Data Cleaning Part 2

Data Wrangling in R

Data Cleaning Part 2

Example of Cleaning: more complicated

For example, let’s say we have a variable about treatment or control conditions coded as treatment, T, treat, Treat, C, Cont, cont, cOnt, Control, and control. Using Excel to find all of these would be a matter of filtering and changing all by hand or using if statements.

Sometimes though, it’s not so simple. That’s where functions that find patterns come to be very useful.

Take a look at the data

```
count(data_gen, status)
```

```
# A tibble: 11 × 2
   status        n
   <chr>     <int>
 1 C            81
 2 Cont         90
 3 Control      91
 4 T            91
 5 Traet       105
 6 Treat       100
 7 cOnt         79
 8 cont         83
 9 control      98
10 treat        86
11 treatment    96
```

Example of Cleaning: more complicated

In R, you could use `case_when()`:

```
#case_when way:
data_gen <-data_gen %>% mutate(status =
            case_when(status
          %in% c("C", "cont", "cOnt", "Cont", "control", "Control")
                                ~ "Control",
                           .default =  status))
count(data_gen, status)
```

```
# A tibble: 6 × 2
  status        n
  <chr>     <int>
1 Control     522
2 T            91
3 Traet       105
4 Treat       100
5 treat        86
6 treatment    96
```

Oh dear! This only fixes some values! It is difficult to notice values like `"Traet"`.

String functions

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`

Useful String Functions

Useful String functions from base R and `stringr`

- `toupper()`, `tolower()` - uppercase or lowercase your data
- `str_sentence()` - uppercase just the first character (in the `stringr` package)
- `paste()` - paste strings together with a space
- `paste0` - paste strings together with no space as default
- `str_trim()` (in the `stringr` package) or `trimws` in base
    - will trim whitespace
- `nchar` - get the number of characters in a string

recoding with `str_to_sentence()`

```
#case_when way:
data_gen <-data_gen %>%
                mutate(status = str_to_sentence(status))
count(data_gen, status)
```

```
# A tibble: 5 × 2
  status        n
  <chr>     <int>
1 Control     522
2 T            91
3 Traet       105
4 Treat       186
5 Treatment    96
```

recoding with **`str_to_sentence()`**

```r
#case_when way:
data_gen <-data_gen %>%
                mutate(status = str_to_sentence(status)) %>%
                mutate(status =
                      case_when(status %in%
                    c("Treatment", "T", "Treat", "Traet", "Treat")
                                ~ "Treatment",
                           .default = status))
count(data_gen, status)
```

```
# A tibble: 2 × 2
  status        n
  <chr>     <int>
1 Control     522
2 Treatment   478
```

OK, now we are getting somewhere!

Reading in again

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

```
ufo <-read_csv(
  "https://sisbid.github.io/Data-Wrangling/data/ufo/ufo_data_complete.csv",
  col_types = cols(`duration (seconds)` = "c"))
```

```
Warning: One or more parsing issues, call `problems()` on your data frame for
  dat <- vroom(...)
  problems(dat)
```

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

```
colnames(ufo)
```

```
 [1] "datetime"             "city"                 "state"
 [4] "country"              "shape"                "duration (seconds)"
 [7] "duration (hours/min)" "comments"             "date posted"
[10] "latitude"             "longitude"
```

```
ufo_clean <- clean_names(ufo)
colnames(ufo_clean)
```

```
 [1] "datetime"           "city"               "state"
 [4] "country"            "shape"              "duration_seconds"
 [7] "duration_hours_min" "comments"           "date_posted"
[10] "latitude"           "longitude"
```

