Data Cleaning

Data Wrangling in R

Data Cleaning

In general, data cleaning is a process of investigating your data for inaccuracies, or recoding it in a way that makes it more manageable.

MOST IMPORTANT RULE - LOOK AT YOUR DATA!

Read in the UFO dataset

Read in data or download from: http://sisbid.github.io/Data-Wrangling/data/ufo/ufo_data_complete.csv.gz

```
ufo <- read_delim("../data/ufo/ufo_data_complete.txt")
```

```
New names:
• `` -> `...12`

Warning: One or more parsing issues, see `problems()` for details

Rows: 88875 Columns: 12
── Column specification ──────────────────────────────────────────
Delimiter: "\t"
chr (9): datetime, city, state, country, shape, duration (hours/min), comments
dbl (3): duration (seconds), longitude, ...12

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this messa
```

The “problems”

You saw warning messages when reading in this dataset. We can see these with the `problems()` function from `readr`.

```
p <-problems(ufo)
p
```

```
# A tibble: 3 × 5
    row   col expected actual file
  <int> <int> <chr>    <chr>  <chr>
1 30894     6 a double 2`     /Users/avahoffman/Dropbox/JHSPH/Data-Wrangling_S
2 39616     6 a double 8`     /Users/avahoffman/Dropbox/JHSPH/Data-Wrangling_S
3 65125     6 a double 0.5`   /Users/avahoffman/Dropbox/JHSPH/Data-Wrangling_S
```

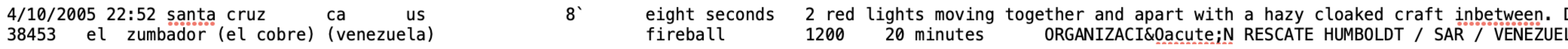
The “problems”

These all became NA values.

```
ufo[(p$row-1),] %>% glimpse()
```

```
Rows: 3
Columns: 12
$ datetime                 <chr> "2/2/2000 19:33", "4/10/2005 22:52", "7/21/2006
$ city                     <chr> "bouse", "santa cruz", "ibague (colombia)"
$ state                    <chr> "az", "ca", NA
$ country                  <chr> "us", "us", NA
$ shape                    <chr> NA, NA, "circle"
$ `duration (seconds)`     <dbl> NA, NA, NA
$ `duration (hours/min)`   <chr> "each a few seconds", "eight seconds", "1/2 sec
$ comments                 <chr> "Driving through Plomosa Pass towards Bouse Loo
$ `date posted`            <chr> "2/16/2000", "4/16/2005", "10/30/2006"
$ latitude                 <chr> "33.9325", "36.9741667", "4.440663"
$ longitude                <dbl> -114.00500, -122.02972, -75.24414
$ ...12                    <dbl> NA, NA, NA
```

The “problems”



```
4/10/2005 22:52 santa cruz      ca      us              8`      eight seconds   2 red lights moving together and apart with a hazy cloaked craft inbetween. D
38453   el  zumbador (el cobre) (venezuela)                     fireball        1200    20 minutes      ORGANIZACI&Oacute;N RESCATE HUMBOLDT / SAR / VENEZUEL
```

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[(p$row-1),] %>% glimpse()
```

```
Rows: 3
Columns: 12
$ datetime                 <chr> "2/2/2000 19:33", "4/10/2005 22:52", "7/21/2006
$ city                     <chr> "bouse", "santa cruz", "ibague (colombia)"
$ state                    <chr> "az", "ca", NA
$ country                  <chr> "us", "us", NA
$ shape                    <chr> NA, NA, "circle"
$ `duration (seconds)`     <chr> "2`", "8`", "0.5`"
$ `duration (hours/min)`   <chr> "each a few seconds", "eight seconds", "1/2 sec
$ comments                 <chr> "Driving through Plomosa Pass towards Bouse Loo
$ `date posted`            <chr> "2/16/2000", "4/16/2005", "10/30/2006"
$ latitude                 <chr> "33.9325", "36.9741667", "4.440663"
$ longitude                <dbl> -114.00500, -122.02972, -75.24414
$ ...12                    <dbl> NA, NA, NA
```

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

```
colnames(ufo)
```

```
 [1] "datetime"             "city"                 "state"                "cou
 [6] "duration (seconds)"   "duration (hours/min)" "comments"             "dat
[11] "longitude"            "...12"
```

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

```
 [1] "datetime"           "city"               "state"              "country"
 [6] "duration_seconds"   "duration_hours_min" "comments"           "date_post
[11] "longitude"          "x12"
```

Recoding Variables

Example of Cleaning: more complicated

For example, let’s say gender was coded as Male, M, m, Female, F, f. 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.

```
table(gender)
```

```
gender
     F FeMAle FEMALE     Fm      M     Ma   mAle   Male   MaLe   MALE    Man
    80     88     76     87     99     76     84     83     79     93     84
```

Example of Cleaning: more complicated

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

```
#case_when way:
data_gen <-data_gen %>% mutate(gender =
                      case_when(gender %in% c("Male", "M", "m", "Man")
                                ~ "Male",
                           TRUE ~ gender))
head(data_gen)
```

```
# A tibble: 6 × 1
  gender
  <chr>
1 F
2 Fm
3 MaLe
4 MaLe
5 FeMAle
6 FEMALE
```

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

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()`

```r
#case_when way:
data_gen <-data_gen %>%
                mutate(gender = str_to_sentence(gender)) %>%
                mutate(gender =
                      case_when(gender %in% c("Male", "M", "m", "Man")
                                ~ "Male",
                           TRUE ~ gender))
head(data_gen)
```

```
# A tibble: 6 × 1
  gender
  <chr>
1 F
2 Fm
3 Male
4 Male
5 Female
6 Female
```

OK, now we are getting somewhere!

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[(p$row-1),]
```

```
# A tibble: 3 × 12
  datetime      city  state country shape duration_seconds duration_hours_… co
  <chr>         <chr> <chr> <chr>   <chr>            <dbl> <chr>            <c
1 2/2/2000 19:… bouse az    us      <NA>               2   each a few seco… Dr
2 4/10/2005 22… sant… ca    us      <NA>               8   eight seconds    2
3 7/21/2006 13… ibag… <NA>  <NA>    circ…              0.5 1/2 segundo      Vi
```

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_… comments date_posted latitude longitude
  <chr>         <chr> <chr> <chr>   <chr>            <dbl> <chr>            <chr>    <chr>       <chr>        <dbl>
1 10/14/2006 2… yuma  va    us      form…              300 5 minutes        ((HOAX?… 4/27/2007   36.615       -82.6
2 7/1/2007 23:… nort… ct    <NA>    unkn…               60 1 minute         Witness… 10/19/2011  41.9856…     -71.9
```

```
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 in the woods
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&#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"
[8] "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"
[8] "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" "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"
```