5-Day Challenge: Data Cleaning in R

**Simon Tran**

Data cleaning is a key part of data science, but it can be deeply frustrating. What are you supposed to do with the .json file you’ve been sent? How can you handle all these missing values in your data? Is there a fast way to get rid of the duplicate entries in your dataset? In this challenge, I’ll tackle some of the common problems that I need to take care of before I can get started with my analysis.

Here’s what I’ll be covering in this 5-Day Challenge:

* Reading in common data file formats: .json and .xlsx
* Filling in missing values
* Identifying & handling outliers
* Removing duplicate records  
  \*Cleaning numbers (percentages, money, dates, and times)

For each day of this challenge, I’ll be using different datasets to perform these tasks.

## 

## Day 1: Reading in Different File Types and Understanding Their Structure

library(tidyverse)  
library(readxl) # for reading in xl files  
library(jsonlite) # for reading in json

Let’s first read in a **JSON file**

house <- read\_json('house\_3.json')

The first dataset I’ll be working on is from **Tbilisi Housing Challenge 2020** on kaggle. The data was extracted from a popular Georgian house retail site on November 12th of 2020. The JSON file represents the ‘raw’ unclean data, straight from over 50k html pages.

Next, let’s take a look at the data structure

head(house,3)

[[1]]  
[[1]]$address  
[1] "შარტავას ქუჩა, Saburtalo, Saburtalo District, Tbilisi"  
  
[[1]]$product\_tree  
[1] "Newly finished apartment for sale,Tbilisi,Saburtalo District,Saburtalo,1 rooms"  
  
[[1]]$time  
[1] "Today 12:33"  
  
[[1]]$views  
[1] "622"  
  
[[1]]$title  
[1] "Newly finished apartment for sale"  
  
[[1]]$id  
[1] ": 10541934"  
  
[[1]]$price\_gel  
[1] "107,100"  
  
[[1]]$price\_usd  
[1] "32,500"  
  
[[1]]$space  
[1] "Area: 28.00 m² "  
  
[[1]]$room  
[1] "1 Room"  
  
[[1]]$bedroom  
[1] "Bedroom"  
  
[[1]]$floor  
[1] "11/11"  
  
[[1]]$description  
[1] "1,5 Ot newly renovated apartment Non-residential, newly built, 28 sqm, 11 (11) (elevator up to 10) living room studio, bedroom and balcony. View of Mtatsminda, on Shartava."  
  
[[1]]$amenities  
[1] "Newly renovated,Nonstandard,Ceiling height,3.00 M,Bedroom 1,Balcony,Bathrooms 1,Heating,Central Heating System,Hot water,Central Heating System,Gas,Passenger elevator"  
  
[[1]]$latitude  
[1] "41.7245209"  
  
[[1]]$longitude  
[1] "44.7537885"  
  
[[1]]$poster\_type  
[1] "Agent"  
  
[[1]]$poster\_id  
[1] "2165179"  
  
  
[[2]]  
[[2]]$address  
[1] "Rustavi, Lower Kartli"  
  
[[2]]$product\_tree  
[1] "Older finished apartment for sale,Lower Kartli,Rustavi,3 rooms"  
  
[[2]]$time  
[1] "Today 11:40"  
  
[[2]]$views  
[1] "276"  
  
[[2]]$title  
[1] "Older finished apartment for sale"  
  
[[2]]$id  
[1] ": 10680693"  
  
[[2]]$price\_gel  
[1] "87,300"  
  
[[2]]$price\_usd  
[1] "26,500"  
  
[[2]]$space  
[1] "Area: 63.00 m² "  
  
[[2]]$room  
[1] "3 Room"  
  
[[2]]$bedroom  
[1] "Bedroom"  
  
[[2]]$floor  
[1] "5/9"  
  
[[2]]$description  
[1] "One-room 3-room apartment for sale on the 5th 5 (9) floor in 19 Rustavi, Rustavi. Iron doors, metal windows, toilet tile flooring, have a loggia as well. The last price is $ 26,500. We have a large selection of apartments, contact us. Tel: 592412552 (Agency, be sure to ask &quot;Misha&quot;) We have a large selection of apartments, contact us."  
  
[[2]]$amenities  
[1] "Newly renovated,Moskow,Ceiling height,2.70 M,Bedroom 2,Bathrooms 1"  
  
[[2]]$latitude  
[1] "41.54309"  
  
[[2]]$longitude  
[1] "45.0112804"  
  
[[2]]$poster\_type  
[1] "Agent"  
  
[[2]]$poster\_id  
[1] "3594609"  
  
  
[[3]]  
[[3]]$address  
[1] "Queen Ketevan Avenue, Isani, Isani District, Tbilisi"  
  
[[3]]$product\_tree  
[1] "Newly finished apartment for sale,Tbilisi,Isani District,Isani,Queen Ketevan Avenue,3 rooms"  
  
[[3]]$time  
[1] "Today 13:27"  
  
[[3]]$views  
[1] "1226"  
  
[[3]]$title  
[1] "Newly finished apartment for sale"  
  
[[3]]$id  
[1] ": 10491378"  
  
[[3]]$price\_gel  
[1] "257,000"  
  
[[3]]$price\_usd  
[1] "78,000"  
  
[[3]]$space  
[1] "Area: 72.00 m² "  
  
[[3]]$room  
[1] "3 Room"  
  
[[3]]$bedroom  
[1] "Bedroom"  
  
[[3]]$floor  
[1] "15/16"  
  
[[3]]$description  
[1] "Newly renovated apartment for sale near Metro 300 Aragveli with furniture and appliances."  
  
