Cleaning Data

Cleaning Data In R

GitHub Repository with this demonstration: https://github.com/unmrds/R-data-cleaning

Zipfile Download: https://github.com/unmrds/R-data-cleaning/archive/master.zip

Check out the **Preflight Check** to see if you have the needed R libraries installed run the `package.check.R` script in the top directory of this R project.

When planning a data analysis the first step, and often most time consuming, is the acquisition and processing of the data into a form that can be used in the analytic procedures you intend to use. Today we are going to focus on a sequence of steps that generally follow the workflow that you will find yourself going through when bringing data into R to perform an analysis.

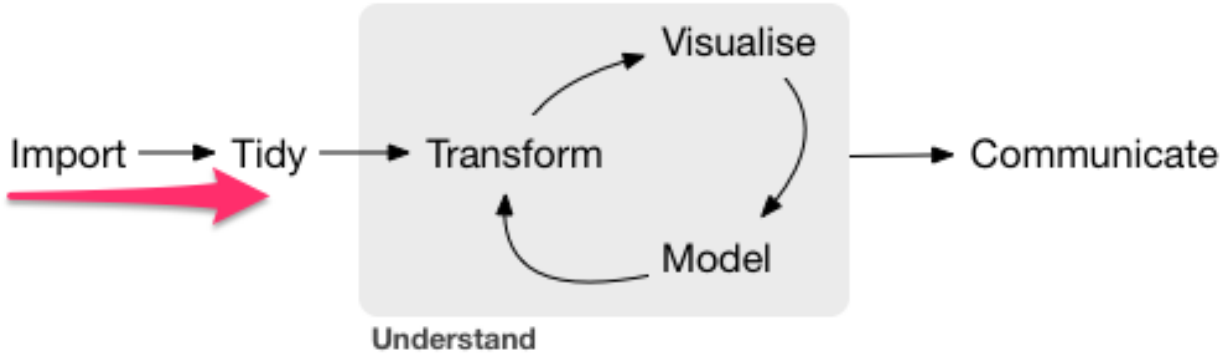


Figure 1: Portion of *R for Data Science*[1] workflow

Deal with issues that may come up when importing data files

1. Identify and correct structural issues in the source data that prevent clean import into R data structures
2. Check and handle data type errors
3. Check and handle missing data values

Tuning up the structure of the data to facilitate analysis

4. Split up fields that contain mutiple values in a single field
5. Check for anomalous values and otherwise explore the data to become familiar with its content and structure.

Beyond what we will cover today - continued structural changes and the rest of the exploration, analysis, and communication process.

Data for today's demonstration

The data for this demonstration are based upon the idigbio_rodents.csv dataset. The data are described as follows in the repository where they are shared:

[1] Hadley Wickham & Garrett Grolemund. 2017. *R for Data Science*. O'Reilly. https://r4ds.had.co.nz

> The idigbio_rodents.csv dataset is just over 10k records and contains data from natural history collections specimen records representing 5 genera from 4 major US collections, limited to US records. All records are from the Order Rodentia. All the data are mapped to the biodiversity data standard Darwin Core (http://rs.tdwg.org/dwc/terms/).

The original data have been modified for use in this demonstration by:

1. Generating new data columns (`latDMS` and `lonDMS`) for latitude and longitude that have sample coordinates presented in Degrees-Minutes-Seconds instead of the originally provided decimal degrees.
2. Generating a column of mixed numeric and text values - `textLatDD`.

This is the `../data/learning.csv` file. These newly created columns in addition to some of the originally provided ones will be used to demonstrate a variety of data cleaning steps in R.

An additional file was developed that only includes the first 10 rows of the file (including headers) but introduces a strucutral error. This file is the `../data/learning_struct.csv` file.

R libraries used in the demonstration

For this demonstration a set of R packages that are part of the *Tidyverse*. The tidyverse collection of packages provide (as described on the project's homepage):

> an opinionated collection of R packages designed for data science. All packages share an underlying design philosophy, grammar, and data structures.

There are currently over 14,000 R packages in the Comprehensive R Archive Network (CRAN). While the tidyverse packages provide a useful degree of consistency and internal interoperablity it is strongly encouraged to examine the broad collection of R packages when working on a particular analysis problem.

If you need to install the tidyverse packages in your enviornment you can execute the `install.packages("tidyverse")` command.

1. Identify and correct structural issues in the source data that prevent clean import into R data structures

R can import a wide variety of *rectangular* data structures: comma-delimited, tab-delimited, excel spreadsheets, fixed-width among the many options. If there are errors in the structure of these files, R import commands may not be able to parse the lines in the data file preventing import. In these cases the returned error messages may provide some clues to where the errors may be found.

One strategy for identifying potential strucutral issues in the source file is to try to import the dataset and review any errors that are returned

Let's try it first with a small file ...

```
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------------------------------------------
```

```
## v ggplot2 3.2.1     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.4
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

```
## -- Conflicts ------------------------------------------------------------------------------------
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
# Import the source CSV file that contains a structural flaw
rawDataStruct <- read_csv("../data/learning_struct.csv",
                     progress = FALSE)
```

```
## Warning: Missing column names filled in: 'X1' [1]
```

```
## Parsed with column specification:
## cols(
##   .default = col_character(),
##   X1 = col_double(),
##   catalogNumber = col_double(),
##   year = col_double(),
##   day = col_double()
## )
```

```
## See spec(...) for full column specifications.
```

```
## Warning: 1 parsing failure.
## row col   expected     actual                          file
##   7  -- 24 columns 26 columns '../data/learning_struct.csv'
```

```
# Display the column definitions for the imported dataset
spec(rawDataStruct)
```

```
## cols(
##   X1 = col_double(),
##   uuid = col_character(),
##   institutionCode = col_character(),
##   collectionCode = col_character(),
##   catalogNumber = col_double(),
##   recordedBy = col_character(),
##   countryCode = col_character(),
##   stateProvince = col_character(),
##   county = col_character(),
##   decimalLatitude = col_character(),
##   decimalLongitude = col_character(),
##   eventDate = col_character(),
##   year = col_double(),
##   month = col_character(),
##   day = col_double(),
##   genus = col_character(),
##   specificEpithet = col_character(),
##   scientificName = col_character(),
##   weight = col_character(),
##   length = col_character(),
##   sex = col_character(),
##   latDMS = col_character(),
##   lonDMS = col_character(),
##   textLatDD = col_character()
## )
```

```
# Report the problems that were encountered when the data were imported.
problems(rawDataStruct)
```

```
## # A tibble: 1 x 5
##     row col   expected   actual     file
##   <int> <chr> <chr>      <chr>      <chr>
## 1     7 <NA>  24 columns 26 columns '../data/learning_struct.csv'
```

