

# 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. This script will check to see if the packages are installed, and if they are not will do so. The script finishes with a listing of the currently installed packages so you can verify that the packages we need are installed. Check out the README.md file in the workshop repository for full setup instructions.

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.

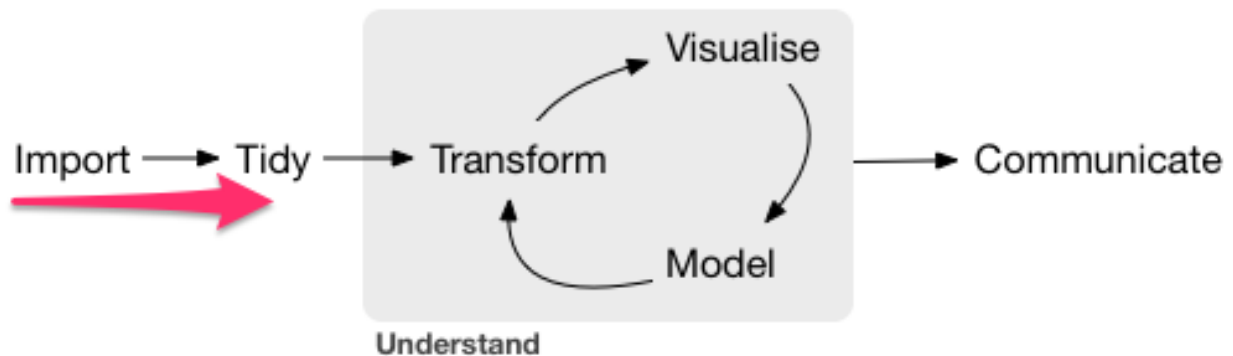


Figure 1: Portion of *R for Data Science*<sup>1</sup> workflow

### 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 multiple 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.

---

<sup>1</sup>Hadley Wickham & Garrett Golemund. 2017. *R for Data Science*. O'Reilly. <https://r4ds.had.co.nz>

## 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:

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 structural error. This file is the `../data/learning_struct.csv` file.

## R libraries used in the demonstration

For this demonstration a combination of R packages that are part of the *Tidyverse* and additional specialized data evaluation packages will be used. 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.

In addition to `tidyverse`, this workshop also uses the `mice`, `VIM`, and `assertr` packages.

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 interoperability it is strongly encouraged to examine the broad collection of R packages when working on a particular analysis problem.

If you need to install any of the needed packages in your environment you can execute the `install.packages("<package name>")` command in the R console.

## 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 structural 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 ...

```
##### Working with a CSV file with structural problems #####  
  
library(tidyverse)  
  
## -- Attaching packages -----  
## v ggplot2 3.3.0      v purrr  0.3.3  
## v tibble  3.0.0      v dplyr  0.8.5
```

```

## v tidyr 1.0.2      v stringr 1.4.0
## v readr 1.3.1      v forcats 0.5.0

## -- Conflicts ----- tidyverse
## 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>
```

The output of the `read_csv` command and of the `problems` function indicate that there was a problem of some sort around row 7 during the import of the CSV:

```
1 parsing failure. row col expected actual file 7 – 24 columns 26 columns '../data/learning_struct.csv'
```

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

After the structural problem has been resolved load the full dataset for use in the rest of the workshop

```
##### Load the full CSV file without the structural error #####
```

```
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 <dtm>, 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 (and the “tibble” in the tidyverse) - a structured collection of vectors of the same length
- factors - a vector 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.

```
##### Handling data type errors on import #####
```

```
## Take a look at the types and content of a couple of columns
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 <dtm>, 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.

```
## Specify the column data type when importing
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 <dtm>, 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 <dtm>,
## #   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.

```
##### Check and handle missing values #####

## Manually by printing out sum (FALSE = 0, TRUE = 1) of is.na test results
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"
```

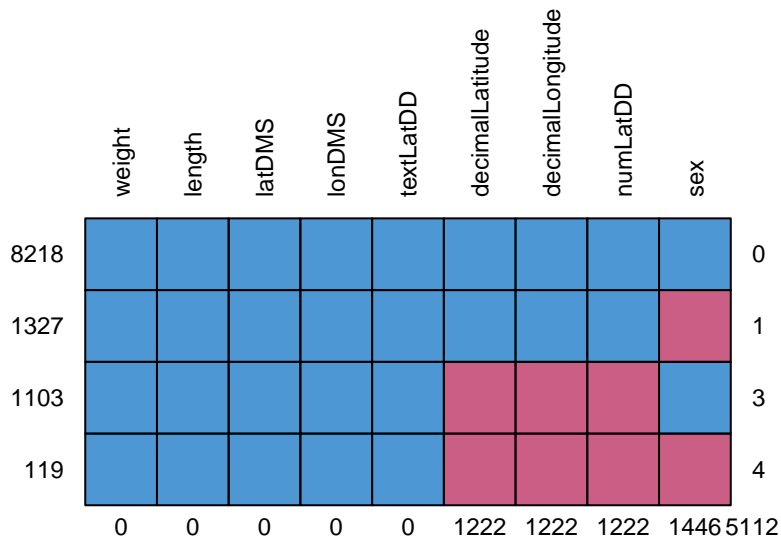
```
paste("numLatDD: number of NA values",
      sum(is.na(rawData$numLatDD)),
      sep = " ")
```

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

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

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

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



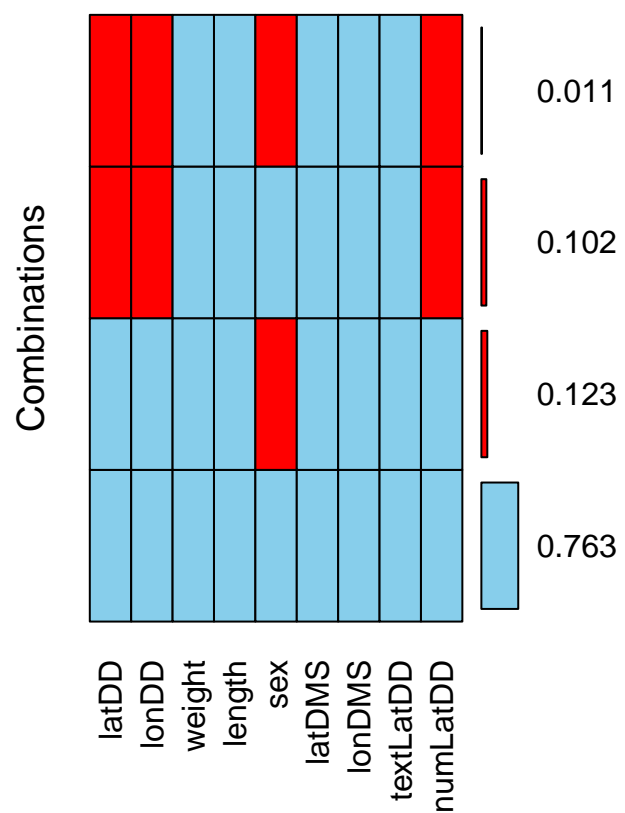
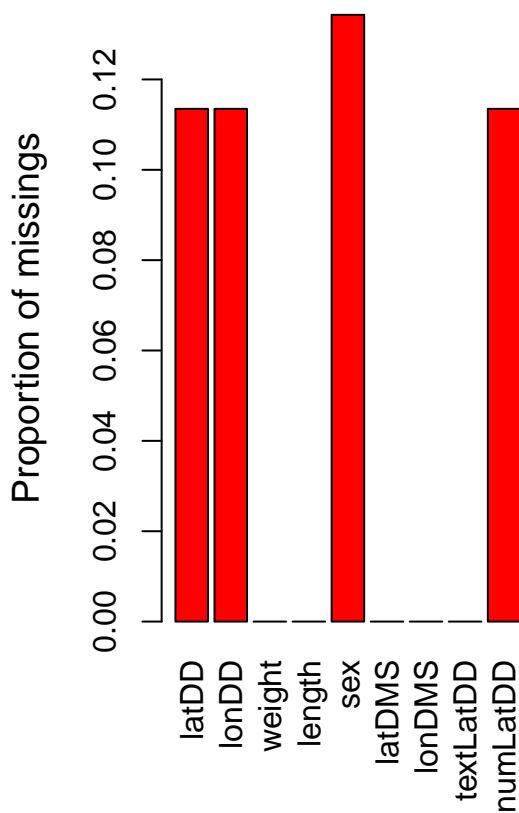
```
##      weight length latDMS lonDMS textLatDD decimalLatitude decimalLongitude numLatDD sex
## 8218      1      1      1      1      1      1222      1222      1222      1446 5112
## 1327      1      1      1      1      1      1222      1222      1222      1446 5112
## 1103      1      1      1      1      1      1222      1222      1222      1446 5112
## 119      1      1      1      1      1      1222      1222      1222      1446 5112
##      0      0      0      0      0      1222      1222      1222      1446 5112
```

Another method of viewing similar information

