String Basics in the tidyverse
Intro to Applied Political Data Science

Ryan T. Moore

9 January 2020

Data Types

- Numeric
- Integer
- Complex
- Logical
- **Character**
- Factor

Strings

String Data

- Candidate names, donor names, employers
- School names, addresses
- Precinct labels

Basic String Tools

```r
library(stringr)
```

Concatenate Strings

```r
library(stringr)
str_c("x", "y")
```

```
## [1] "xy"
```

```r
str_c(c("x", "y"), collapse = ", ")
```

```
## [1] "x, y"
```

Escaping

Escaping

Escaping

\

Escaping

\

If character means something special, must *escape* it to refer to it literally.

Escaping

In R,

to refer to…	You must type …
`"`	`\"`
`'`	`\'`
`\`	`\\`
`<newline>`	`\n`
`<return>`	`\r`
`<tab>`	`\t`

String length

```r
ch <- c("Dem", "Rep", "Indep")
str_length(ch)
```

```
## [1] 3 3 5
```

String length

```r
ch <- c("Hello", "Hi!", "Good day")
str_length(ch)
```

String length

```r
ch <- c("Hello", "Hi!", "Good day")
str_length(ch)
```

```
## [1] 5 3 8
```

Substrings

```r
ch <- c("Dem", "Rep", "Indepen")
str_sub(ch, 2, 5)
```

```
## [1] "em"    "ep"    "ndep"
```

Substrings

```r
ch <- c("Hello", "Hi!", "Good day")
str_sub(ch, 3)
```

Substrings

```r
ch <- c("Hello", "Hi!", "Good day")
str_sub(ch, 3)
```

```
## [1] "llo"     "!"         "od day"
```

String case

```
ch <- c("Dem", "Rep", "Indepen")
str_to_upper(ch)
```

```
## [1] "DEM"     "REP"     "INDEPEN"
```

String case

```r
ch <- c("Hello", "Hi!", "Good day")
str_to_lower(ch)
```

String case

```r
ch <- c("Hello", "Hi!", "Good day")
str_to_lower(ch)
```

```
## [1] "hello"    "hi!"      "good day"
```

Trimming whitespace

```r
ch <- c(" Dem", " Rep ", "Indepen  dent")
str_trim(ch)
```

Trimming whitespace

```r
ch <- c(" Dem", " Rep ", "Indepen  dent")
str_trim(ch)
```

```
## [1] "Dem"            "Rep"            "Indepen  dent"
```

Trimming whitespace

```
ch <- c(" Dem", " Rep ", "Indepen  dent")
str_trim(ch)
```

```
## [1] "Dem"            "Rep"            "Indepen  dent"
```

```
str_squish(ch)
```

```
## [1] "Dem"           "Rep"           "Indepen dent"
```

Trimming whitespace

```r
ch <- "Hello, Hi, and   Good day!  "
str_trim(ch)
```

Trimming whitespace

```
ch <- "Hello, Hi, and   Good day!  "
str_trim(ch)
```

```
## [1] "Hello, Hi, and   Good day!"
```

Trimming whitespace

```r
ch <- "Hello, Hi, and   Good day!  "
str_squish(ch)
```

Trimming whitespace

```r
ch <- "Hello, Hi, and   Good day!  "
str_squish(ch)
```

```
## [1] "Hello, Hi, and Good day!"
```

Sorting strings

```r
ch <- c("Dem", " Rep", "Independent")
str_sort(ch, locale = "en")
```

Sorting strings

```r
ch <- c("Dem", " Rep", "Independent")
str_sort(ch, locale = "en")
```

```
## [1] " Rep"          "Dem"              "Independe
```

Sorting strings

```r
ch <- c("Hello", "Hi!", "Good day")
str_sort(ch)
```

Sorting strings

```r
ch <- c("Hello", "Hi!", "Good day")
str_sort(ch)
```

```
## [1] "Good day" "Hello"    "Hi!"
```

`str_count()` returns number of matches:

```
str_count(c("aab2", "a1b2"), "a")
```

`str_count()` returns number of matches:

```r
str_count(c("aab2", "a1b2"), "a")
```

```
## [1] 2 1
```

`str_subset()` returns only the strings that have a match.