```
# Display the imported table
rawDataStruct
```

```
## # A tibble: 9 x 24
##      X1 uuid  institutionCode collectionCode catalogNumber recordedBy
##   <dbl> <chr> <chr>           <chr>                 <dbl> <chr>
## 1     1 0603~ mvz             mammal specim~       219088 collector~
## 2     2 0fb1~ mvz             mammal specim~       233524 collector~
## 3     3 1a69~ mvz             mammal specim~       234346 collector~
## 4     4 1a99~ mvz             mammal specim~       233951 collector~
## 5     5 1f3b~ mvz             mammal specim~       235290 collector~
## 6     6 203f~ uam             mammal specim~        85106 collector~
## 7     7 21ce~ omnh            mammals               50048 caldwell
## 8     8 23d3~ mvz             mammal specim~       216309 collector~
## 9     9 244b~ msb             mammal specim~       294933 collector~
## # ... with 18 more variables: countryCode <chr>, stateProvince <chr>,
## #   county <chr>, decimalLatitude <chr>, decimalLongitude <chr>,
## #   eventDate <chr>, year <dbl>, month <chr>, day <dbl>, genus <chr>,
## #   specificEpithet <chr>, scientificName <chr>, weight <chr>, length <chr>,
## #   sex <chr>, latDMS <chr>, lonDMS <chr>, textLatDD <chr>
```

Let's take a look at the source data file and see if we can find the problem...

Now let's try it with the full dataset for which the structural problem has been resolved.

```
library(tidyverse)

# Import the source CSV file that does not contain the structural problem highlighted above
rawData <- read_csv("../data/learning.csv",
                    progress = FALSE)
```

```
## Warning: Missing column names filled in: 'X1' [1]
```

```
## Parsed with column specification:
## cols(
##   .default = col_character(),
##   X1 = col_double(),
##   catalogNumber = col_double(),
##   decimalLatitude = col_double(),
##   decimalLongitude = col_double(),
##   eventDate = col_datetime(format = ""),
##   year = col_double(),
##   month = col_double(),
##   day = col_double(),
##   weight = col_double(),
##   length = col_double()
## )
```

```
## See spec(...) for full column specifications.
```

```
# Display the column definitions for the imported dataset
spec(rawData)
```

```
## cols(
##   X1 = col_double(),
##   uuid = col_character(),
##   institutionCode = col_character(),
##   collectionCode = col_character(),
##   catalogNumber = col_double(),
##   recordedBy = col_character(),
```

```
##   countryCode = col_character(),
##   stateProvince = col_character(),
##   county = col_character(),
##   decimalLatitude = col_double(),
##   decimalLongitude = col_double(),
##   eventDate = col_datetime(format = ""),
##   year = col_double(),
##   month = col_double(),
##   day = col_double(),
##   genus = col_character(),
##   specificEpithet = col_character(),
##   scientificName = col_character(),
##   weight = col_double(),
##   length = col_double(),
##   sex = col_character(),
##   latDMS = col_character(),
##   lonDMS = col_character(),
##   textLatDD = col_character()
## )
```

```
# Report the problems that were encountered when the data were imported.
problems(rawData)
```

```
## [1] row      col      expected actual
## <0 rows> (or 0-length row.names)
```

```
# Display the imported table
rawData
```

```
## # A tibble: 10,767 x 24
##       X1 uuid  institutionCode collectionCode catalogNumber recordedBy
##    <dbl> <chr> <chr>           <chr>                 <dbl> <chr>
##  1     1 0603~ mvz             mammal specim~       219088 collector~
##  2     2 0fb1~ mvz             mammal specim~       233524 collector~
##  3     3 1a69~ mvz             mammal specim~       234346 collector~
##  4     4 1a99~ mvz             mammal specim~       233951 collector~
##  5     5 1f3b~ mvz             mammal specim~       235290 collector~
##  6     6 203f~ uam             mammal specim~        85106 collector~
##  7     7 21ce~ omnh            mammals               50048 caldwell,~
##  8     8 23d3~ mvz             mammal specim~       216309 collector~
##  9     9 244b~ msb             mammal specim~       294933 collector~
## 10    10 2682~ uam             mammal specim~        50255 collector~
## # ... with 10,757 more rows, and 18 more variables: countryCode <chr>,
## #   stateProvince <chr>, county <chr>, decimalLatitude <dbl>,
## #   decimalLongitude <dbl>, eventDate <dttm>, year <dbl>, month <dbl>,
## #   day <dbl>, genus <chr>, specificEpithet <chr>, scientificName <chr>,
## #   weight <dbl>, length <dbl>, sex <chr>, latDMS <chr>, lonDMS <chr>,
## #   textLatDD <chr>
```

Some questions:

1. How do the data types for the columns from this import process differ from those in the previous subset (at least before we fixed it)? Why do you think this is the case?
2. Where there any errors identified during the import? If no, does this mean that there are no potential problems or issues with the imported data? **Let's take a look**
3. How would you explain the values in the `eventDate` column when compared to the `year`, `month`, and `day` columns?

4. What were the different ways in which missing data values were handled?

2. Checking and handling data type errors

Depending on the types of data that are encountered in each column of the imported dataset different R import functions will automatically "type" the column (or in R terminology set the "mode" of the column) based on some sample of rows from the source file. In the case of `readr`, the data reading package used by tidyverse, the first 1000 lines of data will be read to determine the data type that should be used for each column. The core R data types are:

- character
- numeric (real or decimal)
- integer
- logical
- complex

These core data types can then be used as the foundation for more complex data types such as *dates*, *times* and *datetimes*.

The core data structures that can be used to organize collections of these data types include:

- vector - a sequence of data values of the same type
- list - a sequence of data values of the same or different types, and structures
- matrix - a vector for which *dimensions*
- data frame - a structured collection of vectors of the same length
- factors - a vactor vector is a set of integer values that are associated with a collection of categorical character values

Let's focus on the `catalogNumber` and `textLatDD` columns in our sample datasets.

```
spec(rawData)
```

```
## cols(
##   X1 = col_double(),
##   uuid = col_character(),
##   institutionCode = col_character(),
##   collectionCode = col_character(),
##   catalogNumber = col_double(),
##   recordedBy = col_character(),
##   countryCode = col_character(),
##   stateProvince = col_character(),
##   county = col_character(),
##   decimalLatitude = col_double(),
##   decimalLongitude = col_double(),
##   eventDate = col_datetime(format = ""),
##   year = col_double(),
##   month = col_double(),
##   day = col_double(),
##   genus = col_character(),
##   specificEpithet = col_character(),
##   scientificName = col_character(),
##   weight = col_double(),
##   length = col_double(),
##   sex = col_character(),
##   latDMS = col_character(),
##   lonDMS = col_character(),
##   textLatDD = col_character()
```

```
## )
```

```
rawData %>%
  select(catalogNumber,textLatDD)
```

```
## # A tibble: 10,767 x 2
##    catalogNumber textLatDD
##            <dbl> <chr>
##  1        219088 37.7609527778
##  2        233524 37.8999569
##  3        234346 37.8999569
##  4        233951 37.8999569
##  5        235290 37.8999569
##  6         85106 43.2751187
##  7         50048 34.53903
##  8        216309 36.97099
##  9        294933 36.584948
## 10         50255 44.2611111111
## # ... with 10,757 more rows
```