```
## View the combinations of NA values across multiple columns with the
## agr function from the VIM package
library(VIM)
```

```
## 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/alexxkova/VIM/issues
##
## Attaching package: 'VIM'
## The following object is masked from 'package:datasets':
##
## sleep
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 package<sup>^</sup>[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<sup>2</sup>. 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.

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

```
##### Handling multi-value columns #####

## Take a look at the recordedBy, latDMS, and lonDMS columns
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~ "37°45'39.430~ "-122° 6'48.37~
## 2 collector(s): william z. lidicker jr.         "37°53'59.845~ "-123°38'18.03~
## 3 collector(s): william z. lidicker jr.         "37°53'59.845~ "-123°38'18.03~
## 4 collector(s): william z. lidicker jr.         "37°53'59.845~ "-123°38'18.03~
## 5 collector(s): william z. lidicker jr.         "37°53'59.845~ "-123°38'18.03~
## 6 collector(s): tom manning; preparator(s): amb~ "43°16'30.427~ "-123°12'32.02~
## 7 caldwell, j. p. and vitt, l. j.              "34°32'20.508~ "-95° 6'42.444~
## 8 collector(s): james l. patton                  "36°58'15.564~ "-119°54'16.74~
## 9 collector(s): troy l. best; preparator(s): tr~ "36°35' 5.813~ "-108° 0'55.75~
## 10 collector(s): karl j. martin; preparator(s): ~ "44°15'40.000~ "-124°25' 1.00~
## # ... 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. Some great resources include:

- Jeffrey E.F. Friedl (2006) *Mastering Regular Expressions*. 3rd Ed. O’Reilly. <http://shop.oreilly.com/product/9780596528126.do> and for UNM affiliates through our Safari Online Learning subscription: <https://learning.oreilly.com/library/view/mastering-regular-expressions/0596528124/> - this is a comprehensive, in-depth treatment of regular expressions from the most simple to most complex - including discussions of implementations of regular expression support in different programming languages.
- Steven Levithan, Jan Goyvaerts (2012). *Regular Expressions Cookbook*. 2nd Ed. O’Reilly. <http://shop.oreilly.com/product/0636920023630.do> and for UNM affiliates <https://learning.oreilly.com/library/view/regular-expressions-cookbook/9781449327453/> - a good set of regular expression common problems, and their solutions in multiple language implementations.

<sup>2</sup>[https://en.wikipedia.org/wiki/First\\_normal\\_form](https://en.wikipedia.org/wiki/First_normal_form)

- Ian Kopacka (2016). *Basic Regular Expressions in R - Cheat Sheet*. Online resource: <https://rstudio.com/wp-content/uploads/2016/09/RegExCheatsheet.pdf>

The following figure describes the structure of the regular expressions used in the sample code:

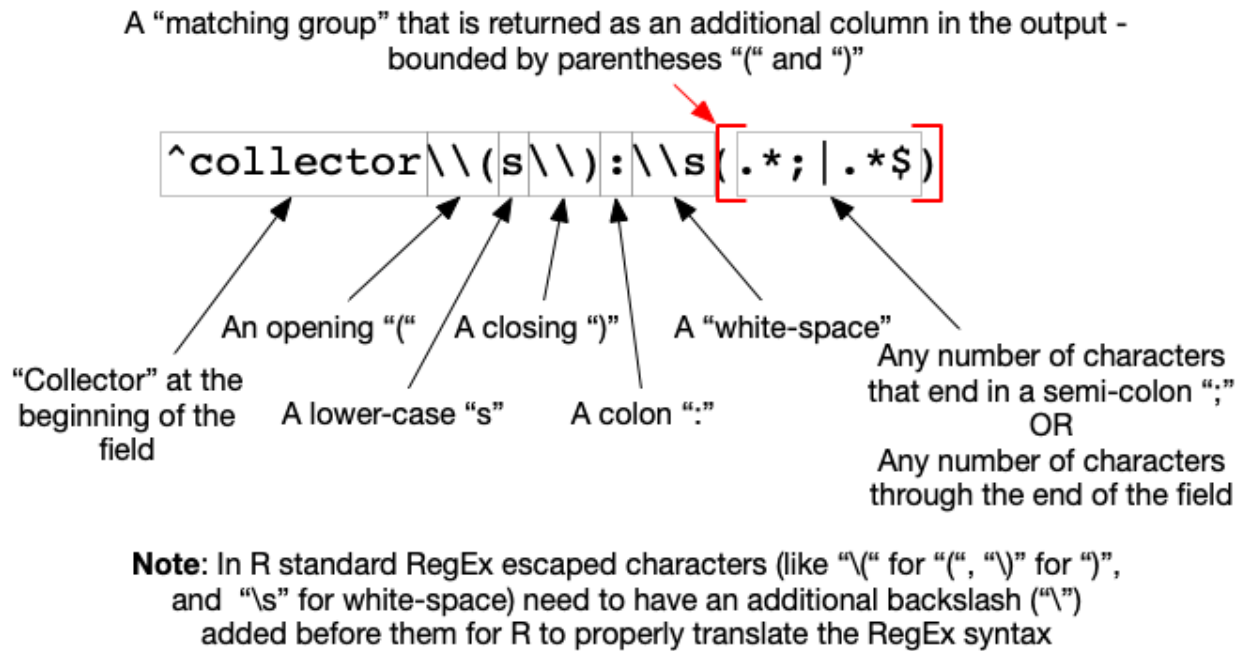


Figure 2: Description of the regular expression used to extract targeted substrings from the combined “recordedBy” field