`str_subset()` returns only the strings that have a match.

```r
str_subset(c("aab2", "a1b2"), "a1")
```

```
## [1] "a1b2"
```

```r
str_subset(c("aab2", "a1b2"), ".[0-9].")
str_subset(c("aab2", "a1b2"), ".[0-9].")
```

```
## [1] "a1b2"
```

`str_extract()` returns only the matching parts.

`str_extract()` returns only the matching parts.

```r
str_extract(c("aab2", "a1b2"), "a1")
```

```
## [1] NA   "a1"
```

```r
str_extract(c("aab2", "a1b2"), "a")
str_extract(c("aab2", "a1b2"), "a")
```

```
## [1] "a" "a"
```

```r
str_extract_all(c("aab2", "a1b2"), "a")
str_extract_all(c("aab2", "a1b2"), "a")
```

```
## [[1]]
## [1] "a" "a"
##
## [[2]]
## [1] "a"
```

`str_match()` is like `str_extract()`, but returns components of the match separately.

`str_match()` is like `str_extract()`, but returns components of the match separately.

```r
str(sentences)
```

```
##  chr [1:720] "The birch canoe slid on the smooth pla
```

`str_match()` is like `str_extract()`, but returns components of the match separately.

```
str(sentences)
```

```
##  chr [1:720] "The birch canoe slid on the smooth pla
```

```
a_phr <- str_subset(sentences, "\\b[Aa] ([^ ]+)")
```

`str_match()` is like `str_extract()`, but returns components of the match separately.

```r
str(sentences)
```

```
##  chr [1:720] "The birch canoe slid on the smooth pla
```

```r
a_phr <- str_subset(sentences, "\\b[Aa] ([^ ]+)")
```

```r
str_match(a_phr, "\\b([Aa]) ([^ ]+)")
```

```
##          [,1]          [,2] [,3]
##   [1,] "a well."      "a"  "well."
##   [2,] "a chicken"    "a"  "chicken"
##   [3,] "A rod"        "A"  "rod"
##   [4,] "A pot"        "A"  "pot"
##   [5,] "a hole"       "a"  "hole"
##   [6,] "a button"     "a"  "button"
##   [7,] "A king"       "A"  "king"
##   [8,] "a flop"       "a"  "flop"
##   [9,] "A saw"        "A"  "saw"
```

`str_replace()` replaces first match in string.

`str_replace()` replaces first match in string.

```
str_replace(words[1:10], "s", "*")
```

```
##  [1] "a"         "able"      "about"     "ab*olute" "ac
##  [7] "achieve"  "acro*s"    "act"       "active"
```

```r
str_replace_all(words[1:10], "s", "*")
str_replace_all(words[1:10], "s", "*")
```

```
##  [1] "a"          "able"       "about"      "ab*olute"   "accept
##  [7] "achieve"    "acro**"     "act"        "active"
```

Split strings

`str_split()` returns a list or matrix of components.

Split strings

`str_split()` returns a list or matrix of components.

```r
str_split(words[1:10], "c")
```

```
## [[1]]
## [1] "a"
##
## [[2]]
## [1] "able"
##
## [[3]]
## [1] "about"
##
## [[4]]
## [1] "absolute"
##
## [[5]]
## [1] "a"    ""     "ept"
```

Split strings

```r
str_split(words[1:10], "c", simplify = TRUE)
```

```
##       [,1]       [,2]     [,3]
##  [1,] "a"        ""       ""
##  [2,] "able"     ""       ""
##  [3,] "about"    ""       ""
##  [4,] "absolute" ""       ""
##  [5,] "a"        ""       "ept"
##  [6,] "a"        ""       "ount"
##  [7,] "a"        "hieve"  ""
##  [8,] "a"        "ross"   ""
##  [9,] "a"        "t"      ""
## [10,] "a"        "tive"   ""
```

Exercises §14.4.3.1

2. From the Harvard sentences data, extract:

a) The first word from each sentence.
b) All words ending in `ing`.
c) All plurals.