Midterm I

September 30 2022

Section I: Gapminder Data

Health and income outcomes for 142 countries from 1952 to 2007 in increments of 5 years. The variables in the dataset are `country`, `continent`, `year`, `lifeExp`, `pop`, and `gdpPercap`. The descriptions for the variables are:

- `country` : name of the country, factor with 142 levels
- `continent`: name of the continent, factor with 5 levels
- `year` : ranges from 1952 to 2007 in increments of 5 years (12 distinct years)
- `lifeExp`: life expectancy at birth, in years
- `pop` : population
- `gdpPercap` : GDP per capita (US$, inflation-adjusted)

```
glimpse(gapminder)
Rows: 1,704
Columns: 6
$ country   <fct> "Afghanistan", "Afghanistan", "Afghanistan", "Afghanistan", ~
$ continent <fct> Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, Asia, ~
$ year      <int> 1952, 1957, 1962, 1967, 1972, 1977, 1982, 1987, 1992, 1997, ~
$ lifeExp   <dbl> 28.801, 30.332, 31.997, 34.020, 36.088, 38.438, 39.854, 40.8~
$ pop       <int> 8425333, 9240934, 10267083, 11537966, 13079460, 14880372, 12~
$ gdpPercap <dbl> 779.4453, 820.8530, 853.1007, 836.1971, 739.9811, 786.1134, ~
```

Part 1: What do the following codes do?

Provide a thorough and intuitive (2-3 sentences) description of the output from each of the following R chunks. The chunks either produce a new data set or a new plot; if it's a new data set, give the dimensions in addition to your description. Write your descriptions in regular English, without using variable names.

a.

```
gapminder %>%
  filter(
    continent == "Africa",
    country == "Somalia",
     year %in% seq(1952, 2007, 5)) %>%
 slice_min(lifeExp, n=1)
```

```
gapminder %>%
  filter(
    continent == "Africa",
    country == "Somalia",
     year %in% seq(1952, 2007, 5)) %>%
 slice_min(lifeExp, n=1)
# A tibble: 1 x 6
  country continent  year lifeExp     pop gdpPercap
  <fct>   <fct>     <int>   <dbl>   <int>     <dbl>
1 Somalia Africa     1952    33.0 2526994     1136.
```

