String Manipulation with stringr in R

Teodor Chakarov

2022-05-16

Contents

Tutorium in R

Exercise: Categorical data in Tidyverse - Number 7

By: Teodor Chakarov 12141198

Basics

```
# Define line1
line1 <- "The table was a large one, but the three were all crowded together at one corner of it:"

# Define line2
line2 <- '"No room! No room!" they cried out when they saw Alice coming.'

# Define line3
line3 <- '"There\'s plenty of room!" said Alice indignantly, and she sat down in a large arm-chair at o
```

```
# Putting lines in a vector
lines <- c(line1, line2, line3)

# Print lines
lines
```

```
## [1] "The table was a large one, but the three were all crowded together at one corner of it:"
## [2] "\"No room! No room!\" they cried out when they saw Alice coming."
## [3] "\"There's plenty of room!\" said Alice indignantly, and she sat down in a large arm-chair at on
```

```r
# Use writeLines() on lines
writeLines(lines)
```

```
## The table was a large one, but the three were all crowded together at one corner of it:
## "No room! No room!" they cried out when they saw Alice coming.
## "There's plenty of room!" said Alice indignantly, and she sat down in a large arm-chair at one end o
```

```r
# Write lines with a space separator
writeLines(lines, sep = " ")
```

```
## The table was a large one, but the three were all crowded together at one corner of it: "No room! No
```

```r
# Use writeLines() on the string "hello\n\U1F30D"
writeLines("hello\n\U1F30D")
```

```
## hello
## <U+0001F30D>
```

```r
# Should display: To have a \ you need \\
writeLines("To have a \\ you need \\\\")
```

```
## To have a \ you need \\
```

```r
# Should display:
# This is a really
# really really
# long string
writeLines("This is a really \nreally really \nlong string")
```

```
## This is a really
## really really
## long string
```

```r
# Use writeLines() with
# "\u0928\u092e\u0938\u094d\u0924\u0947 \u0926\u0941\u0928\u093f\u092f\u093e"
writeLines("\u0928\u092e\u0938\u094d\u0924\u0947 \u0926\u0941\u0928\u093f\u092f\u093e")
```

```
## <U+0928><U+092E><U+0938><U+094D><U+0924><U+0947> <U+0926><U+0941><U+0928><U+093F><U+092F><U+093E>
```

Truning numbers to strings

```r
# Some vectors of numbers
percent_change <- c(4, -1.91, 3.00, -5.002)
income <- c(72.19, 1030.18, 10291.93, 1189192.18)
p_values <- c(0.12, 0.98, 0.0000191, 0.00000000002)

# Format c(0.0011, 0.011, 1) with digits = 1
format(c(0.0011, 0.011, 1), digits = 1)
```

```
## [1] "0.001" "0.011" "1.000"
```

```
# Format c(1.0011, 2.011, 1) with digits = 1
format(c(1.0011, 2.011, 1), digits = 1)
```

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

```
# Format percent_change to one place after the decimal point
format(percent_change, digits = 2)
```

```
## [1] " 4.0" "-1.9" " 3.0" "-5.0"
```

```
# Format income to whole numbers
format(income, digits = 2)
```

```
## [1] "     72" "   1030" "  10292" "1189192"
```

```
# Format p_values in fixed format
format(p_values, scientific = FALSE)
```

```
## [1] "0.12000000000" "0.98000000000" "0.00001910000" "0.00000000002"
```

```
formatted_income <- format(income, digits = 2)

# Print formatted_income
formatted_income
```

```
## [1] "     72" "   1030" "  10292" "1189192"
```

```
# Call writeLines() on the formatted income
writeLines(formatted_income)
```

```
##      72
##    1030
##   10292
## 1189192
```

```
# Define trimmed_income
trimmed_income <- format(income, digits = 2, trim = TRUE)

# Call writeLines() on the trimmed_income
writeLines(trimmed_income)
```

```
## 72
## 1030
## 10292
## 1189192
```

```
# Define pretty_income
pretty_income <- format(income, digits = 2, big.mark = ",")

# Call writeLines() on the pretty_income
writeLines(pretty_income)
```

```
##        72
##     1,030
##    10,292
## 1,189,192
```

Alternative with C

```
# From the format() exercise
x <- c(0.0011, 0.011, 1)
y <- c(1.0011, 2.011, 1)

# formatC() on x with format = "f", digits = 1
formatC(x, format = "f", digits = 1)
```

```
## [1] "0.0" "0.0" "1.0"
```

```
# formatC() on y with format = "f", digits = 1
formatC(y, format = "f", digits = 1)
```

```
## [1] "1.0" "2.0" "1.0"
```

```
# Format percent_change to one place after the decimal point
formatC(percent_change, format = "f", digits = 1)
```

```
## [1] "4.0"  "-1.9" "3.0"  "-5.0"
```

```
# percent_change with flag = "+"
formatC(percent_change, format = "f", digits = 1, flag = "+")
```

```
## [1] "+4.0" "-1.9" "+3.0" "-5.0"
```

```
# Format p_values using format = "g" and digits = 2
formatC(p_values, format = "g", digits = 2)
```

```
## [1] "0.12"    "0.98"    "1.9e-05" "2e-11"
```

