Cleaning Data Project

Julian Lopez

Cleaning Data Project

Objectives for this project

To complete this project you'll need to do a few things within this file. As you go through the notebook, you will have further instruction on how to complete these objectives.

1. Go through this notebook, reading along.
2. Fill in empty or incomplete code chunks when prompted to do so.
3. Run each code chunk as you come across it by clicking the tiny green triangles at the right of each chunk. You should see the code chunks print out various output when you do this.
4. At the very top of this file, put your own name in the `author:` place. Currently it says `"DataTrail Team"`. Be sure to put your name in quotes.
5. In the `Conclusions` section, write up responses to each of these questions posed here.
6. When you are satisfied with what you've written and added to this document you'll need to save it. In the menu, go to `File` > `Save`. Now the `nb.html` output resulting file will have your new output saved to it.
7. Open up the resulting `countries_project.nb.html` file and click `View in Web Browser`. Does it look good to you? Did all the changes appear here as you expected.
8. Upload your `Rmd` and your `nb.html` to your assignment folder (this is something that will be dependent on what your instructors have told you – or if you are taking this on your own, just collect these projects in one spot, preferably a Google Drive)!
9. Pat yourself on the back for finishing this project!

The goal of this analysis

Across the countries of the world, how is literacy related to life expectancy?

Set up

Let's load these packages for use.

```
## you can add more, or change...these are suggestions
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.2     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.5.0     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.0
## v purrr     1.0.1
```

```
## -- Conflicts --------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(readr)
library(dplyr)
library(ggplot2)
library(tidyr)
```

Set up directories

Here we are going to make a directory or folder called `data`, if it doesn't already exist. No need for you to change this code.

```
if (!dir.exists("data")) {
  dir.create("data")
}
```

Get the data

We are going to use two different country datasets from Kaggle.

This UN data-derived dataset with various stats about different countries: https://www.kaggle.com/datasets/sudalairajkumar/undata-country-profiles The resulting CSV file is called `country_profile_variables.csv`.

This literacy dataset with statistics about literacy across countries https://www.kaggle.com/datasets/programmerrdai/literacy

The resulting CSV file is called `cross-country-literacy-rates`.

Both datasets have been downloaded and placed in the `raw` folder (also called a directory) within the `data` folder of the `03_Cleaning_the_Data` project folder. Use `readr::read_csv()` function to read each dataset into R. Call the first one `un_df` and the second `literacy_df`.

```
# Read in the dataset for un_df
getwd()
```

```
## [1] "/cloud/project/03_Cleaning_Data"
```

```
un_df <- read_csv("data/raw/country_profile_variables.csv")
```

```
## New names:
## Rows: 229 Columns: 50
## -- Column specification
## -------------------------------------------------------- Delimiter: "," chr
## (34): country, Region, Surface area (km2), GDP growth rate (annual %, co... dbl
## (16): Population in thousands (2017), Population density (per km2, 2017)...
## i Use `spec()` to retrieve the full column specification for this data. i
## Specify the column types or set `show_col_types = FALSE` to quiet this message.
## * `Mobile-cellular subscriptions (per 100 inhabitants)` -> `Mobile-cellular
##   subscriptions (per 100 inhabitants)...40`
## * `Mobile-cellular subscriptions (per 100 inhabitants)` -> `Mobile-cellular
##   subscriptions (per 100 inhabitants)...41`
```

```
# Read in the dataset for literacy_df
literacy_df <- read_csv("data/raw/cross-country-literacy-rates.csv")
```

```
## Rows: 1423 Columns: 4
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (2): Entity, Code
## dbl (2): Year, Literacy rates (World Bank, CIA World Factbook, and other sou...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

Use functions we have discussed previously to see what variables are in both `un_df` and `literacy_df`.

