Getting / cleaning data 2

More on using regular expressions

Regular expressions

The `str_detect` function will look through each element of a character vector for a designated pattern. If the pattern is there, it will return `TRUE`, and otherwise `FALSE`. The convention is:

```
## Generic code
str_detect(string = [vector you want to check],
           pattern = [pattern you want to check for])
```

For example, to create a logical vector specifying which of the Titanic passenger names include "Mrs.", you can call:

```
mrs <- str_detect(titanic_train$Name, "Mrs\\.")
head(mrs)
```

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

Regular expressions

The result is a logical vector, so `str_detect` can be used in `filter` to subset data to only rows where the passenger's name includes "Mrs.":

```
titanic_train %>%
  filter(str_detect(Name, "Mrs\\.")) %>%
  select(Name) %>%
  slice(1:3)
```

```
##                                                     Name
## 1 Cumings, Mrs. John Bradley (Florence Briggs Thayer)
## 2        Futrelle, Mrs. Jacques Heath (Lily May Peel)
## 3   Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)
```

Regular expressions

The `str_extract` function can be used to extract a string (if it exists) from each value in a character vector. It follows similar conventions to `str_detect`:

```
## Generic code
str_extract(string = [vector you want to check],
            pattern = [pattern you want to check for])
```

Regular expressions

For example, you might want to extract "Mrs." if it exists in a passenger's name:

```
titanic_train %>%
  mutate(mrs = str_extract(Name, "Mrs\\.")) %>%
  select(Name, mrs) %>%
  slice(1:3)
```

```
##                                                  Name  mrs
## 1                             Braund, Mr. Owen Harris <NA>
## 2 Cumings, Mrs. John Bradley (Florence Briggs Thayer) Mrs.
## 3                              Heikkinen, Miss. Laina <NA>
```

Notice that now we're creating a new column (`mrs`) that either has "Mrs." (if there's a match) or is missing (`NA`) if there's not a match.

Regular expressions

For this first example, we were looking for an exact string ("Mrs"). However, you can use patterns that match a particular pattern, but not an exact string. For example, we could expand the regular expression to find "Mr." or "Mrs.":

```
titanic_train %>%
  mutate(title = str_extract(Name, "Mr[s]*\\.")) %>%
  select(Name, title) %>%
  slice(1:3)
```

```
##                                                 Name title
## 1                             Braund, Mr. Owen Harris   Mr.
## 2 Cumings, Mrs. John Bradley (Florence Briggs Thayer)  Mrs.
## 3                              Heikkinen, Miss. Laina  <NA>
```

This pattern uses `[s]*` to match zero or more "s"s at this spot in the pattern.

Regular expressions

In the previous code, we found "Mr." and "Mrs.", but missed "Miss.". We could tweak the pattern again to try to capture that, as well. For all three, we have the pattern that it starts with "M", has some lowercase letters, and then ends with ".".

```
titanic_train %>%
  mutate(title = str_extract(Name, "M[a-z]+\\.")) %>%
  select(Name, title) %>%
  slice(1:3)
```

```
##                                                 Name title
## 1                             Braund, Mr. Owen Harris   Mr.
## 2 Cumings, Mrs. John Bradley (Florence Briggs Thayer)  Mrs.
## 3                              Heikkinen, Miss. Laina Miss.
```

Regular expressions

Sometimes, you want to match a pattern, but then only subset a part of it. For example, each passenger seems to have a title ("Mr.", "Mrs.", etc.) that comes after "," and before ".". We can use this pattern to find the title, but then we get some extra stuff with the match:

```
titanic_train %>%
  mutate(title = str_extract(Name, ", [A-Z][a-z]*\\.")) %>%
  select(title) %>%
  slice(1:3)
```

```
##     title
## 1   , Mr.
## 2  , Mrs.
## 3 , Miss.
```

Regular expressions

We are getting things like ", Mr. ", when we really want "Mr". We can use the `str_match` function to do this. We group what we want to extract from the pattern in parentheses, and then the function returns a matrix. The first column is the full pattern match, and each following column gives just what matches within the groups.

```
head(str_match(titanic_train$Name,
               pattern = ", ([A-Z][a-z]*)\\."))
```

```
##      [,1]      [,2]
## [1,] ", Mr."   "Mr"
## [2,] ", Mrs."  "Mrs"
## [3,] ", Miss." "Miss"
## [4,] ", Mrs."  "Mrs"
## [5,] ", Mr."   "Mr"
## [6,] ", Mr."   "Mr"
```

Regular expressions

To get just the title, then, we can run:

```
titanic_train %>%
  mutate(title =
           str_match(Name, ", ([A-Z][a-z]*)\\.")[ , 2]) %>%
  select(Name, title) %>%
  slice(1:3)
```

```
##                                                 Name title
## 1                             Braund, Mr. Owen Harris    Mr
## 2 Cumings, Mrs. John Bradley (Florence Briggs Thayer)   Mrs
## 3                              Heikkinen, Miss. Laina  Miss
```

The `[ , 2]` pulls out just the second column from the matrix returned by `str_match`.