Concatinate strings

```
years <- 2010:2013
pretty_income <- format(income, digits = 2, big.mark = ",", trim = TRUE)
pretty_percent <- formatC(percent_change, format = "f", digits = 1, flag = "+")

# Add $ to pretty_income
paste("$", pretty_income, sep = "")
```

```
## [1] "$72"        "$1,030"     "$10,292"    "$1,189,192"
```

```
# Add % to pretty_percent
paste(pretty_percent, "%", sep = "")
```

```
## [1] "+4.0%" "-1.9%" "+3.0%" "-5.0%"
```

```
# Create vector with elements like 2010: +4.0%`
year_percent <- paste(years, ": ", pretty_percent, "%", sep = "")

# Collapse all years into single string
paste(year_percent, collapse = ", ")
```

```
## [1] "2010: +4.0%, 2011: -1.9%, 2012: +3.0%, 2013: -5.0%"
```

```
# Define the names vector
income_names <- c("Year 0", "Year 1", "Year 2", "Project Lifetime")

# Create pretty_income
pretty_income <- format(income, digits = 2, big.mark = ",")

# Create dollar_income
dollar_income <- paste("$", pretty_income, sep = "")

# Create formatted_names
formatted_names <- format(income_names, justify = "right")

# Create rows
rows <- paste(formatted_names, dollar_income, sep = "   ")

# Write rows
writeLines(rows)
```

```
##           Year 0   $       72
##           Year 1   $    1,030
##           Year 2   $   10,292
## Project Lifetime   $1,189,192
```

```
toppings <- c("anchovies", "artichoke", "bacon", "breakfast bacon", "Canadian bacon",
  "cheese", "chicken", "chili peppers", "feta", "garlic", "green peppers",
  "grilled onions", "ground beef", "ham", "hot sauce", "meatballs",
  "mushrooms", "olives", "onions", "pepperoni", "pineapple", "sausage",
  "spinach", "sun-dried tomato", "tomatoes")
```

```
# Randomly sample 3 toppings
my_toppings <- sample(toppings, size = 3)
my_toppings
```

```
## [1] "ham"       "hot sauce" "onions"
```

```
# Paste "and " to last element: my_toppings_and
my_toppings_and <- paste(c("", "", "and "), my_toppings, sep = "")
my_toppings_and
```

```
## [1] "ham"        "hot sauce"  "and onions"
```

```
# Collapse with comma space: these_toppings
these_toppings <- paste(my_toppings_and, collapse = ", ")
these_toppings
```

```
## [1] "ham, hot sauce, and onions"
```

```
# Add rest of sentence: my_order
my_order <- paste("I want to order a pizza with ", these_toppings, ".", sep = "")
my_order
```

```
## [1] "I want to order a pizza with ham, hot sauce, and onions."
```

```
# Order pizza with writeLines()
writeLines(my_order)
```

```
## I want to order a pizza with ham, hot sauce, and onions.
```

Stringr library

```
library(stringr)
library(babynames)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
my_toppings <- c("cheese", NA, NA)
my_toppings_and <- paste(c("", "", "and "), my_toppings, sep = "")

# Print my_toppings_and
my_toppings_and
```

```
## [1] "cheese" "NA"     "and NA"
```

```
# Use str_c() instead of paste(): my_toppings_str
my_toppings_str <- str_c(c("", "", "and "), my_toppings)

# Print my_toppings_str
my_toppings_str
```

```
## [1] "cheese" NA       NA
```

```
# paste() my_toppings_and with collapse = ", "
paste(my_toppings_and, collapse = ", ")
```

```
## [1] "cheese, NA, and NA"
```

```
# str_c() my_toppings_str with collapse = ", "
str_c(my_toppings_str, collapse = ", ")
```

```
## [1] NA
```

```
# Extracting vectors for boys' and girls' names
babynames <- read.csv("https://raw.githubusercontent.com/hadley/babynames/master/data-raw/babynames_samp
babynames_2014 <- filter(babynames, year == 1961)
boy_names <- filter(babynames_2014, sex == "M")$name
girl_names <- filter(babynames_2014, sex == "F")$name

# Take a look at a few boy_names
head(boy_names)
```

```
## [1] "Tom"
```

```
# Extract first letter from boy_names
boy_first_letter <- str_sub(boy_names, 1, 1)

# Tabulate occurrences of boy_first_letter
table(boy_first_letter)
```

```
## boy_first_letter
## T
## 1
```

```
# Extract the last letter in boy_names, then tabulate
boy_last_letter <- str_sub(boy_names, -1, -1)
table(boy_last_letter)
```

```
## boy_last_letter
## m
## 1
```

```
# Extract the first letter in girl_names, then tabulate
girl_first_letter <- str_sub(girl_names, 1, 1)
table(girl_first_letter)
```

```
## < table of extent 0 >
```

```
# Extract the last letter in girl_names, then tabulate
girl_last_letter <- str_sub(girl_names, -1, -1)
table(girl_last_letter)
```

```
## < table of extent 0 >
```

Problem with importing babinames

```
# Look for pattern "zz" in boy_names
#contains_zz <- str_detect(boy_names, fixed("zz"))

# Examine str() of contains_zz
#str(contains_zz)

# How many names contain "zz"?
#sum(contains_zz)

# Which names contain "zz"?
#boy_names[contains_zz]

# Which rows in boy_df have names that contain "zz"?
#boy_df[contains_zz, ]
```

```
# Find boy_names that contain "zz"
#str_subset(boy_names, fixed("zz"))

# Find girl_names that contain "zz"
#str_subset(girl_names, fixed("zz"))

# Find girl_names that contain "U"
#starts_U <- str_subset(girl_names, fixed("U"))
#starts_U

# Find girl_names that contain "U" and "z"
#str_subset(starts_U, fixed("z"))
```

```
# Count occurrences of "a" in girl_names
#number_as <- str_count(girl_names, fixed("a"))

# Count occurrences of "A" in girl_names
#number_As <- str_count(girl_names, fixed("A"))

# Histograms of number_as and number_As
#hist(number_as)
#hist(number_As)
```

```
# Find total "a" + "A"
#total_as <- number_as + number_As

# girl_names with more than 4 a's
#girl_names[total_as > 4]
```

