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

This data is originally from kaggle: https://www.kaggle.com/datasets/NUFORC/ufo-sightings

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

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

Rows: 88875 Columns: 11
── Column specification ────────────────────────────────────────────────────────
Delimiter: ","
chr (10): datetime, city, state, country, shape, duration (hours/min), comme.
dbl  (1): duration (seconds)

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 mess
```

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, 5423, 5614, 5849, 60
$ col      <int> 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 1
$ expected <chr> "11 columns", "11 columns", "11 columns", "11 columns", "11 c
$ actual   <chr> "12 columns", "12 columns", "12 columns", "12 columns", "12 c
$ file     <chr> "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "
```

Any unique problems?

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

```
# A tibble: 5 × 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
```

The "problems"

```
colnames(ufo)
```

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

```
glimpse(ufo)
```

```
Rows: 88,875
Columns: 11
$ datetime                <chr> "10/10/1949 20:30", "10/10/1949 21:00", "10/10/
$ city                    <chr> "san marcos", "lackland afb", "chester (uk/eng
$ state                   <chr> "tx", "tx", NA, "tx", "hi", "tn", NA, "ct", "a
$ country                 <chr> "us", NA, "gb", "us", "us", "us", "gb", "us", "
$ shape                   <chr> "cylinder", "light", "circle", "circle", "ligh
$ `duration (seconds)`    <dbl> 2700, 7200, 20, 20, 900, 300, 180, 1200, 180, 1
$ `duration (hours/min)` <chr> "45 minutes", "1-2 hrs", "20 seconds", "1/2 hou
$ comments                <chr> "This event took place in early fall around 194
$ `date posted`           <chr> "4/27/2004", "12/16/2005", "1/21/2008", "1/17/2
$ latitude                <chr> "29.8830556", "29.38421", "53.2", "28.9783333",
$ longitude               <chr> "-97.9411111", "-98.581082", "-2.916667", "-96.
```

Taking a deeper look

The slice function can show us particular row numbers

```
p %>% filter(col == 6)
```

```
# A tibble: 4 × 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 towards Bouse Lo
$ `date posted`           <chr> "2/16/2000"
$ latitude                <chr> "33.9325000"
$ longitude               <chr> "-114.0050000"
```

Check the actual data

```
Carries-MacBook-Pro:ufo carriewright$ awk 'FNR>=30893 && FNR<=30895' ufo_data_complete.csv
2/2/2000 19:00,seattle (corner of mercer and queen anne avenues),wa,us,,30,30 seconds,"I heard a loud aircraft sound.Looked up and saw two very
large unidenified aircraft. flying very slow&#44 very low&#44and very close together",2/16/2000,47.6063889,-122.3308333
2/2/2000 19:33,bouse,az,us,,2`,each a few seconds,"Driving through Plomosa Pass towards Bouse Looking NE towards Prescott about 60 degrees up fr
om horizon myself&#44husband and daughter wit",2/16/2000,33.9325000,-114.0050000
2/2/2000 19:49,seattle,wa,us,other,120,aprox. 2 min,"Watched two flying objects headed NW&#44 very low&#44 very slow&#44 at junction of Duamish
River and West Seattle Bridge.",2/16/2000,47.6063889,-122.3308333
```

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_complete.csv"
ufo <-read_csv(url, col_types = cols(`duration (seconds)` = "c"))
```

```
Warning: One or more parsing issues, call `problems()` on your data frame for
  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 × 4
  expected   actual       col     n
  <chr>      <chr>      <int> <int>
1 11 columns 12 columns    12   196
```

Check one of our previous problematic rows:

```
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)`    <chr> "2`"
$ `duration (hours/min)` <chr> "each a few seconds"
$ comments                <chr> "Driving through Plomosa Pass towards Bouse Lo
$ `date posted`           <chr> "2/16/2000"
$ latitude                <chr> "33.9325000"
$ longitude               <chr> "-114.0050000"
```

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).

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

```
# A tibble: 2 × 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
```

```
ufo_clean <- ufo %>% slice(-(pull(p, row)-1))
```

Checking

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

```
[1] 196
```

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

```
# A tibble: 1 × 3
  expected   actual         n
  <chr>      <chr>      <int>
1 11 columns 12 columns   196
```

read in for the shifted rows

To keep these rows we would need to import again with different specifications. We could just get the shifted rows, rename the columns to be correct and the join this with the existing data if we wanted to try to keep all rows. We will learn about merging data later. But for now let's not worry about these rows.

