```
# # Data Science Learning Objectives:
#
# # Importing data from various types of Excel files and CSV files
# # Apply action verbs in dplyr for data wrangling
# # How to pivot between “long” and “wide” datasets
# Joining together multiple data sets using dplyr
# How to create effective longitudinal data visualizations with ggplot2
# How to add text, color, and labels to ggplot2 plots
# How to create faceted ggplot2 plots

# Statistical Learning Objectives:
# Introduction to correlation coefficient as a summary statistic
# Relationship between correlation and linear regression
# Correlation is not causation

# Required packages:


install.packages('here')
install.packages('readxl')
install.packages('readr')
install.packages('dplyr')
install.packages('magrittr')
install.packages('stringr')
install.packages('purrr')
install.packages('tidyr')
install.packages('forcats')
install.packages('ggplot2')
install.packages('directlabels')
install.packages('ggrepel')
install.packages('broom')
install.packages('patchwork')
install.packages('tidyverse')
install.packages('tinytex')



library('here')
library('readxl')
library('readr')
library('dplyr')
library('magrittr')
library('stringr')
library('purrr')
library('tidyr')
library('forcats')
library('ggplot2')
library('directlabels')
library('ggrepel')
library('broom')
library('patchwork')
library('tidyverse')
library('tinytex')
tinytex::install_tinytex()

# Import datasets using readxl function and saving dataframe

co2_emissions_1 <- readxl::read_xlsx(here("data","raw","yearly_co2_emissions_1000_tonnes.xlsx"))
gdp_growth_1 <- readxl::read_xlsx(here("data","raw","gdp_per_capita_yearly_growth.xlsx"))
energy_use_1 <- readxl::read_xlsx(here("data","raw","energy_use_per_person.xlsx"))
us_disaster_1 <- readxl::read_xlsx(here("data","raw","us_natural_disasters.xlsx"), skip = 2)
us_temperature_1 <- readxl::read_xlsx(here("data","raw","temperatures.xlsx"), skip = 4)

co2_emissions_1 %>%
```

```
  slice_head(n = 3)

set.seed(123)

co2_emissions_1 %>%
  slice_sample(n = 3)

co2_emissions_1 %>%
  dplyr::glimpse()

co2_emissions_1 %>%
  select(country)

names(co2_emissions_1)

# Save and import data as RDA file
save(co2_emissions_1,
     gdp_growth_1,
     energy_use_1,
     us_disaster_1,
     us_temperature_1,
     file = here::here("data", "imported", "co2_data_imported.rda"))

load(here::here("data", "imported", "co2_data_imported.rda"))

# convert wide data format to long format using pivot_longer

co2_emissions_1 %<>%
  pivot_longer(cols = -country,
             names_to = "Year",
             values_to = "Emissions")
set.seed(123)

co2_emissions_1 %>%
  slice_sample(n = 6)


# Rename rows
co2_emissions_1 %<>%
  dplyr::rename(Country = country) %>%
  dplyr::mutate(Year = as.numeric(Year),
              Label = "CO2 Emissions (Metric Tons)")

set.seed(123)
co2_emissions_1 %>%
  slice_sample(n = 6)

# pull out distinct country names using distinct()

co2_emissions_1 %>%
  distinct(Country) %>%
  pull()

# Check gdp_growth df
gdp_growth_1 %>%
  slice_head(n = 3)

# use dim() function to evaluate dimensions of an object
dim(gdp_growth_1)

names(gdp_growth_1)

glimpse(gdp_growth_1)
colnames(gdp_growth_1)
```

```
# convert wide data format to long format using pivot_longer for gdp_growth
# Rename rows using rename & mutate

gdp_growth_1 %<>%
  pivot_longer(cols = -country,
               names_to = "Year",
               values_to = "gdp_growth") %>%
  rename(Country = country) %>%
  mutate(Year = as.numeric(Year),
         Label = "GDP Growth/Capita (%)") %>%
  rename(GDP = gdp_growth)


gdp_growth_1 %>%
  slice_head(n = 6)

gdp_growth_1 %>%
  count(Year)

# pullout distinct country names
gdp_growth_1 %>%
  distinct(Country) %>%
  pull()

# energy_use data
energy_use_1 %>%
  slice_head(n = 3)

energy_use_1 %>%