[[3]]$amenities  
[1] "Newly renovated,Nonstandard,Bedroom 1,Veranda 5 m²,Bathrooms 1,Heating,Central Heating System,Parking,Parking Place,Storeroom,Pantry,Hot water,Central Heating System,Gas,Furniture,Passenger elevator,Television,Air conditioner"  
  
[[3]]$latitude  
[1] "41.6895023"  
  
[[3]]$longitude  
[1] "44.8200504"  
  
[[3]]$poster\_type  
[1] "Owner"  
  
[[3]]$poster\_id  
[1] "3533391"

json\_structure <- capture.output(str(house))  
  
print(json\_structure[1:16])

[1] "List of 41663"   
 [2] " $ :List of 18"   
 [3] " ..$ address : chr \"შარტავას ქუჩა, Saburtalo, Saburtalo District, Tbilisi\""   
 [4] " ..$ product\_tree: chr \"Newly finished apartment for sale,Tbilisi,Saburtalo District,Saburtalo,1 rooms\""   
 [5] " ..$ time : chr \"Today 12:33\""   
 [6] " ..$ views : chr \"622\""   
 [7] " ..$ title : chr \"Newly finished apartment for sale\""   
 [8] " ..$ id : chr \": 10541934\""   
 [9] " ..$ price\_gel : chr \"107,100\""   
[10] " ..$ price\_usd : chr \"32,500\""   
[11] " ..$ space : chr \"Area: 28.00 m² \""   
[12] " ..$ room : chr \"1 Room\""   
[13] " ..$ bedroom : chr \"Bedroom\""   
[14] " ..$ floor : chr \"11/11\""   
[15] " ..$ description : chr \"1,5 Ot newly renovated apartment Non-residential, newly built, 28 sqm, 11 (11) (elevator up to 10) living room \"| \_\_truncated\_\_"  
[16] " ..$ amenities : chr \"Newly renovated,Nonstandard,Ceiling height,3.00 M,Bedroom 1,Balcony,Bathrooms 1,Heating,Central Heating System,\"| \_\_truncated\_\_"

# you can pull out individiual entries using double bracket notation  
# JSON is essentially lists inside a list  
  
house[[3]][[1]] # 3rd home and address row

[1] "Queen Ketevan Avenue, Isani, Isani District, Tbilisi"

house[[12]][[8]] # 12th home and price row

[1] "70,000"

house[[450]][[8]] # 450th home and price row

[1] "145,000"

From the example outputs above, “Queen Ketevan Avenue, Isani, Isani District, Tbilisi” is the address of the 3rd home. “70,000” represents the price of the 12th home. Similarly, “145,000” represents the price of the 450th home. So, that’s all for understanding the structure of a JSON dataset.

Next, let’s read in a new file type: **.XLSX file**

library(rio) # package that can import all sheets with one line of code

Warning: package 'rio' was built under R version 4.2.3

snap19 <- import\_list('FY19.xls')

New names:  
New names:  
New names:  
New names:  
New names:  
New names:  
New names:  
New names:  
• `` -> `...2`  
• `` -> `...3`  
• `` -> `...4`  
• `` -> `...5`

This dataset covers the US Supplemental Nutrition Assistance Program, more commonly known as SNAP. The program is the successor to the Food Stamps program previously in place. The program provides food assistance to low-income families in the form of a debit card. The US Dept of Agriculture, which maintains consumption data, does not release raw data on what foods are consumed - only summary reports. NERO, MARO, SERO, etc., are all different regions in the US. For example, Arizona belongs to the WRO (West Regional Office), and New York belongs to NERO (Northeast Regional Office).

