Assignment 3

Ingrid-Liv Morkken & Thibiga Kuddyar

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.5     v dplyr   1.0.7
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.0.1     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(readr)
library(gapminder)
library(magrittr)
```

```
##
## Attaching package: 'magrittr'

## The following object is masked from 'package:purrr':
##
##     set_names

## The following object is masked from 'package:tidyr':
##
##     extract
```

```
library(dplyr)
options(scipen
        = 999)
```

1

The file ddf_concepts.cvs contains descriptions of a set of different variables including unemployment rate, adults with hiv, labour force participation, aid, causes of deaths, cell phones, cars, health factors, countries GDP etc. Each of the 600 variables further includes data.

2

The file ddf–entities–geo–country.csv contains information of all the countries in the world where each country is categorized based on g77 and OECD, UN members/recognition, income groups, religion, latitude, longitude and region (Europe and world).

3

df–entities–geo–un_sdg_region.csv contains information about UN recognized regions.

4

The gapminder dataset consist of 1704 rows and includes the following variables:

- Country
- Continent
- Year
- LifeExp
- Pop
- gdpPercap

Australia and New Zealand is assigned to Asia.

5

```
g_c <- read_csv("data/ddf--entities--geo--country.csv")
```

```
## Rows: 273 Columns: 22

## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (17): country, g77_and_oecd_countries, income_3groups, income_groups, is...
## dbl  (3): iso3166_1_numeric, latitude, longitude
## lgl  (2): is--country, un_state

##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
print(g_c)
```

```
## # A tibble: 273 x 22
##    country   g77_and_oecd_countries income_3groups income_groups   `is--country`
##    <chr>     <chr>                  <chr>          <chr>           <lgl>
##  1 abkh      others                 <NA>           <NA>            TRUE
##  2 abw       others                 high_income    high_income     TRUE
```

```
##  3 afg       g77                   low_income     low_income      TRUE
##  4 ago       g77                   middle_income  lower_middle_i~ TRUE
##  5 aia       others                <NA>           <NA>            TRUE
##  6 akr_a_dhe others                <NA>           <NA>            TRUE
##  7 ala       others                <NA>           <NA>            TRUE
##  8 alb       others                middle_income  upper_middle_i~ TRUE
##  9 and       others                high_income    high_income     TRUE
## 10 ant       others                <NA>           <NA>            TRUE
## # ... with 263 more rows, and 17 more variables: iso3166_1_alpha2 <chr>,
## #   iso3166_1_alpha3 <chr>, iso3166_1_numeric <dbl>, iso3166_2 <chr>,
## #   landlocked <chr>, latitude <dbl>, longitude <dbl>,
## #   main_religion_2008 <chr>, name <chr>, un_sdg_ldc <chr>,
## #   un_sdg_region <chr>, un_state <lgl>, unhcr_region <chr>,
## #   unicef_region <chr>, unicode_region_subtag <chr>, world_4region <chr>,
## #   world_6region <chr>
```

By running the chunk above, only countries with code *aiso3166_1_aplha3* will be included thus the tibble will be named *g_c*.

The chunk below creates new variables by using "mutate" and the "case-when" function allows to vectorize several statements simultaneous.

```
g_c <- g_c %>%
  mutate(continent = case_when(
     world_4region == "asia" & un_sdg_region %in% c("un_australia_and_new_zealand", "un_oceania_exc_aus
    world_4region ==  "asia" & !(un_sdg_region %in% c("un_australia_and_new_zealand", "un_oceania_exc_a
    world_4region == "africa" ~ "Africa",
    world_4region == "americas" ~ "Americas",
    world_4region ==  "europe" ~ "Europe")
  ) %>%
 filter(!is.na(iso3166_1_alpha3))
```

6a

```
length(unique(g_c$country))
```

```
## [1] 247
```

After filtrating the data from #5, are we left with 247 countries. This is done by using the function "lenght(unique())" which calculates the number of values in a vector.

6b

We are using *pipe* which will allows to forward a value into next function. Then we use "group_by()" to take an existing table and converts into grouped table. Finally, we summarise by reducing the dataframe by using "summarize()".

```
g_c %>%
  group_by(continent) %>%
  summarise(countries = length(unique(country)))
```

```
## # A tibble: 5 x 2
##   continent countries
##   <chr>         <int>
## 1 Africa           59
## 2 Americas         55
## 3 Asia             47
## 4 Europe           58
## 5 Oceania          28
```

There are now a number of:

- 59 countries in Africa
- 55 countries in Americas
- 47 countries in Europe
- 28 countries in Oceania

7

```
lifeExp <- read_csv(
  "data/countries-etc-datapoints/ddf--datapoints--life_expectancy_years--by--geo--time.csv",
col_types = cols(
    time = col_date(
      format ="%Y"
      )))
lifeExp <- lifeExp %>%
rename(
  year = time)
length(
  unique(
    lifeExp$geo
    ))
```

```
## [1] 195
```

```
names(
  lifeExp
  )
```

```
## [1] "geo"                  "year"                 "life_expectancy_years"
```

The chunk above shows how the variable "Life Expectancy" has been added to the dataset *g_c*. We have changed the format from *time* to *date* with %Y which gives a four-digit year.

8

```
length(unique(lifeExp$geo))
```

```
## [1] 195
```

By running the chunk above, it shows that there are 195 countries that have information about the life expectancy.

9

We use pipe again and select the variables to be included in the in the reduced dataset by using "select()". Then we insert variables from *LifeExp* into the *g_c* by using "left_join()". We are using "rm()" to reduce enormous number of observations.