str_detect and filter

Now let's fix our ufo data and remove those pesky backticks in the `duration_seconds` variable. First let's find them with `str_detect`.

```
ufo_clean %>%
  filter(str_detect(
    string = duration_seconds,
    pattern = "`"))
```

```
# A tibble: 3 × 11
  datetime        city   state country shape duration_seconds duration_hours_
  <chr>           <chr>  <chr> <chr>   <chr> <chr>            <chr>
1 2/2/2000 19:33  bouse  az    us      <NA>  2`               each a few seco
2 4/10/2005 22:52 santa… ca    us      <NA>  8`               eight seconds
3 7/21/2006 13:00 ibagu… <NA>  <NA>    circ… 0.5`             1/2 segundo
# ℹ 4 more variables: comments <chr>, date_posted <chr>, latitude <chr>,
#   longitude <chr>
```

str_remove

```
ufo_clean  <- ufo_clean %>%
  mutate(duration_seconds =
           str_remove(string = duration_seconds,
                     pattern = "`"))
```

Lets also mutate to be as.numeric again

```
ufo_clean  <- ufo_clean %>%
  mutate(duration_seconds = as.numeric(duration_seconds))

glimpse(ufo_clean)
```

```
Rows: 88,875
Columns: 11
$ datetime          <chr> "10/10/1949 20:30", "10/10/1949 21:00", "10/10/195
$ city              <chr> "san marcos", "lackland afb", "chester (uk/england
$ state             <chr> "tx", "tx", NA, "tx", "hi", "tn", NA, "ct", "al", "
$ country           <chr> "us", NA, "gb", "us", "us", "us", "gb", "us", "us",
$ shape             <chr> "cylinder", "light", "circle", "circle", "light", "
$ duration_seconds  <dbl> 2700, 7200, 20, 20, 900, 300, 180, 1200, 180, 120,
$ duration_hours_min <chr> "45 minutes", "1-2 hrs", "20 seconds", "1/2 hour",
$ comments          <chr> "This event took place in early fall around 1949-50
$ date_posted       <chr> "4/27/2004", "12/16/2005", "1/21/2008", "1/17/2004"
$ latitude          <chr> "29.8830556", "29.38421", "53.2", "28.9783333", "2
$ longitude         <chr> "-97.9411111", "-98.581082", "-2.916667", "-96.645
```

Substringing

`stringr`

- `str_sub(x, start, end)` - substrings from position start to position end

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 a pattern optional (i.e. it matches 0 or 1 times)
- `^` 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 where the pattern were detected
- `str_extract` - returns only the pattern that was detected
- `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:

```
ufo_clean %>%
  filter(str_detect(comments, "two aliens")) %>%
  head()
```

```
# A tibble: 2 × 11
  datetime         city  state country shape duration_seconds duration_hours_
  <chr>            <chr> <chr> <chr>   <chr>            <dbl> <chr>
1 10/14/2006 02:00 yuma  va    us      form…              300 5 minutes
2 7/1/2007 23:00   nort… ct    <NA>    unkn…               60 1 minute
# ℹ 4 more variables: comments <chr>, date_posted <chr>, latitude <chr>,
#   longitude <chr>
```

To Take a look at comments… need to select it first

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

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

‘Find’ functions: `str_subset()` is easier

`str_subset()` gives the values that match the pattern: (or if we used negate = TRUE it would find the opposite)

```
ufo_clean %>% pull(comments) %>%
  str_subset( "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

```
ufo_clean %>%
  mutate(aliens  = str_extract(comments, "two aliens")) %>%
  count(aliens)
```

```
# A tibble: 2 × 2
  aliens          n
  <chr>       <int>
1 two aliens      2
2 <NA>        88873
```

- Look for any comment that starts with “aliens”

```
ufo_clean %>% pull(comments) %>%str_subset( "^aliens")
```

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

Using Regular Expressions

That contains space then ship maybe with stuff in between

```
ufo_clean %>% pull(comments) %>%
  str_subset("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 fo
[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
```

```
ufo_clean %>% pull(comments) %>%
  str_subset("space.ship") %>% head(4) # no "spaceship" must have character in between
```

```
[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 bright
[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&#33&#
```

time information

