Working with strings with `stringr`

Stephanie J. Spielman

Data Science for Biologists, Spring 2020

Working with strings

```r
# https://www.kentuckyderby.com/horses
horse1 <- "Silver Prospector"
horse2 <- "Candy Tycoon"
horse3 <- "Shoplifted"

all_horses <- c(horse1, horse2, horse3)
print(all_horses)
## [1] "Silver Prospector" "Candy Tycoon"      "Shoplifted"
```

Useful `stringr` functions

side = c("both", "left", "right")

- General template: `str_ACTION(input_string, ...)`

- `str_count(input_string, character_to_count)` ==> integer

- `str_detect(input_string, substring_to_detect)` ==> logical
- `str_replace(input_string, search, replace)` ==> string
  - `str_replace_all()` replace ALL occurrences instead of only first
- `str_remove(input_string, substring_to_remove)` ==> string
  - `str_remove_all()` removes ALL occurrences instead of only first
- `str_starts(input_string, substring)` ==> logical
- `str_ends(input_string, substring)` ==> logical
- `str_to_upper(input_string)` ==> string (totally uppercase)
- `str_to_lower(input_string)` ==> string (totally lowercase)
- `str_to_title(input_string)` ==> string (totally cap'd after space)
- `str_trim(input_string, side)` ==> trimmed string
  - side is one of `c("both", "left", "right")`
- `str_squish(input_string)` ==> trimmed string

Counting instances of substrings `str_count()`

```
horse1
## [1] "Silver Prospector"
all_horses
## [1] "Silver Prospector" "Candy Tycoon"      "Shoplifted"
```

```
str_count(horse1, "c")
## [1] 1
str_count(horse1, "spec")
## [1] 1
str_count(all_horses, "c")
## [1] 1 1 0
str_count(all_horses, "C")
## [1] 0 1 0
```

Detecting instances of substrings
`str_detect()`

```
horse1
## [1] "Silver Prospector"
all_horses
## [1] "Silver Prospector" "Candy Tycoon"      "Shoplifted"
```

```
str_detect(horse1, "c")
## [1] TRUE
str_detect(horse1, "spec")
## [1] TRUE
str_detect(all_horses, "c")
## [1]  TRUE  TRUE FALSE
str_detect(all_horses, "C")
## [1] FALSE  TRUE FALSE

str_detect(horse1, "Sil", negate=T)
## [1] FALSE
!(str_detect(horse1, "Sil"))
## [1] FALSE
```

Replacing instances of substrings

`str_detect()`

```
horse1
## [1] "Silver Prospector"
horse2
## [1] "Candy Tycoon"
all_horses
## [1] "Silver Prospector" "Candy Tycoon"      "Shoplifted"
```

```
str_replace(horse1, "c", "!!!")
## [1] "Silver Prospe!!!tor"
str_replace(all_horses, "c", "!!!")
## [1] "Silver Prospe!!!tor" "Candy Ty!!!oon"      "Shoplifted"

str_replace_all(horse2, "y", "WHY")
## [1] "CandWHY TWHYcoon"
```

Checking beginnings, endings

```
horse3
## [1] "Shoplifted"
all_horses
## [1] "Silver Prospector" "Candy Tycoon"      "Shoplifted"
```

```
str_starts(horse3, "S")
## [1] TRUE
str_starts(horse3, "s")
## [1] FALSE
str_starts(horse3, "Silver")
## [1] FALSE
str_starts(horse3, "Full")
## [1] FALSE

str_ends(horse3, "definitely not how horse3 ends")
## [1] FALSE

str_starts(all_horses, "Partial")
## [1] FALSE FALSE FALSE
```

Changing cases

```
horse3
## [1] "Shoplifted"
all_horses
## [1] "Silver Prospector" "Candy Tycoon"      "Shoplifted"
```

```
str_to_upper(horse3)
## [1] "SHOPLIFTED"
str_to_lower(horse3)
## [1] "shoplifted"

new_horse <- "mischevious alex"
str_to_title(new_horse)
## [1] "Mischevious Alex"

str_to_upper(all_horses)
## [1] "SILVER PROSPECTOR" "CANDY TYCOON"      "SHOPLIFTED"
```

Trimming whitespace

```
newer_horse <- "Tiz the Law\n\n\n"
str_trim(newer_horse)
## [1] "Tiz the Law"
```

```
newest_horse <- "\r\r\rUntitled"
str_trim(newest_horse)
## [1] "Untitled"
```

```
str_trim(newest_horse, side = "right")
## [1] "\r\r\rUntitled"
str_trim(newest_horse, side = "left")
## [1] "Untitled"
```

What is whitespace?

> These can all be used as **regular expressions**

Symbol	Type of whitespace
`\s`	any type of whitespace
`\t`	a tab stroke
`\n`	a new line (enter on UNIX)
`\r`	return carriage (enter on PC)
`" "`	I literally typed the space key (but in quotes so you can see there is a space). There's no special symbol, just space!

How to integrate with data analysis?

- Many ways, but for you guys...
  - `mutate()` new columns based on existing string columns
  - `filter()` rows based on fulfilling certain conditions

```
names(msleep)
##  [1] "name"         "genus"        "vore"         "order"
##  [5] "conservation" "sleep_total"  "sleep_rem"    "sleep_cycle"
##  [9] "awake"        "brainwt"      "bodywt"
msleep %>%
  ## select all columns that are characters (fancy select thing!!!)
  dplyr::select_if(is.character)-> msleep_str

head(msleep_str, 3)
## # A tibble: 3 x 5
##   name            genus      vore  order     conservation
##   <chr>           <chr>      <chr> <chr>     <chr>
## 1 Cheetah         Acinonyx   carni Carnivora lc
## 2 Owl monkey      Aotus      omni  Primates  <NA>
## 3 Mountain beaver Aplodontia herbi Rodentia  nt
```

Examples!

```
## capitalize genius
msleep_str %>%
  dplyr::mutate(genus_upper = str_to_upper(genus)) %>%
  dplyr::select(genus, genus_upper) %>%
  head(2) ## only show top 3 rows
## # A tibble: 2 x 2
##   genus    genus_upper
##   <chr>    <chr>
## 1 Acinonyx ACINONYX
## 2 Aotus    AOTUS
```

```
## title name
msleep_str %>%
  dplyr::mutate(name_titled = str_to_title(name)) %>%
  dplyr::select(name, name_titled) %>%
  head(3)
## # A tibble: 3 x 2
##   name             name_titled
##   <chr>            <chr>
## 1 Cheetah          Cheetah
## 2 Owl monkey       Owl Monkey
## 3 Mountain beaver  Mountain Beaver
## any monkeys?!?!??!!!
msleep_str %>%
  dplyr::filter(str_detect(name, "monkey"))
## # A tibble: 3 x 5
##   name            genus        vore  order    conservation
##   <chr>           <chr>        <chr> <chr>    <chr>
## 1 Owl monkey      Aotus        omni  Primates <NA>
## 2 Patas monkey    Erythrocebus omni  Primates lc
## 3 Squirrel monkey Saimiri      omni  Primates <NA>
```

```
## any rats? trust me this is going somewhere
msleep_str %>%
  dplyr::filter(str_detect(name, "rat"))
## # A tibble: 5 x 5
##   name                       genus      vore  order
conservation
##   <chr>                      <chr>      <chr> <chr>    <chr>
## 1 African giant pouched rat Cricetomys omni  Rodentia <NA>
## 2 Round-tailed muskrat       Neofiber   herbi Rodentia nt
## 3 Laboratory rat             Rattus     herbi Rodentia lc
## 4 Cotton rat                 Sigmodon   herbi Rodentia <NA>
## 5 Mole rat                   Spalax     <NA>  Rodentia <NA>
```

Introducing REGULAR EXPRESSIONS!

- `\\b` means "word boundary"

```
msleep_str %>%
  ## Look for pattern: rat must be its OWN WORD
  dplyr::filter(str_detect(name, "\\brat\\b"))
## # A tibble: 4 x 5
##   name                       genus      vore  order
conservation
##   <chr>                      <chr>      <chr> <chr>     <chr>
## 1 African giant pouched rat Cricetomys omni  Rodentia <NA>
## 2 Laboratory rat             Rattus     herbi Rodentia lc
## 3 Cotton rat                 Sigmodon   herbi Rodentia <NA>
## 4 Mole rat                   Spalax     <NA>  Rodentia <NA>
```

One more..

- `[]` means set of matching characters

```
msleep_str %>%
  ## Now, either case
  dplyr::filter(str_detect(name, "\\b[Rr]at\\b"))
## # A tibble: 4 x 5
##   name                       genus      vore  order
conservation
##   <chr>                      <chr>      <chr> <chr>    <chr>
## 1 African giant pouched rat Cricetomys omni  Rodentia <NA>
## 2 Laboratory rat             Rattus     herbi Rodentia lc
## 3 Cotton rat                 Sigmodon   herbi Rodentia <NA>
## 4 Mole rat                   Spalax     <NA>  Rodentia <NA>
```