Tidyverse for Gapminder Dataset

Pouria

Last edited 12 January, 2022

Contents

Description

This is an introduction to the programming language R, focused on a powerful set of tools known as the Tidyverse. You'll learn the intertwined processes of data manipulation and visualization using the tools dplyr and ggplot2. You'll learn to manipulate data by filtering, sorting, and summarizing a real dataset of historical country data in order to answer exploratory questions. You'll then learn to turn this processed data into informative line plots, bar plots, histograms, and more with the ggplot2 package. You'll get a taste of the value of exploratory data analysis and the power of Tidyverse tools. This is a suitable introduction for those who have no previous experience in R and are interested in performing data analysis.

This document will include the following topics:

1. Data wrangling
2. Data visualization
3. Grouping and summarizing
4. Other useful types of visualization

Analyses will usually involve a cycle between these steps of data transformation and visualization, as well as additional components of the data science workflow, like statistical modeling.

Before anything, you to load the required libraries:

```
library(gapminder)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)
library(magrittr) # used for pipe operator
```

Data wrangling

In this chapter, you'll learn to do three things with a table: filter for particular observations, arrange the observations in a desired order, and mutate to add or change a column. You'll see how each of these steps allows you to answer questions about your data. You will also learn the essential skills of data visualization using the ggplot2 package, and you'll see how the dplyr and ggplot2 packages work closely together to create informative graphs.

First, take a look at the `gapminder` data set

```
gapminder
```

```
## # A tibble: 1,704 x 6
##    country     continent  year lifeExp      pop gdpPercap
##    <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
##  1 Afghanistan Asia       1952    28.8  8425333      779.
##  2 Afghanistan Asia       1957    30.3  9240934      821.
##  3 Afghanistan Asia       1962    32.0 10267083      853.
##  4 Afghanistan Asia       1967    34.0 11537966      836.
##  5 Afghanistan Asia       1972    36.1 13079460      740.
##  6 Afghanistan Asia       1977    38.4 14880372      786.
##  7 Afghanistan Asia       1982    39.9 12881816      978.
##  8 Afghanistan Asia       1987    40.8 13867957      852.
##  9 Afghanistan Asia       1992    41.7 16317921      649.
## 10 Afghanistan Asia       1997    41.8 22227415      635.
## # ... with 1,694 more rows
```

```
summary(gapminder)
```

```
##         country        continent        year         lifeExp
##  Afghanistan:  12   Africa  :624   Min.   :1952   Min.   :23.60
##  Albania    :  12   Americas:300   1st Qu.:1966   1st Qu.:48.20
##  Algeria    :  12   Asia    :396   Median :1980   Median :60.71
##  Angola     :  12   Europe  :360   Mean   :1980   Mean   :59.47
##  Argentina  :  12   Oceania : 24   3rd Qu.:1993   3rd Qu.:70.85
##  Australia  :  12                  Max.   :2007   Max.   :82.60
##  (Other)    :1632
##       pop              gdpPercap
##  Min.   :6.001e+04   Min.   :   241.2
##  1st Qu.:2.794e+06   1st Qu.:  1202.1
##  Median :7.024e+06   Median :  3531.8
##  Mean   :2.960e+07   Mean   :  7215.3
##  3rd Qu.:1.959e+07   3rd Qu.:  9325.5
##  Max.   :1.319e+09   Max.   :113523.1
##
```

filter()

Using the verbs from package `dplyr`, we want to `filter()` particular observations from the year 1952.

```
gapminder_1952 <- gapminder %>%
  filter(year == 1952)

gapminder_1952
```

```
## # A tibble: 142 x 6
##    country     continent  year lifeExp      pop gdpPercap
##    <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
##  1 Afghanistan Asia       1952    28.8  8425333      779.
##  2 Albania     Europe     1952    55.2  1282697     1601.
##  3 Algeria     Africa     1952    43.1  9279525     2449.
##  4 Angola      Africa     1952    30.0  4232095     3521.
##  5 Argentina   Americas   1952    62.5 17876956     5911.
##  6 Australia   Oceania    1952    69.1  8691212    10040.
##  7 Austria     Europe     1952    66.8  6927772     6137.
##  8 Bahrain     Asia       1952    50.9   120447     9867.
##  9 Bangladesh  Asia       1952    37.5 46886859      684.
## 10 Belgium     Europe     1952    68    8730405     8343.
## # ... with 132 more rows
```

Now, suppose you would want to find the observation that has the highest GDP per capita.

```
# indexing with base R
gapminder[gapminder$gdpPercap==max(gapminder$gdpPercap),]
```

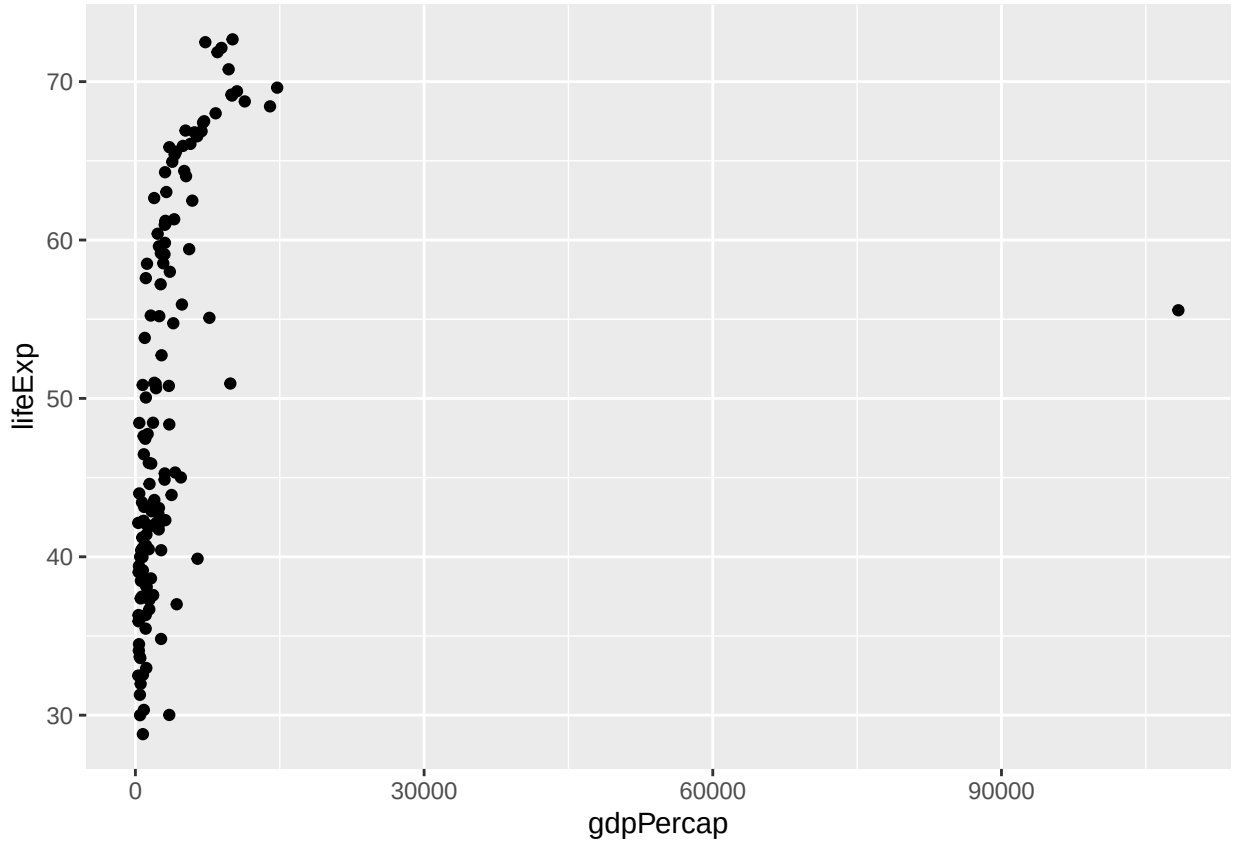
```
## # A tibble: 1 x 6
##   country continent  year lifeExp    pop gdpPercap
##   <fct>   <fct>     <int>   <dbl>  <int>     <dbl>
## 1 Kuwait  Asia       1957    58.0 212846   113523.
```

```
# subsetting with dplyr
gapminder %>%
  arrange(desc(gdpPercap)) %>%
  slice(1)
```

```
## # A tibble: 1 x 6
##   country continent  year lifeExp    pop gdpPercap
##   <fct>   <fct>     <int>   <dbl>  <int>     <dbl>
## 1 Kuwait  Asia       1957    58.0 212846   113523.
```