  glimpse()


# convert wide data format to long format using pivot_longer for energy_use
# Rename rows using rename & mutate
energy_use_1 %<>%
  pivot_longer(cols = -country,
               names_to = "Year",
               values_to = "energy_use") %>%
  rename(Country = country) %>%
  mutate(Year = as.numeric(Year),
         Label = "Energy Use (kg, oil-eq./capita)") %>%
  rename(Energy = energy_use)

set.seed(123)

energy_use_1 %>%
  slice_sample(n = 3)

# check country variable with pull() function

energy_use_1 %>%
  distinct(Country)%>%
  pull()

# Take a look at the disasters in the US

glimpse(us_disaster_1)

#  Select variables "Year" using count count and contains

us_disaster_1 %>%
  select(Year, contains("Count"))

us_disaster_1 %>%
  slice_head(n = 6)
```

```
# Create variable that will sum of all the different types of disasters each year
# with rowSums() function

yearly_disasters <- us_disaster_1 %>%
  select(-Year) %>%
  rowSums()
yearly_disasters

# Add the same to the us_disaster tibble using bind_cols function of dplyr package
us_disaster_1 %>%
  bind_cols(Disasters = yearly_disasters)

# However, we can actually create and add this new variable directly to the us_disaster tibble by
using the mutate() function of dplyr and using the . notation.

us_disaster_1 %<>%
  mutate(Disasters = rowSums(select(., -Year)))

us_disaster_1 %>%
  glimpse()

# Great, now we are going to remove some of these variables and just keep the variables of
interest using select().

# We are also going to add a new variable called Country to indicate that this data is from the
United States. Again this will create a new variable where every value is United States.

us_disaster_1 %<>%
  dplyr::select(Year, Disasters) %>%
  mutate(Country = "United States") %>%
  pivot_longer(cols = c(-Country, -Year),
               names_to = "Indicator",
               values_to = "Value") %>%
  mutate(Label = "Number of Disasters")

us_disaster_1 %>%
  slice_head(n = 6)

# Temperature

us_temperature_1 %>%
  slice_head(n = 6)

# Fix date column using stringr package

us_temperature_1 %>%
  pull(Date) %>%
  str_length()

# As the values are are 6 characters long lets check that they end with "12" by specifying what
pattern to look for with pull()

us_temperature_1 %>%
  pull(Date) %>%
  str_ends(pattern = "12")

# As the values are TRUE for all the variables in Date ending with "12"
# We should use str_sub() function of the stringr package to remove "12" from each Date value
#  by indicating start & stop characters
# In this case the start would be 1 and the 4th character would be where we want to stop, so we
would use start = 1, stop = 4. We can do this inside of the mutate() function to modify the Date
variable. In doing so, we will not need to use pull() to pull the values for the Date variable.

us_temperature_1 %<>%
```

```
  mutate(Date = str_sub(Date, start = 1, end = 4))

us_temperature_1

# We also want to remove the Anomaly variable, which is an indicator of how different the national
average temperature for that year was from the average temperature from 1901-2000 which was
52.02°F.

# Then, we also want to create a Country variable. We will also change the name of the Date
variable to Year so that it will be consistent with our other datasets. We also also want the Year
to be numeric. We can accomplish both renaming and changing to numeric by using the mutate()
function.

# We also want to create an Indicator variable so that we can later tell what data the values in
this tibble represent if we combine it with other tibbles and a Label variable, so that we will
have informative labels if we make a plot with this data later.

# Finally, we remove the Date variable and also order the columns just like the other us data
using the select() function.