```
# test the creation of a numLatDD column as a numeric column and see what rows were converted to NA
rawData %>%
  mutate(numLatDD = as.numeric(rawData$textLatDD)) %>%
  filter(is.na(numLatDD)) %>%
  select(textLatDD, numLatDD) %>%
  print() %>%
  group_by(textLatDD) %>%
  summarize(count = n())
```

```
## Warning: NAs introduced by coercion
```

```
## # A tibble: 1,222 x 2
##    textLatDD numLatDD
##    <chr>        <dbl>
##  1 missing         NA
##  2 missing         NA
##  3 missing         NA
##  4 missing         NA
##  5 missing         NA
##  6 missing         NA
##  7 missing         NA
##  8 missing         NA
##  9 missing         NA
## 10 missing         NA
## # ... with 1,212 more rows
```

```
## # A tibble: 1 x 2
##   textLatDD count
##   <chr>     <int>
## 1 missing    1222
```

```
# create a numeric column based on the previously tested conversion of the textLatDD column
rawData$numLatDD <- as.numeric(rawData$textLatDD)
```

```
## Warning: NAs introduced by coercion
```

```
rawData
```

```
## # A tibble: 10,767 x 25
##       X1 uuid  institutionCode collectionCode catalogNumber recordedBy
##    <dbl> <chr> <chr>           <chr>                 <dbl> <chr>
##  1     1 0603~ mvz             mammal specim~       219088 collector~
##  2     2 0fb1~ mvz             mammal specim~       233524 collector~
##  3     3 1a69~ mvz             mammal specim~       234346 collector~
##  4     4 1a99~ mvz             mammal specim~       233951 collector~
##  5     5 1f3b~ mvz             mammal specim~       235290 collector~
##  6     6 203f~ uam             mammal specim~        85106 collector~
##  7     7 21ce~ omnh            mammals               50048 caldwell,~
##  8     8 23d3~ mvz             mammal specim~       216309 collector~
##  9     9 244b~ msb             mammal specim~       294933 collector~
## 10    10 2682~ uam             mammal specim~        50255 collector~
## # ... with 10,757 more rows, and 19 more variables: countryCode <chr>,
## #   stateProvince <chr>, county <chr>, decimalLatitude <dbl>,
## #   decimalLongitude <dbl>, eventDate <dttm>, year <dbl>, month <dbl>,
## #   day <dbl>, genus <chr>, specificEpithet <chr>, scientificName <chr>,
## #   weight <dbl>, length <dbl>, sex <chr>, latDMS <chr>, lonDMS <chr>,
## #   textLatDD <chr>, numLatDD <dbl>
```

We can also accomplish a similar outcome by specifying the column type that should be created as part of the import process.

```
rawData2 <- read_csv("../data/learning.csv",
                     col_types = cols(
                        textLatDD = col_double()
                        ),
                     progress = FALSE)
```

```
## Warning: Missing column names filled in: 'X1' [1]
```

```
## Warning: 1222 parsing failures.
## row       col expected  actual                   file
##  11 textLatDD a double missing '../data/learning.csv'
##  21 textLatDD a double missing '../data/learning.csv'
##  25 textLatDD a double missing '../data/learning.csv'
##  32 textLatDD a double missing '../data/learning.csv'
##  39 textLatDD a double missing '../data/learning.csv'
## ... ......... ........ ....... ......................
## See problems(...) for more details.
```

```
# Display the column definitions for the imported dataset
spec(rawData2)
```

```
## cols(
##   X1 = col_double(),
##   uuid = col_character(),
##   institutionCode = col_character(),
##   collectionCode = col_character(),
##   catalogNumber = col_double(),
##   recordedBy = col_character(),
##   countryCode = col_character(),
##   stateProvince = col_character(),
##   county = col_character(),
##   decimalLatitude = col_double(),
##   decimalLongitude = col_double(),
```

```
##   eventDate = col_datetime(format = ""),
##   year = col_double(),
##   month = col_double(),
##   day = col_double(),
##   genus = col_character(),
##   specificEpithet = col_character(),
##   scientificName = col_character(),
##   weight = col_double(),
##   length = col_double(),
##   sex = col_character(),
##   latDMS = col_character(),
##   lonDMS = col_character(),
##   textLatDD = col_double()
## )
```

```
# Report the problems that were encountered when the data were imported.
problems(rawData2)
```

```
## # A tibble: 1,222 x 5
##      row col       expected actual  file
##    <int> <chr>     <chr>    <chr>   <chr>
##  1    11 textLatDD a double missing '../data/learning.csv'
##  2    21 textLatDD a double missing '../data/learning.csv'
##  3    25 textLatDD a double missing '../data/learning.csv'
##  4    32 textLatDD a double missing '../data/learning.csv'
##  5    39 textLatDD a double missing '../data/learning.csv'
##  6    44 textLatDD a double missing '../data/learning.csv'
##  7    54 textLatDD a double missing '../data/learning.csv'
##  8    55 textLatDD a double missing '../data/learning.csv'
##  9    59 textLatDD a double missing '../data/learning.csv'
## 10    63 textLatDD a double missing '../data/learning.csv'
## # ... with 1,212 more rows
```

```
# Display the imported table
rawData2
```

```
## # A tibble: 10,767 x 24
##       X1 uuid  institutionCode collectionCode catalogNumber recordedBy
##    <dbl> <chr> <chr>           <chr>                 <dbl> <chr>
##  1     1 0603~ mvz             mammal specim~       219088 collector~
##  2     2 0fb1~ mvz             mammal specim~       233524 collector~
##  3     3 1a69~ mvz             mammal specim~       234346 collector~
##  4     4 1a99~ mvz             mammal specim~       233951 collector~
##  5     5 1f3b~ mvz             mammal specim~       235290 collector~
##  6     6 203f~ uam             mammal specim~        85106 collector~
##  7     7 21ce~ omnh            mammals               50048 caldwell,~
##  8     8 23d3~ mvz             mammal specim~       216309 collector~
##  9     9 244b~ msb             mammal specim~       294933 collector~
## 10    10 2682~ uam             mammal specim~        50255 collector~
## # ... with 10,757 more rows, and 18 more variables: countryCode <chr>,
## #   stateProvince <chr>, county <chr>, decimalLatitude <dbl>,
## #   decimalLongitude <dbl>, eventDate <dttm>, year <dbl>, month <dbl>,
## #   day <dbl>, genus <chr>, specificEpithet <chr>, scientificName <chr>,
## #   weight <dbl>, length <dbl>, sex <chr>, latDMS <chr>, lonDMS <chr>,
## #   textLatDD <dbl>
```

```
# convert the catalogNumberTxt column to a character column and see what the result is
rawData %>%
  mutate(catalogNumberTxt = as.character(catalogNumber)) %>%
  filter(is.na(catalogNumberTxt))
```

```
## # A tibble: 0 x 26
## # ... with 26 variables: X1 <dbl>, uuid <chr>, institutionCode <chr>,
## #   collectionCode <chr>, catalogNumber <dbl>, recordedBy <chr>,
## #   countryCode <chr>, stateProvince <chr>, county <chr>,
## #   decimalLatitude <dbl>, decimalLongitude <dbl>, eventDate <dttm>,
## #   year <dbl>, month <dbl>, day <dbl>, genus <chr>, specificEpithet <chr>,
## #   scientificName <chr>, weight <dbl>, length <dbl>, sex <chr>, latDMS <chr>,
## #   lonDMS <chr>, textLatDD <chr>, numLatDD <dbl>, catalogNumberTxt <chr>
```

3. Check and handle missing values

It is important to understand the potential impact that missing data will have on your analysis. As we've already seen the import process may automatically produce missing data values in your analysis dataframe (or *tibble* in the context of tidyverse based processes). Some functions enable you to efficiently visualize the patterns of missing values in your dataset - allowing for the analysis of large datasets that otherwise would not be feasible to review manually.

