

# Data Cleaning Part 1

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](http://sisbid.github.io/Data-Wrangling/data/ufo/ufo_data_complete.csv.gz)

```
ufo <- read_delim(  
  "https://sisbid.github.io/Data-Wrangling/data/ufo/ufo_data  
  delim = ",")
```

Warning: One or more parsing issues, call `problems()` on y  
e.g.:

```
dat <- vroom(...)  
problems(dat)
```

Rows: 88875 Columns: 11

-- Column specification -----

Delimiter: ", "

chr (10): datetime, city, state, country, shape, duration

dbl (1): duration (seconds)

i Use `spec()` to retrieve the full column specification fo

i Specify the column types or set `check\_col\_types = FALSE`

## The “problems”

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

If we scroll through we can see some interesting notes.

```
p <-problems(ufo)
p %>% glimpse()
```

```
Rows: 200
```

```
Columns: 5
```

```
$ row      <int> 878, 1713, 1815, 2858, 3734, 4756, 5389, 5
$ col      <int> 12, 12, 12, 12, 12, 12, 12, 12, 12, 12
$ expected <chr> "11 columns", "11 columns", "11 columns",
$ actual   <chr> "12 columns", "12 columns", "12 columns",
$ file     <chr> "", "", "", "", "", "", "", "", "", ""
```

## Any unique problems?

```
count(p, expected, actual, col)
```

```
# A tibble: 5 x 4
```

	expected	actual	col	n
	<chr>	<chr>	<int>	<int>
1	11 columns	"12 columns"	12	196
2	a double	"0.5`"	6	1
3	a double	"2631600  "	6	1
4	a double	"2`"	6	1
5	a double	"8`"	6	1



## Taking a deeper look

The slice function can show us particular row numbers

```
p %>% filter(col != 12)
```

```
# A tibble: 4 x 5
```

	row	col	expected	actual	file
	<int>	<int>	<chr>	<chr>	<chr>
1	30894	6	a double	"2`"	""
2	39616	6	a double	"8`"	""
3	45691	6	a double	"2631600	" ""
4	65125	6	a double	"0.5`"	""

## Taking a deeper look

The slice function can show us particular row numbers

```
slice(ufo, 30894 -1) %>% glimpse()
```

```
Rows: 1
```

```
Columns: 11
```

```
$ datetime <chr> "2/2/2000 19:33"
```

```
$ city <chr> "bouse"
```

```
$ state <chr> "az"
```

```
$ country <chr> "us"
```

```
$ shape <chr> NA
```

```
$ `duration (seconds)` <dbl> NA
```

```
$ `duration (hours/min)` <chr> "each a few seconds"
```

```
$ comments <chr> "Driving through Plomosa Pass"
```

```
$ `date posted` <chr> "2/16/2000"
```

```
$ latitude <chr> "33.9325000"
```

```
$ longitude <chr> "-114.0050000"
```

```
knitr::include_graphics(here::here("images", "raw_ufo_col6"))
```



## Reading in again

Now we have a chance to keep but clean these values! We will read in `duration` (seconds) now as a character type. Use `?read_csv` to see documentation about special ways of reading in data.

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

Warning: One or more parsing issues, call ``problems()`` on your data frame, e.g.:

```
dat <- vroom(...)  
problems(dat)
```

## Look at the problems again

Looks like an extra column for these rows.

```
p <- problems(ufo)
count(p, expected, actual, col)
```

```
# A tibble: 1 x 4
  expected actual      col      n
  <chr>    <chr>    <int> <int>
1 11 columns 12 columns    12   196
```

Our previous problematic rows have values instead of NAs

```
slice(ufo, 30894 -1) %>% glimpse()
```

```
Rows: 1
Columns: 11
$ datetime      <chr> "2/2/2000 19:33"
$ city          <chr> "bouse"
$ state         <chr> "az"
$ country       <chr> "us"
```

## Drop the remaining shifted problematic rows for now

Multiply by negative one to drop the rows. Use the `slice` function to “select” those rows based on the index. Need to offset for `-1` because `problems()` gives us the index row based on the raw data, not the read in data (which has a header). We will multiply by negative one to select out those rows as well.

```
head(p, n = 2)
```

```
# A tibble: 2 x 5
  row   col expected      actual      file
<int> <int> <chr>      <chr>      <chr>
1   878    12 11 columns 12 columns ""
2  1713    12 11 columns 12 columns ""
```

```
(pull(p, row) -1) %>% head()
```

```
[1] 877 1712 1814 2857 3733 4755
```

```
((pull(p, row) -1) * -1) %>% head()
```

```
[1] 877 1712 1814 2857 3733 4755
```