```
g_c <- g_c %>%
  select (
    country,
    name,
    iso3166_1_alpha3,
    un_sdg_region,
    world_4region,
    continent,
    world_6region
    ) %>%
  left_join(
    lifeExp, by = c(
      "country" = "geo"
      ))
names(
  g_c
  )
```

```
## [1] "country"              "name"                 "iso3166_1_alpha3"
## [4] "un_sdg_region"        "world_4region"        "continent"
## [7] "world_6region"        "year"                 "life_expectancy_years"
```

```
rm(lifeExp)
```

10

We creates a new dataset called "g-c-min" which shows the first observations of "LifeExp" in different countries. This time we use a new function called "table()" which create categorical representation of data.

```
g_c_min <- g_c %>%
  group_by(country) %>%
  summarise(min_year = min(year))
table(g_c_min$min_year)
```

```
##
## 1800-01-01 1950-01-01
##        186          9
```

The observation is that 186 countries has data on **life expectancy years** from 1800 whereas only nine countries has data on **life expectancy years** from 1950.

11

```
g_c_min %>%
filter(min_year == "1950-01-01")
```

```
## # A tibble: 9 x 2
##   country min_year
##   <chr>   <date>
## 1 and     1950-01-01
## 2 dma     1950-01-01
## 3 kna     1950-01-01
## 4 mco     1950-01-01
## 5 mhl     1950-01-01
## 6 nru     1950-01-01
## 7 plw     1950-01-01
## 8 smr     1950-01-01
## 9 tuv     1950-01-01
```

The nine countries that only have life expectancy data from 1950 are displayed in the table above.

12

We create a new dataset called "pop" and then we insert the variables from that dataset to *g_c*.

```
pop <- read_csv("data/countries-etc-datapoints/ddf--datapoints--population_total--by--geo--time.csv",
col_types = cols(time = col_date(format = "%Y")))
```

```
g_c <- g_c %>%
  left_join(pop, by = c("country" = "geo", "year" = "time"))
rm(pop)
```

13

We repeat the same operations as in task 12 in task 13, but we use a different file and the new dataset is called "gdp_pc".

```
gdp_pc <- read_csv(
  # triks for å unngå filnavn som går ut i margen og utenfor arket
  paste0("data/countries-etc-datapoints/",
  "ddf--datapoints--gdppercapita_us_inflation_adjusted--by--geo--time.csv"
  ),
col_types = cols(time = col_date(format = "%Y")))
```

```
g_c <- g_c %>%
  left_join(gdp_pc, by = c("country" = "geo", "year" = "time"))
rm(gdp_pc)
```

In addition, we chagne the names of some of the variables in the *g_c* dataset by using "rename()".

```
g_c <- g_c %>%
  rename("lifeExp" = "life_expectancy_years") %>%
  rename ("pop" = "population_total") %>%
  rename ("gdpPercap" = "gdppercapita_us_inflation_adjusted" )
```

The chunk below gives an overview of the names of the variables in the *g_c* dataset.

```
names(g_c)
```

```
##  [1] "country"          "name"             "iso3166_1_alpha3" "un_sdg_region"
##  [5] "world_4region"    "continent"        "world_6region"    "year"
##  [9] "lifeExp"          "pop"              "gdpPercap"
```

14

We create data called Tbl1 which includes data from every 5th year, from 1800 till 2015, including 2019. The function "paste()" put together vectors by converting them into character. Meanwhile "parse_date()" converts the textual representation of R code into an internal form.

```
Tbl1 <- paste(c(seq(1800, 2015, by = 5), 2019),
 "01-01", sep = "-" )  %>%
 parse_date(format = "%Y-%m-%d")
```

We are again using *pipe* and this time we use "%in%" to identify "year" in *Tbl1".

```
g_c_5 <- g_c %>%
  filter(year %in% Tbl1) %>%
  select(country, name, continent, year, lifeExp, pop, gdpPercap)
```

We use "dim()" to give us the dimesion of the materix. In this case we have 8505 observations and 7 variables.

```
dim(g_c_5)
```

```
## [1] 8505    7
```

```
g_c_gdp_fy <- g_c_5 %>%
  group_by(gdpPercap) %>%
  summarise(min_year = min(year))
```

We use "count" which give us a tibble.

```
g_c_gdp_fy %>%
  count(min_year = g_c_gdp_fy$min_year)
```

```
## # A tibble: 14 x 2
##    min_year       n
##    <date>     <int>
##  1 1800-01-01     1
##  2 1960-01-01    86
##  3 1965-01-01    93
##  4 1970-01-01   108
##  5 1975-01-01   112
##  6 1980-01-01   133
##  7 1985-01-01   142
##  8 1990-01-01   161
##  9 1995-01-01   178
## 10 2000-01-01   186
## 11 2005-01-01   189
## 12 2010-01-01   191
## 13 2015-01-01   188
## 14 2019-01-01   186
```

15

```
g_c <- g_c %>%
  filter(!is.na(gdpPercap)) %>%
  group_by(country) %>%
  summarise(nr=n()) %>%
  arrange ((country))
```

We use *61* since it is the highest recorded observation.

