Data Cleaning Part 2

Data Wrangling in R

Data Cleaning Part 2

Reading in again

Now we have a chance to keep but clean these values!

```
ufo <- read_delim("../data/ufo/ufo_data_complete.txt",
                  col_types = cols("duration (seconds)" = col_character()))
```

```
New names:
• `` -> `...12`
```

```
ufo %>% glimpse()
```

```
Rows: 88,875
Columns: 12
$ datetime                [3m [38;5;246m<chr> [39m [23m "10/10/1949 20:30", "10/
$ city                    [3m [38;5;246m<chr> [39m [23m "san marcos", "lackland
$ state                   [3m [38;5;246m<chr> [39m [23m "tx", "tx", NA, "tx", "k
$ country                 [3m [38;5;246m<chr> [39m [23m "us", NA, "gb", "us", "u
$ shape                   [3m [38;5;246m<chr> [39m [23m "cylinder", "light", "ci
$ `duration (seconds)`    [3m [38;5;246m<chr> [39m [23m "2700", "7200", "20", "2
$ `duration (hours/min)`  [3m [38;5;246m<chr> [39m [23m "45 minutes", "1-2 hrs",
$ comments                [3m [38;5;246m<chr> [39m [23m "This event took place i
$ `date posted`           [3m [38;5;246m<chr> [39m [23m "4/27/2004", "12/16/2005
$ latitude                [3m [38;5;246m<chr> [39m [23m "29.8830556", "29.38421"
$ longitude               [3m [38;5;246m<dbl> [39m [23m -97.941111, -98.581082,
$ ...12                   [3m [38;5;246m<dbl> [39m [23m NA, NA, NA, NA, NA, NA,
```

Clean names with the `clean_names()` function from the `janitor` package

```
colnames(ufo)
```

```
 [1] "datetime"            "city"                "state"                "cou
 [5] "shape"               "duration (seconds)"  "duration (hours/min)" "com
 [9] "date posted"         "latitude"            "longitude"            "...
```

```
ufo = clean_names(ufo)
colnames(ufo)
```

```
 [1] "datetime"          "city"              "state"             "country"
 [5] "shape"             "duration_seconds"  "duration_hours_min" "comments"
 [9] "date_posted"       "latitude"          "longitude"         "x12"
```

The `stringr` package

Like `dplyr`, the `stringr` package:

- Makes some things more intuitive
- Is different than base R
- Is used on forums for answers
- Has a standard format for most functions: `str_`
    - the first argument is a string like first argument is a `data.frame` in `dplyr`

str_remove

Now let's fix our ufo data and remove those pesky backticks in the `duration_seconds` variable.

```
ufo <- ufo %>% mutate(duration_seconds = str_remove(string = duration_seconds,
                                                    pattern = "`"))
ufo <- ufo %>% mutate(duration_seconds = as.numeric(duration_seconds))
ufo[(ufo$row-1),]
```

```
Warning: Unknown or uninitialised column: `row`.
```

```
# A tibble: 0 × 12
# ℹ 12 variables: datetime <chr>, city <chr>, state <chr>, country <chr>, shap
#   duration_seconds <dbl>, duration_hours_min <chr>, comments <chr>, date_pos
#   latitude <chr>, longitude <dbl>, x12 <dbl>
```

Paste can add things back to variables

```
head(Orange)
```

```
  Tree  age circumference
1    1  118            30
2    1  484            58
3    1  664            87
4    1 1004           115
5    1 1231           120
6    1 1372           142
```

```
Orange %>% mutate(Tree = paste(Tree, "Tree", sep = "_"))
```

```
     Tree  age circumference
1  1_Tree  118            30
2  1_Tree  484            58
3  1_Tree  664            87
4  1_Tree 1004           115
5  1_Tree 1231           120
6  1_Tree 1372           142
7  1_Tree 1582           145
8  2_Tree  118            33
9  2_Tree  484            69
10 2_Tree  664           111
11 2_Tree 1004           156
12 2_Tree 1231           172
13 2_Tree 1372           203
14 2_Tree 1582           203
15 3_Tree  118            30
```

Paste0 doesn't need a separator

```
head(Orange)
```

```
  Tree  age circumference
1    1  118            30
2    1  484            58
3    1  664            87
4    1 1004           115
5    1 1231           120
6    1 1372           142
```

```
Orange %>% mutate(Tree = paste0(Tree, "Tree"))
```

```
    Tree  age circumference
1  1Tree  118            30
2  1Tree  484            58
3  1Tree  664            87
4  1Tree 1004           115
5  1Tree 1231           120
6  1Tree 1372           142
7  1Tree 1582           145
8  2Tree  118            33
9  2Tree  484            69
10 2Tree  664           111
11 2Tree 1004           156
12 2Tree 1231           172
13 2Tree 1372           203
14 2Tree 1582           203
15 3Tree  118            30
```

Substringing

`stringr`

- `str_sub(x, start, end)` - substrings from position start to position end
- `str_split(string, pattern)` - splits strings up - returns list! [we'll revisit in "Functional Programming"]

Substringing

Examples:

```
str_sub("I like friesian horses", 8,12)
```

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

```
#123456789101112
#I like fries
str_sub(c("Site A", "Site B", "Site C"), 6,6)
```

```
[1] "A" "B" "C"
```

Splitting/Find/Replace and Regular Expressions

- R can do much more than find exact matches for a whole string
- Like Perl and other languages, it can use regular expressions.
- What are regular expressions?
  - Ways to search for specific strings
  - Can be very complicated or simple
  - Highly Useful - think “Find” on steroids

A bit on Regular Expressions

- http://www.regular-expressions.info/reference.html
- They can use to match a large number of strings in one statement
- `.` matches any single character
- `*` means repeat as many (even if 0) more times the last character
- `?` makes the last thing optional
- `^` matches start of vector `^a` - starts with “a”
- `$` matches end of vector `b$` - ends with “b”

'Find' functions: `stringr`

`str_detect`, `str_subset`, `str_replace`, and `str_replace_all` search for matches to argument pattern within each element of a character vector: they differ in the format of and amount of detail in the results.

