Ivester Institute for Business Analytics and Insights: Intro to R for BUSN Analytics

Katherine A Ireland

2025-09-10

0) Background: Why use R for BUSN Analytics?

- Works well for the full data science pipeline: getting data, data wrangling, to analysis and visualization
- Consistent grammar → fast, repeatable visuals for reports/decks
- Clear mapping from business questions → visual encodings
- Works across disciplines + data

Quick references

- R for Data Science (R4DS), Chapter on Data Visualization (https://r4ds.had.co.nz/data-visualisation.html)
- dplyr documentation (https://cran.r-project.org/web/packages/dplyr/vignettes/dplyr.html)
- ggplot2 documentation (https://ggplot2.tidyverse.org/)
- Posit (RStudio) ggplot2 Cheatsheet (https://posit.co/resources/cheatsheets/)

the Gapminder dataset

- The Gapminder dataset (https://www.gapminder.org/) comes from the Gapminder Foundation, a non-profit started by Hans Rosling.

- It was created to make global development statistics accessible, visual, and easy to understand.

- The data combines information from sources like the United Nations, World Bank, and other international agencies.

- It’s widely used in teaching because it’s clean, tidy, and covers long-term trends in health and wealth across the world.

1) dplyr verbs: the powerhouse behind data processing and analytics

Next, we will go through several useful dplyr verbs one-by-one on our gapminder dataset.

Dplyr verbs operate on different aspects of your dataset: on rows (filter, slice, arrange), on columns (select, rename, mutate, relocate), on groups of rows (summarise)

filter() → Keep only certain rows.

What it does:
Think of `filter()` as a row *sieve*. You provide a condition, and only rows that meet the condition stay in the dataset.

Example: Get a specific row value.

```
# Example: filtering by two row values
gap %>%
  filter(continent == "Asia", year == 2007)
```

```
## # A tibble: 33 × 6
##    country          continent  year lifeExp        pop gdpPercap
##    <fct>            <fct>     <int>   <dbl>      <int>     <dbl>
##  1 Afghanistan      Asia       2007    43.8   31889923      975.
##  2 Bahrain          Asia       2007    75.6     708573    29796.
##  3 Bangladesh       Asia       2007    64.1  150448339     1391.
##  4 Cambodia         Asia       2007    59.7   14131858     1714.
##  5 China            Asia       2007    73.0 1318683096     4959.
##  6 Hong Kong, China Asia       2007    82.2    6980412    39725.
##  7 India            Asia       2007    64.7 1110396331     2452.
##  8 Indonesia        Asia       2007    70.6  223547000     3541.
##  9 Iran             Asia       2007    71.0   69453570    11606.
## 10 Iraq             Asia       2007    59.5   27499638     4471.
## # ℹ 23 more rows
```

2. `select()` → Choose specific columns.

What it does:
Imagine a spreadsheet with many columns. `select()` lets you pick just the variables you care about (and reorder them if you want).

Example: Keep only five columns.

```
gap %>%
  select(country, continent, year, lifeExp, gdpPercap)
```

```
## # A tibble: 1,704 × 5
##    country     continent  year lifeExp gdpPercap
##    <fct>       <fct>     <int>   <dbl>     <dbl>
##  1 Afghanistan Asia       1952    28.8      779.
##  2 Afghanistan Asia       1957    30.3      821.
##  3 Afghanistan Asia       1962    32.0      853.
##  4 Afghanistan Asia       1967    34.0      836.
##  5 Afghanistan Asia       1972    36.1      740.
##  6 Afghanistan Asia       1977    38.4      786.
##  7 Afghanistan Asia       1982    39.9      978.
##  8 Afghanistan Asia       1987    40.8      852.
##  9 Afghanistan Asia       1992    41.7      649.
## 10 Afghanistan Asia       1997    41.8      635.
## # ℹ 1,694 more rows
```

3. `arrange()` → Reorder rows.

What it does:

Think of `arrange()` as sorting your table. By default it sorts ascending (smallest → largest), but you can use `desc()` for descending order.

