Intro to strings and regular expressions

Warmup activity

Work on the activity (handout) with a neighbor, then we will discuss as a class

Warmup

- Are You Pikachu Or Meowth
- 23 Photos That Definitively Prove The Moon Landing Was Faked
- Chinese Exports Fall 22.6% in April
- From a Beijing Suburb, Vibrant Strings
- 27 Happy Gifts For People Who Love Jamaica
- Two largest known prime numbers discovered just two weeks apart, one qualifies for $100k prize
- Pakistan : New policy on renewable energy launched
- Strong earthquake hits Pakistan, north India, Afghanistan
- National Debate About G.O.P. Hits Home in Utah
- This 7-Picture Test Will Determine What Type Of Harry Potter Fan You Are
- 22 Fred And George Weasley Moments That'll Make You Laugh, Cry, And Everything In Between
- 21 Puppies Who Absolutely Cannot Be Trusted

Characterizing clickbait headlines

Clickbait headlines often contain numbers:

- 23 Photos That Definitively Prove The Moon Landing Was Faked
- 21 Puppies Who Absolutely Cannot Be Trusted

Clickbait headlines are often written in first or second person:

- Are You Pikachu Or Meowth
- This 7-Picture Test Will Determine What Type Of Harry Potter Fan You Are

Another example

We conduct a survey, and the results contain the following responses:

- “I am 31 years old”
- “I just turned 52”
- “My age is 83”

If we want to explore statistics about respondents’ ages (summary statistics, visualizations, regression models, etc.), what do we need to do first?

Strings

Strings are data that consist of a sequence of characters, and store information like names and text responses. We use single or double quotes when creating a string:

```
1 ex_str <- "Hello!"
2 ex_str
```

```
[1] "Hello!"
```

The number of characters in a string is called its *length*:

```
1 str_length(ex_str)
```

```
[1] 6
```

Extracting information from strings

Working with text data requires us to identify and extract useful information in strings. For example, we may wish to extract the number from a string:

```
 1 str_extract("I am 31 years old", "31")
```

```
[1] "31"
```

```
 1 str_extract("21 Puppies Who Absolutely Cannot Be Trusted", "21")
```

```
[1] "21"
```

`str_extract`: extracts the first match in a string to a specified pattern

Question: Are there any issues with the way we are extracting numbers here?

More general patterns: regular expressions

Instead of specifying a specific number, we can ask R to find *any* number:

```
1 str_extract("My son is 9 years old", "\\d")
```

```
[1] "9"
```

- `\d` is a special character that means *match any digit*
- In R, we need to add an additional escape character, so we enter this as `\\d`

Looking for numbers

What do you think will happen if I run the following code?

```
1 str_extract("My son is 9 years old", "d")
```

Looking for numbers

What do you think will happen if I run the following code?

```
1 str_extract("My son is 9 years old", "d")
```

```
[1] "d"
```

This just looks for the letter `"d"`! To get the special character meanining “any digit”, we need the escape character(s):

```
1 str_extract("My son is 9 years old", "\\d")
```

```
[1] "9"
```

Looking for numbers

What do you think will happen if I run the following code?

```
1 str_extract("My son is 19 years old", "\\d")
```

Looking for numbers

What do you think will happen if I run the following code?

```
1 str_extract("My son is 19 years old", "\\d")
```

```
[1] "1"
```

The pattern `\d` will just return the *first* match. To get the full "19", we need to match any contiguous sequence of digits:

```
1 str_extract("My son is 19 years old", "\\d+")
```

```
[1] "19"
```

`+` means "one or more occurrences"

Looking for numbers

What do you think will happen if I run the following code?

```
1 str_extract("My son is 19 years old, and I am 51", "\\d+")
```

Looking for numbers

What do you think will happen if I run the following code?

```
1 str_extract("My son is 19 years old, and I am 51", "\\d+")
```

```
[1] "19"
```

`str_extract` returns the *first* match to the pattern. To get *all* matches:

```
1 str_extract_all("My son is 19 years old, and I am 51", "\\d+")
```

```
[[1]]
[1] "19" "51"
```

Looking for numbers

String functions in the `stringr` package are also vectorized:

```
ex_strings <- c("My son is 19 years old, and I am 51",
                "21 Puppies Who Absolutely Cannot Be Trusted")

str_extract(ex_strings, "\\d+")
```

```
[1] "19" "21"
```