```
g_c_61 <- g_c %>%
  filter(nr == 61)
```

This give us 84 observations.

16

We create a new dataset without NA observations called *l_min_y*.

```
l_min_y <- g_c_5 %>%
filter(!is.na(gdpPercap)) %>%
group_by(country) %>%
summarise(min_year = min(year))
```

```
dim(l_min_y)
```

```
## [1] 191   2
```

```
l_min_y_60 <- l_min_y$country[l_min_y$min_year == "1960-01-01"]
my_gapminder_1960 <- g_c_5 %>%
filter(country %in% l_min_y_60)
```

```
dim(my_gapminder_1960)
```

```
## [1] 3870    7
```

```
length(unique(my_gapminder_1960$country))
```

```
## [1] 86
```

```
(m_v <- my_gapminder_1960[is.na(my_gapminder_1960$gdpPercap) == TRUE,])
```

```
## # A tibble: 2,754 x 7
##    country name      continent year       lifeExp    pop gdpPercap
##    <chr>   <chr>     <chr>     <date>       <dbl>  <dbl>     <dbl>
##  1 arg     Argentina Americas  1800-01-01    33.2 534000        NA
##  2 arg     Argentina Americas  1805-01-01    33.2 465622        NA
##  3 arg     Argentina Americas  1810-01-01    33.2 419661        NA
##  4 arg     Argentina Americas  1815-01-01    33.2 465972        NA
##  5 arg     Argentina Americas  1820-01-01    33.2 530996        NA
##  6 arg     Argentina Americas  1825-01-01    33.2 582027        NA
##  7 arg     Argentina Americas  1830-01-01    33.2 634974        NA
##  8 arg     Argentina Americas  1835-01-01    33.2 698047        NA
##  9 arg     Argentina Americas  1840-01-01    33.2 776366        NA
## 10 arg     Argentina Americas  1845-01-01    33.2 920317        NA
## # ... with 2,744 more rows
```