```
ufo_colnames <-colnames(ufo)
ufo_colnames<-c(ufo_colnames, "empty") # add a new column name for 12th column

ufo2 <-
read.table("https://sisbid.github.io/Data-Wrangling/data/ufo/ufo_data_complete.csv",
                  header = TRUE, sep = ",", col.names = ufo_colnames, fill = TRUE, na = "",
           colClasses = c(rep("character",10), "numeric")) # based on what the values seem to be currently
```

```
Warning in
read.table("https://sisbid.github.io/Data-Wrangling/data/ufo/ufo_data_complete.csv",
: header and 'col.names' are of different lengths
```

```
ufo2 <-ufo2 %>% filter(empty == 0)
```

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

```
colnames(ufo_clean)
```

```
 [1] "datetime"             "city"                 "state"
 [4] "country"              "shape"                "duration (seconds)"
 [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"
```

Duplicates

Distinct values

The `distinct()` function can help us remove duplicated rows.

```
dup_df
```

```
  ID   Name value
1  1      R    25
2  3      R    60
3  2 Python    30
4  2 Python    30
```

```
distinct(dup_df) # to get just distinct rows
```

```
  ID   Name value
1  1      R    25
2  3      R    60
3  2 Python    30
```

Aggregate values

`aggregate` helps us aggregate the data using mean or sum

```
dup_df %>% aggregate(value ~ Name, mean)
```

```
    Name value
1 Python  30.0
2      R  42.5
```

```
dup_df %>% aggregate(value ~ Name, sum)
```

```
    Name value
1 Python    60
2      R    85
```

Recoding Variables

Exact Swaps - **case_when** function

```
case_when(test ~ value if test is true,
      .default ~ value if all above tests are not true) # defaults to NA
```

```
ufo_clean %>%
  mutate(country =
           case_when(country == "gb" ~ "Great Britain"),
         .default = country) %>%
  glimpse()
```

```
Rows: 88,679
Columns: 12
$ 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> NA, NA, "Great Britain", NA, NA, NA, "Great Britai
$ shape             <chr> "cylinder", "light", "circle", "circle", "light", "
$ duration_seconds  <chr> "2700", "7200", "20", "20", "900", "300", "180", "1
$ 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", "21
$ longitude         <chr> "-97.9411111", "-98.581082", "-2.916667", "-96.6458
$ .default          <chr> NA, NA, "Great Britain", NA, NA, NA, "Great Britai
```

case_when note

.default may look like TRUE ~

```
ufo_clean %>%
  mutate(country =
           case_when(country == "gb" ~ "Great Britain",
         TRUE ~ country)) %>%
  glimpse()
```

```
Rows: 88,679
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, "Great Britain", "us", "us", "us", "Grea
$ shape             <chr> "cylinder", "light", "circle", "circle", "light", "
$ duration_seconds  <chr> "2700", "7200", "20", "20", "900", "300", "180", "1
$ 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", "21
$ longitude         <chr> "-97.9411111", "-98.581082", "-2.916667", "-96.6458
```

How many countries?

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

```
# A tibble: 6 × 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,
         .default ~ value if all above tests are not true) # defaults to NA
```

The `.default` 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",
                              .default = country))%>%
  glimpse()
```

```
Rows: 88,679
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> "United States", NA, "Great Britain", "United State
$ shape              <chr> "cylinder", "light", "circle", "circle", "light", "
$ duration_seconds   <chr> "2700", "7200", "20", "20", "900", "300", "180", "1
$ 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"
```

`case_when` - another example

The `.default` value can also be a single different value.

```
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",
               .default = "Other"
             ))
ufo_clean %>% select(country, region) %>% head()
```

```
# A tibble: 6 × 2
  country region
  <chr>   <chr>
1 us      North America
2 <NA>    Other
3 gb      Great Britain
4 us      North America
5 us      North America
6 us      North America
```

Summary

- Sometimes need to read the data in iteratively and check for problems with `problems`
- Duplicated values can be reduced down to distinct rows using `distinct()`
- Duplicated values can ablso be reduced down by aggregating using mathematical functions like `mean` or `sum` using `aggregate`
- `case_when` can use conditionals, need to specify what value for if no conditions are met using `.defualt`(can be the original value of a variable if we use the variable name).
- `case_when` needs to be used inside `mutate` to modify or create new columns
- Note: you might see the `recode` is a function, which is like `case_when` but only makes exact swaps, (while `case_when` can do this and more!)