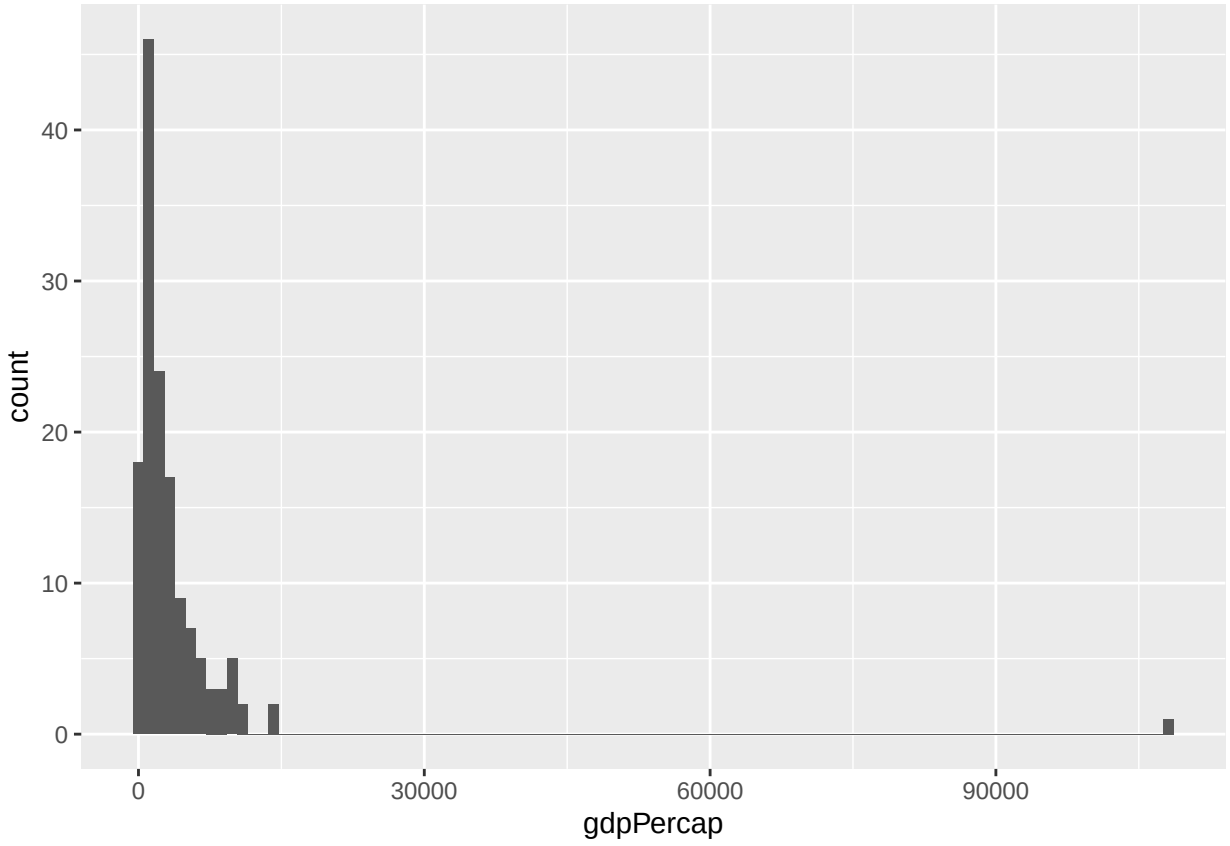
Now suppose we want to create a scatter plot of the data from year 1952 with life expectancy (`pop`) on y-axis and GDP per capita (`gdpPercap`) on the x-axis.

```
ggplot(gapminder_1952, aes(x = gdpPercap , y = lifeExp)) +
  geom_point()
```



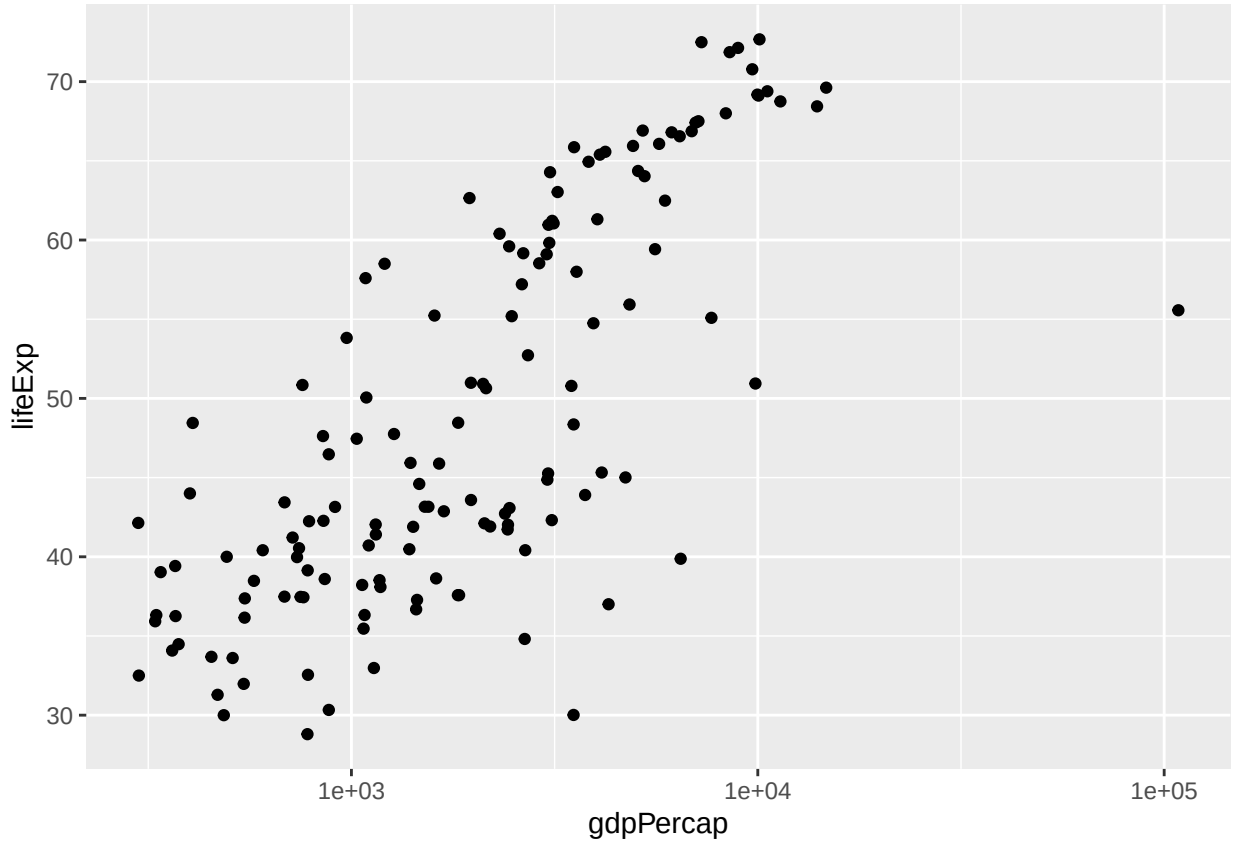
Any interesting observation? Correct! higher income countries tend to have higher life expectancy. One problem, however, is that the plot for the most part is not quite readable because a lot of countries got crammed into the leftmost part of the x-axis. The reason for this is the distribution of GDP per capita (`gdpPercap`) spans several orders of magnitude. To visualize this use `histogram()`

```
ggplot(gapminder_1952, aes(x = gdpPercap)) +
  geom_histogram(bins = 100)
```



In such circumstances, it is helpful to work with a logarithmic scale.

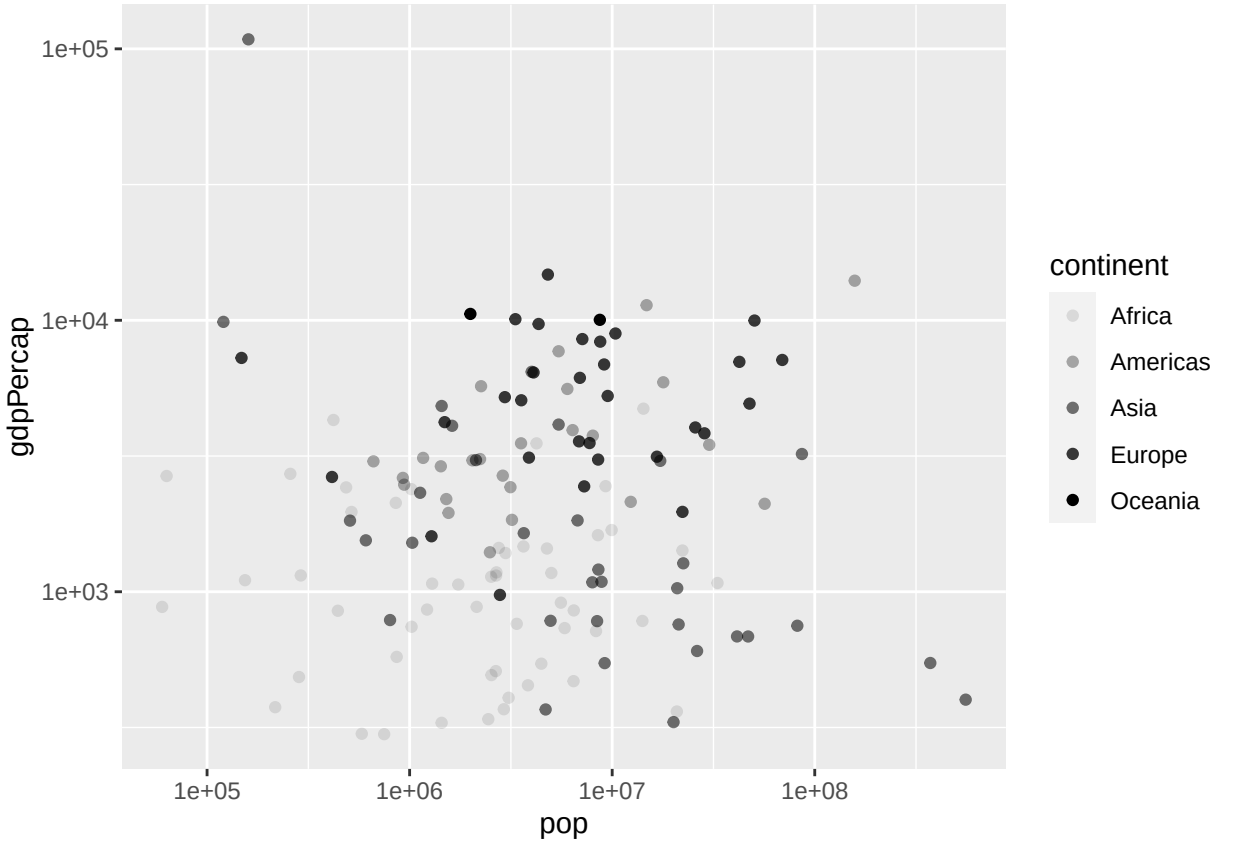
```
ggplot(gapminder_1952, aes(x = gdpPercap , y = lifeExp)) +
  geom_point() +
  scale_x_log10()
```