```
## Define and use some R regular expressions for extracting text from the
## recordedBy column
collectorExtract <- "^collector\\(s\\):\\s(\\. *;|\\. *$)"
preparatorExtract <- "preparator\\(s\\):\\s(\\. *;|\\. *$)"

collector_string <- str_match(rawData$recordedBy, collectorExtract)
preparator_string <- str_match(rawData$recordedBy, preparatorExtract)

print(head(collector_string))
```

```
##      [,1]
## [1,] "collector(s): ana lilia trujano álvarez, eric ghilarducci"
## [2,] "collector(s): william z. lidicker jr."
## [3,] "collector(s): william z. lidicker jr."
## [4,] "collector(s): william z. lidicker jr."
## [5,] "collector(s): william z. lidicker jr."
## [6,] "collector(s): tom manning;"
##      [,2]
## [1,] "ana lilia trujano álvarez, eric ghilarducci"
## [2,] "william z. lidicker jr."
## [3,] "william z. lidicker jr."
## [4,] "william z. lidicker jr."
## [5,] "william z. lidicker jr."
## [6,] "tom manning;"
```

```
print(head(preparator_string))
```

```
##           [,1]                [,2]
## [1,] NA                      NA
## [2,] NA                      NA
## [3,] NA                      NA
## [4,] NA                      NA
## [5,] NA                      NA
## [6,] "preparator(s): amber baxter" "amber baxter"
```

```
rawData$collectors <- collector_string[,2]
rawData$preparators <- preparator_string[,2]
```

*# check the first ten rows to see what the output looks like*

```
head(rawData, n=10) %>%
  select(recordedBy, collectors, preparators)
```

```
## # A tibble: 10 x 3
##   recordedBy                collectors                preparators
##   <chr>                  <chr>                  <chr>
## 1 collector(s): ana lilia trujano álv~ ana lilia trujano álvarez, ~ <NA>
## 2 collector(s): william z. lidicker j~ william z. lidicker jr.    <NA>
## 3 collector(s): william z. lidicker j~ william z. lidicker jr.    <NA>
## 4 collector(s): william z. lidicker j~ william z. lidicker jr.    <NA>
## 5 collector(s): william z. lidicker j~ william z. lidicker jr.    <NA>
## 6 collector(s): tom manning; preparat~ tom manning;            amber baxt~
## 7 caldwell, j. p. and vitt, l. j.    <NA>                  <NA>
## 8 collector(s): james l. patton        james l. patton          <NA>
## 9 collector(s): troy l. best; prepara~ troy l. best;           troy l. be~
## 10 collector(s): karl j. martin; prepa~ karl j. martin;         paul ollig
```

What would the next logical step in the process be for cleaning up the *recordedBy* or newly generated columns?

Breaking apart the latDMS and lonDMS columns into their constituent parts

```
## Define a regular expression and use it to extract pieces from a DMS string
dmsExtract <- "\\s*(-*[:digit:]+)°\\s*([[:digit:]]+)'\\s*([[:digit:]]+)'\\s*([[:digit:]]+)'\\s*([[:digit:]]+)"
```

```
latSubstrings <- str_match(rawData$latDMS, dmsExtract)
print(head(latSubstrings))
```

```
##           [,1]                [,2] [,3] [,4]
## [1,] "37°45'39.430" "37" "45" "39.430"
## [2,] "37°53'59.845" "37" "53" "59.845"
## [3,] "37°53'59.845" "37" "53" "59.845"
## [4,] "37°53'59.845" "37" "53" "59.845"
## [5,] "37°53'59.845" "37" "53" "59.845"
## [6,] "43°16'30.427" "43" "16" "30.427"
```

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

```
lonSubstrings <- str_match(rawData$lonDMS, dmsExtract)
print(head(lonSubstrings))
```

```
##           [,1]                [,2] [,3] [,4]
```



```
## [1,] "-122° 6'48.370" "-122" "6" "48.370"
## [2,] "-123°38'18.039" "-123" "38" "18.039"
## [3,] "-123°38'18.039" "-123" "38" "18.039"
## [4,] "-123°38'18.039" "-123" "38" "18.039"
## [5,] "-123°38'18.039" "-123" "38" "18.039"
## [6,] "-123°12'32.023" "-123" "12" "32.023"
```

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

```
head(rawData, n=10) %>%
  select(latDMS, latD, latM, latS,
         lonDMS, lonD, lonM, lonS)
```

```
## # A tibble: 10 x 8
##   latDMS      latD latM latS lonDMS      lonD lonM lonS
##   <chr>      <dbl> <dbl> <dbl> <chr>      <dbl> <dbl> <dbl>
## 1 "37°45'39.430"N" 37    45 39.4 "-122° 6'48.370"E" -122    6 48.4
## 2 "37°53'59.845"N" 37    53 59.8 "-123°38'18.039"E" -123   38 18.0
## 3 "37°53'59.845"N" 37    53 59.8 "-123°38'18.039"E" -123   38 18.0
## 4 "37°53'59.845"N" 37    53 59.8 "-123°38'18.039"E" -123   38 18.0
## 5 "37°53'59.845"N" 37    53 59.8 "-123°38'18.039"E" -123   38 18.0
## 6 "43°16'30.427"N" 43    16 30.4 "-123°12'32.023"E" -123   12 32.0
## 7 "34°32'20.508"N" 34    32 20.5 "-95° 6'42.444"E"  -95    6 42.4
## 8 "36°58'15.564"N" 36    58 15.6 "-119°54'16.740"E" -119   54 16.7
## 9 "36°35' 5.813"N" 36    35  5.81 "-108° 0'55.757"E" -108    0 55.8
## 10 "44°15'40.000"N" 44    15 40    "-124°25' 1.000"E" -124   25  1
```

## 5. Check value ranges and explore data

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

Checking the `weight` and `length` columns.