Example: sorting by population (ascending)

```
gap %>%
  arrange(pop) %>%    # sort ascending by population
  head(10)
```

```
## # A tibble: 10 × 6
##    country               continent  year lifeExp   pop gdpPercap
##    <fct>                 <fct>     <int>   <dbl> <int>     <dbl>
##  1 Sao Tome and Principe Africa     1952    46.5 60011      880.
##  2 Sao Tome and Principe Africa     1957    48.9 61325      861.
##  3 Djibouti              Africa     1952    34.8 63149     2670.
##  4 Sao Tome and Principe Africa     1962    51.9 65345     1072.
##  5 Sao Tome and Principe Africa     1967    54.4 70787     1385.
##  6 Djibouti              Africa     1957    37.3 71851     2865.
##  7 Sao Tome and Principe Africa     1972    56.5 76595     1533.
##  8 Sao Tome and Principe Africa     1977    58.6 86796     1738.
##  9 Djibouti              Africa     1962    39.7 89898     3021.
## 10 Sao Tome and Principe Africa     1982    60.4 98593     1890.
```

```
# Next: get highest life expectancy (2007) using filter in combo w arrange
gap %>%
  filter(year == 2007) %>%
  arrange(desc(lifeExp)) %>%
  head(10)
```

```
## # A tibble: 10 × 6
##    country          continent  year lifeExp       pop gdpPercap
##    <fct>            <fct>     <int>   <dbl>     <int>     <dbl>
##  1 Japan            Asia       2007    82.6 127467972    31656.
##  2 Hong Kong, China Asia       2007    82.2   6980412    39725.
##  3 Iceland          Europe     2007    81.8    301931    36181.
##  4 Switzerland      Europe     2007    81.7   7554661    37506.
##  5 Australia        Oceania    2007    81.2  20434176    34435.
##  6 Spain            Europe     2007    80.9  40448191    28821.
##  7 Sweden           Europe     2007    80.9   9031088    33860.
##  8 Israel           Asia       2007    80.7   6426679    25523.
##  9 France           Europe     2007    80.7  61083916    30470.
## 10 Canada           Americas   2007    80.7  33390141    36319.
```

4. `mutate` create new columns.

What it does: adds new variables (columns) or transforms existing ones.
- You can base the new variable on any calculation, combination, or condition from existing columns.

Example: compare gdp across countries

```
#first get gdp from gdp per capita
gap2<- gap %>%mutate(gdp = gdpPercap * pop)   # total GDP)
#View(gap2)


#scale in millions
gap2<- gap %>%
  mutate(gdp_million = (gdpPercap * pop) / 1e6)
#add in additional filtering
Arg_1982 <- gap2 %>% filter(country == "Argentina", year == 1982) %>%
  select(country, year, gdp_million)

##your turn! Let's try adding in this filter with mutate (filter(year == 2007, country %in% c("China", "India", "United States", "Luxembourg"))) to compare gdp results across these countries

##another mutate example: what's going on here?
gap %>%
  mutate(pop_millions = pop / 1e6) %>%
  select(country, year, pop, pop_millions) %>%
  head(5)
```

```
## # A tibble: 5 × 4
##   country      year      pop pop_millions
##   <fct>       <int>    <int>        <dbl>
## 1 Afghanistan  1952  8425333         8.43
## 2 Afghanistan  1957  9240934         9.24
## 3 Afghanistan  1962 10267083        10.3 
## 4 Afghanistan  1967 11537966        11.5 
## 5 Afghanistan  1972 13079460        13.1 
```

```
##and here?
gap %>%
  group_by(country) %>%
  mutate(is_growing = pop > lag(pop)) %>%
  select(country, year, pop, is_growing) %>%
  filter(country == "China") %>%
  head(10)
```

```
## # A tibble: 10 × 4
## # Groups:   country [1]
##    country  year        pop is_growing
##    <fct>   <int>      <int> <lgl>
##  1 China    1952  556263527 NA
##  2 China    1957  637408000 TRUE
##  3 China    1962  665770000 TRUE
##  4 China    1967  754550000 TRUE
##  5 China    1972  862030000 TRUE
##  6 China    1977  943455000 TRUE
##  7 China    1982 1000281000 TRUE
##  8 China    1987 1084035000 TRUE
##  9 China    1992 1164970000 TRUE
## 10 China    1997 1230075000 TRUE
```

group_by and summarise.

