes, and test whether the observed distribution of SN students differs from the simulated one. In addition, we simulate random classroom assignment within schools, compute the interquartile range of the proportion SN across classes, and compare the simulated interquartile range with the one we observe in the data. Neither simulation procedure uncovers any worrisome pattern in the assignment of SN students to classes. Fourth, we decompose the variation in the fraction of SN peers across and within schools. To do so, we examine the residual variation in the proportion of SN peer after partialling out the school-track-year fixed effects. We find that the residual distribution in the proportion of SN peers is consistent with variation from a random process. Overall, the balancing tests we performed indicate that the key identification assumption of (conditionally) ignorable assignment of SN students to classes is plausible.</p> <p>175 Our identification relies on variation between classes within school-track-years. Although families can potentially choose their district of residence and thereby influence schooling options for their children, possible selection into schools does not confound our results.^{\footnote{Endogenous class formation could still occur if parents request to transfer their children to a class with a lower SN fraction. To investigate this potential threat, we acquired the official education statistics from the Swiss Federal Statistical Office (SOL, \textit{Statistik der Lernenden} in German) for the years 2012-2015. Importantly, the SOL has a classroom ID which allows us to reconstruct the classes within each school-year-track. For the state of St. Gallen we find that no}</p> <p>B: ● 104k x010/manuscript_R1_v2.tex 77:0 37% LF UTF-8 LaTeX/FPS</p> <p>Type ? for help</p> <p>*Edit Control Panel* diff. 4 of 15 Quick Help</p> <p>Auto-refining is ON</p>
--	---





A note on text editing

- Text editors allow you to integrate your work and edit text efficiently.
- Sometimes IDEs with GUI may be more convenient.
- Features: Regex search and replace, diff, syntax checking, formatting, completion, documentation lookup, version control support, ...
- Consider using version control. Git is the predominant version control software used today. [ProGit](#) is a good and free resource.

Regular expressions

Regular expressions

- A regular expression is a character sequence that describes a set of strings.
- Regular expressions are constructed analogously to arithmetic expressions, by using various operators to combine smaller expressions.
- Usually used for find/replace operations on strings, or for validation.
- Pervasive in Unix text processing programs (**grep** was originally written by Ken Thompson).
- **grep**: **g**lobally search a **r**egular **e**xpression and **p**rint.
- Not limited to command line search tools.

Regular expressions

- *Pattern matching*: Find one of a specified set of strings in text.
- Examples:
 - Diagnoses in medical records.
 - Addresses or zip codes in concatenated admin records.
 - Sequences within a genome, e.g. a virus signature.
 - Validate data-entry fields (URL, date, email, credit card #).
 - [Example using a regex tester](#).

Obligatory xkcd

WHENEVER I LEARN A NEW SKILL I CONCOCT ELABORATE FANTASY SCENARIOS WHERE IT LETS ME SAVE THE DAY.

OH NO! THE KILLER MUST HAVE FOLLOWED HER ON VACATION!



BUT TO FIND THEM WE'D HAVE TO SEARCH THROUGH 200 MB OF EMAILS LOOKING FOR SOMETHING FORMATTED LIKE AN ADDRESS!



IT'S HOPELESS!

EVERYBODY STAND BACK.



I KNOW REGULAR EXPRESSIONS.



Examples

AHVN 13: `756\.[0-9]{4}\.[0-9]{4}\.[0-9]{4}\.[0-9]{2}`

Matches: `756.1234.5678.90`

Does not match: `123.45.678.675`

US-SSN: `[0-9]{3}-[0-9]{2}-[0-9]{4}`

Matches: `166-11-4433`

Does not match: `11-55555555`

Email addresses: `[a-z]+@([a-z]+\.)+(ch|edu|com)`

Matches: `someone@unibas.ch`

Does not match: `someone@invalid.domain`

Screening job candidates

“ [First name]! and pre/2 [last name] w/7
bush or gore or republican! or democrat! or charg!
or accus! or criticiz! or blam! or defend! or iran contra
or clinton or spotted owl or florida recount or sex!
or controversies! or fraud! or investigat! or bankrupt!
or layoff! or downsiz! or PNTR or NAFTA or outsourc!
or indict! or enron or kerry or iraq or wmd! or arrest!
or intox! or fired or racis! or intox! or slur!
or controversies! or abortion! or gay! or homosexual!
or gun! or firearm! ”

— *LexisNexis search string used by Monica Goodling
to illegally screen candidates for DOJ positions*



LexisNexis™

<http://www.justice.gov/oig/special/s0807/final.pdf>

Regular expressions

- Characters in a regular expression are either regular characters (literal meaning) or metacharacters (special meaning).
- Generally, letters and numbers match themselves.
- Normally case sensitive, but can be set to ignore case.
- Careful with punctuation, most of it has special meanings.
- To match metacharacters literally, they need to be *escaped*, i.e. preceded by a backslash \.

Matching string literals

Regular expression	Input string
<code>input</code>	This <code>input</code> string is short.
<code>15</code>	The due date is <code>15</code> .12.