```
paste("Number of NAs in my_gapminder_1960 is", dim(m_v)[1], sep = " ")
```

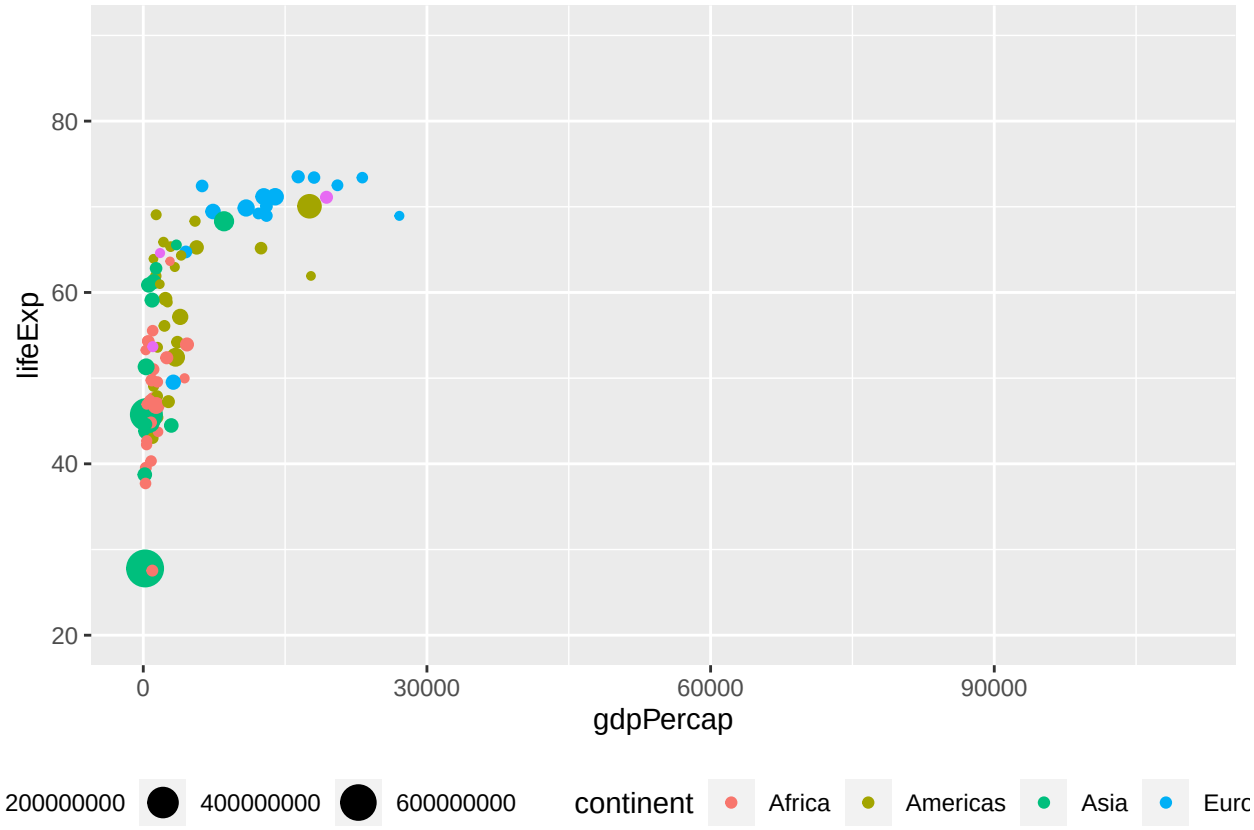
```
## [1] "Number of NAs in my_gapminder_1960 is 2754"
```

There are 2754 numbers of NAs in this dataset.

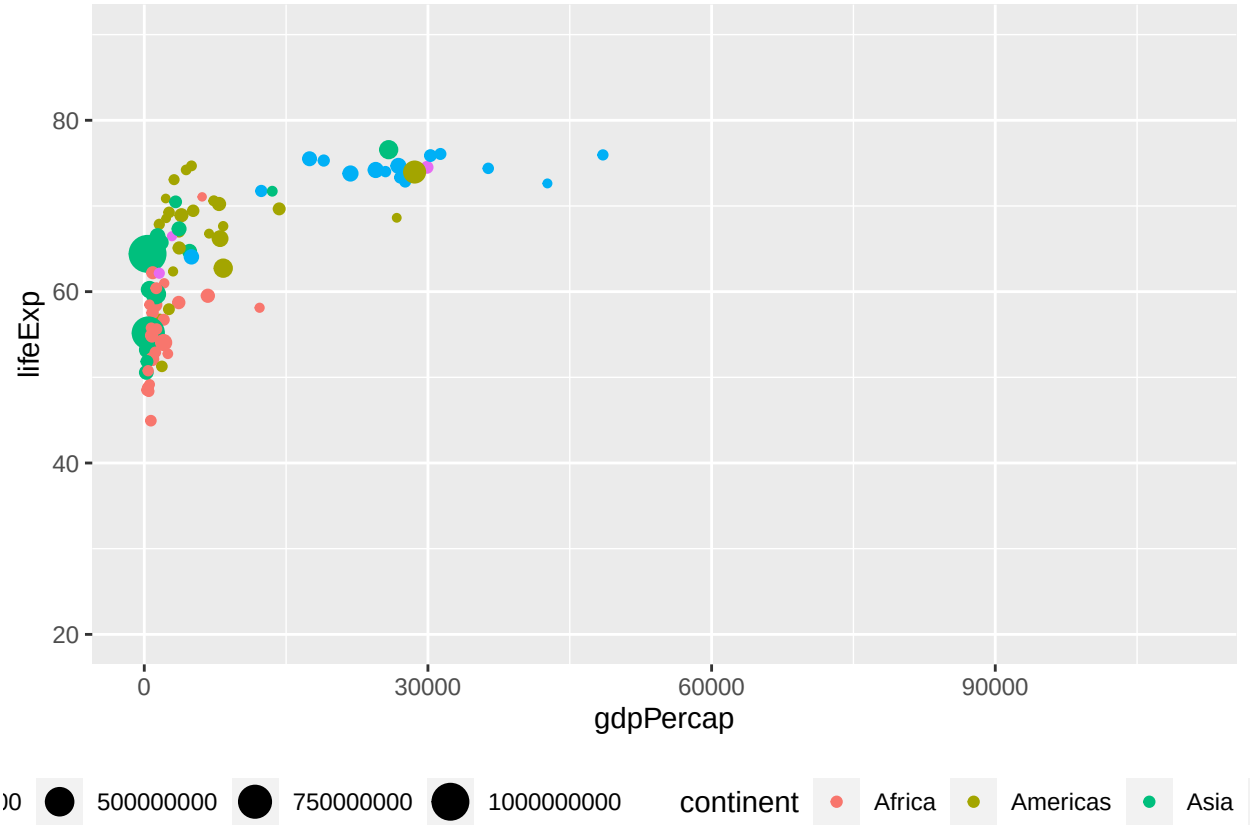
17

We use “ggplot()” to create graphs.

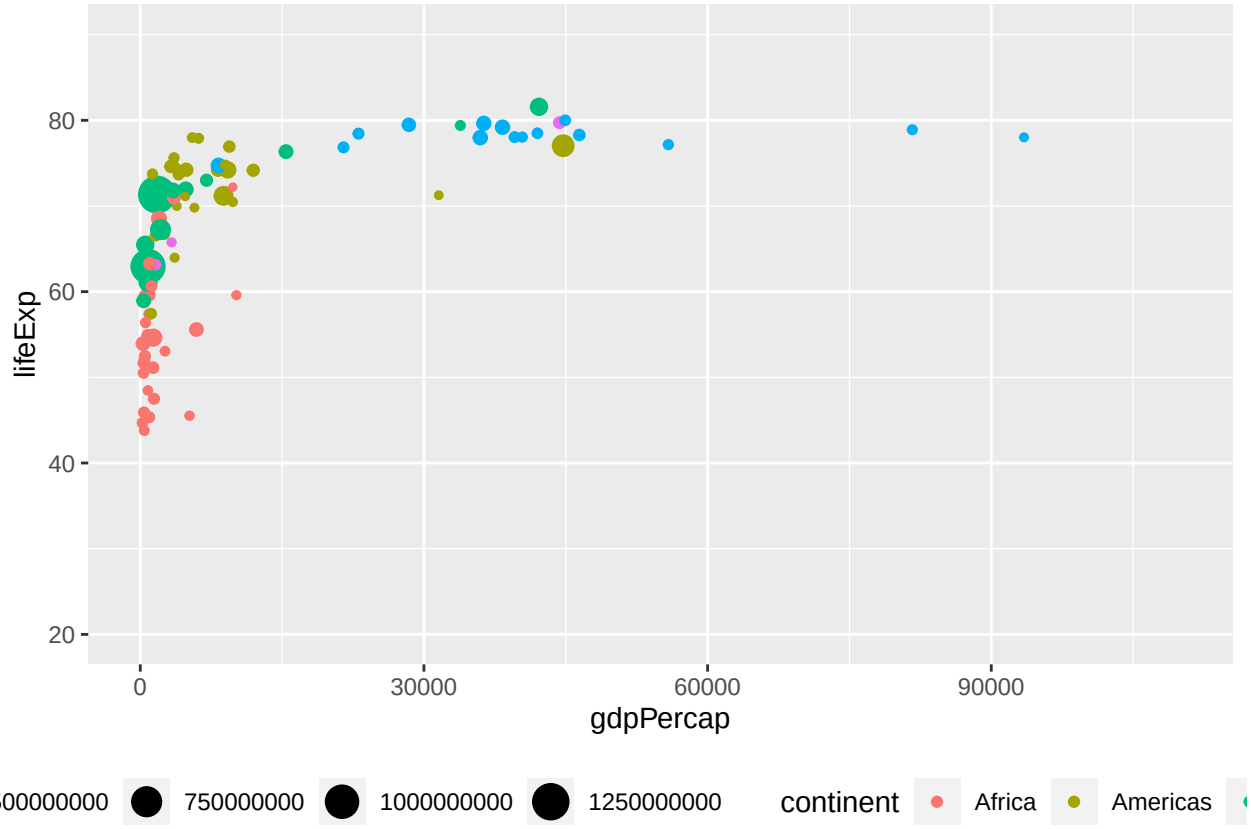
```
# Formaterer koden så den er lettere å lese
my_gapminder_1960 %>%
filter(year == "1960-01-01") %>%
  ggplot(
    mapping = aes(x = gdpPercap,
                  y = lifeExp,
                  size = pop,
                  colour = continent)
    ) +
  geom_point() +
  coord_cartesian(ylim = c(20, 90), xlim = c(0,110000)) +
  theme(legend.position = "bottom")
```