Regular expressions

Here are some of the most common titles:

```
titanic_train %>%
  mutate(title =
           str_match(Name, ", ([A-Z][a-z]*)\\.")[ , 2]) %>%
  group_by(title) %>% summarize(n = n()) %>%
  arrange(desc(n)) %>% slice(1:5)
```

```
## `summarise()` ungrouping output (override with `.groups` argu
```

```
## # A tibble: 5 x 2
##   title      n
##   <chr>  <int>
## 1 Mr       517
## 2 Miss     182
## 3 Mrs      125
## 4 Master    40
## 5 Dr         7
```

Regular expressions

The following slides have a few other examples of regular expressions in action with this dataset.

Get just names that start with ("^") the letter "A":

```
titanic_train %>%
  filter(str_detect(Name, "^A")) %>%
  select(Name) %>%
  slice(1:3)
```

```
##                                                  Name
## 1                             Allen, Mr. William Henry
## 2                           Andersson, Mr. Anders Johan
## 3 Asplund, Mrs. Carl Oscar (Selma Augusta Emilia Johansson)
```

Regular expressions

Get names with "II" or "III" ({2,} says to match at least two times):

```
titanic_train %>%
  filter(str_detect(Name, "I{2,}")) %>%
  select(Name) %>%
  slice(1:3)
```

```
##                                          Name
## 1  Carter, Master. William Thornton II
## 2 Roebling, Mr. Washington Augustus II
```

Regular expressions

Get names with "Andersen" or "Anderson" (alternatives in square brackets):

```
titanic_train %>%
  filter(str_detect(Name, "Anders[eo]n")) %>%
  select(Name)
```

```
##                                                  Name
## 1 Andersen-Jensen, Miss. Carla Christine Nielsine
## 2                            Anderson, Mr. Harry
## 3                     Walker, Mr. William Anderson
## 4                      Olsvigen, Mr. Thor Anderson
## 5      Soholt, Mr. Peter Andreas Lauritz Andersen
```

Regular expressions

Get names that start with ("^" outside of brackets) the letters "A" and "B":

```
titanic_train %>%
  filter(str_detect(Name, "^[AB]")) %>%
  select(Name) %>%
  slice(1:3)
```

```
##                          Name
## 1  Braund, Mr. Owen Harris
## 2 Allen, Mr. William Henry
## 3 Bonnell, Miss. Elizabeth
```

Regular expressions

Get names that end with ("$") the letter "b" (either lowercase or uppercase):

```
titanic_train %>%
  filter(str_detect(Name, "[bB]$")) %>%
  select(Name)
```

```
##                          Name
## 1   Emir, Mr. Farred Chehab
## 2 Goldschmidt, Mr. George B
## 3           Cook, Mr. Jacob
## 4          Pasic, Mr. Jakob
```