```
paste("decimalLatitude: number of NA values", sum(is.na(rawData$decimalLatitude)), sep = " ")
```

```
## [1] "decimalLatitude: number of NA values 1222"
```

```
paste("decimalLongitude: number of NA values", sum(is.na(rawData$decimalLongitude)), sep = " ")
```

```
## [1] "decimalLongitude: number of NA values 1222"
```

```
paste("weight: number of NA values", sum(is.na(rawData$weight)), sep = " ")
```

```
## [1] "weight: number of NA values 0"
```

```
paste("length: number of NA values", sum(is.na(rawData$length)), sep = " ")
```

```
## [1] "length: number of NA values 0"
```

```
paste("sex: number of NA values", sum(is.na(rawData$sex)), sep = " ")
```

```
## [1] "sex: number of NA values 1446"
```

```
paste("latDMS: number of NA values", sum(is.na(rawData$latDMS)), sep = " ")
```

```
## [1] "latDMS: number of NA values 0"
```

```
paste("lonDMS: number of NA values", sum(is.na(rawData$lonDMS)), sep = " ")
```

```
## [1] "lonDMS: number of NA values 0"
```

```
paste("textLatDD: number of NA values", sum(is.na(rawData$textLatDD)), sep = " ")
```

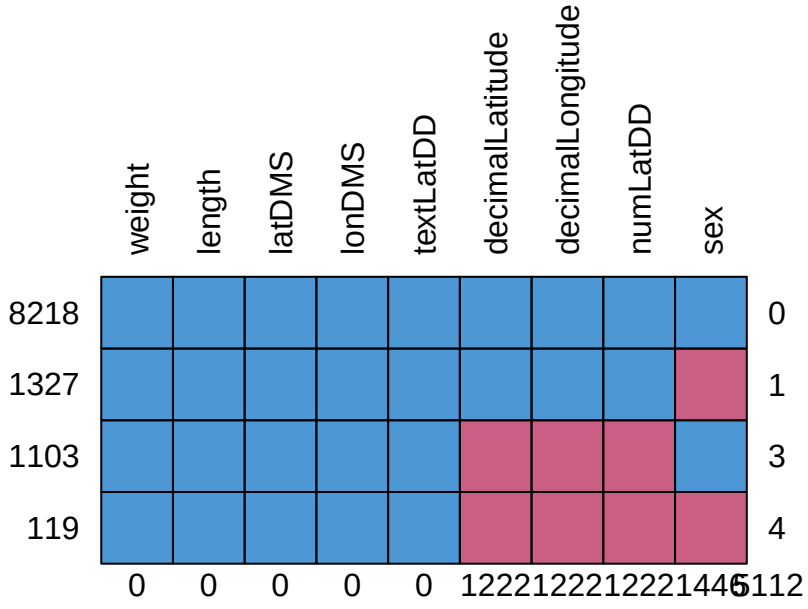
```
## [1] "textLatDD: number of NA values 0"
```

```
paste("numLatDD: number of NA values", sum(is.na(rawData$numLatDD)), sep = " ")
```

```
## [1] "numLatDD: number of NA values 1222"
```

In this analysis we will be using the `md.pattern` function that is part of the `mice` and `VIM` packages. If you haven't already installed the `mice` package you can do so by executing the `install.packages("mice")` command.

```
##
## Attaching package: 'mice'

## The following objects are masked from 'package:base':
##
##     cbind, rbind

## Loading required package: colorspace

## Loading required package: grid

## Loading required package: data.table

##
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':
##
##     between, first, last

## The following object is masked from 'package:purrr':
##
##     transpose

## VIM is ready to use.
##  Since version 4.0.0 the GUI is in its own package VIMGUI.
##
##           Please use the package to use the new (and old) GUI.

## Suggestions and bug-reports can be submitted at: https://github.com/alexkowa/VIM/issues

##
## Attaching package: 'VIM'

## The following object is masked from 'package:datasets':
##
##     sleep
```

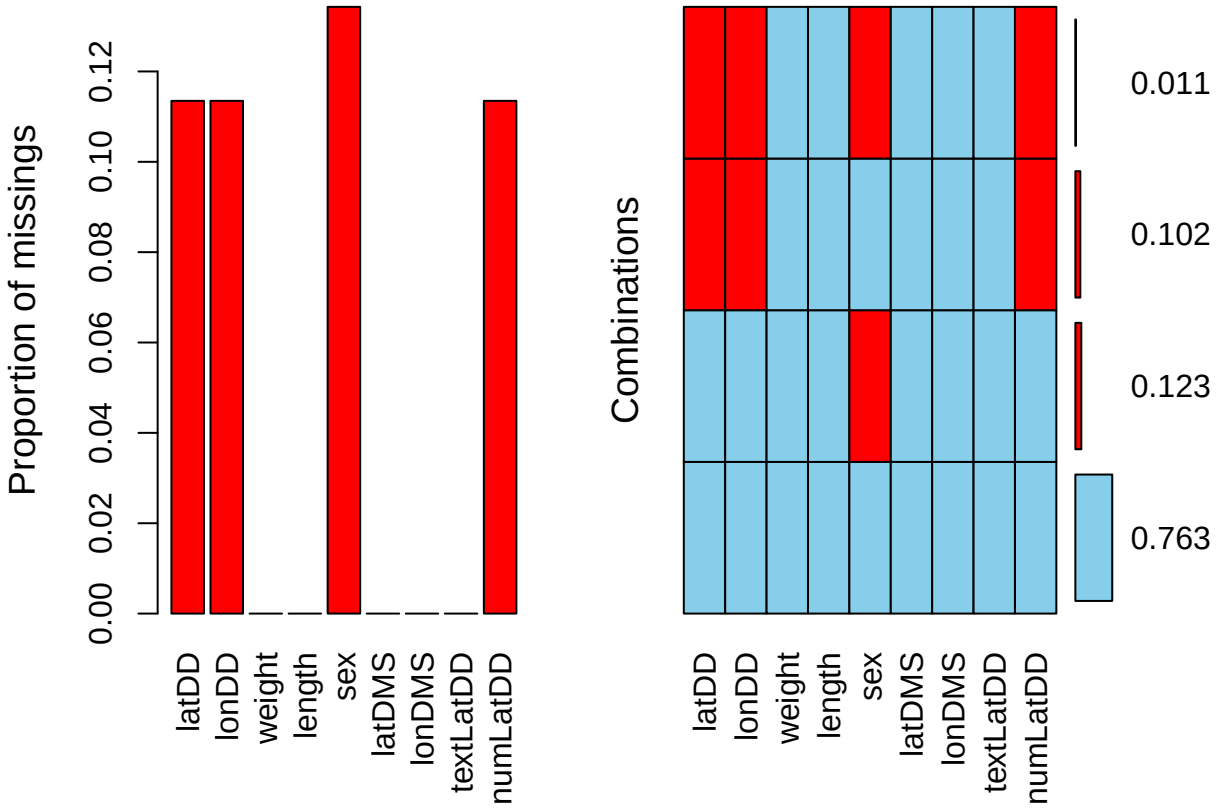



```
##      weight length latDMS lonDMS textLatDD decimalLatitude decimalLongitude numLatDD  sex
## 8218      1      1      1      1         1               1                1        1    1    0
```