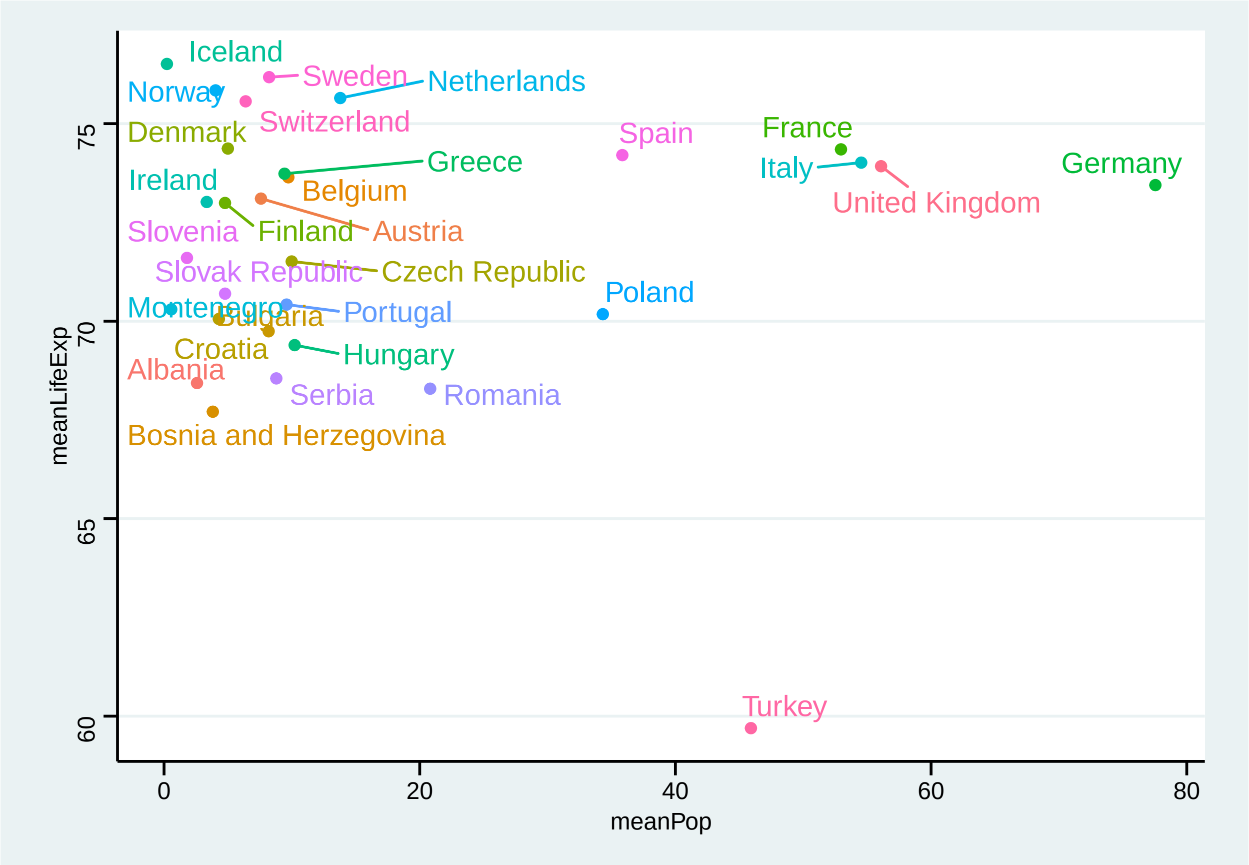
b.

```
# A tibble: 5 x 2
  continent   totalPop
  <fct>          <dbl>
1 Oceania     24549947
2 Europe     586098529
3 Americas   898871184
4 Africa     929539692
5 Asia      3811953827
```

```
gapminder %>%
  filter(year == 2007) %>%
  group_by(continent) %>%
  summarise(totalPop = sum(pop)) %>%
  arrange(totalPop)
# A tibble: 5 x 2
  continent   totalPop
  <fct>          <dbl>
1 Oceania     24549947
2 Europe     586098529
3 Americas   898871184
4 Africa     929539692
5 Asia      3811953827
```

c.

```
gapminder %>%
group_by(country, continent) %>%
summarize(meanPop = mean(pop)/1000000, meanLifeExp = mean(lifeExp)) %>%
  filter(continent == "Europe") %>%
ggplot(aes(x = meanPop, y = meanLifeExp, color = country, label = country)) +
  geom_point() +
  scale_color_discrete(guide = "none") +
  ggrepel::geom_text_repel()
```



```
gapminder %>%
group_by(country, continent) %>%
summarize(meanPop = mean(pop)/1000000, meanLifeExp = mean(lifeExp)) %>%
  filter(continent == "Europe") %>%
ggplot(aes(x = meanPop, y = meanLifeExp, color = country, label = country)) +
  geom_point() +
  scale_color_discrete(guide = "none") +
  ggrepel::geom_text_repel()
```

d.

```
gapminder %>%
  filter(year == 2007) %>%
  mutate(rank = min_rank(desc(lifeExp))) %>%
  filter(rank < 50) %>%
  arrange(rank) %>%
  slice_min(rank, n = 10) %>%
  select(-continent, -pop, -year, -gdpPercap)
# A tibble: 10 x 3
   country           lifeExp  rank
   <fct>               <dbl> <int>
 1 Japan                82.6     1
 2 Hong Kong, China     82.2     2
 3 Iceland             81.8     3
 4 Switzerland         81.7     4
 5 Australia           81.2     5
 6 Spain               80.9     6
 7 Sweden              80.9     7
 8 Israel              80.7     8
 9 France              80.7     9
10 Canada              80.7    10
```

```
gapminder %>%
  filter(year == 2007) %>%
  mutate(rank = min_rank(desc(lifeExp))) %>%
  filter(rank < 50) %>%
  arrange(rank) %>%
  slice_min(rank, n = 10) %>%
  select(-continent, -pop, -year, -gdpPercap)
```