us_temperature_1 %<>%
  dplyr::select(-Anomaly) %>%
  mutate(Country = "United States",
         Year = as.numeric(Date),
         Indicator = "Temperature",
         Label = "Temperature (Fahrenheit)") %>%
  select(Year, Country, Indicator, Value, Label)

us_temperature_1 %>%
  slice_head(n = 6)

# Joining the Data using DPLYR package
# but first, let us check the consistency of the data by using summary() function

summary(co2_emissions_1)
summary(energy_use_1)
summary(gdp_growth_1)

# Use full_join statements by using Country & Year columns which are present in all the datasets
and tend to overlap
# even though label is present but they dont overlap
# specify columns/variables we will be joining by sing the by= argument in the full_join()
function

data_wide_1 <- co2_emissions_1 %>%
  full_join(gdp_growth_1, by = c("Country", "Year", "Label")) %>%
  full_join(energy_use_1, by = c("Country", "Year", "Label"))

set.seed(123)

data_wide_1 %>%
  slice_sample(n = 6)

# save(data_wide_1, file = here::here("www", "data", "exercise", "wide_data.rda"))

data_wide_1 %>%
  glimpse()

data_long_1 <- data_wide_1 %>%
  pivot_longer(cols = c(-Country, -Year, -Label),
               names_to = "Indicator",
               values_to = "Value")

set.seed(123)
data_long_1 %>%
```

```
  slice_sample(n = 6)

# combine the above data with US data

us_disaster_1 %>%
  slice_head(n = 6)

us_temperature_1 %>%
  slice_head(n = 6)

# We will now use the bind_rows() function of the dplyr package which will just append the
us_temperature data and the us_disaster data after the data_long data.

data_long_1 <-
  list(data_long_1, us_disaster_1, us_temperature_1) %>%
  bind_rows() %>%
  mutate(Country = as.factor(Country))

# We also converted the Country column to a factor in the last line of the code chunk.

# We can check the top and bottom of the new data_long tibble to see that our us_temperature data
is at the bottom. To see the end of our tibble we can use slice_tail() function of the dplyr
package.

data_long_1 %>%
  slice_head(n = 6)

data_long_1 %>%
  slice_tail(n = 6)

set.seed(123)
data_long_1 %>%
  slice_sample(n = 10)

# Example of difference between full_join and bond_rows function
data_wide_br <-
  list(co2_emissions_1, gdp_growth_1, energy_use_1) %>%
  bind_rows()

data_wide_br %>%
  filter(Country == "India",
         Year == 2000)

# full_join
data_wide_fj_label <-
  list(co2_emissions_1, gdp_growth_1, energy_use_1) %>%
  reduce(full_join, by = c("Country", "Year", "Label"))

data_wide_fj_label %>%
  filter(Country == "India", Year == "2000")

dim(data_wide_br)

dim(data_wide_fj_label)

setequal(data_wide_fj_label, data_wide_br)

# join only country and label

data_wide_fj <-
  list(co2_emissions_1, gdp_growth_1, energy_use_1) %>%
  reduce(full_join, by = c("Country", "Year"))

data_wide_fj %>%
  filter
```

```
# We will create a new variable called Region that will indicate if the data is about the United
States or a different country based on the values in the Country variable. To do this, we will use
the case_when() function of the dplyr package.

# For example, if the Country variable is equal to "United States" the value for the new variable
will also be "United States", where as if the Country variable is not equal to "United States" but
is some other character string value, such as "Afghanistan", then the value for the new variable
will be "Rest of the World". We can specify that something is not equal by using the != operator.

# The new values for the new variable Region are indicated after the specific conditional
statements by using the ~ symbol.