```
## 1327       1       1       1       1       1                 1                    1         1     0     1
## 1103       1       1       1       1       1                 0                    0         0     1     3
## 119        1       1       1       1       1                 0                    0         0     0     4
##           0       0       0       0       0              1222                 1222      1222  1446  5112
```

Another method of viewing similar information

```
rawData %>%
  select(decimalLatitude, decimalLongitude, weight, length, sex, latDMS, lonDMS, textLatDD, numLatDD) %
  rename(latDD = decimalLatitude, lonDD = decimalLongitude) %>%
  aggr(numbers=TRUE)
```



4. Multi-value columns

A core principle of having well structured data that is read for analysis is that:

In the context of "Tidy" data that underlie the tools developed as part of the tidyverse packageˆ[Hadley Wickham & Garrett Grolemund. 2017. *R for Data Science*. O'Reilly. - section on Tidy Data https://r4ds.had.co.nz/tidy-data.html

There are three interrelated rules which make a dataset tidy:

1. Each variable must have its own column.
2. Each observation must have its own row.
3. **Each value must have its own cell.**

This issue also relates to the idea of *atomicity* in Codd's definition of *First Normal Form* when *normalizing* a relational database[2]. While we're not going to get into relational data modeling in R here, well structured data allow for the use of relational data models in your analysis independent of a separate database server.

[2]https://en.wikipedia.org/wiki/First_normal_form

In this example we are going to focus on three columns: `recordedBy`, `latDMS`, and `lonDMS`.

```
rawData %>%
  select(recordedBy, latDMS, lonDMS)
```

```
## # A tibble: 10,767 x 3
##    recordedBy                                                  latDMS            lonDMS
##    <chr>                                                       <chr>             <chr>
##  1 collector(s): ana lilia trujano álvarez, eric ghilarducci "37°45'39.430\"N" "-122° 6'48.370\"E"
##  2 collector(s): william z. lidicker jr.                      "37°53'59.845\"N" "-123°38'18.039\"E"
##  3 collector(s): william z. lidicker jr.                      "37°53'59.845\"N" "-123°38'18.039\"E"
##  4 collector(s): william z. lidicker jr.                      "37°53'59.845\"N" "-123°38'18.039\"E"
##  5 collector(s): william z. lidicker jr.                      "37°53'59.845\"N" "-123°38'18.039\"E"
##  6 collector(s): tom manning; preparator(s): amber baxter     "43°16'30.427\"N" "-123°12'32.023\"E"
##  7 caldwell, j. p. and vitt, l. j.                            "34°32'20.508\"N" "-95° 6'42.444\"E"
##  8 collector(s): james l. patton                              "36°58'15.564\"N" "-119°54'16.740\"E"
##  9 collector(s): troy l. best; preparator(s): troy l. best    "36°35' 5.813\"N" "-108° 0'55.757\"E"
## 10 collector(s): karl j. martin; preparator(s): paul ollig    "44°15'40.000\"N" "-124°25' 1.000\"E"
## # ... with 10,757 more rows
```

Breaking apart the `recordedBy` column using the `str_match` function from the `stringr` package. This uses *regular expressions* for defining the text patterns that should be found and processed in the process of breaking the column apart into new columns. Regular expressions are an art to themselves and there are many resources for learning their effective use - ranging from one-page cheat-sheets to full length books. The following figure describes the structure of the regular expressions used in the sample code:

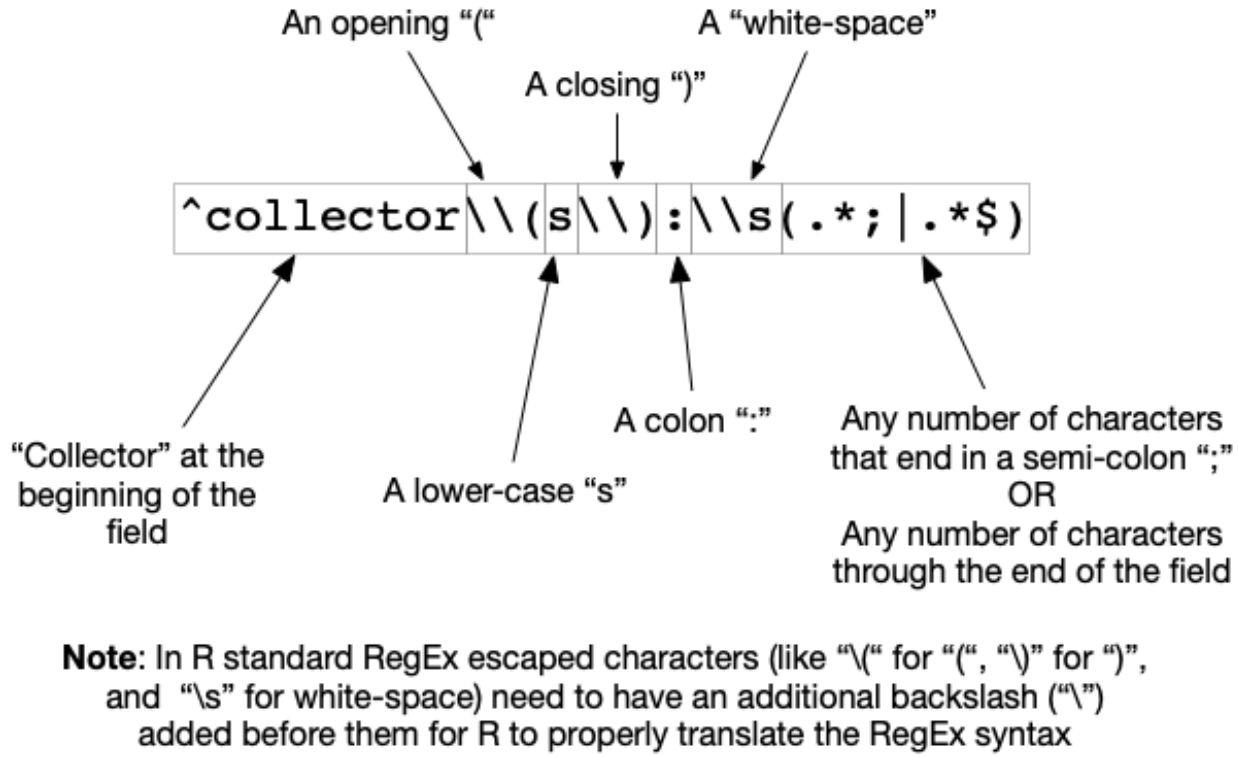


Figure 2: Description of the regular expression used to extract targeted substrings from the combined "recordedBy" field

