Data Visualization (1)

Basics

Haohan Chen

POLI3148 Data Science in PPA (The University of Hong Kong)

Last update: October 12, 2023

Motivation

Data Visualization

Life is short. Use graphs!

What does data visualization do? Let's start with some examples.

- Hans Roling's Gapminder: https://www.gapminder.org/tools/
- Our World in Data: https://ourworldindata.org/
- V-Dem data visualization tools: https://v-dem.net/graphing/graphing-tools/

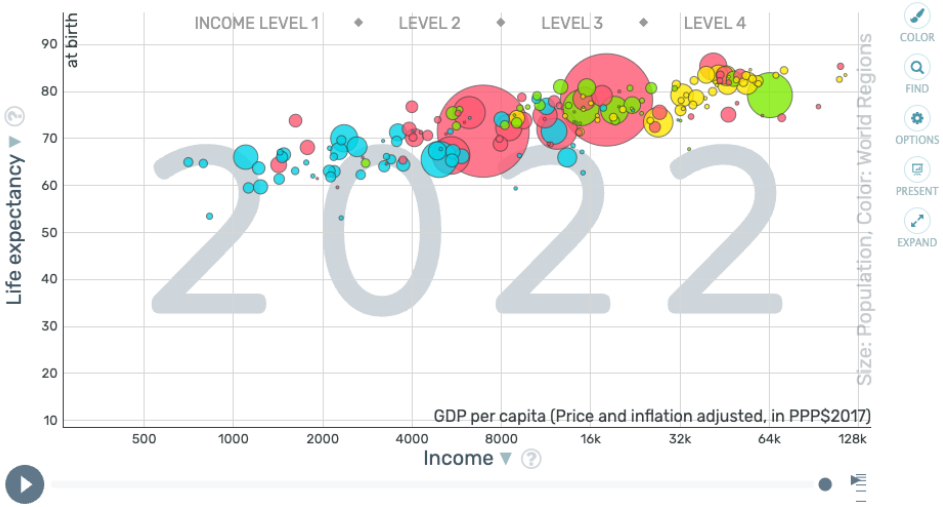
Our Task: Extend the "Health and Wealth" Thesis

https://www.gapminder.org/fw/world-health-chart/

> *Hans Rosling shows that income and health go hand in hand. People live longer in richer countries. Or the other way around. Countries are richer where people live longer. There are no high income countries with a short life expectancy, and no low income countries with a long life expectancy. Still, there's a huge difference in life expectancy between countries on the same income level, depending on how the money is distributed and how it is used.*

Our Task: Extend the "Health and Wealth" Thesis

https://www.gapminder.org/fw/world-health-chart/



Data Preparation

Load the Data

```
library(tidyverse)

d <- bind_rows(
  read_csv("_DataPublic_/vdem/1789_1827/vdem_1789_1827_external.csv"),
  read_csv("_DataPublic_/vdem/1867_1905/vdem_1867_1905_external.csv"),
  read_csv("_DataPublic_/vdem/1906_1944/vdem_1906_1944_external.csv"),
  read_csv("_DataPublic_/vdem/1945_1983/vdem_1945_1983_external.csv"),
  read_csv("_DataPublic_/vdem/1984_2022/vdem_1984_2022_external.csv")
) |>
  select(country_text_id, year, e_regiongeo, e_pelifeex, e_gdppc, e_mipopula, e_wb_pop) |>
  # Note: Look up the codebook for these variables
  rename("region" = "e_regiongeo", "life_expectancy" = "e_pelifeex", "gdppc" = "e_gdppc",
         "population_ClioInfra" = "e_mipopula", "population_WorldBank" = "e_wb_pop") |>
  filter(year >= 1800)

d |> print(n = 3)
```

```
## # A tibble: 23,593 x 7
##   country_text_id  year region life_expectancy gdppc population_ClioInfra
##   <chr>           <dbl>  <dbl>           <dbl> <dbl>                <dbl>
## 1 MEX              1800     17            26.9  1.35                5100
## 2 MEX              1801     17            26.9  1.34                5174.
## 3 MEX              1802     17            26.9  1.32                5249.
## # i 23,590 more rows
## # i 1 more variable: population_WorldBank <dbl>
```

Multiple Population Data Sources

We have two population data sources, with different coverage of years.

9.6.2 Population total (E) (e_mipopula)

Question: What is the total population (in thousands)?

