Strings

Prof. Maria Tackett

Click for PDF of slides

stringr

stringr

In addition to the **tidyverse**, we will use the package **stringr**.

```
library(tidyverse)
library(stringr)
```

stringr provides tools to work with character strings.

- Functions in **stringr** have consistent and memorable names
- All begin with **str_** (**str_count**, **str_detect**, **str_trim**, etc)
- All take a vector of strings as their first argument

Preliminaries

Character strings in R are defined by double quotation marks.

They can include letters, numbers, punctuation, whitespace, etc.

```
string1 <- "STA 199 is my favorite class."
string1
```

```
## [1] "STA 199 is my favorite class."
```

You can combine character strings in a vector.

```
string2 <- c("STA 199", "Data Science", "Duke")
string2
```

```
## [1] "STA 199"      "Data Science" "Duke"
```

Include a quotation in a string?

Why doesn't the code below work?

```
string3 <- "I said "Hello" to my class"
```

Include a quotation in a string?

Why doesn't the code below work?

```
string3 <- "I said "Hello" to my class"
```

To include a double quote in a string, *escape it* using a backslash ****.

Include a quotation in a string?

Why doesn't the code below work?

```
string3 <- "I said "Hello" to my class"
```

To include a double quote in a string, *escape it* using a backslash ****.

```
string4 <- "I said \"Hello\" to my class."
```

Include a quotation in a string?

Why doesn't the code below work?

```
string3 <- "I said "Hello" to my class"
```

To include a double quote in a string, *escape it* using a backslash ****.

```
string4 <- "I said \"Hello\" to my class."
```

What if you want to include an actual backslash?

Include a quotation in a string?

Why doesn't the code below work?

```
string3 <- "I said "Hello" to my class"
```

To include a double quote in a string, *escape it* using a backslash ****.

```
string4 <- "I said \"Hello\" to my class."
```

What if you want to include an actual backslash?

```
string5 <- "\\"
```

This may seem tedious but it will come up later!

writeLines

writeLines shows the contents of the string not including escapes.

```
string4
```

```
## [1] "I said \"Hello\" to my class."
```

```
writeLines(string4)
```

```
## I said "Hello" to my class.
```

```
string5
```

```
## [1] "\\"
```

```
writeLines(string5)
```

```
## \
```

U.S. States

To demonstrate functions from **`stringr`** we will use a vector of all 50 states.

```
states
```

```
##  [1] "alabama"        "alaska"         "arizona"        "arkansas"
##  [5] "california"     "colorado"       "connecticut"    "delaware"
##  [9] "florida"        "georgia"        "hawaii"         "idaho"
## [13] "illinois"       "indiana"        "iowa"           "kansas"
## [17] "kentucky"       "louisiana"      "maine"          "maryland"
## [21] "massachusetts"  "michigan"       "minnesota"      "mississippi"
## [25] "missouri"       "montana"        "nebraska"       "nevada"
## [29] "new hampshire"  "new jersey"     "new mexico"     "new york"
## [33] "north carolina" "north dakota"   "ohio"           "oklahoma"
## [37] "oregon"         "pennsylvania"   "rhode island"   "south carolina"
## [41] "south dakota"   "tennessee"      "texas"          "utah"
## [45] "vermont"        "virginia"       "washington"     "west virginia"
## [49] "wisconsin"      "wyoming"
```

str_length

Given a string, return the number of characters.

```
string1 <- "STA 199 is my favorite class."
str_length(string1)
```

```
## [1] 29
```

Given a vector of strings, return the number of characters in each string.

```
str_length(states)
```

```
##  [1]  7  6  7  8 10  8 11  8  7  7  6  5  8  7  4  6  8  9  5  8 13  8  9 11  8
## [26]  7  8  6 13 10 10  8 14 12  4  8  6 12 12 14 12  9  5  4  7  8 10 13  9  7
```

str_length

Given a string, return the number of characters.

```
string1 <- "STA 199 is my favorite class."
str_length(string1)
```

```
## [1] 29
```

Given a vector of strings, return the number of characters in each string.

```
str_length(states)
```

```
##  [1]  7  6  7  8 10  8 11  8  7  7  6  5  8  7  4  6  8  9  5  8 13  8  9 11  8
## [26]  7  8  6 13 10 10  8 14 12  4  8  6 12 12 14 12  9  5  4  7  8 10 13  9  7
```

- Alabama: 7
- Alaska: 6
- Arizona: 7
- California: 10
- Colorado: 8
- Connecticut: 11

str_c

Combine two or more strings.

```
str_c("STA 199", "is", "my", "favorite", "class")
```

```
## [1] "STA 199ismyfavoriteclass"
```

str_c

Combine two or more strings.

```
str_c("STA 199", "is", "my", "favorite", "class")
```

```
## [1] "STA 199ismyfavoriteclass"
```

Use **sep** to specify how the strings are separated.

```
str_c("STA 199", "is", "my", "favorite", "class", sep = " ")
```

```
## [1] "STA 199 is my favorite class"
```