data_long_1 %<>%
  mutate(Region = case_when(Country == "United States" ~ "United States",
                            Country != "United States" ~ "Rest of the World"))

data_long_1 %>%
  arrange(Country) %>%
  slice_head(n = 6)

# Remove NA values in countries using drop_na() of tidr package to remove missing values

data_long_with_miss <-
  data_long_1 %>%
  arrange(Country)

data_long_1 %<>%
  drop_na() %>%
  arrange(Country)

data_long_1 %>%
  slice_head(n = 6)


# Plot CO2 emissions over the years using filter() function of dplyr package. Keep all rows where
the Indicator variable is equal to the word Emissions

data_long_1 %>%
  filter(Indicator == "Emissions")

# next, sum the emissions across countries for each year by using group_by() and summarize()
function

data_long_1 %>%
  filter(Indicator == "Emissions") %>%
  group_by(Year) %>%
  summarize(Emissions = sum(Value))

# Define the x-axis as Year variable & y-axis as Value variable, grouped by Country variable
# As we see co2 levels has risen sharply, we have to use labs() function


data_long_1 %>%
  filter(Indicator == "Emissions") %>%
  group_by(Year) %>%
  summarize(Emissions = sum(Value)) %>%
  ggplot(aes(x = Year, y = Emissions)) +
  geom_line(size = 1.5) +
  labs(title = "World" ~CO[2]~ "Emissions per Year (1751-2014)",
       caption = "Limited to reporting countries",
       y = "Emissions (Metric Tonnes)")

# Use theme() function to change the font size of the x-axis, y- axis, axis titles and captions
```

```
data_long_1 %>%
  filter(Indicator == "Emissions") %>%
  group_by(Year) %>%
  summarize(Emissions = sum(Value)) %>%
  ggplot(aes(x = Year, y = Emissions)) +
  geom_line(size = 1.5) +
  labs(title = "World" ~CO[2]~ "Emissions per Year (1751-2014)",
       caption = "Limited to reporting countries",
       y = "Emissions (Metric Tonnes)") +
  theme_linedraw() +
  theme(axis.text.x = element_text(size = 12),
        axis.text.y = element_text(size = 12),
        axis.title.x = element_text(size = 12),
        axis.title.y = element_text(size = 12),
        plot.caption = element_text(size = 12),
        plot.title = element_text(size = 16))

# save the theme in a dataframe to avoid re-typing

my_theme <-
theme_linedraw() +
  theme(axis.text.x = element_text(size = 12),
        axis.text.y = element_text(size = 12),
        axis.title.x = element_text(size = 12),
        axis.title.y = element_text(size = 12),
        plot.caption = element_text(size = 12),
        plot.title = element_text(size = 16))

# Use the newly created my_theme df to be added in the plot and save the plot in an object called
CO2_world

CO2_world <-
data_long_1 %>%
  filter(Indicator == "Emissions") %>%
  group_by(Year) %>%
  summarize(Emissions = sum(Value)) %>%
  ggplot(aes(x = Year, y = Emissions)) +
  geom_line(size = 1.5) +
  labs(title = "World" ~CO[2]~ "Emissions per Year (1751-2014)",
       caption = "Limited to reporting countries",
       y = "Emissions (Metric Tonnes)") +
  my_theme

# save the plot into a RDA file using function save () function
# use dev.off() function to close the graphical device that we will use to create the png version
of the plot so that we are ready to make another plot like this.

