

# Data 607 - Project 2 - Data Transformation

William Jasmine

2022-10-7

## Introduction

This project includes work done to tidy three “messy” data sets that originated as CSV files. The sections below outline the steps required to clean these data sets, and then analyzes each of the resulting “clean” data frames to answer a specific research question. The three .csv files used, as well as a description of the data they include can be found below:

1. **unclean\_gdp\_data.csv**: Includes the GDP of most world countries from 1961-2021. Source (Provided in class discussion by Benjamin Ingbar)
2. **unclean\_student\_data.csv**: Includes the test results for a class of students over the course of three different school terms. Source (Provided in class discussion by Jhalak Das)
3. **unclean\_atmosphere\_data.csv**: Includes a number of measurements (i.e. pressure) regarding the air in our atmosphere at various heights above sea level. Source (Provided in class discussion by Neil Hodgkinson)

The data is also stored in the following Github location.

## Import Data

The following cells import each of the .csv files listed above from the specified github location.

```
csv_data = getURL("https://raw.githubusercontent.com/williamzjasmine/CUNY_SPS_DS/master/DATA_607/Project2/unclean_gdp_data.csv")
gdp_df = read.csv(text = csv_data)
head(gdp_df)
```

```
##           i..Country.Name Country.Code Indicator.Name Indicator.Code
## 1                Aruba          ABW GDP (current US$) NY.GDP.MKTP.CD
## 2 Africa Eastern and Southern          AFE GDP (current US$) NY.GDP.MKTP.CD
## 3                Afghanistan          AFG GDP (current US$) NY.GDP.MKTP.CD
## 4 Africa Western and Central          AFW GDP (current US$) NY.GDP.MKTP.CD
## 5                Angola          AGO GDP (current US$) NY.GDP.MKTP.CD
## 6                Albania          ALB GDP (current US$) NY.GDP.MKTP.CD
##           X1960      X1961      X1962      X1963      X1964      X1965
## 1           NA           NA           NA           NA           NA           NA
## 2 21290586003 21808473825 23707015394 28210036878 26118787467 29682172751
## 3   537777811  548888896  546666678   751111191   800000044  1006666638
## 4 10404135069 11127894641 11943187848 12676330765 13838369295 14862225760
## 5           NA           NA           NA           NA           NA           NA
## 6           NA           NA           NA           NA           NA           NA
##           X1966      X1967      X1968      X1969      X1970      X1971
## 1           NA           NA           NA           NA           NA           NA
```