```
colnames(un_df)
```

```
##  [1] "country"
##  [2] "Region"
##  [3] "Surface area (km2)"
##  [4] "Population in thousands (2017)"
##  [5] "Population density (per km2, 2017)"
##  [6] "Sex ratio (m per 100 f, 2017)"
##  [7] "GDP: Gross domestic product (million current US$)"
##  [8] "GDP growth rate (annual %, const. 2005 prices)"
##  [9] "GDP per capita (current US$)"
## [10] "Economy: Agriculture (% of GVA)"
## [11] "Economy: Industry (% of GVA)"
## [12] "Economy: Services and other activity (% of GVA)"
## [13] "Employment: Agriculture (% of employed)"
## [14] "Employment: Industry (% of employed)"
## [15] "Employment: Services (% of employed)"
## [16] "Unemployment (% of labour force)"
## [17] "Labour force participation (female/male pop. %)"
## [18] "Agricultural production index (2004-2006=100)"
## [19] "Food production index (2004-2006=100)"
## [20] "International trade: Exports (million US$)"
## [21] "International trade: Imports (million US$)"
## [22] "International trade: Balance (million US$)"
## [23] "Balance of payments, current account (million US$)"
## [24] "Population growth rate (average annual %)"
## [25] "Urban population (% of total population)"
## [26] "Urban population growth rate (average annual %)"
## [27] "Fertility rate, total (live births per woman)"
## [28] "Life expectancy at birth (females/males, years)"
## [29] "Population age distribution (0-14 / 60+ years, %)"
## [30] "International migrant stock (000/% of total pop.)"
## [31] "Refugees and others of concern to UNHCR (in thousands)"
## [32] "Infant mortality rate (per 1000 live births"
## [33] "Health: Total expenditure (% of GDP)"
## [34] "Health: Physicians (per 1000 pop.)"
## [35] "Education: Government expenditure (% of GDP)"
## [36] "Education: Primary gross enrol. ratio (f/m per 100 pop.)"
## [37] "Education: Secondary gross enrol. ratio (f/m per 100 pop.)"
## [38] "Education: Tertiary gross enrol. ratio (f/m per 100 pop.)"
## [39] "Seats held by women in national parliaments %"
## [40] "Mobile-cellular subscriptions (per 100 inhabitants)...40"
## [41] "Mobile-cellular subscriptions (per 100 inhabitants)...41"
## [42] "Individuals using the Internet (per 100 inhabitants)"
## [43] "Threatened species (number)"
```

```
## [44] "Forested area (% of land area)"
## [45] "CO2 emission estimates (million tons/tons per capita)"
## [46] "Energy production, primary (Petajoules)"
## [47] "Energy supply per capita (Gigajoules)"
## [48] "Pop. using improved drinking water (urban/rural, %)"
## [49] "Pop. using improved sanitation facilities (urban/rural, %)"
## [50] "Net Official Development Assist. received (% of GNI)"
```

```
colnames(literacy_df)
```

```
## [1] "Entity"
## [2] "Code"
## [3] "Year"
## [4] "Literacy rates (World Bank, CIA World Factbook, and other sources)"
```

Cleaning the datasets

You'll notice both datasets have pretty messy looking column names that are annoying to use. Use the `janitor::clean_names()` function to clean these up. Don't forget to reassign the data objects!

```
original_un_df <- un_df
origina_literacy_df <- literacy_df
un_df <- original_un_df

install.packages("janitor")
```

```
## Installing package into '/cloud/project/renv/library/R-4.3/x86_64-pc-linux-gnu'
## (as 'lib' is unspecified)
```

```
library(janitor)
```

```
##
## Attaching package: 'janitor'
```

```
## The following objects are masked from 'package:stats':
##
##     chisq.test, fisher.test
```

```
un_df <- un_df %>%
  janitor::clean_names()
```

Clean up literacy_df The `literacy_df` data frame shows multiple rows for each country. For this analysis, let's only keep the latest year recorded for each country. You can use `arrange()` and `desc()` (or a minus sign) to first order the data by the latest year recorded for each country. Then use the `distinct()` function to keep only rows from the lastest year for each country by essentially selecting for the first mention of each country. Hint you will need to use an argument within `distinct()` to not lose the other columns. (Again don't forget to reassign your data objects to keep your work.)