Let's test this for the variables population (`pop`) on x-axis and `gdpPercap` on y-axis.

```
ggplot(gapminder_1952, aes(x = pop , y = gdpPercap)) +
  geom_point(mapping=aes(alpha=continent)) +
  scale_x_log10() +
  scale_y_log10()
```

```
## Warning: Using alpha for a discrete variable is not advised.
```



In the last plot, for `x = pop` and `y = gdpPercap`, suppose you would like to also visualize information about other variables in the same plot. Other interesting variables that could give us more insight into the data are life expectancy (`lifeExp`) and continent (`continent`). The variable `continent` is a categorical variable. Let's see how many levels it has

```
levels(gapminder_1952$continent)
```

```
## [1] "Africa"   "Americas" "Asia"     "Europe"   "Oceania"
```

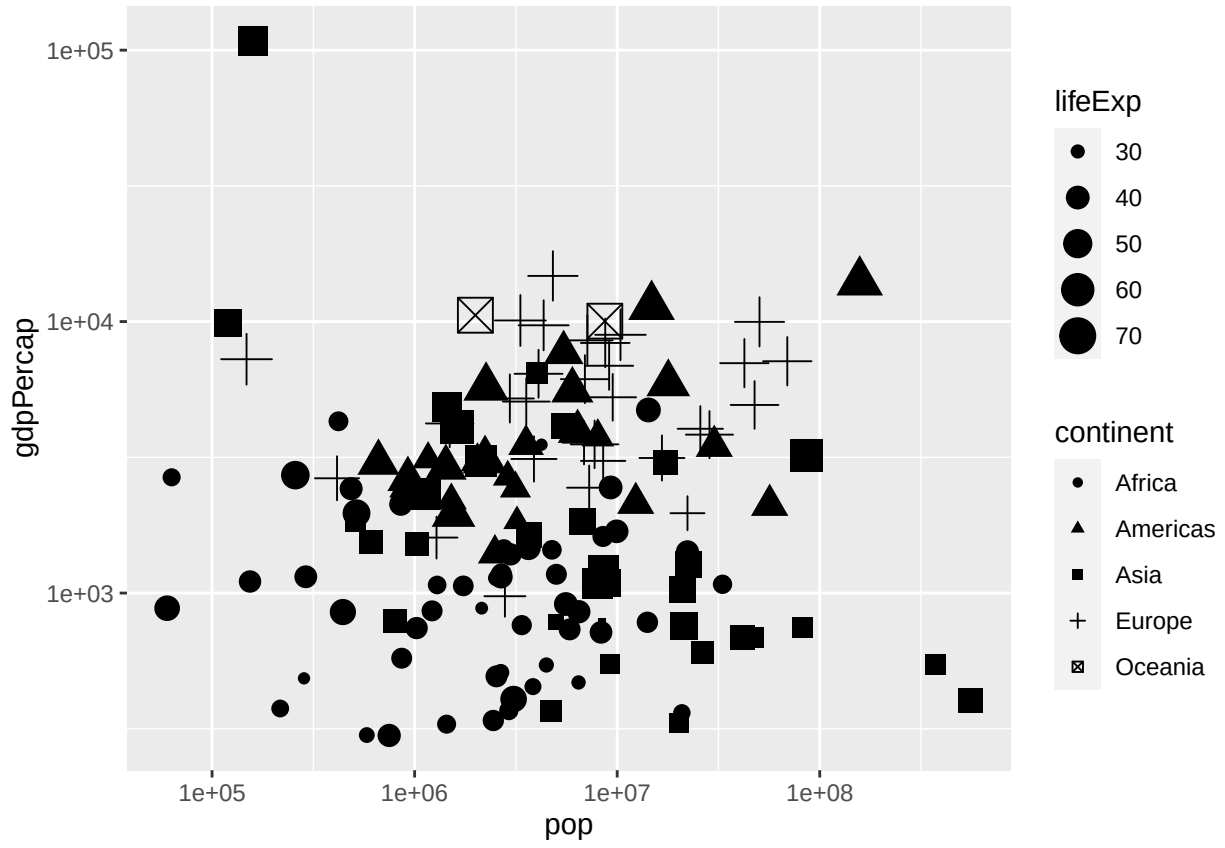
```
# or using dplyr
gapminder_1952 %>%
  summarize(levels(continent))
```

```
## # A tibble: 5 x 1
##   `levels(continent)`
##   <chr>
## 1 Africa
## 2 Americas
## 3 Asia
## 4 Europe
## 5 Oceania
```

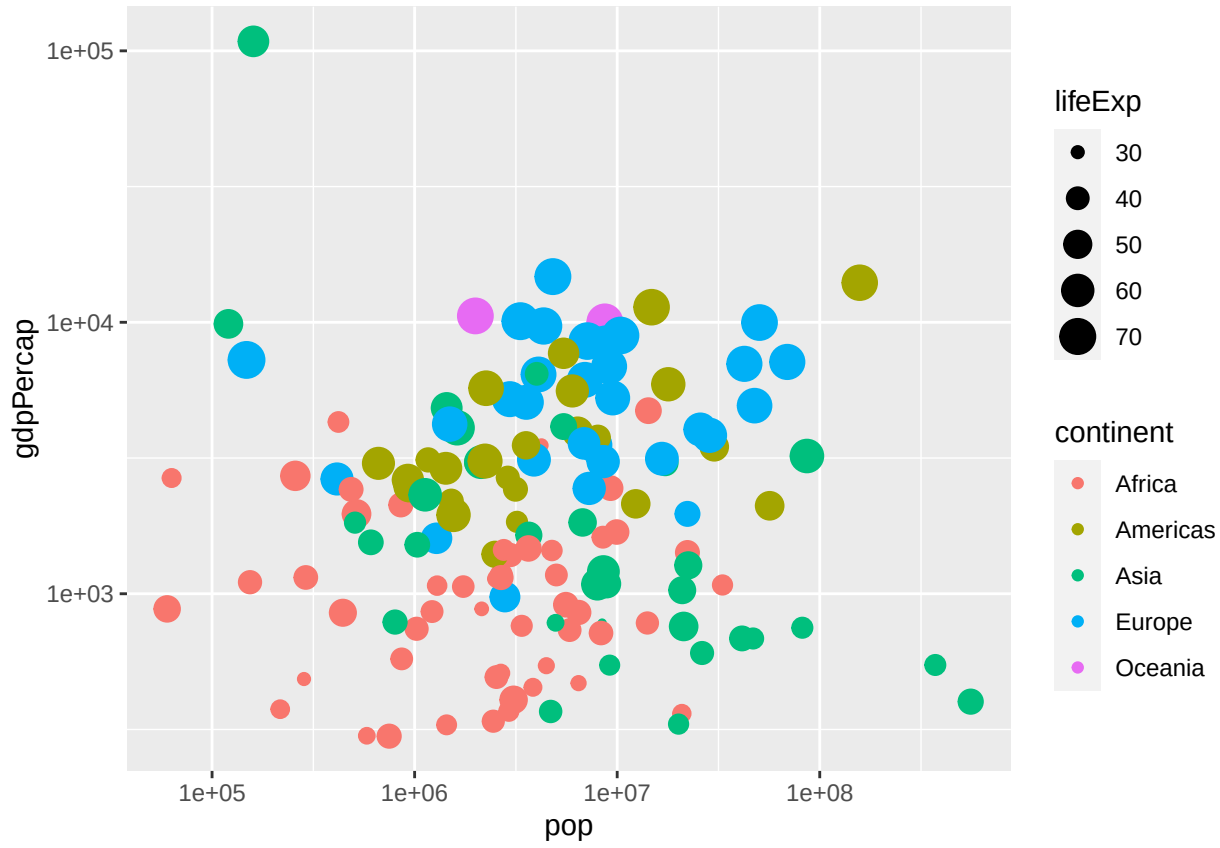
```
#factor(gapminder_1952$continent) %>% levels()

# levels(c(1, 2, 3, 4, 5))
# factor(c(1, 2, 3, 4, 5)) %>% levels()
```

Since the variable `continent` has a few number of levels we can visualize that on the data via either `shape` or `color`. Also, the life expectancy information could be visualized via using `lifeExp` to tune the `size` of the data points.

```
ggplot(gapminder_1952, aes(x = pop , y = gdpPercap, shape=continent, size=lifeExp)) +
  geom_point() +
  scale_x_log10() +
  scale_y_log10()
```



