Expressões regulares

Prof. Walmes Zeviani

walmes@ufpr.br

Laboratório de Estatística e Geoinformação
Departamento de Estatística
Universidade Federal do Paraná

Objetivo e justificativa

- Análise de texto sempre requer **processamento de caracteres**.
- Definir o que é expressão regular.
- Listar material e recursos para aprendizado.
- Apresentar os recursos do R.
- Introduzir o `sed` e `awk`.

Expressões regulares

Detalhes

- Regular Expression = *regex* ou *regexp*.
- Sequencia concisa de (meta)caracteres que definem um padrão.
- 1950s - Stephen Cole Kleene (matemático).
- O conceito passou a ser usado nos editores de texto do Unix.
- Usado em:
  - Motores de busca.
  - Editores/processadores de texto.
  - Análise léxica.
  - Para *procurar* ou *procurar e substituir*.
  - Extrair informações chave de documentos de texto (TEL, CEP, CPF, email, datas, horários, cifras monetárias).
- Sintaxes:
  - Padrão POSIX.
  - Padrão Perl.

Exemplo

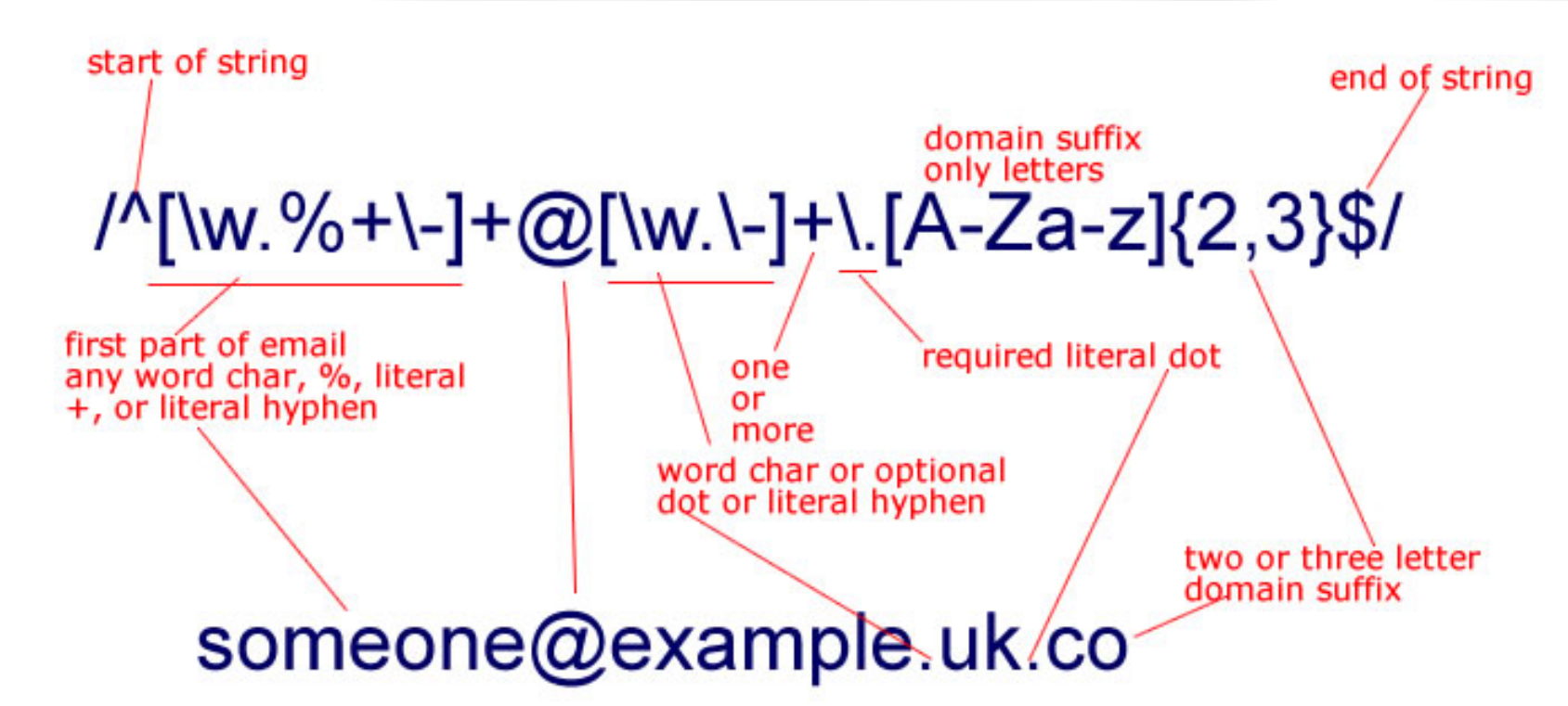

Figura 1. Expressão regular para bater com emails. Fonte: http://learnwebtutorials.com/why-regular-expression-is-so-confusing.