```
literacy_df <- literacy_df %>%
  janitor:: clean_names() %>%
  arrange(desc(year)) %>%
  distinct(entity, .keep_all = TRUE)
```

Clean up un_df There's a number of variables in `un_df` that are numeric but are being treated as a character. Use the `glimpse()` function of the `dplyr` package to take a look at the variables to determine how often there are numeric values that are being treated as a character.

```r
dplyr::glimpse(un_df)
```

```
## Rows: 229
## Columns: 50
## $ country                                                    <chr> "Afghanis~
## $ region                                                     <chr> "Southern~
## $ surface_area_km2                                           <chr> "652864",~
## $ population_in_thousands_2017                               <dbl> 35530, 29~
## $ population_density_per_km2_2017                            <dbl> 54.4, 106~
## $ sex_ratio_m_per_100_f_2017                                 <dbl> 106.3, 10~
## $ gdp_gross_domestic_product_million_current_us              <dbl> 20270, 11~
## $ gdp_growth_rate_annual_percent_const_2005_prices           <chr> "-2.4", "~
## $ gdp_per_capita_current_us                                  <dbl> 623.2, 39~
## $ economy_agriculture_percent_of_gva                         <chr> "23.3", "~
## $ economy_industry_percent_of_gva                            <dbl> 23.3, 26.~
## $ economy_services_and_other_activity_percent_of_gva         <dbl> 53.3, 51.~
## $ employment_agriculture_percent_of_employed                 <chr> "61.6", "~
## $ employment_industry_percent_of_employed                    <chr> "10.0", "~
## $ employment_services_percent_of_employed                    <chr> "28.5", "~
## $ unemployment_percent_of_labour_force                       <chr> "8.6", "1~
## $ labour_force_participation_female_male_pop_percent         <chr> "19.3/83.~
## $ agricultural_production_index_2004_2006_100                <dbl> 125, 134,~
## $ food_production_index_2004_2006_100                        <dbl> 125, 134,~
## $ international_trade_exports_million_us                     <chr> "1458", "~
## $ international_trade_imports_million_us                     <chr> "3568", "~
## $ international_trade_balance_million_us                     <chr> "-2110", ~
## $ balance_of_payments_current_account_million_us             <chr> "-5121", ~
## $ population_growth_rate_average_annual_percent              <chr> "3.2", "-~
## $ urban_population_percent_of_total_population               <dbl> 26.7, 57.~
## $ urban_population_growth_rate_average_annual_percent        <chr> "4.0", "2~
## $ fertility_rate_total_live_births_per_woman                 <chr> "5.3", "1~
## $ life_expectancy_at_birth_females_males_years               <chr> "63.5/61.~
## $ population_age_distribution_0_14_60_years_percent          <chr> "43.2/4.1~
## $ international_migrant_stock_000_percent_of_total_pop       <chr> "382.4/1.~
## $ refugees_and_others_of_concern_to_unhcr_in_thousands       <chr> "1513.1",~
## $ infant_mortality_rate_per_1000_live_births                 <chr> "68.6", "~
## $ health_total_expenditure_percent_of_gdp                    <dbl> 8.2, 5.9,~
## $ health_physicians_per_1000_pop                             <chr> "0.3", "1~
## $ education_government_expenditure_percent_of_gdp            <chr> "3.3", "3~
## $ education_primary_gross_enrol_ratio_f_m_per_100_pop        <chr> "91.1/131~
## $ education_secondary_gross_enrol_ratio_f_m_per_100_pop      <chr> "39.7/70.~
## $ education_tertiary_gross_enrol_ratio_f_m_per_100_pop       <chr> "3.7/13.3~
## $ seats_held_by_women_in_national_parliaments_percent        <dbl> 27.7, 22.~
## $ mobile_cellular_subscriptions_per_100_inhabitants_40       <chr> "61.6", "~
## $ mobile_cellular_subscriptions_per_100_inhabitants_41       <chr> "8.3", "6~
## $ individuals_using_the_internet_per_100_inhabitants         <dbl> 42, 130, ~
## $ threatened_species_number                                  <chr> "2.1", "2~
## $ forested_area_percent_of_land_area                         <chr> "9.8/0.3"~
## $ co2_emission_estimates_million_tons_tons_per_capita        <dbl> 63, 84, 5~
## $ energy_production_primary_petajoules                       <dbl> 5, 36, 55~
## $ energy_supply_per_capita_gigajoules                        <chr> "78.2/47.~
## $ pop_using_improved_drinking_water_urban_rural_percent      <chr> "45.1/27.~
## $ pop_using_improved_sanitation_facilities_urban_rural_percent <chr> "21.43", ~
## $ net_official_development_assist_received_percent_of_gni    <dbl> -99, -99,~
```

Let's take a look at `life_expectancy_at_birth_females_males_years` as an example. Use the `head()` function and print out the first 10 rows to see.