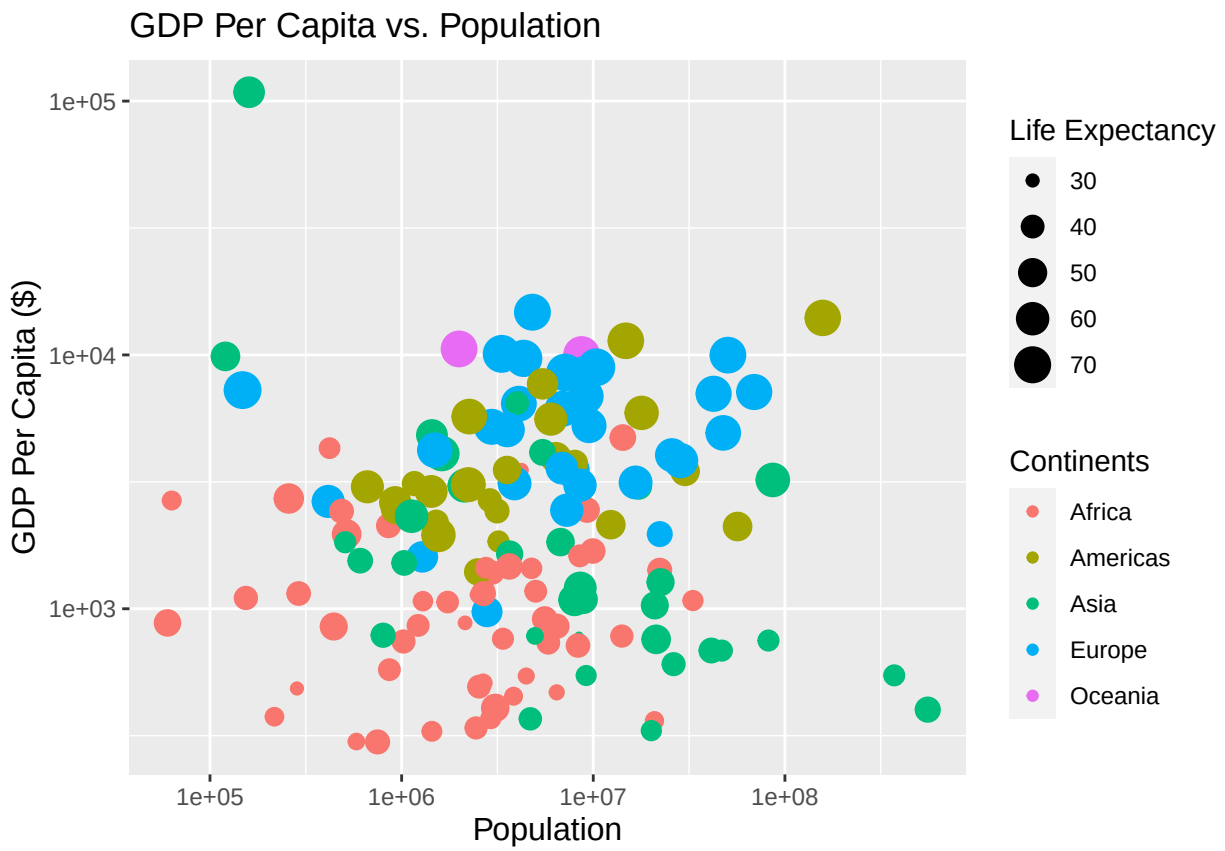

```
ggplot(gapminder_1952, aes(x = pop , y = gdpPercap, color=continent, size=lifeExp)) +
  geom_point() +
  scale_x_log10() +
  scale_y_log10()
```



Consider adding the layers `scale_color_manual(values=c('#999999', '#ex2', ...))` and `scale_shape_manual(values 16, 17, ...))` to the plot for manual choice of colors and shapes, respectively.

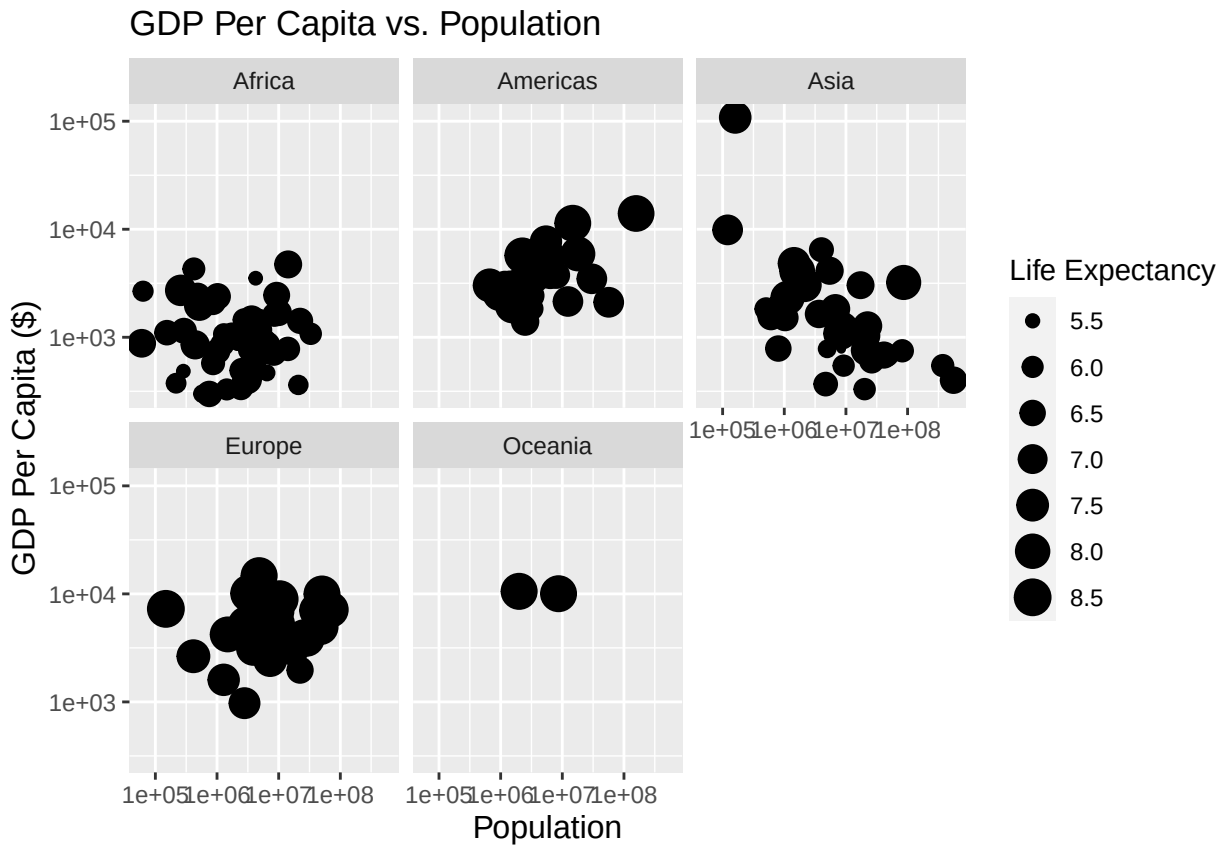
Below is shown other properties related to the panel and axes. To further your knowledge, visit here.

```
ggplot(gapminder_1952, aes(x = pop , y = gdpPercap, color=continent, size=lifeExp)) +
  geom_point() +
  scale_x_log10() +
  scale_y_log10() +
  labs(
    x = "Population",
    y = "GDP Per Capita ($)",
    title = "GDP Per Capita vs. Population",
    color = "Continents",
    size = "Life Expectancy"
    ) +
  theme(
    legend.position = "right",
    axis.title.x = element_text(size = 12),
    axis.title.y = element_text(size = 12) )
```



Now, suppose instead of separating observations from different continents by different colors, we would want to divide the graph into **subplots based on a variable**, such as `continent`. We can achieve this by adding a `facet_wrap(~variable)` layer to the plot.

```
ggplot(gapminder_1952, aes(x = pop , y = gdpPercap, size=lifeExp^.5)) +
  geom_point() +
  scale_x_log10() +
  scale_y_log10() +
  facet_wrap(~continent) +
  labs(
    x = "Population",
    y = "GDP Per Capita ($)",
    title = "GDP Per Capita vs. Population",
    color = "Continents",
    size = "Life Expectancy"
    ) +
  theme(
    legend.position = "right",
    axis.title.x = element_text(size = 12),
    axis.title.y = element_text(size = 12) )
```



As a practice, create a scatter plot of the `gapminder` data with GDP per capita on the x-axis and life expectancy on the y-axis, with continents represented by color and population by size divided by years in subplots. Put any of the axes on a log scale if need be.

As the last note on the verb `filter()`, we can use it for more than one condition. Say we would like to retrieve only the observations from China in the year 2002

```
gapminder %>%
  filter(country == "China", year == 2002)
```

```
## # A tibble: 1 x 6
##   country continent  year lifeExp        pop gdpPercap
##   <fct>   <fct>     <int>   <dbl>      <int>     <dbl>
## 1 China   Asia       2002    72.0 1280400000     3119.
```

arrange()

Now suppose we intend to sort observations in ascending or descending order of a particular variable, say `lifeExp`. We will use verb `arrange` from `dplyr` to achieve this.

```
# arrange in ascending order
gapminder %>%
  arrange(lifeExp)
```

```
## # A tibble: 1,704 x 6
```

```
##    country      continent  year lifeExp     pop gdpPercap
##    <fct>        <fct>     <int>   <dbl>   <int>     <dbl>
##  1 Rwanda       Africa     1992    23.6 7290203      737.
##  2 Afghanistan  Asia       1952    28.8 8425333      779.
##  3 Gambia       Africa     1952    30    284320      485.
##  4 Angola       Africa     1952    30.0 4232095     3521.
##  5 Sierra Leone Africa     1952    30.3 2143249      880.
##  6 Afghanistan  Asia       1957    30.3 9240934      821.
##  7 Cambodia     Asia       1977    31.2 6978607      525.
##  8 Mozambique   Africa     1952    31.3 6446316      469.
##  9 Sierra Leone Africa     1957    31.6 2295678     1004.
## 10 Burkina Faso Africa     1952    32.0 4469979      543.
## # ... with 1,694 more rows
```

```
gapminder %>%
  arrange(desc(lifeExp))
```

```
## # A tibble: 1,704 x 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 Japan            Asia       2002    82   127065841    28605.
##  4 Iceland          Europe     2007    81.8    301931    36181.
##  5 Switzerland      Europe     2007    81.7   7554661    37506.
##  6 Hong Kong, China Asia       2002    81.5   6762476    30209.
##  7 Australia        Oceania    2007    81.2  20434176    34435.
##  8 Spain            Europe     2007    80.9  40448191    28821.
##  9 Sweden           Europe     2007    80.9   9031088    33860.
## 10 Israel           Asia       2007    80.7   6426679    25523.
## # ... with 1,694 more rows
```

Sometimes we want to sort the observations from just a specific year, e.g., 1957 in descending order of population. For this, we would have to use multiple `dplyr` verbs, being `filter()` and `arrange()` here. We first use `filter()` to extract observations from just the year 1957, and then use `arrange()` to sort those observations in descending order of population (`pop`).

```
gapminder %>%
  filter(year == 1957) %>%
  arrange(desc(year))
```

```
## # A tibble: 142 x 6
##    country     continent  year lifeExp      pop gdpPercap
##    <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
##  1 Afghanistan Asia       1957    30.3  9240934      821.
##  2 Albania     Europe     1957    59.3  1476505     1942.
##  3 Algeria     Africa     1957    45.7 10270856     3014.
##  4 Angola      Africa     1957    32.0  4561361     3828.
##  5 Argentina   Americas   1957    64.4 19610538     6857.
##  6 Australia   Oceania    1957    70.3  9712569    10950.
##  7 Austria     Europe     1957    67.5  6965860     8843.
##  8 Bahrain     Asia       1957    53.8   138655    11636.
```

```
##  9 Bangladesh  Asia       1957    39.3 51365468      662.
## 10 Belgium     Europe     1957    69.2  8989111     9715.
## # ... with 132 more rows
```

mutate()

Suppose we want a variable in `gapminder` data set measured in a different unit, e.g., life expectancy (`lifeExp`) to be measured in months instead of years. For this, one can use `mutate()` verb to change this column.

This is a good time to practice writing equations. Say we would like to write this mutation in form of an equation. We will use LaTeX equation. For inline, you could use $lifeExp_{months} = lifeExp * 12$. Also, there are two ways to write this equation on a separate line:

$$lifeExp_{months} = lifeExp * 12$$

or

$$lifeExp_{months} = lifeExp \times 12 \tag{1}$$

As a reference, _ and ^ are used for subscript and superscript, respectively.

```
gapminder %>%
  mutate(lifeExp = lifeExp*12)
```

```
## # A tibble: 1,704 x 6
##    country     continent  year lifeExp      pop gdpPercap
##    <fct>       <fct>     <int>   <dbl>    <int>     <dbl>
##  1 Afghanistan Asia       1952    346.  8425333      779.
##  2 Afghanistan Asia       1957    364.  9240934      821.
##  3 Afghanistan Asia       1962    384. 10267083      853.
##  4 Afghanistan Asia       1967    408. 11537966      836.
##  5 Afghanistan Asia       1972    433. 13079460      740.
##  6 Afghanistan Asia       1977    461. 14880372      786.
##  7 Afghanistan Asia       1982    478. 12881816      978.
##  8 Afghanistan Asia       1987    490. 13867957      852.
##  9 Afghanistan Asia       1992    500. 16317921      649.
## 10 Afghanistan Asia       1997    501. 22227415      635.
## # ... with 1,694 more rows
```

However, a more efficient practice is to create a new column for the mutation as follows:

```
gapminder %>%
  mutate(lifeExpMonths = lifeExp*12)
```

```
## # A tibble: 1,704 x 7
##    country     continent  year lifeExp      pop gdpPercap lifeExpMonths
##    <fct>       <fct>     <int>   <dbl>    <int>     <dbl>         <dbl>
##  1 Afghanistan Asia       1952    28.8  8425333      779.          346.
##  2 Afghanistan Asia       1957    30.3  9240934      821.          364.
##  3 Afghanistan Asia       1962    32.0 10267083      853.          384.
##  4 Afghanistan Asia       1967    34.0 11537966      836.          408.
```

```
##  5 Afghanistan Asia       1972    36.1 13079460      740.          433.
##  6 Afghanistan Asia       1977    38.4 14880372      786.          461.
##  7 Afghanistan Asia       1982    39.9 12881816      978.          478.
##  8 Afghanistan Asia       1987    40.8 13867957      852.          490.
##  9 Afghanistan Asia       1992    41.7 16317921      649.          500.
## 10 Afghanistan Asia       1997    41.8 22227415      635.          501.
## # ... with 1,694 more rows
```

Now, suppose you want to combine all the three verbs you have learned thus far, to sort and find the countries with the highest life expectancy in months, in the year 2007.

```
gapminder %>%
  filter(year == 2007) %>%
  mutate(lifeExpMonths = lifeExp * 12) %>%
  arrange(desc(lifeExpMonths))
```

```
## # A tibble: 142 x 7
##    country          continent  year lifeExp       pop gdpPercap lifeExpMonths
##    <fct>            <fct>     <int>   <dbl>     <int>     <dbl>         <dbl>
##  1 Japan            Asia       2007    82.6 127467972    31656.          991.
##  2 Hong Kong, China Asia       2007    82.2   6980412    39725.          986.
##  3 Iceland          Europe     2007    81.8    301931    36181.          981.
##  4 Switzerland      Europe     2007    81.7   7554661    37506.          980.
##  5 Australia        Oceania    2007    81.2  20434176    34435.          975.
##  6 Spain            Europe     2007    80.9  40448191    28821.          971.
##  7 Sweden           Europe     2007    80.9   9031088    33860.          971.
##  8 Israel           Asia       2007    80.7   6426679    25523.          969.
##  9 France           Europe     2007    80.7  61083916    30470.          968.
## 10 Canada           Americas   2007    80.7  33390141    36319.          968.
## # ... with 132 more rows
```

Grouping and summarizing

So far you've been answering questions about individual country-year pairs, but you may be interested in aggregations of the data, such as the average life expectancy of all countries within each year. Here you'll learn to use the group by and summarize verbs, which collapse large datasets into manageable summaries.

summarize()

Suppose you would want to summarize many observations into a single data point, e.g., the median life expectancy across all countries and years and save it into the new variable `medianLifeExp`

```
gapminder %>%
  summarize(medianLifeExp = median(lifeExp))
```

```
## # A tibble: 1 x 1
##   medianLifeExp
##           <dbl>
## 1          60.7
```

Now rather than summarizing the entire dataset, you may be interested to find the median life expectancy for only the particular year 1957.

```
gapminder %>%
  filter(year == 1957) %>%
  summarize(medianLifeExp = median(lifeExp))
```

```
## # A tibble: 1 x 1
##   medianLifeExp
##           <dbl>
## 1          48.4
```

Summarize the data across all countries from year 1957 to median life expectancy and maximum GDP per capita.

group_by()

What if we weren't interested just in the average for the year 2007, but for each of the years in the dataset? You could rerun this code and change the year each time, but that's very tedious!

Instead, you can use the `group_by()` verb, which tells `dplyr` to summarize within groups instead of summarizing the entire dataset. Notice that this replaces doing the `filter()` for a specific year.

Now, suppose you would want to perform the same summary as in last code chunk within each year in the dataset rather than only for 1957.

```
gapminder %>%
  group_by(year) %>%
  summarize(medianLifeExp = median(lifeExp),
            maxGdpPercap = max(gdpPercap))
```

```
## # A tibble: 12 x 3
##     year medianLifeExp maxGdpPercap
##    <int>         <dbl>        <dbl>
##  1  1952          45.1      108382.
##  2  1957          48.4      113523.
##  3  1962          50.9       95458.
##  4  1967          53.8       80895.
##  5  1972          56.5      109348.
##  6  1977          59.7       59265.
##  7  1982          62.4       33693.
##  8  1987          65.8       31541.
##  9  1992          67.7       34933.
## 10  1997          69.4       41283.
## 11  2002          70.8       44684.
## 12  2007          71.9       49357.
```

Alternatively, rather than comparing across time, you might be interested in comparing among continents for only the year 1957.

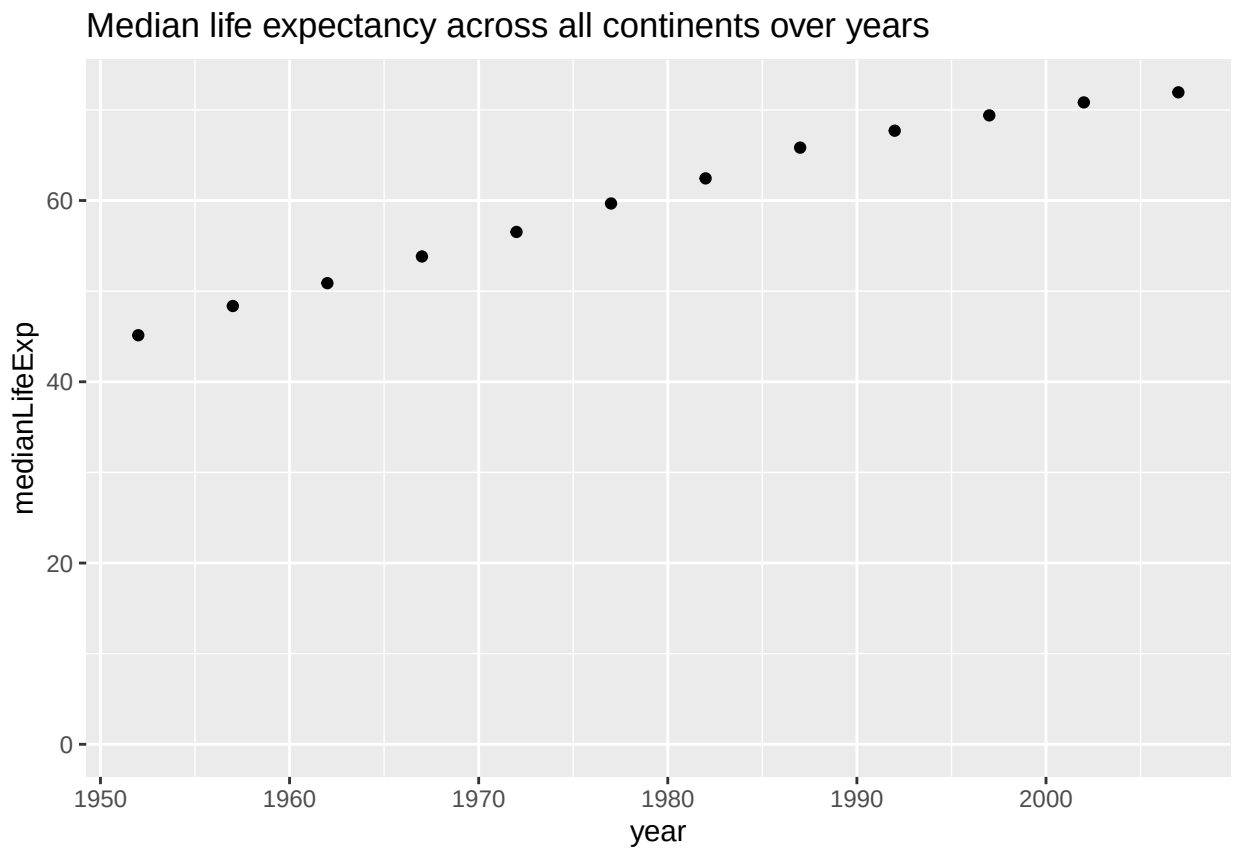
```
gapminder %>%
  filter(year == 1957) %>%
  group_by(continent) %>%
  summarize(medianLifeExp = median(lifeExp),
            maxGdpPercap = max(gdpPercap))
```

```
## # A tibble: 5 x 3
##   continent medianLifeExp maxGdpPercap
##   <fct>             <dbl>        <dbl>
## 1 Africa             40.6        5487.
## 2 Americas           56.1       14847.
## 3 Asia               48.3      113523.
## 4 Europe             67.6       17909.
## 5 Oceania            70.3       12247.
```

What if we were to find the median life expectancy and maximum GDP per capita within each combination of continent and year?