```
collectorExtract <- "^collector\\(s\\):\\s(.*;|.*$)"
preparatorExtract <- "preparator\\(s\\):\\s(.*;|.*$)"
#str_match(rawData$recordedBy, collectorExtract)[,2]
#str_match(rawData$recordedBy, preparatorExtract)[,2]
rawData$collectors <- str_match(rawData$recordedBy, collectorExtract)[,2]
rawData$preparators <- str_match(rawData$recordedBy, preparatorExtract)[,2]
```

What would the next logical step in the process be for cleaning up the ***recordedBy*** *column?*

Breaking apart the `latDMS` and `lonDMS` columns into their constituent parts

```
dmsExtract <- "\\s*(-*[:digit:]+)°\\s*([:digit:]+)\\'\\s*([:digit:]+)"

latSubstrings <- str_match(rawData$latDMS, dmsExtract)
rawData$latD <- as.numeric(latSubstrings[,2])
rawData$latM <- as.numeric(latSubstrings[,3])
rawData$latS <- as.numeric(latSubstrings[,4])

glimpse(latSubstrings)
```

```
##  chr [1:10767, 1:4] "37°45'39" "37°53'59" "37°53'59" "37°53'59" "37°53'59" "43°16'30" "34°32'20" "36
```

```
lonSubstrings <- str_match(rawData$lonDMS, dmsExtract)
rawData$lonD <- as.numeric(lonSubstrings[,2])
rawData$lonM <- as.numeric(lonSubstrings[,3])
rawData$lonS <- as.numeric(lonSubstrings[,4])

glimpse(lonSubstrings)
```

```
##  chr [1:10767, 1:4] "-122° 6'48" "-123°38'18" "-123°38'18" "-123°38'18" "-123°38'18" "-123°12'32" "-
```

```
glimpse(rawData)
```

```
## Observations: 10,767
## Variables: 33
## $ X1               <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 1
## $ uuid             <chr> "060380ea-7b06-474e-8d2e-b6e4a8c21e1a", "0fb17a79-a8ce-45b6-b57a-2f640e8cccb
## $ institutionCode  <chr> "mvz", "mvz", "mvz", "mvz", "mvz", "uam", "omnh", "mvz", "msb", "uam", "msb
## $ collectionCode   <chr> "mammal specimens", "mammal specimens", "mammal specimens", "mammal specimen
## $ catalogNumber    <dbl> 219088, 233524, 234346, 233951, 235290, 85106, 50048, 216309, 294933, 50255
## $ recordedBy       <chr> "collector(s): ana lilia trujano álvarez, eric ghilarducci", "collector(s):
## $ countryCode      <chr> "usa", "usa", "usa", "usa", "usa", "usa", "usa", "usa", "usa", "usa", "usa"
## $ stateProvince    <chr> "california", "california", "california", "california", "california", "orego
## $ county           <chr> "contra costa county", "contra costa county", "contra costa county", "contra
## $ decimalLatitude  <dbl> 37.76095, 37.89996, 37.89996, 37.89996, 37.89996, 43.27512, 34.53903, 36.970
## $ decimalLongitude <dbl> -121.88656, -122.36166, -122.36166, -122.36166, -122.36166, -122.79110, -94
## $ eventDate        <dttm> 2005-11-23, 1959-06-21, 1962-11-22, 1960-07-31, 1964-07-04, 1996-10-23, 20
## $ year             <dbl> 2005, 1959, 1962, 1960, 1964, 1996, 2011, 2005, 1989, 1996, 2013, 1977, 199
## $ month            <dbl> 11, 6, 11, 7, 7, 10, 1, 8, 6, 5, 8, 8, 11, 6, 7, 6, 4, 8, 12, 10, 1, 9, 7, 8
## $ day              <dbl> 22, 20, 21, 30, 3, 22, 17, 6, 4, 19, 10, 12, 30, 17, 17, 2, 21, 6, 26, 26, 1
## $ genus            <chr> "microtus", "microtus", "microtus", "microtus", "microtus", "myodes", "micro
## $ specificEpithet  <chr> "californicus", "californicus", "californicus", "californicus", "californicu
## $ scientificName   <chr> "microtus californicus californicus", "microtus californicus californicus",
## $ weight           <dbl> 30.5, 22.0, 49.0, 33.0, 29.0, 23.5, 24.0, 27.0, 125.0, 12.0, 56.0, 33.5, 33
## $ length           <dbl> 165, 143, 187, 169, 159, 141, 121, 176, 294, 110, 210, 148, 149, 170, 182, 1
## $ sex              <chr> "male", "female", "male", "female", "female", "female", NA, "female", "femal
```