## Day 2: Dealing with Missing Values

library(tidyverse)  
library(mice) # package for categorical & numeric imputation

Warning: package 'mice' was built under R version 4.2.3

Attaching package: 'mice'

The following object is masked from 'package:stats':  
  
 filter

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

library(dplyr)

Let’s read in the data:

train <- read.csv('train (1).csv')

I will be using the train.csv file from the titanic to deal with the missing values.

head(train,3)

PassengerId Survived Pclass  
1 1 0 3  
2 2 1 1  
3 3 1 3  
 Name Sex Age SibSp Parch  
1 Braund, Mr. Owen Harris male 22 1 0  
2 Cumings, Mrs. John Bradley (Florence Briggs Thayer) female 38 1 0  
3 Heikkinen, Miss. Laina female 26 0 0  
 Ticket Fare Cabin Embarked  
1 A/5 21171 7.2500 S  
2 PC 17599 71.2833 C85 C  
3 STON/O2. 3101282 7.9250 S

train <- tibble(train) # tibble() preserve all the variable types  
  
head(train)

# A tibble: 6 × 12  
 PassengerId Survived Pclass Name Sex Age SibSp Parch Ticket Fare Cabin  
 <int> <int> <int> <chr> <chr> <dbl> <int> <int> <chr> <dbl> <chr>  
1 1 0 3 Braund… male 22 1 0 A/5 2… 7.25 ""   
2 2 1 1 Cuming… fema… 38 1 0 PC 17… 71.3 "C85"  
3 3 1 3 Heikki… fema… 26 0 0 STON/… 7.92 ""   
4 4 1 1 Futrel… fema… 35 1 0 113803 53.1 "C12…  
5 5 0 3 Allen,… male 35 0 0 373450 8.05 ""   
6 6 0 3 Moran,… male NA 0 0 330877 8.46 ""   
# … with 1 more variable: Embarked <chr>

Tibble has a cleaner print format, making it easier to view and understand the data. Tibble also has a stricter syntax, which helps prevent common data manipulation errors

str(train)

tibble [891 × 12] (S3: tbl\_df/tbl/data.frame)  
 $ PassengerId: int [1:891] 1 2 3 4 5 6 7 8 9 10 ...  
 $ Survived : int [1:891] 0 1 1 1 0 0 0 0 1 1 ...  
 $ Pclass : int [1:891] 3 1 3 1 3 3 1 3 3 2 ...  
 $ Name : chr [1:891] "Braund, Mr. Owen Harris" "Cumings, Mrs. John Bradley (Florence Briggs Thayer)" "Heikkinen, Miss. Laina" "Futrelle, Mrs. Jacques Heath (Lily May Peel)" ...  
 $ Sex : chr [1:891] "male" "female" "female" "female" ...  
 $ Age : num [1:891] 22 38 26 35 35 NA 54 2 27 14 ...  
 $ SibSp : int [1:891] 1 1 0 1 0 0 0 3 0 1 ...  
 $ Parch : int [1:891] 0 0 0 0 0 0 0 1 2 0 ...  
 $ Ticket : chr [1:891] "A/5 21171" "PC 17599" "STON/O2. 3101282" "113803" ...  
 $ Fare : num [1:891] 7.25 71.28 7.92 53.1 8.05 ...  
 $ Cabin : chr [1:891] "" "C85" "" "C123" ...  
 $ Embarked : chr [1:891] "S" "C" "S" "S" ...

Let’s check for missing values from the dataset

any(is.na(train))

[1] TRUE

Since there are missing values in this dataset. Is the data missing at random (MAR) or is it Meaningfully Missing (MM)?

# make a missing map!  
library(Amelia)

Warning: package 'Amelia' was built under R version 4.2.3

Loading required package: Rcpp

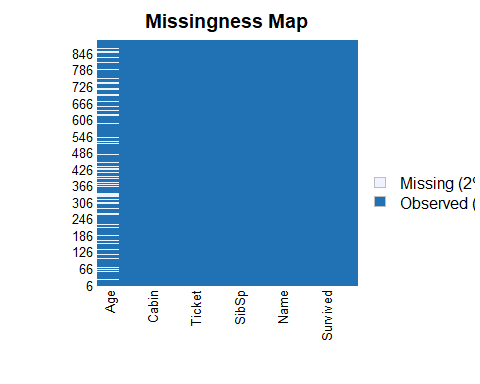
Warning: package 'Rcpp' was built under R version 4.2.2

##   
## Amelia II: Multiple Imputation  
## (Version 1.8.1, built: 2022-11-18)  
## Copyright (C) 2005-2023 James Honaker, Gary King and Matthew Blackwell  
## Refer to http://gking.harvard.edu/amelia/ for more information  
##

missmap(train)

Warning: Unknown or uninitialised column: `arguments`.  
Unknown or uninitialised column: `arguments`.

Warning: Unknown or uninitialised column: `imputations`.



It looks like the 2% of the missing data comes from ‘Age’. The age of some passengers was simply not recorded or was lost over time. We could remove this column. However, if the column with missing values is important for the analysis and for the main question we’re interested in, then removing the column may not be the right move. But let’s practice removing it for this example.