save(CO2_world, file =here::here("plots", "CO2_world.rda"))
png(here::here("plots", "CO2_world.png"))
CO2_world
dev.off()

# start finding the most and leat contributors of CO2 emissions and how it has changed overtime

data_long_1 %>%
  filter(Indicator == "Emissions") %>%
  ggplot(aes(x = Year, y = Value, group = Country)) +
  geom_line() +
  ylab("Emissions") +
  my_theme

CO2_countries <-
  data_long_1 %>%
  filter(Indicator == "Emissions") %>%
  ggplot(aes(x = Year, y = Value, group = Country)) +
```

```
  geom_line(alpha = 0.4) +
  labs(title = "Country" ~CO[2]~ "Emissions per Year (1751-2014)",
       caption = "Limited to reporting countries",y = "Emissions(Metric Tonnes)") +
       my_theme

CO2_countries

CO2_countries +
  geom_line(data = data_long_1 %>%
              filter(Indicator == "Emissions",
                     Country == "United States"),
            aes(x = Year, y = Value, color = Country)) +
  scale_colour_manual(values = c("red"))

# As USA is one of the largest countries with emissions let us find which are the top 10 emitting
countries for 2014 the last
#  year in the dataset using desc() function of dplyr package and arrange() the output

top_10_count <-
  data_long_1 %>%
  filter(Indicator == "Emissions", Year == 2014) %>%
  mutate(rank = dense_rank(desc(Value))) %>%
  filter(rank <= 10) %>%
  arrange(rank)

top_10_count


# Create a plot containing the 10 countries and differentiate by color

Top10b <- data_long_1 %>%
  filter(Country %in% pull(top_10_count, Country)) %>%
  filter(Indicator == "Emissions") %>%
  filter(Year >= 1900) %>%
  ggplot(aes(x = Year, y = Value, color = Country)) +
  geom_line() +
  scale_color_viridis_d() +
  labs(title = "Top 10 Emissions-producing Countries in 2010 (1900-2014)",
      subtitle = "Ordered by Emissions Produced in 2014",
      y = "Emissions (Metric Tons)",
      x = "Year") +
  my_theme

Top10b

# Add text-labels to the plot denote lines by country name directly

Top10b +

  geom_text(data = data_long_1 %>%
              filter(Country %in% pull(top_10_count, Country)) %>%
              filter(Indicator == "Emissions") %>%
              filter(Year == last(Year)),
              aes(label = Country)) +
  theme(legend.position = "none")

# Add check_overlap argument within geom_text to remove overlapping variables

Top10b +

  geom_text(data = data_long_1 %>%
              filter(Country %in% pull(top_10_count, Country)) %>%
              filter(Indicator == "Emissions") %>%
              filter(Year == last(Year)),
            aes(label = Country),
```

```
            check_overlap = TRUE) +
  scale_x_continuous(expand = c(0.15, 0)) +
  theme(legend.position = "none")

# Format labels
direct.label(Top10b) +
  scale_x_continuous(expand = c(0.3, 0))

# Alternative method using geom_dl() function

Top10b +
  scale_x_continuous(expand = c(0.3, 0)) +
  geom_dl(aes(label = Country), method = list("last.bumpup")) +
  theme(legend.position = "none")

# Highlight names using last.polygon method

direct.label(Top10b, method = list("last.polygons")) +
  scale_x_continuous(expand = c(0.3, 0))

# For better clarity we will use geom_text_repel() function from the ggrepel package to avid
overlapping and better control
# on the plot

Top10b +
  geom_text_repel(data = data_long_1 %>%
                    filter(Country %in% pull(top_10_count, Country)) %>%
                  filter(Indicator == "Emissions") %>%
                  filter(Year == last(Year)),
                  aes(label = Country, x = Year, y = Value)) +
  theme(legend.position = "none") +
  scale_x_continuous(expand = c(0.3, 0))

# Add pointing indicators in the line plot

Top10b +
  geom_text_repel(data = data_long_1 %>%
                    filter(Country %in% pull(top_10_count, Country)) %>%