e.

```
gapminder %>%
  group_by(continent, year) %>%
  summarize(
      mean_le = mean(lifeExp),
      median_le = quantile(lifeExp, 0.50),
      min_le = min(lifeExp),
      max_le = max(lifeExp),
      se_le = sd(lifeExp)/sqrt(n()))%>%
 slice_max(mean_le, n=1) %>%
 ungroup()
# A tibble: 5 x 7
  continent  year mean_le median_le min_le max_le se_le
  <fct>     <int>   <dbl>     <dbl>  <dbl>  <dbl> <dbl>
1 Africa     2007    54.8      52.9   39.6   76.4 1.34
2 Americas   2007    73.6      72.9   60.9   80.7 0.888
3 Asia       2007    70.7      72.4   43.8   82.6 1.39
4 Europe     2007    77.6      78.6   71.8   81.8 0.544
5 Oceania    2007    80.7      80.7   80.2   81.2 0.516
```

```
gapminder %>%
  group_by(continent, year) %>%
  summarize(
      mean_le = mean(lifeExp),
      median_le = quantile(lifeExp, 0.50),
      min_le = min(lifeExp),
      max_le = max(lifeExp),
      se_le = sd(lifeExp)/sqrt(n()))%>%
 slice_max(mean_le, n=1) %>%
 ungroup()
```

Part 2: Miscellaneous

a. Use piping operation to combine the following set of codes into one chained line of code to produce the same output.