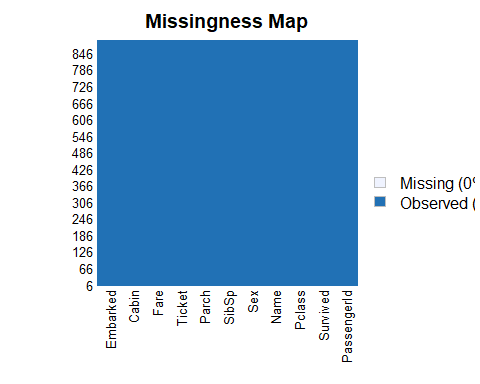
train\_removed = subset(train, select = -c(Age))  
  
head(train\_removed)

# A tibble: 6 × 11  
 PassengerId Survived Pclass Name Sex SibSp Parch Ticket Fare Cabin Embar…¹  
 <int> <int> <int> <chr> <chr> <int> <int> <chr> <dbl> <chr> <chr>   
1 1 0 3 Brau… male 1 0 A/5 2… 7.25 "" S   
2 2 1 1 Cumi… fema… 1 0 PC 17… 71.3 "C85" C   
3 3 1 3 Heik… fema… 0 0 STON/… 7.92 "" S   
4 4 1 1 Futr… fema… 1 0 113803 53.1 "C12… S   
5 5 0 3 Alle… male 0 0 373450 8.05 "" S   
6 6 0 3 Mora… male 0 0 330877 8.46 "" Q   
# … with abbreviated variable name ¹​Embarked

missmap(train\_removed)

Warning: Unknown or uninitialised column: `arguments`.  
Unknown or uninitialised column: `arguments`.

Warning: Unknown or uninitialised column: `imputations`.



You can see that there are no more missing values. That’s one way to deal with missing values. Again, if ‘Age’ was important in our analysis then this method would not be preferable. The next method that may be helpful would be imputation.

I’ll use the “Mice Algorithm” (Multiple Imputation by Chained Equations), which is a statistical technique used to impute missing data in a dataset. It is based on the idea that missing values can be imputed by predicting them from other variables in the dataset.

imp <- mice(train, m = 5, method = "pmm") # pmm means predictive mean matching

iter imp variable  
 1 1 Age  
 1 2 Age  
 1 3 Age  
 1 4 Age  
 1 5 Age  
 2 1 Age  
 2 2 Age  
 2 3 Age  
 2 4 Age  
 2 5 Age  
 3 1 Age  
 3 2 Age  
 3 3 Age  
 3 4 Age  
 3 5 Age  
 4 1 Age  
 4 2 Age  
 4 3 Age  
 4 4 Age  
 4 5 Age  
 5 1 Age  
 5 2 Age  
 5 3 Age  
 5 4 Age  
 5 5 Age

Warning: Number of logged events: 5

summary(imp)

Class: mids  
Number of multiple imputations: 5   
Imputation methods:  
PassengerId Survived Pclass Name Sex Age   
 "" "" "" "" "" "pmm"   
 SibSp Parch Ticket Fare Cabin Embarked   
 "" "" "" "" "" ""   
PredictorMatrix:  
 PassengerId Survived Pclass Name Sex Age SibSp Parch Ticket Fare  
PassengerId 0 1 1 0 0 1 1 1 0 1  
Survived 1 0 1 0 0 1 1 1 0 1  
Pclass 1 1 0 0 0 1 1 1 0 1  
Name 1 1 1 0 0 1 1 1 0 1  
Sex 1 1 1 0 0 1 1 1 0 1  
Age 1 1 1 0 0 0 1 1 0 1  
 Cabin Embarked  
PassengerId 0 0  
Survived 0 0  
Pclass 0 0  
Name 0 0  
Sex 0 0  
Age 0 0  
Number of logged events: 5   
 it im dep meth out  
1 0 0 constant Name  
2 0 0 constant Sex  
3 0 0 constant Ticket  
4 0 0 constant Cabin  
5 0 0 constant Embarked

completedData <- complete(imp,1)

The missing values have been replaced with the imputed values in the first of the five datasets. This is not 100% completed. I think there might be other ways to handle missing values, but this is just for practice.

## 

## Day 3: Identifying and Handling Outliers

library(outliers)  
library(ggplot2)

Let’s read in the data:

tx\_salary <- read.csv('texas\_salaries.csv')

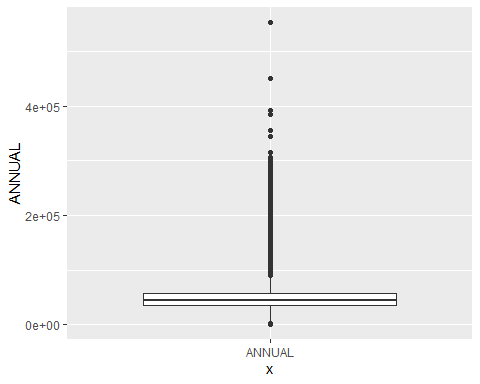
Database of compensation for Texas state employees, as published by [The Texas Tribune](https://salaries.texastribune.org/)

head(tx\_salary,5)