Source(s): Clio Infra (clio-infra.eu), drawing on Goldewijk, Beusen, Janssen (2010), History Database of Global Environment (www.pbl.nl/hyde).

Notes: Missing data within a time–series is interpolated using linear interpolation.

Data release: 2-13.

Citation: Clio Infra (clio-infra.eu).

Years: 1800-2000

9.6.9 Population (E) (e_wb_pop)

Question: What is the total population?

Clarification: Total population is based on the de facto definition of population, which counts all residents regardless of legal status or citizenship. The values shown are midyear estimates.

Scale: Continuous

Source(s): (1) United Nations Population Division. World Population Prospects: 2019 Revision. (2) Census reports and other statistical publications from national statistical offices, (3) Eurostat: Demographic Statistics, (4) United Nations Statistical Division. Population and Vital Statistics Reprot (various years), (5) U.S. Census Bureau: International Database, and (6) Secretariat of the Pacific Community: Statistics and Demography Programme.

Data release: 9-13.

Citation: World Bank (2022)

Years: 1960-2021

Multiple Population Data Sources

Consistency? Check years when the two datasets both cover.

```
d_pop_overlap <- d |> select(country_text_id, year, starts_with("population_")) |>
  drop_na()
print(d_pop_overlap, n = 3)
```

```
## # A tibble: 5,818 x 4
##   country_text_id  year population_ClioInfra population_WorldBank
##   <chr>           <dbl>                <dbl>                <dbl>
## 1 MEX              1960               38578.             37771861
## 2 MEX              1961               39998.             38966049
## 3 MEX              1962               41418.             40195318
## # i 5,815 more rows
```

```
unique(d_pop_overlap$year)
```

```
##  [1] 1960 1961 1962 1963 1964 1965 1966 1967 1968 1969 1970 1971 1972 1973 1974
## [16] 1975 1976 1977 1978 1979 1980 1981 1982 1983 1984 1985 1986 1987 1988 1989
## [31] 1990 1991 1992 1993 1994 1995 1996 1997 1998 1999 2000
```

```
cor(d_pop_overlap$population_ClioInfra, d_pop_overlap$population_WorldBank)
```

```
## [1] 0.9997128
```

Set a Rule to Merge the Two Population Columns

- Different units: Divide `population_WorldBank` by 1000 (so that the unit of population is "in thousands"
- Different coverage but almost perfect correlation
  - For years that only one dataset has coverage, take the value from the dataset that has available data points.
  - For years that both datasets have coverage, take their `mean`.

That means, effectively, we are taking the `mean` and allow `na.rm = TRUE`. Think about it.

Merge the Two Population Columns

Below is an implementation of the rule we have just set. The output of this step is a new variable called `population` which aggregate data from both sources.

```
# STEP 1: "Harmonize" the units
d <- d |> mutate(population_WorldBank = population_WorldBank / 1000)

# STEP 2 Method 1: Slower but use only tidyverse functionality
d <- d |> rowwise() |>
  mutate(population = mean(c_across(c("population_ClioInfra", "population_WorldBank")), na.rm = TRUE))

# STEP 2 Method 2: Faster but use a non-tidyverse function rowMeans()
# and create a temporary vector tmp_population, which I remove after use with rm()
tmp_population <- d |> select(population_ClioInfra, population_WorldBank) |> rowMeans(na.rm = TRUE)
d <- d |> mutate(population = !!(tmp_population))
rm(tmp_population)

# Remove the columns we no longer need
d <- d |> select(-population_ClioInfra, -population_WorldBank)

d |> print(n = 3)
```

```
## # A tibble: 23,593 x 6
## # Rowwise:
##   country_text_id  year region life_expectancy gdppc population
##   <chr>           <dbl>  <dbl>           <dbl> <dbl>      <dbl>
## 1 MEX              1800     17            26.9  1.35       5100
## 2 MEX              1801     17            26.9  1.34       5174.
## 3 MEX              1802     17            26.9  1.32       5249.
## # i 23,590 more rows
```