```
## $ latDMS           <chr> "37°45'39.430\"N", "37°53'59.845\"N", "37°53'59.845\"N", "37°53'59.845\"N",
## $ lonDMS           <chr> "-122° 6'48.370\"E", "-123°38'18.039\"E", "-123°38'18.039\"E", "-123°38'18.
## $ textLatDD        <chr> "37.7609527778", "37.8999569", "37.8999569", "37.8999569", "37.8999569", "43
## $ numLatDD         <dbl> 37.76095, 37.89996, 37.89996, 37.89996, 37.89996, 43.27512, 34.53903, 36.97
## $ collectors       <chr> "ana lilia trujano álvarez, eric ghilarducci", "william z. lidicker jr.", "
## $ preparators      <chr> NA, NA, NA, NA, NA, "amber baxter", NA, NA, "troy l. best", "paul ollig", NA
## $ latD             <dbl> 37, 37, 37, 37, 37, 43, 34, 36, 36, 44, NA, 39, 45, 37, 37, 32, 37, 36, 36,
## $ latM             <dbl> 45, 53, 53, 53, 53, 16, 32, 58, 35, 15, NA, 59, 32, 53, 30, 30, 53, 28, 58,
## $ latS             <dbl> 39, 59, 59, 59, 59, 30, 20, 15, 5, 40, NA, 36, 24, 59, 12, 27, 59, 39, 1, 13
## $ lonD             <dbl> -122, -123, -123, -123, -123, -123, -95, -119, -108, -124, NA, -109, -109, -
## $ lonM             <dbl> 6, 38, 38, 38, 38, 12, 6, 54, 0, 25, NA, 46, 10, 38, 51, 17, 38, 46, 41, 32
## $ lonS             <dbl> 48, 18, 18, 18, 18, 32, 42, 16, 55, 1, NA, 28, 48, 18, 8, 58, 18, 52, 6, 6,
```

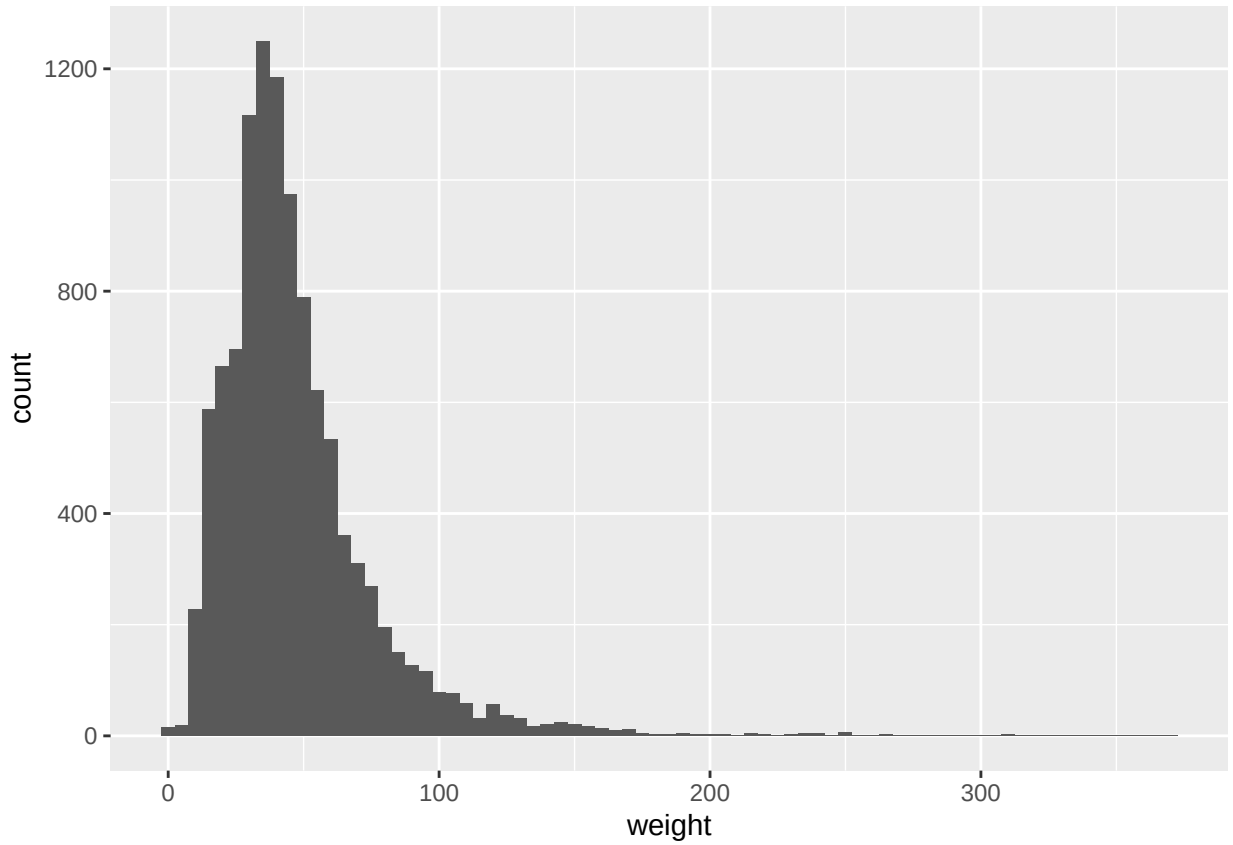
5. Check value ranges and explore data

As part of the examiniation of these columns we will use the `assertr`. If you need to install the `assertr` package in your enviornment you can execute the `install.packages("assertr")` command.

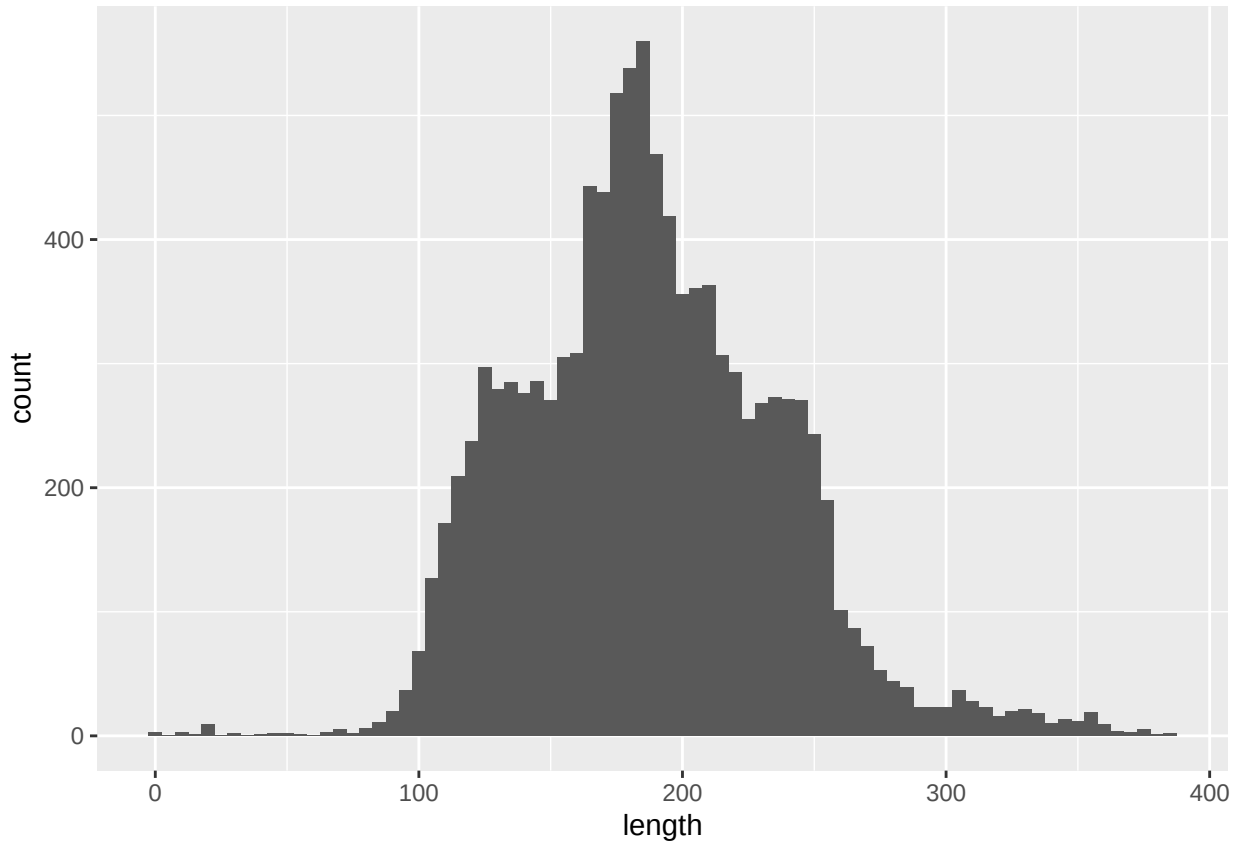
Checking the `weight` and `length` columns.