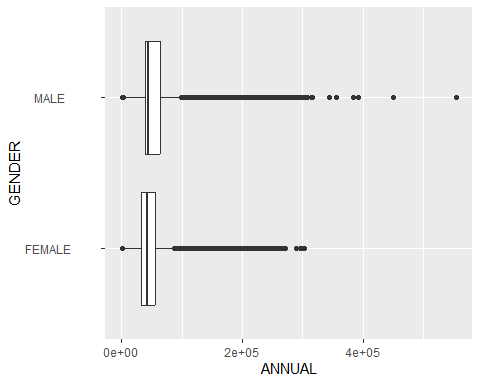
AGY AGENCY.NAME  
1 241 COMPTROLLER OF PUBLIC ACCOUNTS, JUDICIARY SECTION   
2 212 OFFICE OF COURT ADMINISTRATION   
3 510 TEXAS BEHAVIORAL HEALTH EXECUTIVE COUNCIL   
4 520 BOARD OF EXAMINERS OF PSYCHOLOGISTS   
5 537 DEPARTMENT OF STATE HEALTH SERVICES   
 LAST.NAME FIRST.NAME MI CLASS.CODE  
1 RUCKER MORTON V JD25   
2 RUCKER MORTON V 3524   
3 SPINKS DARREL D 1623   
4 SPINKS DARREL D E178   
5 ADAMS III LEE A 1323   
 CLASS.TITLE ETHNICITY  
1 JUDGE, RETIRED WHITE   
2 GENERAL COUNSEL IV WHITE   
3 DIRECTOR IV WHITE   
4 EXEC DIR, BD OF EXAMS OF PSYCHOLOGISTS WHITE   
5 INSPECTOR III BLACK   
 GENDER STATUS EMPLOY.DATE  
1 MALE URP - UNCLASSIFIED REGULAR PART-TIME 02/18/88  
2 MALE CTP - CLASSIFIED TEMPORARY PART-TIME 02/01/15  
3 MALE CRF - CLASSIFIED REGULAR FULL-TIME 03/01/20  
4 MALE ERP - EXEMPT REGULAR PART-TIME 03/04/20  
5 MALE CRF - CLASSIFIED REGULAR FULL-TIME 09/01/19  
 HRLY.RATE HRS.PER.WK MONTHLY ANNUAL STATE.NUMBER duplicated  
1 75.96150 29 9545.82 114549.84 127717 True  
2 81.04454 4 1404.77 16857.24 127717 True  
3 0.00000 40 10000.00 120000.00 147334 True  
4 49.40717 20 4281.95 51383.40 147334 True  
5 0.00000 40 3447.25 41367.00 129635 True  
 multiple\_full\_time\_jobs combined\_multiple\_jobs summed\_annual\_salary  
1 NA 131407.1  
2 NA NA  
3 NA 171383.4  
4 NA NA  
5 1 NA  
 hide\_from\_search  
1   
2 True  
3   
4 True  
5

Next, let’s plot it to get an idea of the outliers:

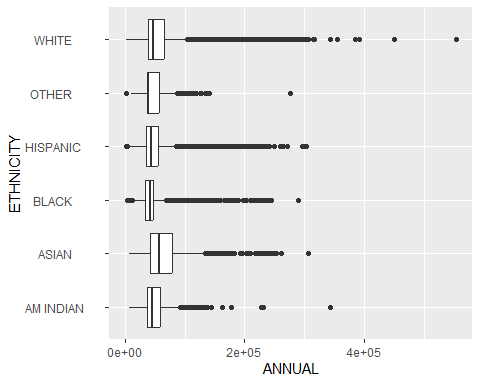
ggplot(tx\_salary,aes(x="ANNUAL", y=ANNUAL)) + geom\_boxplot() # plotting annual salary



# salary outliers within gender  
ggplot(tx\_salary,aes(GENDER,ANNUAL)) + geom\_boxplot() + coord\_flip()

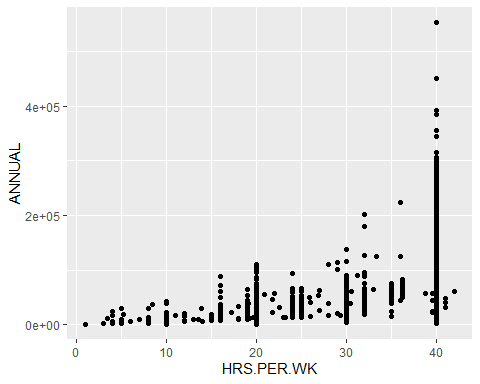


# salary outliers by ethnicity  
ggplot(tx\_salary,aes(ETHNICITY,ANNUAL)) + geom\_boxplot() + coord\_flip()

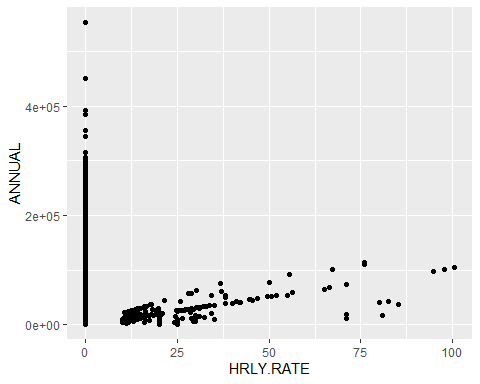


Interestingly, while mean salaries are rather similar across groups, and all groups have outliers, the most extreme outliers on the high end are white and male. The white group does not have outliers on the low end, whereas black, Hispanic, and others all do.