|      |              |              |              |              |              |              |
|------|--------------|--------------|--------------|--------------|--------------|--------------|
| ## 2 | 32239121547  | 33514552047  | 36521482937  | 41828336213  | 44862605393  | 49478916698  |
| ## 3 | 1399999967   | 1673333418   | 1373333367   | 1408888922   | 1748886596   | 1831108971   |
| ## 4 | 15832591204  | 14426038230  | 14880349280  | 16882092549  | 23504605476  | 20832817218  |
| ## 5 | NA           | NA           | NA           | NA           | NA           | NA           |
| ## 6 | NA           | NA           | NA           | NA           | NA           | NA           |
| ##   | X1972        | X1973        | X1974        | X1975        | X1976        | X1977        |
| ## 1 | NA           | NA           | NA           | NA           | NA           | NA           |
| ## 2 | 53514844534  | 69600788111  | 86057777551  | 91649152687  | 91124551926  | 103416000000 |
| ## 3 | 1595555476   | 1733333264   | 2155555498   | 2366666616   | 2555555567   | 2953333418   |
| ## 4 | 25264953766  | 31273819026  | 44214484997  | 51444731784  | 62129390375  | 65315008068  |
| ## 5 | NA           | NA           | NA           | NA           | NA           | NA           |
| ## 6 | NA           | NA           | NA           | NA           | NA           | NA           |
| ##   | X1978        | X1979        | X1980        | X1981        | X1982        | X1983        |
| ## 1 | NA           | NA           | NA           | NA           | NA           | NA           |
| ## 2 | 115345000000 | 1.34671e+11  | 170654000000 | 174387000000 | 167266000000 | 174918000000 |
| ## 3 | 3300000109   | 3.69794e+09  | 3641723322   | 3478787909   | NA           | NA           |
| ## 4 | 71199708192  | 8.86284e+10  | 112031000000 | 211003000000 | 187164000000 | 138115000000 |
| ## 5 | NA           | NA           | 5930503401   | 5550483036   | 5550483036   | 5784341596   |
| ## 6 | NA           | NA           | NA           | NA           | NA           | NA           |
| ##   | X1984        | X1985        | X1986        | X1987        | X1988        | X1989        |
| ## 1 | NA           | NA           | 405586592    | 487709497    | 596648045    | 695530726    |
| ## 2 | 160134000000 | 1.36297e+11  | 152518000000 | 186145000000 | 204140000000 | 217539000000 |
| ## 3 | NA           | NA           | NA           | NA           | NA           | NA           |
| ## 4 | 114263000000 | 1.16507e+11  | 107498000000 | 110322000000 | 108943000000 | 101769000000 |
| ## 5 | 6131475065   | 7.55356e+09  | 7072063346   | 8083872012   | 8769250550   | 10201099039  |
| ## 6 | 1857338012   | 1.89705e+09  | 2097326250   | 2080796250   | 2051236250   | 2253090000   |
| ##   | X1990        | X1991        | X1992        | X1993        | X1994        | X1995        |
| ## 1 | 764804469    | 872067039    | 958659218    | 1083240223   | 1245810056   | 1320670391   |
| ## 2 | 253224000000 | 273403000000 | 238255000000 | 236527000000 | 240120000000 | 269637000000 |
| ## 3 | NA           | NA           | NA           | NA           | NA           | NA           |
| ## 4 | 121802000000 | 117457000000 | 118282000000 | 98826369836  | 86281743753  | 108221000000 |
| ## 5 | 11228764963  | 10603784541  | 8307810974   | 5768720422   | 4438321017   | 5538749260   |
| ## 6 | 2028553750   | 1099559028   | 652174991    | 1185315468   | 1880951520   | 2392764853   |
| ##   | X1996        | X1997        | X1998        | X1999        | X2000        | X2001        |
| ## 1 | 1379888268   | 1531843575   | 1665363128   | 1722905028   | 1873184358   | 1896648045   |
| ## 2 | 268414000000 | 282185000000 | 265814000000 | 262172000000 | 283925000000 | 258819000000 |
| ## 3 | NA           | NA           | NA           | NA           | NA           | NA           |
| ## 4 | 125763000000 | 127064000000 | 130107000000 | 137521000000 | 140410000000 | 148013000000 |
| ## 5 | 7526446606   | 7648377413   | 6506229607   | 6152922943   | 9129594819   | 8936063723   |
| ## 6 | 3199641336   | 2258513974   | 2545964541   | 3212121651   | 3480355258   | 3922100794   |
| ##   | X2002        | X2003        | X2004        | X2005        | X2006        | X2007        |
| ## 1 | 1962011173   | 2044134078   | 2254748603   | 2359776536   | 2469832402   | 2677653631   |
| ## 2 | 264870000000 | 352659000000 | 438834000000 | 512211000000 | 575921000000 | 661179000000 |
| ## 3 | 4055179566   | 4515558808   | 5226778809   | 6209137625   | 6971285595   | 9747879532   |
| ## 4 | 176938000000 | 204645000000 | 254093000000 | 310558000000 | 393305000000 | 461791000000 |
| ## 5 | 15285594828  | 17812704825  | 23552047248  | 36970918699  | 52381006892  | 65266452081  |
| ## 6 | 4348068242   | 5611496257   | 7184685782   | 8052073539   | 8896072919   | 10677324144  |
| ##   | X2008        | X2009        | X2010        | X2011        | X2012        | X2013        |
| ## 1 | 2843016760   | 2553631285   | 2453631285   | 2637988827   | 2615083799   | 2727932961   |
| ## 2 | 708287000000 | 719217000000 | 860478000000 | 964418000000 | 973043000000 | 983937000000 |
| ## 3 | 10109305183  | 12416161049  | 15856678596  | 17805113119  | 19907317066  | 20146404996  |
| ## 4 | 566481000000 | 507044000000 | 591596000000 | 670983000000 | 727570000000 | 820793000000 |
| ## 5 | 88538610805  | 70307166934  | 81699556137  | 109437000000 | 124998000000 | 133402000000 |
| ## 6 | 12881353508  | 12044208086  | 11926922829  | 12890764531  | 12319830437  | 12776220507  |

```
##           X2014           X2015           X2016           X2017           X2018           X2019
## 1 2.791061e+09   2963128492   2983798883 3.092179e+09   3202234637   3310055866
## 2 1.003680e+12  924253000000  882355000000 1.020650e+12  991022000000  997534000000
## 3 2.049713e+10  19134211764   18116562465 1.875347e+10  18053228579  18799450743
## 4 8.649900e+11  760734000000  690546000000 6.837490e+11  741690000000  794543000000
## 5 1.372440e+11  87219290029   49840494026 6.897276e+10  77792940077  69309104807
## 6 1.322815e+10  11386850130   11861199831 1.301969e+10  15156432310  15401830754
##           X2020           X2021
## 1   2496648045           NA
## 2 921646000000 1.082100e+12
## 3   20116137326           NA
## 4 784446000000 8.358080e+11
## 5   53619071176 7.254699e+10
## 6   15131866271 1.826004e+10
```

```
csv_data = getURL("https://raw.githubusercontent.com/williamzjasmine/CUNY_SPS_DS/master/DATA_607/Projec
```

```
student_df = read.csv(text = csv_data)
head(student_df)
```

```
##   id  name phone sex.and.age test.number term.1 term.2 term.3
## 1  1  Mike   134      m_12    test 1      76      84      87
## 2  2  Linda  270      f_13    test 1      88      90      73
## 3  3   Sam   210      m_11    test 1      78      74      80
## 4  4 Esther  617      f_12    test 1      68      75      74
## 5  5  Mary   114      f_14    test 1      65      67      64
## 6  6   Dan   114      f_14    test 1      81      67      90
```

```
csv_data = getURL("https://raw.githubusercontent.com/williamzjasmine/CUNY_SPS_DS/master/DATA_607/Projec
```

```
atmos_df = read.csv(text = csv_data)
head(atmos_df)
```

```
##   X  Altitude X.Air.press   X.ppO2   X.Alv.pO2   X...sat.O2 X.Alv.pCO2
## 1 1      Ft/m      mmHg  Air=21%  mmHg on air  >90% desired  >35 best
## 2 2 Sea level      760    159      104      97      40
## 3 3   10k/3k      523    110      67      90      36
## 4 4   20k/6.1k     349    73      40      73      24
## 5 5   30k/9.1k     226    47      18      24      24
## 6 6   40k/12k     141    29      NaN      NaN      NaN
##   X.Alv.pO2.with.O2.100. X...sat.O2_1 X.Alv.pCO2_1
## 1      NA      100% O2      100% O2
## 2      673      100      40
## 3      436      100      40
## 4      262      100      40
## 5      139      99      40
## 6       58      84      36
```

Looking above, it is clear that each of the .csv files were successfully imported, and the head of each is printed above. The three files have now been turned into three R dataframes: `gdp_df`, `student_df`, and `atmos_df`. The following sections will separately clean and analyze each of these dataframes.

# GDP Dataset

## Cleaning the Data

The first step is to clean the column names so that its easier to access the required data for each subsequent cleaning step. The column names have a number of issues, which are listed below along with how they can be fixed:

1. The year columns all start with a X. This can be fixed by using a `str_replace_all`.
2. The categorical variable columns all have . characters (which replaced the spaces in the original .csv file). This can be fixed by using a `str_replace_all`.
3. The first column name `i..Country.Name` has a special character and can be fixed by just renaming the column completely to `country_name`.
4. The column names include capital letters, which can be fixed using the `tolower()` function.