```
head(un_df$life_expectancy_at_birth_females_males_years, 10)
```

```
##  [1] "63.5/61.0" "79.9/75.6" "76.5/74.1" "77.8/71.1" "-99"       "63.0/57.4"
##  [7] ".../..."   "78.2/73.3" "79.8/72.2" "77.0/70.6"
```

Let's also take a closer look at `co2_emission_estimates_million_tons_tons_per_capita` as an example.

```
head(un_df$co2_emission_estimates_million_tons_tons_per_capita, 10)
```

```
##  [1]   63   84 5900  -99    1 3902    0  -99 3167   48
```

Two things we can notice here. One is that really `life_expectancy_at_birth_females_males_years` looks like it is two variables, one for female and one for male. We'll want to split this into two columns.

Another is that they are using two different items to note missing data `-99` and `...` For R to deal with missing data appropriately we will want to change these to be `NA`. This is also true for the `co2_emission_estimates_million_tons_tons_per_capita` variable which is being considered by R as a numeric value (based on how it is labeled as `dbl`).

Let's start with some of the missing data. To replace these `-99`(the numeric version) and `"-99"` (the character version) and `.../...` , we can use the function: `naniar::replace_with_na()`. We'll want to do this for every variable, not just the one we looked at above. When we replace each of these they will have a single `NA` value even though we have two values for the observations where we have numeric values.

```
## ?replace_with_na_all
un_df <- un_df %>%
 naniar::replace_with_na_all(condition = ~.x ==-99)
un_df <- un_df %>%
 naniar::replace_with_na_all(condition = ~.x =="-99")
un_df <- un_df %>%
 naniar::replace_with_na_all(condition = ~.x ==".../...")
```

Let's check to see if this did what we expected.

```
-99 %in% un_df
```

```
## [1] FALSE
```

```
"-99" %in% un_df
```

```
## [1] FALSE
```

```
".../..." %in% un_df
```

```
## [1] FALSE
```

Great, hopefully it looks like we have replaced the the `-99` ,`"-99"`, and `.../...` values. It also looks like we don't have any values like this `.../49.6` or this `56.2/...`. If we did we would need to do something a little fancier (possibly with the `stringr` package) but for now we don't need to worry about that. It is always good to check your data though!

Let's split the column `life_expectancy_at_birth_females_males_years` into two columns so we can more appropriately deal with these data. To do this we can use a handy function called `tidyr::separate()`. Call the new resulting columns `"life_expectancy_at_birth_females"` and `"life_expectancy_at_birth_males"` by specifying this with the `into` argument. We can specify that we want to separate by the `/` symbol by using the `sep` argument. For observations where we have an `NA` value, both new columns will receive an `NA`.

```
## ?tidyr::separate()
un_df <- un_df %>%
  tidyr::separate(life_expectancy_at_birth_females_males_years,  c("life_expectancy_at_birth_females",
```

After splitting these data, you'll notice they are still characters, so you will need to coerce them to numeric variables with a `mutate()` step.

In fact, there are a lot more columns like this. But for now just make these life_expectancy variables into numeric and don't worry about the others.

```
un_df$life_expectancy_at_birth_females <-
as.numeric(un_df$life_expectancy_at_birth_females)

un_df$life_expectancy_at_birth_males <-
as.numeric(un_df$life_expectancy_at_birth_males)
```

The `un_df` has a `region` column.

Let's run `summary()` on it.

```
summary(un_df$region)
```

```
##    Length     Class      Mode
##       229 character character
```

The regions listed are repeated but because they are characters, summary doesn't give us useful information about the categories. This `region` columns would be best treated as factors. Turn this `region` column into a factor.

```
un_df$region <-
as.factor(un_df$region)
```

Re-run `summary` on your region columns to confirm that the data make more sense now.

```
summary(un_df$region)
```

```
##         Caribbean    CentralAmerica       CentralAsia     EasternAfrica
##                25                 8                 5                19
##       EasternAsia     EasternEurope         Melanesia        Micronesia
##                 7                10                 5                 7
##      MiddleAfrica    NorthernAfrica   NorthernAmerica    NorthernEurope
##                 9                 7                 5                13
##           Oceania         Polynesia South-easternAsia      SouthAmerica
##                 2                 9                11                14
##    SouthernAfrica      SouthernAsia    SouthernEurope     WesternAfrica
##                 5                 9                16                16
##       WesternAsia     WesternEurope
##                18                 9
```

Joining the data

Now we have two generally clean datasets that both have information about countries. Use a `dplyr::join` function to join all the rows for countries that are in both `un_df` and `literacy_df` but exclude rows from countries that aren't in both. Look up `?dplyr::join` for more help. You will need to use the `by` argument.

Call this new data frame `countries_df`.

```
## ?dplyr::join
countries_df <- dplyr::inner_join(un_df,literacy_df, by = c("country" = "entity"))
```

We will be returning to this dataset in a future project. Save `countries_df` to an RDS file here. Save this file to the `data` folder and name it `countries_df.rds`.

```
## getwd()
saveRDS(countries_df, file = "/cloud/project/03_Cleaning_Data/data/countries_df.rds")
```

Reshape data in preparation for plotting

We want to see how literacy is related to life expectancy. But have literacy split up by gender. Let's reshape our data into something that will be easier to plot.

First, select only the following columns: `country`, `year`, `life_expectancy_at_birth_females`, `life_expectancy_at_birth_males`, and `literacy_rates_world_bank_cia_world_factbook_and_other_sources`. Name this new `selected` data frame `plotting_df`.

In this same step, let's also shorten the column name `literacy_rates_world_bank_cia_world_factbook_and_other_source` to just `literacy_rates`.

```
plotting_df <- countries_df %>%
  select(country,year,life_expectancy_at_birth_females,life_expectancy_at_birth_males,literacy_rates_wo
  rename(literacy_rates = literacy_rates_world_bank_cia_world_factbook_and_other_sources)
```

Now, let's reshape this data so that it is longer. We will want to modify the `life_expectancy_at_birth_females` and `life_expectancy_at_birth_males` variables so that we instead have one column for `life_expectancy` values and one for `gender` which specifies what column the values originally came from. We want to keep `country`, `year`, `literacy_rates`. Use the `pivot_longer` function.

```
plotting_df <- plotting_df%>%
  pivot_longer(cols = starts_with("life_expectancy_at_birth_"), names_to = "gender", values_to = "life_
```

You'll notice your new column `gender` has the whole old variable name. We don't really need all that. Use string manipulations so that you only have either `males` or `females`.