# hrs per week and annual salary should be highly correlated and predictable.  
# we do want to know if there are observations with low hours but high salaries  
# or with high hour but low salaries.  
# Are there people with comparable salaries but very different hours worked?  
ggplot(tx\_salary,aes(HRS.PER.WK, ANNUAL)) + geom\_point()



# as expected, a majority are 40 hrs per week  
# but there are many observations of cases where people are working 40 hrs  
# and making similar annual salaries to others working 20-35 hours.  
# let's look at hrly rate  
  
ggplot(tx\_salary,aes(HRLY.RATE, ANNUAL)) + geom\_point()



Looks like a lot of missing data there - it would not make sense for people with hourly rates of 0 to also have an annual salary, so the hourly rate is likely not a good variable to use in this dataset.

Let’s identify which rows contain outliers. We can see them in the visualizations... but where are they in the data?

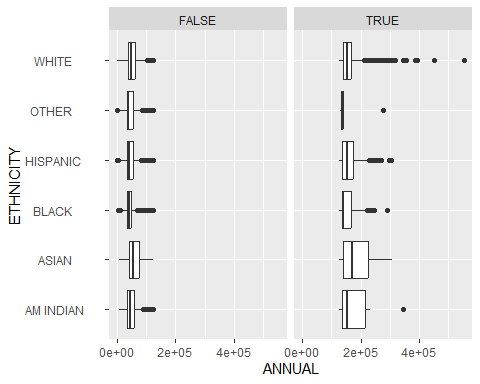
# calculate the z-scores  
outlier\_scores <- scores(tx\_salary$ANNUAL)

# create a vector that holds TRUE if outlier\_score is greater than   
# 3 or less than -3  
is\_outlier <- outlier\_scores > 3 | outlier\_scores < -3

# add a column to data to indicate which are outliers on the var selected  
tx\_salary$annual\_outlier <- is\_outlier

Now outlier and non-outlier values can be graphed or analyzed separately

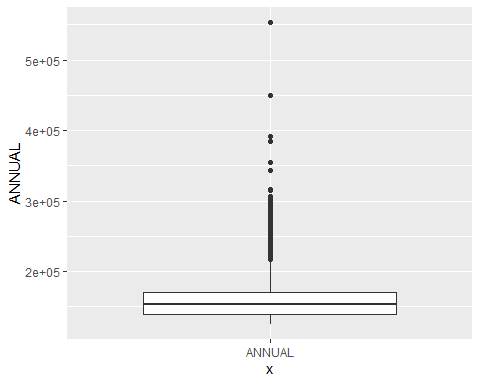
ggplot(tx\_salary,aes(ETHNICITY,ANNUAL)) + geom\_boxplot() + coord\_flip() +  
facet\_wrap(~annual\_outlier)