Each of these fixes is performed in the code chunk below, and the commented numbers correspond to the where each of the above steps is performed.

```
new_col_names <-  
  colnames(gdp_df) %>% #1  
    str_replace_all("X", '') %>% #2  
    str_replace_all('\\.', '_') %>% #3  
    tolower() #4  
new_col_names[1] <- 'country_name'  
  
colnames(gdp_df) <- new_col_names  
head(gdp_df, 1)
```

```
##   country_name country_code   indicator_name indicator_code 1960 1961 1962  
## 1      Aruba      ABW GDP (current US$) NY.GDP.MKTP.CD   NA   NA   NA  
##   1963 1964 1965 1966 1967 1968 1969 1970 1971 1972 1973 1974 1975 1976 1977  
## 1   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA   NA  
##   1978 1979 1980 1981 1982 1983 1984 1985      1986      1987      1988  
## 1   NA   NA   NA   NA   NA   NA   NA   NA 405586592 487709497 596648045  
##      1989      1990      1991      1992      1993      1994      1995  
## 1 695530726 764804469 872067039 958659218 1083240223 1245810056 1320670391  
##      1996      1997      1998      1999      2000      2001      2002  
## 1 1379888268 1531843575 1665363128 1722905028 1873184358 1896648045 1962011173  
##      2003      2004      2005      2006      2007      2008      2009  
## 1 2044134078 2254748603 2359776536 2469832402 2677653631 2843016760 2553631285  
##      2010      2011      2012      2013      2014      2015      2016  
## 1 2453631285 2637988827 2615083799 2727932961 2791061453 2963128492 2983798883  
##      2017      2018      2019      2020 2021  
## 1 3092178771 3202234637 3310055866 2496648045   NA
```

As is clear in the output above, the column names are all fixed.

Next step is to convert the numerous year columns into a single field (as opposed to each year having its own column). Completing this step means that the data will be in a more desirable long format, and can be easily accomplished using the `pivot_longer` function. This is done in the code chunk below:

```
cols_to_keep <- c('country_name', 'country_code', 'indicator_name', 'indicator_code')  
gdp_df <- pivot_longer(gdp_df,  
  cols = !all_of(cols_to_keep),  
  names_to = 'year',  
  values_to = 'gdp')  
  
head(gdp_df)
```

```
## # A tibble: 6 x 6
##   country_name country_code indicator_name indicator_code year   gdp
##   <chr>         <chr>         <chr>         <chr>         <chr> <dbl>
## 1 Aruba        ABW           GDP (current US$) NY.GDP.MKTP.CD 1960    NA
## 2 Aruba        ABW           GDP (current US$) NY.GDP.MKTP.CD 1961    NA
## 3 Aruba        ABW           GDP (current US$) NY.GDP.MKTP.CD 1962    NA
## 4 Aruba        ABW           GDP (current US$) NY.GDP.MKTP.CD 1963    NA
## 5 Aruba        ABW           GDP (current US$) NY.GDP.MKTP.CD 1964    NA
## 6 Aruba        ABW           GDP (current US$) NY.GDP.MKTP.CD 1965    NA
```

The `pivot_longer` specifies in the `names_to` column that the year columns are to be melted into a single field called `year`. The values in the cells are then also melted into a single field called `gdp`.

While the data now has the desired structure, it is clear from the output that there are a number of NA entries, representing years in which the GDP was not recorded for a given country. While it might make sense to replace these NA values with a string like "not recorded", we are unable to do so since that would mean two different datatypes would be stored in the same column, which is not possible.

There is however one last data cleaning step that is possible: the cells below print the unique values present in both the `indicator_name` and `indicator_code` columns:

```
print(unique(gdp_df$indicator_name))
```

```
## [1] "GDP (current US$)"
```

```
print(unique(gdp_df$indicator_code))
```

```
## [1] "NY.GDP.MKTP.CD"
```

As it is clear in the output above, each of the columns only one value, telling us that all of the measurements are GDP values measured in US dollars. Since all the measurements in the `gdp` column are of the same type and unit, these columns provide no additional information. As such, they are removed in the cell below.

```
gdp_df <-
  gdp_df %>% select(country_name, country_code, year, gdp)
head(gdp_df)
```

```
## # A tibble: 6 x 4
##   country_name country_code year   gdp
##   <chr>         <chr>         <chr> <dbl>
## 1 Aruba        ABW           1960    NA
## 2 Aruba        ABW           1961    NA
## 3 Aruba        ABW           1962    NA
## 4 Aruba        ABW           1963    NA
## 5 Aruba        ABW           1964    NA
## 6 Aruba        ABW           1965    NA
```

The data frame above is now in its final form, and is ready to be analyzed.

## Analysis

One possible research question we can now answer with the cleaned data is to determine the global trend in GDP year over year (both as total dollar amount and percent change).

The first step in doing so is to aggregate the global GDP by year for all countries. This is done below using a `groupby` and `summarise` function:

```
global_gdp_df <-
  gdp_df %>%
    filter(!is.na(gdp)) %>%
    group_by(year) %>%
    summarise(global_gdp = sum(gdp))
head(global_gdp_df)
```

```
## # A tibble: 6 x 2
##   year global_gdp
##   <chr>      <dbl>
## 1 1960    9.50e12
## 2 1961    9.76e12
## 3 1962    1.04e13
## 4 1963    1.12e13
## 5 1964    1.23e13
## 6 1965    1.35e13
```

The output above shows the newly created `global_gdp_df` dataframe, which contains the global GDP each year in USD. The cell below adds to that dataframe the percent change in global GDP using the `lag` window function:

```
global_gdp_df <-
  global_gdp_df %>%
    mutate(percent_change = (global_gdp - lag(global_gdp)) / lag(global_gdp))
head(global_gdp_df)
```

```
## # A tibble: 6 x 3
##   year global_gdp percent_change
##   <chr>      <dbl>      <dbl>
## 1 1960    9.50e12         NA
## 2 1961    9.76e12     0.0269
## 3 1962    1.04e13     0.0660
## 4 1963    1.12e13     0.0785
## 5 1964    1.23e13     0.0983
## 6 1965    1.35e13     0.0969
```

The code cell below checks to make sure that the calculation done above is correct but finding the percent change in global GDP from 1960 to 1961 and comparing it to the analogous `percent_change` value in the dataframe:

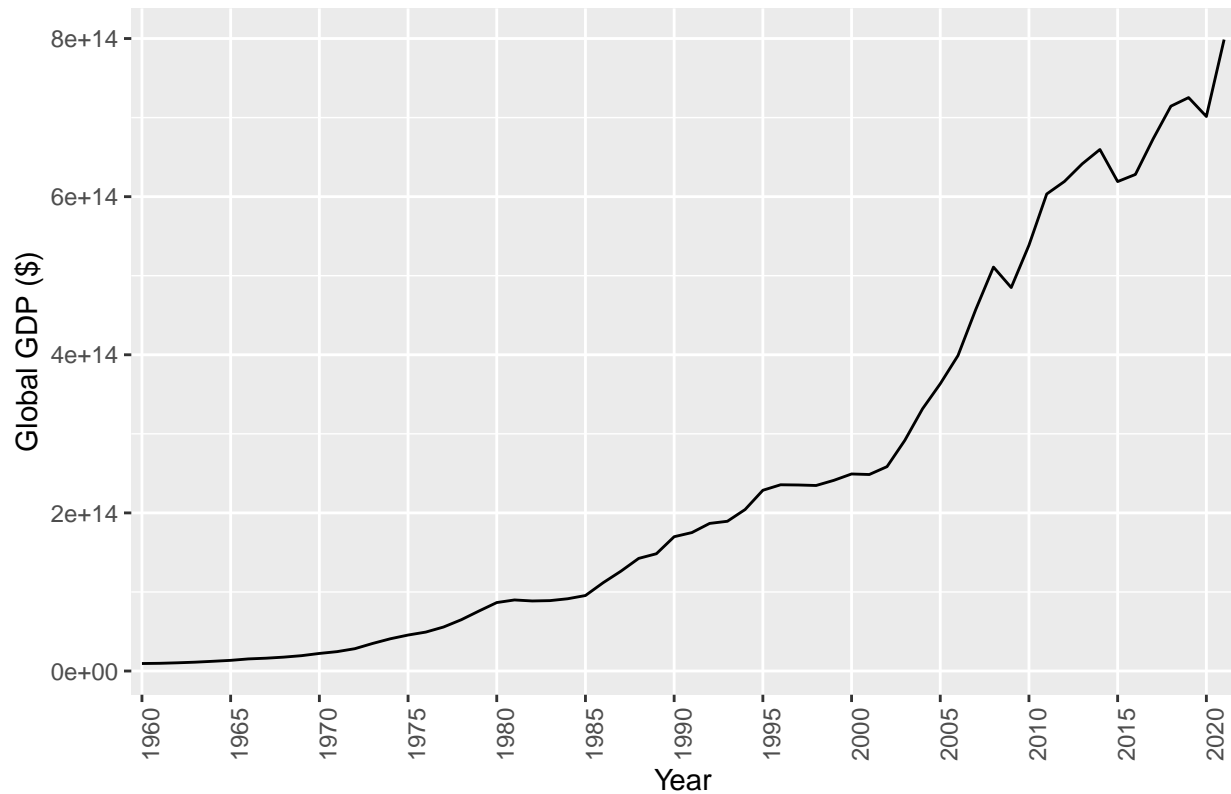
```
tmp = global_gdp_df$global_gdp
(tmp[2] - tmp[1]) / tmp[1] == global_gdp_df$percent_change[2]
```

```
## [1] TRUE
```

The TRUE output above means that the percent change calculation worked correctly and that these values can now be plotted:

```
ggplot(data = global_gdp_df) +
  geom_line(mapping = aes(x = year, y = global_gdp, group=1)) +
  labs(
    x = "Year",
    y = "Global GDP ($)",
    title = "Total Global GDP From 1960 - 2021",
  ) +
  scale_x_discrete(breaks=seq(1960, 2020, 5)) +
  theme(axis.text.x = element_text(angle = 90))
```

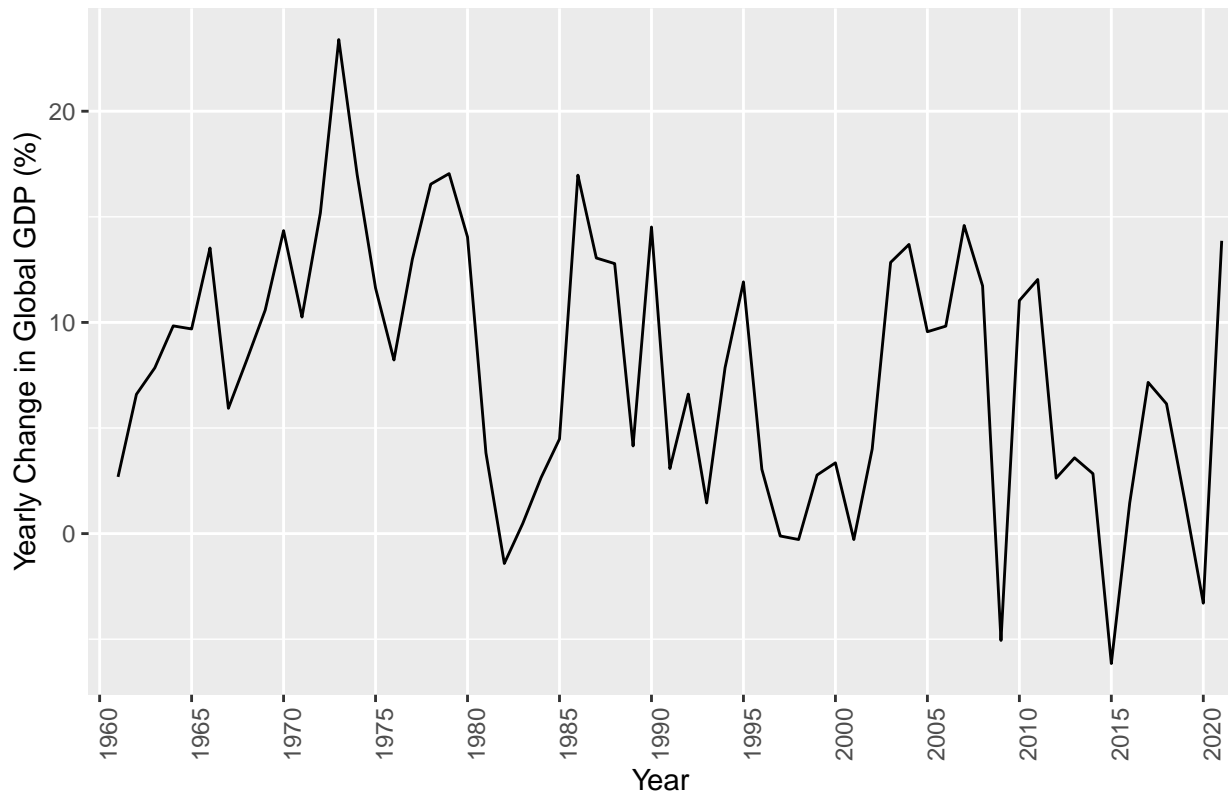
Total Global GDP From 1960 – 2021



```
ggplot(data = global_gdp_df) +
  geom_line(mapping = aes(x = year, y = percent_change * 100, group=1)) +
  labs(
    x = "Year",
    y = 'Yearly Change in Global GDP (%)',
    title = "Year Over Year Percent Change in GDP From 1961-2021 GDP",
  ) +
  scale_x_discrete(breaks=seq(1960, 2020, 5)) +
  theme(axis.text.x = element_text(angle = 90))
```

## Warning: Removed 1 row(s) containing missing values (geom\_path).

Year Over Year Percent Change in GDP From 1961–2021 GDP



In the above plots, we can make pinpoint the location of important global economic events. Most recently, we see the dips in global GDP (both value and percentage) in 2009 and 2020, due to the 2008 recession and Covid-19, respectively.