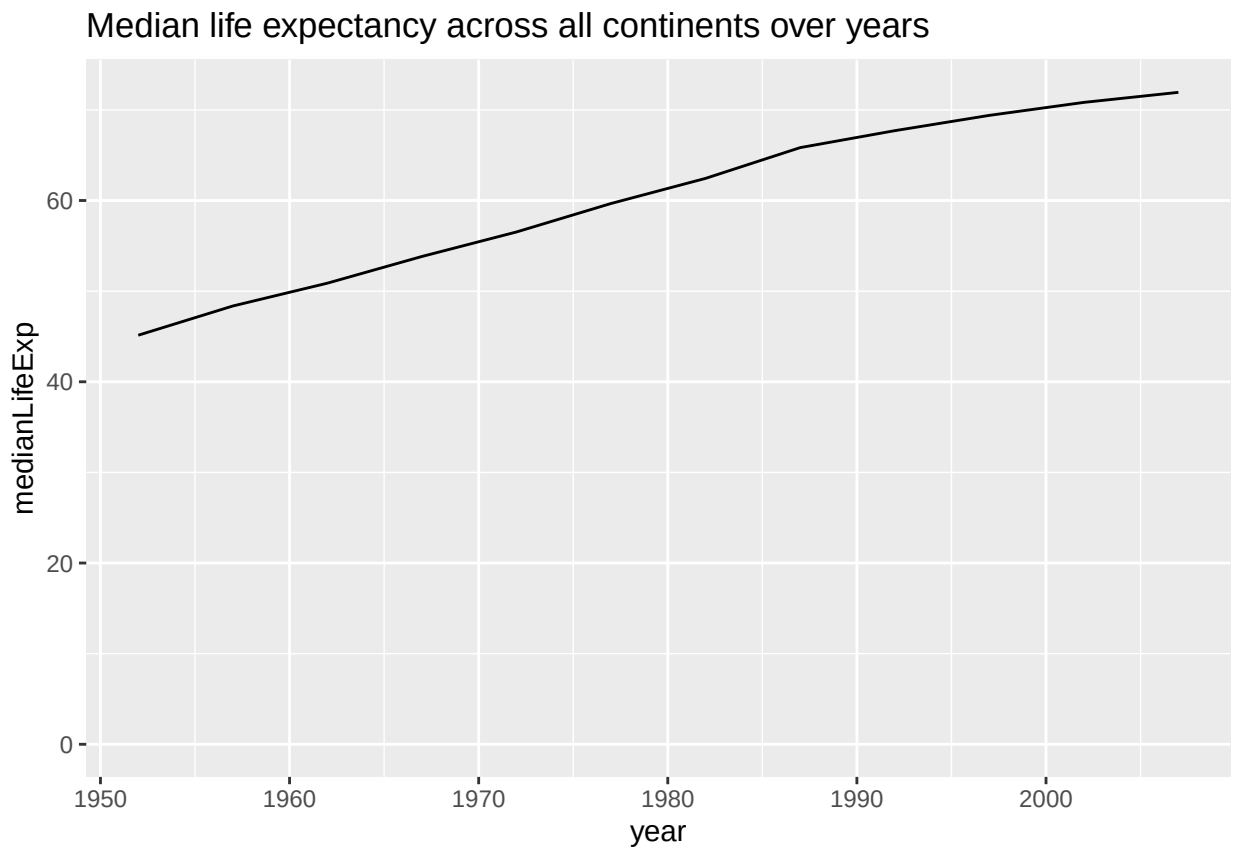
We would like to use the summarized data to create a scatter plot that shows the change of median life expectancy across all continents over time.

```
by_year <- gapminder %>%
  group_by(year) %>%
  summarize(medianLifeExp = median(lifeExp))

ggplot(by_year, aes(x=year, y=medianLifeExp)) +
  geom_point() +
  expand_limits(y = 0) + # to ensure the y-axis includes zero
  labs(
    title = "Median life expectancy across all continents over years"
    )
```



One can alternatively use a line plot to visualize this trend over time.

```
ggplot(by_year, aes(x = year, y = medianLifeExp)) +
  expand_limits(y = 0) +
  geom_line() +
  labs(
    title = "Median life expectancy across all continents over years"
    )
```

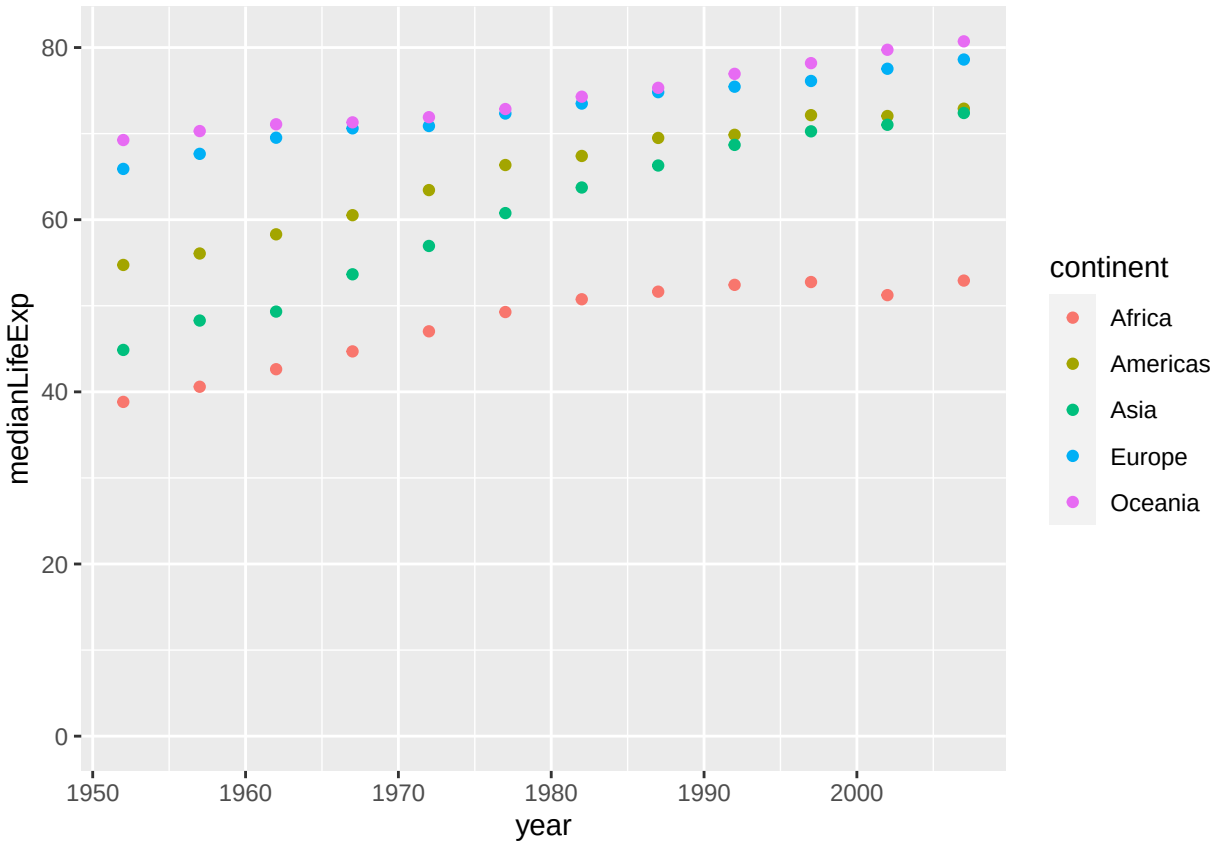


You have also learned to summarize after grouping by both year and continent, to see how the changes in median life expectancy have occurred separately within each continent. Since you now have data over time within each continent, you need a way to separate it in a visualization.

```
# Summarize medianGdpPercap within each continent within each year: by_year_continent
by_year_continent <- gapminder %>%
  group_by(continent, year) %>%
  summarize(medianLifeExp = median(lifeExp))
```