Matching string literals

Regular expression	Input string
<code>input</code>	This <code>input</code> string is short.
<code>15</code>	The due date is <code>15.12</code> .
<code>15.12</code>	The due date is <code>15.12</code> .

Matching string literals

Regular expression	Input string
<code>input</code>	This <code>input</code> string is short.
<code>15</code>	The due date is <code>15.12</code> .
<code>15.12</code>	The due date is <code>15.12</code> .
but:	
<code>15.12</code>	The due date is <code>15712</code> .
match <code>.</code> literal:	
<code>15\\.12</code>	The due date is <code>15712</code> .
<code>15\\.12</code>	The due date is <code>15.12</code> .

Regex basics

Operation	Regular expression	Input string
concatenation	<code>foobar</code>	Matches <code>foobar</code> but not <code>foo</code> or <code>bar</code> .
disjunction	<code>this that</code>	Matches <code>this</code> or <code>that</code> .
closure	<code>like.* apples</code>	I <code>like apples</code> , Peter <code>likes apples</code> . Mary also <code>likesALKFHEDL</code> apples.
	<code>like. apples</code>	Mary also <code>likesALKFHEDL</code> apples.
parentheses	<code>(He She) likes</code>	<code>She likes</code> apples. <code>He likes</code> apples.
	<code>(He She).*(very)*much\.</code>	<code>She likes apples very very much.</code>

- More complicated patterns can be expressed via concatenation, disjunction, repetition and scope.
- Precedence in descending order.

Quantifiers

Character	Matches
*	0 or more instances of preceding char
+	1 or more instances of preceding char
?	0 or one instance of preceding char
{m}	exactly m instances of preceding char
{m,n}	m through n instances of preceding char
{m,}	m or more instances of preceding char
{,n}	up to n instances of preceding char
?	add to a quantifier to match ungreedy

- Quantifiers match greedily by default (i.e. the longest string possible).

Ex: `^begin.*end` will match `'begin bla bla end bla end'`.

`^begin.*?end` will match `'begin bla bla end bla end'`.

Groups, ranges and character classes

Character	Matches	Example RE	Matches
.	Any character, except \n	like.	likes like! like like
(a b)	a or b	(you me)	you or me
[ab]	Character range	202[01]	2019 2020 2021 2022
[a-z]	Character range	[A-Z][a-z]*	Capitalized words
[0-9]	Digit range	20[0-9]{2}	Years in the 21st century
[^ab]	Any character but (negation)	20[^0][01]	2000 2010 2020 2025 2031

- Quantifiers, ranges and other shortcuts improve expressiveness.

Ex: `[A-E]+` is shorthand for `(A|B|C|D|E)(A|B|C|D|E)*`.

- More character classes (sometimes) available.

Ex: `\w` for words (`[A-Za-z0-9_]`), or `\d` for digits (`[0-9]`), `\a`, `\s`, ...

Anchors and other special characters

Character	Matches	Example RE	Matches
<code>^</code>	Beginning of a string	<code>^New</code>	New research in this field
<code>\$</code>	End of a string	<code>[A-Za-z]+!\$</code>	A breakthrough! Finally!
<code>\n</code>	Newline		
<code>\t</code>	Tab		
...	...		

- Strings can stretch multiple lines.
- Character encodings can sometimes cause problems. Stick to UTF-8.

Regex syntaxes

- More elaborate regex syntaxes also support positive and negative lookahead/lookbehind, conditionals and group references.
Ex: `^(?!.*word).*` matches lines not containing a word.
- Different syntaxes (basic, extended, perl, vim, ...) mostly similar with regard to basic features.
- Perl-compatible regular expressions (PCRE) is the de-facto standard.
- Most regex implementations feature switches to invert the search pattern, to ignore case, and more.

<http://www.ex-parrot.com/~pdw/Mail-RFC822-Address.html>

[illegible]

Remarks

- Writing a regular expression is like writing a program.
 - Requires understanding the programming model.
 - Can be easier to write than read.
 - Can be difficult to debug.
-
- ➡ Break up problems into smaller pieces. Try not to do everything in one large regex. Comment liberally.
 - ➡ Regular expression tools help (e.g. regex101.com).
 - ➡ Pin a cheatsheet to your office wall.

- Regexes are a powerful tool.
- Easy to grasp, complex to master.
- Using them in applications can be complex and error-prone.
- Regular expressions are not parsers.

“Some people, when confronted with a problem, think ‘I know, I’ll use regular expressions.’ Now they have two problems.”

Emacs newsgroup

Next lecture: Representing text as data

References

References



Chacon, S. and B. Straub (2014). *Pro Git*. Apress.



Fitzgerald, M. (2012). *Introducing Regular Expressions: Unraveling Regular Expressions, Step-by-Step*. 1. ed. Beijing: O'Reilly.



Matloff, N. (2011). *The Art of R Programming: A Tour of Statistical Software Design*. No Starch Press.



Shotts, W. E. (2019). *The Linux Command Line: A Complete Introduction*. Second edition. San Francisco: No Starch Press.