Lab

https://sisbid.github.io/Data-Wrangling/09_Data_Cleaning/lab/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 × 12
  datetime         city  state country shape duration_seconds duration_hours_r
  <chr>            <chr> <chr> <chr>   <chr> <chr>            <chr>
1 10/10/1949 20:30 san … tx    us      cyli… 2700             45 minutes
2 10/10/1949 21:00 lack… tx    <NA>    light 7200             1-2 hrs
3 10/10/1955 17:00 ches… <NA>  gb      circ… 20               20 seconds
4 10/10/1956 21:00 edna  tx    us      circ… 20               1/2 hour
5 10/10/1960 20:00 kane… hi    us      light 900              15 minutes
6 10/10/1961 19:00 bris… tn    us      sphe… 300              5 minutes
# ℹ 5 more variables: comments <chr>, date_posted <chr>, latitude <chr>,
#   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 × 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 × 1
   city
   <chr>
 1 lackland afb
 2 mercedies
 3 texas city/galveston
 4 houston/tomball
 5 bettendorf
 6 dallas/ft. worth (mansfield)
 7 halletsville
 8 gulf of mexico
 9 haltom
10 aubrey/frisco
# ℹ 289 more rows
```

Checkin Utah as well

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

```
# A tibble: 2 × 3
  country state     n
  <chr>   <chr> <int>
1 us      ut      659
2 <NA>    ut      138
```

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

```
# A tibble: 138 × 1
   city
   <chr>
 1 canyonlands np
 2 ogden/clinton
 3 sandy
 4 salt lake valley
 5 sandy
 6 duchenne
 7 west valley
 8 salt flats
 9 west valley
10 west valley
# ℹ 128 more rows
```

Get US States

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

```
 [1] "ak" "al" "ar" "az" "ca" "co" "ct" "dc" "de" "fl" "ga" "hi" "ia" "id" "il
[16] "in" "ks" "ky" "la" "ma" "md" "me" "mi" "mn" "mo" "ms" "mt" "nc" "nd" "ne
[31] "nh" "nj" "nm" "nv" "ny" "oh" "ok" "or" "pa" "pr" "ri" "sc" "sd" "tn" "tx
[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" "sa" "sk" "yk" "y
[16] NA
```

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

Get Great Britan states

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

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

```
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" "hi" "ia" "id" "il
[16] "in" "ks" "ky" "la" "ma" "md" "me" "mi" "mn" "mo" "ms" "mt" "nc" "nd" "ne
[31] "nh" "nj" "nm" "nv" "ny" "oh" "ok" "or" "pa" "pr" "ri" "sc" "sd" "tn" "tx
[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" "nc" "ns" "nt" "ri
[16] "sk" "tn" "wv" "yt" NA   "ab" "bc" "mb" "nb" "nf" "ns" "nt" "on" "pe" "pq
[31] "qc" "sa" "sk" "yk" "yt" NA
```

```
US_states <- setdiff(US_states, c(AU_states, GB_states, CA_states))
US_states
```

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

Continued

```
AU_states <- setdiff(AU_states, c(US_states, GB_states, CA_states))
CA_states <- setdiff(CA_states, c(US_states, GB_states, AU_states))
GB_states <- setdiff(GB_states, c(US_states, AU_states, CA_states))
```

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 × 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)) ~ "United States",
                (is.na(country) & state %in% c(CA_states)) ~ "Canada",
                (is.na(country) & state %in% c(AU_states)) ~ "Australia",
                (is.na(country) & state %in% c(GB_states)) ~ "Great Britain",
                   TRUE ~ country))
```

results

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

```
# A tibble: 10 × 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 × 13
  datetime         city  state country shape duration_seconds duration_hours_r
  <chr>            <chr> <chr> <chr>   <chr> <chr>            <chr>
1 10/25/1997 22:00 st. … vi    <NA>    light 8                5-8 secds
# ℹ 6 more variables: comments <chr>, date_posted <chr>, latitude <chr>,
#   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] "nurenburg (germany)"
 [14] "senftenberg (germany)"
 [15] "schwalmtal (germany)"
 [16] "neuss (germany)"
 [17] "babenhausen (germany)"
 [18] "berlin (germany)"
 [19] "mittenwald (germany)"
 [20] "ransbach-baumbach (germany)"
 [21] "ansbach (germany)"
 [22] "miesau (germany)"
 [23] "bensheim (germany)"
 [24] "muenster (germany)"
 [25] "chemnitz (germany)"
 [26] "kirchzell (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 × 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
```