Sanity Check

```
summary(d) # Summary statistics of each variable
```

```
##  country_text_id         year          region      life_expectancy
##  Length:23593       Min.   :1800   Min.   : 1.00   Min.   : 1.50
##  Class :character   1st Qu.:1912   1st Qu.: 5.00   1st Qu.:35.50
##  Mode  :character   Median :1952   Median : 9.00   Median :50.30
##                     Mean   :1944   Mean   : 9.46   Mean   :51.37
##                     3rd Qu.:1989   3rd Qu.:14.00   3rd Qu.:67.10
##                     Max.   :2022   Max.   :19.00   Max.   :85.30
##                                                    NA's   :1232
##      gdppc            population
##  Min.   :  0.286   Min.   :     17.9
##  1st Qu.:  1.599   1st Qu.:   1246.3
##  Median :  2.774   Median :   4234.3
##  Mean   :  7.194   Mean   :  23083.3
##  3rd Qu.:  7.606   3rd Qu.:  11914.2
##  Max.   :156.628   Max.   :1412360.0
##  NA's   :4571      NA's   :2981
```

Always watch out for when you see `NA`, especially when the number is non-trivial!

Check Data Availability

```
check_data_available <- d |>
  mutate(Available = (!is.na(life_expectancy) & !is.na(gdppc) & !is.na(population)))

# Check number of missing values by country-year
table(check_data_available$Available, useNA = "always")
```

```
##
## FALSE  TRUE  <NA>
##  6003 17590     0
```

```
check_data_available |> print(n = 3)
```

```
## # A tibble: 23,593 x 7
## # Rowwise:
##   country_text_id  year region life_expectancy gdppc population Available
##   <chr>           <dbl>  <dbl>           <dbl> <dbl>      <dbl> <lgl>
## 1 MEX              1800     17            26.9  1.35      5100  TRUE
## 2 MEX              1801     17            26.9  1.34      5174. TRUE
## 3 MEX              1802     17            26.9  1.32      5249. TRUE
## # i 23,590 more rows
```

Check Data Availability (con'd)

```
check_data_available_wide <- check_data_available |>
  select(country_text_id, year, Available) |>
  pivot_wider(names_from = "country_text_id", values_from = "Available",
              names_prefix = "c_") |>
  arrange(year)

check_data_available_wide |> print(n = 3)
```

```
## # A tibble: 184 x 203
##    year c_MEX c_SWE c_CHE c_JPN c_MMR c_RUS c_EGY c_YEM c_COL c_POL c_BRA c_USA
##   <dbl> <lgl> <lgl> <lgl> <lgl> <lgl> <lgl> <lgl> <lgl> <lgl> <lgl> <lgl> <lgl>
## 1  1800 TRUE  TRUE  TRUE  TRUE  FALSE FALSE TRUE  FALSE TRUE  NA    TRUE  FALSE
## 2  1801 TRUE  TRUE  TRUE  TRUE  FALSE FALSE TRUE  FALSE TRUE  NA    TRUE  FALSE
## 3  1802 TRUE  TRUE  TRUE  TRUE  FALSE FALSE TRUE  FALSE TRUE  NA    TRUE  FALSE
## # i 181 more rows
## # i 190 more variables: c_PRT <lgl>, c_BOL <lgl>, c_HTI <lgl>, c_PER <lgl>,
## #   c_VDR <lgl>, c_AFG <lgl>, c_ARG <lgl>, c_ETH <lgl>, c_IND <lgl>,
## #   c_KOR <lgl>, c_THA <lgl>, c_VEN <lgl>, c_IDN <lgl>, c_NPL <lgl>,
## #   c_AUS <lgl>, c_CHL <lgl>, c_FRA <lgl>, c_DEU <lgl>, c_GTM <lgl>,
## #   c_IRN <lgl>, c_LBR <lgl>, c_MAR <lgl>, c_NLD <lgl>, c_ESP <lgl>,
## #   c_TUN <lgl>, c_TUR <lgl>, c_GBR <lgl>, c_URY <lgl>, c_CHN <lgl>, ...
```

Check Data Availability (con'd)

```
# Check, for each year, the availability of each column
check_data_available_by_column <- d |>
  group_by(year) |>
  summarise(
    life_expectancy = sum(is.na(life_expectancy)),
    gdppc = sum(is.na(gdppc)),
    population = sum(is.na(population))
  )
  # summarise_at(vars(life_expectancy, gdppc, population), ~sum(!is.na(.)))
  # above is an alternative way to write the summarise() step

check_data_available_by_column |> print(n = 3)
```

```
## # A tibble: 184 x 4
##    year life_expectancy gdppc population
##   <dbl>           <int> <int>      <int>
## 1  1800              16    31         25
## 2  1801              16    31         25
## 3  1802              17    32         26
## # i 181 more rows
```

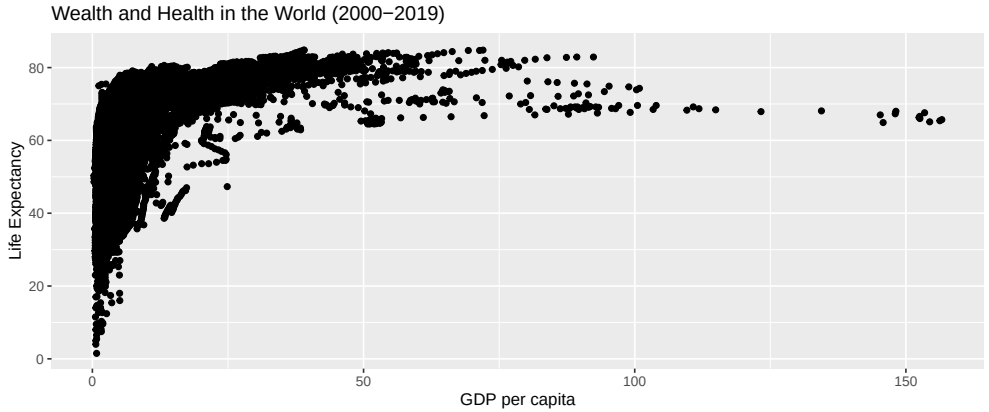
OK. All look good! We are ready to make our nice figures.

Data Viz Basics

Simplest Possible Visualization

Plot a scatter plot with ALL the data points.

```
d |> ggplot(aes(x = gdppc, y = life_expectancy)) +
  geom_point() +
  labs(x = "GDP per capita", y = "Life Expectancy",
       title = "Wealth and Health in the World (2000-2019)",
       caption = "By Haohan Chen. Data source: V-Dem v.13")
```



Store Your First Data Visualization

To make your nice data visualization stay. You can either (temporarily) save it in your R Environment, or save it as a file in your folder.

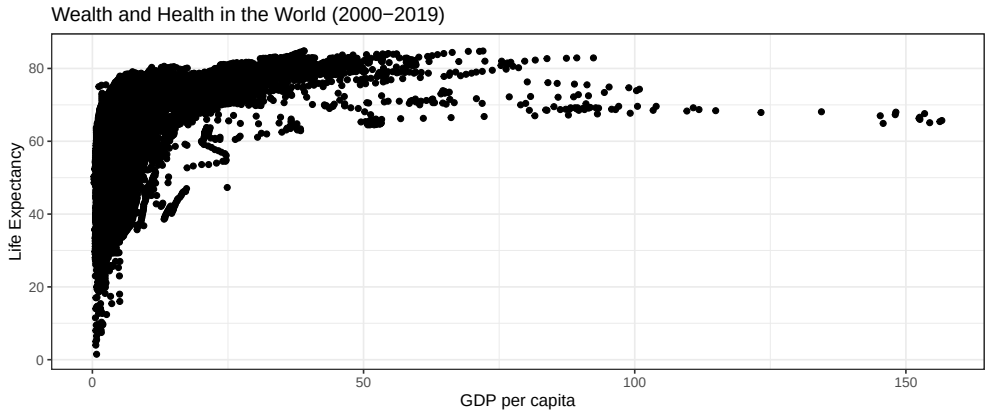
```
# Store in R environment (temporary)
p_all <- d |> ggplot(aes(x = gdppc, y = life_expectancy)) +
  geom_point() +
  labs(x = "GDP per capita", y = "Life Expectancy",
       title = "Wealth and Health in the World (2000-2019)",
       caption = "By Haohan Chen. Data source: V-Dem v.13")

# Save plot as a .rds file in your folder
saveRDS(p_all, "Lec_06/3_data_visualization_1/figures/welath_and_health_all.rds")

# Save plot as a PDF file in your folder
ggsave(filename = "Lec_06/3_data_visualization_1/figures/welath_and_health_all.pdf",
       plot = p_all, width = 9, height = 4)
```

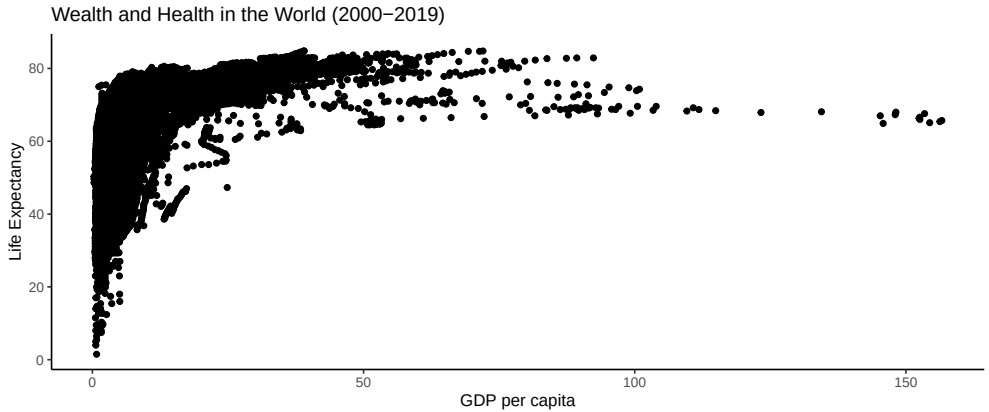
Set Themes: theme_bw

```
p_all + theme_bw()
```



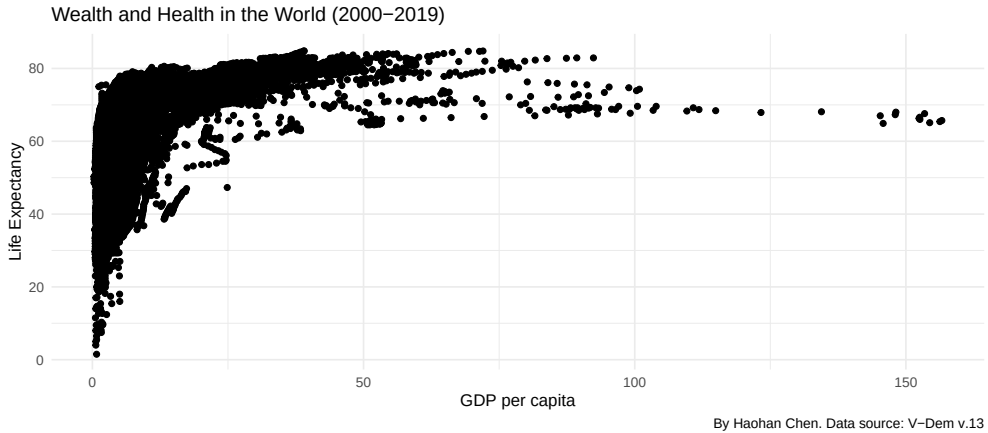
Set Themes: theme_classic

```
p_all + theme_classic()
```



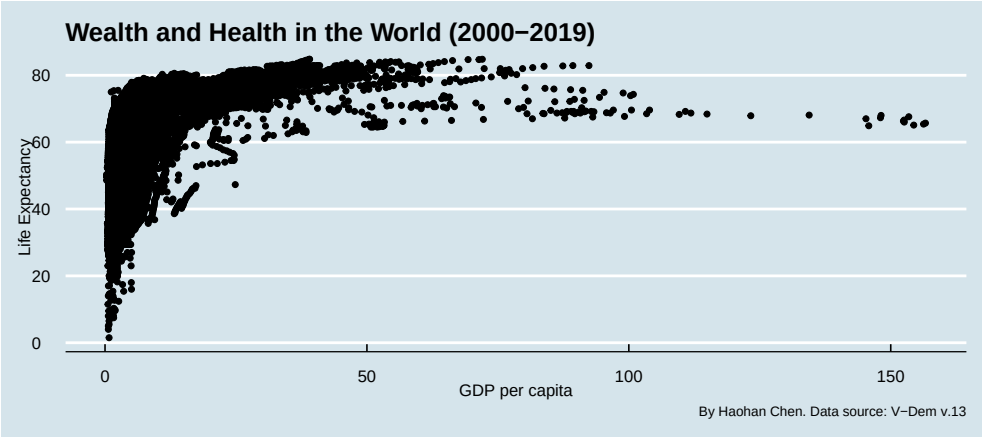
Set Themes: theme_minimal

```
p_all + theme_minimal()
```



Other Fancy Themes: The Economist

```
# install.packages("ggthemes") # install the package upon your first use.
# Take a look at the package's website: https://yutannihilation.github.io/allYourFigureAreBelongToUs/ggthemes/
library(ggthemes)
p_all + theme_economist()
```



Other Fancy Themes: The WSJ

```
p_all + theme_wsj(base_size = 6)
```