# create a dataframe with only the outliers  
tx\_outliers <- tx\_salary[outlier\_scores > 3 | outlier\_scores < -3, ]  
  
# call it  
head(tx\_outliers)

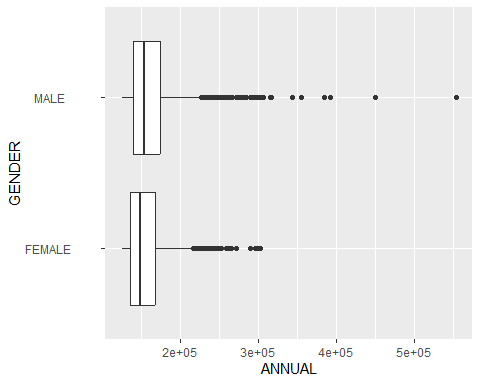
AGY AGENCY.NAME  
46 101 SENATE   
55 101 SENATE   
58 101 SENATE   
61 101 SENATE   
64 101 SENATE   
108 303 TEXAS FACILITIES COMMISSION   
 LAST.NAME FIRST.NAME MI CLASS.CODE  
46 LUPTON ANGUS C 7101   
55 SANCHEZ LUIS M 7101   
58 STEINBACH CHRISTOPHER J 7101   
61 TERRY STEPHEN C 7101   
64 WENDLER LARA 7101   
108 MARTINEZ MARTY M 1623   
 CLASS.TITLE ETHNICITY  
46 LEG. OFFICIAL/ADMINISTRATOR WHITE   
55 LEG. OFFICIAL/ADMINISTRATOR HISPANIC   
58 LEG. OFFICIAL/ADMINISTRATOR WHITE   
61 LEG. OFFICIAL/ADMINISTRATOR WHITE   
64 LEG. OFFICIAL/ADMINISTRATOR WHITE   
108 DIRECTOR IV HISPANIC   
 GENDER STATUS EMPLOY.DATE  
46 MALE URF - UNCLASSIFIED REGULAR FULL-TIME 04/13/06  
55 MALE URF - UNCLASSIFIED REGULAR FULL-TIME 10/12/07  
58 MALE URF - UNCLASSIFIED REGULAR FULL-TIME 12/23/14  
61 MALE URF - UNCLASSIFIED REGULAR FULL-TIME 01/10/17  
64 FEMALE URF - UNCLASSIFIED REGULAR FULL-TIME 09/01/05  
108 MALE CRF - CLASSIFIED REGULAR FULL-TIME 08/01/19  
 HRLY.RATE HRS.PER.WK MONTHLY ANNUAL STATE.NUMBER duplicated  
46 0 40 10850.00 130200.0 148005 True  
55 0 40 10500.00 126000.0 11355 True  
58 0 40 13000.00 156000.0 189382 True  
61 0 40 10500.00 126000.0 159364 True  
64 0 40 12300.00 147600.0 89658 True  
108 0 40 10833.34 130000.1 1199398 True  
 multiple\_full\_time\_jobs combined\_multiple\_jobs summed\_annual\_salary  
46 NA True NA  
55 NA True NA  
58 NA True NA  
61 NA True NA  
64 NA True NA  
108 NA True NA  
 hide\_from\_search annual\_outlier  
46 TRUE  
55 TRUE  
58 TRUE  
61 TRUE  
64 TRUE  
108 TRUE

ggplot(tx\_outliers,aes(x="ANNUAL", y=ANNUAL)) + geom\_boxplot()

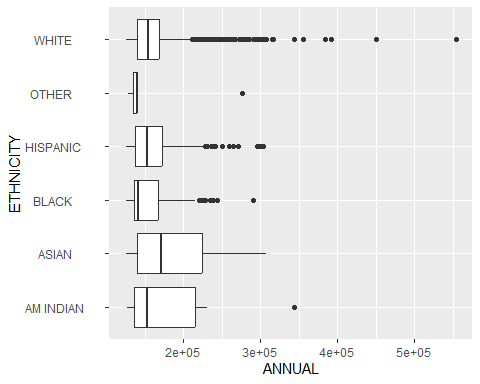


Outliers from the outliers.

ggplot(tx\_outliers,aes(GENDER,ANNUAL)) + geom\_boxplot() + coord\_flip()



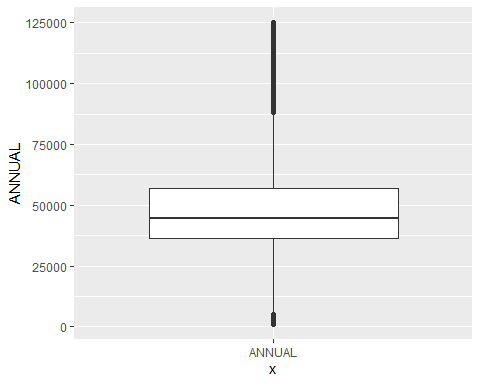
ggplot(tx\_outliers,aes(ETHNICITY,ANNUAL)) + geom\_boxplot() + coord\_flip()



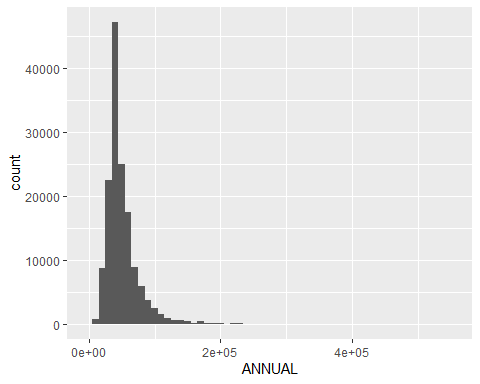
There are both reasons for keeping and removing outliers. Sometimes, they are unnecessary noise and it is okay to get rid of them since we are trying to predict things for the masses and not for the extremes. In other cases, they are an important part of the story that the data tells. With salary data reporting, it would often be the case that removing outliers would be unrealistic and inaccurate, but we’ll still practice the strats provided on this data set since we know there are outliers!

tx\_norms <- tx\_salary[tx\_salary$annual\_outlier == FALSE, ]

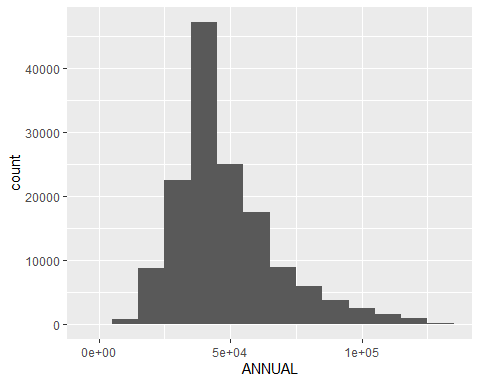
ggplot(tx\_norms,aes(x="ANNUAL", y=ANNUAL)) + geom\_boxplot()