```
##### Check value ranges and explore data #####
```

```
## assert GitHub repository with instructions and examples:
```

```
##   https://github.com/ropensci/assertr
```

```
library(assertr)
```

```
## you can just run this if you want execution of your workflow to stop if
```

```
## any of your tests fail
```

```
# rawData %>%
```

```
#   chain_start %>%
```

```
#   assert(within_bounds(1,Inf), weight) %>% # assert checks individual values
```

```
#   assert(within_bounds(1,Inf), length) %>%
```

```
#   insist(within_n_sds(3), weight) %>% # insist checks against calculated vals
```

```
#   insist(within_n_sds(3), length) %>%
```

```
#   chain_end
```

```
## you can run this if you want to prevent the errors that are generated
```

```
## by your tests from halting execution of your workflow
```

```

tryCatch({rawData %>%
  chain_start %>%
  assert(within_bounds(1,Inf), weight) %>% # assert checks individual values
  assert(within_bounds(1,Inf), length) %>%
  insist(within_n_sds(3), weight) %>% # insist checks against calculated vals
  insist(within_n_sds(3), length) %>%
  chain_end
}, warning = function(w) {
  paste("A warning was generated: ", w, sep = "")
}, error = function(e) {
  print(e)
}, finally = {
  print("this is the end of the validation check ...")
}
)

```

## There are 312 errors across 4 verbs:

```

## -
##      verb redux_fn      predicate column index  value
## 1  assert      NA within_bounds(1, Inf) weight  2772   0.00
## 2  assert      NA within_bounds(1, Inf) weight  2801   0.00
## 3  assert      NA within_bounds(1, Inf) weight  3118   0.00
## 4  assert      NA within_bounds(1, Inf) weight  3159   0.00
## 5  assert      NA within_bounds(1, Inf) weight  5417   0.00
## 6  assert      NA within_bounds(1, Inf) weight  6788   0.00
## 7  assert      NA within_bounds(1, Inf) weight  7334   0.00
## 8  assert      NA within_bounds(1, Inf) weight  8309   0.00
## 9  assert      NA within_bounds(1, Inf) weight  9539   0.00
## 10 assert      NA within_bounds(1, Inf) weight 10040   0.00
## 11 assert      NA within_bounds(1, Inf) weight 10628   0.00
## 12 assert      NA within_bounds(1, Inf) weight 10708   0.00
## 13 assert      NA within_bounds(1, Inf) length  2801   0.00
## 14 assert      NA within_bounds(1, Inf) length  8896   0.00
## 15 assert      NA within_bounds(1, Inf) length 10628   0.00
## 16 insist      NA      within_n_sds(3) weight    52 269.00
## 17 insist      NA      within_n_sds(3) weight   117 142.00
## 18 insist      NA      within_n_sds(3) weight   162 164.70
## 19 insist      NA      within_n_sds(3) weight   256 135.00
## 20 insist      NA      within_n_sds(3) weight   258 215.00
## 21 insist      NA      within_n_sds(3) weight   316 155.20
## 22 insist      NA      within_n_sds(3) weight   757 317.19
## 23 insist      NA      within_n_sds(3) weight   764 198.00
## 24 insist      NA      within_n_sds(3) weight   777 138.00
## 25 insist      NA      within_n_sds(3) weight   839 135.00
## 26 insist      NA      within_n_sds(3) weight   855 160.00
## 27 insist      NA      within_n_sds(3) weight   886 152.00
## 28 insist      NA      within_n_sds(3) weight   891 157.00
## 29 insist      NA      within_n_sds(3) weight   971 146.00
## 30 insist      NA      within_n_sds(3) weight   994 235.00
## 31 insist      NA      within_n_sds(3) weight  1035 157.00
## 32 insist      NA      within_n_sds(3) weight  1125 165.00
## 33 insist      NA      within_n_sds(3) weight  1168 153.00
## 34 insist      NA      within_n_sds(3) weight  1217 145.80
## 35 insist      NA      within_n_sds(3) weight  1244 142.00

```

## 36	insist	NA	within_n_sds(3)	weight	1283	140.00
## 37	insist	NA	within_n_sds(3)	weight	1418	159.00
## 38	insist	NA	within_n_sds(3)	weight	1575	142.00
## 39	insist	NA	within_n_sds(3)	weight	1581	146.00
## 40	insist	NA	within_n_sds(3)	weight	1625	161.00
## 41	insist	NA	within_n_sds(3)	weight	1628	141.00
## 42	insist	NA	within_n_sds(3)	weight	1688	168.00
## 43	insist	NA	within_n_sds(3)	weight	1754	150.00
## 44	insist	NA	within_n_sds(3)	weight	1780	240.20
## 45	insist	NA	within_n_sds(3)	weight	1945	150.00
## 46	insist	NA	within_n_sds(3)	weight	2000	226.00
## 47	insist	NA	within_n_sds(3)	weight	2076	170.00
## 48	insist	NA	within_n_sds(3)	weight	2109	152.00
## 49	insist	NA	within_n_sds(3)	weight	2165	172.00
## 50	insist	NA	within_n_sds(3)	weight	2183	141.00
## 51	insist	NA	within_n_sds(3)	weight	2186	139.00
## 52	insist	NA	within_n_sds(3)	weight	2207	237.00
## 53	insist	NA	within_n_sds(3)	weight	2242	155.00
## 54	insist	NA	within_n_sds(3)	weight	2266	140.00
## 55	insist	NA	within_n_sds(3)	weight	2279	167.00
## 56	insist	NA	within_n_sds(3)	weight	2370	211.00
## 57	insist	NA	within_n_sds(3)	weight	2469	140.00
## 58	insist	NA	within_n_sds(3)	weight	2483	144.80
## 59	insist	NA	within_n_sds(3)	weight	2510	230.00
## 60	insist	NA	within_n_sds(3)	weight	2529	156.00
## 61	insist	NA	within_n_sds(3)	weight	2535	150.00
## 62	insist	NA	within_n_sds(3)	weight	2708	240.00
## 63	insist	NA	within_n_sds(3)	weight	2727	253.00
## 64	insist	NA	within_n_sds(3)	weight	2794	162.00
## 65	insist	NA	within_n_sds(3)	weight	2836	236.00
## 66	insist	NA	within_n_sds(3)	weight	2948	162.00
## 67	insist	NA	within_n_sds(3)	weight	2962	181.00
## 68	insist	NA	within_n_sds(3)	weight	3004	178.10
## 69	insist	NA	within_n_sds(3)	weight	3017	156.00
## 70	insist	NA	within_n_sds(3)	weight	3026	160.00
## 71	insist	NA	within_n_sds(3)	weight	3030	140.00
## 72	insist	NA	within_n_sds(3)	weight	3074	252.00
## 73	insist	NA	within_n_sds(3)	weight	3108	139.00
## 74	insist	NA	within_n_sds(3)	weight	3125	140.00
## 75	insist	NA	within_n_sds(3)	weight	3143	138.10
## 76	insist	NA	within_n_sds(3)	weight	3216	140.00
## 77	insist	NA	within_n_sds(3)	weight	3222	166.00
## 78	insist	NA	within_n_sds(3)	weight	3244	135.00
## 79	insist	NA	within_n_sds(3)	weight	3309	153.00
## 80	insist	NA	within_n_sds(3)	weight	3332	160.00
## 81	insist	NA	within_n_sds(3)	weight	3399	169.00
## 82	insist	NA	within_n_sds(3)	weight	3447	140.00
## 83	insist	NA	within_n_sds(3)	weight	3486	135.00
## 84	insist	NA	within_n_sds(3)	weight	3496	160.00
## 85	insist	NA	within_n_sds(3)	weight	3526	370.00
## 86	insist	NA	within_n_sds(3)	weight	3538	151.00
## 87	insist	NA	within_n_sds(3)	weight	3557	216.00
## 88	insist	NA	within_n_sds(3)	weight	3583	230.60
## 89	insist	NA	within_n_sds(3)	weight	3630	135.00

## 90	insist	NA	within_n_sds(3)	weight	3647	152.00
## 91	insist	NA	within_n_sds(3)	weight	3724	157.00
## 92	insist	NA	within_n_sds(3)	weight	3924	290.00
## 93	insist	NA	within_n_sds(3)	weight	3968	165.00
## 94	insist	NA	within_n_sds(3)	weight	4026	237.00
## 95	insist	NA	within_n_sds(3)	weight	4051	146.00
## 96	insist	NA	within_n_sds(3)	weight	4284	245.00
## 97	insist	NA	within_n_sds(3)	weight	4325	135.00