`str_to_lower` and `str_to_upper`

Convert the case of a string from lower to upper or upper to lower.

```
str_to_upper(states)
```

```
##  [1] "ALABAMA"        "ALASKA"        "ARIZONA"        "ARKANSAS"
##  [5] "CALIFORNIA"     "COLORADO"      "CONNECTICUT"    "DELAWARE"
##  [9] "FLORIDA"        "GEORGIA"       "HAWAII"         "IDAHO"
## [13] "ILLINOIS"       "INDIANA"       "IOWA"           "KANSAS"
## [17] "KENTUCKY"       "LOUISIANA"     "MAINE"          "MARYLAND"
## [21] "MASSACHUSETTS"  "MICHIGAN"      "MINNESOTA"      "MISSISSIPPI"
## [25] "MISSOURI"       "MONTANA"       "NEBRASKA"       "NEVADA"
## [29] "NEW HAMPSHIRE"  "NEW JERSEY"    "NEW MEXICO"     "NEW YORK"
## [33] "NORTH CAROLINA" "NORTH DAKOTA"  "OHIO"           "OKLAHOMA"
## [37] "OREGON"         "PENNSYLVANIA"  "RHODE ISLAND"   "SOUTH CAROLINA"
## [41] "SOUTH DAKOTA"   "TENNESSEE"     "TEXAS"          "UTAH"
## [45] "VERMONT"        "VIRGINIA"      "WASHINGTON"     "WEST VIRGINIA"
## [49] "WISCONSIN"      "WYOMING"
```

str_sub

Extract parts of a string from **start** to **end**, inclusive.

```
str_sub(states, 1, 4)
```

```
##  [1] "alab" "alas" "ariz" "arka" "cali" "colo" "conn" "dela" "flor" "geor"
## [11] "hawa" "idah" "illi" "indi" "iowa" "kans" "kent" "loui" "main" "mary"
## [21] "mass" "mich" "minn" "miss" "miss" "mont" "nebr" "neva" "new " "new "
## [31] "new " "new " "nort" "nort" "ohio" "okla" "oreg" "penn" "rhod" "sout"
## [41] "sout" "tenn" "texa" "utah" "verm" "virg" "wash" "west" "wisc" "wyom"
```

str_sub

Extract parts of a string from **start** to **end**, inclusive.

```
str_sub(states, 1, 4)
```

```
##  [1] "alab" "alas" "ariz" "arka" "cali" "colo" "conn" "dela" "flor" "geor"
## [11] "hawa" "idah" "illi" "indi" "iowa" "kans" "kent" "loui" "main" "mary"
## [21] "mass" "mich" "minn" "miss" "miss" "mont" "nebr" "neva" "new " "new "
## [31] "new " "new " "nort" "nort" "ohio" "okla" "oreg" "penn" "rhod" "sout"
## [41] "sout" "tenn" "texa" "utah" "verm" "virg" "wash" "west" "wisc" "wyom"
```

```
str_sub(states, -4, -1)
```

```
##  [1] "bama" "aska" "zona" "nsas" "rnia" "rado" "icut" "ware" "rida" "rgia"
## [11] "waii" "daho" "nois" "iana" "iowa" "nsas" "ucky" "iana" "aine" "land"
## [21] "etts" "igan" "sota" "ippi" "ouri" "tana" "aska" "vada" "hire" "rsey"
## [31] "xico" "york" "lina" "kota" "ohio" "homa" "egon" "ania" "land" "lina"
## [41] "kota" "ssee" "exas" "utah" "mont" "inia" "gton" "inia" "nsin" "ming"
```

`str_sub` and `str_to_upper`

Can combine **`str_sub`** and **`str_to_upper`** to capitalize each state.

```
str_sub(states, 1, 1) <- str_to_upper(str_sub(states, 1, 1))
states
```

```
##  [1] "Alabama"        "Alaska"       "Arizona"        "Arkansas"
##  [5] "California"     "Colorado"     "Connecticut"    "Delaware"
##  [9] "Florida"        "Georgia"      "Hawaii"         "Idaho"
## [13] "Illinois"       "Indiana"      "Iowa"           "Kansas"
## [17] "Kentucky"       "Louisiana"    "Maine"          "Maryland"
## [21] "Massachusetts"  "Michigan"     "Minnesota"      "Mississippi"
## [25] "Missouri"       "Montana"      "Nebraska"       "Nevada"
## [29] "New hampshire"  "New jersey"   "New mexico"     "New york"
## [33] "North carolina" "North dakota" "Ohio"           "Oklahoma"
## [37] "Oregon"         "Pennsylvania" "Rhode island"   "South carolina"
## [41] "South dakota"   "Tennessee"    "Texas"          "Utah"
## [45] "Vermont"        "Virginia"     "Washington"     "West virginia"
## [49] "Wisconsin"      "Wyoming"
```

