Notes on Tidyverse

Shashank Sule

9/15/2019

Tidyverse is the collection of tools which help you visualize and explore data. The example we will use is the gapminder data set.

```
require(gapminder)
```

```
## Loading required package: gapminder
```

```
require(dplyr)
```

```
## Loading required package: dplyr

## Warning: package 'dplyr' was built under R version 3.4.4

##
## 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
```

```
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
```

Note that `gapminder` is a `tibble`, a bit different from `data.frame`.

Verbs from `dplyr`

Verbs are the functions within `dplyr`. Every time you use a verb, you use a pipe which composes verbs.

1. `filter`

`filter` literally filters your data frame according to the criteria you give. Suppose you want to filter the dataset for 2007.

```
gapminder %>%
  filter(year == 2007) #Indexing the array by boolean variables
```

```
## Warning: package 'bindrcpp' was built under R version 3.4.4
```

```
## # A tibble: 142 x 6
##    country     continent  year lifeExp       pop gdpPercap
##    <fct>       <fct>     <int>   <dbl>     <int>     <dbl>
##  1 Afghanistan Asia       2007    43.8  31889923       975
##  2 Albania     Europe     2007    76.4   3600523      5937
##  3 Algeria     Africa     2007    72.3  33333216      6223
##  4 Angola      Africa     2007    42.7  12420476      4797
##  5 Argentina   Americas   2007    75.3  40301927     12779
##  6 Australia   Oceania    2007    81.2  20434176     34435
##  7 Austria     Europe     2007    79.8   8199783     36126
##  8 Bahrain     Asia       2007    75.6    708573     29796
##  9 Bangladesh  Asia       2007    64.1 150448339      1391
## 10 Belgium     Europe     2007    79.4  10392226     33693
## # ... with 132 more rows
```

```
#You can also double filter

gapminder %>% filter(year == 2007, country == "United States")
```

```
## # A tibble: 1 x 6
##   country       continent  year lifeExp       pop gdpPercap
##   <fct>         <fct>     <int>   <dbl>     <int>     <dbl>
## 1 United States Americas   2007    78.2 301139947     42952
```

2. `arrange`

`arrange` sorts the dataset according to the column you want to arrange by. By default it arranges according to ascending order. To sort in descending order you do `arrange(desc(#colname))`

```
# 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
```

3. `mutate`

`mutate` either changes an existing column in the data frame or adds a new one. Note that it is not a "static" function: it does not make any changes to the data frame you pass as an argument. Instead, it returns a new data frame that is a result of passing the original data frame through `mutate`.

```
# 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
```

```r
# Use mutate to create a new column called lifeExpMonths

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
```

```r
# Filter, mutate, and arrange the gapminder dataset

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

4. `summarize`

`summarize` returns a `tibble` with "summarizing" statistics of the dataset you give.

```
# Summarize to find the median life expectancy

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

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

```
# Filter for 1957 then summarize the median life expectancy
gapminder %>% filter(year == 2007) %>%
 summarize(medianLifeExp = median(lifeExp))
```

```
## # A tibble: 1 x 1
##   medianLifeExp
##           <dbl>
## 1          71.9
```

```
# Summarize can even return two values

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

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

5. `group_by`

In the previous example, filter allowed you to select a value of one particular column, which then you summarized. In order to summarize over all values of a particular column, we use `group_by`. It's equivalent to using a for loop on `filter` where the loop runs over the all the possible arguments passable to filter (which is what you would do to get an overall summary table if you didn't know about `group_by`).

```
# 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
```

```
# 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
```

```
#You can even pass two arguments to group_by. In that case it returns the summary tabl