Regular expressions

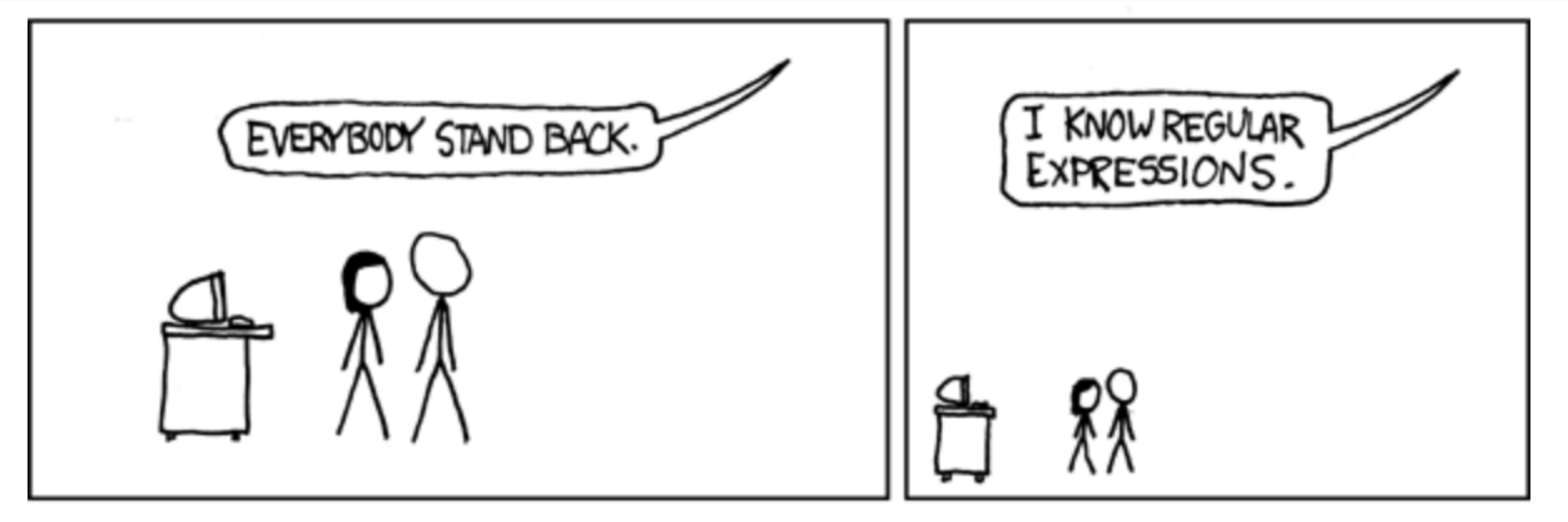
There is a family of older, base R functions called `grep` that does something very similar.

You may see these functions in example code.

Regular expressions

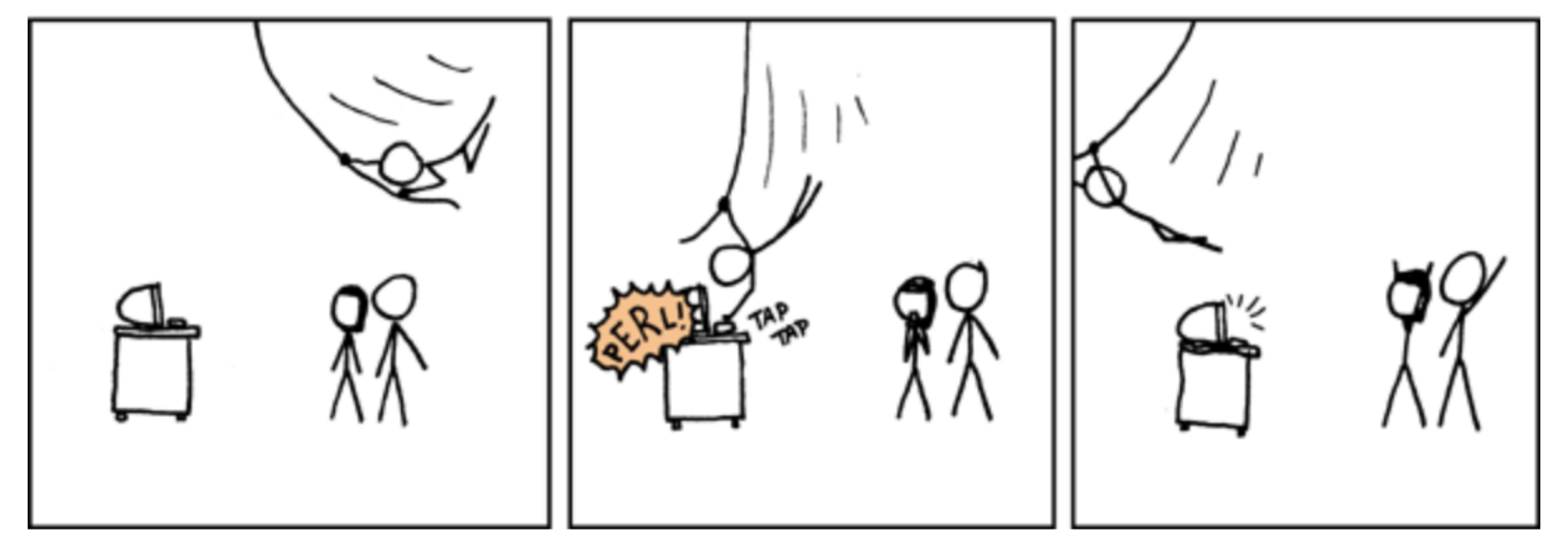
Souce: xkcd

Regular expressions



Souce: xkcd

Regular expressions



Souce: xkcd

Regular expressions

For more on these patterns, see:

- Help file for the `stringi-search-regex` function in the `stringi` package (which should install when you install `stringr`)
- Chapter 14 of R For Data Science
- http://gskinner.com/RegExr: Interactive tool for helping you build regular expression pattern strings