```
# kode stylet vha. add-in Styler > Style selection
my_gapminder_1960 %>%
  filter(year == "1980-01-01") %>%
  ggplot(mapping = aes(x = gdpPercap, y = lifeExp, size = pop, colour = continent)) +
  geom_point() +
  coord_cartesian(ylim = c(20, 90), xlim = c(0, 110000)) +
  theme(legend.position = "bottom")
```

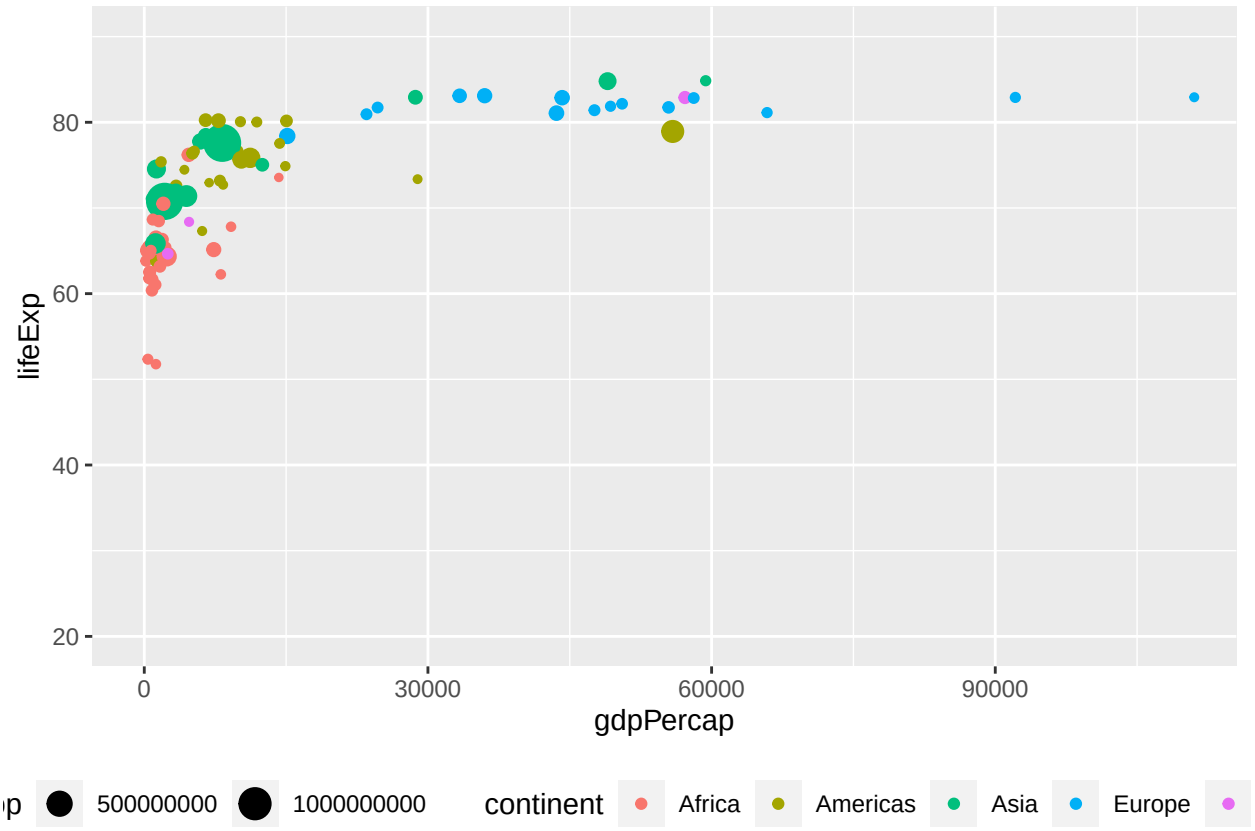