- `str_detect` - returns `TRUE` if `pattern` is found
- `str_subset` - returns only the strings which pattern were detected
  - convenient wrapper around `x[str_detect(x, pattern)]`
- `str_extract` - returns only strings which pattern were detected, but ONLY the pattern
- `str_replace` - replaces `pattern` with `replacement` the first time
- `str_replace_all` - replaces `pattern` with `replacement` as many times matched

'Find' functions: Finding Indices

These are the indices where the pattern match occurs:

```
str_detect(ufo$comments, "two aliens") %>% head()
```

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

```
str_detect(ufo$comments, "two aliens") %>% table()
```

```
.
FALSE  TRUE
88747     2
```

```
which(str_detect(ufo$comments, "two aliens"))
```

```
[1]  1730 61724
```

'Find' functions: Finding Logicals

`filter()` using `str_detect()` gives a tibble:

```
filter(ufo, str_detect(comments, "two aliens"))
```

```
# A tibble: 2 × 12
  datetime     city  state country shape duration_seconds duration_hours_min comments date_posted
  <chr>        <chr> <chr> <chr>   <chr>            <dbl> <chr>              <chr>    <chr>
1 10/14/2006 … yuma  va    us      form…              300 5 minutes          ((HOAX?… 4/27/2007
2 7/1/2007 23… nort… ct    <NA>    unkn…               60 1 minute           Witness… 10/19/2011
# ℹ 3 more variables: latitude <chr>, longitude <dbl>, x12 <dbl>
```

```
filter(ufo, str_detect(comments, "two aliens")) %>% select(comments)
```

```
# A tibble: 2 × 1
  comments
  <chr>
1 ((HOAX??))  two aliens appeared from a bright light to peacefully investigate the surroundings…
2 Witnessed two aliens walking along baseball field fence.
```

'Find' functions: `str_subset()` is easier

`str_subset()` gives the values that match the pattern:

```
str_subset(ufo$comments, "two aliens")
```

```
[1] "((HOAX??))  two aliens appeared from a bright light to peacefully investigate the surroundings in the woods"
[2] "Witnessed two aliens walking along baseball field fence."
```

Showing difference in `str_extract`

`str_extract` extracts just the matched string

```
ss = str_extract(ufo$comments, "two aliens")
head(ss)
```

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

```
ss[ !is.na(ss)]
```

```
[1] "two aliens" "two aliens"
```

- Look for any comment that starts with “aliens”

```
str_subset(ufo$comments, "^aliens.*")
```

```
[1] "aliens speak german???" "aliens exist"           "aliens in srilanka"
```

Using Regular Expressions

That contains space then ship maybe with stuff in between

```
str_subset(ufo$comments, "space.?ship") %>% head(4) # gets "spaceship" or "space ship" or...
```

```
[1] "I saw the cylinder shaped looked like a spaceship hovring above the east side of the Air Force base. Saw it for
[2] "description of a spaceship spotted over Birmingham Alabama in 1967."
[3] "A space ship was descending to the ground"
[4] "On Monday october 3&#44 2005&#44 I spotted two spaceships in the sky.  The first spotted ship was what seemed t
```

```
str_subset(ufo$comments, "space.ship") %>% head(4) # no "spaceship" must have character in bw
```

```
[1] "A space ship was descending to the ground"
[2] "I saw a Silver space ship rising into the early morning sky over Houston&#44 Texas."
[3] "Saw a space ship hanging over the southern (Manzano) portion of the Sandia Mountains on evening. It was brightl
[4] "saw space ship for 5 min&#33 Got scared crapless&#33&#33&#33&#33&#33&#33&#33&#33&#33&#33&#33&#33&#33&#33&#3
```

str_replace()

Let's say we wanted to make the time information more consistent. Using `case_when()` would be very tedious and error-prone!

We can use `str_replace()` to do so.

```
head(ufo$duration_hours_min, 8)
```

```
[1] "45 minutes"   "1-2 hrs"      "20 seconds"   "1/2 hour"     "15 minutes"
[7] "about 3 mins" "20 minutes"
```

```
ufo %>% mutate(duration_hours_min =
                 str_replace(string = duration_hours_min,
                             pattern = "minutes",
                             replacement ="mins")) %>%
  pull(duration_hours_min) %>%
  head(8)
```

```
[1] "45 mins"      "1-2 hrs"      "20 seconds"   "1/2 hour"     "15 mins"
[7] "about 3 mins" "20 mins"
```

Dates and times

The [lubridate](https://lubridate.tidyverse.org/) package is amazing, there's no reason to use anything else.

```
library(lubridate)#need to load this one!
head(ufo$datetime)
```

```
[1] "10/10/1949 20:30" "10/10/1949 21:00" "10/10/1955 17:00" "10/10/1956 21:00"
[5] "10/10/1960 20:00" "10/10/1961 19:00"
```

```
ufo$date_posted = mdy(ufo$date_posted)
```

```
Warning: 194 failed to parse.
```

```
head(ufo$date_posted)
```

```
[1] "2004-04-27" "2005-12-16" "2008-01-21" "2004-01-17" "2004-01-22" "2007-04-27"
```