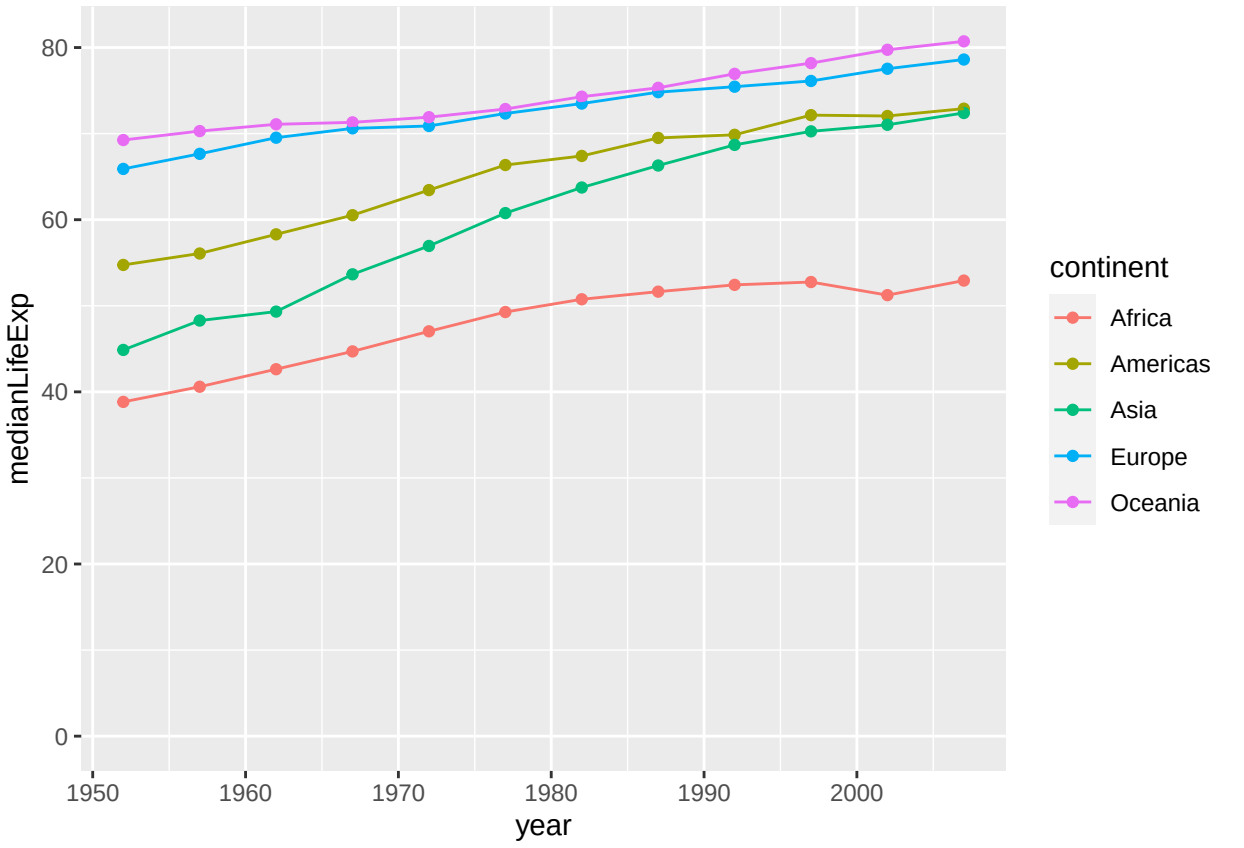
```
## `summarise()` has grouped output by 'continent'. You can override using the `.groups` argument.
```

```
# Plot the change in medianGdpPercap in each continent over time
ggplot(by_year_continent, aes(x = year, y = medianLifeExp, color = continent)) +
  geom_point() +
  expand_limits(y = 0)
```



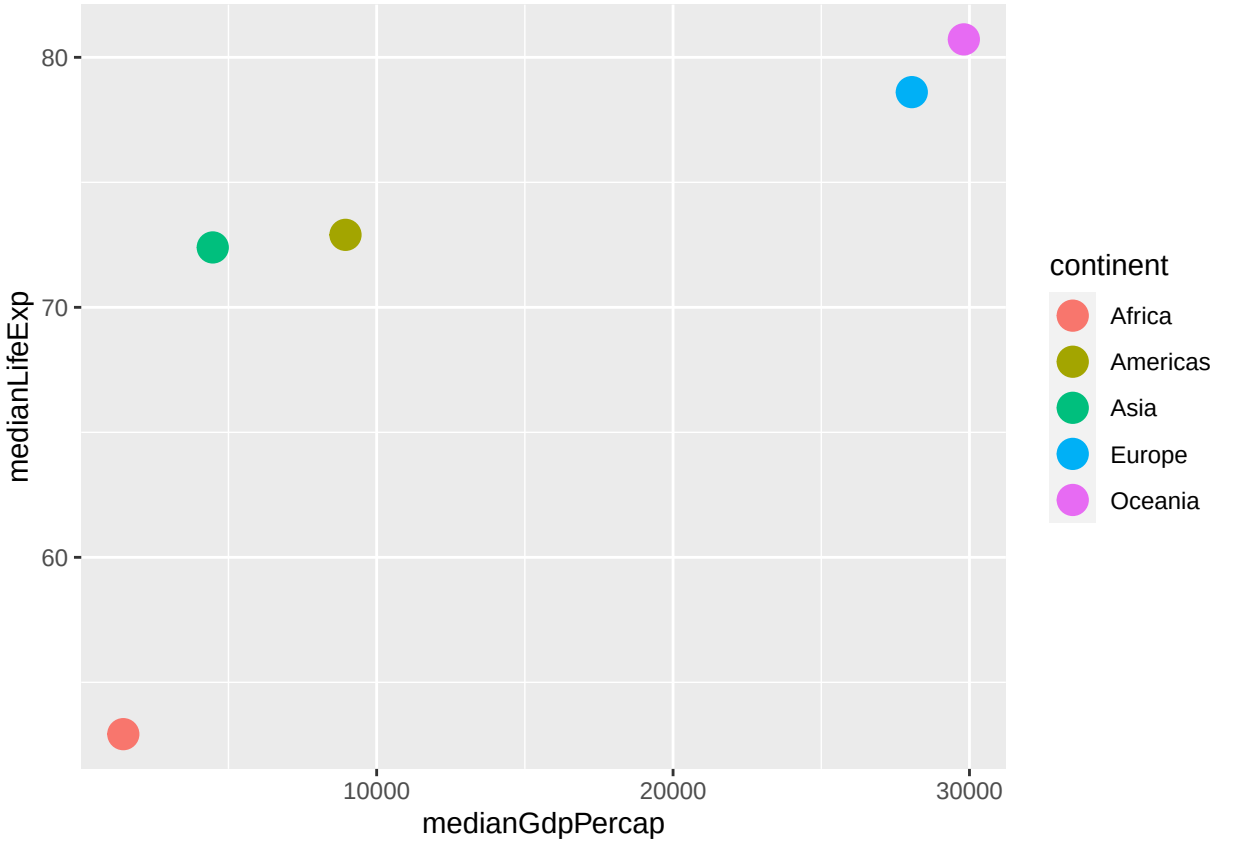
A rather pretty alternative to this is to have the data points connected, i.e., do a line plot.

```
# Plot the change in medianGdpPercap in each continent over time
ggplot(by_year_continent, aes(x = year, y = medianLifeExp, color = continent)) +
  geom_point() +
  geom_line() +
  expand_limits(y = 0)
```



Another way of exploring your data visually is to plot summarized data to compare continents w.r.t. median GDP per capita and median life expectancy withina single year, e.g., 2007.

```
# Summarize the median GDP and median life expectancy per continent in 2007
by_continent_2007 <- gapminder %>%
    filter(year == 2007) %>%
    group_by(continent) %>%
    summarize(medianGdpPercap = median(gdpPercap),medianLifeExp = median(lifeExp))

# Use a scatter plot to compare the median GDP and median life expectancy
ggplot(by_continent_2007, aes(x = medianGdpPercap, y = medianLifeExp, color = continent)) +
    geom_point(size=5)
```



Other types of visualization

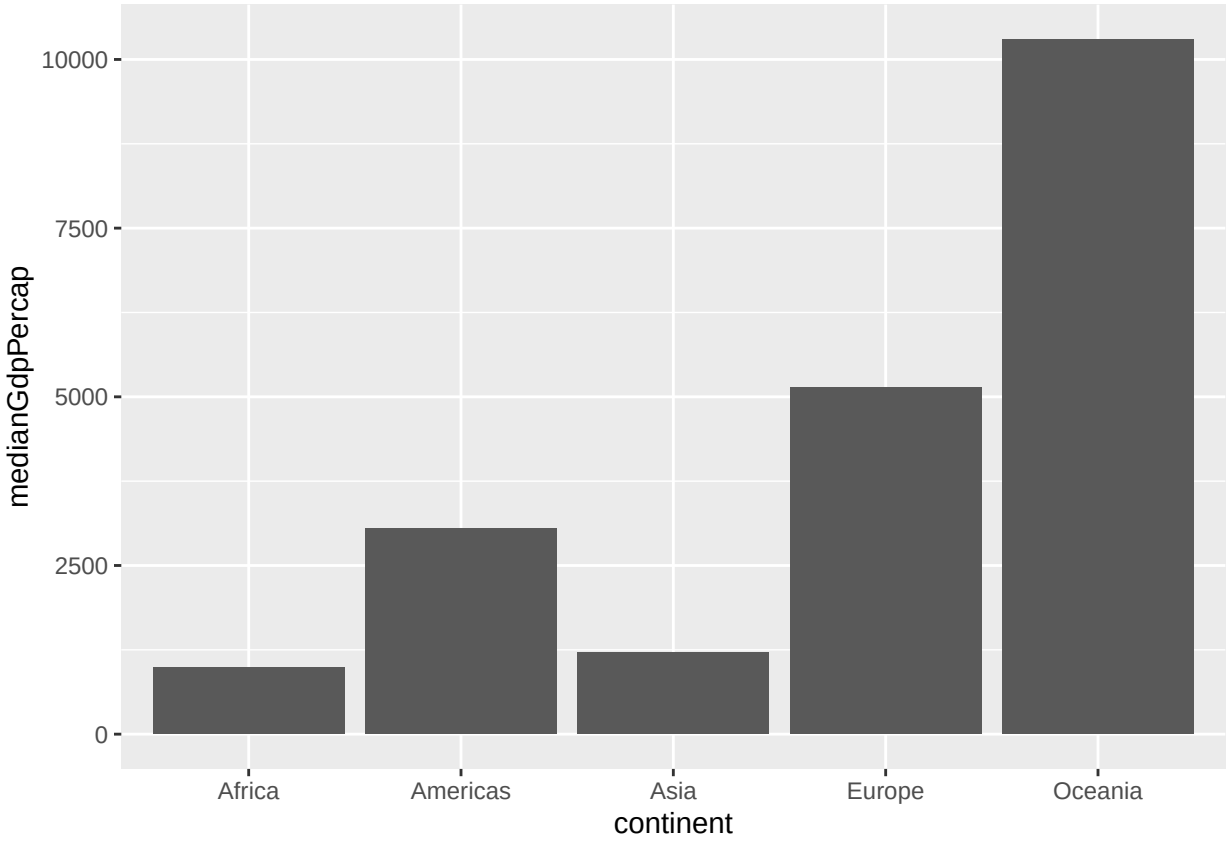
bar plot

You learned to calculate summarized values within groups. For example, the code below finds the median GDP per capita within each continent in the year 1952. That creates a table that looks like this, with one observation for each continent.

```
by_continent_1952 <- gapminder %>%
    filter(year == 1952) %>%
    group_by(continent) %>%
    summarize(medianGdpPercap = median(gdpPercap))
```