## Checking

```
nrow(ufo) - nrow(ufo_clean)
```

```
[1] 196
```

```
count(p, expected, actual)
```

```
# A tibble: 1 x 3
```

	expected	actual	n
	<chr>	<chr>	<int>
1	11 columns	12 columns	196

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

```
colnames(ufo_clean)
```

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

```
ufo_clean <- clean_names(ufo_clean)
```

```
colnames(ufo_clean)
```

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

## Recoding Variables

## Exact Swaps - recode function

```
within mutate...  
recode(variable, value_old = value_new,  
        other_value_old = other_value_new)
```

```
ufo_clean %>%  
  mutate(country =  
    recode(country, gb = "Great Britain")) %>%  
  glimpse()
```

Rows: 88,679

Columns: 11

```
$ datetime      <chr> "10/10/1949 20:30", "10/10/1949  
$ city          <chr> "san marcos", "lackland afb", "c  
$ state         <chr> "tx", "tx", NA, "tx", "hi", "tn"  
$ country       <chr> "us", NA, "Great Britain", "us"  
$ shape         <chr> "cylinder", "light", "circle", "  
$ duration_seconds <chr> "2700", "7200", "20", "20", "900  
$ duration_hours_min <chr> "45 minutes", "1-2 hrs", "20 sec  
$ comments      <chr> "This event took place in early
```

## Exact Swaps - recode function

```
ufo_clean %>% mutate(country =  
  recode(country,  
    gb = "Great Britain",  
    us = "United States")) %>%  
  
  glimpse()
```

Rows: 88,679

Columns: 11

```
$ datetime      <chr> "10/10/1949 20:30", "10/10/1949  
$ city          <chr> "san marcos", "lackland afb", "  
$ state         <chr> "tx", "tx", NA, "tx", "hi", "tn"  
$ country       <chr> "United States", NA, "Great Brit  
$ shape         <chr> "cylinder", "light", "circle", "  
$ duration_seconds <chr> "2700", "7200", "20", "20", "900  
$ duration_hours_min <chr> "45 minutes", "1-2 hrs", "20 sec  
$ comments      <chr> "This event took place in early  
$ date_posted   <chr> "4/27/2004", "12/16/2005", "1/21  
$ latitude      <chr> "29.8830556", "29.38421", "53.21  
$ longitude     <chr> "-97.0411111", "-99.5810000", "
```



## How many countries?

```
ufo_clean %>% count(country)
```

```
# A tibble: 6 x 2
```

	country	n
	<chr>	<int>
1	au	593
2	ca	3266
3	de	112
4	gb	2050
5	us	70293
6	<NA>	12365

## case\_when() regions to create a new variable based on conditions of other variables

```
case_when(test ~ value if test is true,  
          test2 ~ vlue if test2 is true,  
          TRUE ~ value if all above tests are not true) # de
```

```
ufo_clean <- ufo_clean %>% mutate(  
  region = case_when(  
    country %in% c("us", "ca") ~ "North America",  
    country %in% c("de") ~ "Europe",  
    country %in% "gb" ~ "Great Britain",  
    TRUE ~ "Other"  
  ))
```

```
ufo_clean %>% select(country, region) %>% head()
```

```
# A tibble: 6 x 2
```

	country	region
	<chr>	<chr>
1	us	North America
2	<NA>	Other

## case\_when - another example

The TRUE value can also just be the original values.

```
ufo_clean %>% mutate(country = case_when(  
  country == "gb" ~ "Great Britain",  
  country == "us" ~ "United States",  
  country == "au" ~ "Australia",  
  country == "DE" ~ "Germany",  
  TRUE ~ country))%>%  
  
glimpse()
```

Rows: 88,679

Columns: 12

```
$ datetime      <chr> "10/10/1949 20:30", "10/10/1949  
$ city          <chr> "san marcos", "lackland afb", "c  
$ state         <chr> "tx", "tx", NA, "tx", "hi", "tn"  
$ country       <chr> "United States", NA, "Great Brit  
$ shape         <chr> "cylinder", "light", "circle", "  
$ duration_seconds <chr> "2700", "7200", "20", "20", "900  
$ duration_hours_min <chr> "45 minutes", "1-2 hrs", "20 se
```

## Summary

- ▶ `recode` makes exact swaps
- ▶ `case_when` can use conditionals, need to specify what value for if no conditions are met (can be the original value of a variable if we use the variable name).