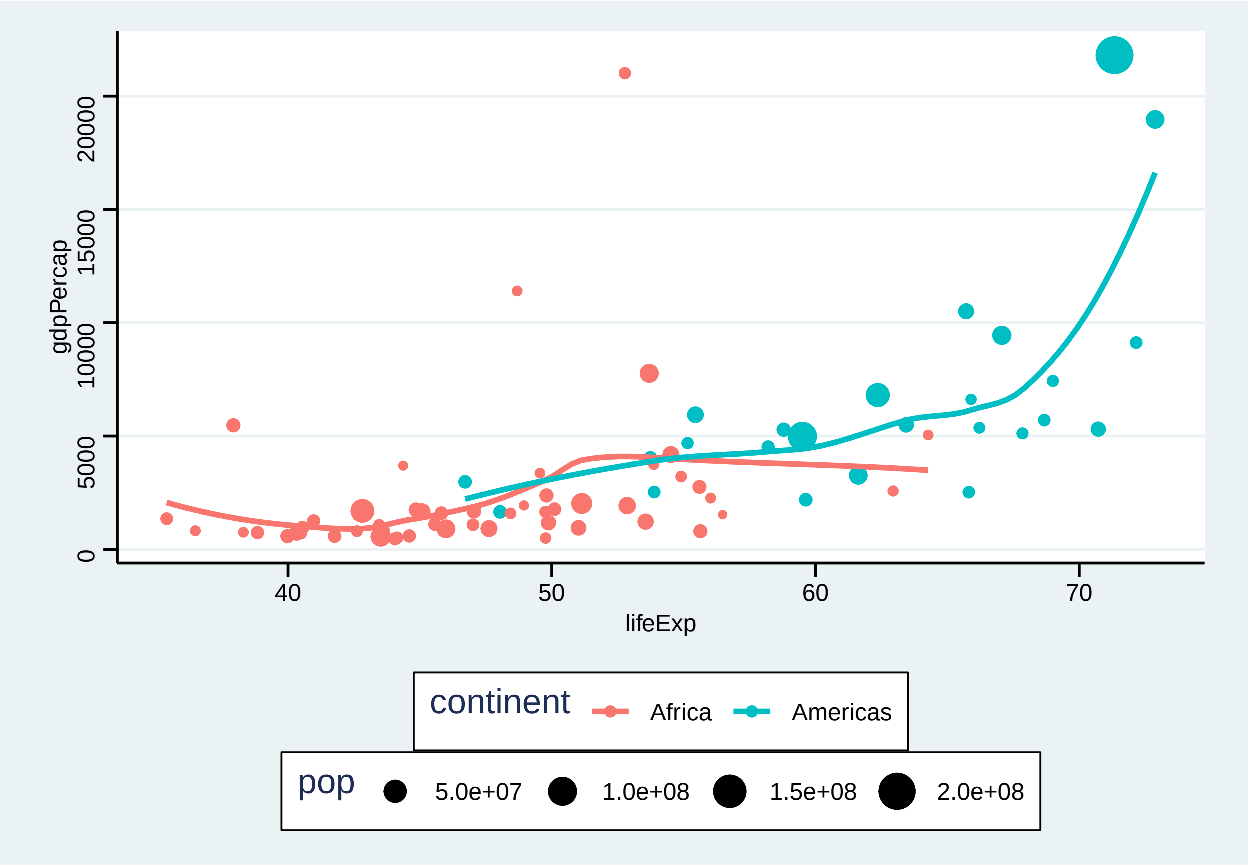
```
arrange(filter(gapminder, year == 1952), desc(gdpPercap))
# A tibble: 142 x 6
   country        continent  year lifeExp       pop gdpPercap
   <fct>          <fct>     <int>   <dbl>     <int>     <dbl>
 1 Kuwait         Asia       1952    55.6    160000   108382.
 2 Switzerland    Europe     1952    69.6   4815000    14734.
 3 United States  Americas   1952    68.4 157553000    13990.
 4 Canada         Americas   1952    68.8  14785584    11367.
 5 New Zealand    Oceania    1952    69.4   1994794    10557.
 6 Norway         Europe     1952    72.7   3327728    10095.
 7 Australia      Oceania    1952    69.1   8691212    10040.
 8 United Kingdom Europe     1952    69.2  50430000     9980.
 9 Bahrain        Asia       1952    50.9    120447     9867.
10 Denmark        Europe     1952    70.8   4334000     9692.
# ... with 132 more rows
```

```
# A tibble: 142 x 6
   country        continent  year lifeExp       pop gdpPercap
   <fct>          <fct>     <int>   <dbl>     <int>     <dbl>
 1 Kuwait         Asia       1952    55.6    160000   108382.
 2 Switzerland    Europe     1952    69.6   4815000    14734.
 3 United States  Americas   1952    68.4 157553000    13990.
 4 Canada         Americas   1952    68.8  14785584    11367.
 5 New Zealand    Oceania    1952    69.4   1994794    10557.
 6 Norway         Europe     1952    72.7   3327728    10095.
 7 Australia      Oceania    1952    69.1   8691212    10040.
 8 United Kingdom Europe     1952    69.2  50430000     9980.
 9 Bahrain        Asia       1952    50.9    120447     9867.
10 Denmark        Europe     1952    70.8   4334000     9692.
# ... with 132 more rows
```

b. Use piping operation to combine the following set of codes into one chained line of code to produce the same output. Also, comment on the output of the code.