Another string function

Instead of extracting a pattern, we may wish to *detect* whether the string *contains* a pattern:

```
1 ex_strings <- c("23 Photos That Definitively Prove The Moon Landing \
2                 "21 Puppies Who Absolutely Cannot Be Trusted",
3                 "Pakistan : New policy on renewable energy launched"
4
5 str_detect(ex_strings, "\\d+")
```

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

We can also see *where* the match occurs:

```
1 str_view(ex_strings, "\\d+")
```

```
[1] │ <23> Photos That Definitively Prove The Moon Landing Was Faked
[2] │ <21> Puppies Who Absolutely Cannot Be Trusted
```

Another string function

Instead of extracting a pattern, we may wish to *detect* whether the string *contains* a pattern:

```
ex_strings <- c("23 Photos That Definitively Prove The Moon Landing W
                "21 Puppies Who Absolutely Cannot Be Trusted",
                "Pakistan : New policy on renewable energy launched"

str_detect(ex_strings, "\\d+")
```

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

And we can select only the strings which contain the pattern:

```
str_subset(ex_strings, "\\d+")
```

```
[1] "23 Photos That Definitively Prove The Moon Landing Was Faked"
[2] "21 Puppies Who Absolutely Cannot Be Trusted"
```

Back to clickbait

```
1 str_subset(headlines, "\\d+")
```

```
[1] "23 Photos That Definitively Prove The Moon Landing Was Faked"
[2] "Chinese Exports Fall 22.6% in April"
[3] "27 Happy Gifts For People Who Love Jamaica"
[4] "Two largest known prime numbers discovered just two weeks apart,
one qualifies for $100k prize"
[5] "This 7-Picture Test Will Determine What Type Of Harry Potter Fan
You Are"
[6] "22 Fred And George Weasley Moments That'll Make You Laugh, Cry,
And Everything In Between"
[7] "21 Puppies Who Absolutely Cannot Be Trusted"
```

Are all of these headlines clickbait?

Back to clickbait

We can look for a pattern at the *beginning* of a string with the ^ character:

```
1 str_subset(headlines, "^\\d+")
```

```
[1] "23 Photos That Definitively Prove The Moon Landing Was Faked"
[2] "27 Happy Gifts For People Who Love Jamaica"
[3] "22 Fred And George Weasley Moments That'll Make You Laugh, Cry,
And Everything In Between"
[4] "21 Puppies Who Absolutely Cannot Be Trusted"
```

Anchors

We can look for a pattern at the *beginning* of a string with the ^ character:

```
1 str_subset(headlines, "^\\d+")
```

```
[1] "23 Photos That Definitively Prove The Moon Landing Was Faked"
[2] "27 Happy Gifts For People Who Love Jamaica"
[3] "22 Fred And George Weasley Moments That'll Make You Laugh, Cry,
And Everything In Between"
[4] "21 Puppies Who Absolutely Cannot Be Trusted"
```

Question: When might I want to look for a pattern at the *end* of a string?

Anchors

```
str_subset(headlines, "^\\d+")
```

```
[1] "23 Photos That Definitively Prove The Moon Landing Was Faked"
[2] "27 Happy Gifts For People Who Love Jamaica"
[3] "22 Fred And George Weasley Moments That'll Make You Laugh, Cry,
And Everything In Between"
[4] "21 Puppies Who Absolutely Cannot Be Trusted"
```

```
str_detect("my_file.png", "csv$")
```

```
[1] FALSE
```

```
str_detect("file2.csv", "csv$")
```

```
[1] TRUE
```

```
str_detect("csv_folder/accident.xlsx", "csv")
```

```
[1] TRUE
```

```
str_detect("csv_folder/accident.xlsx", "csv$")
```

```
[1] FALSE
```

Regular expressions

Regular expression: a tool for specifying a search pattern in text. (Note: regular expressions are not specific to R, and are used in many languages and platforms)

Some regular expressions so far:

- `\d` any digit
- `+` one or more occurrences
- `^` anchors at the beginning
- `$` anchors at the end

Class activity

- Work independently or with a neighbor on the class activity
- At the end of class, submit your work as an HTML file on Canvas (one per group, list all your names)

For next time, read:

- Chapter 15.1 - 15.3 in *R for Data Science*