DSA2101

Essential Data Analytics Tools: Data Visualization

Yuting Huang

Weeks 11 Introduction to `ggplot2`

Plans ahead

Week 12

- Tuesday: Exploring data through visualization.
- Consultations will be moved to Tuesday and Wednesday.
  - **Get your first draft ready** before meeting us.
  - **Aims:** Get some feedback from us on your project.
- No in-person tutorials; no lecture on Friday: Watch videos on Canvas.

Week 13

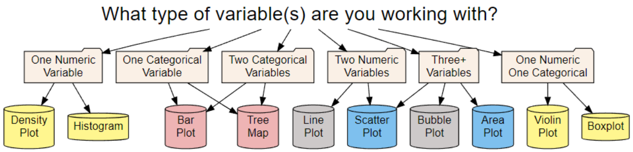
- Tuesday: Exploring data through visualization.
- Friday: Review on Final.
- Group project is due on Friday April 14, 11:59pm.

Re-cap: Choosing the right plot

There are many `geoms` available in the `ggplot2` package.

The choice of which one to use largely depends on two questions:

- What are you trying to communicate?
- What type of variable(s) do you want to show?



Source: Alfredo Hernandez Sanchez

Pre-requisites

`ggplot2` is included in the `tidyverse` package.

```
library(tidyverse)
```

Artwork by Allison Horst

Outline

Bar chart: `geom_col()`

We use bar charts to visualize **categorical** variables. There are two types of bar charts:

- `geom_col()` makes the height of the bar represent **values** in the data.
- `geom_bar()` counts the number of cases in each group and makes the height proportional to **counts**.

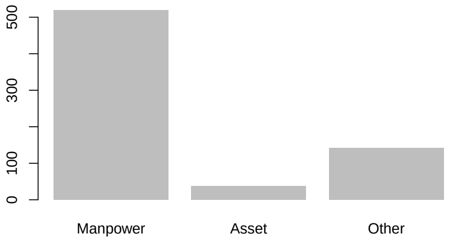
Some of the aesthetics that these `geoms` use are:

- `x`
- `y`
- `alpha`
- `color`
- `fill`
- `linetype`
- `size`

Budget dataset

We used a data set on a company's operation budget in Week 2 and created a bar chart with base R'plotting.

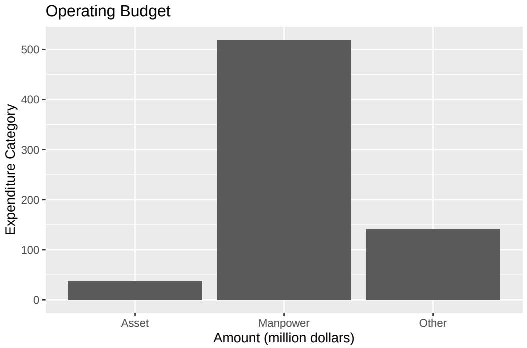
```
budget_cat = c("Manpower", "Asset", "Other")    # create data frame
amount = c(519.4, 38.0, 141.4)
op_budget = data.frame(budget_cat, amount)
barplot(op_budget$amount, border = NA, names.arg = op_budget$budget_cat)
```



Bar charts on budget

Let us recreate the bar plot and add labels using `ggplot2`.

```
ggplot(op_budget, aes(x = budget_cat, y = amount)) +
  geom_col() +
  labs(title = "Operating Budget",
       x = "Amount (million dollars)", y = "Expenditure Category")
```



Bar charts on budget (revised code)

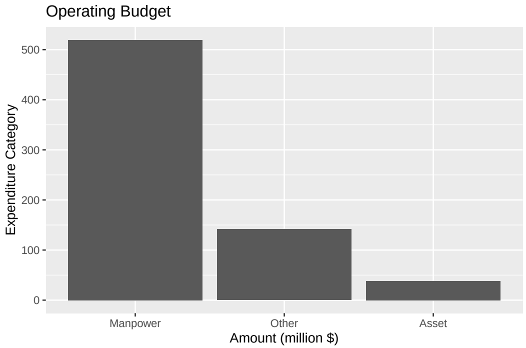
Notice how the expenditure categories have been arranged.

- The category (`budget_cat`) was stored as a factor. When plotted, the categories were arranged in alphabetical order.
- In bar charts, we typically want to arrange them in order of tallest to shortest. We can `reorder()` the levels of the factor using the `mutate()` function.

```
op_budget = op_budget %>%
  mutate(budget_cat = reorder(budget_cat, -amount))
```

Bar charts on budget (revised code)

```r
ggplot(op_budget, aes(x = budget_cat, y = amount)) +
  geom_col() +
  labs(title = "Operating Budget",
       x = "Amount (million $)", y = "Expenditure Category")
```



IMDA data

Now let us turn to the IMDA data we saw in Week 4. The data set contains several variables:

- age group
- year
- activity
- percentage of group who ever participated in that activity

With `ggplot2`, we can easily plot the data.

- Read in the data:

```
imda_data = readRDS("../data/imda.rds")
str(imda_data)
```

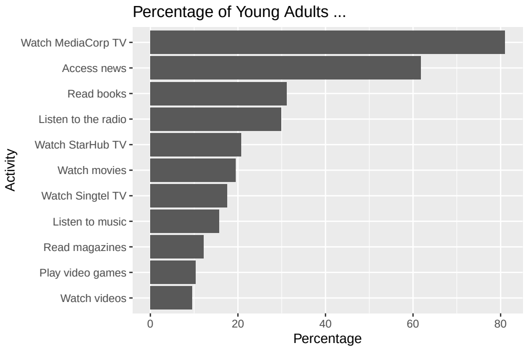
```
## 'data.frame':    210 obs. of  6 variables:
##  $ ever_used     : chr  "97.1" "32.9" "39.5" "30.9" ...
##  $ age           : chr  "15-19" "15-19" "15-19" "15-19" ...
##  $ sample_size   : chr  "161" "161" "161" "161" ...
##  $ year          : chr  "2013" "2013" "2013" "2013" ...
##  $ _id           : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ media_activity: chr  "Watch MediaCorp TV" "Watch Singtel TV" "Watch StarH
```

- Extract responses from the 2015 survey for young adults aged 20–29:

```
young_adults = filter(imda_data, age == "20-29", year == 2015) %>%
  mutate(pct = as.numeric(ever_used),
         media_activity = reorder(media_activity, pct))
```

IMDA data in `geom_col()`

```r
ggplot(young_adults, aes(x = media_activity, y = pct)) +
  geom_col() +
  coord_flip() +
  labs(title = "Percentage of Young Adults ...",
       x = "Activity", y = "Percentage")
```



IMDA data, facets

Now we focus only on the following three categories:

- Watch MediaCorp TV
- Watch StarHub TV
- Watch Singtel TV

We want to study how the proportion of people who have ever used a media device for these activities vary with age group and year.

```
tv = imda_data %>%
  filter(media_activity %in% c("Watch MediaCorp TV",
                               "Watch StarHub TV", "Watch Singtel TV")) %>%
  mutate(pct = as.numeric(ever_used),
         age = reorder(as.factor(age), as.numeric(substr(age, 1, 2))))
```

IMDA data, facets

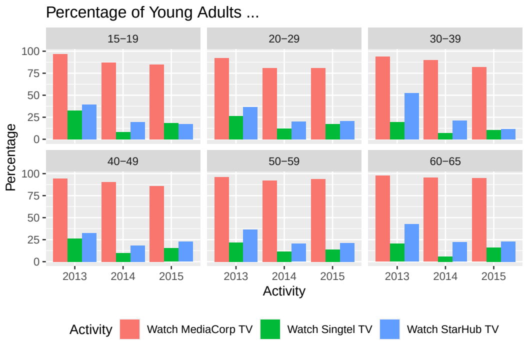
We can use facets for a factor variable to split our data into subsets, according to levels of the variable.

- With a `facet_wrap()` layer, one sub-plot is created for each level of the factor. This allows us to make comparisons across and within the sub-plots easily.

```r
p1 = ggplot(tv, aes(x = year, y = pct)) +
  geom_col(aes(fill = media_activity),
           position = "dodge") +
  facet_wrap(~ age) +
  labs(title = "Percentage of Young Adults ...",
       x = "Activity", y = "Percentage", fill = "Activity") +
  theme(legend.position = "bottom")
```

IMDA data, facets

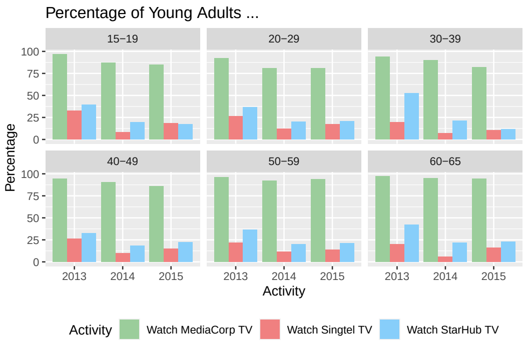
```
p1
```



IMDA data, colors

- Change the color manually by calling the names of color.

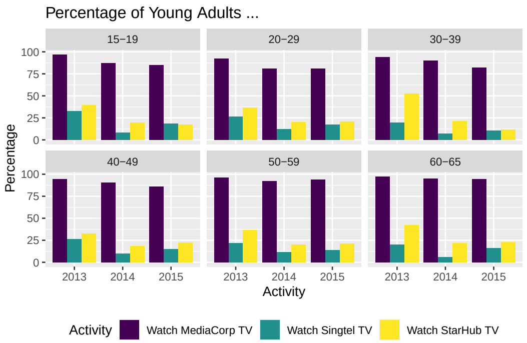
```
p1 + scale_fill_manual(values = c("darkseagreen3","lightcoral","lightskyblue"))
```



IMDA data, colors

- Use a color package, e.g., the **viridis** colorblind-friendly palette.

```
# install.packages("viridis")
library(viridis)
p1 + scale_fill_viridis(option = "viridis", discrete = TRUE)
```



Smoothed line: `geom_smooth()`

Many times, we see scatter plots with a smoothed line.

- Allows the eye to see the patterns in the data.
- Depict the trends in time series data.
- Detect (nonlinear) relationships between variables.

There are several types of smoothers, using different criteria to fit the lines of best fit. We shall study just a couple of them in brief detail:

- linear regression models
- loess smoother

Aesthetics

Some aesthetics that this geom understands are

- `x`
- `y`
- `alpha`
- `color`
- `fill`
- `linetype`

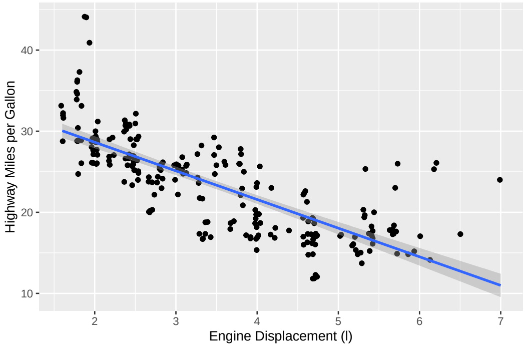
`mpg` data, linear smoother

Recall the scatter plot we made to understand the relationship between `displ` and `hwy`.

- Let us now add a smooth linear regression model (`lm`) line to the data.
- In the code on the next page, notice how the variables are mapped only once, within the `ggplot()` call.

mpg data, linear smoother

```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_point(position = "jitter") +
  geom_smooth(method = "lm") +
  labs(x = "Engine Displacement (l)", y = "Highway Miles per Gallon")
```



Linear regression smoother

`method = lm` invokes a simple linear regression smoother.

- The blue line is the line of best fit; the gray regions represent 95% confidence intervals for the mean.
- The line does not appear to be suitable for this data, which have some non-linearity.

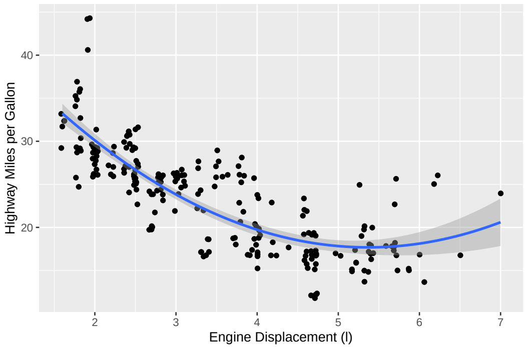
To allow for nonlinearity, we have a couple of options:

1. A higher-order polynomial term in the linear regression, or
2. A `loess` smoother – a nonparametric method that uses local weighted regression to fit a smooth curve.

mpg data, higher order polynomials

Fitting a quadratic function:

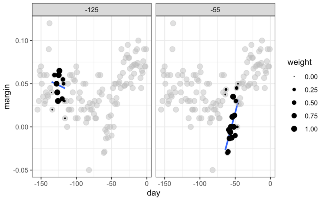
```r
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_point(position = "jitter") +
  geom_smooth(method = "lm", formula = y ~ poly(x, 2)) +
  labs(x = "Engine Displacement (l)", y = "Highway Miles per Gallon")
```

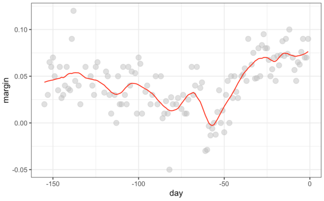


Loess smoother

`method = loess` is used for fitting lines to a scatterplot where sparse data points or weak inter-relationships interfere with our ability to see the overall trend.

- For every x_i in the data, `loess` defines a span (between 0 and 1) and fits a line using data points within that span.
- The fitted value at x_i becomes an estimate $\hat{f}(x_i)$.
- Then it connects all estimated $\hat{f}(x_i)$ and results in a smooth curve.



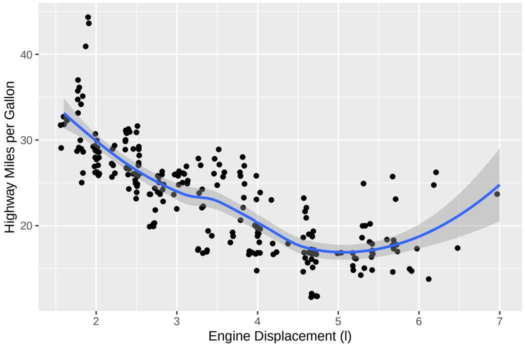


Source: Rafael A. Irizarry

mpg data, loess smoother

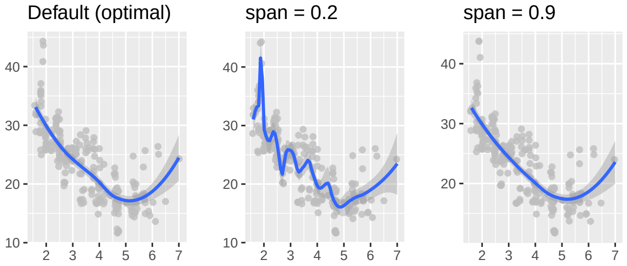
The loess smoother is the default smoother used by geom_smooth().

```
ggplot(mpg, aes(x = displ, y = hwy)) +
  geom_point(position = "jitter") +
  geom_smooth(method = "loess", span = 0.6) +
  labs(x = "Engine Displacement (l)", y = "Highway Miles per Gallon")
```



loess smoother with different span

Smoothness is a relative term. Different `span` gives us different estimates.



`mpg` data, loess smoother by group

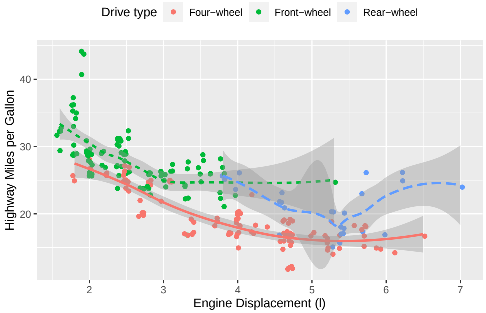
The new curve reflects the presence of several cars that have large engines and efficient highway mileage.

Now suppose that we want to study how this relationship varies with the drive type:

- Front-wheel drive, rear-wheel drive, and four-wheel drive

mpg data, loess smoother by group

```
ggplot(mpg, aes(x = displ, y = hwy, col = drv)) +
  geom_point(position = "jitter") +
  geom_smooth(aes(linetype = drv),
              method = "loess", show.legend = FALSE) +
  labs(x = "Engine Displacement (l)", y = "Highway Miles per Gallon") +
  scale_color_discrete(name = "Drive type",
                       labels = c("Four-wheel", "Front-wheel", "Rear-wheel")) +
  theme(legend.position = "top")
```



Other smoothers

The `loess` smoother is computationally intensive. So when the number of data points is large (≥ 1000 observations), `ggplot()` will use a generalized additive model by default.

- Other smoothers can be used to model binary data (e.g., logistic regression).
- We won't go further into these. Once you take a class of them, you will be able to use them with confidence.

Rug plot: `geom_rug()`

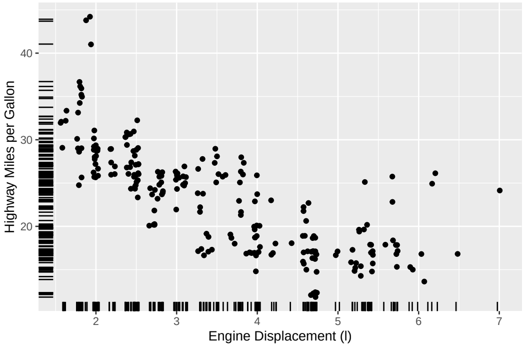
A **rug plot** is a compact visualization designed to supplement a 2D display with **marginal distributions**.

The aesthetics that it takes are:

- `x`
- `y`
- `alpha`
- `color`
- `group`
- `linetype`
- `size`

mpg rug plot

```r
ggplot(mpg, aes(x = displ, y = hwy)) +
  geom_point(position = "jitter") +
  geom_rug(position = "jitter") +
  labs(x = "Engine Displacement (l)", y = "Highway Miles per Gallon")
```



Boxplot: `geom_boxplot()`

Boxplots is a standardized way of displaying the distribution of a continuous variable based on a five-number summary.

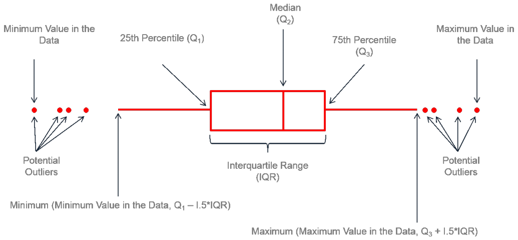
- Minimum
- Lower quartile (Q1)
- Median
- Upper quartile (Q3)
- Maximum

Boxplot description

Useful for identifying potential **outliers**.

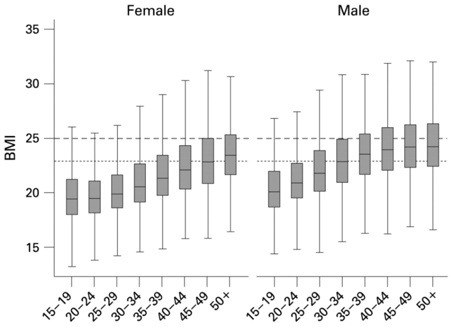
- An outlier is an observation that is very different from the majority of the data.
- Tukey's definition of an outlier: anything outside the range of

$$[Q_1 - 1.5 \times (Q_3 - Q_1), Q_3 + 1.5 \times (Q_3 - Q_1)]$$


Median (Q2)
Minimum Value in the Data
25th Percentile (Q1)
75th Percentile (Q3)
Maximum Value in the Data
Potential Outliers
Interquartile Range (IQR)
Potential Outliers
Minimum (Minimum Value in the Data, Q1 – 1.5*IQR)
Maximum (Maximum Value in the Data, Q3 + 1.5*IQR)

Boxplot description

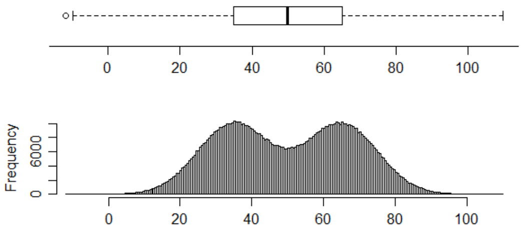
Boxplot is also useful for comparing groups with respect to their "center" and "spread".



Boxplot vs. histogram

However, boxplot cannot not portray certain features of a distribution, such as distinct values and gaps in the data.

- If a distribution is **unimodal**, a box plot does gives an indication about the skewness of a distribution.
- If it is **multi-modal**, then boxplot cannot tell us where the peaks are.



Aesthetics

The essential aesthetics that it takes are:

- `x`
- `lower`
- `upper`
- `middle`
- `ymin`
- `ymax`

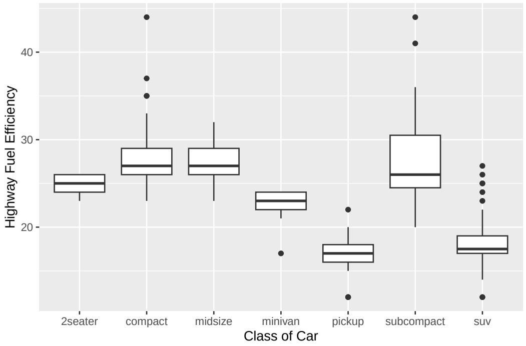
Other aesthetics include

- `color`
- `fill`

mpg boxplot

Let us return to the mpg dataset, and consider how hwy varies with the class of the vehicle.

```
ggplot(mpg, aes(x = class, y = hwy)) +
  geom_boxplot() +
  labs(x = "Class of Car", y = "Highway Fuel Efficiency")
```



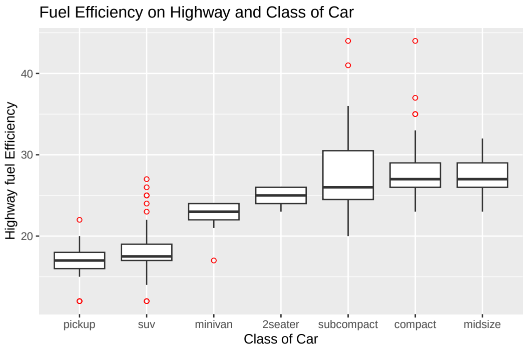
`mpg` boxplot (revised code)

- `class` is an unordered factor. Hence the levels are ordered alphabetically when plotting.
- We can `reorder()` them according to the median in each individual boxplot.

```
mpg2 = mpg %>%
  mutate(class = reorder(class, hwy, FUN = median))
```

mpg boxplot (revised code)

```
ggplot(data = mpg2, mapping = aes(x = class, y = hwy)) +
  geom_boxplot(outlier.color = "red", outlier.shape = 1) +
  labs(title = "Fuel Efficiency on Highway and Class of Car",
       y = "Highway fuel Efficiency", x = "Class of Car")
```



More on outliers

Outliers are very common data issues. For example,

- An old monitoring device may read out nonsensical measurements before completely failing.
- Human errors is also a source of outliers, particularly when data entry is done manually.

Now we discuss formally the approaches that help detect them, summaries that take into account their presence, and ways to remove them.

Example

Let us use the heights dataset on individual earning and heights:

```
heights = read.csv("../data/heights.csv",
                   header = TRUE, stringsAsFactors = TRUE)
```

Compute the mean and median for `earn`:

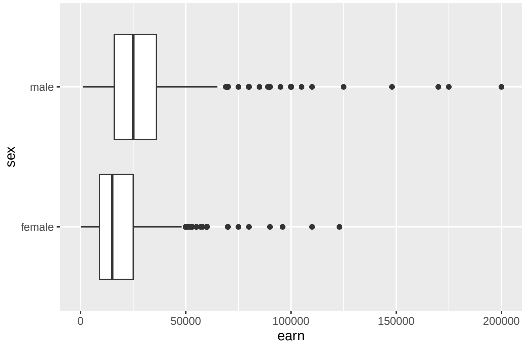
```
heights %>%
  group_by(sex) %>%
  summarize(avg = mean(earn), med = median(earn))
```

```
## # A tibble: 2 x 3
##   sex        avg   med
##   <fct>    <dbl> <dbl>
## 1 female 18280. 15000
## 2 male   29786. 25000
```

Detect outliers

A comparison between `avg` and `med` shows that the earnings data is likely to be right skewed. We can confirm this by creating a boxplot.

```
ggplot(heights, mapping = aes(x = sex, y = earn)) +
  geom_boxplot() +
  coord_flip()
```



We can see some rather extreme values. To see these values, we can quickly look at the largest values.

```
heights %>%
  group_by(sex) %>%
  arrange(desc(earn), .by_group = TRUE) %>% top_n(3, earn)
```

```
## # A tibble: 6 x 6
## # Groups:   sex [2]
##      earn height sex       ed   age race 
##     <dbl>  <dbl> <fct>  <int> <int> <fct>
## 1 123000   61.4 female    14    58 white
## 2 110000   66.3 female    18    48 other
## 3  96000   63.1 female    14    27 white
## 4 200000   69.7 male      18    34 white
## 5 175000   70.6 male      16    48 white
## 6 170000   71.0 male      18    45 white
```

Outliers

Based on Tukey's definition, the following are outliers

```
heights %>%
  group_by(sex) %>%
  mutate(max_earn = quantile(earn, 0.75) + 1.5 * IQR(earn),
         min_earn = quantile(earn, 0.25) - 1.5 * IQR(earn)) %>%
  filter(earn > max_earn | earn < min_earn) %>%
  select(earn, sex) %>% head(6)
```

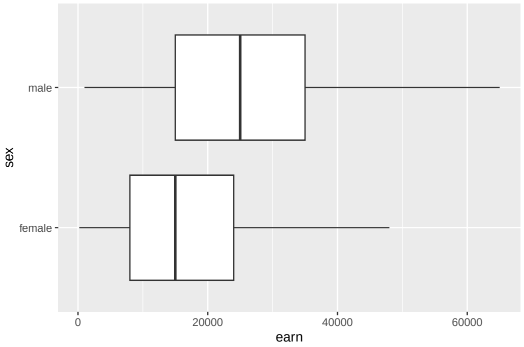
```
## # A tibble: 6 x 2
## # Groups:   sex [2]
##    earn sex
##   <dbl> <fct>
## 1 60000 female
## 2 50000 female
## 3 51000 female
## 4 75000 male
## 5 60000 female
## 6 70000 female
```

Remove outliers

To remove the outliers, we use `filter()` to include only those within the Tukey's range.

```
heights2 = heights %>%
  group_by(sex) %>%
  mutate(max_earn = quantile(earn, 0.75) + 1.5 * IQR(earn),
         min_earn = quantile(earn, 0.25) - 1.5 * IQR(earn)) %>%
  filter(earn <= max_earn & earn >= min_earn) %>% ungroup()

ggplot(data = heights2) +
  geom_boxplot(aes(x = earn, y = sex))
```



2D Histograms

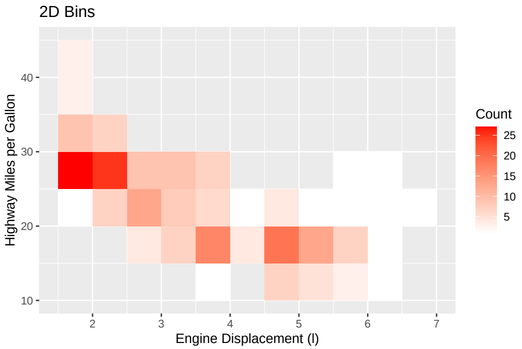
We studied histograms for the distribution of a single continuous variable.

2D histograms depict the distribution of **two** continuous variables.

- It divides the x-y plane into rectangles.
- Then counts the number of cases falling in each rectangle, and maps the counts to the rectangle's `fill`.
  - It is the `fill` aesthetic that represent counts, using a **continuous** color scheme.
- Serves as a useful alternative to `geom_point()` in the presence of **overplotting**.

mpg, 2D histograms

```
p1 = ggplot(mpg, aes(x = displ, y = hwy)) +
  geom_bin2d(binwidth = c(0.5, 5)) +
  scale_fill_gradient(name = "Count", low = "white", high = "red") +
  labs(title = "2D Bins",
       x = "Engine Displacement (l)", y = "Highway Miles per Gallon")
p1
```



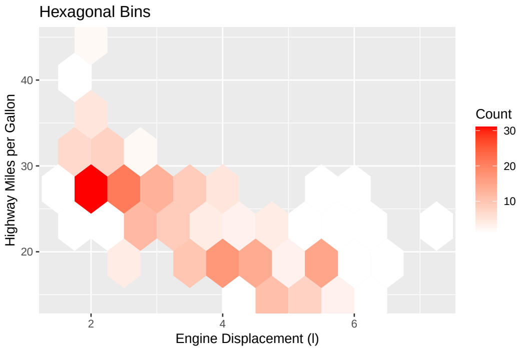
2D histograms with hexagonal bins

The tiles in `geom_bin2d()` create a “block” visual effect, so it is often swapped out for hexagonal bins.

- `geom_hex()` divides the x-y plane into with hexagonal bins.
- Then counts the number of cases and map it to the `fill` of the hexagons.

mpg, 2D histograms with hexagonal bins

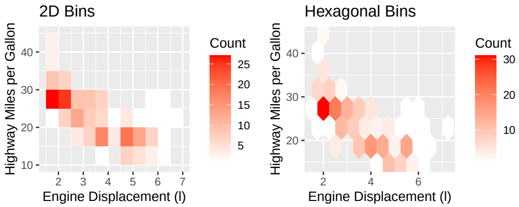
```
p2 = ggplot(mpg, aes(x = displ, y = hwy)) +
  geom_hex(binwidth = c(0.5, 5)) +
  scale_fill_gradient(name = "Count", low = "white", high = "red") +
  labs(title = "Hexagonal Bins",
       x = "Engine Displacement (l)", y = "Highway Miles per Gallon")
p2
```



Arranging several plots

- When we make base-R plots, we use `mfrow` within `par()` to create side-by-side plots.
- To do similar things with `ggplot()`, we can use the `gridExtra` package.

```r
#install.packages("gridExtra")
library(gridExtra)
grid.arrange(p1, p2, nrow = 1)
```



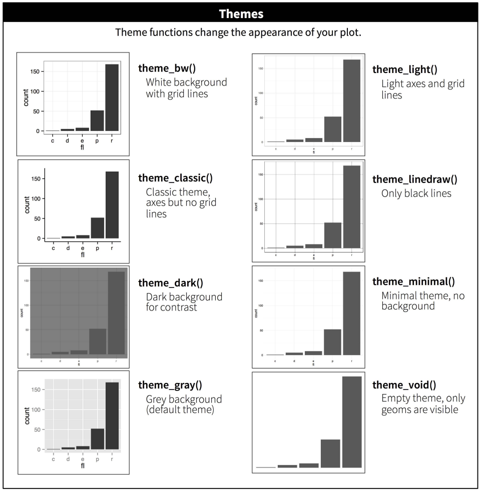
`ggplot2` themes

The default background of a `ggplot2` graph is always light gray. There are several reasons that the designers have:

1. White grid lines are visible, yet easy to tune out, keeping the data prominent.
2. The grey background gives a similar color to typographic text, preventing it from jumping out.
3. It creates a continuous field of color which ensures that the plot is perceived as a single visual entity.

You may agree or disagree with these points. If you would like to alter some of these elements, they can be done by selecting a different theme for your plot.

ggplot2 themes



Themes

Theme functions change the appearance of your plot.

theme_bw()
White background with grid lines

theme_light()
Light axes and grid lines

theme_classic()
Classic theme, axes but no grid lines

theme_linedraw()
Only black lines

theme_dark()
Dark background for contrast

theme_minimal()
Minimal theme, no background

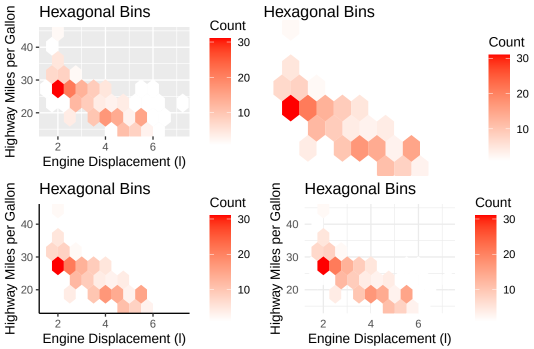
theme_gray()
Grey background (default theme)

theme_void()
Empty theme, only geoms are visible

ggplot2 themes

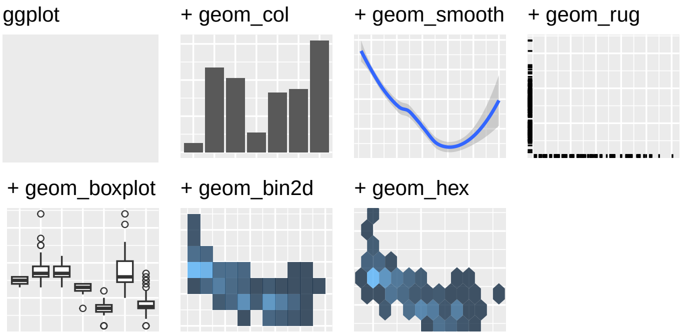
```r
p3 = p2 + theme_void()
p4 = p2 + theme_classic()
p5 = p2 + theme_minimal()

grid.arrange(p2, p3, p4, p5, nrow = 2)
```



Second summary on `ggplot2`

Summary on some of the `geoms` we learned this week:



Layered grammar of graphics

Our initial template can be extend to:

```
ggplot(data = <DATA>) +
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>),
                  stat = <STAT>,
                  position = <POSITION>) +
  <COORDINATE_FUNCTION> +
  <FACET_FUNCTION>
```

- The 7 parameters in the template compose a formal grammar of graphics.
- Any plot can be described as a combination of the 7 parameters.

Common layers

- To modify a particular geom, work with the `geom_*()` layer.
- To modify the mapping of a particular aesthetic, look into one of the `scale_*_*()` layers.
  - For instance, to modify the colors used in the `fill` aesthetic, look up `scale_fill_*()`
- To modify low-level elements of a graph, such as tick marks, tick positions, look up the `theme()` layer.

Which geom to use?

The following thought process may help:

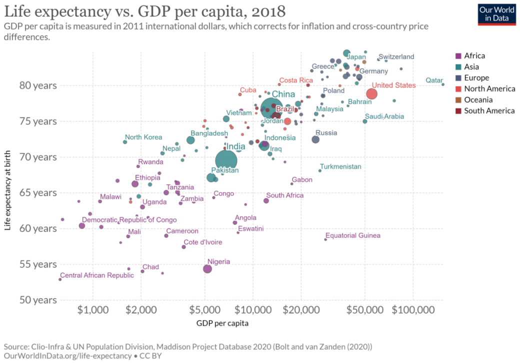
- Think first in terms of the question you want to answer.
- What are the types of the variable (numeric, factor, integer, or date)? This would narrow down the choice of `geoms`.
- Then think about the audience of your chart. How detailed a comparison do you think they will need to make?

Create a basic plot and look closely at it. Think about whether the question can be answered. If not, come up with some ideas:

- Do we need to add another variable?
- Do we need to transform the data?
- Do we need to add a smoother?
- Why are there so many outliers? Do they follow some pattern that could be explained by another variable?

Case study: New insights on poverty

In Week 9, we saw a graph on the relationship between wealth and health across countries.



New insights on poverty

The `gapminder` data set is available in the `dslabs` package.

- The chart we create today won't be as nice as the original, but it will be a good starting point for making more advanced graphics.

```
library(ggrepel)
library(ggthemes)
library(dslabs)
data(gapminder)
```

When starting with a new data set, it is useful to produce **numerical summaries** to overview the variables.

```
summary(gapminder)
```

```
##                    country           year      infant_mortality life_expectancy
##  Albania               :   57   Min.   :1960   Min.   :  1.50   Min.   :13.20
##  Algeria               :   57   1st Qu.:1974   1st Qu.: 16.00   1st Qu.:57.50
##  Angola                :   57   Median :1988   Median : 41.50   Median :67.54
##  Antigua and Barbuda:   57   Mean   :1988   Mean   : 55.31   Mean   :64.81
##  Argentina             :   57   3rd Qu.:2002   3rd Qu.: 85.10   3rd Qu.:73.00
##  Armenia               :   57   Max.   :2016   Max.   :276.90   Max.   :83.90
##  (Other)               :10203                  NA's   :1453
##    fertility        population              gdp                   continent
##  Min.   :0.840   Min.   :3.124e+04   Min.   :4.040e+07   Africa  :2907
##  1st Qu.:2.200   1st Qu.:1.333e+06   1st Qu.:1.846e+09   Americas:2052
##  Median :3.750   Median :5.009e+06   Median :7.794e+09   Asia    :2679
##  Mean   :4.084   Mean   :2.701e+07   Mean   :1.480e+11   Europe  :2223
##  3rd Qu.:6.000   3rd Qu.:1.523e+07   3rd Qu.:5.540e+10   Oceania : 684
##  Max.   :9.220   Max.   :1.376e+09   Max.   :1.174e+13
##  NA's   :187     NA's   :185         NA's   :2972
##              region
##  Western Asia   :1026
##  Eastern Africa : 912
##  Western Africa : 912
##  Caribbean      : 741
##  South America  : 684
##  Southern Europe: 684
##  (Other)        :5586
```

Missing values

To examine the relationship between health and wealth, we need country-level data on:

- Population: `population`
- GDP per capita: `population/gdp`

Both variables contain missing values. Data on GDP were updated only up to the year of 2011.

$\Rightarrow$ We will restrict our attention to observations in 2011.

Graphical summaries

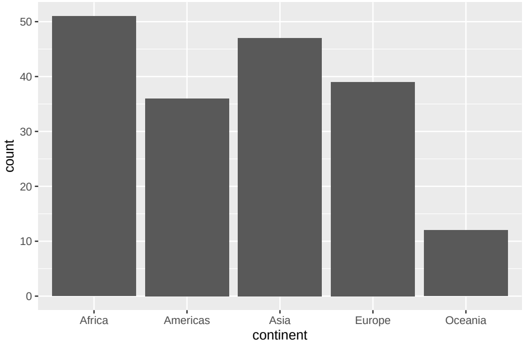
We continue our exploration with **graphical summaries**.

- Bar plot on categorical variables.
- Histogram, density plot, and eCDF for continuous variables (across groups).
- Box plot for continuous variables across groups.
- Line plot is suitable for time-series data.
- Scatter plot for two continuous variables.

Bar plot

Bar plots are often used to visualize the distribution of **categorical** variables, like `continent` in this data set.

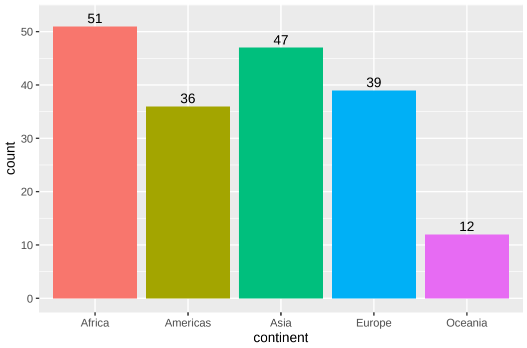
```r
gapminder %>% filter(year == 2011) %>%
  ggplot(aes(x = continent)) +
  geom_bar()
```



Bar plot

To make this more colorful, we map the `fill` attribute to `continent`, and add text labels to the bar chart.

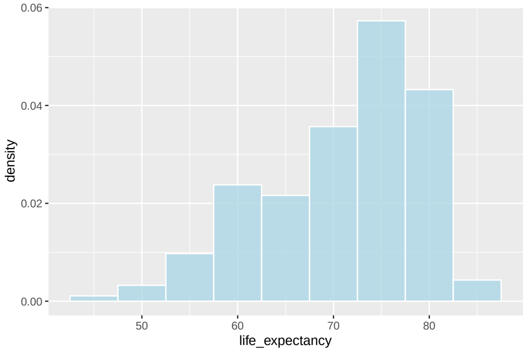
```
gapminder %>% filter(year == 2011) %>%
  ggplot(aes(x = continent, fill = continent)) +
  geom_bar(show.legend = FALSE) +
  geom_text(aes(label = ..count..), stat = "count", nudge_y = 1.5)
```



Histogram and density plots

Histogram and density plots are commonly used to visualize the distribution of a continuous variable.

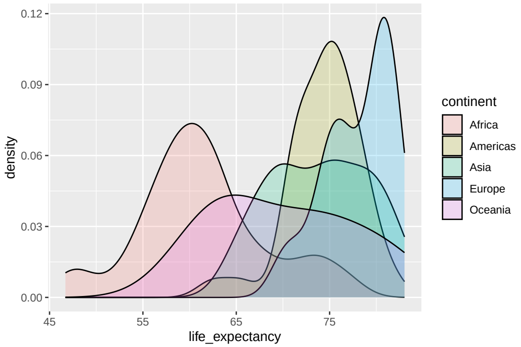
```r
gapminder %>% filter(year == 2011) %>%
  ggplot(aes(x = life_expectancy)) +
  geom_histogram(aes(y = ..density..), binwidth = 5,
                 color = "white", fill = "lightblue", alpha = 0.8)
```



Notice that the distribution of `life_expectancy` is slightly **skewed to the left**.

- We can further examine this by adding another attribute `fill = continent` inherited in the aesthetics mapping of `geom_density()`.

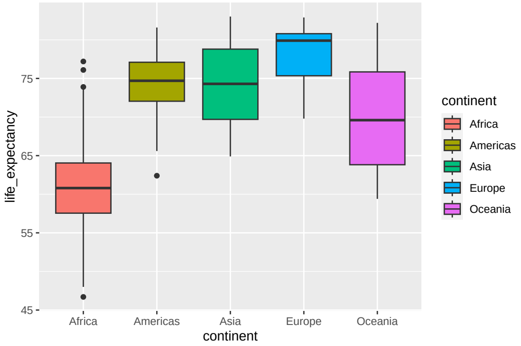
```
gapminder %>% filter(year == 2011) %>%
  ggplot(aes(x = life_expectancy, fill = continent)) +
  geom_density(alpha = 0.2)
```



Box plots by group

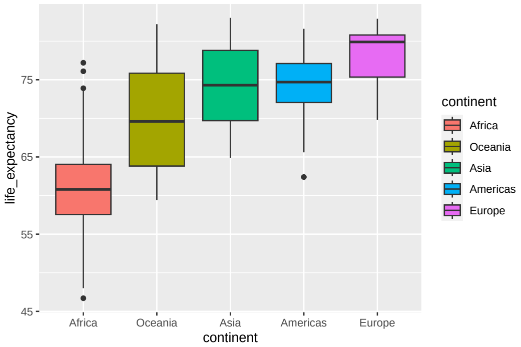
Alternatively, we can view the distributions using a box plot.

```
gapminder %>% filter(year == 2011) %>%
  ggplot(aes(x = continent, y = life_expectancy, fill = continent)) +
  geom_boxplot()
```



The continents on the horizontal axis are ordered alphabetically by default. Rearrange them so that they are sorted by median life expectancy.

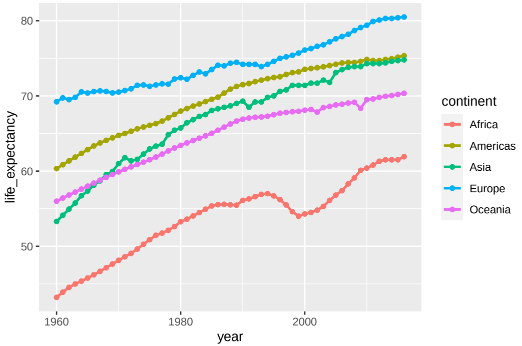
```
gapminder %>% filter(year == 2011) %>%
  mutate(continent = reorder(continent, life_expectancy, FUN = median)) %>%
  ggplot(aes(x = continent, y = life_expectancy, fill = continent)) +
  geom_boxplot()
```



Trends over time

Take a look at the trend of life expectancy over time:

```r
gapminder %>%
  group_by(continent, year) %>%
  summarize(life_expectancy = median(life_expectancy)) %>%
  ggplot(aes(x = year, y = life_expectancy, color = continent)) +
  geom_line(size = 1) +
  geom_point(size = 1.5)
```



Health and wealth

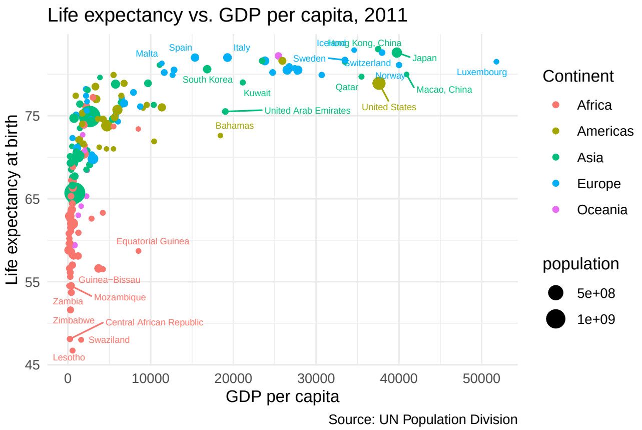
Our first attempt to making the scatter plot using 2011 data.

- Prepare the data and then create the plot:

```r
# Prepare data plotting
df = gapminder %>%
  mutate(gdp_pc = gdp/population) %>%
  filter(year == 2011 & !is.na(gdp_pc))

# Create the plot
p = ggplot(data = df, aes(x = gdp_pc, y = life_expectancy, color = continent))
  geom_point(aes(size = population))  +
  geom_text_repel(aes(label = country), show.legend = FALSE, size = 2) +
  labs(title = "Life expectancy vs. GDP per capita, 2011",
       x = "GDP per capita", y = "Life expectancy at birth",
       color = "Continent", caption = "Source: UN Population Division") +
  theme_minimal()
```

```
p
```



Life expectancy vs. GDP per capita, 2011
Life expectancy at birth
GDP per capita
Source: UN Population Division
45
55
65
75
0
10000
20000
30000
40000
50000
Continent
Africa
Americas
Asia
Europe
Oceania
population
5e+08
1e+09
Malta
Spain
Italy
Iceland
Hong Kong, China
Japan
Sweden
Switzerland
Luxembourg
Norway
South Korea
Kuwait
Qatar
Macao, China
United Arab Emirates
United States
Bahamas
Equatorial Guinea
Guinea−Bissau
Mozambique
Zambia
Zimbabwe
Central African Republic
Swaziland
Lesotho

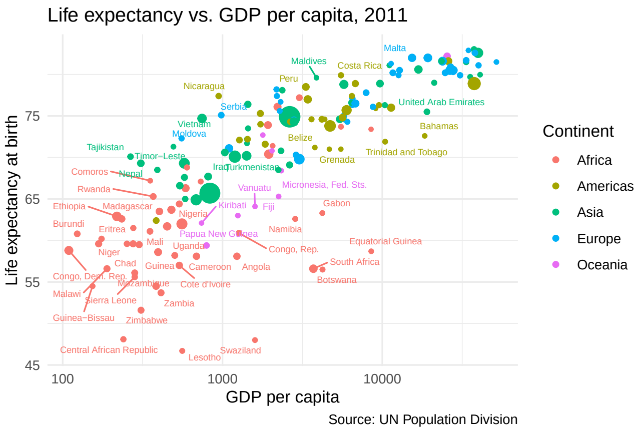
Revise the plot

1. Convert the x-axis into a log scale.
2. Polish the themes and colors.
3. Polish the x and y-axis tick labels.

```r
p2 = ggplot(data = df, aes(x = gdp_pc, y = life_expectancy, color = continent))
  geom_point(aes(size = population)) +
  scale_x_log10() +
  geom_text_repel(aes(label = country), show.legend = FALSE, size = 2) +
  labs(title = "Life expectancy vs. GDP per capita, 2011",
       x = "GDP per capita", y = "Life expectancy at birth",
       color = "Continent", caption = "Source: UN Population Division") +
  theme_minimal() +
  guides(size = "none")
```

Revise the plot

```
p2
```



Continue to revise the plot

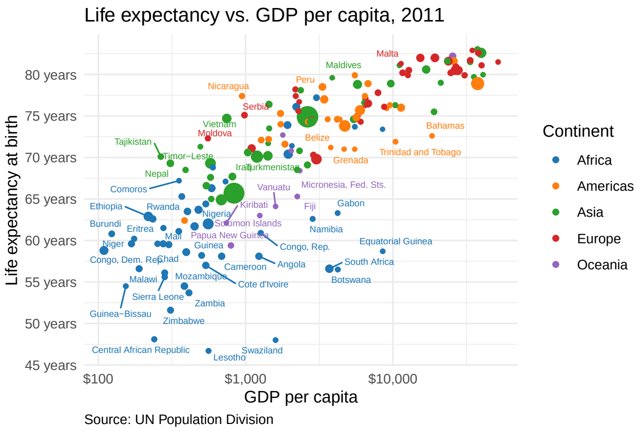
1. Convert the x-axis into a log scale.
2. Polish the themes and colors.
3. Polish the x and y-axis tick labels.

```
p3 = p2 + scale_color_tableau(palette = "Classic 10") +
  scale_x_continuous(trans = "log10",
                     labels = scales::label_dollar(accuracy = 1)) +
  scale_y_continuous(breaks = seq(45, 85, by = 5),
                     labels = scales::label_number(suffix = " years")) +
  theme(plot.caption = element_text(hjust = 0))
```

Read more on Tableau Color Palettes under `ggthemes`.

Final plot

```
p3
```


Life expectancy vs. GDP per capita, 2011
Life expectancy at birth
80 years
75 years
70 years
65 years
60 years
55 years
50 years
45 years
$100
$1,000
$10,000
GDP per capita
Source: UN Population Division
Continent
Africa
Americas
Asia
Europe
Oceania
Malta
Maldives
Peru
Nicaragua
Serbia
Vietnam
Moldova
Tajikistan
Timor−Leste
Nepal
Belize
Bahamas
Trinidad and Tobago
Grenada
Iraq
Turkmenistan
Comoros
Vanuatu
Micronesia, Fed. Sts.
Ethiopia
Rwanda
Kiribati
Fiji
Gabon
Nigeria
Burundi
Eritrea
Solomon Islands
Namibia
Mali
Papua New Guinea
Niger
Guinea
Congo, Rep.
Equatorial Guinea
Congo, Dem. Rep.
Chad
Angola
South Africa
Cameroon
Malawi
Mozambique
Cote d'Ivoire
Botswana
Sierra Leone
Zambia
Guinea−Bissau
Zimbabwe
Central African Republic
Swaziland
Lesotho