```
#library(assertr)
#
#rawData %>%
#  chain_start %>%
#  assert(within_bounds(1,Inf), weight) %>%
#  assert(within_bounds(1,Inf), length) %>%
#  insist(within_n_sds(3), weight) %>%
#  insist(within_n_sds(3), length) %>%
#  chain_end
```

```
library(ggplot2)
ggplot(rawData, aes(x=weight)) +
  geom_histogram(binwidth = 5)
```




```
ggplot(rawData, aes(x=length)) +
  geom_histogram(binwidth = 5)
```



6. Bringing it all together to produce our analysis dataset

```
# import the data with explicit column definitions for the textLatDD and catalogNumber columns
analysisData <- read_csv("../data/learning.csv",
                     col_types = cols(
                        textLatDD = col_double(),
                        catalogNumber = col_character()
                        ),
                     progress = FALSE)
```

```
## Warning: Missing column names filled in: 'X1' [1]
```

```
## Warning: 1222 parsing failures.
## row       col expected  actual                   file
##  11 textLatDD a double missing '../data/learning.csv'
##  21 textLatDD a double missing '../data/learning.csv'
##  25 textLatDD a double missing '../data/learning.csv'
##  32 textLatDD a double missing '../data/learning.csv'
##  39 textLatDD a double missing '../data/learning.csv'
## ... ......... ........ ....... ......................
## See problems(...) for more details.
```

```
# split up the recordedBy column
collectorExtract <- "^collector\\(s\\):\\s(.*;|.*$)"
preparatorExtract <- "preparator\\(s\\):\\s(.*;|.*$)"
analysisData$collectors <- str_match(analysisData$recordedBy, collectorExtract)[,2]
analysisData$preparators <- str_match(analysisData$recordedBy, preparatorExtract)[,2]
```

```
# split up the latDMS and lonDMS columns
dmsExtract <- "\\s*(-*[:digit:]+)°\\s*([:digit:]+)\\'\\s*([:digit:]+)"

latSubstrings <- str_match(analysisData$latDMS, dmsExtract)
analysisData$latD <- as.numeric(latSubstrings[,2])
analysisData$latM <- as.numeric(latSubstrings[,3])
analysisData$latS <- as.numeric(latSubstrings[,4])

lonSubstrings <- str_match(analysisData$lonDMS, dmsExtract)
analysisData$lonD <- as.numeric(lonSubstrings[,2])
analysisData$lonM <- as.numeric(lonSubstrings[,3])
analysisData$lonS <- as.numeric(lonSubstrings[,4])

glimpse(analysisData)
```

```
## Observations: 10,767
## Variables: 32
## $ X1                <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 1
## $ uuid              <chr> "060380ea-7b06-474e-8d2e-b6e4a8c21e1a", "0fb17a79-a8ce-45b6-b57a-2f640e8cccb
## $ institutionCode   <chr> "mvz", "mvz", "mvz", "mvz", "mvz", "uam", "omnh", "mvz", "msb", "uam", "msb
## $ collectionCode    <chr> "mammal specimens", "mammal specimens", "mammal specimens", "mammal specimen
## $ catalogNumber     <chr> "219088", "233524", "234346", "233951", "235290", "85106", "50048", "216309
## $ recordedBy        <chr> "collector(s): ana lilia trujano álvarez, eric ghilarducci", "collector(s):
## $ countryCode       <chr> "usa", "usa", "usa", "usa", "usa", "usa", "usa", "usa", "usa", "usa", "usa"
## $ stateProvince     <chr> "california", "california", "california", "california", "california", "orego
## $ county            <chr> "contra costa county", "contra costa county", "contra costa county", "contra
## $ decimalLatitude   <dbl> 37.76095, 37.89996, 37.89996, 37.89996, 37.89996, 43.27512, 34.53903, 36.970
## $ decimalLongitude  <dbl> -121.88656, -122.36166, -122.36166, -122.36166, -122.36166, -122.79110, -94
## $ eventDate         <dttm> 2005-11-23, 1959-06-21, 1962-11-22, 1960-07-31, 1964-07-04, 1996-10-23, 20
## $ year              <dbl> 2005, 1959, 1962, 1960, 1964, 1996, 2011, 2005, 1989, 1996, 2013, 1977, 199
## $ month             <dbl> 11, 6, 11, 7, 7, 10, 1, 8, 6, 5, 8, 8, 11, 6, 7, 6, 4, 8, 12, 10, 1, 9, 7, 8
## $ day               <dbl> 22, 20, 21, 30, 3, 22, 17, 6, 4, 19, 10, 12, 30, 17, 17, 2, 21, 6, 26, 26, 1
## $ genus             <chr> "microtus", "microtus", "microtus", "microtus", "microtus", "myodes", "micro
## $ specificEpithet   <chr> "californicus", "californicus", "californicus", "californicus", "californicu
## $ scientificName    <chr> "microtus californicus californicus", "microtus californicus californicus",
## $ weight            <dbl> 30.5, 22.0, 49.0, 33.0, 29.0, 23.5, 24.0, 27.0, 125.0, 12.0, 56.0, 33.5, 33
## $ length            <dbl> 165, 143, 187, 169, 159, 141, 121, 176, 294, 110, 210, 148, 149, 170, 182, 1
## $ sex               <chr> "male", "female", "male", "female", "female", "female", NA, "female", "femal
## $ latDMS            <chr> "37°45'39.430\"N", "37°53'59.845\"N", "37°53'59.845\"N", "37°53'59.845\"N",
## $ lonDMS            <chr> "-122° 6'48.370\"E", "-123°38'18.039\"E", "-123°38'18.039\"E", "-123°38'18.0
## $ textLatDD         <dbl> 37.76095, 37.89996, 37.89996, 37.89996, 37.89996, 43.27512, 34.53903, 36.970
## $ collectors        <chr> "ana lilia trujano álvarez, eric ghilarducci", "william z. lidicker jr.", "w
## $ preparators       <chr> NA, NA, NA, NA, NA, "amber baxter", NA, NA, "troy l. best", "paul ollig", NA
## $ latD              <dbl> 37, 37, 37, 37, 37, 43, 34, 36, 36, 44, NA, 39, 45, 37, 37, 32, 37, 36, 36,
## $ latM              <dbl> 45, 53, 53, 53, 53, 16, 32, 58, 35, 15, NA, 59, 32, 53, 30, 30, 53, 28, 58,
## $ latS              <dbl> 39, 59, 59, 59, 59, 30, 20, 15, 5, 40, NA, 36, 24, 59, 12, 27, 59, 39, 1, 13
## $ lonD              <dbl> -122, -123, -123, -123, -123, -123, -95, -119, -108, -124, NA, -109, -109, -
## $ lonM              <dbl> 6, 38, 38, 38, 38, 12, 6, 54, 0, 25, NA, 46, 10, 38, 51, 17, 38, 46, 41, 32
## $ lonS              <dbl> 48, 18, 18, 18, 18, 32, 42, 16, 55, 1, NA, 28, 48, 18, 8, 58, 18, 52, 6, 6,
```

Export the code from the workshop both as a documented and undocumented R script

```
#library(knitr)
#purl("cleaning_data.Rmd", "cleaning_data_nodocs.R", documentation = 0)
```