                    filter(Indicator == "Emissions") %>%
                    filter(Year == last(Year)),
                  aes(label = Country, x = Year, y = Value),
                  nudge_x = 10,
                  hjust = 1,
                  vjust = 1,
                  segment.size = 0.25,
                  force = 1) +
  theme(legend.position = "none") +
  scale_x_continuous(expand = c(0.3, 0)) +
  scale_x_continuous(expand = c(0.3, 0))

# Save the line plot
save(Top10b, file =here::here("plots", "Top10b.rda"))
png(here::here("plots", "Top10b.png"))
Top10b
dev.off()


# Create Tile plot using geom_tile and fct_reorder() function of the forcats to include and order
# the countries based on last emission year

Top10t <-
  data_long_1 %>%
  filter(Country %in% pull(top_10_count, Country)) %>%
  filter(Indicator == "Emissions") %>%
  filter(Year >= 1900) %>%
```

```
  ggplot(aes(x = Year, y = fct_reorder(Country, Value, last))) +
  geom_tile(aes(fill = log(Value))) +
  scale_fill_viridis_c()

Top10t <- Top10t +
  scale_x_continuous(breaks = seq(1900, 2014, by = 5),
                     labels = seq(1900, 2014, by = 5)) +
  labs(title = "Top 10" ~CO[2]~ "Emission-producing Countries",
       subtitle = "Ordered by Emissions Produced in 2014",
       fill = "Ln(CO2 Emissions (Metric Tonnes))") +
  theme_classic() +
  theme(axis.text.x = element_text(size = 12, angle = 90, color = "black"),
        axis.text.y = element_text(size = 12, color = "black"),
        axis.title = element_blank(),
        plot.caption = element_text(size = 12),
        plot.title = element_text(size = 16),
        legend.position = "bottom")
Top10t

# Save the plot

save(Top10t, file =here::here("plots", "Top10t.rda"))
png(here::here("plots", "Top10t.png"))
Top10t
dev.off()

# Germany had a very low emission rate post 1945, US has seen an year on year increase
# However, China surpassed all
# white portion indicates there are no emission data for that country.
# let us add more than 1 variable by using facet_wrap() function of ggplot2
# and create sub-plots simultaneously for 'Disasters', 'Emissions', 'Energy'
# 'GDP', 'Temperature'

ggplot(data_long_1, aes(x = Year, y = Value, group = Country)) +
  geom_line(alpha = 0.2) +
  geom_line(data = data_long_1 %>%
            filter(Country == "United States"),
            aes(x = Year, y = Value, color = Country)) +
  scale_colour_manual(values = c("blue")) +
  labs(title = "Distribution of Indicators by Year and Value",
       y = "indicator Value") +
  my_theme +
  theme(strip.text = element_text(size = 16, face = "bold")) +
  facet_wrap(Indicator ~.,
             scales = "free_y",
             strip.position = "right",
             ncol = 1)

data_long_1 %>%
  filter(!(Indicator %in% c("Disasters", "Temparature"))) %>%
  ggplot(aes(x = Year, y = Value, group = Country)) +
  geom_line() +
  facet_grid(Indicator ~ Region, scales = "free_y") +
  labs(title = "Distribution of indicators by Year and Value",
       y = "Indicator Value") +
  my_theme +
  theme(strip.text = element_text(size = 16, face = "bold"))



# Line Segment plot and calculating Mean from the variables

data_long_us <-
  data_long_1 %>%
  filter(Country == "United States", Year >= 1980, Year <= 2010) %>%
```

```
  group_by(Indicator) %>%
  mutate(Mean = mean(Value), Diff_from_mean = Value - Mean) %>%
  ungroup() %>%
  mutate(Diff_color = sign(Diff_from_mean)) %>%
  mutate(Diff_color = as.factor(Diff_color))

glimpse(data_long_us)

# use the geom_segment() function to draw a straight line between points (x, y) and (xend, yend).
# In our case, this creates a plot that shows a bar for the difference between the observation and
the mean across all the years.