```
pull(ufo_clean, duration_hours_min) %>% head(n = 20)
```

```
 [1] "45 minutes"      "1-2 hrs"         "20 seconds"      "1/2 hour"       
 [5] "15 minutes"      "5 minutes"       "about 3 mins"    "20 minutes"     
 [9] "3  minutes"      "several minutes" "5 min."          "3 minutes"      
[13] "30 min."         "3 minutes"       "30 seconds"      "20minutes"      
[17] "2 minutes"       "20-30 min"       "20 sec."         "2 min"          
```

str_replace()

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

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

```
ufo_clean %>% 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"
[6] "5 mins"       "about 3 mins" "20 mins"
```

Separating columns

Better yet, you might notice that this data isn’t tidy- there are more than two entries for each value - amount of time and unit. We could separate this using `separate()` from the `tidyr` package.

```
ufo_clean %>% separate(duration_hours_min,
                 into = c("duration_amount", "duration_unit"),
                 sep = " ") %>%
  select(duration_amount, duration_unit) %>% head()
```

```
# A tibble: 6 × 2
  duration_amount duration_unit
  <chr>           <chr>
1 45              minutes
2 1-2             hrs
3 20              seconds
4 1/2             hour
5 15              minutes
6 5               minutes
```

As you can see there is still plenty of cleaning to do!

more seperating

```
ufo_clean <- ufo_clean %>% separate(datetime,
                 into = c("date", "time"),
                 sep = " ")
ufo_clean %>% select(date, time) %>% head()
```

```
# A tibble: 6 × 2
  date       time
  <chr>      <chr>
1 10/10/1949 20:30
2 10/10/1949 21:00
3 10/10/1955 17:00
4 10/10/1956 21:00
5 10/10/1960 20:00
6 10/10/1961 19:00
```

Dates and times

The `[lubridate]`(https://lubridate.tidyverse.org/) package is amazing for dates. Most important functions are those that look like ymd or mdy etc. They specify how a date should be interpreted.

```
library(lubridate)#need to load this one!

ufo_clean <- ufo_clean %>% mutate(date = mdy(date))
head(ufo_clean)
```

```
# A tibble: 6 × 12
  date       time  city  state country shape duration_seconds duration_hours_min
  <date>     <chr> <chr> <chr> <chr>   <chr>            <dbl> <chr>
1 1949-10-10 20:30 san … tx    us      cyli…             2700 45 minutes
2 1949-10-10 21:00 lack… tx    <NA>    light             7200 1-2 hrs
3 1955-10-10 17:00 ches… <NA>  gb      circ…               20 20 seconds
4 1956-10-10 21:00 edna  tx    us      circ…               20 1/2 hour
5 1960-10-10 20:00 kane… hi    us      light              900 15 minutes
6 1961-10-10 19:00 bris… tn    us      sphe…              300 5 minutes
# ℹ 4 more variables: comments <chr>, date_posted <chr>, latitude <chr>,
#   longitude <chr>
```

str_*functions

```
str_detect(string = c("abcdd", "two"), pattern = "dd")
```

```
[1]  TRUE FALSE
```

```
str_subset(string = c("abcdd", "two"), pattern = "dd")
```

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

```
str_extract(string = c("abcdd", "two"), pattern = "dd")
```

```
[1] "dd" NA
```

```
str_sub(string = c("abcdd", "two"), start = 1, end = 3)
```

```
[1] "abc" "two"
```

Summary

- `stringr` package has lots of helpful functions that work on vectors or variables in a data frame
- `str_detect` helps find patterns
- `str_detect` and `filter` can help you filter data based on patterns within value
- `str_extract` helps extract a pattern
- `str_sub` extracts pieces of strings based on the position of the the characters
- `str_subset` gives the values that match a pattern
- `separate` can separate columns into two
- `^` indicates the start of a string
- `$` indicates the end of a string
- the `lubridate` package is useful for dates and times

Lab

https://sisbid.github.io/Data-Wrangling/09_Data_Cleaning/lab/data-cleaning-lab-part2.Rmd