```
my_gapminder_1960 %>%
filter(year == "2000-01-01") %>%
  ggplot(mapping = aes(x = gdpPercap, y = lifeExp, size = pop, colour = continent)) + geom_point() + co
```




```
my_gapminder_1960 %>%
filter(year == "2019-01-01") %>%
  ggplot(mapping = aes(x = gdpPercap, y = lifeExp, size = pop, colour = continent)) + geom_point() + co
```

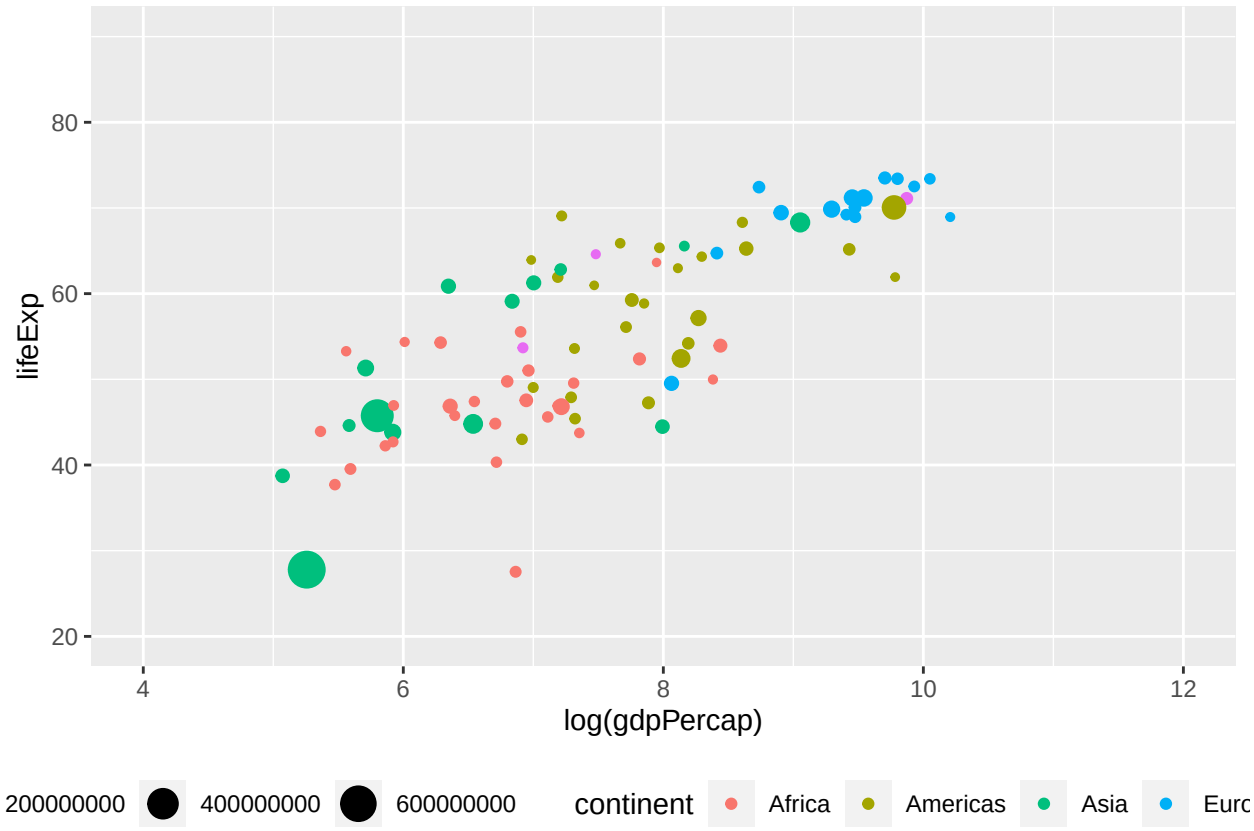
```
## Warning: Removed 1 rows containing missing values (geom_point).
```



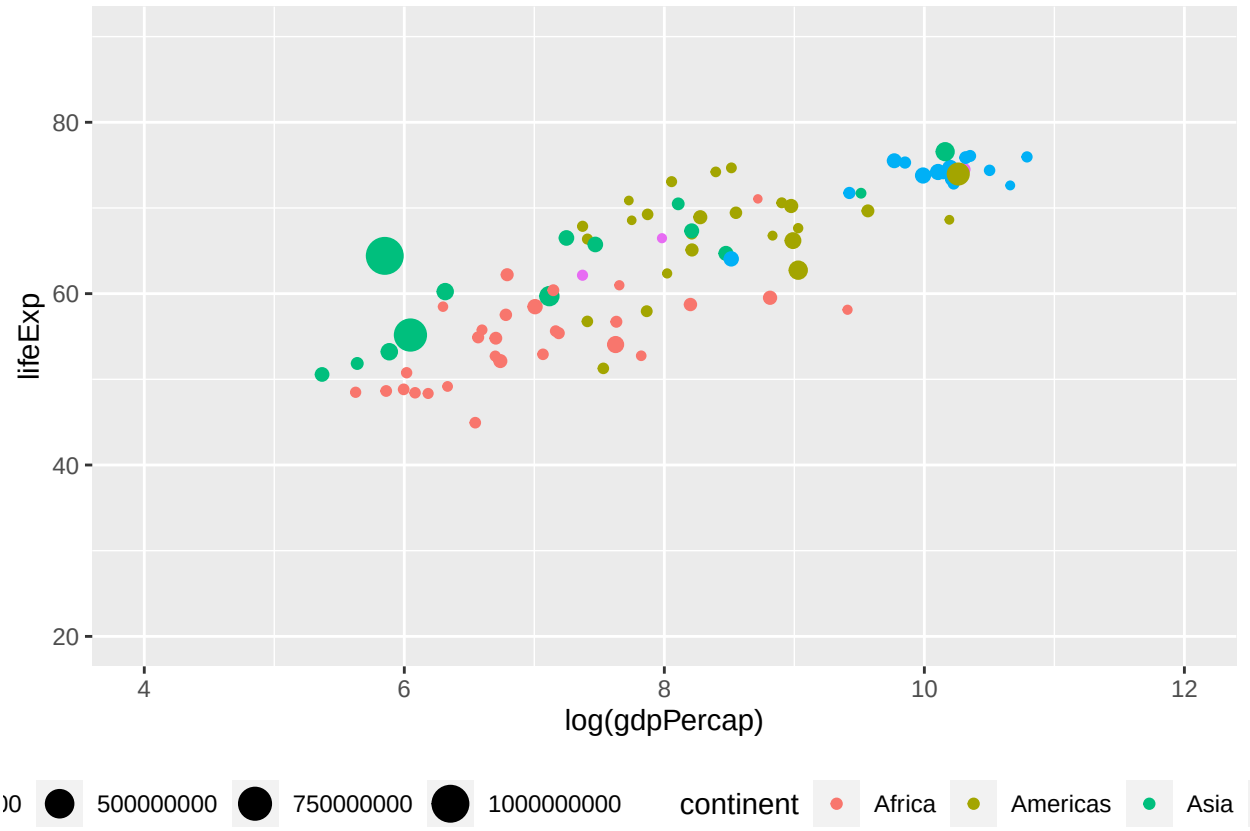
18

We use again ggplot to create graphs but this time with "log()".

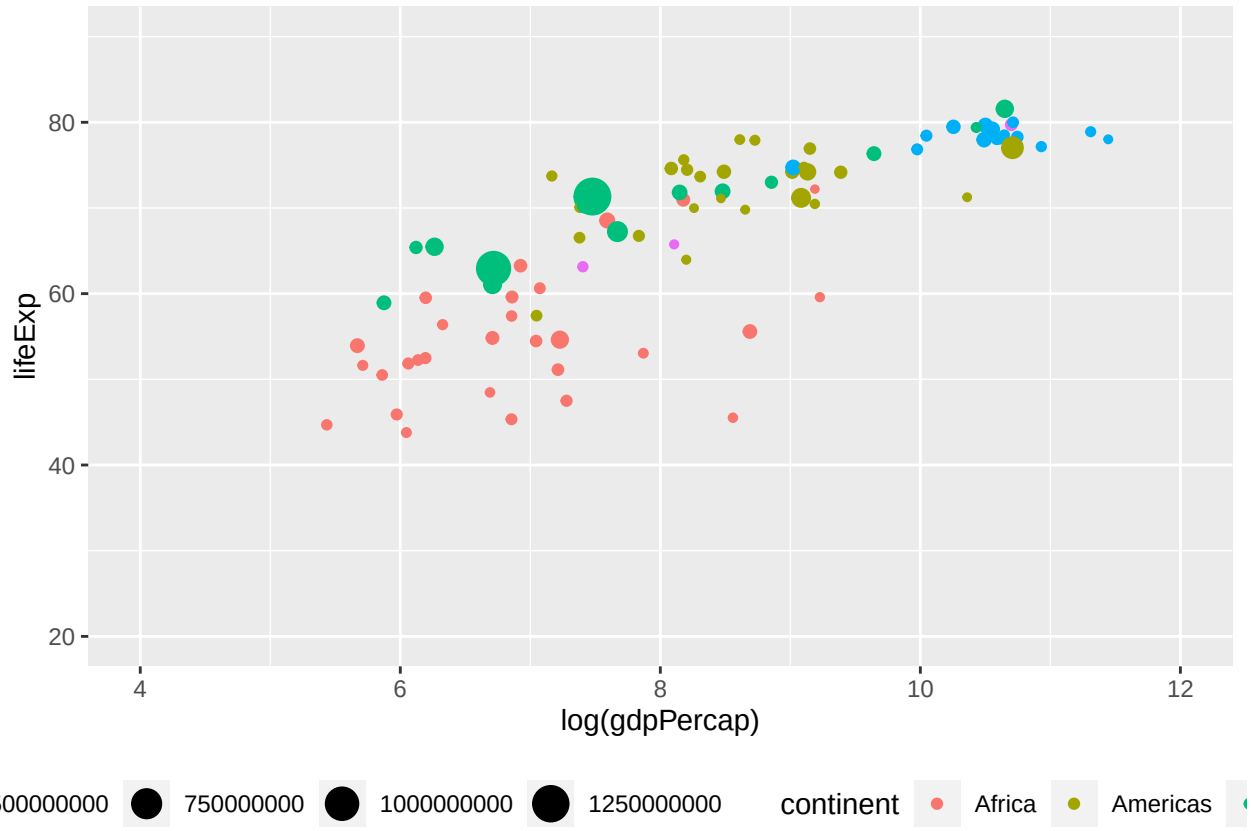
```
my_gapminder_1960 %>%
filter(year == "1960-01-01") %>%
  ggplot(mapping = aes(x = log(gdpPercap), y = lifeExp, size = pop, colour = continent)) + geom_point()
```




```
my_gapminder_1960 %>%
filter(year == "1980-01-01") %>%
  ggplot(mapping = aes(x = log(gdpPercap), y = lifeExp, size = pop, colour = continent)) + geom_point()
```

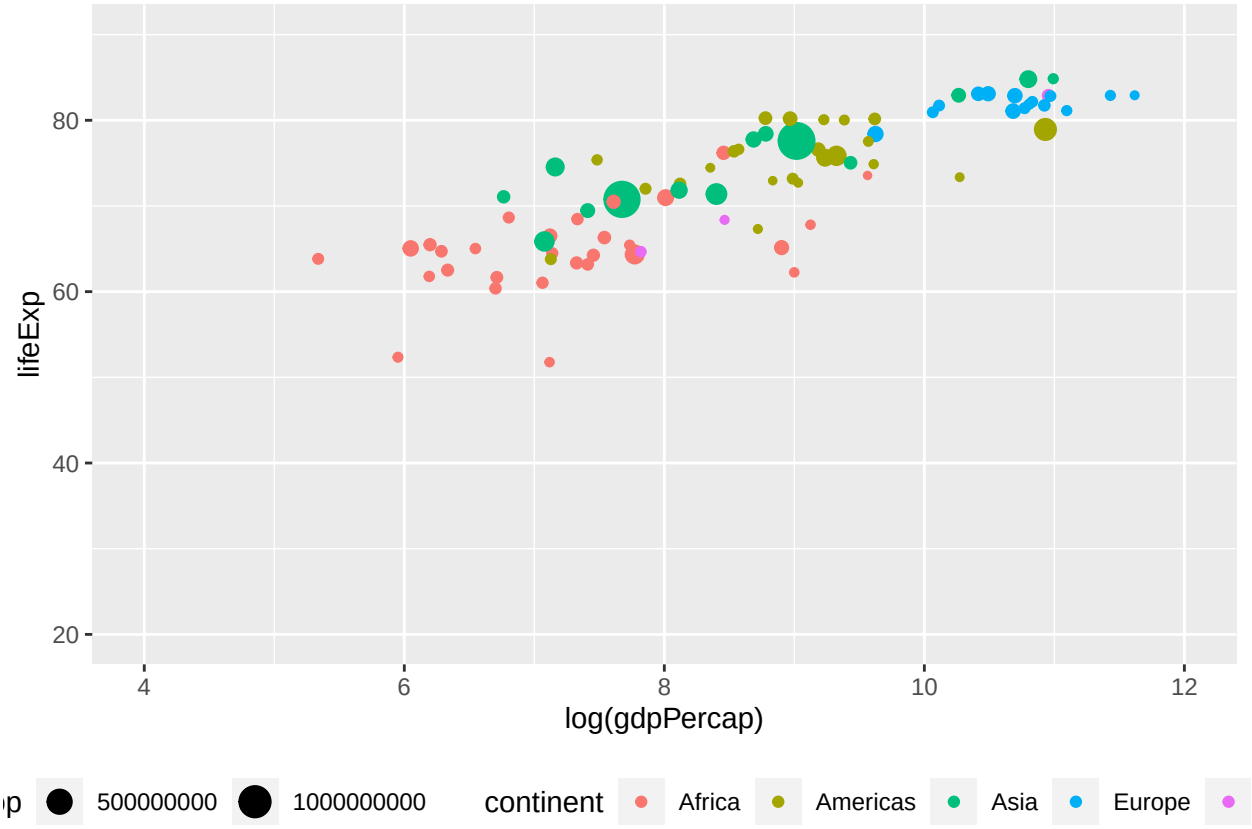



```
my_gapminder_1960 %>%
filter(year == "2000-01-01") %>%
  ggplot(mapping = aes(x = log(gdpPercap), y = lifeExp, size = pop, colour = continent)) + geom_point()
```




```
my_gapminder_1960 %>%
filter(year == "2019-01-01") %>%
  ggplot(mapping = aes(x = log(gdpPercap), y = lifeExp, size = pop, colour = continent)) + geom_point()
```

```
## Warning: Removed 1 rows containing missing values (geom_point).
```



19

We can see clear signs of development from 1959 to 2019. Several countries, especially countries in Africa and Asia, have become better at reporting. In addition, GDP has had a general growth and thus the average life expectancy in various countries has increased.

20

We use the function "write.table()" to export a dataframe to a file.

```
write.table(g_c, file="my_gapminder.csv", sep = ",")
write.table(g_c_61, file="my_gapminder_red.csv", sep = ",")
```

```
#siste
```