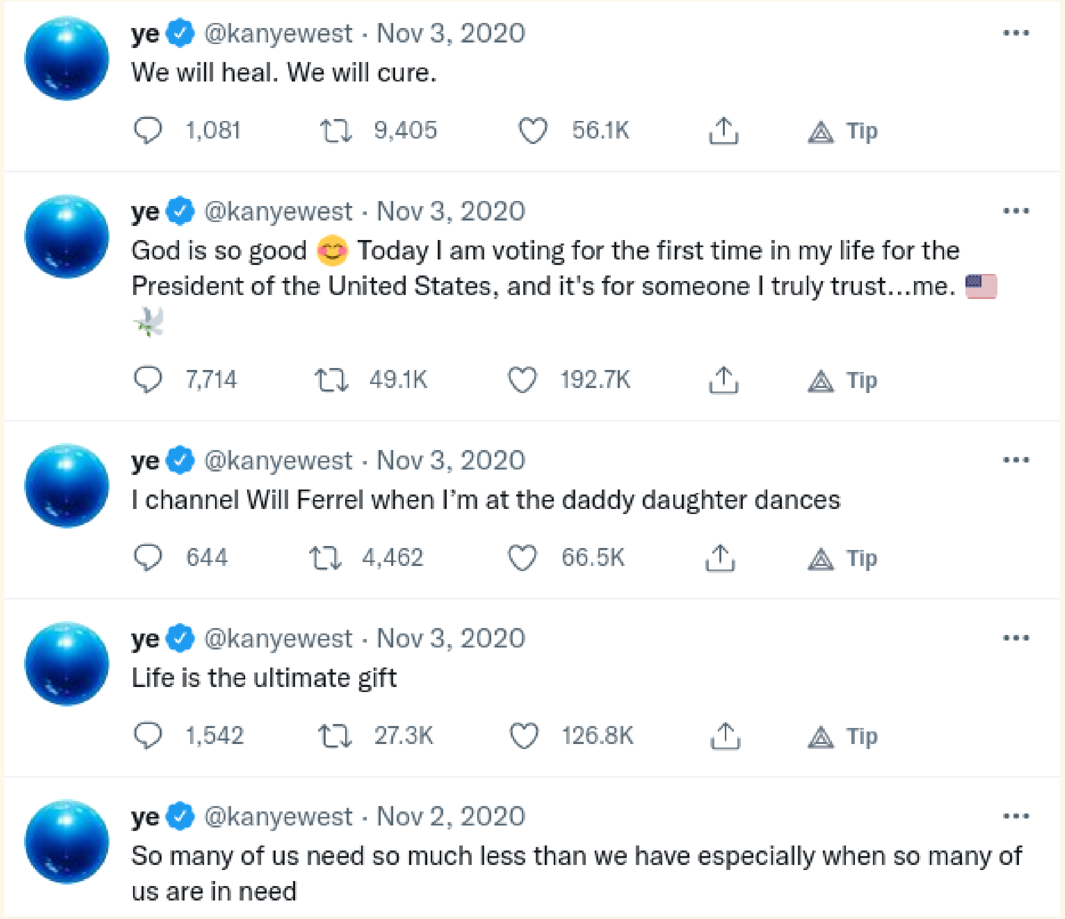
Basic String Manipulation

Fall 2022

October 05 2022

Kanye West tweets analysis



ye @kanyewest · Nov 3, 2020

We will heal. We will cure.

1,081 9,405 56.1K Tip

ye @kanyewest · Nov 3, 2020

God is so good 😊 Today I am voting for the first time in my life for the President of the United States, and it's for someone I truly trust…me. 🇺🇸 🕊️

7,714 49.1K 192.7K Tip

ye @kanyewest · Nov 3, 2020

I channel Will Ferrel when I'm at the daddy daughter dances

644 4,462 66.5K Tip

ye @kanyewest · Nov 3, 2020

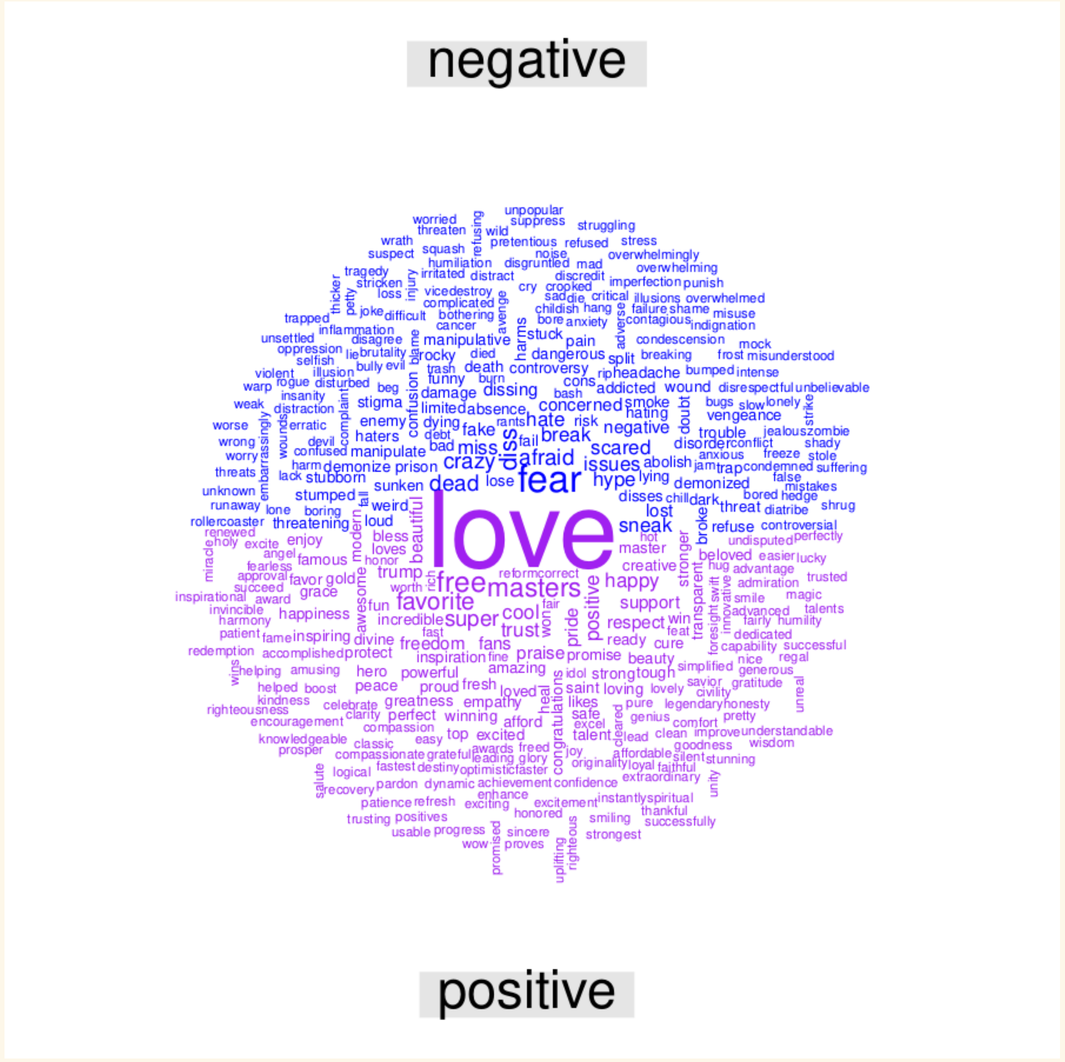
Life is the ultimate gift

1,542 27.3K 126.8K Tip

ye @kanyewest · Nov 2, 2020

So many of us need so much less than we have especially when so many of us are in need

Kanye West Twitter Feed



Tweets that look like this …

```
# A tibble: 5 × 1
  text
  <chr>
1 "love is the absence of pride"
2 "Donda House 🙏👽👽 https://t.co/i4pfZiZmeN"
3 "enhance structure"
4 "Film concept: a being purchases a Merriam-Webster’s dictionary from Walgreen…
5 "Me Troy Justin and Scooter will have dinner and build on how we will make ou…
```

… are changed to a cleaner version

```
kanye %>% select(text_clean) %>% slice(1:5)
# A tibble: 5 × 1
  text_clean
  <chr>
1 love is the absence of pride
2 donda house
3 enhance structure
4 film concept a being purchases a merriam webster s dictionary from walgreens …
5 me troy justin and scooter will have dinner and build on how we will make our…
```

Let's Define Strings

- A string is any sequence of characters
- Define a string by surrounding text with either single quotes or double quotes.

```
s <- "Hello!"     # double quotes define a string
s <- 'Hello!'     # single quotes define a string
```

- The `cat()` or `writeLines()` function displays a string as it is represented inside R.

```
cat(s)
Hello!
```

```
writeLines(s)
Hello!
```

```
s <- `Hello`     # backquotes do not define a string
s <- "10""    # error - unclosed quotes
```

String Parsing

pulling apart some text or `string` to do something with it

- The most common tasks in string processing include:
    - extracting numbers from strings, e.g. "12%"
    - removing unwanted characters from text, e.g. "New Jersey_* "
    - finding and replacing characters, e.g. "2,150"
    - extracting specific parts of strings, e.g. "Temp 40 F$"
    - splitting strings into multiple values, e.g. "Boston, MA"

Regular expressions: Regex

Regular expressions are a language for expressing patterns in strings

- Regex can include special characters unlike a regular string
- To use regex in R, you need to use the stringr package

`stringr` package

- detecting, locating, extracting and replacing elements of strings.
- begin with `str_` and take the string as the first argument

Special characters

- The "escape" backslash \ is used to escape the special use of certain characters

```
writeLines("\"")
"
writeLines("\\")
\
writeLines("Math\\Stats")
Math\Stats
```

- To include both single and double quotes in string, escape with \

```
s <- '5\'10"'     # outer single quote
writeLines(s)
5'10"
```

```
s <- "5'10\""     # outer double quote
writeLines(s)
5'10"
```

Combining strings

```
str_c("iron", "wine")
[1] "ironwine"
str_flatten(c("iron", "wine"), collapse = " and ")
[1] "iron and wine"
```

```
a <- c("a", "b", "c")
b <- c("A", "B", "C")
str_c(a, b)
[1] "aA" "bB" "cC"
```

```
building <- "CMC"
room <- "102"
begin_time <- "11:10 a.m."
end_time <- "12:20 p.m."
days <- "MWF"
class <- "STAT 220"
```

```
str_c(class, "meets from", begin_time, "to", end_time,
      days, "in", building, room, sep = " ")
[1] "STAT 220 meets from 11:10 a.m. to 12:20 p.m. MWF in CMC 102"
```

Lengths of strings

We can manage the lengths of strings using the following set of functions:

- `str_length()`
- `str_pad()`
- `str_trunc()`
- `str_trim()`

`str_length()`

tells you how many characters are in each entry of a character vector

```
gapminder %>% names()
[1] "country"   "continent" "year"      "lifeExp"   "pop"       "gdpPercap"
```

```
# length of each column names
gapminder %>% names() %>% str_length()
[1] 7 9 4 7 3 9
```

`str_pad()`

standardizes the length of strings in a character vector by padding it on the left or right ends with a specified character (by default, a space)

```
gapminder %>% names() %>%
  str_pad(width = 9, # minimum width of the string to fill/pad positions up to
          side = "both", # side to insert the extra characters on
          pad = "+")
[1] "+country+" "continent" "++year+++" "+lifeExp+" "+++pop+++" "gdpPercap"
```

str_trunc()

standardizes string lengths by controlling the maximum width and truncating strings longer than this

```
gapminder %>% names() %>%
  str_trunc(width = 7, #the maximum width to allow for strings
          side= "right",
          # entries which have been truncated will show this
          # to indicate that something has been removed
          ellipsis = "...")
[1] "country" "cont..." "year"    "lifeExp" "pop"     "gdpP..."
```

`str_trim()`

removes empty spaces on the ends of a string

```
#add some whitespace
padded_names <- gapminder %>% names() %>% str_pad(12, "both")
padded_names
[1] "  country   " " continent  " "    year    " "  lifeExp   " "    pop     "
[6] " gdpPercap  "
```

```
#remove it
str_trim(padded_names)
[1] "country"   "continent" "year"      "lifeExp"   "pop"       "gdpPercap"
```

`str_glue()`

allows one to interpolate strings and values that have been assigned to names in R

```
y <- Sys.Date() # current date
str_glue("today is {y}")
today is 2022-10-05
```

```
# base R equivalent
paste0("today is ", y)
[1] "today is 2022-10-05"
```

```
nm <- "Alex"
str_glue("Hi, my name is {nm}.")
Hi, my name is Alex.
```

```
a <- 5
str_glue("a = {a}")
a = 5
```

str_sub()

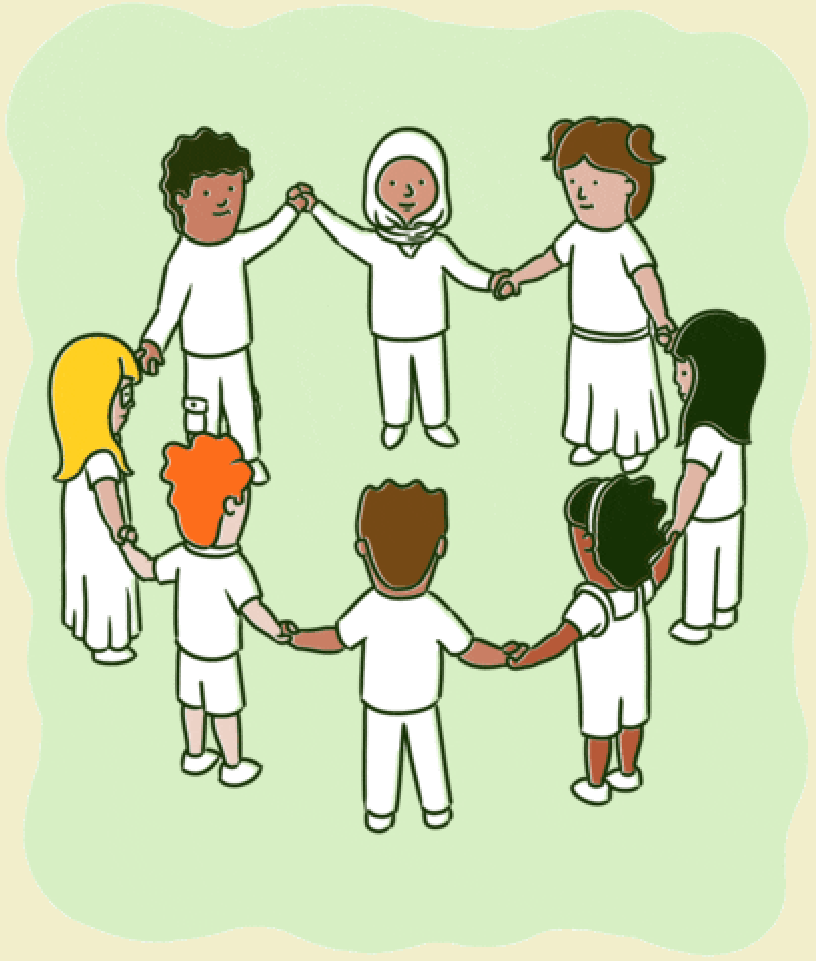
Extract and replace substrings from a character vector

```r
phrase <- "cellar door"
str_sub(phrase, start = 1, end = 6)
[1] "cellar"
```

```r
str_sub(phrase, start = c(1,8), end = c(6,11))
[1] "cellar" "door"
```

Group Activity 1

05:00



- Let's go over to maize server/ local Rstudio and our class moodle
- Get the class activity 11.Rmd file
- Work on problem 1
- Ask me questions

More Special Characters

- The `|` symbol inside a regex means `"or"`.
- Use `\n` to matche a newline character
- Use `\s` to match white space characters (spaces, tabs, and newlines)
- Use `\w` to match alphanumeric characters (letters and numbers)
  - can also use `[:alpha:]`
- Use `\d` to represent digits (numbers)
  - can also use `[:digit:]`

Click here for extensive lists

More Special Characters

- `^` = start of a string
- `$` = end of a string
- `.` = any character

Quantifiers

- `*` = matches the preceding character any number of times
- `+` = matches the preceding character once
- `?` = matches the preceding character at most once (i.e. optionally)
- `{n}` = matches the preceding character exactly n times

Try more regexes here

str_detect()

```
days <- c("Monday", "Tuesday", "Wednesday",
              "Thursday", "Friday", "Saturday", "Sunday")
```

```
str_detect(days, "^[Ss]un.*")
[1] FALSE FALSE FALSE FALSE FALSE FALSE  TRUE
```

```
days %>%
  str_which("^T")  # indices of matching entries
[1] 2 4
```

str_subset()

returns all values in a vector which match a pattern

```
gapminder$country %>%
  unique() %>%
  str_subset("^[CU].*a$")
[1] "Cambodia"   "Canada"     "China"      "Colombia"   "Costa Rica"
[6] "Croatia"    "Cuba"       "Uganda"
```

```
# columns with names starting with "c"
gapminder %>%
  names() %>%
  str_subset("^c")
[1] "country"   "continent"
```

str_sub()

extracts parts of strings based on their position with the start and end arguments

```
gapminder %>%
  names() %>%
  str_sub(start = 1, end = 6) # return the 1st 6 characters of each column name
[1] "countr" "contin" "year"   "lifeEx" "pop"    "gdpPer"
```

str_extract()

extract just the part of the string matching the specified regex instead of the entire entry

```
name_phone <- c("Moly: 250-999-8878",
       "Ali: 416-908-2044",
       "Eli: 204-192-9829",
       "May: 250-209-7047")
str_extract(name_phone, "[:alpha:]*")
[1] "Moly" "Ali"  "Eli"  "May"
```

str_split()

splits a string into a list or matrix of pieces based on a supplied pattern

```
str_split(c("a_3", "d_4"), pattern = "_") # returns a list
[[1]]
[1] "a" "3"

[[2]]
[1] "d" "4"
```

```
str_split(c("a_3", "d_4"), pattern = "_", simplify = TRUE) # returns a matrix
     [,1] [,2]
[1,] "a"  "3"
[2,] "d"  "4"
```