## 98	insist	NA	within_n_sds(3)	weight	4377	165.00
## 99	insist	NA	within_n_sds(3)	weight	4491	151.60
## 100	insist	NA	within_n_sds(3)	weight	4502	173.00
## 101	insist	NA	within_n_sds(3)	weight	4690	205.00
## 102	insist	NA	within_n_sds(3)	weight	4728	166.30
## 103	insist	NA	within_n_sds(3)	weight	4733	158.20
## 104	insist	NA	within_n_sds(3)	weight	4768	248.00
## 105	insist	NA	within_n_sds(3)	weight	4962	146.00
## 106	insist	NA	within_n_sds(3)	weight	5055	177.00
## 107	insist	NA	within_n_sds(3)	weight	5106	190.00
## 108	insist	NA	within_n_sds(3)	weight	5120	136.10
## 109	insist	NA	within_n_sds(3)	weight	5219	151.00
## 110	insist	NA	within_n_sds(3)	weight	5275	140.00
## 111	insist	NA	within_n_sds(3)	weight	5283	264.90
## 112	insist	NA	within_n_sds(3)	weight	5342	146.00
## 113	insist	NA	within_n_sds(3)	weight	5355	146.00
## 114	insist	NA	within_n_sds(3)	weight	5398	158.50
## 115	insist	NA	within_n_sds(3)	weight	5510	148.00
## 116	insist	NA	within_n_sds(3)	weight	5618	146.50
## 117	insist	NA	within_n_sds(3)	weight	5775	145.00
## 118	insist	NA	within_n_sds(3)	weight	5789	155.00
## 119	insist	NA	within_n_sds(3)	weight	5799	157.00
## 120	insist	NA	within_n_sds(3)	weight	5817	134.00
## 121	insist	NA	within_n_sds(3)	weight	5837	250.00
## 122	insist	NA	within_n_sds(3)	weight	5841	165.00
## 123	insist	NA	within_n_sds(3)	weight	5900	170.00
## 124	insist	NA	within_n_sds(3)	weight	5914	167.10
## 125	insist	NA	within_n_sds(3)	weight	5924	265.00
## 126	insist	NA	within_n_sds(3)	weight	6037	174.00
## 127	insist	NA	within_n_sds(3)	weight	6147	195.00
## 128	insist	NA	within_n_sds(3)	weight	6174	154.00
## 129	insist	NA	within_n_sds(3)	weight	6181	213.00
## 130	insist	NA	within_n_sds(3)	weight	6203	177.00
## 131	insist	NA	within_n_sds(3)	weight	6270	219.00
## 132	insist	NA	within_n_sds(3)	weight	6291	138.00
## 133	insist	NA	within_n_sds(3)	weight	6305	160.00
## 134	insist	NA	within_n_sds(3)	weight	6342	145.00
## 135	insist	NA	within_n_sds(3)	weight	6382	148.00
## 136	insist	NA	within_n_sds(3)	weight	6426	135.00
## 137	insist	NA	within_n_sds(3)	weight	6441	135.00
## 138	insist	NA	within_n_sds(3)	weight	6608	200.00
## 139	insist	NA	within_n_sds(3)	weight	6621	145.00
## 140	insist	NA	within_n_sds(3)	weight	6626	220.00
## 141	insist	NA	within_n_sds(3)	weight	6748	190.00
## 142	insist	NA	within_n_sds(3)	weight	6760	155.00
## 143	insist	NA	within_n_sds(3)	weight	6778	135.00

## 144	insist	NA	within_n_sds(3)	weight	6779	144.00
## 145	insist	NA	within_n_sds(3)	weight	6869	194.90
## 146	insist	NA	within_n_sds(3)	weight	7062	135.00
## 147	insist	NA	within_n_sds(3)	weight	7110	250.00
## 148	insist	NA	within_n_sds(3)	weight	7118	205.00
## 149	insist	NA	within_n_sds(3)	weight	7169	143.00
## 150	insist	NA	within_n_sds(3)	weight	7172	149.10
## 151	insist	NA	within_n_sds(3)	weight	7193	168.50
## 152	insist	NA	within_n_sds(3)	weight	7223	215.00
## 153	insist	NA	within_n_sds(3)	weight	7232	183.00
## 154	insist	NA	within_n_sds(3)	weight	7411	240.00
## 155	insist	NA	within_n_sds(3)	weight	7443	136.00
## 156	insist	NA	within_n_sds(3)	weight	7551	148.00
## 157	insist	NA	within_n_sds(3)	weight	7556	156.20
## 158	insist	NA	within_n_sds(3)	weight	7626	251.00
## 159	insist	NA	within_n_sds(3)	weight	7798	143.00
## 160	insist	NA	within_n_sds(3)	weight	7844	171.00
## 161	insist	NA	within_n_sds(3)	weight	7915	158.00
## 162	insist	NA	within_n_sds(3)	weight	7929	149.00
## 163	insist	NA	within_n_sds(3)	weight	7932	170.00
## 164	insist	NA	within_n_sds(3)	weight	7944	135.00
## 165	insist	NA	within_n_sds(3)	weight	7957	160.00
## 166	insist	NA	within_n_sds(3)	weight	7958	250.00
## 167	insist	NA	within_n_sds(3)	weight	7983	134.30
## 168	insist	NA	within_n_sds(3)	weight	7987	190.00
## 169	insist	NA	within_n_sds(3)	weight	8110	150.00
## 170	insist	NA	within_n_sds(3)	weight	8193	145.00
## 171	insist	NA	within_n_sds(3)	weight	8212	192.00
## 172	insist	NA	within_n_sds(3)	weight	8258	148.00
## 173	insist	NA	within_n_sds(3)	weight	8280	145.00
## 174	insist	NA	within_n_sds(3)	weight	8335	154.50
## 175	insist	NA	within_n_sds(3)	weight	8355	145.00
## 176	insist	NA	within_n_sds(3)	weight	8517	145.00
## 177	insist	NA	within_n_sds(3)	weight	8547	230.00
## 178	insist	NA	within_n_sds(3)	weight	8635	153.00
## 179	insist	NA	within_n_sds(3)	weight	8692	310.20
## 180	insist	NA	within_n_sds(3)	weight	8704	168.00
## 181	insist	NA	within_n_sds(3)	weight	8781	172.00
## 182	insist	NA	within_n_sds(3)	weight	8848	148.00
## 183	insist	NA	within_n_sds(3)	weight	8962	165.00
## 184	insist	NA	within_n_sds(3)	weight	9004	136.00
## 185	insist	NA	within_n_sds(3)	weight	9021	146.00
## 186	insist	NA	within_n_sds(3)	weight	9053	148.10
## 187	insist	NA	within_n_sds(3)	weight	9110	235.00
## 188	insist	NA	within_n_sds(3)	weight	9194	145.00
## 189	insist	NA	within_n_sds(3)	weight	9226	155.00
## 190	insist	NA	within_n_sds(3)	weight	9248	168.00
## 191	insist	NA	within_n_sds(3)	weight	9288	150.00
## 192	insist	NA	within_n_sds(3)	weight	9520	145.50
## 193	insist	NA	within_n_sds(3)	weight	9548	238.00
## 194	insist	NA	within_n_sds(3)	weight	9610	145.00
## 195	insist	NA	within_n_sds(3)	weight	9684	141.00
## 196	insist	NA	within_n_sds(3)	weight	9712	144.00
## 197	insist	NA	within_n_sds(3)	weight	10000	186.00

## 198	insist	NA	within_n_sds(3)	weight	10262	148.00
## 199	insist	NA	within_n_sds(3)	weight	10276	141.90
## 200	insist	NA	within_n_sds(3)	weight	10310	161.00
## 201	insist	NA	within_n_sds(3)	weight	10316	133.70
## 202	