```
plotting_df <- plotting_df %>%
  mutate(gender = str_remove(gender, "life_expectancy_at_birth_"))
```

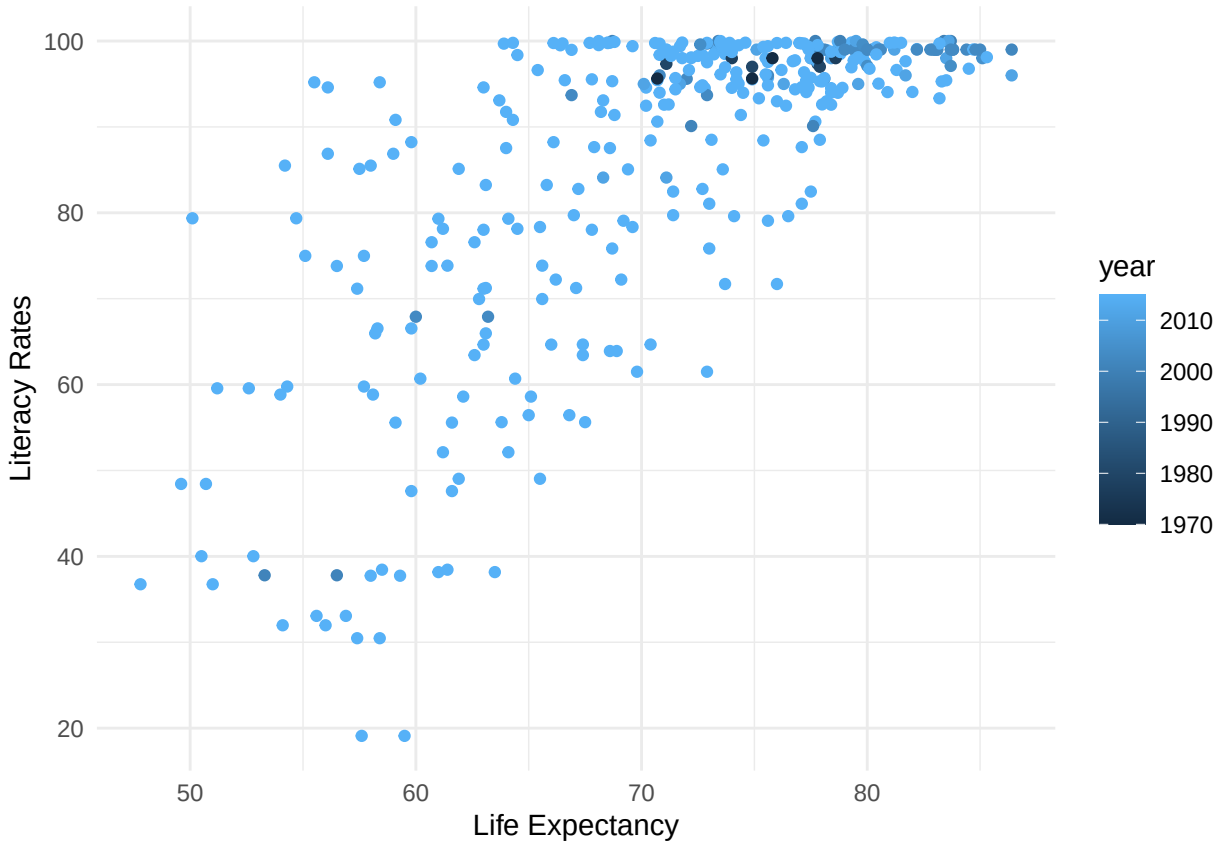
Plot the data!

Let's make a scatter plot of life expectancy and literacy rates! Color the plot by year.

```
lit_plot <- plotting_df %>%
  ggplot(aes(life_expectancy, literacy_rates, color = year)) +
  geom_point() +
  theme_minimal() +
  ylab("Literacy Rates") +
  xlab("Life Expectancy")

lit_plot
```

```
## Warning: Removed 24 rows containing missing values or values outside the scale range
## (`geom_point()`).
```



This looks like literacy and life expectancy are positively related to each other. In other words, in countries with higher literacy rates, there's also a longer life expectancy.

In this plot we've color coded by the year that the literacy data was recorded. This way we are aware of how differences in years might influence this relationship.

But! Really, we've lumped together `female` and `male` data points with each country which isn't really the right way to treat this data. Let's use `facet_wrap()` to make this one plot into two separate plots.

We can also add a trend line to get a better sense of how these two variables relate to one another using `geom_smooth()`.

```
lit_plot + facet_wrap(~gender) + geom_smooth()
```

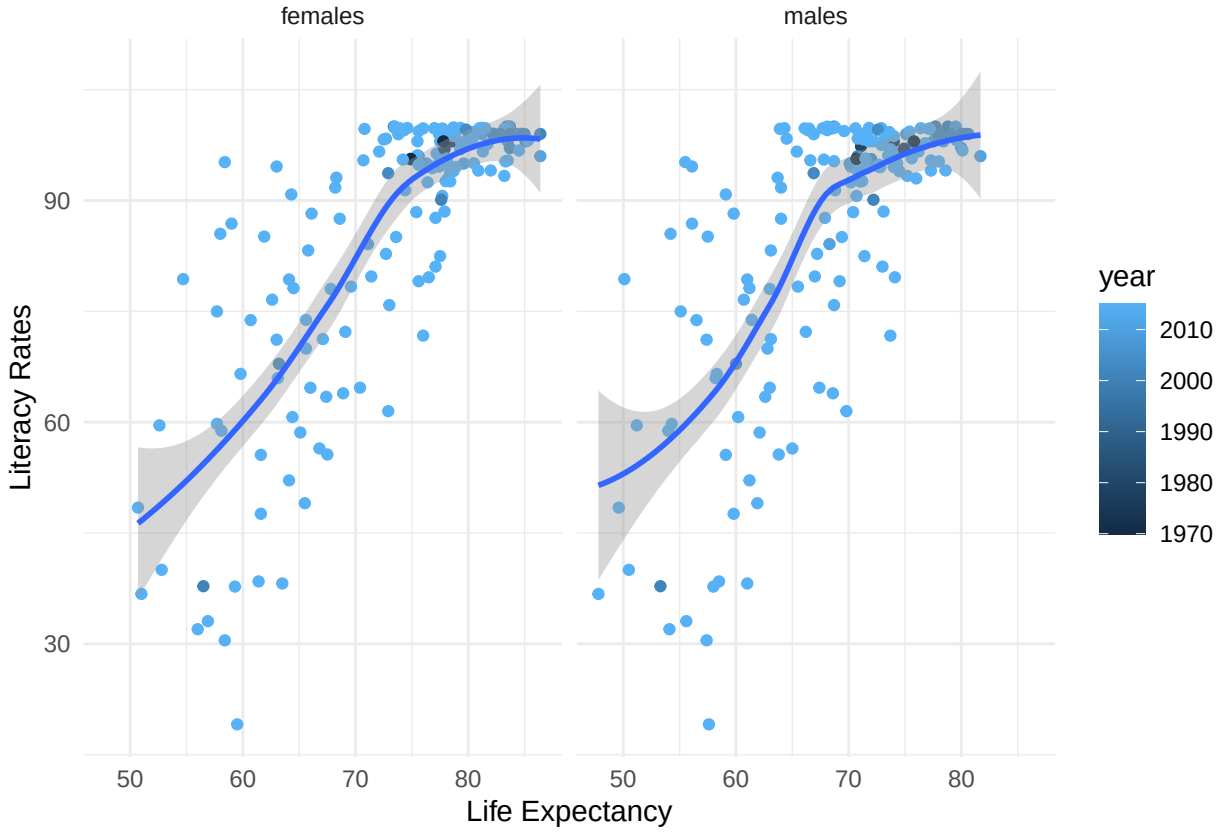
```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```

```
## Warning: Removed 24 rows containing non-finite outside the scale range
## (`stat_smooth()`).
```

```
## Warning: The following aesthetics were dropped during statistical transformation:
## colour.
## i This can happen when ggplot fails to infer the correct grouping structure in
##   the data.
## i Did you forget to specify a `group` aesthetic or to convert a numerical
##   variable into a factor?
## The following aesthetics were dropped during statistical transformation:
## colour.
## i This can happen when ggplot fails to infer the correct grouping structure in
##   the data.
## i Did you forget to specify a `group` aesthetic or to convert a numerical
```

```
##   variable into a factor?
```

```
## Warning: Removed 24 rows containing missing values or values outside the scale range
## (`geom_point()`).
```



It looks like this relationship is stable across `male` and `female`. It also looks like a stronger relationship for the life expectancy and literacy values in the middle to lower range. Let's get a stat on the relationship.