Parsing strings into variables

```
date_ranges <- c("23.01.2017 - 29.01.2017", "30.01.2017 - 06.02.2017")
split_dates <- str_split(date_ranges, fixed(" - "))
split_dates
```

```
## [[1]]
## [1] "23.01.2017" "29.01.2017"
##
## [[2]]
## [1] "30.01.2017" "06.02.2017"
```

```
split_dates_n <- str_split(date_ranges, fixed(" - "), n = 2, simplify = TRUE)
split_dates_n
```

```
##      [,1]         [,2]
## [1,] "23.01.2017" "29.01.2017"
## [2,] "30.01.2017" "06.02.2017"
```

```
# Split dates with n and simplify specified
split_dates_n <- str_split(date_ranges, fixed(" - "), n = 2, simplify = TRUE)
split_dates_n
```

```
##      [,1]         [,2]
## [1,] "23.01.2017" "29.01.2017"
## [2,] "30.01.2017" "06.02.2017"
```

```
# Subset split_dates_n into start_dates and end_dates
start_dates <- split_dates_n[ , 1]

# Split start_dates into day, month and year pieces
str_split(start_dates, fixed("."), n = 3, simplify = TRUE)
```

```
##      [,1] [,2] [,3]
## [1,] "23" "01" "2017"
## [2,] "30" "01" "2017"
```

```
both_names <- c("Box, George", "Cox, David")

# Split both_names into first_names and last_names
both_names_split <- str_split(both_names, fixed(", "), n = 2, simplify = TRUE)
```

```r
# Get first names
first_names <- both_names_split[, 2]

# Get last names
last_names <- both_names_split[, 1]
```

String statistics

```r
# Split lines into words
words <- str_split(lines, " ")

# Number of words per line
lapply(words, length)
```

```
## [[1]]
## [1] 18
##
## [[2]]
## [1] 12
##
## [[3]]
## [1] 21
```

Replacing and matching strings

```r
# Some IDs
ids <- c("ID#: 192", "ID#: 118", "ID#: 001")

# Replace "ID#: " with ""
id_nums <- str_replace(ids, fixed("ID#: "), "")

# Turn id_nums into numbers
id_ints <- as.numeric(id_nums)
```

```r
# Some (fake) phone numbers
phone_numbers <- c("510-555-0123", "541-555-0167")

# Use str_replace() to replace "-" with " "
str_replace(phone_numbers, fixed("-"), " ")
```

```
## [1] "510 555-0123" "541 555-0167"
```

```r
# Use str_replace_all() to replace "-" with " "
str_replace_all(phone_numbers, fixed("-"), " ")
```

```
## [1] "510 555 0123" "541 555 0167"
```

```r
# Turn phone numbers into the format xxx.xxx.xxxx
str_replace_all(phone_numbers, fixed("-"), ".")
```

```
## [1] "510.555.0123" "541.555.0167"
```

```r
genes <- readRDS("dna.rds")
# Find the number of nucleotides in each sequence
str_length(genes)
```

```
## [1] 441 462 993
```

```r
# Find the number of A's occur in each sequence
str_count(genes, fixed("A"))
```

```
## [1] 118 117 267
```

```r
# Return the sequences that contain "TTTTTT"
str_subset(genes, fixed("TTTTTT"))
```

```
## [1] "TTAAGGAACGATCGTACGCATGATAGGGTTTTGCAGTGATATTAGTGTCTCGGTTGACTGGATCTCATCAATAGTCTGGATTTTGTTGATAAGTAC
```

```r
# Replace all the "A"s in the sequences with a "_"
str_replace_all(genes, fixed("A"), "_")
```

```
## [1] "TT_G_GT___TT__TCC__TCTTTG_CCC___TCTCTGCTGG_TCCTCTGGT_TTTC_TGTTGG_TG_CGTC__TTTCT__T_TTTC_CCC__CC
## [2] "TT__GG__CG_TCGT_CGC_TG_T_GGGTTTTGC_GTG_T_TT_GTGTCTCGGTTG_CTGG_TCTC_TC__T_GTCTGG_TTTTGTTG_T__GT_
## [3] "_TG______C__TTT_TCC_____C__C__C___TC_GCTTCGT____TC_TTCTTTTCCCGCC__TT_G_GC__C__CTTGGCTTG_TCG__GT
```

```r
# Define some full names
names <- c("Diana Prince", "Clark Kent")

# Split into first and last names
names_split <- str_split(names, fixed(" "), simplify = TRUE)

# Extract the first letter in the first name
abb_first <- str_sub(names_split[, 1], 1, 1)

# Combine the first letter ". " and last name
str_c(abb_first, ". ", names_split[, 2])
```

```
## [1] "D. Prince" "C. Kent"
```

Regex matching

```r
library(rebus)
```

```
##
## Attaching package: 'rebus'
```

```
## The following object is masked from 'package:stringr':
##
##     regex
```

```
library(stringr)
library(htmlwidgets)
# Some strings to practice with
x <- c("cat", "coat", "scotland", "tic toc")

# Print END
END
```

```
## <regex> $
```

```
str_view(x, pattern = START %R% "c")
```

```
## PhantomJS not found. You can install it with webshot::install_phantomjs(). If it is installed, pleas
```

```
# Match the string that is exactly "cat"
str_view(x, pattern = START %R% "cat" %R% END)
```

```
str_view(c("cat", "coat", "scotland", "tic toc"),
  pattern = "c" %R% ANY_CHAR %R% "t")
```

```
# Match two characters
str_view(x, pattern = ANY_CHAR %R% ANY_CHAR)
```

```
# Match a string with exactly three characters
str_view(x, pattern = START %R% ANY_CHAR %R% ANY_CHAR %R% ANY_CHAR %R% END)
```