Folhas de cola

As folhas de cola são um guia de consulta rápido para fazer a contrução de expressões regulares.

- http://mesguerra.org/tricks/regexp/regexp.pdf.
- https://www.rstudio.com/wp-content/uploads/2016/09/RegExCheatsheet.pdf.
- http://www.rexegg.com/regex-quickstart.html.

Testadores online

Os testadores *online* servem para avaliar a expressão regular conhecida. Alguns editores possuem recursos para desenvolvimento e aplicação de regex, como o Emacs: https://www.masteringemacs.org/article/re-builder-interactive-regexp-builder.

- http://regexr.com/.
- http://www.regextester.com/.
- https://regex101.com/.
- http://www.freeformatter.com/regex-tester.html.

Praticar

- Data: dd/mm/yyyy.
- CEP: 00.000-000.
- Valor monetário: R$ 000,00.

Regex no R

Recursos básicos

```
help(regex, help_type = "html")
```

1

- `grep()` e `grepl()`: Detecta o padrão.
- `sub()` e `gsub()`: Substitui o padrão.
- `regexpr()` e `gregexpr()`: Localiza o padrão.
- `strsplit()`: Divide no padrão.

1. Dentro de *strings*, deve-se usar duplo contra barra para tornar metacaracteres literais.
2. O comportamento default é POSIX. Faça `perl = TRUE` para usar o Perl.

Exemplo de detecção

```
s <- colors()

# Cores que tenham "red".
grep(pattern = "red", x = s)
```

```
##  [1] 100 372 373 374 375 376 476 503 504 505 506 507 524 525 526 527
## [17] 528 552 553 554 555 556 641 642 643 644 645
```

```
grep(pattern = "red", x = s, value = TRUE)
```

```
##  [1] "darkred"          "indianred"        "indianred1"
##  [4] "indianred2"       "indianred3"       "indianred4"
##  [7] "mediumvioletred"  "orangered"        "orangered1"
## [10] "orangered2"       "orangered3"       "orangered4"
## [13] "palevioletred"    "palevioletred1"   "palevioletred2"
## [16] "palevioletred3"   "palevioletred4"   "red"
## [19] "red1"             "red2"             "red3"
## [22] "red4"             "violetred"        "violetred1"
## [25] "violetred2"       "violetred3"       "violetred4"
```

Exemplo de substituição

```r
# Que tenham "red" seguido de um número.
grep(pattern = "red\\d", x = s, value = TRUE)
```

```
##  [1] "indianred1"     "indianred2"     "indianred3"
##  [4] "indianred4"     "orangered1"     "orangered2"
##  [7] "orangered3"     "orangered4"     "palevioletred1"
## [10] "palevioletred2" "palevioletred3" "palevioletred4"
## [13] "red1"           "red2"           "red3"
## [16] "red4"           "violetred1"     "violetred2"
## [19] "violetred3"     "violetred4"
```

```r
# grep(pattern = "red[0-9]", x = s, value = TRUE)

# Extrair o que está a esqueda de "red".
red <- grep(pattern = "red\\d", x = s, value = TRUE)
sub(pattern = "^(.*)red.*", replacement = "\\1", x = red)
```

```
##  [1] "indian"     "indian"     "indian"     "indian"
##  [5] "orange"     "orange"     "orange"     "orange"
##  [9] "paleviolet" "paleviolet" "paleviolet" "paleviolet"
## [13] ""           ""           ""           ""
## [17] "violet"     "violet"     "violet"     "violet"
```

Pacote `stringr` e `stringi`

- `stringi`: Character String Processing Facilities, https://cran.r-project.org/package=stringi.
- `stringr`: Simple, Consistent Wrappers for Common String Operations, https://cran.r-project.org/package=stringr.
- Consulte as *vignettes* do `stringr` para uma rápida introdução aos recursos.

```
library(stringr)
ls("package:stringr")

library(stringi)
ls("package:stringi")
```

Recursos do Linux

grep

- Buscador de *regex* em arquivos.
- Serve para mostrar quais arquivos contém determinada *regex*.
- https://www.freebsd.org/cgi/man.cgi?query=grep.

```
# Exibe a lista de arquivos que contém ocorrência de "title:".
grep -r 'title:' *

# # Restringe para arquivos de extensão Rmd e numera as linhas.
grep -n -r --include=*.Rmd 'title:' *

# Esconde (hide) o nome do arquivo.
grep -h -r --include=*.Rmd 'title:' *
```

sed: String Editor

- sed: Stream EDitor.
- https://www.freebsd.org/cgi/man.cgi?query=sed&sektion=&n=1.
- https://www.gnu.org/software/sed/manual/sed.html.
- https://www.gnu.org/software/sed/manual/sed.pdf (81 páginas).
- http://www.grymoire.com/Unix/Sed.html.
- Permite processar texto dentro de um diretório.
  - Trocar *Walmes Zeviani* por *Prof. Walmes Zeviani* em todos os arquivos de texto.
- Muito rápido em execução.

```
sed OPCOES... [REGEX] [ENTRADA...]
      |           |       |
      |           |       ` entradas de texto
      |           ` instrução regex
      ` opções que variam o processamento
```

Exemplos de recursos do sed

```
# Mostra só a primeira linha.
sed -n '1p' ../data/toda-forma-de-amor.txt
```

```
## Eu não pedi pra nascer
```

```
# Mostra intervalo de linhas.
sed -n '1,5p' ../data/toda-forma-de-amor.txt
```

```
## Eu não pedi pra nascer
## Eu não nasci pra perder
## Nem vou sobrar de vítima
## Das circunstâncias
```

Exemplos de recursos do sed

```
# Linhas com ocorrência.
sed -n '/a gente/p' ../data/toda-forma-de-amor.txt
```

```
## E a gente vive junto
## E a gente se dá bem
## E a gente vai à luta
## E a gente vive junto
## E a gente se dá bem
## E a gente vai à luta
```

```
# Encontra e substitui.
sed 's/a gente/A GENTE/g' ../data/toda-forma-de-amor.txt | tail
```

```
## E só traz o que quer
## Eu sou teu homem
## Você é minha mulher
##
## E A GENTE vive junto
## E A GENTE se dá bem
## Não desejamos mal a quase ninguém
## E A GENTE vai à luta
## E conhece a dor
## Consideramos justa toda forma de amor
```

Exemplos de recursos do sed

```
# Adiciona virgula no final de cada linha.
cp ../data/toda-forma-de-amor.txt teste.txt
sed -i '$!s/$/,/' teste.txt

# Adiciona um [ na primeira linha.
# sed -i '1s/^/[/' teste.txt
sed -i '1i [' teste.txt

# Remove a vírgula e adiciona um ] na última linha.
# sed -i '$s/,$/]/' teste.txt
echo "]" >> teste.txt

head -n 3 teste.txt
tail -n 3 teste.txt

# sed '1i [' toda-forma-de-amor.txt
# sed '$si ]' toda-forma-de-amor.txt
```

```
## [
## Eu não pedi pra nascer,
## Eu não nasci pra perder,
## E conhece a dor,
## Consideramos justa toda forma de amor
## ]
```

`awk`: pattern-directed scanning and processing language

- `awk`: abreviação dos autores Aho, Weinberger e Kernighan.
- Escrito em 1977 no AT&T Bell Laboratories.
- GNU awk (gwak) é a versão mais popular.
- Paul Rubin escreveu o gawk em 1986.
- `awk` significa o programa e a linguagem (assim como o R).
- `awk` procura e processa linhas de texto em arquivos.
- https://www.freebsd.org/cgi/man.cgi?query=awk.
- https://www.gnu.org/software/gawk/manual/gawk.pdf (540 páginas).
- https://www.math.utah.edu/docs/info/gawk_19.html.

Usei para processar microdados do ENEM.

- http://portal.inep.gov.br/microdados.
- Enem 2015 é um ZIP com 1.1GB.

Livros

- DOUGHERTY; ROBBINS (1997)
  - sed & awk: UNIX Power Tools.
  - https://books.google.com.br/books?id=Xu0G31e-4gIC.
- ROBBINS (2001)
  - Effective Awk Programming: Text Processing and Pattern Matching.
  - https://books.google.com.br/books?id=blLqsSsuoUcC.

Resumo

- Expressões regulares são **indispensáveis** para tratamento de texto.
- O R possui utilidades para *regex* no pacote *base*.
- O pacote *stringr* contém wrappers para processamento de texto.
- No Linux, `sed` e `awk` são úteis para trabalhar lotes de arquivos e arquivos gigantes.
- O cientista de dados, principalmente o que irá trabalhar com texto, deve ser **faixa preta em regex**.

Próxima semana

- Não teremos aula na próxima semana: canaval.
- Não teremos aula na semana 11-15/03: cursos na UFLA.
- Não teremos aula na semana 18-22/03: curso na UFV.
- Nestas 3 semanas serão indicados materiais para leitura no moodle e atividades avaliativas baseado nos materiais indicados.

Referências

DOUGHERTY, D.; ROBBINS, A. **sed & awk: UNIX Power Tools**. O'Reilly Media, 1997.

ROBBINS, A. **Effective Awk Programming: Text Processing and Pattern Matching**. O'Reilly Media, Incorporated, 2001.