Instead of just printing the table, you might want to represent the summary visually. For that, you would use a bar plot by adding the layer `geom_col()`. Notice that bar plot is great for when you have only one value per levels of a categorical variable, in this case the continent.

```
ggplot(by_continent_1952, aes(x = continent, y = medianGdpPercap)) +
    geom_col()
```

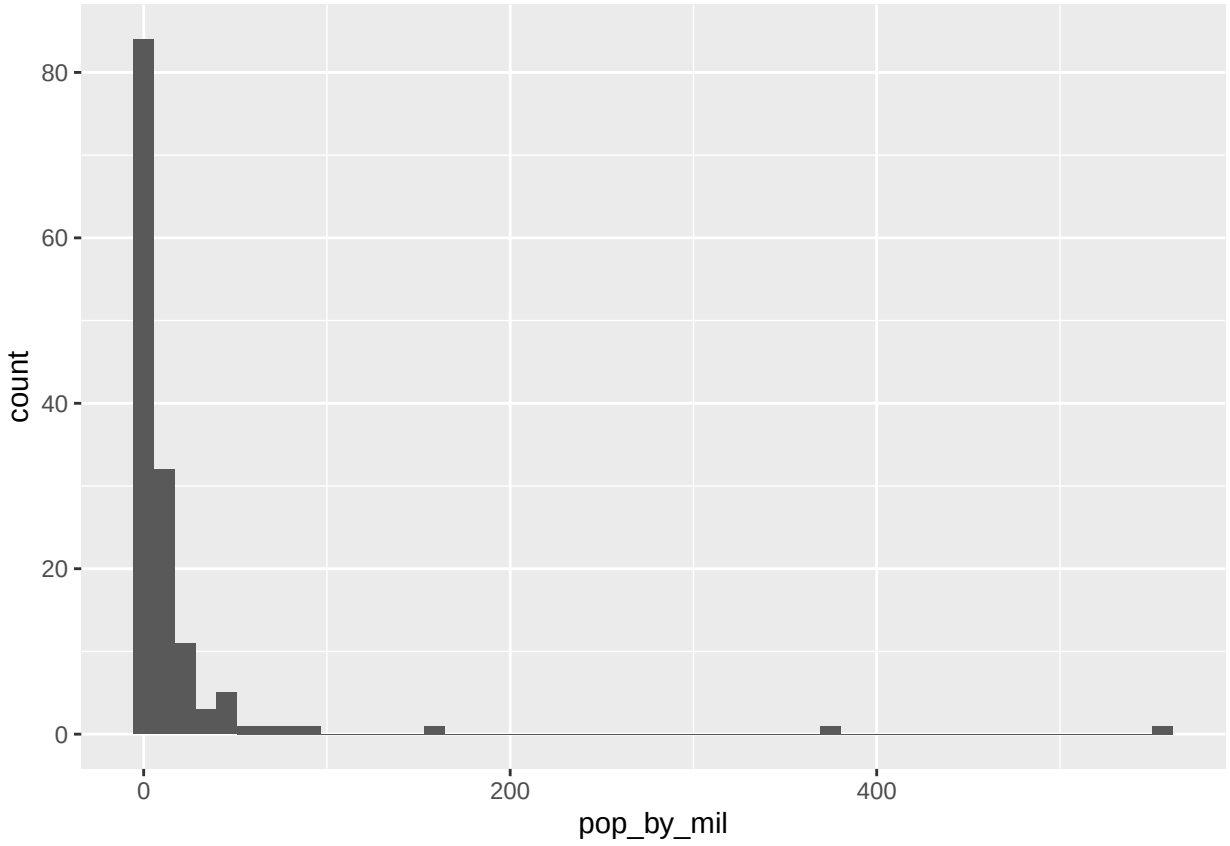


What if we were to compare the GDP per capita between the two countries in the Oceania continent?

1-D distribution plot (histogram)

A histogram is used to show the distribution of only one dimension of the data. You have used this before, where you examined the distribution of `gdpPercapita` to see if a log-scale transformation was needed for more readable visualization. Now, suppose you would want to get a sense of the distribution of the variable population (but in millions) across all countries in year 1952. A histogram is created with `geom_histogram()`.

```
gapminder_1952 <- gapminder %>%
  filter(year == 1952) %>%
  mutate(pop_by_mil = pop / 1000000)

# Create a histogram of population (pop_by_mil)
ggplot(gapminder_1952, aes(x = pop_by_mil)) +
  geom_histogram(bins = 50)
```

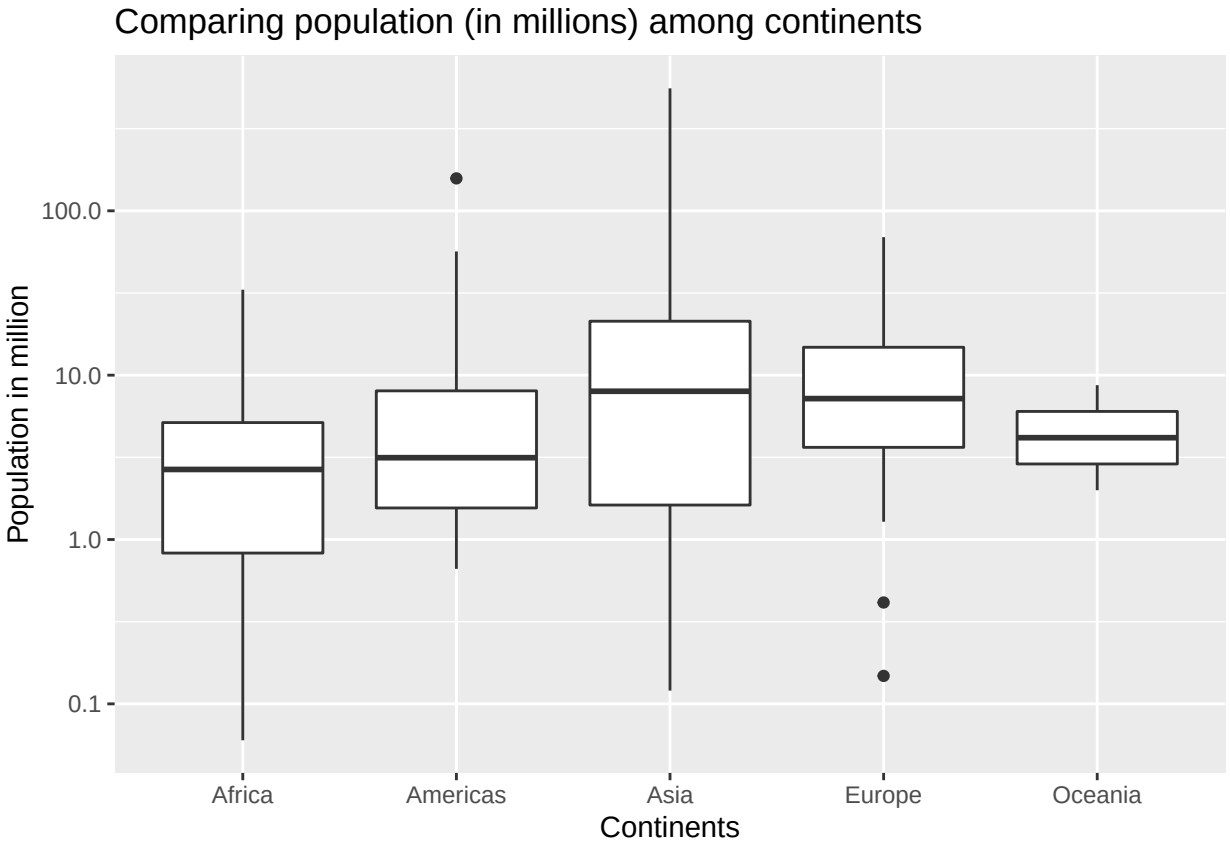


This histogram represents the distribution of the variable `pop_by_mil` across different countries in year 1952. Every bar represents a bin of population, and the height represents how many countries fall into that bin.

2-D distribution plot (box plot)

In previous section, `histogram` was used to show the distribution of population across all continents, without distinguishing them. But what if the goal is to compare the distribution of `pop_by_mil` among continents? The solution is to use box plot. One can create box plots with `geom_boxplot()` and two `aes`; `x` is the category, in this case `continent`, and `y` is the values of the target variable that one wants to compare, in this case `pop_by_mil`.

```
# Create a box plot of population (pop_by_mil) among continents
ggplot(gapminder_1952, aes(x = continent, y = pop_by_mil)) +
  geom_boxplot() +
  scale_y_log10() +
  labs(
    title = "Comparing population (in millions) among continents",
    x = 'Continents',
    y = 'Population in million'
    )
```


Comparing population (in millions) among continents
Population in million
100.0
10.0
1.0
0.1
Africa
Americas
Asia
Europe
Oceania
Continents