What it does: Summarises data into groups by calculating values like mean, sum, count, etc.

Example: compare gdp across countries.

- First use group_by() to define categories.
- use summarise() to collapse each group into one row with summary statistics.

```
# Average life expectancy by continent in 2007
gap %>%
  filter(year == 2007) %>%
  group_by(continent) %>%
  summarise(avg_lifeExp = mean(lifeExp), .groups = "drop") %>%
  arrange(desc(avg_lifeExp))
```

```
## # A tibble: 5 × 2
##   continent avg_lifeExp
##   <fct>           <dbl>
## 1 Oceania          80.7
## 2 Europe           77.6
## 3 Americas         73.6
## 4 Asia             70.7
## 5 Africa           54.8
```

count

What it does: generates quick frequencies of data - Quickly counts the number of rows in each category.

Example: number of rows per continent (sorted).

```
gap %>% count(continent, sort = TRUE)
```

```
## # A tibble: 5 × 2
##   continent     n
##   <fct>     <int>
## 1 Africa      624
## 2 Asia        396
## 3 Europe      360
## 4 Americas    300
## 5 Oceania      24
```

slice_max→ Get the “top N” rows

What it does: Selects the highest (or lowest - with slice_min()) values within a column.

Example: top 5 countries by GDP per capita in 2007

```
# Top 5 countries by GDP per capita in 2007
gap %>%
  filter(year == 2007) %>%
  slice_max(gdpPercap, n = 5, with_ties = FALSE) %>%
  select(country, continent, gdpPercap)
```

```
## # A tibble: 5 × 3
##   country       continent gdpPercap
##   <fct>         <fct>         <dbl>
## 1 Norway        Europe       49357.
## 2 Kuwait        Asia         47307.
## 3 Singapore     Asia         47143.
## 4 United States Americas     42952.
## 5 Ireland       Europe       40676.
```