## Student Dataset

### Data Cleaning

The first step in cleaning the `student_df` dataframe is, once again, to clean the column names. For this dataframe, the only issue with the column names is that they contain `.` characters. They are removed below using the `str_replace_all` function.

```
new_col_names <-
  colnames(student_df) %>%
  str_replace_all("\\.", '_')

colnames(student_df) <- new_col_names
head(student_df, 1)
```

| ##   | id | name | phone | sex_and_age | test_number | term_1 | term_2 | term_3 |
|------|----|------|-------|-------------|-------------|--------|--------|--------|
| ## 1 | 1  | Mike | 134   | m_12        | test 1      | 76     | 84     | 87     |

The next data cleaning steps are to:

1. Transform the `sex_and_age` column into two separate columns, one containing the student's gender, and the other containing their age.



2. Remove the word “test” from the values in the `test_number` field, given that we already know the numbers correspond to different tests thanks to the column name.

Both of these steps are completed below using a single `mutate` in tandem with the `str_extract` function:

```
student_df <-
  student_df %>%
    mutate(gender = str_extract(sex_and_age, '(m|f)'),
           age = str_extract(sex_and_age, '\\d\\d'),
           test_no = strtoi(str_extract(test_number, '\\d')) %>%
             select(-sex_and_age, -test_number))

head(student_df)
```

```
##   id  name phone term_1 term_2 term_3 gender age test_no
## 1  1  Mike  134     76     84     87      m  12      1
## 2  2  Linda 270     88     90     73      f  13      1
## 3  3   Sam  210     78     74     80      m  11      1
## 4  4 Esther 617     68     75     74      f  12      1
## 5  5  Mary  114     65     67     64      f  14      1
## 6  6   Dan  114     81     67     90      f  14      1
```

The final step in this case, is to use the `pivot_longer` function to melt the “term” fields into a single column. This is done in the code chunk below:

```
cols_to_melt <- c('term_1', 'term_2', 'term_3')
student_df <- pivot_longer(student_df,
                           cols = all_of(cols_to_melt),
                           names_to = 'term',
                           values_to = 'test_score')
head(student_df)
```

```
## # A tibble: 6 x 8
##   id name phone gender age test_no term test_score
##   <int> <chr> <int> <chr> <chr>   <int> <chr>      <int>
## 1     1 Mike  134 m     12     1 term_1      76
## 2     1 Mike  134 m     12     1 term_2      84
## 3     1 Mike  134 m     12     1 term_3      87
## 4     2 Linda 270 f     13     1 term_1      88
## 5     2 Linda 270 f     13     1 term_2      90
## 6     2 Linda 270 f     13     1 term_3      73
```

Like with the original `test_number` column, we can remove the extraneous word “term” from the values in the `term` column:

```
student_df$term <- str_extract(student_df$term, '\\d')
head(student_df)
```

```
## # A tibble: 6 x 8
##   id name phone gender age test_no term test_score
##   <int> <chr> <int> <chr> <chr>   <int> <chr>      <int>
## 1     1 Mike  134 m     12     1 1      76
## 2     1 Mike  134 m     12     1 2      84
## 3     1 Mike  134 m     12     1 3      87
## 4     2 Linda 270 f     13     1 1      88
## 5     2 Linda 270 f     13     1 2      90
## 6     2 Linda 270 f     13     1 3      73
```

The output represents the final dataframe, which is now ready to be analyzed.

## Analysis

Given that the cleaned `student_df` dataframe contains information regarding students test scores for a number of different school terms, an interesting research task might be to check the normality of the distribution of the student's final letter grades after each term.

To do this, we first need to figure out each student's average test score after each term. This is done below using a `group_by` and `summarise` function:

```
avg_test_scores <-
  student_df %>%
    group_by(name, term) %>%
      summarise(avg_test_score = mean(test_score))

## `summarise()` has grouped output by 'name'. You can override using the
## `.groups` argument.

head(avg_test_scores)

## # A tibble: 6 x 3
## # Groups:   name [2]
##   name    term avg_test_score
##   <chr>  <chr>         <dbl>
## 1 Dan     1             84.5
## 2 Dan     2             74
## 3 Dan     3             89.5
## 4 Esther 1             69
## 5 Esther 2             75
## 6 Esther 3             76
```

The following code uses an if statement to categorize the student's final grades as letter grades:

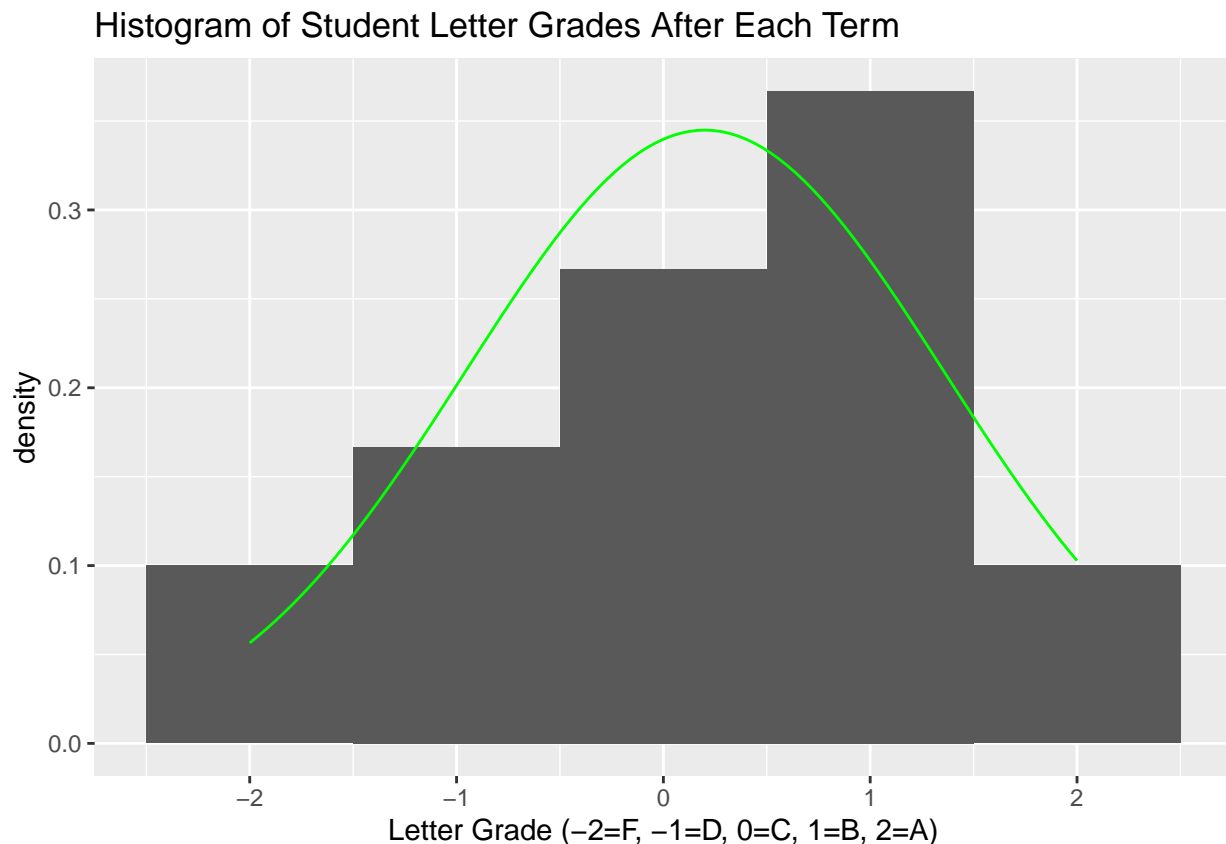
```
avg_test_scores <-
  avg_test_scores %>%
    mutate(
      letter_grade =
        ifelse(avg_test_score >= 90, 'A',
              ifelse(avg_test_score >= 80, 'B',
                    ifelse(avg_test_score >= 70, 'C',
                          ifelse(avg_test_score >= 65, 'D',
                                'F')))),
      hist_grade =
        ifelse(avg_test_score >= 90, 2,
              ifelse(avg_test_score >= 80, 1,
                    ifelse(avg_test_score >= 70, 0,
                          ifelse(avg_test_score >= 65, -1,
                                -2))))
    )

table(avg_test_scores$letter_grade)

##
##  A  B  C  D  F
##  3 11  8  5  3
```

The output above prints the distributions of letter grades from all students after each term, but in order to get a better idea of whether or not the data is normal it is plotted below along with a normal distribution:

```
ggplot(data = avg_test_scores, aes(x = hist_grade)) +  
  geom_blank() +  
  geom_histogram(aes(y = ..density..), bins=5) +  
  stat_function(fun = dnorm,  
               args = c(mean = mean(avg_test_scores$hist_grade),  
                         sd = sd(avg_test_scores$hist_grade)),  
               col = "green") +  
  labs(  
    x = "Letter Grade (-2=F, -1=D, 0=C, 1=B, 2=A)",  
    title = "Histogram of Student Letter Grades After Each Term",  
  )
```



Because a histogram requires numeric values to generate the bins, the `%x%` label defines the scale that was used to produce the graphic. These values were stored in the `hist_grade` column of the `avg_test_scores` dataframe, and it was the mean and standard deviation of this column that was used to create the normal curve seen in green. A quick visual analysis of this graphic does show that the data appears normal, but this assumption could definitely be made more clear if we had more data.

## Atmosphere Dataset

### Cleaning Data

The cell below gives another look at the unclean `atmosphpere_df` dataframe:

```
head(atmos_df)
```

```
##   X Altitude X.Air.press   X.ppO2   X.Alv.pO2   X...sat.O2 X.Alv.pCO2
## 1 1      Ft/m      mmHg Air=21% mmHg on air >90% desired >35 best
## 2 2 Sea level      760      159      104      97      40
## 3 3   10k/3k      523      110      67      90      36
## 4 4  20k/6.1k      349      73      40      73      24
## 5 5  30k/9.1k      226      47      18      24      24
## 6 6  40k/12k      141      29      NaN      NaN      NaN
##   X.Alv.pO2.with.O2.100. X...sat.O2_1 X.Alv.pCO2_1
## 1      NA      100% O2      100% O2
## 2      673      100      40
## 3      436      100      40
## 4      262      100      40
## 5      139      99      40
## 6       58      84      36
```

The first step in this case is to remove the extraneous X index column of the dataframe, as R automatically indexes rows of a dataframe itself. This is done in the cell below:

```
atmos_df <- atmos_df %>%
  select(-X)
head(atmos_df)
```

```
##   Altitude X.Air.press   X.ppO2   X.Alv.pO2   X...sat.O2 X.Alv.pCO2
## 1      Ft/m      mmHg Air=21% mmHg on air >90% desired >35 best
## 2 Sea level      760      159      104      97      40
## 3   10k/3k      523      110      67      90      36
## 4  20k/6.1k      349      73      40      73      24
## 5  30k/9.1k      226      47      18      24      24
## 6  40k/12k      141      29      NaN      NaN      NaN
##   X.Alv.pO2.with.O2.100. X...sat.O2_1 X.Alv.pCO2_1
## 1      NA      100% O2      100% O2
## 2      673      100      40
## 3      436      100      40
## 4      262      100      40
## 5      139      99      40
## 6       58      84      36
```

Our next step is to once again clean the column names of the `atmos_df` dataframe. There are a number of cleaning steps that need to be performed, each of which is listed below:

1. Remove the X characters from the column names.
2. Remove the . characters in front of the column names.
3. Replace the . characters inside the column names with \_ characters.