insist	NA	within_n_sds(3)	weight	10400	144.60
## 203	insist	NA	within_n_sds(3)	weight	10583	310.00
## 204	insist	NA	within_n_sds(3)	weight	10676	148.00
## 205	insist	NA	within_n_sds(3)	weight	10765	142.00
## 206	insist	NA	within_n_sds(3)	length	29	350.00
## 207	insist	NA	within_n_sds(3)	length	150	19.40
## 208	insist	NA	within_n_sds(3)	length	259	356.00
## 209	insist	NA	within_n_sds(3)	length	345	386.00
## 210	insist	NA	within_n_sds(3)	length	697	342.00
## 211	insist	NA	within_n_sds(3)	length	704	341.00
## 212	insist	NA	within_n_sds(3)	length	839	351.00
## 213	insist	NA	within_n_sds(3)	length	886	344.00
## 214	insist	NA	within_n_sds(3)	length	1002	363.00
## 215	insist	NA	within_n_sds(3)	length	1232	342.00
## 216	insist	NA	within_n_sds(3)	length	1490	344.00
## 217	insist	NA	within_n_sds(3)	length	1567	373.00
## 218	insist	NA	within_n_sds(3)	length	1575	349.00
## 219	insist	NA	within_n_sds(3)	length	1628	373.00
## 220	insist	NA	within_n_sds(3)	length	1671	345.00
## 221	insist	NA	within_n_sds(3)	length	2121	343.00
## 222	insist	NA	within_n_sds(3)	length	2146	357.00
## 223	insist	NA	within_n_sds(3)	length	2183	358.00
## 224	insist	NA	within_n_sds(3)	length	2242	350.00
## 225	insist	NA	within_n_sds(3)	length	2243	368.00
## 226	insist	NA	within_n_sds(3)	length	2292	10.70
## 227	insist	NA	within_n_sds(3)	length	2397	341.00
## 228	insist	NA	within_n_sds(3)	length	2449	337.00
## 229	insist	NA	within_n_sds(3)	length	2529	358.00
## 230	insist	NA	within_n_sds(3)	length	2801	0.00
## 231	insist	NA	within_n_sds(3)	length	3026	381.00
## 232	insist	NA	within_n_sds(3)	length	3030	353.00
## 233	insist	NA	within_n_sds(3)	length	3059	337.00
## 234	insist	NA	within_n_sds(3)	length	3216	363.00
## 235	insist	NA	within_n_sds(3)	length	3274	336.00
## 236	insist	NA	within_n_sds(3)	length	3323	361.00
## 237	insist	NA	within_n_sds(3)	length	3402	355.00
## 238	insist	NA	within_n_sds(3)	length	3500	19.50
## 239	insist	NA	within_n_sds(3)	length	3526	370.00
## 240	insist	NA	within_n_sds(3)	length	3579	8.90
## 241	insist	NA	within_n_sds(3)	length	3581	359.00
## 242	insist	NA	within_n_sds(3)	length	3630	377.00
## 243	insist	NA	within_n_sds(3)	length	3668	350.00
## 244	insist	NA	within_n_sds(3)	length	4002	338.00
## 245	insist	NA	within_n_sds(3)	length	4225	348.00
## 246	insist	NA	within_n_sds(3)	length	4325	350.00
## 247	insist	NA	within_n_sds(3)	length	4377	344.00
## 248	insist	NA	within_n_sds(3)	length	4391	343.00
## 249	insist	NA	within_n_sds(3)	length	4406	335.00
## 250	insist	NA	within_n_sds(3)	length	4634	352.00
## 251	insist	NA	within_n_sds(3)	length	4709	352.00

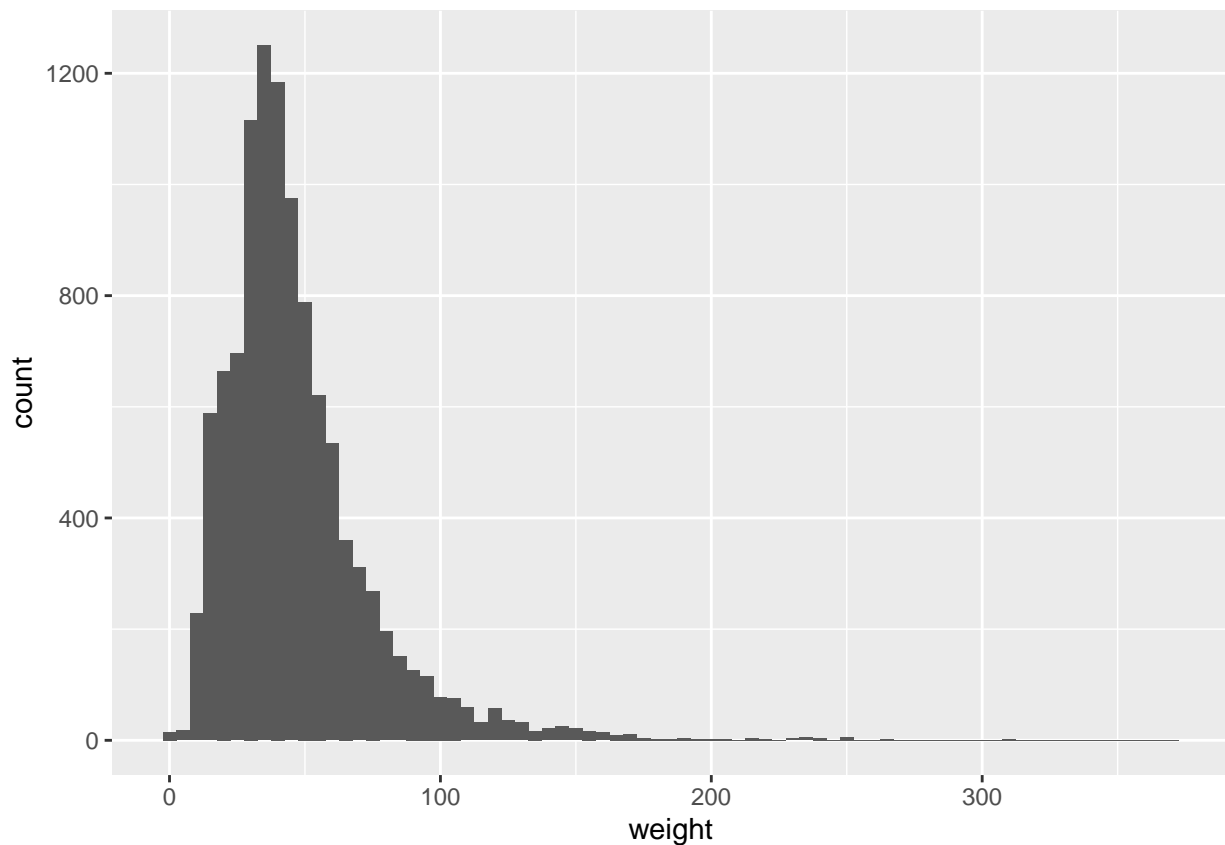
## 252	insist	NA	within_n_sds(3)	length	4757	357.00
## 253	insist	NA	within_n_sds(3)	length	4885	362.00
## 254	insist	NA	within_n_sds(3)	length	4912	19.20
## 255	insist	NA	within_n_sds(3)	length	4948	355.00
## 256	insist	NA	within_n_sds(3)	length	5067	22.00
## 257	insist	NA	within_n_sds(3)	length	5275	357.00
## 258	insist	NA	within_n_sds(3)	length	5299	28.00
## 259	insist	NA	within_n_sds(3)	length	5329	364.00
## 260	insist	NA	within_n_sds(3)	length	5398	384.00
## 261	insist	NA	within_n_sds(3)	length	5491	341.00
## 262	insist	NA	within_n_sds(3)	length	5625	337.00
## 263	insist	NA	within_n_sds(3)	length	5715	32.00
## 264	insist	NA	within_n_sds(3)	length	5760	335.00
## 265	insist	NA	within_n_sds(3)	length	5843	12.50
## 266	insist	NA	within_n_sds(3)	length	5912	18.40
## 267	insist	NA	within_n_sds(3)	length	5914	371.00
## 268	insist	NA	within_n_sds(3)	length	6113	335.00
## 269	insist	NA	within_n_sds(3)	length	6215	337.00
## 270	insist	NA	within_n_sds(3)	length	6291	343.00
## 271	insist	NA	within_n_sds(3)	length	6342	353.00
## 272	insist	NA	within_n_sds(3)	length	6522	341.00
## 273	insist	NA	within_n_sds(3)	length	6621	364.00
## 274	insist	NA	within_n_sds(3)	length	6622	346.00
## 275	insist	NA	within_n_sds(3)	length	6764	16.00
## 276	insist	NA	within_n_sds(3)	length	6953	19.50
## 277	insist	NA	within_n_sds(3)	length	6999	20.50
## 278	insist	NA	within_n_sds(3)	length	7020	357.00
## 279	insist	NA	within_n_sds(3)	length	7369	335.00
## 280	insist	NA	within_n_sds(3)	length	7425	340.00
## 281	insist	NA	within_n_sds(3)	length	7453	351.00
## 282	insist	NA	within_n_sds(3)	length	7506	356.00
## 283	insist	NA	within_n_sds(3)	length	7535	345.00
## 284	insist	NA	within_n_sds(3)	length	7640	353.00
## 285	insist	NA	within_n_sds(3)	length	7844	362.00
## 286	insist	NA	within_n_sds(3)	length	8003	338.00
## 287	insist	NA	within_n_sds(3)	length	8193	351.00
## 288	insist	NA	within_n_sds(3)	length	8405	344.00
## 289	insist	NA	within_n_sds(3)	length	8527	353.00
## 290	insist	NA	within_n_sds(3)	length	8557	358.00
## 291	insist	NA	within_n_sds(3)	length	8713	352.00
## 292	insist	NA	within_n_sds(3)	length	8794	19.00
## 293	insist	NA	within_n_sds(3)	length	8896	0.00
## 294	insist	NA	within_n_sds(3)	length	9021	354.00
## 295	insist	NA	within_n_sds(3)	length	9053	335.00
## 296	insist	NA	within_n_sds(3)	length	9288	353.00
## 297	insist	NA	within_n_sds(3)	length	9461	358.00
## 298	insist	NA	within_n_sds(3)	length	9520	374.00
## 299	insist	NA	within_n_sds(3)	length	9542	353.00
## 300	insist	NA	within_n_sds(3)	length	9604	340.00
## 301	insist	NA	within_n_sds(3)	length	9621	355.00
## 302	insist	NA	within_n_sds(3)	length	9689	357.00
## 303	insist	NA	within_n_sds(3)	length	9712	355.00
## 304	insist	NA	within_n_sds(3)	length	9981	345.00
## 305	insist	NA	within_n_sds(3)	length	10108	18.80