str_sort

Sort a string. Here we sort in decreasing alphabetical order.

```
str_sort(states, decreasing = TRUE)
```

```
##  [1] "Wyoming"        "Wisconsin"      "West virginia"  "Washington"
##  [5] "Virginia"       "Vermont"        "Utah"           "Texas"
##  [9] "Tennessee"      "South dakota"   "South carolina" "Rhode island"
## [13] "Pennsylvania"   "Oregon"         "Oklahoma"       "Ohio"
## [17] "North dakota"   "North carolina" "New york"       "New mexico"
## [21] "New jersey"     "New hampshire"  "Nevada"         "Nebraska"
## [25] "Montana"        "Missouri"       "Mississippi"    "Minnesota"
## [29] "Michigan"       "Massachusetts"  "Maryland"       "Maine"
## [33] "Louisiana"      "Kentucky"       "Kansas"         "Iowa"
## [37] "Indiana"        "Illinois"       "Idaho"          "Hawaii"
## [41] "Georgia"        "Florida"        "Delaware"       "Connecticut"
## [45] "Colorado"       "California"     "Arkansas"       "Arizona"
## [49] "Alaska"         "Alabama"
```

Regular Expressions

A regular expression is a sequence of characters that allows you to describe string patterns. We use them to search for patterns.

- extract a phone number from text data
- determine if an email address is valid
- determine if a password has the required number of letters, characters, and symbols
- count the number of times "statistics" occurs in a corpus of text
- ...

Regular Expressions

To demonstrate will will use a vector of all of the states bordering North Carolina.

```
nc_states <- c("North Carolina", "South Carolina",
               "Virginia", "Tennessee", "Georgia")
nc_states
```

```
## [1] "North Carolina" "South Carolina" "Virginia"       "Tennessee"
## [5] "Georgia"
```

Basic Match

We can match exactly.

```
str_view_all(nc_states, "in")
```

```
North Carolina
South Carolina
Virginia
Tennessee
Georgia
```

Basic Match

Match any character using `.`

```
str_view_all(nc_states, "i.")
```

```
North Carolina
South Carolina
Virginia
Tennessee
Georgia
```

Escaping

How to match a period `.`?

Escaping

How to match a period `.`?

Escape it using `\.` This is the regular expression.

Escaping

How to match a period `.`?

Escape it using `\.` This is the regular expression.

But we represent regular expressions using strings and `\` is also an escape symbol in strings.

Escaping

How to match a period `.`?

Escape it using `\.` This is the regular expression.

But we represent regular expressions using strings and `\` is also an escape symbol in strings.

Escape again!

To create the regular expression `\.`, use the string `"\\."`

Escaping

How to match a period `.`?

Escape it using `\.` This is the regular expression.

But we represent regular expressions using strings and `\` is also an escape symbol in strings.

Escape again!

To create the regular expression `\.`, use the string `"\\."`

```
str_view(c("a.c", "abc", "def"), "a\\.c")
```

```
a.c
```

```
abc
```

```
def
```

Anchors

Match the start of a string using **^**

```
str_view(nc_states, "^G")
```

```
North Carolina
South Carolina
Virginia
Tennessee
Georgia
```

Anchors

Match the end of a string using **$**

```
str_view(nc_states, "a$")
```

```
North Carolina
South Carolina
Virginia
Tennessee
Georgia
```

str_detect

Determine if a character vector matches a pattern.

```
nc_states
```

```
## [1] "North Carolina" "South Carolina" "Virginia"       "Tennessee"
## [5] "Georgia"
```

```
str_detect(nc_states, "a")
```

```
## [1]  TRUE  TRUE  TRUE FALSE  TRUE
```

str_subset

Select elements that match a pattern.

```
str_subset(nc_states, "e$")
```

```
## [1] "Tennessee"
```

str_count

How many matches are there in a string?

```
nc_states
```

```
## [1] "North Carolina" "South Carolina" "Virginia"       "Tennessee"
## [5] "Georgia"
```

```
str_count(nc_states, "a")
```

```
## [1] 2 2 1 0 1
```

str_replace

Replace first match with new strings.

```
str_replace(nc_states, "a", "-")
```

```
## [1] "North C-rolina" "South C-rolina" "Virgini-"       "Tennessee"
## [5] "Georgi-"
```

str_replace_all

Replace all matches with new strings.

```
str_replace_all(nc_states, "a", "-")
```

```
## [1] "North C-rolin-" "South C-rolin-" "Virgini-"       "Tennessee"
## [5] "Georgi-"
```

Many Matches

The regular expressions below match more than one character.

- Match any digit using **`\d`** or **`[[:digit:]]`**
- Match any whitespace using **`\s`** or **`[[:space:]]`**
- Match f, g, or h using **`[fgh]`**
- Match anything but f, g, or h using **`[^fgh]`**
- Match lower-case letters using **`[a-z]`** or **`[[:lower:]]`**
- Match upper-case letters using **`[A-Z]`** or **`[[:upper:]]`**
- Match alphabetic characters using **`[A-z]`** or **`[[:alpha:]]`**

Remember these are regular expressions! To match digits you'll need to *escape* the string, so use **`"\\d"`**, not **`"\d"`**

Additional resources

- **`stringr`** website: https://stringr.tidyverse.org/
- **`stringr`** cheat sheet
- Regular Expressions Cheat Sheet
- Chapter 14: Strings in R for Data Science