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
```

```
# In the above output, summarize gave the summaries at all possible (year, continent)
```

Visualization with ggplot2

ggplot2 is the canonical data visualization tool. We'll load it here

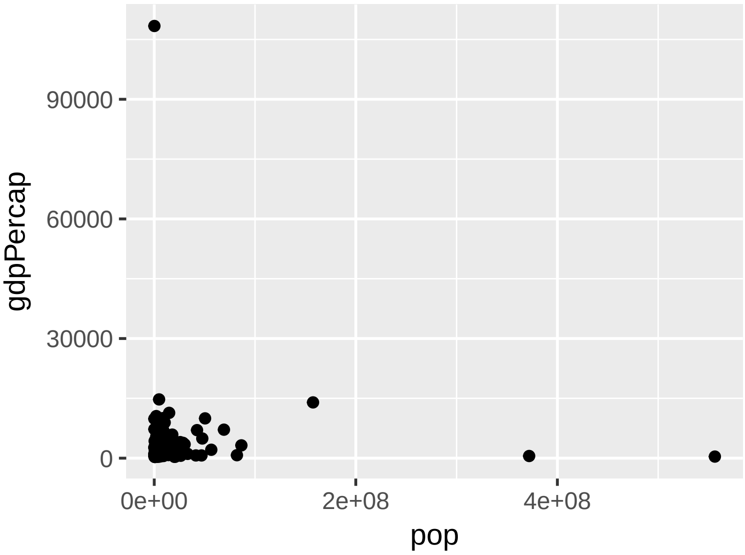
```
library("ggplot2")
```

The basic syntax for making a plot is `ggplot(#dataframe, ae(x=#xcomponent, y=#component))`. The `+` operator will signify additional features to the plot.

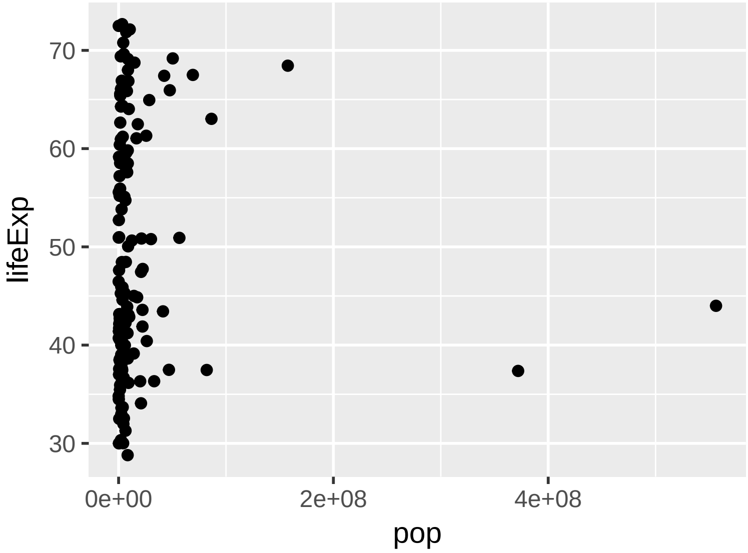
1. Scatter plot

Scatter plots are achieved using `+ geom_point()`.

```
library(dplyr)
library(gapminder)
gapminder_1952 <- gapminder %>%
  filter(year == 1952)
ggplot(gapminder_1952, aes(x = pop, y = gdpPercap)) +
  geom_point()
```



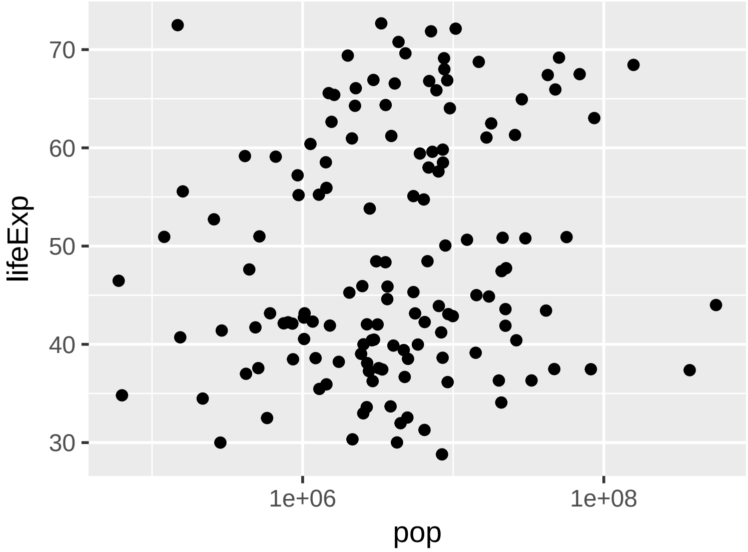
```r
ggplot(gapminder_1952, aes(x=pop, y=lifeExp)) + geom_point()
```



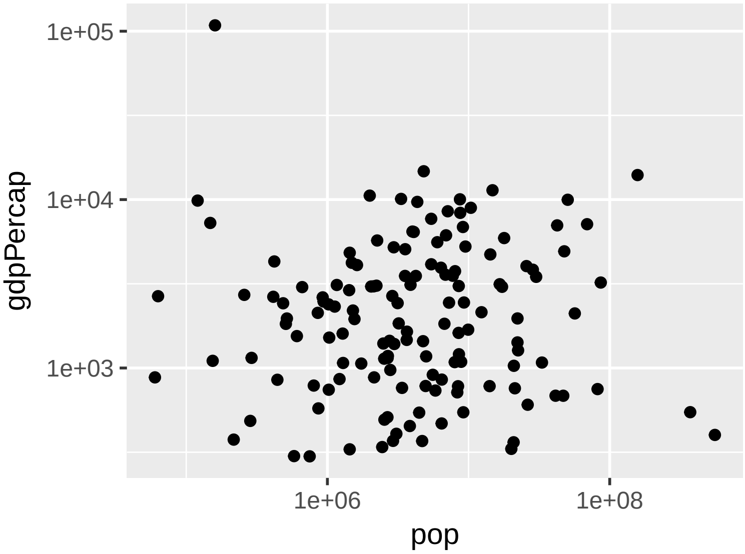
2. Log scale

You can use log axes by adding `+ scale_x_log10()` or `+ scale_y_log10()`.

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



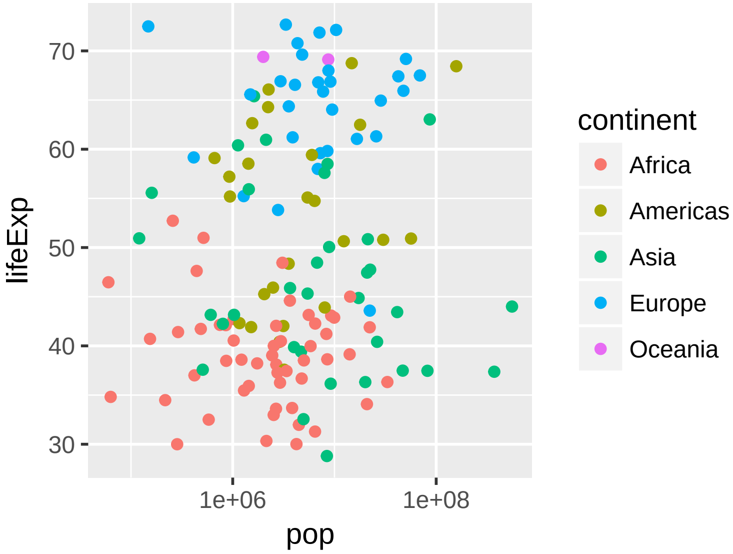
```r
ggplot(gapminder_1952, aes(x=pop, y=gdpPercap)) +
  geom_point() +
  scale_x_log10() +
  scale_y_log10()
```



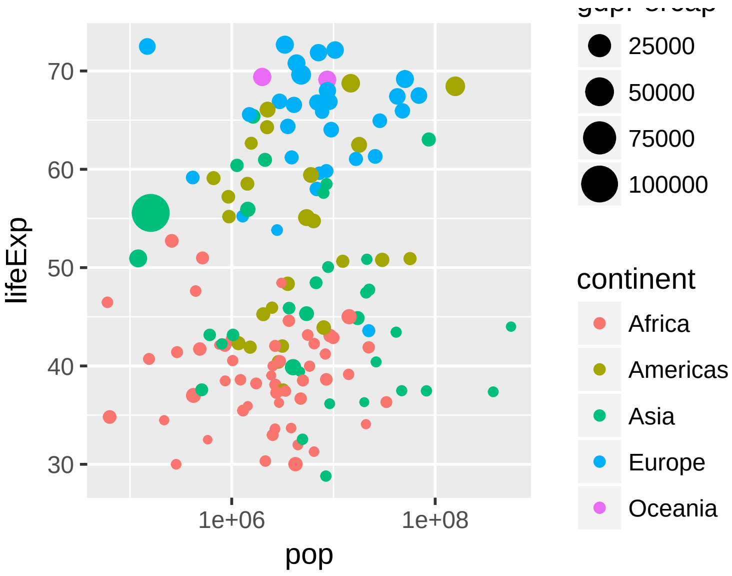
3. Aesthetics

You can add more than just `x` and `y` in the `aes()` parameter in `ggplot`. For example, you can add `color` which is the colour of the data points, and `size` which is their size. For colour, ggplot automatically adds a legend!

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



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

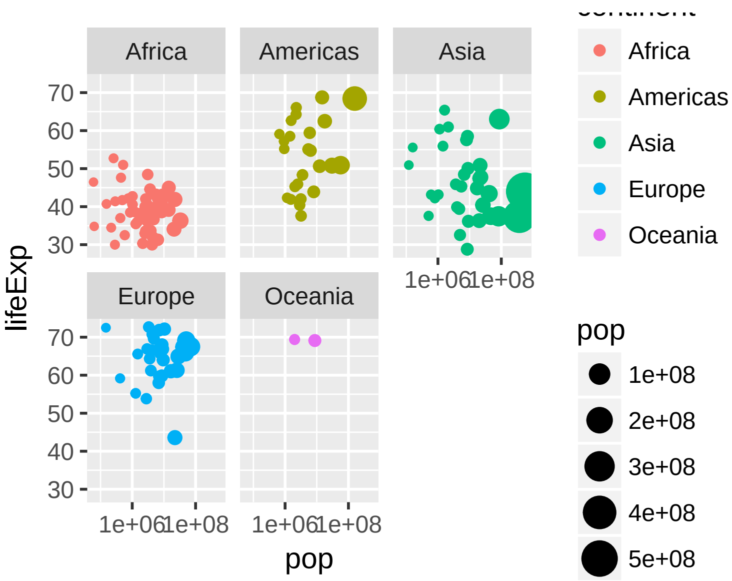


Question: do `color` and `size` only matter when you use the scatter plot? Question: is it possible to modify the legend (say represent the `gdpPercap` values in log scale?)

4. Faceting

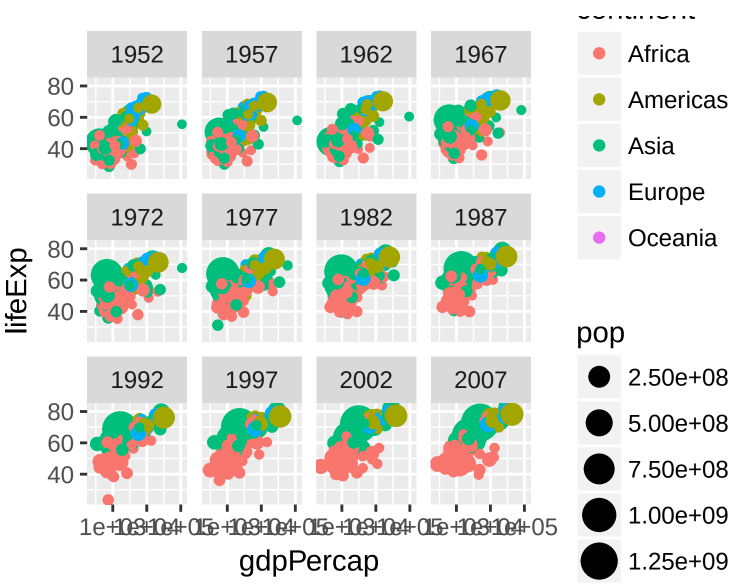
Faceting refers to the practice of making multiple plots within the same figure. In `ggplot` you can make multiple plots according to their indices in the dataset. To facet a plot, we use `+ facet_wrap(~#indexingparameter)`.

```
# Here we break down the analysis of population vs life expectancy according to contin

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



Note that we can apply the `color` and the `facet_wrap` functions to the same parameters. The result is obviously that all the colours go into the same subplot.

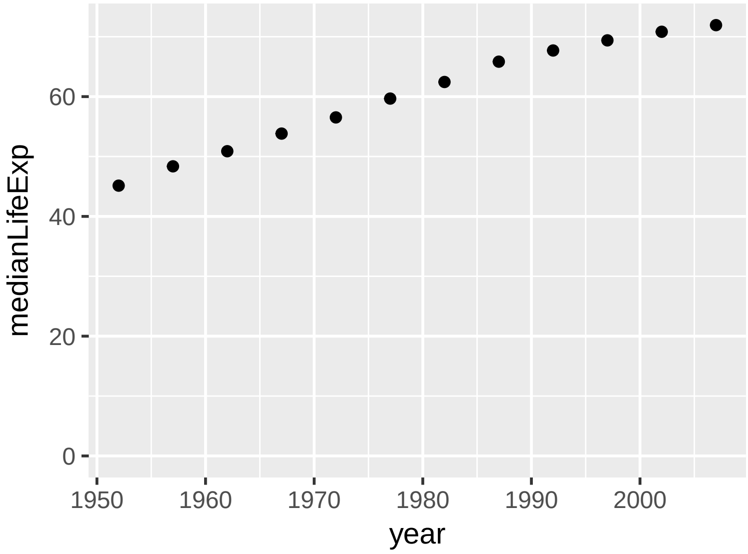
```
# Scatter plot comparing gdpPercap and lifeExp, with color representing continent
# and size representing population, faceted by year
ggplot(gapminder, aes(x=gdpPercap, y=lifeExp, color = continent, size = pop)) +
geom_point() +
scale_x_log10() +
facet_wrap(~ year)
```



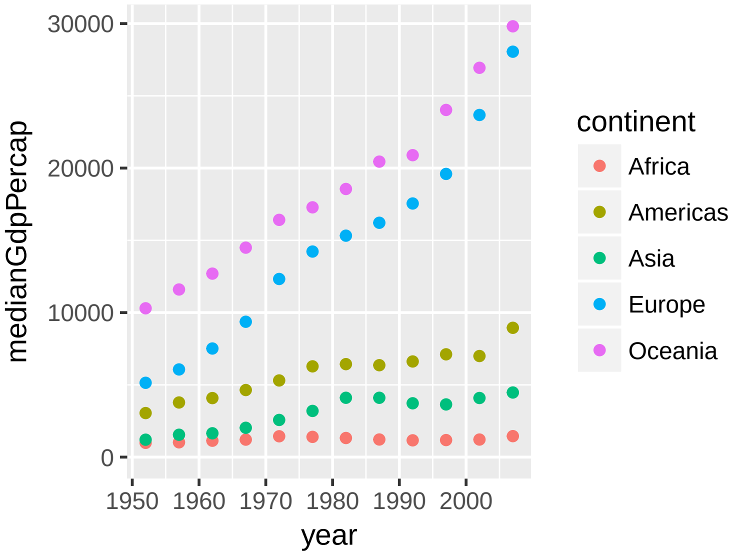
5. Axes limits

Set axes limits using `+ expand_limits(y=#value)`

```r
# Here we'll visualize some summarized data
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)
```



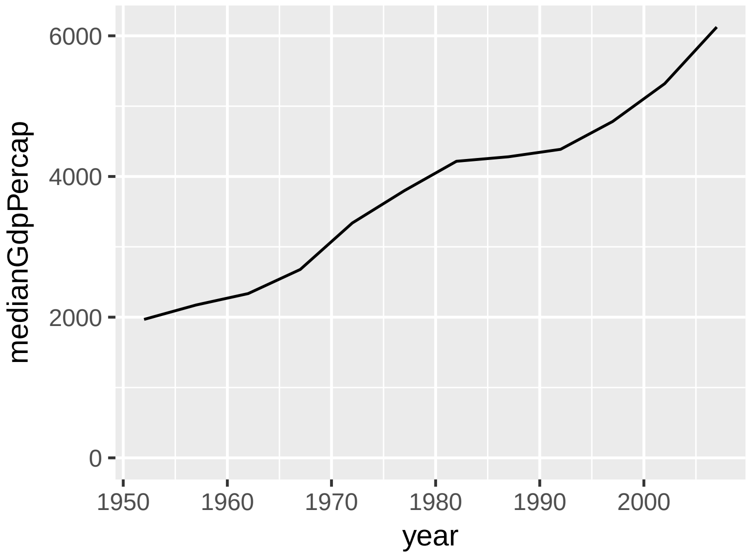
```r
# Summarize medianGdpPercap within each continent within each year: by_year_continent
by_year_continent <- gapminder %>%
 group_by(year, continent) %>%
 summarize(medianGdpPercap = median(gdpPercap))

# 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)
```



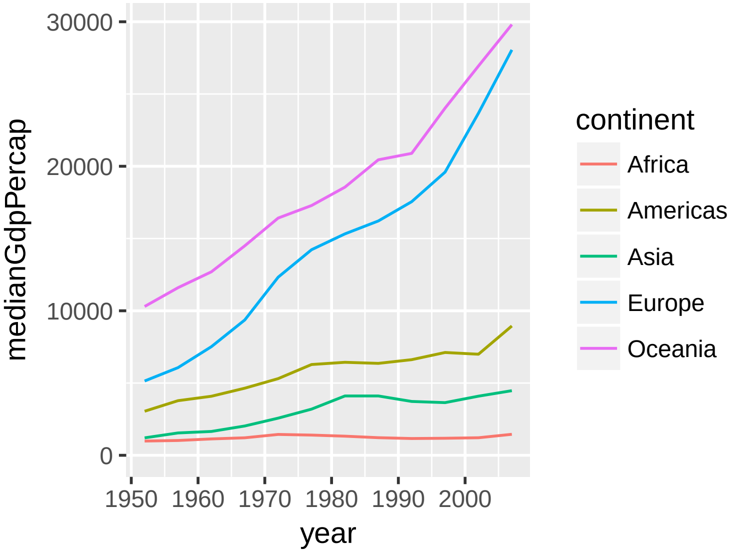
6. Line plot

To make a Line plot, use `+ geom_line()`

```
# 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)
```



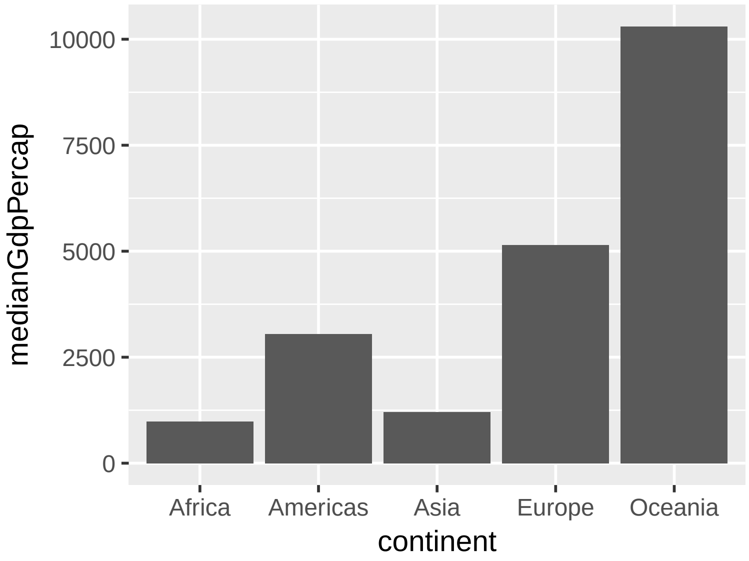
```
# Summarize the median gdpPercap by year & continent, save as by_year_continent
by_year_continent <- gapminder %>%
  group_by(year, continent) %>%
  summarize(medianGdpPercap = median(gdpPercap))

# 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)
```



7. Bar plot

Use + `geom_col()`

```
# 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()
```

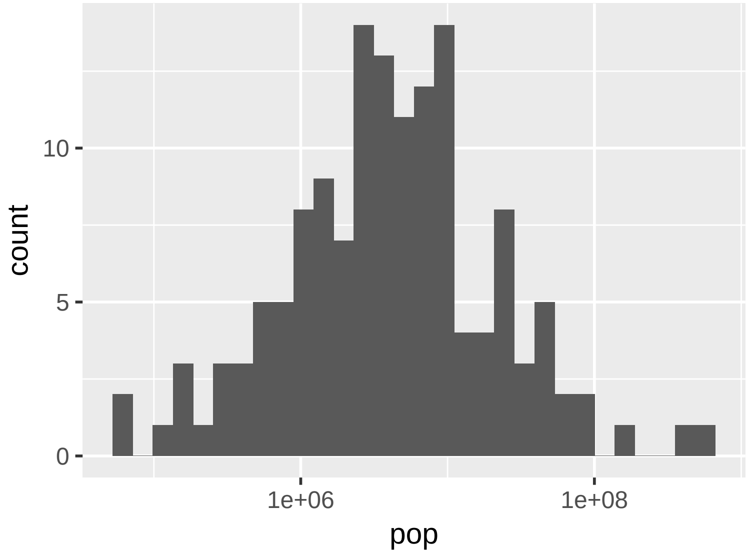


8. Histogram

Use `+ geom_histogram(binwidth=#setwidth)`. Note that for hist, you need only specify `x` in `aes()`.

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

# Create a histogram of population (pop), with x on a log scale

ggplot(gapminder_1952, aes(x=pop)) +
scale_x_log10()+
geom_histogram()
```

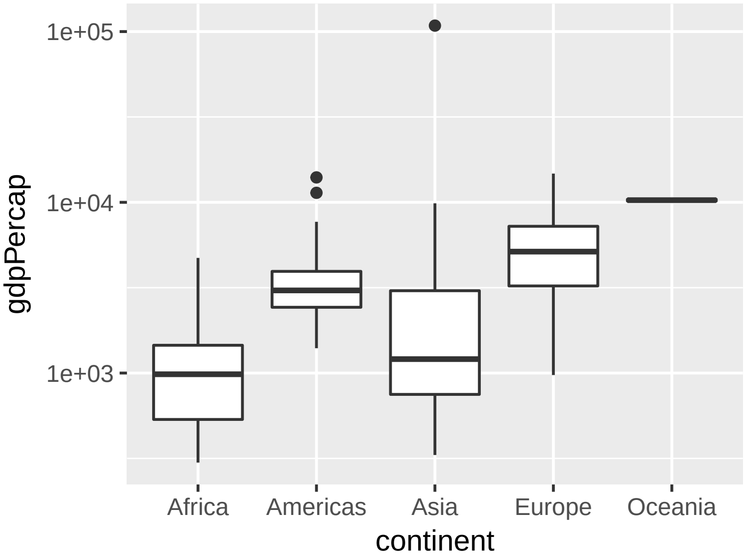
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



9. Box-Whisker plot

Use `+ geom_boxplot()`

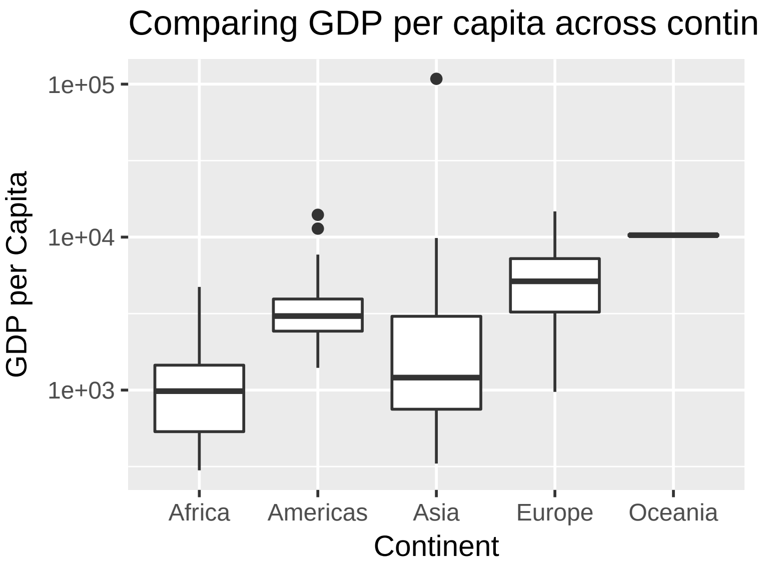
```
gapminder_1952 <- gapminder %>%
  filter(year == 1952)

# Create a boxplot comparing gdpPercap among continents

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



10. Adding title and x and y labels

For title use `+ ggtitle()` and for labels use `labs(x=#namehere, y=#namehere)`

```
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") +
  labs(x="Continent", y="GDP per Capita")
```


Comparing GDP per capita across contin
GDP per Capita
1e+05
1e+04
1e+03
Africa
Americas
Asia
Europe
Oceania
Continent