data_long_us %>%
  filter(Indicator %in% c("Emissions", "Temperature", "Disasters")) %>%
  ggplot(aes(x = Year, y = Value)) +
  geom_segment(aes(x = Year, y = Value, xend = Year,
                   yend = Mean, color = Diff_color),
              size = 3.25) +
  scale_color_manual(values = c("blue", "red")) +
  geom_hline(aes(yintercept = Mean), linetype = 1, color = "black") +
  facet_wrap(Indicator ~ ., scales = "free_y", ncol = 1 ) +
  theme_classic() +
  theme(axis.text.x = element_text(angle = 90),
        axis.title = element_blank(),
  legend.position = "none") +
  labs(title = "US Disasters, Emissions, and Temperatures (1990-2010)",
  subtitle = "Indicator Mean of 1990-2010 Represented by Solid Black Line")

# save the facet
save(data_long_us, file =here::here("plots", "data_long_us.rda"))
png(here::here("plots", "data_long_us.png"))
Top10t
dev.off()

# Create Scatter Plots in two variables: CO2 emissions and temperatures ranging from 1980 to 2014
as
# there are indicative values for those years for those variables

CO2_temp_US_facet <-
  data_long_1 %>%
  filter(Country == "United States", Year >= 1980, Year <= 2014,
         Indicator %in% c("Emissions", "Temperature")) %>%
  ggplot(aes(x = Year, y = Value)) +
  geom_point() +
  geom_smooth(method = "loess", se = FALSE) +
  scale_x_continuous(breaks = seq(1980, 2014, by = 5),
                     labels = seq(1980, 2014, by = 5)) +
  facet_wrap(Label ~ ., scales = "free_y", ncol = 1) +
  theme_classic() +
  theme(axis.text.x = element_text(size = 12, angle = 90, color = "black"),
        axis.title.y = element_text(size = 12, color = "black"),
        strip.text.x = element_text(size = 14),
        axis.title = element_blank(),
        plot.title = element_text(size = 16))
labs(title = "US Emissions and Temperatures(1980-2014)")

CO2_temp_US_facet

# save the plot
save(CO2_temp_US_facet, file =here::here("plots", "CO2_temp_US_facet.rda"))
png(here::here("plots", "CO2_temp_US_facet.png"))
CO2_temp_US_facet
dev.off()

# looking at relationship between CO2 emissions and other variables directly instead of separately
```

```
within their each variable
# By using pivot_wider to transform long data into a wide format

wide_US <-
  data_long_1 %>%
  filter(Country == "United States", Year >= 1980, Year <= 2014) %>%
  select(-Label) %>%
  pivot_wider(names_from = Indicator, values_from = Value)

# Save the wide_US data as rda file

save(wide_US, file = here::here("data", "wrangled","wrangled_US_data.rda"))
readr::write_csv(wide_US, path = here::here("data", "wrangled", "wrangled_US_data.csv"))


# Using the wrangled data now we can look at emissions & temperature

co2_temp_US <-
  wide_US %>%
  ggplot(aes(x = Emissions, y = Temperature)) +
  geom_point() +
  theme_classic() +
  theme(axis.text.x = element_text(size = 12, color = "black"),
        axis.text.y = element_text(size = 12, color = "black"),
        axis.title = element_text(size = 14),
        plot.title = element_text(size = 16)) +
  labs(title = "US Emissions and Temperature (1980 - 2014)",
       x = "Emissions (Metric Tonnes)",
       y = "Temperature (Fahrenheit)")

  co2_temp_US

  # Add a trend line to the plot using the geom_smooth of ggplot package

  co2_temp_US <-
    co2_temp_US +geom_smooth(method = "lm", se = FALSE)

  co2_temp_US

  # save the plot

  save(co2_temp_US, file =here::here("plots", "co2_temp_US.rda"))
  png(here::here("plots", "co2_temp_US.png"))
  co2_temp_US
  dev.off()

  # Data Analysis & Basic summary statistics
  load(here::here("data", "wrangled", "wrangled_US_data.rda"))

  wide_US %>%
    summarize(mean(Emissions), mean(Temperature), sd(Emissions), sd(Temperature))

# Create a summary plot using plot_layout() function of the patchwork R package along with
# grDevices png() function, we can also specify the size of the plot using the res argument


  load(here::here("plots", "CO2_world.rda"))
  load(here::here("plots", "Top10t.rda"))
  load(here::here("plots", "CO2_temp_US_facet.rda"))
  load(here::here("plots", "co2_temp_US.rda"))

  png(here::here("img", "mainplot.png"), units = "in", width = 12, height = 10, res = 300)
  (CO2_world | Top10t) / (CO2_temp_US_facet | co2_temp_US) +
    plot_layout(widths = c(1, 2),
                heights = unit(c(4, 10), c('cm', 'cm')))
```

```
dev.off()
```