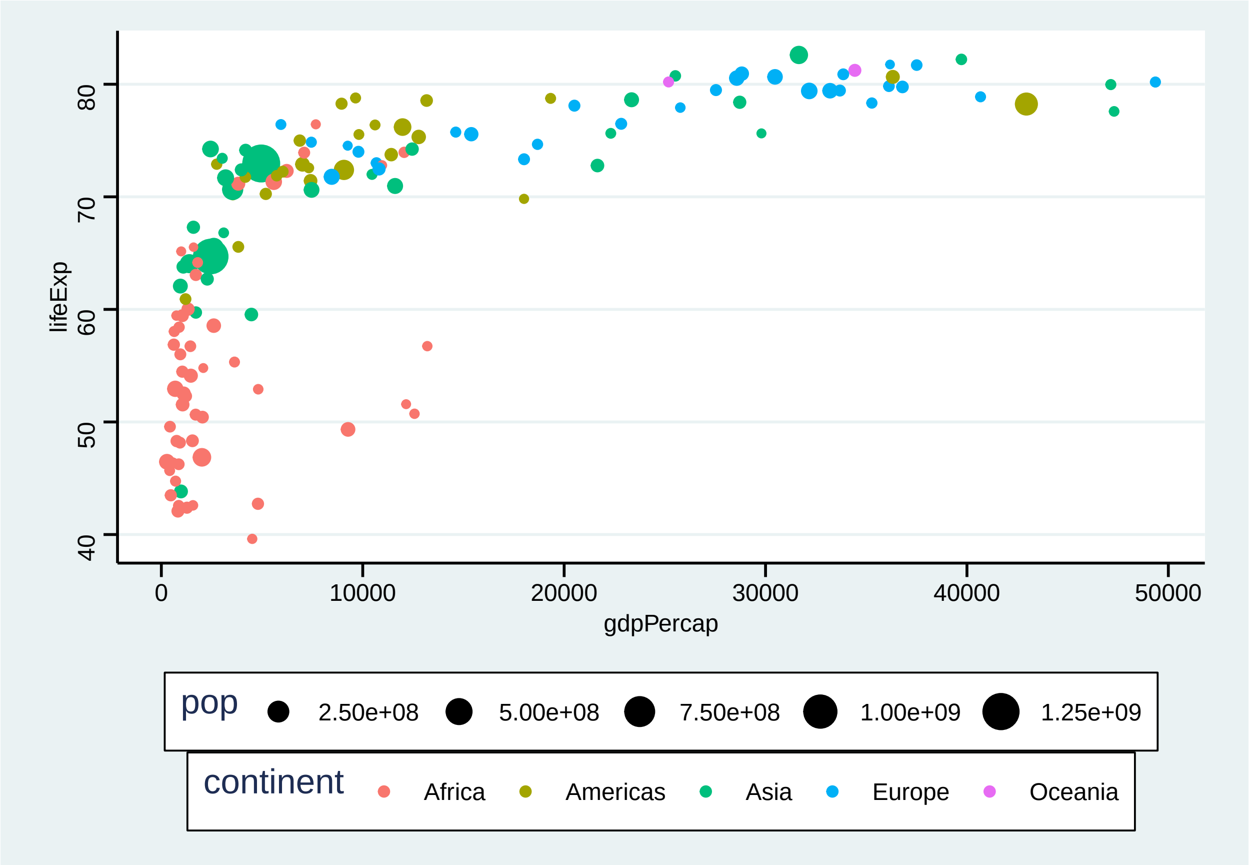
```
plot_data <- filter(gapminder, year == 1972 & continent %in% c("Africa", "Americas"))
ggplot(plot_data, aes(x = lifeExp, y = gdpPercap)) +
  geom_point()
```

```
gapminder %>% filter(year == 1972 & continent %in% c("Africa", "Americas")) %>%
  ggplot(aes(x = lifeExp, y = gdpPercap, color = continent)) +
  geom_point(aes(color = continent, size = pop)) +
  geom_smooth(se=FALSE)
```