Get the stats

We can also use a correlation to ask this question.

```
cor.test(countries_df$life_expectancy_at_birth_females,
         countries_df$literacy_rates_world_bank_cia_world_factbook_and_other_sources)
```

```
##
##  Pearson's product-moment correlation
##
## data:  countries_df$life_expectancy_at_birth_females and countries_df$literacy_rates_world_bank_cia_
## t = 17.102, df = 175, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.7284666 0.8404257
## sample estimates:
##       cor
## 0.7909772
```

```
cor.test(countries_df$life_expectancy_at_birth_males,
         countries_df$literacy_rates_world_bank_cia_world_factbook_and_other_sources)
```

```
##
##  Pearson's product-moment correlation
##
## data:  countries_df$life_expectancy_at_birth_males and countries_df$literacy_rates_world_bank_cia_wo
## t = 13.645, df = 175, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.6380485 0.7825999
## sample estimates:
##       cor
## 0.7179783
```

With using either `male` or `female` this relationship looks pretty strong!

Conclusion

Write up your thoughts about this data science project here and answer the following questions:

- What did we find out about our questions? Those countries with high literacy also have a high life expectancy
- How did we explore our questions? Using data reshaping techniques and tidying the data we were able to plot the information on scatterplot leading to our conclusion high literacy leads to higher life expectancy.
- What did our explorations show us? Higher literacy = higher life expectancy
- What follow up data science questions arise for you regarding this dataset now that we've explored it some? I would like to see this data only for the most developed countries. My goal would be to find out whether high literacy and higher life expectancy still correlate or is there a point at which the two diverge.

Print out session info

Session info is a good thing to print out at the end of your notebooks so that you (and other folks) referencing your notebooks know what software versions and libraries you used to run the notebook.

```
sessionInfo()
```

```
## R version 4.3.3 (2024-02-29)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 20.04.6 LTS
##
## Matrix products: default
## BLAS:   /usr/lib/x86_64-linux-gnu/atlas/libblas.so.3.10.3
## LAPACK: /usr/lib/x86_64-linux-gnu/atlas/liblapack.so.3.10.3;  LAPACK version 3.9.0
##
## locale:
##  [1] LC_CTYPE=C.UTF-8       LC_NUMERIC=C           LC_TIME=C.UTF-8
##  [4] LC_COLLATE=C.UTF-8     LC_MONETARY=C.UTF-8    LC_MESSAGES=C.UTF-8
##  [7] LC_PAPER=C.UTF-8       LC_NAME=C              LC_ADDRESS=C
## [10] LC_TELEPHONE=C         LC_MEASUREMENT=C.UTF-8 LC_IDENTIFICATION=C
##
## time zone: UTC
## tzcode source: system (glibc)
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
```

```
## other attached packages:
##  [1] janitor_2.2.0   lubridate_1.9.3 forcats_1.0.0   stringr_1.5.0
##  [5] dplyr_1.1.2     purrr_1.0.1     readr_2.1.4     tidyr_1.3.0
##  [9] tibble_3.2.1    ggplot2_3.5.0   tidyverse_2.0.0
##
## loaded via a namespace (and not attached):
##  [1] utf8_1.2.4        generics_0.1.3    lattice_0.22-5    stringi_1.8.3
##  [5] hms_1.1.3         digest_0.6.35     magrittr_2.0.3    evaluate_0.23
##  [9] grid_4.3.3        timechange_0.3.0  fastmap_1.1.1     Matrix_1.6-5
## [13] mgcv_1.9-1        fansi_1.0.6       scales_1.3.0      cli_3.6.2
## [17] rlang_1.1.3       crayon_1.5.2      splines_4.3.3     bit64_4.0.5
## [21] munsell_0.5.0     naniar_1.1.0      withr_3.0.0       yaml_2.3.8
## [25] tools_4.3.3       parallel_4.3.3    tzdb_0.4.0        colorspace_2.1-0
## [29] vctrs_0.6.5       R6_2.5.1          lifecycle_1.0.4   snakecase_0.11.0
## [33] bit_4.0.5         vroom_1.6.3       pkgconfig_2.0.3   pillar_1.9.0
## [37] gtable_0.3.4      glue_1.7.0        visdat_0.6.0      highr_0.10
## [41] xfun_0.42         tidyselect_1.2.0  rstudioapi_0.15.0 knitr_1.45
## [45] farver_2.1.1      nlme_3.1-164      htmltools_0.5.7   labeling_0.4.3
## [49] rmarkdown_2.23    compiler_4.3.3
```