Regex with stringr

```
# Match Jeffrey or Geoffrey
#whole_names <- or("Jeffrey", "Geoffrey")
#str_view(boy_names, pattern = whole_names,
#  match = TRUE) # only display matches

# Match Jeffrey or Geoffrey, another way
#common_ending <- or("Je", "Geo") %R% "ffrey"
#str_view(boy_names, pattern = common_ending,
#  match = TRUE)
```

```
# Create character class containing vowels
vowels <- char_class("aeiouAEIOU")

# Print vowels
vowels
```

```
## <regex> [aeiouAEIOU]
```

```r
# See vowels in x with str_view()
str_view(x, vowels) #only matches first vowel
```

```r
# See vowels in x with str_view_all()
str_view_all(x, vowels)
```

```r
# Number of vowels in boy_names
num_vowels <- str_count(boy_names, vowels)
mean(num_vowels)
```

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

```r
# Vowels from last exercise
vowels <- char_class("aeiouAEIOU")

# See names with only vowels
str_view(boy_names,
  pattern = exactly(one_or_more(vowels)),
  match = TRUE)
```

```r
# Use `negated_char_class()` for everything but vowels
#not_vowels <- negated_char_class("aeiouAEIOU")

# See names with no vowels
#str_view(boy_names,
#  pattern = exactly(one_or_more(not_vowels)),
#  match = TRUE)
```

Shortcuts

```r
contact <- c("Call me at 555-555-0191", "123 Main St",
             "(555) 555 0191", "Phone: 555.555.0191 Mobile: 555.555.0192")

# Create a three digit pattern
three_digits <- DGT %R% DGT %R% DGT

# Test it
str_view_all(contact, pattern = three_digits)
```

```r
# Create a separator pattern
separator <- char_class("-.() ")

# Test it
str_view_all(contact, pattern = separator)
```

```r
# Use these components
three_digits <- DGT %R% DGT %R% DGT
four_digits <- three_digits %R% DGT
separator <- char_class("-.() ")
```

```r
# Create phone pattern
phone_pattern <- optional(OPEN_PAREN) %R%
  three_digits %R%
  zero_or_more(separator) %R%
  three_digits %R%
  zero_or_more(separator) %R%
  four_digits

# Use this pattern
three_digits <- DGT %R% DGT %R% DGT
four_digits <- three_digits %R% DGT
separator <- char_class("-.() ")
phone_pattern <- optional(OPEN_PAREN) %R%
  three_digits %R%
  zero_or_more(separator) %R%
  three_digits %R%
  zero_or_more(separator) %R%
  four_digits

# Extract phone numbers
str_extract(contact, phone_pattern)
```

```
## [1] "555-555-0191"   NA               "(555) 555 0191" "555.555.0191"
```

```r
# Extract ALL phone numbers
str_extract_all(contact, phone_pattern)
```

```
## [[1]]
## [1] "555-555-0191"
##
## [[2]]
## character(0)
##
## [[3]]
## [1] "(555) 555 0191"
##
## [[4]]
## [1] "555.555.0191" "555.555.0192"
```

```r
narratives <- readRDS("narratives.rds")

# Pattern to match one or two digits
age <- DGT %R% optional(DGT)

# Test it
str_view(narratives, pattern = age)
```

```r
# Use this pattern
age <- DGT %R% optional(DGT)

# Pattern to match units
```

```
unit <- optional(SPC) %R% or("YO", "YR", "MO")

# Test pattern with age then units
str_view(narratives, pattern = age %R% unit)
```

```
# Use these patterns
age <- DGT %R% optional(DGT)
unit <- optional(SPC) %R% or("YO", "YR", "MO")

# Pattern to match gender
gender <- optional(SPC) %R% or("M", "F")

# Test pattern with age then units then gender
str_view(narratives, pattern = age %R% unit %R% gender)
```

```
# Use these patterns
age <- DGT %R% optional(DGT)
unit <- optional(SPC) %R% or("YO", "YR", "MO")
gender <- optional(SPC) %R% or("M", "F")

# Extract age, unit, gender
str_extract(narratives,  age %R% unit %R% gender)
```

```
##  [1] "19YOM"   "31 YOF"  "82 YOM"  "33 YOF"  "10YOM"   "53 YO F" "13 MOF"
##  [8] "14YR M"  "55YOM"   "5 YOM"
```

Capturing methods for string manipulation

```
hero_contacts <- c("(wolverine@xmen.com)", "wonderwoman@justiceleague.org", "thor@avengers.com")

# Capture parts between @ and . and after .
email <- capture(one_or_more(WRD)) %R%
  "@" %R% capture(one_or_more(WRD)) %R%
  DOT %R% capture(one_or_more(WRD))

# Check match hasn't changed
str_view(hero_contacts, email)
```

```
# Pattern from previous step
email <- capture(one_or_more(WRD)) %R%
  "@" %R% capture(one_or_more(WRD)) %R%
  DOT %R% capture(one_or_more(WRD))

# Pull out match and captures
email_parts <- str_match(hero_contacts, email)
email_parts
```

```
##      [,1]                          [,2]          [,3]            [,4]
## [1,] "wolverine@xmen.com"          "wolverine"   "xmen"          "com"
## [2,] "wonderwoman@justiceleague.org" "wonderwoman" "justiceleague" "org"
## [3,] "thor@avengers.com"           "thor"        "avengers"      "com"
```

```
# Save host
host <- email_parts[, 3]
host
```

```
## [1] "xmen"          "justiceleague" "avengers"
```

```
# View text containing phone numbers
contact
```

```
## [1] "Call me at 555-555-0191"
## [2] "123 Main St"
## [3] "(555) 555 0191"
## [4] "Phone: 555.555.0191 Mobile: 555.555.0192"
```

```
# Add capture() to get digit parts
phone_pattern <- capture(three_digits) %R% zero_or_more(separator) %R%
           capture(three_digits) %R% zero_or_more(separator) %R%
           capture(four_digits)

# Pull out the parts with str_match()
phone_numbers <- str_match(contact, phone_pattern)

# Put them back together
str_c(
  "(",
  phone_numbers[, 2],
  ") ",
  phone_numbers[, 3],
  "-",
  phone_numbers[, 4])
```

```
## [1] "(555) 555-0191" NA               "(555) 555-0191" "(555) 555-0191"
```

```
# narratives has been pre-defined
narratives
```

```
##  [1] "19YOM-SHOULDER STRAIN-WAS TACKLED WHILE PLAYING FOOTBALL W/ FRIENDS "
##  [2] "31 YOF FELL FROM TOILET HITITNG HEAD SUSTAINING A CHI "
##  [3] "ANKLE STR. 82 YOM STRAINED ANKLE GETTING OUT OF BED "
##  [4] "TRIPPED OVER CAT AND LANDED ON HARDWOOD FLOOR. LACERATION ELBOW, LEFT. 33 YOF*"
##  [5] "10YOM CUT THUMB ON METAL TRASH CAN DX AVULSION OF SKIN OF THUMB "
##  [6] "53 YO F TRIPPED ON CARPET AT HOME. DX HIP CONTUSION "
##  [7] "13 MOF TRYING TO STAND UP HOLDING ONTO BED FELL AND HIT FOREHEAD ON RADIATOR DX LACERATION"
##  [8] "14YR M PLAYING FOOTBALL; DX KNEE SPRAIN "
##  [9] "55YOM RIDER OF A BICYCLE AND FELL OFF SUSTAINED A CONTUSION TO KNEE "
## [10] "5 YOM ROLLING ON FLOOR DOING A SOMERSAULT AND SUSTAINED A CERVICAL STRA IN"
```

```
# Add capture() to get age, unit and sex
pattern <- capture(optional(DGT) %R% DGT) %R%
  optional(SPC) %R% capture(or("YO", "YR", "MO")) %R%
  optional(SPC) %R% capture(or("M", "F"))
```

```
# Pull out from narratives
str_match(narratives, pattern)
```

```
##       [,1]      [,2] [,3] [,4]
##  [1,] "19YOM"   "19" "YO" "M"
##  [2,] "31 YOF"  "31" "YO" "F"
##  [3,] "82 YOM"  "82" "YO" "M"
##  [4,] "33 YOF"  "33" "YO" "F"
##  [5,] "10YOM"   "10" "YO" "M"
##  [6,] "53 YO F" "53" "YO" "F"
##  [7,] "13 MOF"  "13" "MO" "F"
##  [8,] "14YR M"  "14" "YR" "M"
##  [9,] "55YOM"   "55" "YO" "M"
## [10,] "5 YOM"   "5"  "YO" "M"
```

Extracting age and gender from text

```
# Edit to capture just Y and M in units
pattern2 <- capture(optional(DGT) %R% DGT) %R%
  optional(SPC) %R% capture(or("Y", "M")) %R% optional(or("O","R")) %R%
  optional(SPC) %R% capture(or("M", "F"))

# Check pattern
str_view(narratives, pattern2)
```

```
# Pull out pieces
str_match(narratives, pattern2)
```

```
##       [,1]      [,2] [,3] [,4]
##  [1,] "19YOM"   "19" "Y"  "M"
##  [2,] "31 YOF"  "31" "Y"  "F"
##  [3,] "82 YOM"  "82" "Y"  "M"
##  [4,] "33 YOF"  "33" "Y"  "F"
##  [5,] "10YOM"   "10" "Y"  "M"
##  [6,] "53 YO F" "53" "Y"  "F"
##  [7,] "13 MOF"  "13" "M"  "F"
##  [8,] "14YR M"  "14" "Y"  "M"
##  [9,] "55YOM"   "55" "Y"  "M"
## [10,] "5 YOM"   "5"  "Y"  "M"
```

Backreferences

Backreferences can be useful in matching because they allow you to find repeated patterns or words.

```
str_view(c("hello", "sweet", "kitten"),
  pattern = capture(LOWER) %R% REF1)
```

```
#boy_names <- tolower(boy_names)

# See names with three repeated letters
```

```
#repeated_three_times <- capture(LOWER) %R% REF1 %R% REF1
#str_view(boy_names,
#         pattern = repeated_three_times,
#         match = TRUE)
```

```
# View text containing phone numbers
contact
```

```
## [1] "Call me at 555-555-0191"
## [2] "123 Main St"
## [3] "(555) 555 0191"
## [4] "Phone: 555.555.0191 Mobile: 555.555.0192"
```

```
# Replace digits with "X"
str_replace(contact, DGT, "X")
```

```
## [1] "Call me at X55-555-0191"
## [2] "X23 Main St"
## [3] "(X55) 555 0191"
## [4] "Phone: X55.555.0191 Mobile: 555.555.0192"
```

```
# Replace all digits with "X"
str_replace_all(contact, DGT, "X")
```

```
## [1] "Call me at XXX-XXX-XXXX"
## [2] "XXX Main St"
## [3] "(XXX) XXX XXXX"
## [4] "Phone: XXX.XXX.XXXX Mobile: XXX.XXX.XXXX"
```

```
# Replace all digits with different symbol
str_replace_all(contact, DGT, c("X", ".", "*", "_"))
```

```
## [1] "Call me at XXX-XXX-XXXX"
## [2] "... Main St"
## [3] "(***) *** ****"
## [4] "Phone: ___.___.____ Mobile: ___.___.____"
```

```
adverbs <- readRDS("adverbs.rds")

# Build pattern to match words ending in "ING"
pattern <- one_or_more(WRD) %R% "ING"
str_view(narratives, pattern)
```

```
# Test replacement
str_replace(narratives, capture(pattern),
  str_c("CARELESSLY", REF1, sep = " "))
```

```
##  [1] "19YOM-SHOULDER STRAIN-WAS TACKLED WHILE CARELESSLY PLAYING FOOTBALL W/ FRIENDS "
##  [2] "31 YOF FELL FROM TOILET HITITNG HEAD CARELESSLY SUSTAINING A CHI "
##  [3] "ANKLE STR. 82 YOM STRAINED ANKLE CARELESSLY GETTING OUT OF BED "
```

```
##  [4] "TRIPPED OVER CAT AND LANDED ON HARDWOOD FLOOR. LACERATION ELBOW, LEFT. 33 YOF*"
##  [5] "10YOM CUT THUMB ON METAL TRASH CAN DX AVULSION OF SKIN OF THUMB "
##  [6] "53 YO F TRIPPED ON CARPET AT HOME. DX HIP CONTUSION "
##  [7] "13 MOF CARELESSLY TRYING TO STAND UP HOLDING ONTO BED FELL AND HIT FOREHEAD ON RADIATOR DX LACE
##  [8] "14YR M CARELESSLY PLAYING FOOTBALL; DX KNEE SPRAIN "
##  [9] "55YOM RIDER OF A BICYCLE AND FELL OFF SUSTAINED A CONTUSION TO KNEE "
## [10] "5 YOM CARELESSLY ROLLING ON FLOOR DOING A SOMERSAULT AND SUSTAINED A CERVICAL STRA IN"
```

```
# One adverb per narrative
adverbs_10 <- sample(adverbs, 10)

# Replace "***ing" with "adverb ***ly"
str_replace(narratives,
  capture(pattern),
  str_c(adverbs_10, REF1, sep = " "))
```

```
##  [1] "19YOM-SHOULDER STRAIN-WAS TACKLED WHILE FERVENTLY PLAYING FOOTBALL W/ FRIENDS "
##  [2] "31 YOF FELL FROM TOILET HITITNG HEAD FEROCIOUSLY SUSTAINING A CHI "
##  [3] "ANKLE STR. 82 YOM STRAINED ANKLE WARMLY GETTING OUT OF BED "
##  [4] "TRIPPED OVER CAT AND LANDED ON HARDWOOD FLOOR. LACERATION ELBOW, LEFT. 33 YOF*"
##  [5] "10YOM CUT THUMB ON METAL TRASH CAN DX AVULSION OF SKIN OF THUMB "
##  [6] "53 YO F TRIPPED ON CARPET AT HOME. DX HIP CONTUSION "
##  [7] "13 MOF ARROGANTLY TRYING TO STAND UP HOLDING ONTO BED FELL AND HIT FOREHEAD ON RADIATOR DX LACE
##  [8] "14YR M TOO PLAYING FOOTBALL; DX KNEE SPRAIN "
##  [9] "55YOM RIDER OF A BICYCLE AND FELL OFF SUSTAINED A CONTUSION TO KNEE "
## [10] "5 YOM REGULARLY ROLLING ON FLOOR DOING A SOMERSAULT AND SUSTAINED A CERVICAL STRA IN"
```

```
library(stringi)

# Names with builtin accents
tay_son_builtin <- c(
  "Nguy\u1ec5n Nh\u1ea1c",
  "Nguy\u1ec5n Hu\u1ec7",
  "Nguy\u1ec5n Quang To\u1ea3n"
)

# Convert to separate accents
tay_son_separate <- stri_trans_nfd(tay_son_builtin)

#Verify that the string prints the same
tay_son_separate
```

```
## [1] "Nguye^~n Nha<U+0323>c" "Nguye^~n Hue<U+0323>^" "Nguye^~n Quang Toa<U+0309>n"
```

```
# Match all accents
str_view_all(tay_son_separate, UP_DIACRITIC)
```

```
# tay_son_separate has been pre-defined
tay_son_separate
```

```
## [1] "Nguye^~n Nha<U+0323>c" "Nguye^~n Hue<U+0323>^" "Nguye^~n Quang Toa<U+0309>n"
```

```
# View all the characters in tay_son_separate
str_view_all(tay_son_separate, ANY_CHAR)
```

```
# View all the graphemes in tay_son_separate
str_view_all(tay_son_separate, GRAPHEME)
```

```
# Combine the diacritics with their letters
tay_son_builtin <- stri_trans_nfc(tay_son_separate)
tay_son_builtin
```

```
## [1] "Nguy<U+1EC5>n Nh<U+1EA1>c" "Nguy<U+1EC5>n Hu<U+1EC7>" "Nguy<U+1EC5>n Quang To<U+1EA3>n"
```

```
# View all the graphemes in tay_son_builtin
str_view_all(tay_son_builtin, GRAPHEME)
```

Case Study

```
earnest <- stri_read_lines("earnest.txt")

# Detect start and end lines
start <- str_which(earnest, fixed("START OF THE PROJECT"))
end <- str_which(earnest, fixed("END OF THE PROJECT"))

# Get rid of gutenberg intro text
earnest_sub  <- earnest[(start + 1):(end - 1)]

# Detect first act
lines_start <- which(str_detect(earnest_sub, fixed("FIRST ACT")))

# Set up index
intro_line_index <- 1:(lines_start - 1)

# Split play into intro and play
intro_text <- earnest_sub[intro_line_index]
play_text <- earnest_sub[-intro_line_index]

# Take a look at the first 20 lines
writeLines(play_text[1:20])
```

```
## FIRST ACT
##
##
## SCENE
##
##
## Morning-room in Algernon's flat in Half-Moon Street.  The room is
## luxuriously and artistically furnished.  The sound of a piano is heard in
## the adjoining room.
##
## [Lane is arranging afternoon tea on the table, and after the music has
```

```
## ceased, Algernon enters.]
##
## Algernon.  Did you hear what I was playing, Lane?
##
## Lane.  I didn't think it polite to listen, sir.
##
## Algernon.  I'm sorry for that, for your sake.  I don't play
## accurately--any one can play accurately--but I play with wonderful
## expression.  As far as the piano is concerned, sentiment is my forte.  I
```

```
# Get rid of empty strings
empty <- stri_isempty(play_text)
play_lines <- play_text[!empty]
```

```
# Pattern for start, word then .
pattern_1 <- START %R% one_or_more(WRD) %R% DOT

# Test pattern_1
str_view(play_lines, pattern_1, match = TRUE)
```

```
str_view(play_lines, pattern_1, match = FALSE)
```

```
# Pattern for start, capital, word then .
pattern_2 <- START %R% ascii_upper() %R% one_or_more(WRD) %R% DOT

# Test pattern_2
str_view(play_lines, pattern_2, match = TRUE)
```

```
str_view(play_lines, pattern_2, match = FALSE)
```

```
# Pattern from last step
pattern_2 <- START %R% ascii_upper() %R% one_or_more(WRD) %R% DOT

# Get subset of lines that match
lines <- str_subset(play_lines, pattern_2)

# Extract match from lines
who <- str_extract(lines, pattern_2)

# Let's see what we have
unique(who)
```

```
##  [1] "Algernon."   "Lane."       "Jack."       "Cecily."     "Ernest."
##  [6] "University." "Gwendolen."  "July."       "Chasuble."   "Merriman."
## [11] "Sunday."     "Mr."         "London."     "Cardew."     "Opera."
## [16] "Markby."     "Oxonian."
```

```
# Create vector of characters
characters <- c("Algernon", "Jack", "Lane", "Cecily", "Gwendolen", "Chasuble",
  "Merriman", "Lady Bracknell", "Miss Prism")
```

```r
# Match start, then character names then .
pattern_3 <- START %R% or1(characters) %R% DOT

# View matches of pattern_3
str_view(play_lines, pattern_3, match = TRUE)
```

```r
# View non-matches of pattern_3
str_view(play_lines, pattern_3, match = FALSE)
```

```r
# Variables from previous step
characters <- c("Algernon", "Jack", "Lane", "Cecily", "Gwendolen", "Chasuble",
  "Merriman", "Lady Bracknell", "Miss Prism")
pattern_3 <- START %R% or1(characters) %R% DOT

# Pull out matches
lines <- str_subset(play_lines, pattern_3)

# Extract match from lines
who <- str_extract(lines, pattern_3)

# Let's see what we have
unique(who)
```

```
## [1] "Algernon."       "Lane."           "Jack."           "Cecily."
## [5] "Gwendolen."      "Lady Bracknell." "Miss Prism."     "Chasuble."
## [9] "Merriman."
```

```r
# Count lines per character
table(who)
```

```
## who
##       Algernon.         Cecily.       Chasuble.      Gwendolen.           Jack.
##             201             154              42             102             219
## Lady Bracknell.           Lane.       Merriman.     Miss Prism.
##              84              21              17              41
```

```r
x <- c("Cat", "CAT", "cAt")
str_view(x, "cat")
```

```r
str_view(str_to_lower(x), "cat")
```

```r
catcidents <- readRDS("catcidents.rds")

head(catcidents)
```

```
## [1] "79yOf Fractured fingeR tRiPPED ovER cAT ANd fell to FlOOr lAst nIGHT AT HOME*"
## [2] "21 YOF REPORTS SUS LACERATION OF HER LEFT HAND WHEN SHE WAS OPENING A CAN OF CAT FOOD JUST PTA.
## [3] "87YOF TRIPPED OVER CAT, HIT LEG ON STEP. DX LOWER LEG CONTUSION "
## [4] "bLUNT CHest trAUma, R/o RIb fX, R/O CartiLAgE InJ To RIB cAge; 32YOM walKiNG DOG, dog took OfF a
## [5] "42YOF TO ER FOR BACK PAIN AFTER PUTTING DOWN SOME CAT LITTER DX: BACK PAIN, SCIATICA"
## [6] "4YOf DOg jUst hAd PUpPieS, Cat TRIED 2 get PuPpIes, pT THru CaT dwn stA Irs, LoST foOTING & FEL
```

```
# Construct pattern of DOG in boundaries
whole_dog_pattern <- whole_word("DOG")

# View matches to word "DOG"
str_view(catcidents, pattern = whole_dog_pattern, match = TRUE)
```

```
# Transform catcidents to upper case
catcidents_upper <- str_to_upper(catcidents)

# View matches to word "DOG" again
str_view(catcidents_upper, pattern = whole_dog_pattern, match = TRUE)
```

```
# Which strings match?
has_dog <- str_detect(catcidents_upper, pattern = whole_dog_pattern)

# Pull out matching strings in original
catcidents[has_dog]
```

```
##  [1] "bLUNT CHest trAUma, R/o RIb fX, R/O CartiLAgE InJ To RIB cAge; 32YOM walKiNG DOG, dog took OfF
##  [2] "4YOf DOg jUst hAd PUpPieS, Cat TRIED 2 get PuPpIes, pT THru CaT dwn stA Irs, LoST foOTING & FE
##  [3] "unhelmeted 14yof riding her bike with her dog when she saw a cat and sw erved c/o head/shoulde
##  [4] "Rt Shoulder Strain.26Yof Was Walking Dog On Leash And Dot Saw A Cat And Pulled Leash."
##  [5] "67 YO F WENT TO WALK DOG, IT STARTED TO CHASE CAT JERKED LEASH PULLED H ER OFF PATIO, FELL HUR
##  [6] "46yof taking dog outside, dog bent her fingers back on a door. dog jerk ed when saw cat. hand
##  [7] "PUSHING HER UTD WITH SHOTS DOG AWAY FROM THE CAT'S BOWL&BITTEN TO FINGE R>>PW/DOG BITE"
##  [8] "DX R SH PN: 27YOF W/ R SH PN X 5D. STATES WAS YANK' BY HER DOG ON LEASH W DOG RAN AFTER CAT; W
##  [9] "39Yof dog pulled her down the stairs while chasing a cat dx: rt ankle inj"
## [10] "44Yof Walking Dog And The Dof Took Off After A Cat And Pulled Pt Down B Y The Leash Strained N
```

Ignoring cases when matching

```
x <- c("Cat", "CAT", "cAt")
str_view(x, "cat")
```

```
str_view(x,
  regex("cat", ignore_case = TRUE))
```

```
# Construct case insensitive pattern
trip_pattern <- regex("TRIP", ignore_case = TRUE)

# View case insensitive matches to "TRIP"
str_view(catcidents, pattern = trip_pattern, match = TRUE)
```

```
# Get subset of matches
trip <- str_subset(catcidents, pattern = trip_pattern)

# Extract matches
str_extract(trip, pattern = trip_pattern)
```

```
## character(0)
```

```r
# Get first five catcidents
cat5 <- catcidents[1:5]

# Take a look at original
writeLines(cat5)
```

```
## 79yOf Fractured fingeR tRiPPED ovER cAT ANd fell to FlOOr lAst nIGHT AT HOME*
## 21 YOF REPORTS SUS LACERATION OF HER LEFT HAND WHEN SHE WAS OPENING A CAN OF CAT FOOD JUST PTA. DX HA
## 87YOF TRIPPED OVER CAT, HIT LEG ON STEP. DX LOWER LEG CONTUSION
## bLUNT CHest trAUma, R/o RIb fX, R/O CartiLAgE InJ To RIB cAge; 32YOM walKiNG DOG, dog took OfF aFtER
## 42YOF TO ER FOR BACK PAIN AFTER PUTTING DOWN SOME CAT LITTER DX: BACK PAIN, SCIATICA
```

```r
# Transform to title case
writeLines(str_to_title(cat5))
```

```
## 79yof Fractured Finger Tripped Over Cat And Fell To Floor Last Night At Home*
## 21 Yof Reports Sus Laceration Of Her Left Hand When She Was Opening A Can Of Cat Food Just Pta. Dx Ha
## 87yof Tripped Over Cat, Hit Leg On Step. Dx Lower Leg Contusion
## Blunt Chest Trauma, R/O Rib Fx, R/O Cartilage Inj To Rib Cage; 32yom Walking Dog, Dog Took Off After
## 42yof To Er For Back Pain After Putting Down Some Cat Litter Dx: Back Pain, Sciatica
```

```r
# Transform to title case with stringi
writeLines(stri_trans_totitle(cat5)) #same
```

```
## 79yof Fractured Finger Tripped Over Cat And Fell To Floor Last Night At Home*
## 21 Yof Reports Sus Laceration Of Her Left Hand When She Was Opening A Can Of Cat Food Just Pta. Dx Ha
## 87yof Tripped Over Cat, Hit Leg On Step. Dx Lower Leg Contusion
## Blunt Chest Trauma, R/O Rib Fx, R/O Cartilage Inj To Rib Cage; 32yom Walking Dog, Dog Took Off After
## 42yof To Er For Back Pain After Putting Down Some Cat Litter Dx: Back Pain, Sciatica
```

```r
# Transform to sentence case with stringi
writeLines(stri_trans_totitle(cat5, type = "sentence"))
```

```
## 79yof fractured finger tripped over cat and fell to floor last night at home*
## 21 yof reports sus laceration of her left hand when she was opening a can of cat food just pta. Dx ha
## 87yof tripped over cat, hit leg on step. Dx lower leg contusion
## Blunt chest trauma, r/o rib fx, r/o cartilage inj to rib cage; 32yom walking dog, dog took off after
## 42yof to er for back pain after putting down some cat litter dx: back pain, sciatica
```