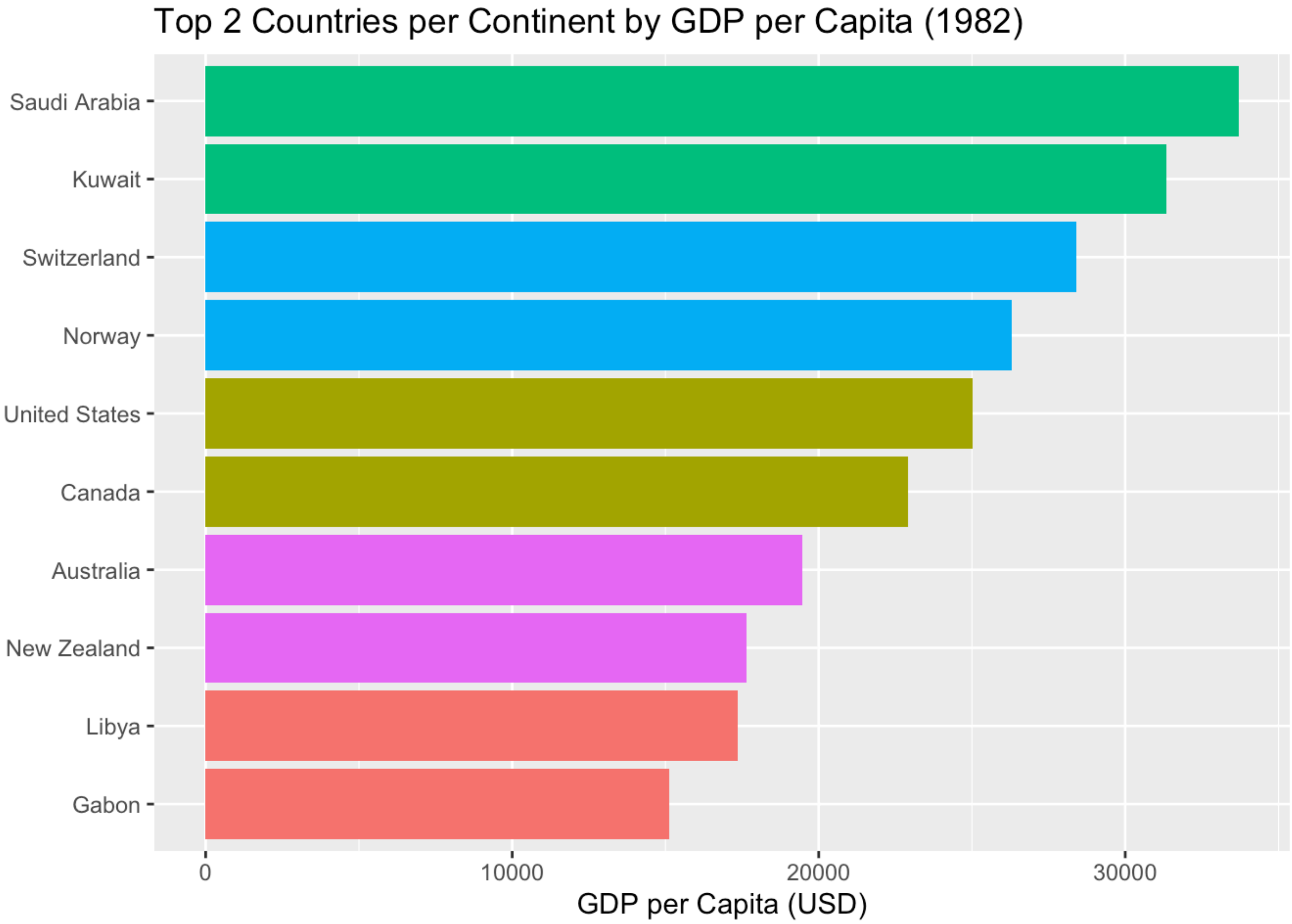
now in groups of two or three answer this question using our dplyr verbs

what top 2 countries per continent had the highest gdp per capita in 1982 and in 2007?

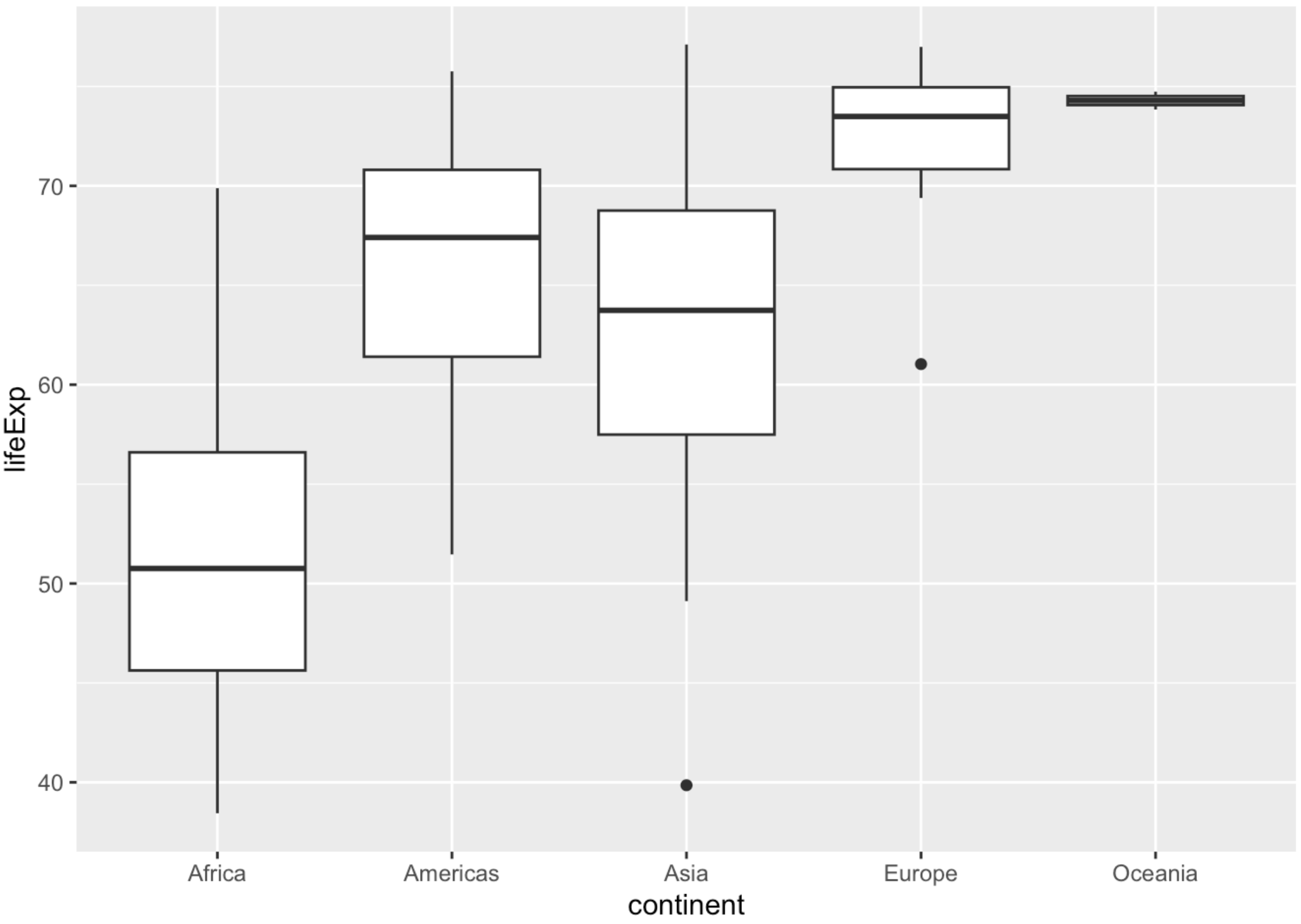
```
top2 <- gap %>% filter(year == 1982) %>%
  group_by(continent) %>%
  slice_max(order_by = gdpPercap, n = 2) %>%
  select(country, gdpPercap)
#View(top2)
```

Visualization w ggplot

```
# Bar plot
top2 %>%
  ggplot(aes(x = reorder(country, gdpPercap), y = gdpPercap, fill = continent)) +
  geom_col(show.legend = FALSE) +
  coord_flip() +
  labs(
    title = "Top 2 Countries per Continent by GDP per Capita (1982)",
    x = NULL,
    y = "GDP per Capita (USD)"
  )
```



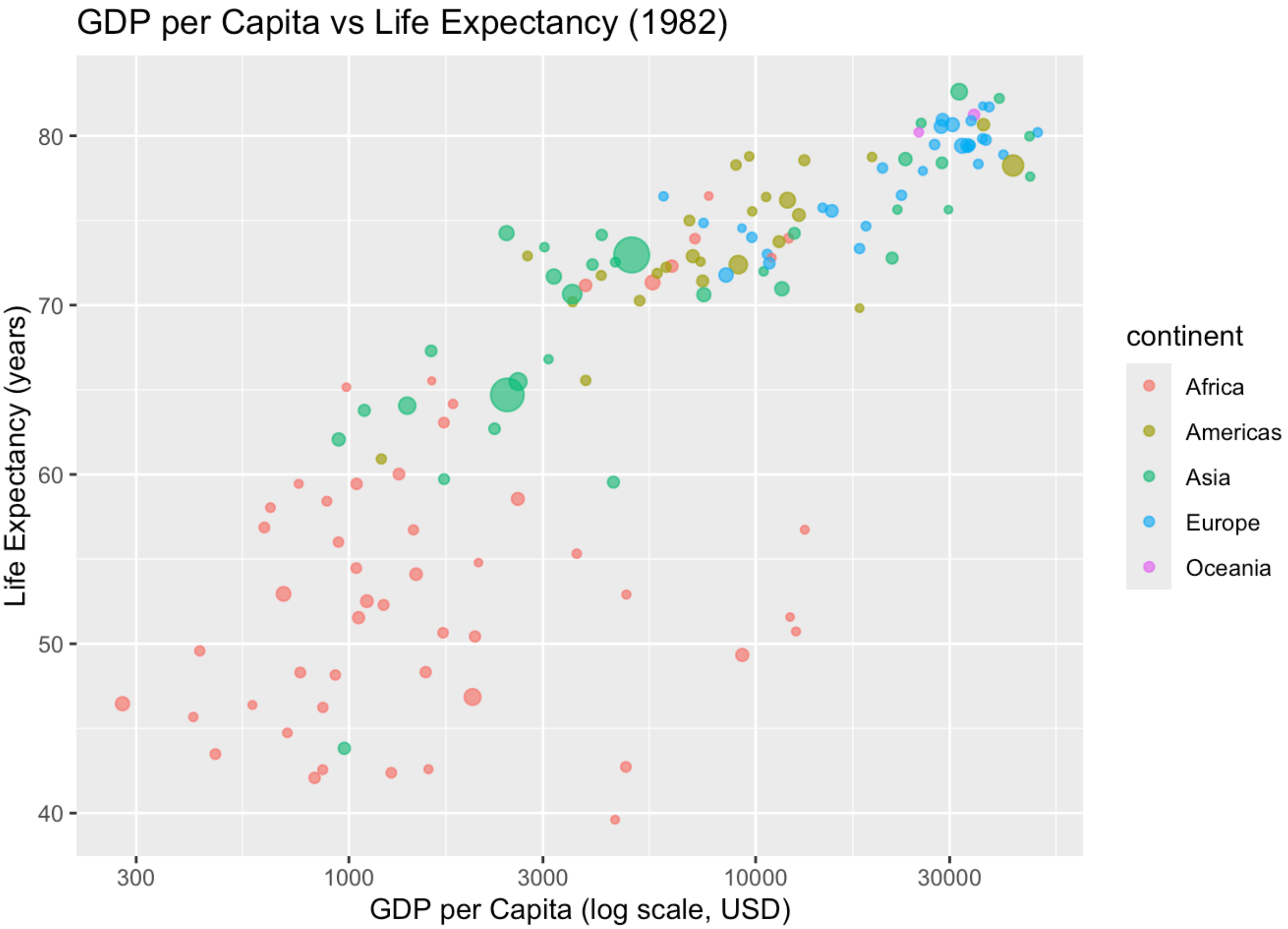
```
#compare distribution of life expectancies across continents in 1982
gap %>%
  filter(year == 1982) %>%
  ggplot(aes(continent, lifeExp)) +
  geom_boxplot()
```



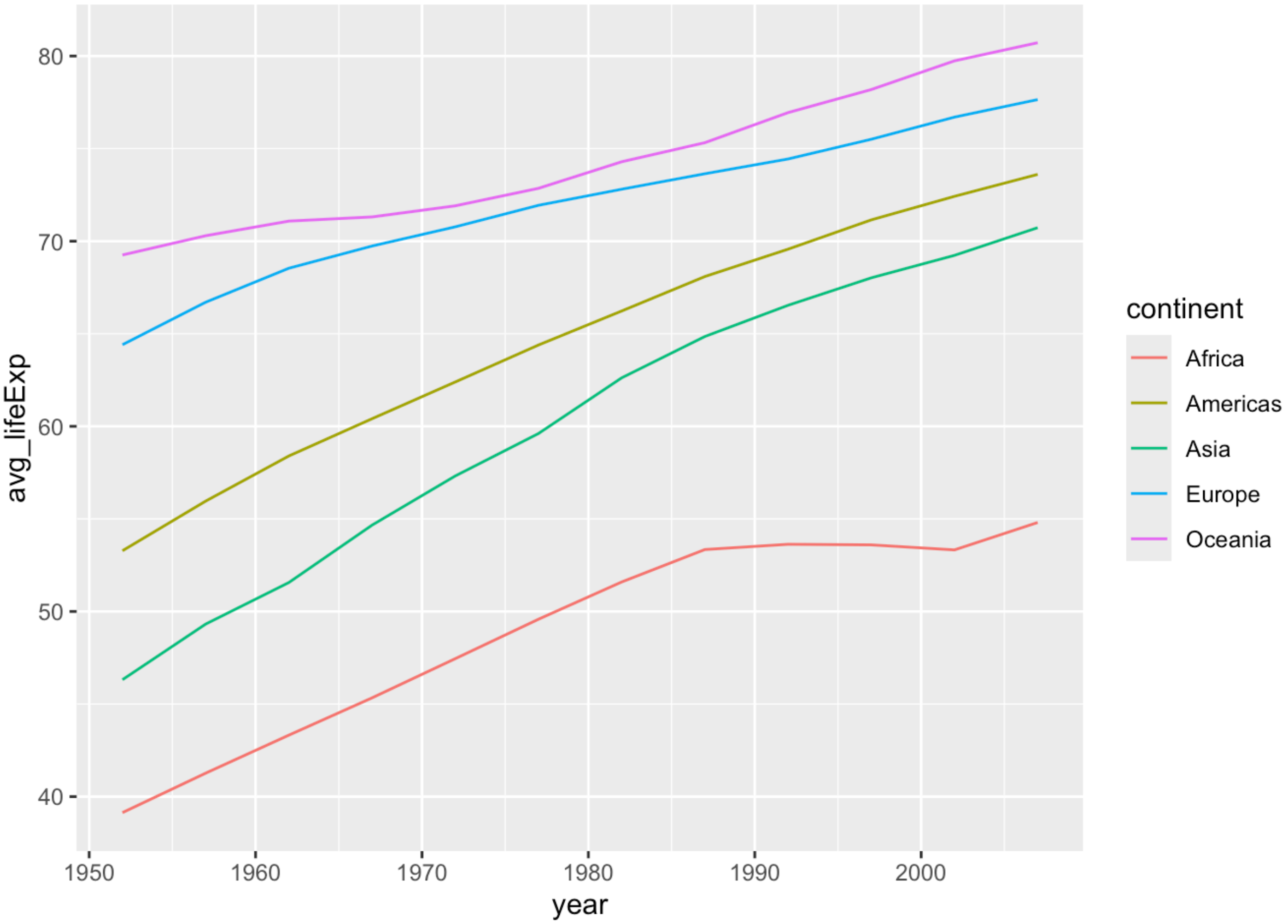
```
#correlation: visualize the relationship btw wealth and life expectancy
gap %>%
  filter(year == 2007) %>%
  summarise(cor(gdpPercap, lifeExp))
```

```
## # A tibble: 1 × 1
##   `cor(gdpPercap, lifeExp)`
##                       <dbl>
## 1                     0.679
```

```
gap %>%
  filter(year == 2007) %>%
  ggplot(aes(x = gdpPercap, y = lifeExp, color = continent, size = pop)) +
  geom_point(alpha = 0.7) +
  scale_x_log10() + #applies log base so easier to see lower income
  labs(
    title = "GDP per Capita vs Life Expectancy (1982)",
    x = "GDP per Capita (log scale, USD)",
    y = "Life Expectancy (years)"
  ) +
  guides(size = "none")
```



```
#trends over time
gap %>%
  group_by(continent, year) %>%
  summarise(avg_lifeExp = mean(lifeExp), .groups = "drop") %>%
  ggplot(aes(year, avg_lifeExp, color = continent)) +
  geom_line()
```


80
70
60
50
40
avg_lifeExp
1950
1960
1970
1980
1990
2000
year
continent
Africa
Americas
Asia
Europe
Oceania