```
## 306 insist      NA      within_n_sds(3) length 10243 337.00
## 307 insist      NA      within_n_sds(3) length 10262 353.00
## 308 insist      NA      within_n_sds(3) length 10473 347.00
## 309 insist      NA      within_n_sds(3) length 10533 345.00
## 310 insist      NA      within_n_sds(3) length 10615 362.00
## 311 insist      NA      within_n_sds(3) length 10628  0.00
## 312 insist      NA      within_n_sds(3) length 10721 373.00
##
## <simpleError: assertr stopped execution>
## [1] "this is the end of the validation check ..."
```

```
## Visual assessment of your data
```

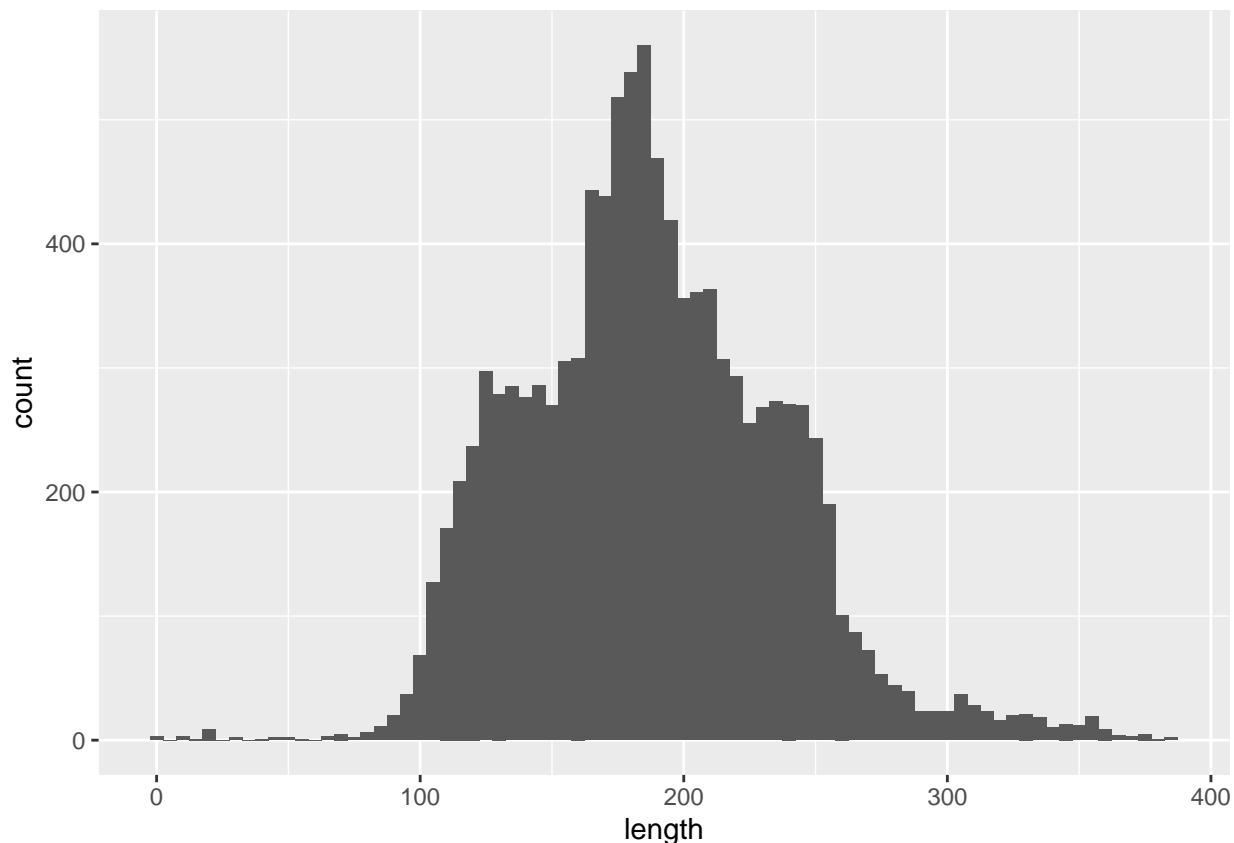
```
library(ggplot2)
```

```
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

```
#####
##### Bringing it all together into a single series of commands #####

# 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(.*;|.*)$"
preparatorExtract <- "preparator\\(s\\):\\s(.*;|.*)$"

collector_string <- str_match(rawData$recordedBy, collectorExtract)
preparator_string <- str_match(rawData$recordedBy, preparatorExtract)

rawData$collectors <- collector_string[,2]
rawData$preparators <- preparator_string[,2]

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

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(analysisData)

```

```

## Rows: 10,767
## Columns: 24
## $ X1          <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,...
## $ uuid        <chr> "060380ea-7b06-474e-8d2e-b6e4a8c21e1a", "0fb17a79-...
## $ institutionCode <chr> "mvz", "mvz", "mvz", "mvz", "mvz", "uam", "omnh", ...
## $ collectionCode <chr> "mammal specimens", "mammal specimens", "mammal sp...
## $ catalogNumber  <chr> "219088", "233524", "234346", "233951", "235290", ...
## $ recordedBy     <chr> "collector(s): ana lilia trujano álvarez, eric ghi...
## $ countryCode    <chr> "usa", "usa", "usa", "usa", "usa", "usa", "usa", "...
## $ stateProvince  <chr> "california", "california", "california", "califor...
## $ county         <chr> "contra costa county", "contra costa county", "con...
## $ decimalLatitude <dbl> 37.76095, 37.89996, 37.89996, 37.89996, 37.89996, ...
## $ decimalLongitude <dbl> -121.88656, -122.36166, -122.36166, -122.36166, -1...
## $ eventDate      <dtm> 2005-11-23, 1959-06-21, 1962-11-22, 1960-07-31, 1...
## $ year           <dbl> 2005, 1959, 1962, 1960, 1964, 1996, 2011, 2005, 19...
## $ month          <dbl> 11, 6, 11, 7, 7, 10, 1, 8, 6, 5, 8, 8, 11, 6, 7, 6...
## $ day            <dbl> 22, 20, 21, 30, 3, 22, 17, 6, 4, 19, 10, 12, 30, 1...
## $ genus          <chr> "microtus", "microtus", "microtus", "microtus", "m...
## $ specificEpithet <chr> "californicus", "californicus", "californicus", "c...
## $ scientificName  <chr> "microtus californicus californicus", "microtus ca...
## $ weight          <dbl> 30.5, 22.0, 49.0, 33.0, 29.0, 23.5, 24.0, 27.0, 12...
## $ length          <dbl> 165, 143, 187, 169, 159, 141, 121, 176, 294, 110, ...
## $ sex             <chr> "male", "female", "male", "female", "female", "fem...
## $ latDMS          <chr> "37°45'39.430\"N", "37°53'59.845\"N", "37°53'59.84...
## $ lonDMS          <chr> "-122° 6'48.370\"E", "-123°38'18.039\"E", "-123°38...
## $ textLatDD       <dbl> 37.76095, 37.89996, 37.89996, 37.89996, 37.89996, ...

```

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)
```