`str_replace()`

`str_replace()` replaces the first instance of the detected pattern with a specified string.

```r
gap_names <- gapminder %>% names

str_replace(gap_names,
            pattern = "^.{3}", # match the 1st 3 characters of each string in the vector
            replacement = "X_")
[1] "X_ntry"   "X_tinent" "X_r"      "X_eExp"   "X_"       "X_Percap"
```

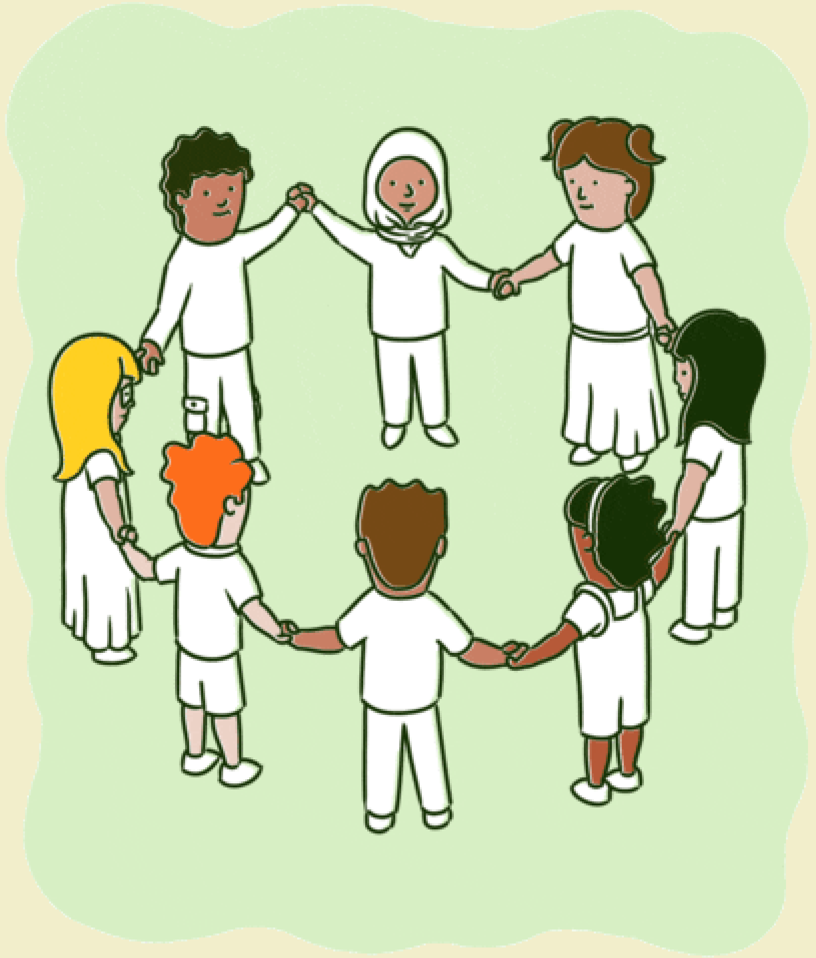
str_replace_all()

```
head(murders, 2)
# A tibble: 2 × 4
  state   population total murder_rate
  <chr>   <chr>      <chr>       <dbl>
1 Alabama 4,853,875  348           7.2
2 Alaska  737,709    59            8
```

```
# detect whether a comma is present
murders %>% mutate(population = str_replace_all(population, ",", ""),
                   total = str_replace_all(total, ",", "")) %>% mutate_at(vars(2:3), as.double)
# A tibble: 51 × 4
   state                population total murder_rate
   <chr>                     <dbl> <dbl>       <dbl>
 1 Alabama                 4853875   348         7.2
 2 Alaska                   737709    59         8
 3 Arizona                 6817565   309         4.5
 4 Arkansas                2977853   181         6.1
 5 California             38993940  1861         4.8
 6 Colorado                5448819   176         3.2
 7 Connecticut             3584730   117         3.3
 8 Delaware                 944076    63         6.7
 9 District of Columbia     670377   162        24.2
10 Florida                20244914  1041         5.1
# … with 41 more rows
```

Group Activity 2

08:00



- Go back to the activity file
- Work on problems 2-3
- Ask me questions