# Lab

<https://sisbid.github.io/Data-Wrangling/labs/data-cleaning-lab.Rmd>

Extra slides

## Strange country values

Sometimes country is NA even though state is known. A conditional more flexible recoding would be helpful...

```
head(ufo_clean)
```

```
# A tibble: 6 x 12
  datetime          city state country shape duration_seconds
  <chr>             <chr> <chr> <chr>  <chr> <chr>
1 10/10/1949 20:30 san ~ tx    us    cyli~ 2700
2 10/10/1949 21:00 lack~ tx    <NA>  light 7200
3 10/10/1955 17:00 ches~ <NA>  gb    circ~ 20
4 10/10/1956 21:00 edna  tx    us    circ~ 20
5 10/10/1960 20:00 kane~ hi    us    light 900
6 10/10/1961 19:00 bris~ tn    us    sphe~ 300
# i 5 more variables: comments <chr>, date_posted <chr>, la
# longitude <chr>, region <chr>
```

## Deeper look

Looking at city... it seems like many of these are in fact in the US.

```
ufo_clean %>% filter(state == "tx") %>% count(country, state)
```

```
# A tibble: 2 x 3
  country state      n
  <chr>   <chr> <int>
1 us      tx      3742
2 <NA>    tx       299
```

```
ufo_clean %>% filter(state == "tx" & is.na(country)) %>% select(city)
```

```
# A tibble: 299 x 1
  city
  <chr>
1 lackland afb
2 mercedies
3 texas city/galveston
4 houston/tomball
5 bettendorf
```



## Checkin Utah as well

```
ufo_clean %>% filter(state == "ut") %>% count(country, state)
```

```
# A tibble: 2 x 3
```

	country	state	n
	<chr>	<chr>	<int>
1	us	ut	659
2	<NA>	ut	138

```
ufo_clean %>% filter(state == "ut" & is.na(country)) %>% summarise(city = ...)
```

```
# A tibble: 138 x 1
```

	city
	<chr>
1	canyonlands np
2	ogden/clinton
3	sandy
4	salt lake valley
5	sandy
6	duchenne

## Get US States

```
ufo_clean %>% filter(country == "us") %>%  
  count(state) %>%  
  pull(state)
```

```
[1] "ak" "al" "ar" "az" "ca" "co" "ct" "dc" "de" "fl" "ga"  
[16] "in" "ks" "ky" "la" "ma" "md" "me" "mi" "mn" "mo" "ms"  
[31] "nh" "nj" "nm" "nv" "ny" "oh" "ok" "or" "pa" "pr" "ri"  
[46] "ut" "va" "vt" "wa" "wi" "wv" "wy"
```

```
US_states <- ufo_clean %>%  
  filter(country == "us") %>%  
  count(state) %>%  
  pull(state)
```

## Get Canada States

```
ufo_clean %>% filter(country == "ca") %>%  
  count(state) %>%  
  pull(state)
```

```
[1] "ab" "bc" "mb" "nb" "nf" "ns" "nt" "on" "pe" "pq" "qc"  
[16] NA
```

```
CA_states <- ufo_clean %>%  
  filter(country == "ca") %>%  
  count(state) %>%  
  pull(state)
```

## Get Great Britain states

```
ufo_clean %>% filter(country == "gb") %>%  
  count(state) %>%  
  pull(state)
```

```
[1] "bc" "la" "ms" "nc" "ns" "nt" "ri" "sk" "tn" "wv" "yt"
```

```
GB_states <- ufo_clean %>%  
  filter(country == "gb") %>%  
  count(state) %>%  
  pull(state)
```

A small overlap with US states.

## Get DE states

```
ufo_clean %>% filter(country == "de") %>%  
  count(state) %>%  
  pull(state)
```

```
[1] NA
```

## Get AU states

```
ufo_clean %>% filter(country == "au") %>%  
  count(state) %>%  
  pull(state)
```

```
[1] "al" "dc" "nt" "oh" "sa" "wa" "yt" NA
```

```
AU_states <- ufo_clean %>%  
  filter(country == "au") %>%  
  count(state) %>%  
  pull(state)
```

Some overlap with US states.

## Get just unique

The `setdiff()` function can show us what is unique or different for the first of 2 listed sets.

```
numbers <-c(1,2,3)
letters <-c("a", "b", 3)
```

```
setdiff(numbers, letters)
```

```
[1] 1 2
```

```
setdiff(letters, numbers)
```

```
[1] "a" "b"
```