Each of these steps is performed below, and is labeled via a comment in the code:

```
new_col_names <-
  colnames(atmos_df) %>%
    str_replace_all('X', '') %>% #1
    str_replace_all('^\\.\\.\\.\\.', '') %>% #2
    str_replace_all('\\.\\.\\.\\.', '_') %>% #3
    tolower()

colnames(atmos_df) <- new_col_names
head(atmos_df)
```

```
##      altitude air_press      ppo2      alv_po2      sat_o2 alv_pco2
## 1      Ft/m      mmHg Air=21% mmHg on air >90% desired >35 best
## 2 Sea level      760      159      104      97      40
## 3 10k/3k      523      110      67      90      36
## 4 20k/6.1k      349      73      40      73      24
## 5 30k/9.1k      226      47      18      24      24
## 6 40k/12k      141      29      NaN      NaN      NaN
## alv_po2_with_o2_100_ sat_o2_1 alv_pco2_1
## 1      NA 100% O2      100% O2
## 2      673      100      40
## 3      436      100      40
## 4      262      100      40
## 5      139      99      40
## 6      58      84      36
```

The next data cleaning step involves removing the first row of the dataframe: inspecting this row in the output above reveals that this row just contains notes on what the measurements are in each column. Because it doesn't have any measurements itself, it should be removed. However, in order for the information in the row to still be accessible, it is saved in the `field_info` dataframe before it is deleted from `atmos_df`.

```
field_info <- atmos_df[1,]
atmos_df <- atmos_df[-1,]
rownames(atmos_df) <- NULL
head(atmos_df)
```

```
##      altitude air_press ppo2 alv_po2 sat_o2 alv_pco2 alv_po2_with_o2_100_
## 1 Sea level      760 159      104      97      40      673
## 2 10k/3k      523 110      67      90      36      436
## 3 20k/6.1k      349 73      40      73      24      262
## 4 30k/9.1k      226 47      18      24      24      139
## 5 40k/12k      141 29      NaN      NaN      NaN      58
## 6 50k/15.2k      87 18      NaN      NaN      NaN      16
## sat_o2_1 alv_pco2_1
## 1      100      40
## 2      100      40
## 3      100      40
## 4      99      40
## 5      84      36
## 6      15      24
```

The next step is to clean the altitude column, as this represents the single “categorical” variable we will use for the final dataframe. Upon inspection of the values in this column, there are a number of data cleaning steps that need to be performed:

1. The “Sea Level” value should be replaced with 0.
2. The feet and meter measurements need to be separated into their own separate columns.
3. Each value needs to be multiplied by 1,000, as indicated by the `k` characters.

Steps 2 and 3 are performed below using a `mutate` in tandem with a combination of `str_extract` and `str_replace` functions.

```
atmos_df <-
  atmos_df %>%
  mutate(
    altitude_ft = strtoi(str_extract(altitude, '\\d\\d\\d')) * 1000,
    altitude_m = str_extract(altitude, '\\/(\\d\\d\\d?\\.?\\d?k)') %>%
      str_replace('k', '') %>%
```

```

      str_replace('\\', '/', ' ') %>%
      as.numeric() * 1000
    )

atmos_df <-
  cbind(
    select(atmos_df, altitude_ft, altitude_m),
    select(atmos_df, -altitude_ft, -altitude_m)
  )
head(atmos_df)

##   altitude_ft altitude_m altitude air_press ppo2 alv_po2 sat_o2 alv_pco2
## 1          NA          NA Sea level      760  159    104    97     40
## 2      10000      3000    10k/3k      523  110     67    90     36
## 3      20000      6100    20k/6.1k    349   73     40    73     24
## 4      30000      9100    30k/9.1k    226  47     18    24     24
## 5      40000     12000    40k/12k    141  29     NaN    NaN    NaN
## 6      50000     15200    50k/15.2k     87  18     NaN    NaN    NaN
##   alv_po2_with_o2_100_ sat_o2_1 alv_pco2_1
## 1              673      100      40
## 2              436      100      40
## 3              262      100      40
## 4              139      99      40
## 5               58      84      36
## 6               16      15      24

```

We can see now that the `altitude_ft` and `altitude_m` fields have been created, and that they are now numeric fields as opposed to a single character field. Finally, the cell below uses the `replace_na` function to fill the single NA value in these columns with 0 (sea level elevation). It also removes the old `altitude` column, as it is no longer required.

```

atmos_df$altitude_ft <- replace_na(atmos_df$altitude_ft, 0)
atmos_df$altitude_m <- replace_na(atmos_df$altitude_m, 0)
atmos_df <- select(atmos_df, -altitude)

atmos_df

##   altitude_ft altitude_m air_press ppo2 alv_po2 sat_o2 alv_pco2
## 1           0           0      760  159    104    97     40
## 2      10000      3000      523  110     67    90     36
## 3      20000      6100      349   73     40    73     24
## 4      30000      9100      226  47     18    24     24
## 5      40000     12000      141  29     NaN    NaN    NaN
## 6      50000     15200       87  18     NaN    NaN    NaN
##   alv_po2_with_o2_100_ sat_o2_1 alv_pco2_1
## 1              673      100      40
## 2              436      100      40
## 3              262      100      40
## 4              139      99      40
## 5               58      84      36
## 6               16      15      24

```

The next step is to change the data types of the remaining character columns, as they are all actually numerical measurement values. While all the values seen above are technically integers, the types of measurements being taken are taken on a continuous scale. As such, these fields are all converted to a numeric type so that

any future measurements that might be added will be able to have decimal point values.

```
atmos_df <-atmos_df %>%
  mutate_if(is.character, as.numeric) %>%
  mutate_if(is.integer, as.numeric)
head(atmos_df)
```

|      | altitude_ft | altitude_m | air_press | ppo2 | alv_po2 | sat_o2 | alv_pco2 |
|------|-------------|------------|-----------|------|---------|--------|----------|
| ## 1 | 0           | 0          | 760       | 159  | 104     | 97     | 40       |
| ## 2 | 10000       | 3000       | 523       | 110  | 67      | 90     | 36       |
| ## 3 | 20000       | 6100       | 349       | 73   | 40      | 73     | 24       |
| ## 4 | 30000       | 9100       | 226       | 47   | 18      | 24     | 24       |
| ## 5 | 40000       | 12000      | 141       | 29   | NaN     | NaN    | NaN      |
| ## 6 | 50000       | 15200      | 87        | 18   | NaN     | NaN    | NaN      |

|      | alv_po2_with_o2_100_ | sat_o2_1 | alv_pco2_1 |
|------|----------------------|----------|------------|
| ## 1 | 673                  | 100      | 40         |
| ## 2 | 436                  | 100      | 40         |
| ## 3 | 262                  | 100      | 40         |
| ## 4 | 139                  | 99       | 40         |
| ## 5 | 58                   | 84       | 36         |
| ## 6 | 16                   | 15       | 24         |

Now that all the measurement columns have been converted to numeric fields, we can complete the final data cleaning step: converting the dataframe into a long format. In this case, the altitude values represent our “categorical” fields, as each atmospheric measurement was taken at every one of the included heights. The remaining measurement fields are then all melted into a single `measurement_type` field, with the actual measurements being stored in a `measurement` column. This is done below using the `pivot_longer` function:

```
atmos_df <-
  pivot_longer(atmos_df, cols = !c(altitude_ft, altitude_m),
               names_to = 'measurement_type', values_to = 'measurement')
head(atmos_df)
```

```
## # A tibble: 6 x 4
##   altitude_ft altitude_m measurement_type      measurement
##       <dbl>      <dbl> <chr>                <dbl>
## 1         0         0 air_press                760
## 2         0         0 ppo2                  159
## 3         0         0 alv_po2                104
## 4         0         0 sat_o2                 97
## 5         0         0 alv_pco2                40
## 6         0         0 alv_po2_with_o2_100_    673
```

The output above represents our final dataframe. The benefit of having the data in this format is that now different measurements can now be easily added as rows. This includes measurements of fields that already exist in the `measurement_type` column, but it is also easy to add a completely new measurement type without having to create a completely new field.