# comparing distribution of initial data to data with outliers removed  
ggplot(tx\_salary, aes(ANNUAL)) + geom\_histogram(binwidth=10000)



ggplot(tx\_norms, aes(ANNUAL)) + geom\_histogram(binwidth=10000)



Another option for dealing with outliers is removing and replacing via imputation

# replace the outliers with NA  
tx\_salary[outlier\_scores > 3 | outlier\_scores < -3, "ANNUAL"] <- NA

# check  
summary(tx\_salary$ANNUAL)

Min. 1st Qu. Median Mean 3rd Qu. Max. NA's   
 1040 36238 44642 48400 56963 125178 2680

# replace with mean (median can be more accurate in salaries,  
# but they are not far off without the outliers!)  
  
# let's use a function and then apply it to the column  
impute <- function(x){  
 if(is.na(x)){  
 return(48400) # mean value above  
 }  
 else {  
 return(x)  
 }  
}

tx\_salary$ANNUAL <- sapply(tx\_salary$ANNUAL, FUN = impute)

# check  
summary(tx\_salary$ANNUAL)

Min. 1st Qu. Median Mean 3rd Qu. Max.   
 1040 36238 44642 48400 56544 125178

To reflect, there are many ways to deal with outliers. The common ways are removing the rows from the data, considering outliers and inliners separately, and removing and replacing them via imputation.

## 

## Day 4: Removing Duplicate Records

Duplicate records can be common when webscraping (the same table may exist on multiple pages, for example). Removing duplicate records is important because it can lead to incorrect conclusions about observations being of higher frequency than they actually are.

For some data, another instance of the same information is important and part of the structure of the set, The stuff that gets duplicated should be unintentional repetitions that were COLLECTED or GATHERED too frequently for how often they actually occurred.

library(tidyverse)  
library(mice)

Let’s read in the data:

steam\_data <- read.csv("steam-200k.csv", header = TRUE, stringsAsFactors = FALSE)  
  
# add col names  
names(steam\_data) <- c('user\_id','game\_title', "behavior\_name","value","x")  
  
#remove last col and create df  
steam\_data <- tibble(steam\_data[, 1:4])

Steam is the world’s most popular PC Gaming hub. With a massive collection that includes everything from AAA blockbusters to small indie titles, great discovery tools can be super valuable for Steam.

This dataset is a list of user behaviors, with columns: user-id, game-title, behavior-name, and value. The behaviors included are ‘purchase’ and ‘play’. The value indicates the degree to which the behavior was performed - in the case of ‘purchase’ the value is always 1, and in the case of ‘play’ the value represents the number of hours the user has played the game.

Next, we can use the **distinct()** function from **dplyr** to identify and remove duplicate rows based on selected columns. For example, if we want to remove duplicates based on the **user\_id** and **game\_title** columns, we can do:

steam\_data\_unique <- distinct(steam\_data, user\_id, game\_title, .keep\_all = TRUE)

Finally, we can write the cleaned data frame to a new CSV file:

write.csv(steam\_data\_unique, "steam-200k-cleaned.csv", row.names = FALSE)

That was just a basic way to remove duplicate values. A good thing to note is that our original steam\_data had 199999 observations, and after removing the duplicate values we’re left with 128804. It’s important to consider why there were duplicates in the first place and whether removing them could result in biased or incomplete data.

## 

## Day 5: Cleaning Numeric Columns

Numeric data can come with all sorts of non-numeric characters such as percentage signs, and number signs of other special characters that may have meaning in another programming language but act only as noise in yours.

R plays it safe - so any column that comes in with special characters is usually labeled and treated as a character column rather than a numeric column.

Regular expressions can be used but should be used as a last resort. They are rigid, easily broken, hard to read, and hard to debug. Instead, parse\_number() from tidyverse can be a great option.

library(tidyverse)  
library(lubridate)

Warning: package 'lubridate' was built under R version 4.2.2

Loading required package: timechange

Warning: package 'timechange' was built under R version 4.2.2

Attaching package: 'lubridate'

The following objects are masked from 'package:base':  
  
 date, intersect, setdiff, union

# character vector of numbers  
to\_parse <- c(100, "10,000", "%100", "$50")  
  
# check to make sure it's numeric  
print("Class before:")

[1] "Class before:"

class(to\_parse)

[1] "character"

# parse numbers  
parsed\_numbers <- parse\_number(to\_parse)  
  
# check class  
print("Class after:")

[1] "Class after:"

class(parsed\_numbers)

[1] "numeric"

# see what it looks like now  
parsed\_numbers

[1] 100 10000 100 50

## 

## End of code.