## Get just unique

```
US_states
```

```
[1] "ak" "al" "ar" "az" "ca" "co" "ct" "dc" "de" "fl" "ga"  
[16] "in" "ks" "ky" "la" "ma" "md" "me" "mi" "mn" "mo" "ms"  
[31] "nh" "nj" "nm" "nv" "ny" "oh" "ok" "or" "pa" "pr" "ri"  
[46] "ut" "va" "vt" "wa" "wi" "wv" "wy"
```

```
c(AU_states, GB_states, CA_states)
```

```
[1] "al" "dc" "nt" "oh" "sa" "wa" "yt" NA "bc" "la" "ms"  
[16] "sk" "tn" "wv" "yt" NA "ab" "bc" "mb" "nb" "nf" "ns"  
[31] "qc" "sa" "sk" "yk" "yt" NA
```

```
US_states <- setdiff(US_states, c(AU_states, GB_states, CA_  
US_states
```

```
[1] "ak" "ar" "az" "ca" "co" "ct" "de" "fl" "ga" "hi" "ia"  
[16] "ky" "ma" "md" "me" "mi" "mn" "mo" "mt" "nd" "ne" "nh"  
[31] "ok" "or" "pa" "pr" "sc" "sd" "tx" "ut" "va" "vt" "wi"
```



## Continued

```
AU_states <- setdiff(AU_states, c(US_states, GB_states, CA_
CA_states <- setdiff(CA_states, c(US_states, GB_states, AU_
GB_states <- setdiff(GB_states, c(US_states, AU_states, CA_
```

How often do rows have a value for country but not a value of "us"?

```
ufo_clean %>%  
  filter(country != "us" & !is.na(country)) %>%  
  count(country)
```

```
# A tibble: 4 x 2  
  country      n  
  <chr>    <int>  
1 au         593  
2 ca        3266  
3 de         112  
4 gb        2050
```

## more complicated case\_when

Let's make an assumption that if the state value is within the data as a state for a specific country, than it comes from that country for the sake of illustration.

```
ufo_clean <- ufo_clean %>% mutate(prob_country =  
  case_when((is.na(country) & state %in% c(US_states))  
            (is.na(country) & state %in% c(CA_states))  
            (is.na(country) & state %in% c(AU_states))  
            (is.na(country) & state %in% c(GB_states))  
            TRUE ~ country))
```

## results

```
count(ufo_clean, prob_country)
```

```
# A tibble: 10 x 2
```

	prob_country	n
	<chr>	<int>
1	Australia	694
2	Canada	536
3	Great Britain	5296
4	United States	5838
5	au	593
6	ca	3266
7	de	112
8	gb	2050
9	us	70293
10	<NA>	1

## results

Take a look at those NAs.

```
ufo_clean %>% filter(is.na(prob_country))
```

```
# A tibble: 1 x 13
```

```
  datetime          city  state country shape duration_seconds  
<chr>             <chr> <chr> <chr>   <chr> <chr>
```

```
1 10/25/1997 22:00 st. ~ vi    <NA>   light 8
```

```
# i 6 more variables: comments <chr>, date_posted <chr>, la
```

```
# longitude <chr>, region <chr>, prob_country <chr>
```

## We could confirm with city info and latitude and longitude

```
ufo_clean %>% filter(country == "de") %>%  
  pull(city)
```

```
[1] "berlin (germany)"  
[2] "berlin (germany)"  
[3] "obernheim (germany)"  
[4] "ottersberg (germany)"  
[5] "urbach (germany)"  
[6] "bremen (30 km south of) (germany)"  
[7] "sembach (germany)"  
[8] "magdeburg (germany)"  
[9] "neuruppin (germany)"  
[10] "lampertheim (germany)"  
[11] "ramstein (germany)"  
[12] "bremen (germany)"  
[13] "nurenborg (germany)"  
[14] "senftenberg (germany)"  
[15] "schwalmtal (germany)"  
[16] "wuppertal (germany)"
```

## Even more specific

```
ufo_clean <- ufo_clean %>% mutate(prob_country =  
  case_when(  
    (is.na(country) & state %in% c(US_states)) |  
    country == "us" ~ "United States",  
    (is.na(country) & state %in% c(CA_states)) |  
    country == "ca" ~ "Canada",  
    (is.na(country) & state %in% c(AU_states)) |  
    country == "au" ~ "Australia",  
    (is.na(country) & state %in% c(GB_states)) |  
    country == "gb" ~ "Great Britain",  
    country == "de" ~ "Germany",  
    TRUE ~ country))
```

We would want to confirm what we recoded with the cities and latitude and longitude, especially to deal with the overlaps in the state lists.

## Check counts

```
ufo_clean %>%  
  count(country, prob_country)
```

```
# A tibble: 10 x 3
```

	country	prob_country	n
	<chr>	<chr>	<int>
1	au	Australia	593
2	ca	Canada	3266
3	de	Germany	112
4	gb	Great Britain	2050
5	us	United States	70293
6	<NA>	Australia	694
7	<NA>	Canada	536
8	<NA>	Great Britain	5296
9	<NA>	United States	5838
10	<NA>	<NA>	1