## Analysis

One interesting research question for this data might be to see the correlation between air pressure and the air pressure due to oxygen (`ppo2` in the dataframe) as we go higher and higher up into the sky. To do this, we can look at a scatter plot of this data. First, the cell below gathers the required information:

```
plt_df <-
  atmos_df %>%
    filter(measurement_type == 'air_press' | measurement_type == 'ppo2') %>%
    pivot_wider(names_from = measurement_type, values_from = measurement)

head(plt_df)
```

```
## # A tibble: 6 x 4
##   altitude_ft altitude_m air_press ppo2
##       <dbl>      <dbl>    <dbl> <dbl>
## 1         0         0        760   159
## 2      10000       3000        523   110
## 3      20000       6100        349    73
## 4      30000       9100        226    47
## 5      40000      12000        141    29
## 6      50000      15200         87    18
```

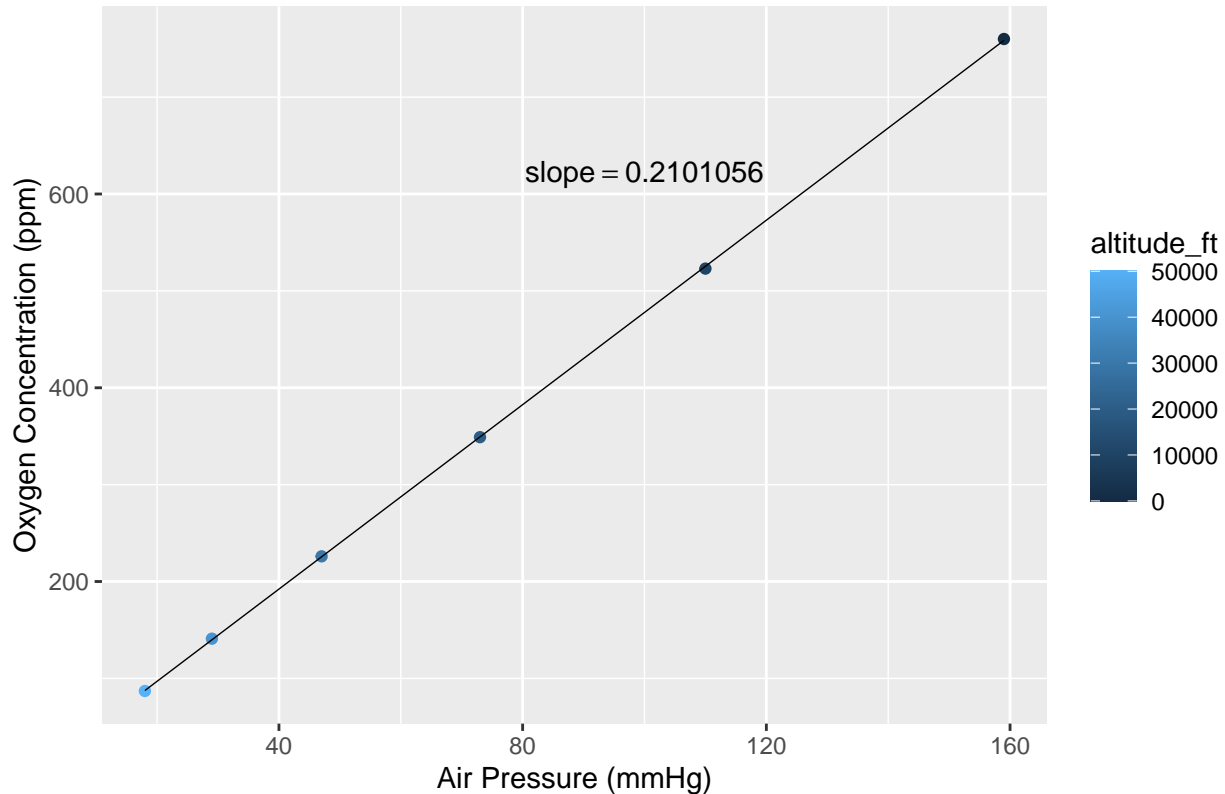
Next, the cell below plots air pressure as a function of oxygen concentration:

```
ggplot(plt_df, aes(x=ppo2, y=air_press, color = altitude_ft)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, size=0.25, color='black') +
  labs(
    x = "Air Pressure (mmHg)",
    y = "Oxygen Concentration (ppm)",
    title = "Oxygen Correlation vs. Air Pressure At Varying Heights",
  ) +
  annotate("text", x=100, y=620,
    label=(paste0("slope==", coef(lm(plt_df$ppo2~plt_df$air_press))[2])),
    parse=TRUE)
```

```
## `geom_smooth()` using formula 'y ~ x'
```



## Oxygen Correlation vs. Air Pressure At Varying Heights



It is clear in the plot above that there is an almost perfect correlation between the total air pressure and the air pressure due to oxygen as altitude increases. The fact that they are so perfectly correlated indicates that the concentration of oxygen does not change as altitude increases, which is consistent with what we know about our atmosphere. This also means that the slope of the line (which represents oxygen air pressure divided by total air pressure) should give the concentration of oxygen in our atmosphere. This is confirmed by the fact that the slope of the line is about 0.21, which is almost the exact proportion of oxygen atoms in our atmosphere.

## Conclusion

The work done above gives three examples of how unclean data sets can be tidied in order to answer hypothetical research questions. While the specifics involved in cleaning and analyzing each data set were different, it is important to note that many of the steps were similar, and the process in general is pretty much the same. These examples help to highlight that data cleaning is an essential part of any data science process.