Introduction to the Tidyverse

Lessons from DataCamp

Neil Yetz

Contents

Introduction **2**

Course Description **2**

Chapter 1: Data wrangling **2**
- Loading the gapminder and dplyr packages . . . 2
- Understanding a data frame . . . 3
- Filtering for one year . . . 4
- Filtering for one country and one year . . . 4
- Arranging observations by life expectancy . . . 5
- Filtering and arranging . . . 5
- Using mutate to change or create a column . . . 6
- Combining filter, mutate, and arrange . . . 7

Chapter 2: Data Visualization **8**
- Variable assignment . . . 8
- Comparing population and GDP per capita . . . 8
- Comparing population and life expectancy . . . 9
- Putting the x-axis on a log scale . . . 10
- Putting the x- and y- axes on a log scale . . . 11
- Adding color to a scatter plot . . . 12
- Adding size and color to a plot . . . 13
- Creating a subgraph for each continent . . . 14
- Faceting by year . . . 15

Chapter 3: Grouping and summarizing **16**
- Summarizing the median life expectancy . . . 16
- Summarizing the median life expectancy in 1957 . . . 17
- Summarizing multiple variables in 1957 . . . 17
- Summarizing by year . . . 18
- Summarizing by continent . . . 18
- Summarizing by continent and year . . . 19
- Comparing median life expectancy and median GDP per continent in 2007 . . . 20
- Visualizing median life expectancy over time . . . 21
- Visualizing median GDP per capita per continent over time . . . 22

Chapter 4: Types of Visualization **23**
- Visualizing median GDP per capita over time . . . 23
- Visualizing median GDP per capita by continent over time . . . 24
- Visualizing median GDP per capita by continent . . . 25
- Visualizing GDP per capita by country in Oceania . . . 26
- Visualizing population . . . 27
- Visualizing population with x-axis on a log scale . . . 28
- Comparing GDP per capita across continents . . . 29
- Adding a title to your graph . . . 30

Introduction

The following documentoutlines the written portion of the lessons from DataCamp's Introduction to the Tidyverse. This works to develop skills within the tidyverse package.

As a note: All text is completely copied and pasted from the course. There are instances where the document refers to the "editor on the right", please note, that in this notebook document all of these instances are noted in the "r-chunks" (areas containing working r-code), which occurs below the text, rather than to the right. Furthermore, This lesson contained instructional videos at the begining of new concepts that are not detailed in this document. However, even without these videos, the instructions are quite clear in indicating what the code is accomplishing.

If you have this document open on "R-Notebook", simply click "run" -> "Run all" (Or just press 'ctrl + alt + r'), let the "r-chunks" run (This might take a bit of time) then click "Preview". There are 5 necessary datasets to run this program, please create an r-project with this data or set a working directory (required files names are available in the "Required data for this session" section)

This document was created by Neil Yetz on 05/17/2018. Please send any questions or concerns in this document to Neil at ndyetz@gmail.com

Course Description

This is an introduction to the programming language R, focused on a powerful set of tools known as the "tidyverse". In the course you'll learn the intertwined processes of data manipulation and visualization through 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. This gives a taste both of the value of exploratory data analysis and the power of tidyverse tools. This is a suitable introduction for people who have no previous experience in R and are interested in learning to perform data analysis.

Chapter 1: 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 lets you answer questions about your data.

Loading the gapminder and dplyr packages

Before you can work with the `gapminder` dataset, you'll need to load two R packages that contain the tools for working with it, then display the `gapminder` dataset so that you can see what it contains.

To your right, you'll see two windows inside which you can enter code: The `script.R` window, and the R Console. All of your code to solve each exercise must go inside `script.R`.

If you hit Submit Answer, your R script is executed and the output is shown in the R Console. DataCamp checks whether your submission is correct and gives you feedback. You can hit Submit Answer as often as you want. If you're stuck, you can ask for a hint or a solution.

You can use the R Console interactively by simply typing R code and hitting Enter. When you work in the console directly, your code will not be checked for correctness so it is a great way to experiment and explore.

INSTRUCTIONS

Use the `library()` function to load the `dplyr` package, just like we've loaded the `gapminder` package for you.

Type `gapminder`, on its own line, to look at the `gapminder` dataset.

```
# Load the gapminder package
#install.packages("gapminder")
library(gapminder)

# Load the dplyr package
#install.packages("dplyr")
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
```

```
# Look at the gapminder dataset
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
```

Understanding a data frame

Now that you've loaded the `gapminder` dataset, you can start examining and understanding it.

We've already loaded the `gapminder` and `dplyr` packages. Type `gapminder` in your R terminal, to the lower right, to display the object.

How many observations (rows) are in the dataset?

Possible Answers (Correct Answer is **Bolded**)

1704

6

1694

1952

Filtering for one year

The `filter` verb extracts particular observations based on a condition. In this exercise you'll filter for observations from a particular year.

INSTRUCTIONS

Add a `filter()` line after the pipe (`%>%`) to extract only the observations from the year 1957. Remember that you use `==` to compare two values.

```
library(gapminder)
library(dplyr)

# Filter the gapminder dataset for the year 1957
gapminder %>%
  filter(year == 1957)
```

```
## # 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
```

Filtering for one country and one year

You can also use the `filter()` verb to set two conditions, which could retrieve a single observation.

Just like in the last exercise, you can do this in two lines of code, starting with `gapminder %>%` and having the `filter()` on the second line. Keeping one verb on each line helps keep the code readable. Note that each time, you'll put the pipe `%>%` at the end of the first line (like `gapminder %>%`); putting the pipe at the beginning of the second line will throw an error.

```
library(gapminder)
library(dplyr)

# Filter for China in 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.
```

Arranging observations by life expectancy

You use `arrange()` to sort observations in ascending or descending order of a particular variable. In this case, you'll sort the dataset based on the `lifeExp` variable.

INSTRUCTIONS

Sort the `gapminder` dataset in ascending order of life expectancy (`lifeExp`). Sort the `gapminder` dataset in descending order of life expectancy.

```
library(gapminder)
library(dplyr)

# Sort in ascending order of lifeExp
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    30.0  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
```

```
# Sort in descending order of lifeExp
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    82.0 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
```

Filtering and arranging

You'll often need to use the pipe operator (`%>%`) to combine multiple dplyr verbs in a row. In this case, you'll combine a `filter()` with an `arrange()` to find the highest population countries in a particular year.

INSTRUCTIONS

Use `filter()` to extract observations from just the year 1957, then use `arrange()` to sort in descending order of population (`pop`).

```
library(gapminder)
library(dplyr)

# Filter for the year 1957, then arrange in descending order of population

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

```
## # A tibble: 142 x 6
##    country        continent  year lifeExp       pop gdpPercap
##    <fct>          <fct>     <int>   <dbl>     <int>     <dbl>
##  1 China          Asia       1957    50.5 637408000      576.
##  2 India          Asia       1957    40.2 409000000      590.
##  3 United States  Americas   1957    69.5 171984000    14847.
##  4 Japan          Asia       1957    65.5  91563009     4318.
##  5 Indonesia      Asia       1957    39.9  90124000      859.
##  6 Germany        Europe     1957    69.1  71019069    10188.
##  7 Brazil         Americas   1957    53.3  65551171     2487.
##  8 United Kingdom Europe     1957    70.4  51430000    11283.
##  9 Bangladesh     Asia       1957    39.3  51365468      662.
## 10 Italy          Europe     1957    67.8  49182000     6249.
## # ... with 132 more rows
```

Using mutate to change or create a column

Suppose we want life expectancy to be measured in months instead of years: you'd have to multiply the existing value by 12. You can use the `mutate()` verb to change this column, or to create a new column that's calculated this way.

INSTRUCTIONS

Use `mutate()` to change the existing `lifeExp` column, by multiplying it by 12: `12 * lifeExp`.

Use `mutate()` to add a new column, called `lifeExpMonths`, calculated as `12 * lifeExp`.

```
library(gapminder)
library(dplyr)

# Use mutate to change lifeExp to be in months
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
```

```
# Use mutate to create a new column called lifeExpMonths
gapminder %>%
  mutate(lifeExpMonths = 12 * lifeExp)
```

```
## # 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
```

Combining filter, mutate, and arrange

In this exercise, you'll combine all three of the verbs you've learned in this chapter, to find the countries with the highest life expectancy, in months, in the year 2007.

INSTRUCTIONS

- In one sequence of pipes on the gapminder dataset:
- `filter()` for observations from the year 2007,
- `mutate()` to create a column `lifeExpMonths`, calculated as `12 * lifeExp`, and
- `arrange()` in descending order of that new column

```
library(gapminder)
library(dplyr)

# Filter, mutate, and arrange the gapminder dataset
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 1.27e8    31656.          991.
##  2 Hong Kong, China Asia       2007    82.2 6.98e6    39725.          986.
##  3 Iceland          Europe     2007    81.8 3.02e5    36181.          981.
##  4 Switzerland      Europe     2007    81.7 7.55e6    37506.          980.
##  5 Australia        Oceania    2007    81.2 2.04e7    34435.          975.
```

```
##  6 Spain            Europe     2007    80.9 4.04e7    28821.          971.
##  7 Sweden           Europe     2007    80.9 9.03e6    33860.          971.
##  8 Israel           Asia       2007    80.7 6.43e6    25523.          969.
##  9 France           Europe     2007    80.7 6.11e7    30470.          968.
## 10 Canada           Americas   2007    80.7 3.34e7    36319.          968.
## # ... with 132 more rows
```

Chapter 2: Data Visualization

You've already been able to answer some questions about the data through dplyr, but you've engaged with them just as a table (such as one showing the life expectancy in the US each year). Often a better way to understand and present such data is as a graph. Here you'll learn the essential skill of data visualization, using the ggplot2 package. Visualization and maniuplation are often intertwined, so you'll see how the dplyr and ggplot2 packages work closely together to create informative graphs.

Variable assignment

Throughout the exercises in this chapter, you'll be visualizing a subset of the gapminder data from the year 1952. First, you'll have to load the ggplot2 package, and create a `gapminder_1952` dataset to visualize.

INSTRUCTIONS

Load the `ggplot2` package after the `gapminder` and `dplyr` packages. Filter `gapminder` for observations from the year 1952, and assign it to a new dataset `gapminder_1952` using the assignment operator (`<-`).

```
# Load the ggplot2 package as well
library(gapminder)
library(dplyr)
library(ggplot2)

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

Comparing population and GDP per capita

In the video you learned to create a scatter plot with GDP per capita on the x-axis and life expectancy on the y-axis (the code for that graph is shown here). When you're exploring data visually, you'll often need to try different combinations of variables and aesthetics.

INSTRUCTIONS

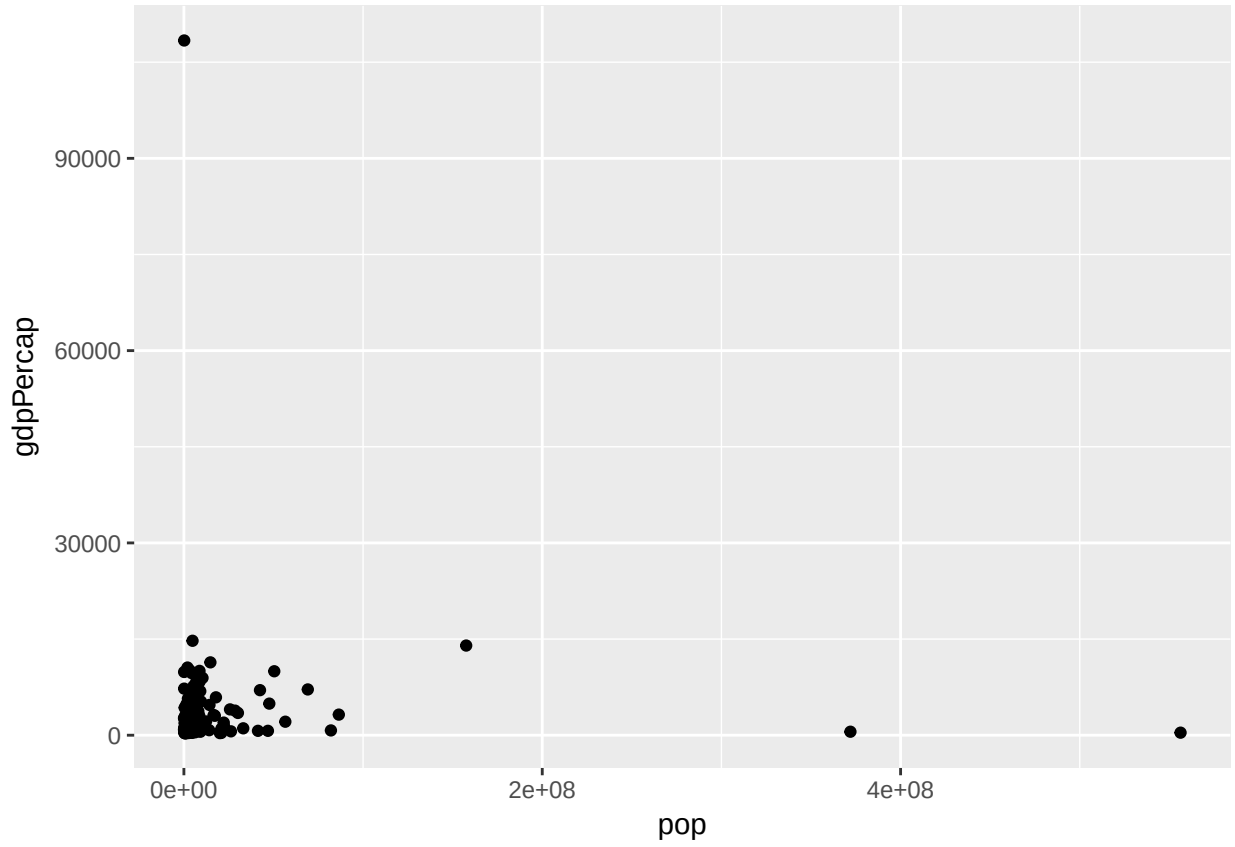
Change the scatter plot of `gapminder_1952` so that (`pop`) is on the x-axis and GDP per capita (`gdpPercap`) is on the y-axis.

```
library(gapminder)
library(dplyr)
library(ggplot2)

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

# Change to put pop on the x-axis and gdpPercap on the y-axis
```

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



Comparing population and life expectancy

In this exercise, you'll use `ggplot2` to create a scatter plot from scratch, to compare each country's population with its life expectancy in the year 1952.

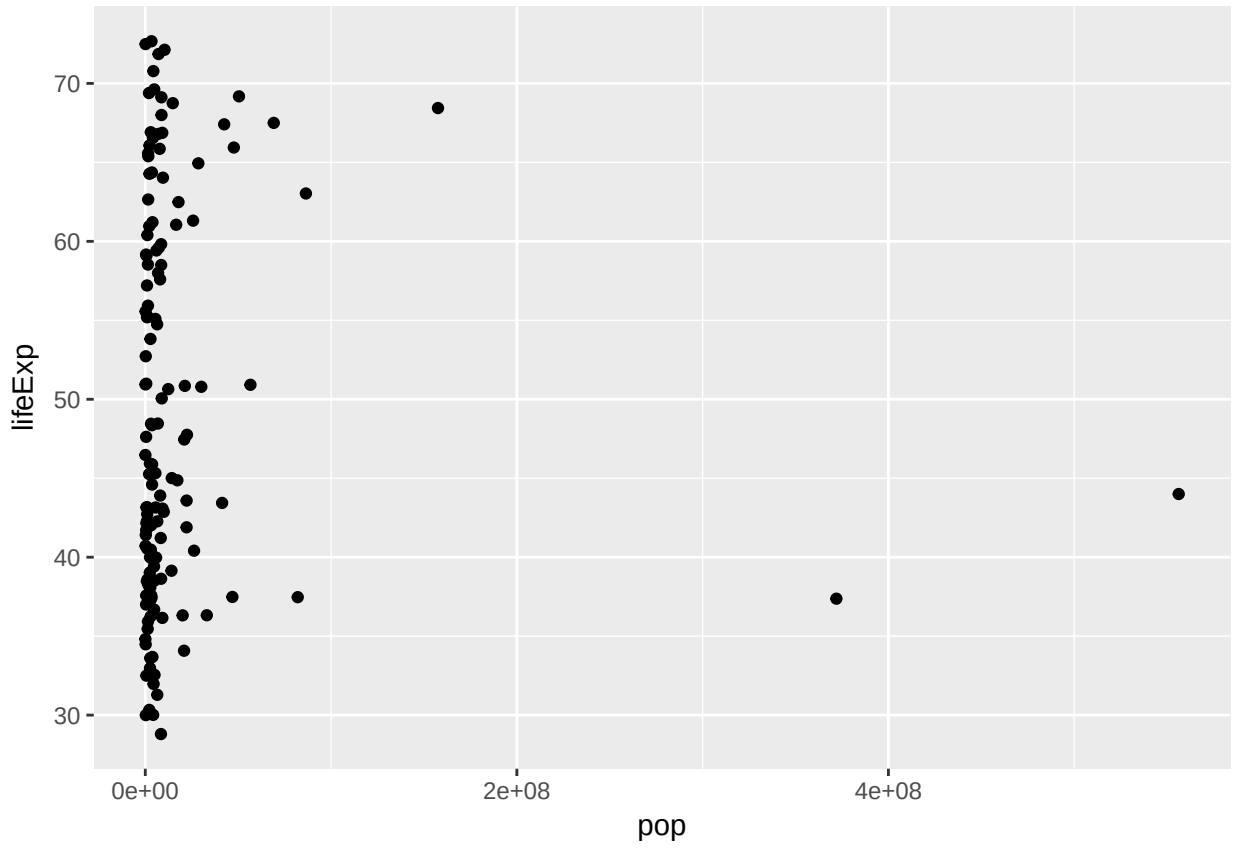
INSTRUCTIONS

Create a scatter plot of `gapminder_1952` with population (`pop`) is on the x-axis and life expectancy (`lifeExp`) on the y-axis.

```
library(gapminder)
library(dplyr)
library(ggplot2)

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

# Create a scatter plot with pop on the x-axis and lifeExp on the y-axis
ggplot(gapminder_1952, aes(x = pop, y = lifeExp)) +
  geom_point()
```



Putting the x-axis on a log scale

You previously created a scatter plot with population on the x-axis and life expectancy on the y-axis. Since population is spread over several orders of magnitude, with some countries having a much higher population than others, it's a good idea to put the x-axis on a log scale.

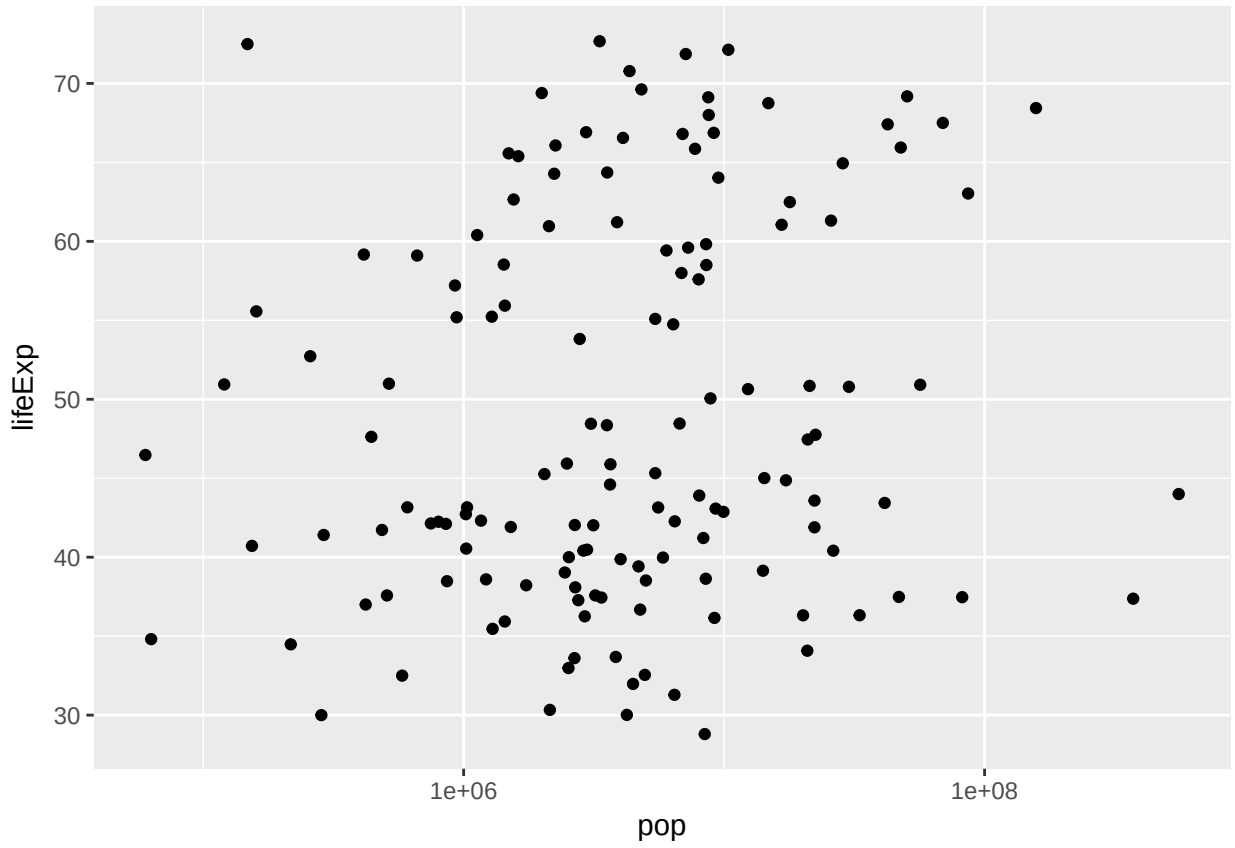
INSTRUCTIONS

Change the existing scatter plot (code provided) to put the x-axis (representing population) on a log scale.

```
library(gapminder)
library(dplyr)
library(ggplot2)

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

# Change this plot to put the x-axis on a log scale
ggplot(gapminder_1952, aes(x = pop, y = lifeExp)) +
  geom_point() +
  scale_x_log10()
```



Putting the x- and y- axes on a log scale

Suppose you want to create a scatter plot with population on the x-axis and GDP per capita on the y-axis. Both population and GDP per-capita are better represented with log scales, since they vary over many orders of magnitude.

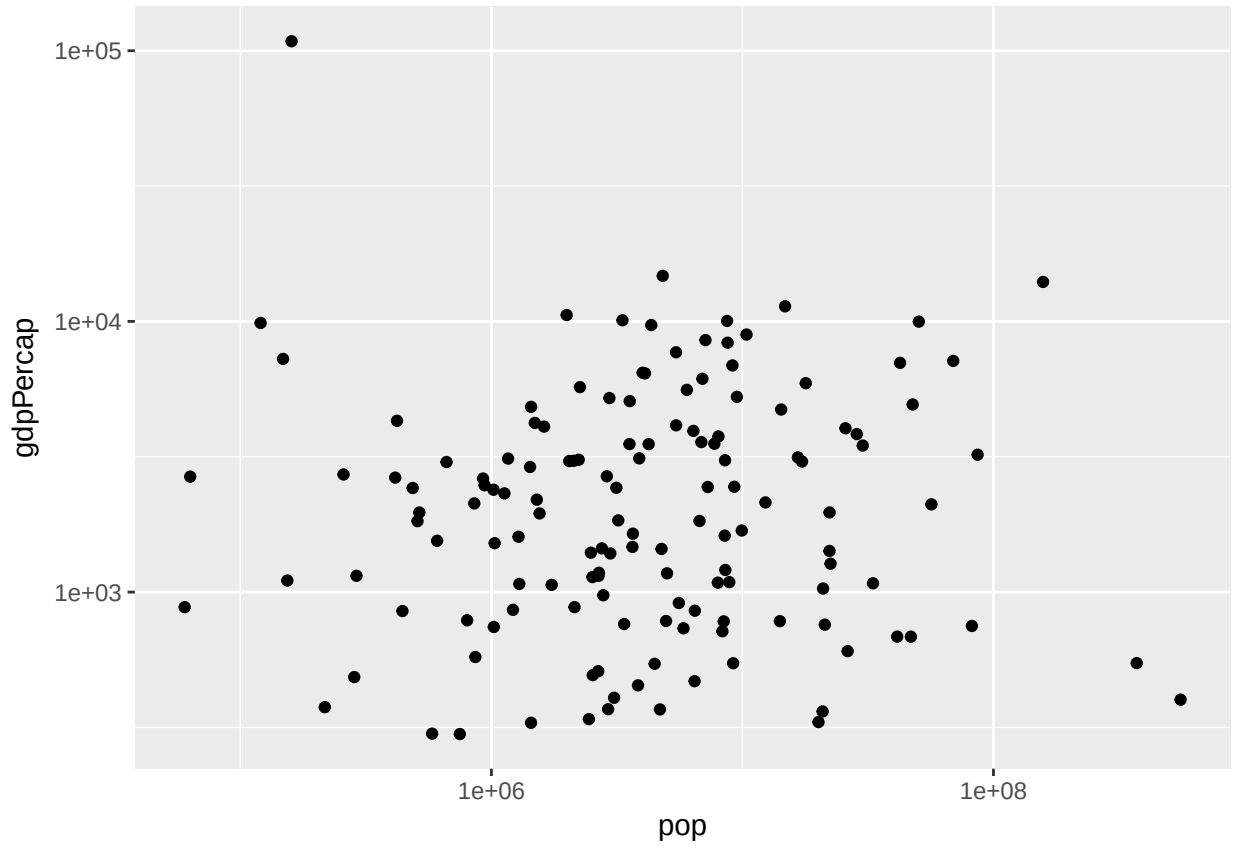
INSTRUCTIONS

Create a scatter plot with population (`pop`) on the x-axis and GDP per capita (`gdpPercap`) on the y-axis. Put both the x- and y- axes on a log scale.

```
library(gapminder)
library(dplyr)
library(ggplot2)

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

# Scatter plot comparing pop and gdpPercap, with both axes on a log scale
ggplot(gapminder_1952, aes(x = pop, y = gdpPercap)) +
  geom_point() +
  scale_x_log10() +
  scale_y_log10()
```



Adding color to a scatter plot

In this lesson you learned how to use the color aesthetic, which can be used to show which continent each point in a scatter plot represents.

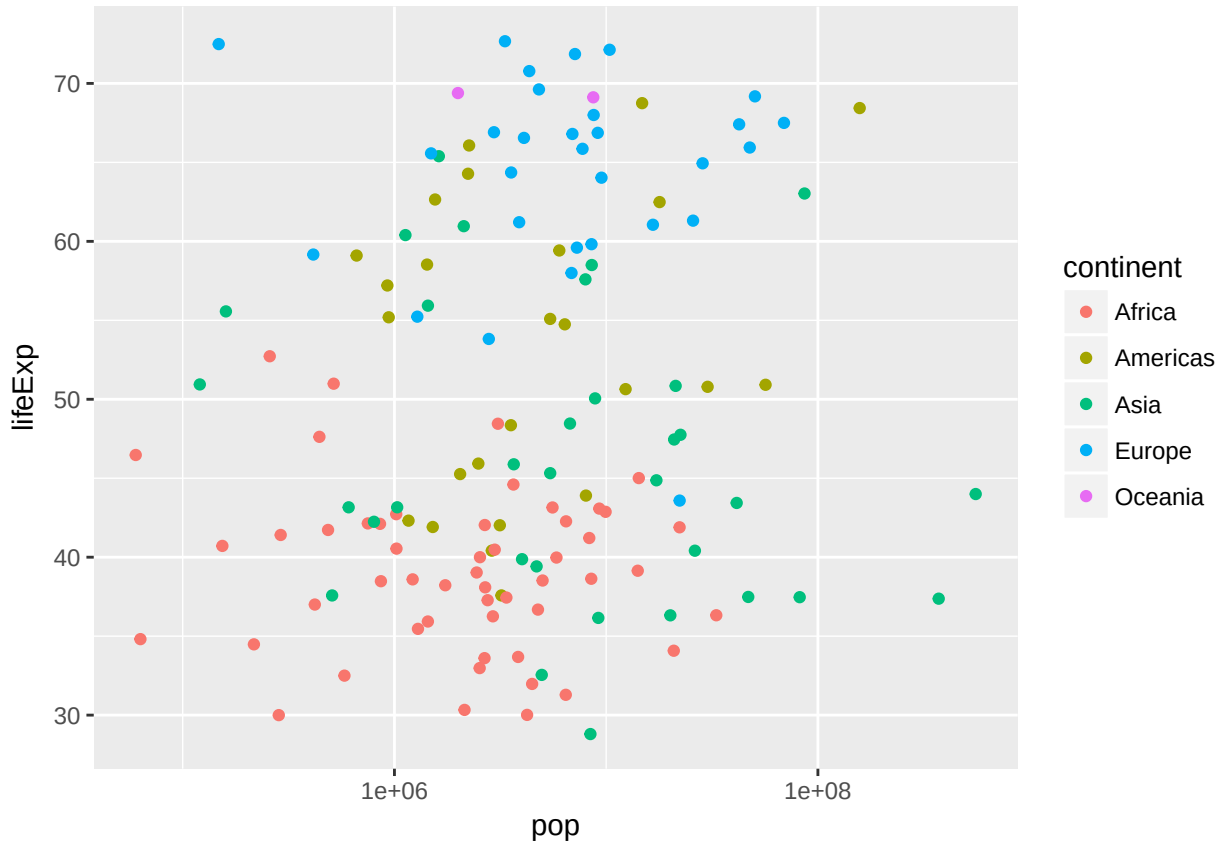
INSTRUCTIONS

Create a scatter plot with population (`pop`) on the x-axis, life expectancy (`lifeExp`) on the y-axis, and with continent (`continent`) represented by the color of the points. Put the x-axis on a log scale.

```
library(gapminder)
library(dplyr)
library(ggplot2)

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

# Scatter plot comparing pop and lifeExp, with color representing continent
ggplot(gapminder_1952, aes(x = pop, y = lifeExp, color = continent)) +
  geom_point() +
  scale_x_log10()
```



Adding size and color to a plot

In the last exercise, you created a scatter plot communicating information about each country's population, life expectancy, and continent. Now you'll use the size of the points to communicate even more.

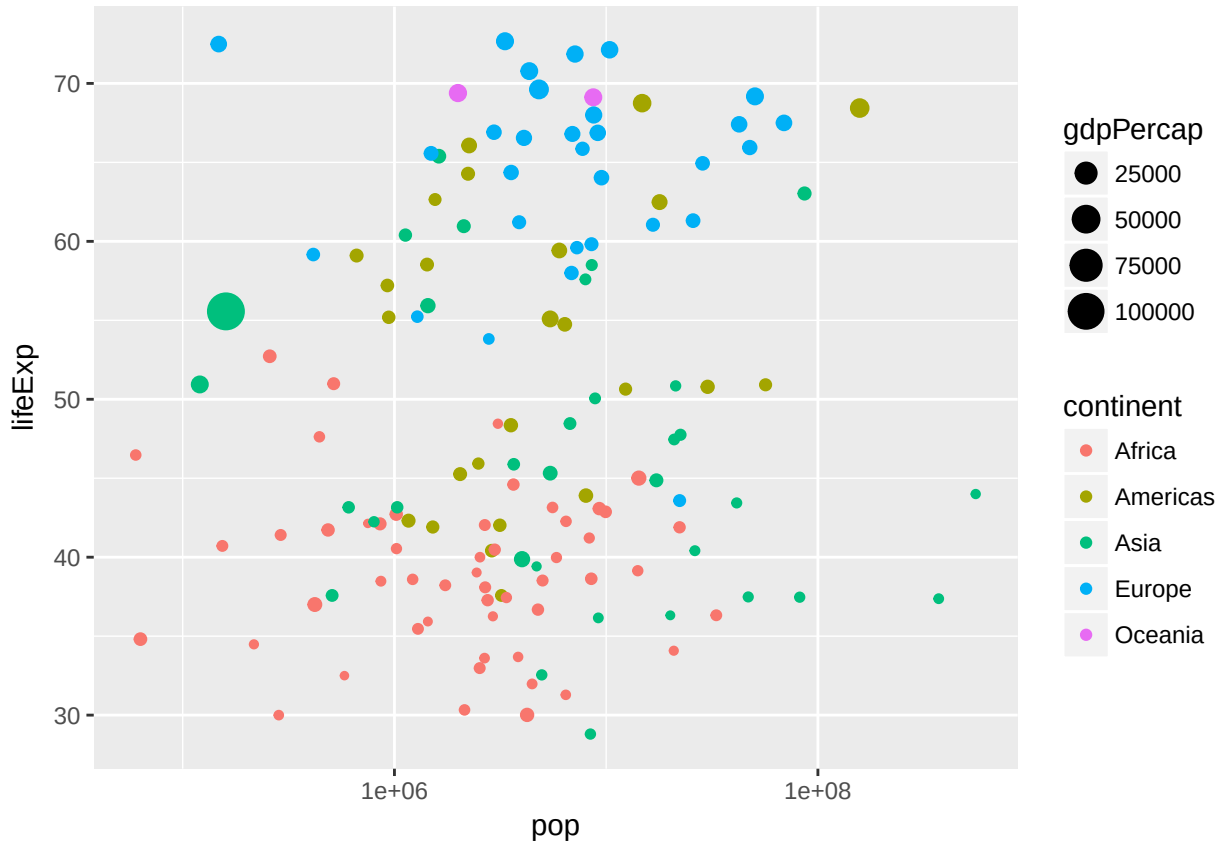
INSTRUCTIONS

Modify the scatter plot so that the size of the points represents each country's GDP per capita (`gdpPercap`).

```
library(gapminder)
library(dplyr)
library(ggplot2)

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

# Add the size aesthetic to represent a country's gdpPercap
ggplot(gapminder_1952, aes(x = pop, y = lifeExp, color = continent,
                           size = gdpPercap)) +
  geom_point() +
  scale_x_log10()
```



Creating a subgraph for each continent

You've learned to use faceting to divide a graph into subplots based on one of its variables, such as the continent.

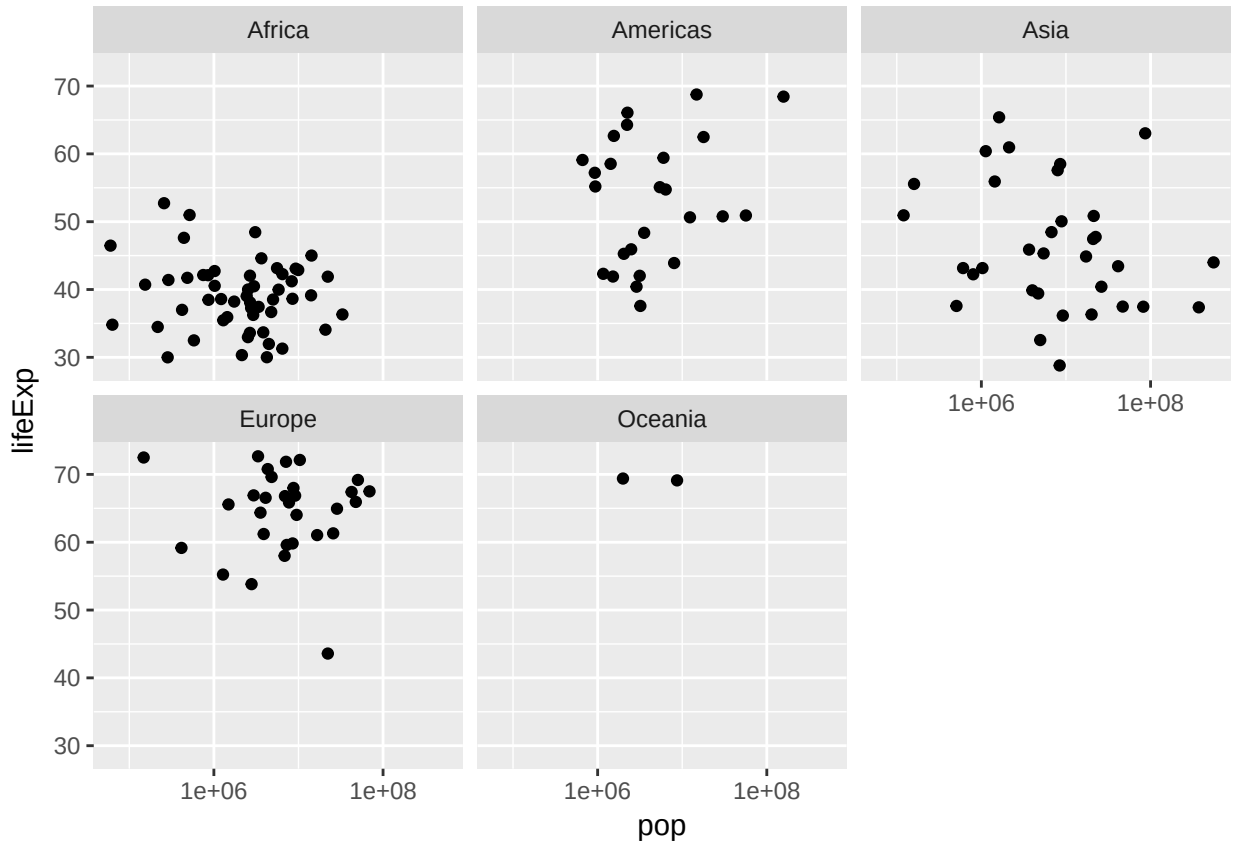
INSTRUCTIONS

Create a scatter plot of `gapminder_1952` with the x-axis representing population (`pop`), the y-axis representing life expectancy (`lifeExp`), and faceted to have one subplot per continent (`continent`). Put the x-axis on a log scale.

```
library(gapminder)
library(dplyr)
library(ggplot2)

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

# Scatter plot comparing pop and lifeExp, faceted by continent
ggplot(gapminder_1952, aes(x = pop, y = lifeExp)) +
  geom_point() +
  scale_x_log10()+
  facet_wrap(~ continent)
```



Faceting by year

All of the graphs in this chapter have been visualizing statistics within one year. Now that you're able to use faceting, however, you can create a graph showing all the country-level data from 1952 to 2007, to understand how global statistics have changed over time.

INSTRUCTIONS

Create a scatter plot of the gapminder data:

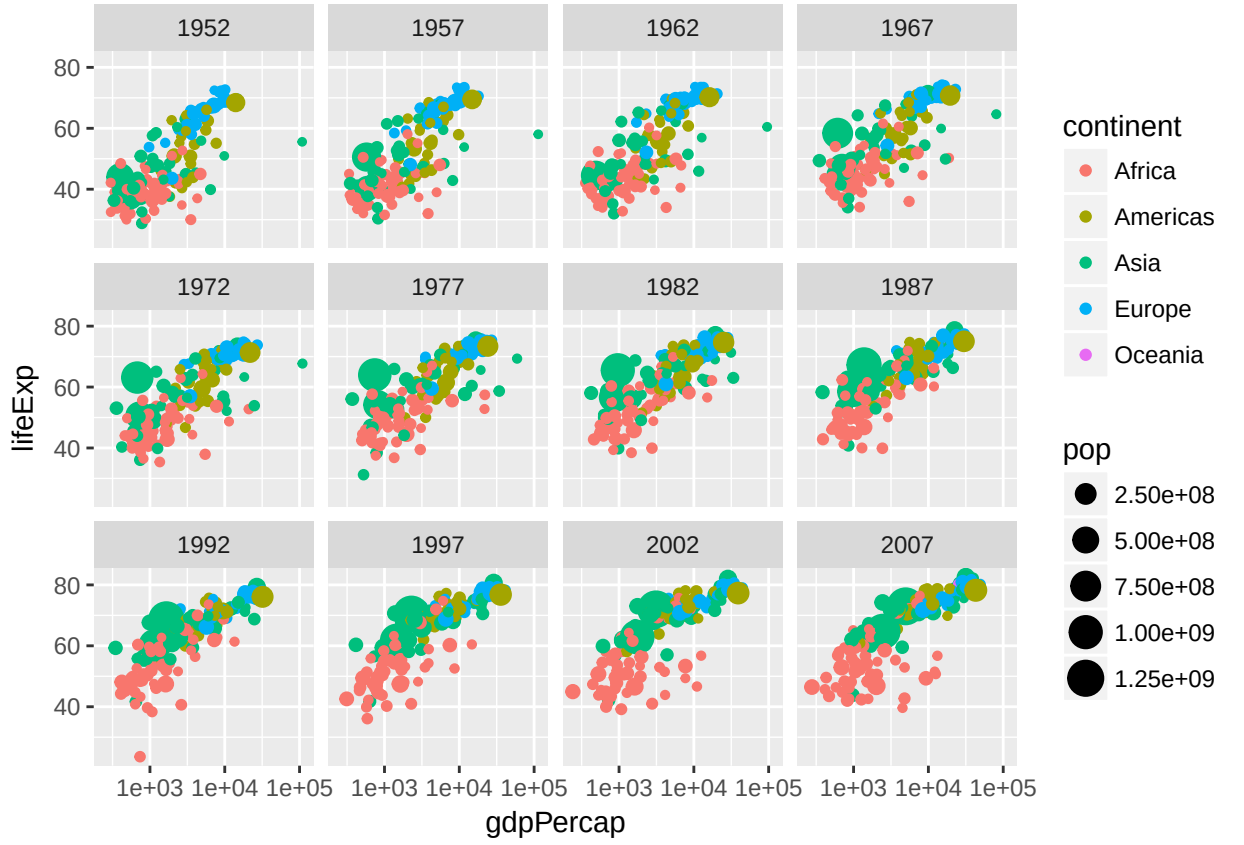
- Put GDP per capita (`gdpPercap`) on the x-axis and life expectancy (`lifeExp`) on the y-axis, with continent (`continent`) represented by color and population (`pop`) represented by size.
- Put the x-axis on a log scale
- Facet by the `year` variable

```
library(gapminder)
library(dplyr)
library(ggplot2)

# Scatter plot comparing gdpPercap and lifeExp, with color representing continent
# and size representing population, faceted by year

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

```
  scale_x_log10() +
  facet_wrap(~ year)
```



Chapter 3: Grouping and summarizing

So far you've been answering questions about individual country-year pairs, but we 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.

Summarizing the median life expectancy

You've seen how to find the mean life expectancy and the total population across a set of observations, but `mean()` and `sum()` are only two of the functions R provides for summarizing a collection of numbers. Here, you'll learn to use the `median()` function in combination with `summarize()`.

By the way, `dplyr` displays some messages when it's loaded that we've been hiding so far. They'll show up in red and start with:

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

```
The following objects are masked from 'package:stats':
```

This will occur in future exercises each time you load `dplyr`: it's mentioning some built-in functions that are overwritten by `dplyr`. You won't need to worry about this message within this course.

INSTRUCTIONS

Use the `median()` function within a `summarize()` to find the median life expectancy. Save it into a column called `medianLifeExp`.

```
library(gapminder)
library(dplyr)

# Summarize to find the median life expectancy
gapminder %>%
  summarize(medianLifeExp = median(lifeExp))
```

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

Summarizing the median life expectancy in 1957

Rather than summarizing the entire dataset, you may want to find the median life expectancy for only one particular year. In this case, you'll find the median in the year 1957.

INSTRUCTIONS

Filter for the year 1957, then use the `median()` function within a `summarize()` to calculate the median life expectancy into a column called `medianLifeExp`

.

```
library(gapminder)
library(dplyr)

# Filter for 1957 then summarize the median life expectancy
gapminder %>%
  filter(year == 1957) %>%
  summarize(medianLifeExp = median(lifeExp))
```

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

Summarizing multiple variables in 1957

The `summarize()` verb allows you to summarize multiple variables at once. In this case, you'll use the `median()` function to find the median life expectancy and the `max()` function to find the maximum GDP per capita.

INSTRUCTIONS

Find both the median life expectancy (`lifeExp`) and the maximum GDP per capita (`gdpPercap`) in the year 1957, calling them `medianLifeExp` and `maxGdpPercap` respectively. You can use the `max()` function to find the maximum.

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

```
# Filter for 1957 then summarize the median life expectancy and the maximum GDP per capita
gapminder %>%
  filter(year == 1957) %>%
  summarize(medianLifeExp = median(lifeExp), maxGdpPercap = max(gdpPercap))
```

```
## # A tibble: 1 x 2
##   medianLifeExp maxGdpPercap
##           <dbl>        <dbl>
## 1          48.4      113523.
```

Summarizing by year

In a previous exercise, you found the median life expectancy and the maximum GDP per capita in the year 1957. Now, you'll perform those two summaries within each year in the dataset, using the `group_by` verb.

INSTRUCTIONS

Find the median life expectancy (`lifeExp`) and maximum GDP per capita (`gdpPercap`) within each year, saving them into `medianLifeExp` and `maxGdpPercap`, respectively.

```
library(gapminder)
library(dplyr)

# Find median life expectancy and maximum GDP per capita in each year
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.
```

Summarizing by continent

You can group by any variable in your dataset to create a summary. Rather then comparing across time, you might be interested in comparing among continents. You'll want to do that within one year of the dataset: let's use 1957.

INSTRUCTIONS

Filter the `gapminder` data for the year 1957. Then find the median life expectancy (`lifeExp`) and maximum GDP per capita (`gdpPercap`) within each continent, saving them into `medianLifeExp` and `maxGdpPercap`, respectively.

```
library(gapminder)
library(dplyr)

# Find median life expectancy and maximum GDP per capita in each continent in 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.
```

Summarizing by continent and year

Instead of grouping just by year, or just by continent, you'll now group by both continent and year to summarize within each.

INSTRUCTIONS

Find the median life expectancy (`lifeExp`) and maximum GDP per capita (`gdpPercap`) within each combination of continent and year, saving them into `medianLifeExp` and `maxGdpPercap`, respectively.

```
library(gapminder)
library(dplyr)

# Find median life expectancy and maximum GDP per capita in each year/continent combination
gapminder %>%
  group_by(continent, year) %>%
  summarize(medianLifeExp = median(lifeExp), maxGdpPercap = max(gdpPercap))
```

```
## # A tibble: 60 x 4
## # Groups:   continent [?]
##    continent  year medianLifeExp maxGdpPercap
##    <fct>     <int>         <dbl>        <dbl>
##  1 Africa     1952          38.8        4725.
##  2 Africa     1957          40.6        5487.
##  3 Africa     1962          42.6        6757.
##  4 Africa     1967          44.7       18773.
##  5 Africa     1972          47.0       21011.
##  6 Africa     1977          49.3       21951.
##  7 Africa     1982          50.8       17364.
##  8 Africa     1987          51.6       11864.
##  9 Africa     1992          52.4       13522.
## 10 Africa     1997          52.8       14723.
## # ... with 50 more rows
```

Comparing median life expectancy and median GDP per continent in 2007

In these exercises you've generally created plots that show change over time. But as another way of exploring your data visually, you can also use ggplot2 to plot summarized data to compare continents within a single year.

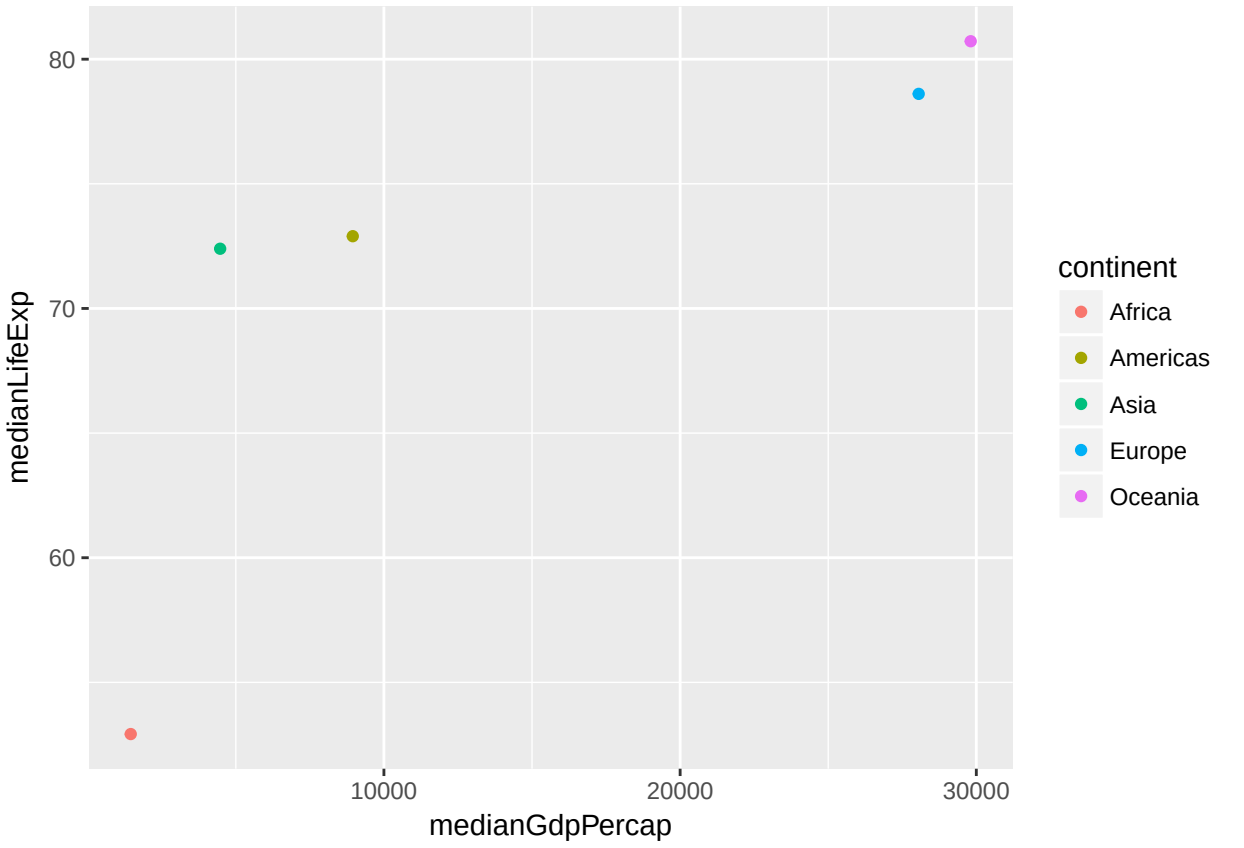
INSTRUCTIONS

- Filter the gapminder dataset for the year 2007, then summarize the median GDP per capita and the median life expectancy within each continent, into columns called `medianLifeExp` and `medianGdpPercap`. Save this as `by_continent_2007`.
- Use the `by_continent_2007` data to create a scatterplot comparing these summary statistics for continents in 2007, putting the median GDP per capita on the x-axis to the median life expectancy on the y-axis. Color the scatter plot by `continent`. You don't need to add `expand_limits(y = 0)` for this plot.

```
library(gapminder)
library(dplyr)
library(ggplot2)

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

# 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()
```



Visualizing median life expectancy over time

In the last chapter, you summarized the gapminder data to calculate the median life expectancy within each year. This code is provided for you, and is saved (with `<-`) as the `by_year` dataset.

Now you can use the ggplot2 package to turn this into a visualization of changing life expectancy over time.

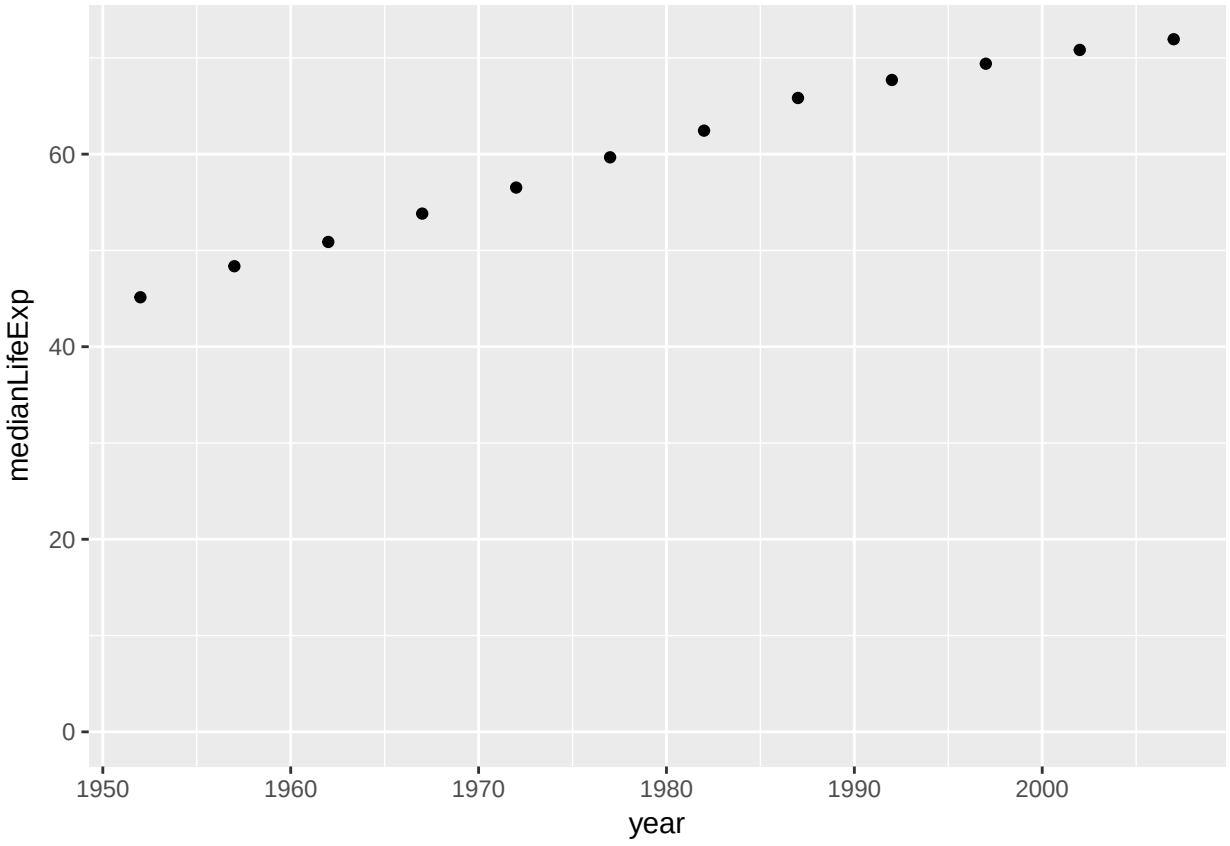
INSTRUCTIONS

Use the `by_year` dataset to create a scatter plot showing the change of median life expectancy over time, with `year` on the x-axis and `medianLifeExp` on the y-axis. Be sure to add `expand_limits(y = 0)` to make sure the plot's y-axis includes zero.

```
library(gapminder)
library(dplyr)
library(ggplot2)

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

# Create a scatter plot showing the change in medianLifeExp over time
ggplot(by_year, aes(x = year, y = medianLifeExp)) +
  geom_point() +
  expand_limits(y = 0)
```



Visualizing median GDP per capita per continent over time

In the last exercise you were able to see how the median life expectancy of countries changed over time. Now you'll examine the median GDP per capita instead, and see how the trend differs among continents.

INSTRUCTIONS

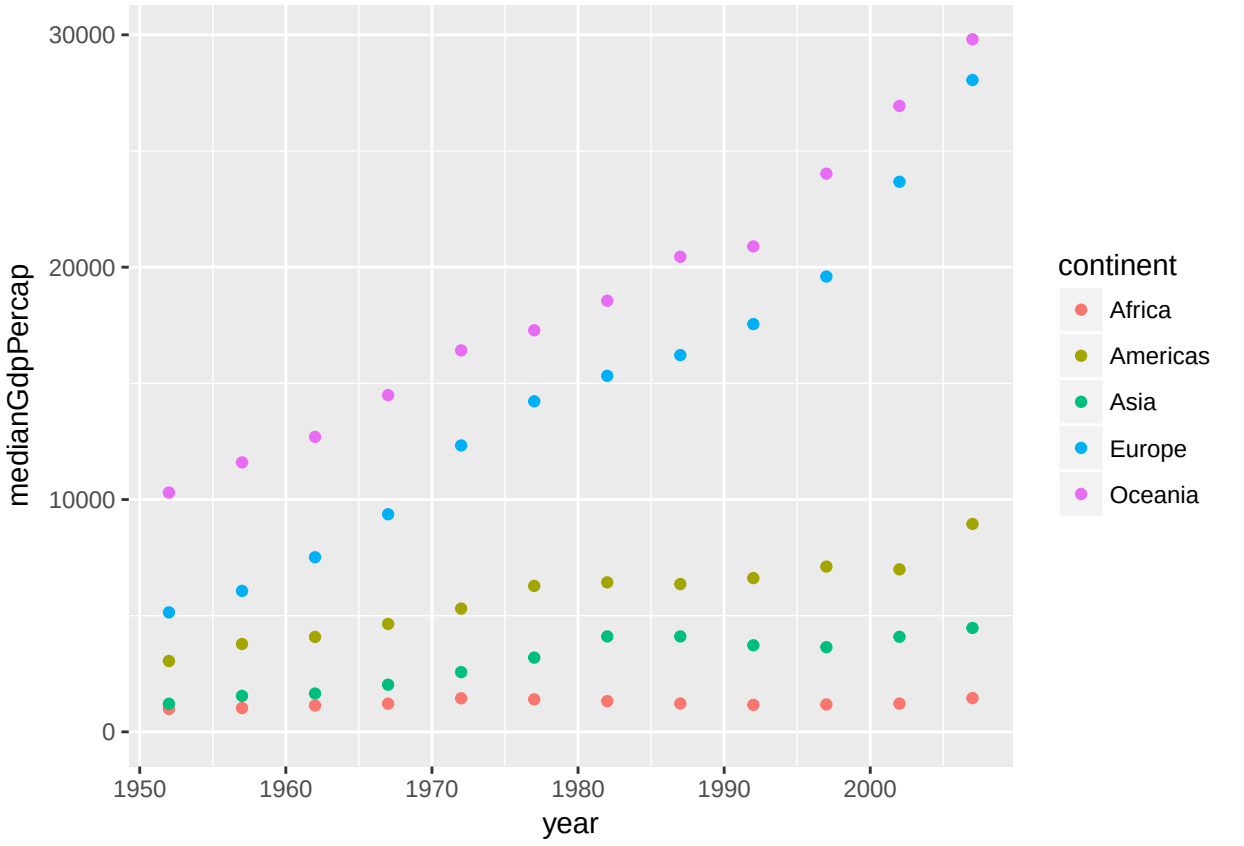
Summarize the gapminder dataset by continent and year, finding the median GDP per capita (`medianGdpPercap`) within each and putting it into a column called `medianGdpPercap`. Use the assignment operator `<-` to save this summarized data as `by_year_continent`.

Create a scatter plot showing the change in `medianGdpPercap` by continent over time. Use color to distinguish between continents, and be sure to add `expand_limits(y = 0)` so that the y-axis starts at zero.

```
library(gapminder)
library(dplyr)
library(ggplot2)

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

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



Chapter 4: Types of Visualization

You've learned to create scatter plots with ggplot2. In this chapter you'll learn to create line plots, bar plots, histograms, and boxplots. You'll see how each plot needs different kinds of data manipulation to prepare for it, and understand the different roles of each of these plot types in data analysis.

Visualizing median GDP per capita over time

A line plot is useful for visualizing trends over time. In this exercise, you'll examine how the median GDP per capita has changed over time.

INSTRUCTIONS

Use `group_by()` and `summarize()` to find the median GDP per capita within each year, calling the output column `medianGdpPercap`. Use the assignment operator `<-` to save it to a dataset called `by_year`.

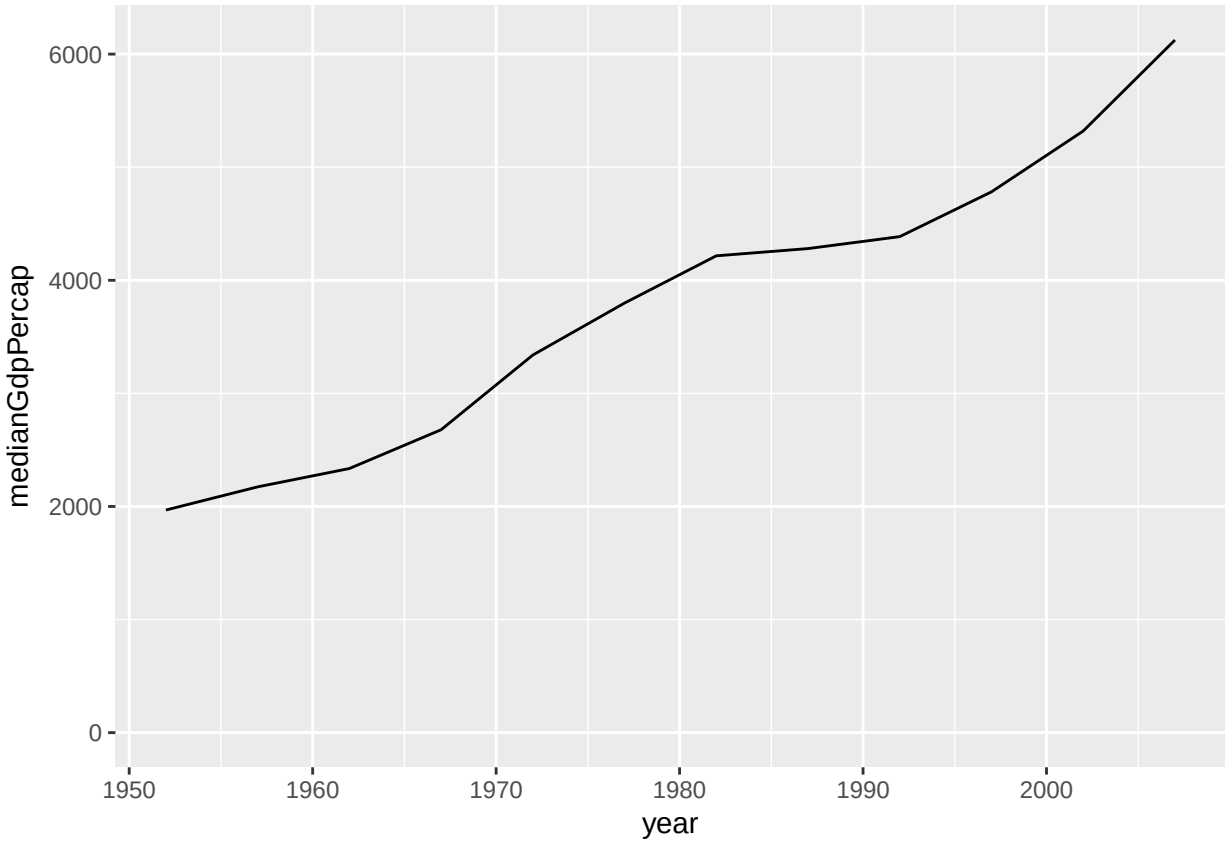
Use the `by_year` dataset to create a line plot showing the change in median GDP per capita over time. Be sure to use `expand_limits(y = 0)` to include 0 on the y-axis.

```
library(gapminder)
library(dplyr)
library(ggplot2)

# Summarize the median gdpPercap by year, then save it as by_year
by_year <- gapminder %>%
```

```
  group_by(year) %>%
  summarize(medianGdpPercap = median(gdpPercap))

# Create a line plot showing the change in medianGdpPercap over time
ggplot(by_year, aes(x = year, y = medianGdpPercap)) +
  geom_line() +
  expand_limits(y = 0)
```



Visualizing median GDP per capita by continent over time

In the last exercise you used a line plot to visualize the increase in median GDP per capita over time. Now you'll examine the change within each continent.

INSTRUCTIONS

Use `group_by()` and `summarize()` to find the median GDP per capita within each year and continent, calling the output column `medianGdpPercap`. Use the assignment operator `<-` to save it to a dataset called `by_year_continent`.

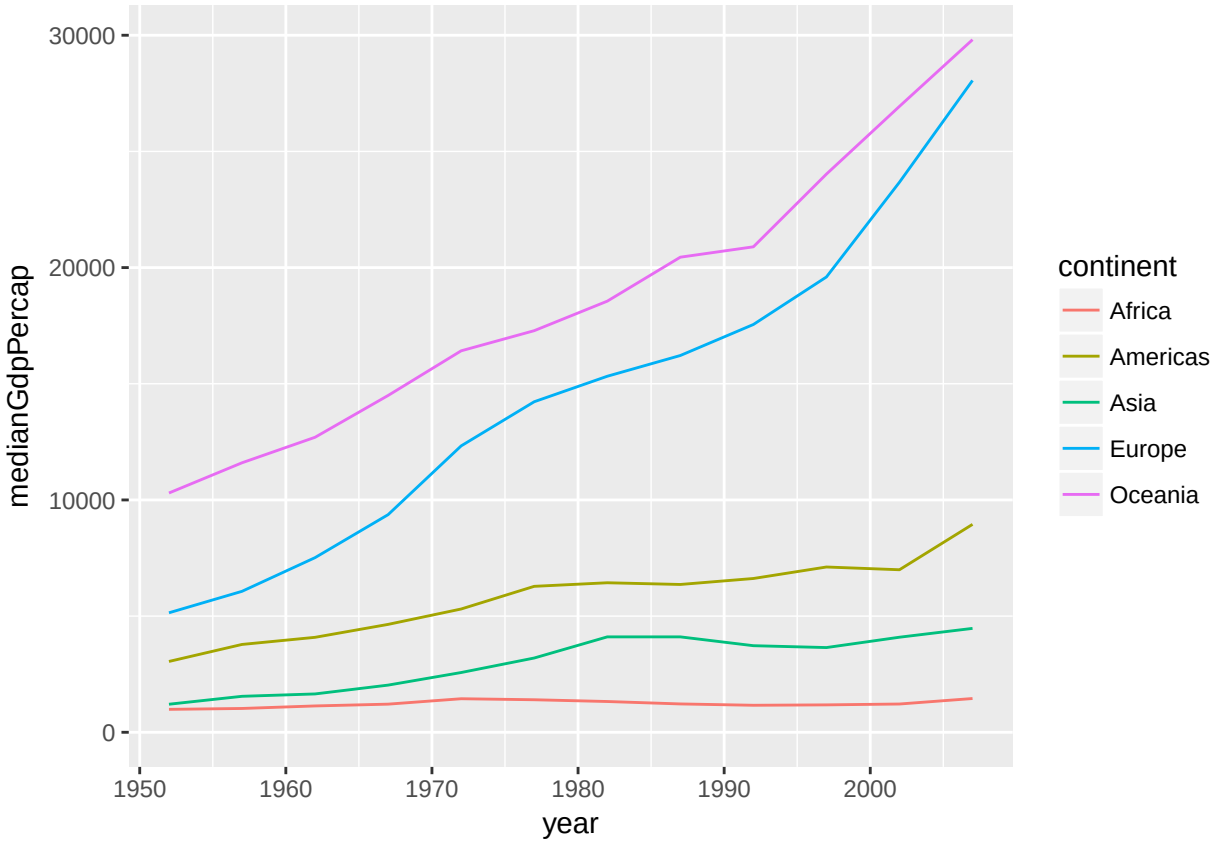
Use the `by_year_continent` dataset to create a line plot showing the change in median GDP per capita over time, with color representing continent. Be sure to use `expand_limits(y = 0)` to include 0 on the y-axis.

```
library(gapminder)
library(dplyr)
library(ggplot2)

# Summarize the median gdpPercap by year & continent, save as by_year_continent
```

```r
by_year_continent <- gapminder %>%
  group_by(year, continent) %>%
  summarize(medianGdpPercap = median(gdpPercap))

# Create a line plot showing the change in medianGdpPercap by continent over time
ggplot(by_year_continent, aes(x = year, y = medianGdpPercap, color = continent)) +
  geom_line() +
  expand_limits(y = 0)
```



Visualizing median GDP per capita by continent

A bar plot is useful for visualizing summary statistics, such as the median GDP in each continent.

INSTRUCTIONS

Use `group_by()` and `summarize()` to find the median GDP per capita within each continent in the year 1952, calling the output column `medianGdpPercap`. Use the assignment operator `<-` to save it to a dataset called `by_continent`.

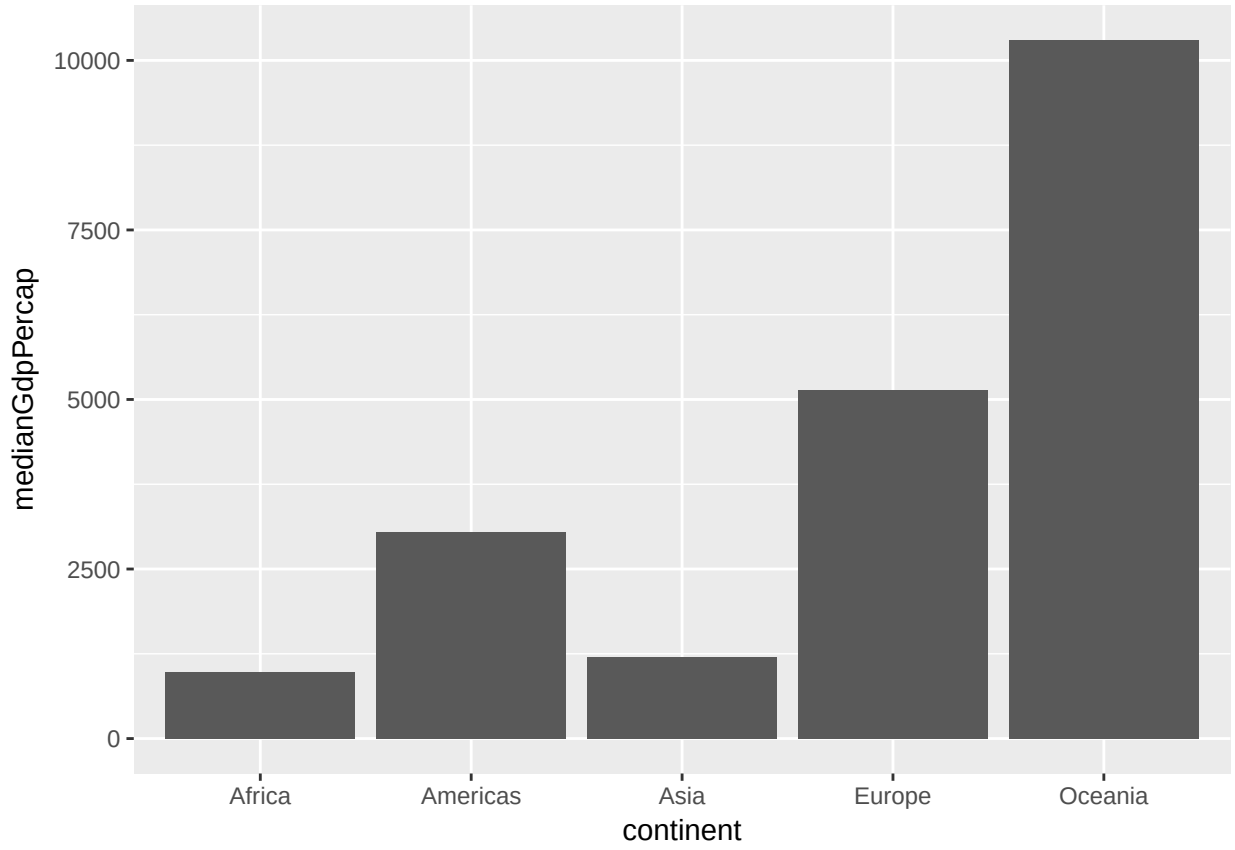
Use the `by_continent` dataset to create a bar plot showing the median GDP per capita in each continent.

```r
library(gapminder)
library(dplyr)
library(ggplot2)

# Summarize the median gdpPercap by year and continent in 1952
by_continent <- gapminder %>%
```

```
  filter(year == 1952) %>%
  group_by(continent) %>%
  summarize(medianGdpPercap = median(gdpPercap))

# Create a bar plot showing medianGdp by continent
ggplot(by_continent, aes(x = continent, y = medianGdpPercap)) +
  geom_col()
```



Visualizing GDP per capita by country in Oceania

You've created a plot where each bar represents one continent, showing the median GDP per capita for each. But the x-axis of the bar plot doesn't have to be the continent: you can instead create a bar plot where each bar represents a country.

In this exercise, you'll create a bar plot comparing the GDP per capita between the two countries in the Oceania continent (Australia and New Zealand).

INSTRUCTIONS

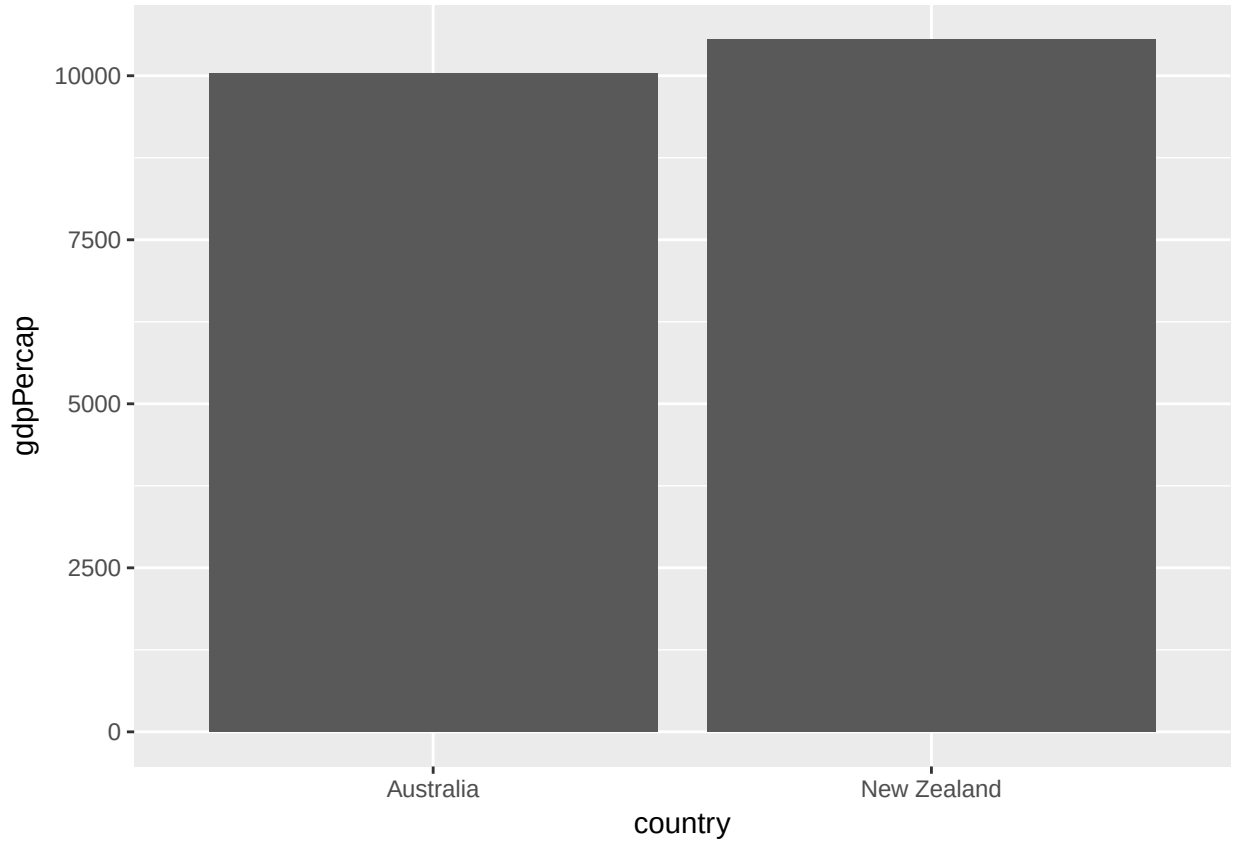
Filter for observations in the Oceania continent in the year 1952. Save this as `oceania_1952`.

Use the `oceania_1952` dataset to create a bar plot, with `country` on the x-axis and `gdpPercap` on the y-axis.

```
library(gapminder)
library(dplyr)
library(ggplot2)
```

```r
# Filter for observations in the Oceania continent in 1952
oceania_1952 <- gapminder %>%
  filter(continent == "Oceania" & year == 1952) %>%
  group_by(country)

# Create a bar plot of gdpPercap by country
ggplot(oceania_1952, aes(x = country, y = gdpPercap)) +
  geom_col()
```



Visualizing population

A histogram is useful for examining the distribution of a numeric variable. In this exercise, you'll create a histogram showing the distribution of country populations in the year 1952.

INSTRUCTIONS

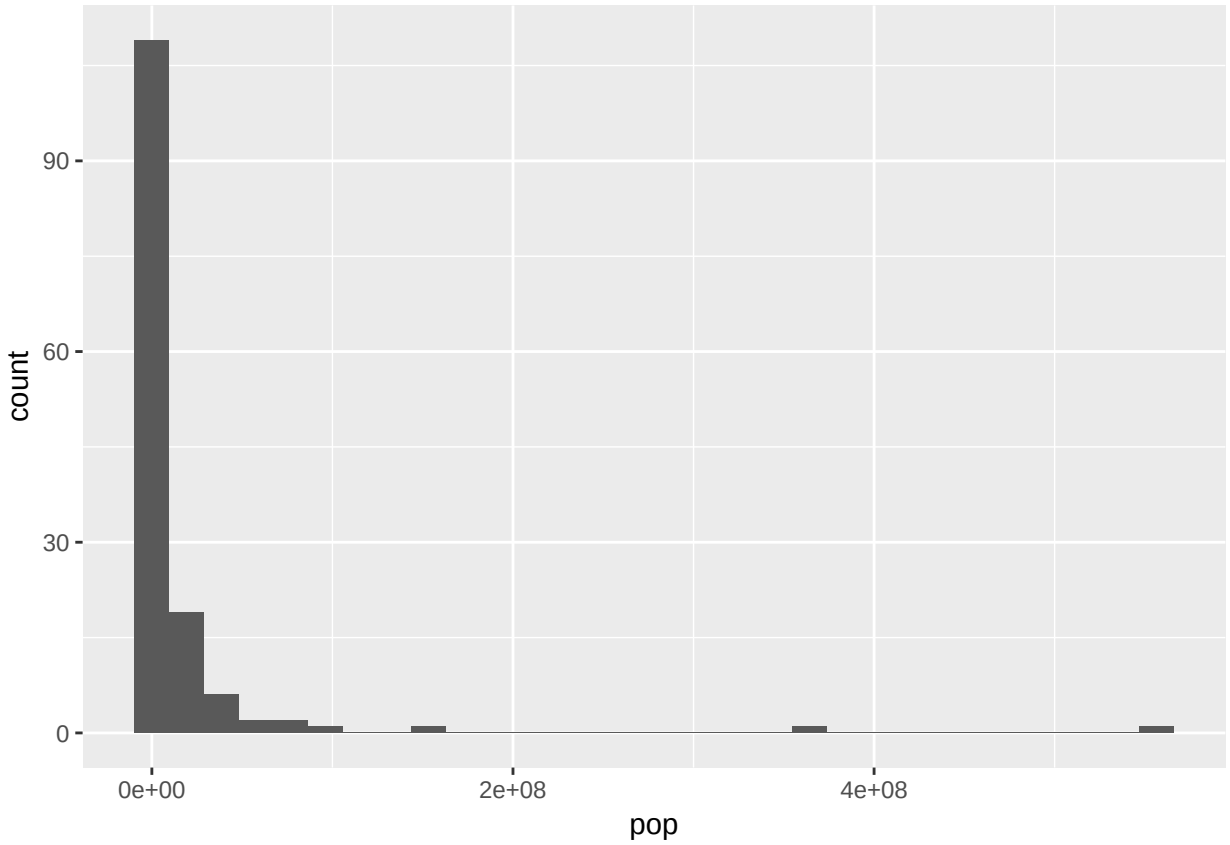
Use the `gapminder_1952` dataset (code for generating that dataset is provided) to create a histogram of country population (`pop`) in the year 1952.

```r
library(gapminder)
library(dplyr)
library(ggplot2)

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

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

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Visualizing population with x-axis on a log scale

In the last exercise you created a histogram of populations across countries. You might have noticed that there were several countries with a much higher population than others, which causes the distribution to be very skewed, with most of the distribution crammed into a small part of the graph. (Consider that it's hard to tell the median or the minimum population from that histogram).

To make the histogram more informative, you can try putting the x-axis on a log scale.

INSTRUCTIONS

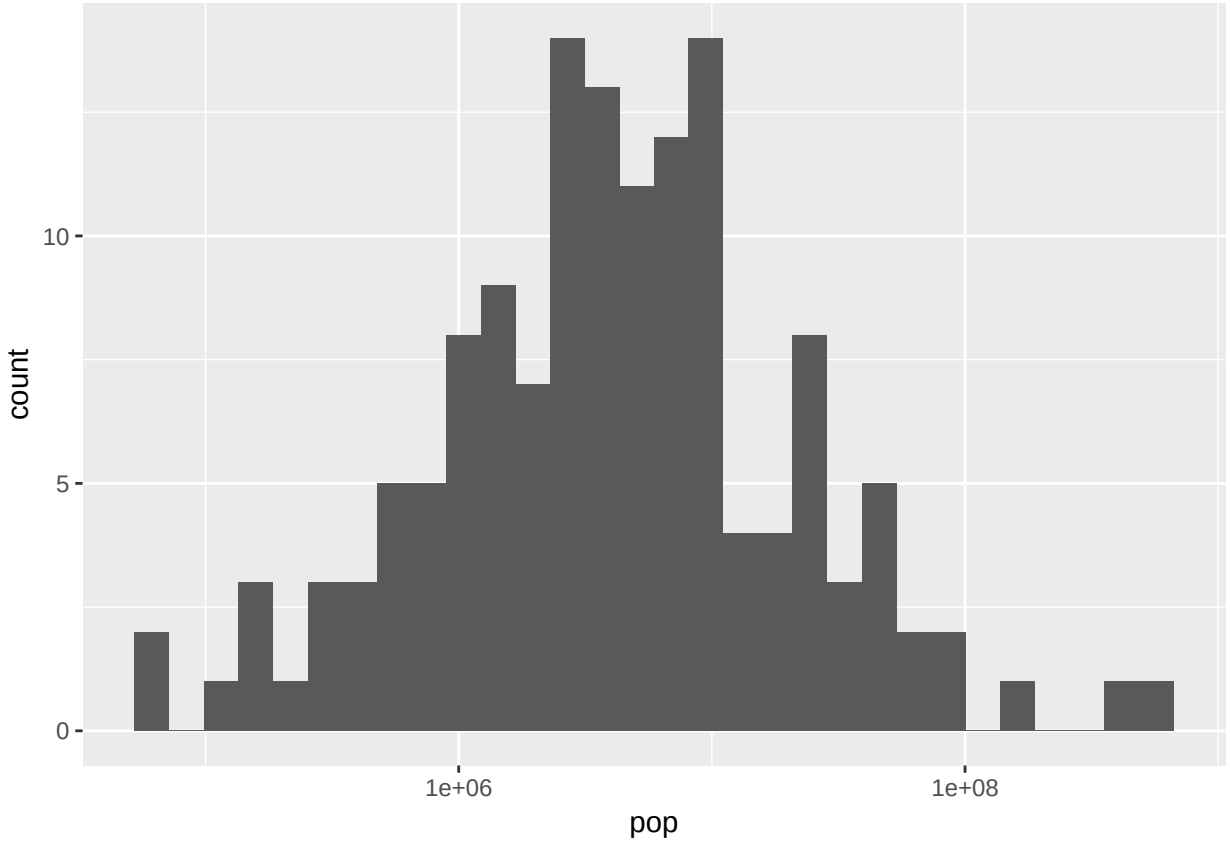
Use the `gapminder_1952` dataset (code is provided) to create a histogram of country population (`pop`) in the year 1952, putting the x-axis on a log scale with `scale_x_log10()`.

```
library(gapminder)
library(dplyr)
library(ggplot2)

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

```
# Create a histogram of population (pop), with x on a log scale
ggplot(gapminder_1952, aes(x = pop)) +
  geom_histogram() +
  scale_x_log10()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Comparing GDP per capita across continents

A boxplot is useful for comparing a distribution of values across several groups. In this exercise, you'll examine the distribution of GDP per capita by continent. Since GDP per capita varies across several orders of magnitude, you'll need to put the y-axis on a log scale.

INSTRUCTIONS

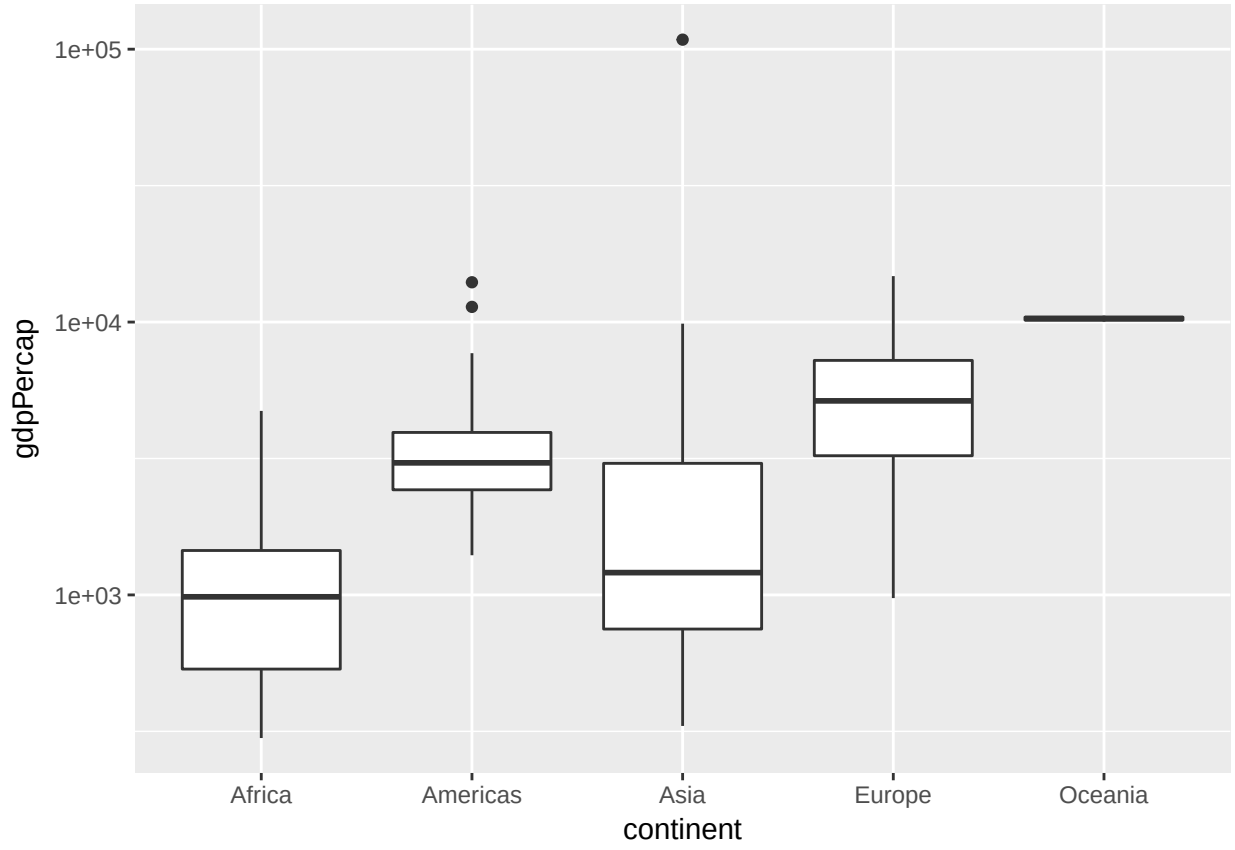
Use the `gapminder_1952` dataset (code is provided) to create a boxplot comparing GDP per capita (`gdpPercap`) among continents. Put the y-axis on a log scale with `scale_y_log10()`.

```
library(gapminder)
library(dplyr)
library(ggplot2)

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

# Create a boxplot comparing gdpPercap among continents
```

```
ggplot(gapminder_1952, aes(x = continent, y = gdpPercap)) +
  geom_boxplot() +
  scale_y_log10()
```



Adding a title to your graph

There are many other options for customizing a `ggplot2` graph, which you can learn about in other DataCamp courses. You can also learn about them from online resources, which is an important skill to develop.

As the final exercise in this course, you'll practice looking up `ggplot2` instructions by completing a task we haven't shown you how to do.

INSTRUCTIONS

Add a title to the graph: Comparing GDP per capita across continents. Use a search engine, such as Google or Bing, to learn how to do so.

```
library(gapminder)
library(dplyr)
library(ggplot2)

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

# Add a title to this graph: "Comparing GDP per capita across continents"
ggplot(gapminder_1952, aes(x = continent, y = gdpPercap)) +
  geom_boxplot() +
```

```
  scale_y_log10() +
  ggtitle("Comparing GDP per capita across continents")
```