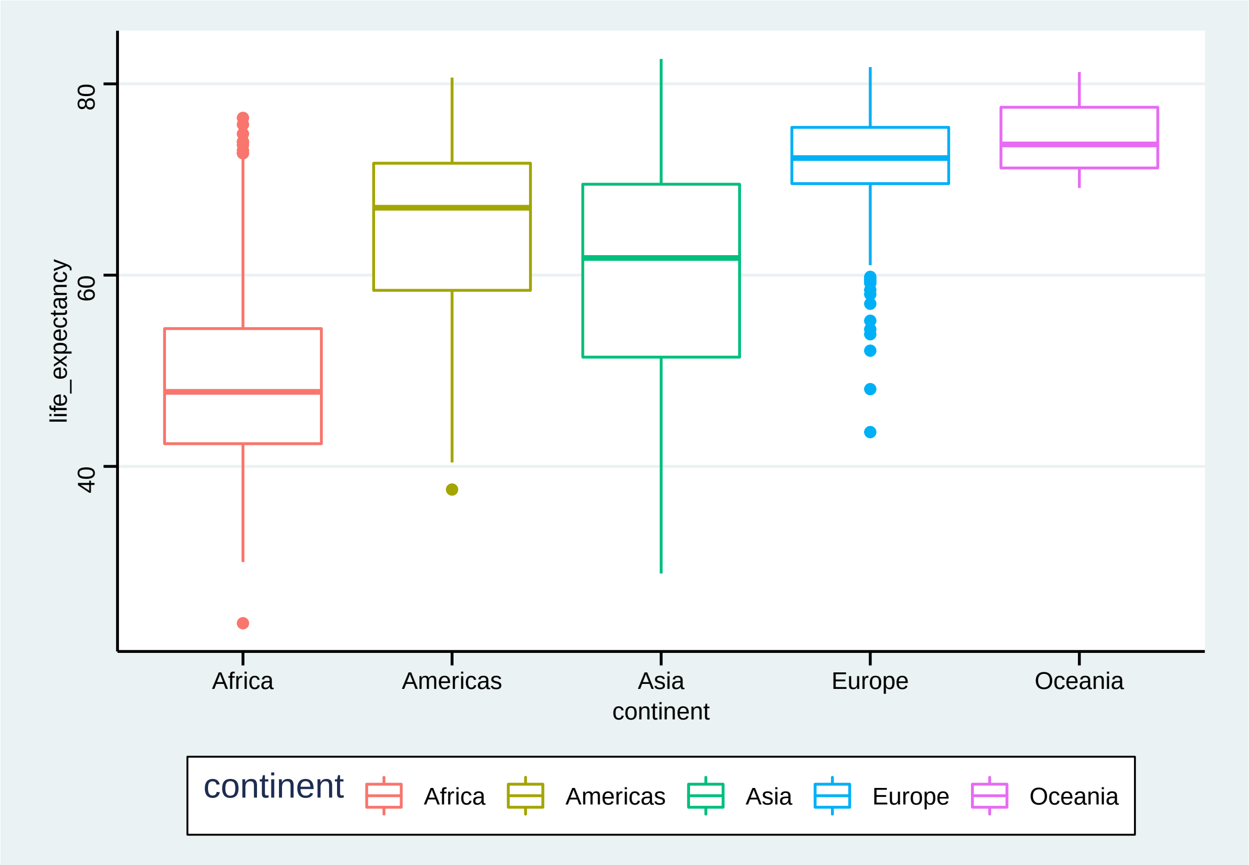
c. How would you improve the aesthetics of the plot below? Write your code modifications.

```
gapminder %>%
  filter(year == 2007) %>%
  ggplot() +
  geom_point(aes(x = gdpPercap, y = lifeExp, size = pop,
                 color = continent))
```



d.

```
gapminder %>%
  group_by(continent, year) %>%
  summarize(life_expectancy = lifeExp) %>%
  ggplot(aes(x = continent, y=life_expectancy, color = continent)) +
  geom_boxplot()
```



Part 3. Consider the following objects to answer the questions below.

```
x <- 1:4
y <- c(TRUE, as.factor(c(2,"3")), 4)
z <- list(z1 = x, z2 = y, z3 = c("cellar","door"))
```

(a) Consider the objects x, y and z. Which are atomic vectors and which are lists?

Answer:

(b) What does the following command evaluate to? Briefly explain your answer.

```
typeof(x)  # line 1
typeof(y) # line 2
```

Answer:

(c) What does the following command evaluate to? Briefly explain your answer.

```
z[["z3"]][1]
```

Answer:

(d) What does the following command evaluate to? Briefly explain your answer.

```
z[3][[1]][2]
```

Answer:

(e) What does the following command evaluate to? Briefly explain your answer.

```
x + y
```

Answer: