**Visualization**

with Prof. Rafael Irizarry

**Jesse's Beast**Course Notes and Screenshots

School Name  
Course Name

Start Date – End Date

**Module No.2**

**Module Title**

Due date:

**Course Title**

with Prof. XXXX

**Module 1: Title**

**Table of Contents**

[with Prof. Rafael Irizarry 0](file:///C:\Users\Jesse's%20Beast\Documents\HarvardX%20classes\Visualization.docx#_Toc17551732)

[1. Introduction to Data Visualization 5](#_Toc17551733)

[1.1.1. Introduction to Distributions 5](#_Toc17551734)

[1.1.2. Data Types 5](#_Toc17551735)

[1.2. Introduction to Distributions 6](#_Toc17551736)

[1.2.1. Describe Heights to ET 6](#_Toc17551737)

[1.2.2. Smooth Density Plots 9](#_Toc17551738)

[1.2.3. Normal Distribution 12](#_Toc17551739)

[1.3. Quantiles, Percentiles, and Boxplots 14](#_Toc17551740)

[1.3.1. Quantile-Quantile Plots 14](#_Toc17551741)

[1.3.2. Percentiles 14](#_Toc17551742)

[1.3.3. Boxplots 15](#_Toc17551743)

[1.4. Exploratory Data Analysis 16](#_Toc17551744)

[1.4.1. Distribution of Female Heights 16](#_Toc17551745)

[2. Introduction to ggplot2 18](#_Toc17551746)

[2.1. Basics of ggplot2 18](#_Toc17551747)

[2.1.1. ggplot 18](#_Toc17551748)

[2.1.2. Graph Components 18](#_Toc17551749)

[2.1.3. Creating a New Plot 19](#_Toc17551750)

[2.2. Customizing Plots 19](#_Toc17551751)

[2.2.1. Layers 19](#_Toc17551752)

[2.2.2. Tinkering 20](#_Toc17551753)

[2.2.3. Scales, Labels, and Colors 20](#_Toc17551754)

[2.2.4. Add-on Packages 21](#_Toc17551755)

[2.2.5. Other Examples 22](#_Toc17551756)

[3. Section 3: Summarizing with dplyr 26](#_Toc17551757)

[3.1. Summarizing with dplyr 26](#_Toc17551758)

[3.1.1. Dplyr 26](#_Toc17551759)

[3.1.2. The Dot Placeholder 27](#_Toc17551760)

[3.1.3. Group By 28](#_Toc17551761)

[3.1.4. Sorting Data Tables 29](#_Toc17551762)

[4. Section 4: Gapminder 32](#_Toc17551763)

[4.1. Introduction to Gapminder 32](#_Toc17551764)

[4.1.1. Case Study: Trends in World Health and Economics 32](#_Toc17551765)

[4.1.2. Gapminder Dataset 32](#_Toc17551766)

[4.1.3. Life Expectancy and Fertility Rates 33](#_Toc17551767)

[4.2. Using the Gapminder Dataset 35](#_Toc17551768)

[4.2.1. Faceting 35](#_Toc17551769)

[4.2.2. Time Series Plots 38](#_Toc17551770)

[4.2.3. Transformations 41](#_Toc17551771)

[4.2.4. Stratify and Boxplot 44](#_Toc17551772)

[4.2.5. Comparing Distributions 48](#_Toc17551773)

[4.2.6. Density Plots 53](#_Toc17551774)

[4.2.7. Ecological Fallacy 56](#_Toc17551775)

[5. Section 5: Data Visualization Principles 59](#_Toc17551776)

[5.1. Data Visualization Principles, Part 1 59](#_Toc17551777)

[5.1.1. Introduction to Data Visualization Principles 59](#_Toc17551778)

[5.1.2. Encoding Data Using Visual Cues 59](#_Toc17551779)

[5.1.3. Know When to Include Zero 60](#_Toc17551780)

[5.1.4. Do Not Distort Quantities 62](#_Toc17551781)

[5.1.5. Order by a Meaningful Value 63](#_Toc17551782)

[5.2. Data Visualization Principles, Part 2 64](#_Toc17551783)

[5.2.1. Show the Data 64](#_Toc17551784)

[5.2.2. Ease Comparisons: Use Common Axes 66](#_Toc17551785)

[5.2.3. Consider Transformations 67](#_Toc17551786)

[5.2.4. Ease Comparisons: Compared Visual Cues Should Be Adjacent 68](#_Toc17551787)

[5.3. Data Visualization Principles, Part 3 69](#_Toc17551788)

[5.3.1. Slope Charts 69](#_Toc17551789)

[5.3.2. Encoding a Third Variable 71](#_Toc17551790)

[5.3.3. Case Study: Vaccines 72](#_Toc17551791)

[5.3.4. Avoid Pseudo and Gratuitous 3D Plots 74](#_Toc17551792)

[5.3.5. Avoid Too Many Significant Digits 75](#_Toc17551793)

**List of Figures**

[Boxplot of propotion of murder by region 6](#_Toc17551660)

[CDF of heights in the class 7](#_Toc17551661)

[Histogram of Heights 8](#_Toc17551662)

[Smooth density plot of Male heights 9](#_Toc17551663)

[histograms of heights with different size bins 9](#_Toc17551664)

[Histogram frequencies of 708 students’ heights 10](#_Toc17551665)

[Histograms of heights with different smoothness degree 10](#_Toc17551666)

[Proportion of values between 65 and 68 11](#_Toc17551667)

[Normal distribution with m=0 and s=1 12](#_Toc17551668)

[Smooth density of Male heights overlaid with the normal distribution 13](#_Toc17551669)

[Histogram and QQ plot of murder rate in the US 15](#_Toc17551670)

[boxplot of the murder rate in the US 15](#_Toc17551671)

[Boxplot of heights of male and female students 16](#_Toc17551672)

[density plot and qq plot of female heights 16](#_Toc17551673)

[Scatterplot of Us gun murders in 2010 18](#_Toc17551674)

[Histogram of Male heights with ggplot 22](#_Toc17551675)

[Density functions of male heights 23](#_Toc17551676)

[QQ plot of Male heights 24](#_Toc17551677)

[3 histogram on the same graphic 25](#_Toc17551678)

[Scatter plot of fertility against life expectancy in 1962 33](#_Toc17551679)

[Fertility against life expectancy in 1962 with continent differenciation 34](#_Toc17551680)

[Scatter plot, Facetted by year and continent, of fertility versus life expectancy 35](#_Toc17551681)

[Scatter plot, Facetted by year, of fertility versus life expectancy 36](#_Toc17551682)

[Scatter plot of life expectancy versus fertility from 1962 to 2012 37](#_Toc17551683)

[Time series of US fertility rate 38](#_Toc17551684)

[Time series curve of the US fertility 38](#_Toc17551685)

[Time series of US, Japan and France fertility between 1960 and 2012 39](#_Toc17551686)

[fertility rate comparison of France, Japan and US between 1960 and 2012 39](#_Toc17551687)

[Comparison of Germany and South Korea life expctency over they years 40](#_Toc17551688)

[Histogram of per day incomes from 1970 41](#_Toc17551689)

[Histogram of the log2 dollar per day in 1970 42](#_Toc17551690)

[Histogram of dollar per day in 1970 with logged scale 43](#_Toc17551691)

[Boxplots of dollar per day by region in 1970 44](#_Toc17551692)

[Readable x label boxplots of dollars per day by regions in 1970 45](#_Toc17551693)

[Boxplots comparison of dollars per day by region in 1970 group by continent 46](#_Toc17551694)

[Log scaled Boxplots comparison of dollars per day by region in 1970 group by continent 47](#_Toc17551695)

[Log scaled Boxplots comparison of dollars per day by region in 1970 group by continent with data points 47](#_Toc17551696)

[Histograms of dollar per day distribution of West and Developing region in 1970 48](#_Toc17551697)

[Histograms comparison of dollar per day distribution of West and Developing region in 1970 and 2010 49](#_Toc17551698)

[Histograms comparison of dollar per day distribution of West and Developing region in 1970 and 2010 without missing data 50](#_Toc17551699)

[Boxplots comparison of regions income distribution between 1970 and 2010 51](#_Toc17551700)

[Boxplots, of 1970 and 2010, comparison of regions income distribution 52](#_Toc17551701)

[Smooth density plots comparison of Developing countries vs West countries between 1970 and 2010 53](#_Toc17551702)

[Smooth density plots comparison of Developing countries vs West countries between 1970 and 2010 with smoothing 54](#_Toc17551703)

[Stacked density plots comparison of countries income between 1970 and 2010 55](#_Toc17551704)

[Region average income vs infant survival rate 57](#_Toc17551705)

[Region income vs infant survival rate 58](#_Toc17551706)

[Pie chart comparison of browser preference between 2000 and 2015 59](#_Toc17551707)

[Barplot chart comparison of browser preference between 2000 and 2015 60](#_Toc17551708)

[Box plot made by FoxNews showing southwest border apprehensions in 2011, 2012 and 2013 60](#_Toc17551709)

[Box plot showing southwest border apprehensions in 2011,2012 and 2013 61](#_Toc17551710)

[Country average life expectancies stratified into continents with 0 inclusion (on the left) and without 0 inclusion (on the right) 61](#_Toc17551711)

[US economy comparison with 4 competing nations with distorted area 62](#_Toc17551712)

[US economy comparison with 4 competing nations 63](#_Toc17551713)

[Murder rates across states with default ordering (on the left) and ordering by the rate (on the right) 63](#_Toc17551714)

[Average of male and females with standard error 64](#_Toc17551715)

[scatter point plot of male and female heights 64](#_Toc17551716)

[scatter point plot of male and female heights with jitter and alpha 65](#_Toc17551717)

[Histogram of male and female heights distribution with different X-axis 66](#_Toc17551718)

[Histogram of male and female heights distribution with same X-axis 66](#_Toc17551719)

[Bar plot of average population sizes for each continent in 2015 67](#_Toc17551720)

[Boxplot log scaled of average population sizes for each continent in 2015 67](#_Toc17551721)

[Comparison of income between 1970 and 2010 across continents 68](#_Toc17551722)

[Comparison of income between 1970 and 2010 across continents with reordering of labels 68](#_Toc17551723)

[Slope Chart of life expectancy between 2010 and 2015 69](#_Toc17551724)

[Bland-Altman plot of difference between 2010 and 2015 70](#_Toc17551725)

[Scatter plot of income per day vs infant mortality rate , with OPEC membership, region and population size 71](#_Toc17551726)

[List of all shapes for the shape argument in R 71](#_Toc17551727)

[Case of measles in California over the years 73](#_Toc17551728)

[Case of measles in the US over the years 74](#_Toc17551729)

[Bad example of a 3D graphic 74](#_Toc17551730)

[Per 10000 disease rates for California 75](#_Toc17551731)

1. Introduction to Data Visualization
   * 1. Introduction to Distributions

Numerical data is often summarized with an average value, For example the quality of a high school is sometimes summarized with one number the average score in a standardized test, occasionally a second number is reported, the standard deviation. So you might read a report stating that scores at this high school were 680 ± 50, where 50 is the standard deviation.

Note that we are summarizing an entire vector of scores with just 2 numbers, in much case those 2 numbers are enough to understand the data, data visualization will help us determines when they are appropriate. The first step in data visualization is learning to summarize list of factors or numeric vectors. The most basic statistical summary of a list of objects or numbers is its **distribution.**

Once a vector has been summarized as a distribution, there are several data visualization techniques to effectively relay this information.

* + 1. Data Types

To determine how to visualize data, we need to know what type of data it is. There is two big category of variables, categorical and numericals, each can be divided into two further groups:

* Categorical ordinals
* Categorical non-ordinals
* Numerical discrete
* Numerical continuous

Variables that defined by a small number of groups are called categorical data, for example the variable sex, which has 2 values male or female or the variable regions of the states (only 4 values). Some categorical can ordered, for example spiciness of a dish can be mild medium or hot, even if they are not numbers per se they can still be ordered we call those variables **ordinal** data.

Continuous variables are numerical variables that can take any value, such as heights if measure with enough precision, for example a pair of twins could be 68.12 inches and 68.11 inches respectively. Count such as population sizes are discrete because they must be round numbers.

When we have many groups with few cases in each group, we refer to this as **discrete numerical** variables, for example the number of packs of cigarette a person smokes a day rounded to the closest pack; so 0, 1, 2… would be considered ordinal while the number of cigarettes smoked; 0, 1, 2, 3..up 60 would be a numerical variable.

* 1. Introduction to Distributions
     1. Describe Heights to ET

Let’s consider, an artificial problem, that you want to describe the heights of your classmates to E.T. an extraterrestrial that has never seen humans. At first we need to collect data. To do this we asked student to report their heights in inches. We asked them to provide sex information because we know there are two different groups of heights, males and females. We collect the data and save it in a data frame.The result data frame is the heights data frame here are the first six entries:

*data("heights")*

*head(heights)*

*sex height*

*1 Male 75*

*2 Male 70*

*3 Male 68*

*4 Male 74*

*5 Male 61*

*6 Female 65*

One way to convey the heights to ET is to simply send him the list of 924 heights. Bu there is a more effective ways to convey this information, and understanding the concepts of distributions will help. To simplify the explanation we focus on male heights firs. The most basic statistical summary of a list of objects or numbers is its **distribution**. Distribution is a compact description of a list with many elements, for example with categorical data, the distribution simply describes the proportions of each unique category. For example the sex represented in the heights data set can be summarized by the proportions of each of the categories, female and male this two category **frequency table** is the simplest form of a distribution we can form:

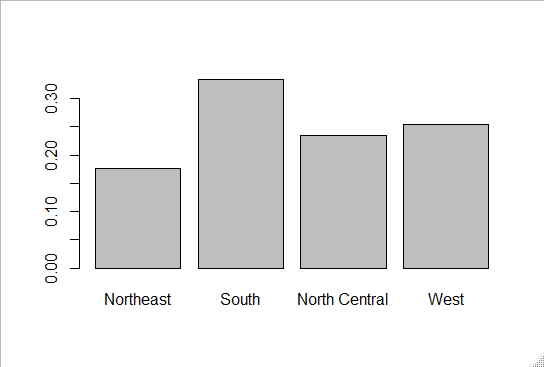
*prop.table(table(heights$sex))*

*Female Male*

*0.2266667 0.7733333*

For this case we don’t need data visualization since one number describes everything we need to know about this data set; 23% of the data set are female the rest are Male. When there is a more categories a simple bar plot describes the distribution for example:

Boxplot of propotion of murder by region

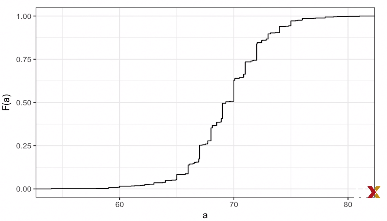


Each height of each bar gives us the proportions for each regions. Although this plot, a graphical representation of a frequency table, does not provide much more insights than the table itself, it is a first example of how we convert a vector into a visualization that succinctly summarizes all the information in that vector.

For numerical variables the process is more complex. In general, when data is not categorical, reporting the frequency of each unique entry is not an effective summary since most of the entries are unique. For example, while several students reported a height of 68 inches only one student reported a height of 68.503937007874 inches, and only one reported a height of 68.8976377952756 inches. We assume they converted from 174 cm and 175cm to inches respectively.

The best way to define a distribution for numerical data is to define a function that reports the proportions of the data below a value A for all possible values of A. This function is called a **cumulative distribution function (CDF)**, the following mathematical notation is used in statistics textbook:

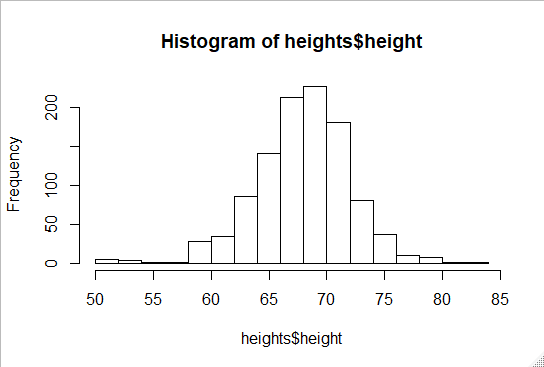
CDF of heights in the class



Because CDFs can be determined mathematically, as opposed to using data as we do here, the word empirical is added to distinguish, and we use the term ECDF. Although ECDF provides all the information we need and it is widely discussed in statistics textbooks, the plot is not very popular in practice. The main reason is that it does not easily convey characteristics of interest, such as, at what value is the distribution centered? Is the distribution symmetric? What range contains 95% of the data? We can decipher it from the plot but it’s not easy. **Histogram**  are preferred because they can answer those questions. They sacrifice a bit of information to produce plots that are much easier to interpret.

The simplest way to make an histogram is to divide a span of our data into non-overlapping bins of the same size, then for each bins we count the number of values that fall into the interval. The histogram plots these counts as bars with the base of the bar the interval. Here’s an histogram of heights:

Histogram of Heights

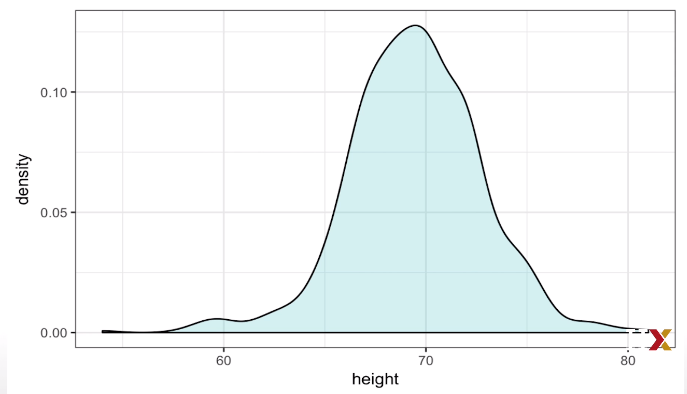


If we send those data, our friend E.T. will learn that the heights span from 55 to 81 and that the majority, with more than 95%, are between 63 and 75 inches. Secondly the heights are symmetric around 69 inches. Lastly by adding all counts E.T. will be able to have an approximation of the size of the sample. It gives almost all the information of the CDF, we provide almost all the information contained in the raw data that is 708 heights, with just 23 bin counts. The only problem is that all values are treated as the same when computing the bin heights. So for example the histogram does not distinguish between 64 64.1 and 64.2, because they are in the same interval so in the same bin.

* + 1. Smooth Density Plots

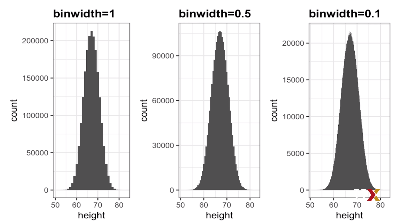
Smooth density plots are similar to histograms, but are aesthetically more appealing. Here`s what are smooth density plot looks like:

Smooth density plot of Male heights



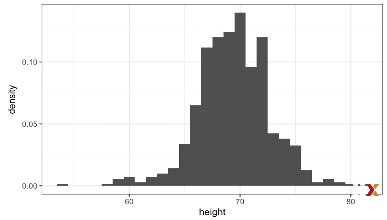
Note that we no longer have sharp edges at the interval boundaries and that many of the local peaks have been removed. Also notice that the scale of the y-axis changed from counts to something called **density**. We assume that our list of observed values comes from a much larger list of unobserved values. So in the case of male heights, you can imagine our lists of students heights comes from hypothetical list containing all the heights of all male students in all the world measured very precisely. So as an example, let’s have a millions of these, this list like any other list has a distribution and this is really what we want to report, since it’s more general. Unfortunate, we don`t get to see it. However, we can assume that help us perhaps approximate. Because we’re assuming that we have a million of very precise measures we can make an histogram with very small bins. If we do this, consecutive bins will be similar. This explain the ``smoothness``. The smooth density is basically the curve that goes through the top of the histogram bars when the bins are very small. Here’s what happen when you reduce the size of the bins:

histograms of heights with different size bins



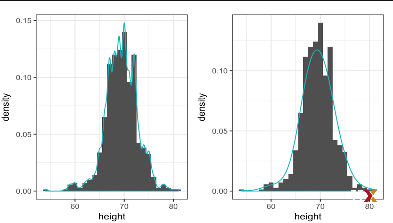
To make the curve not depend on the hypothetical size of the hypothetical list we compute the curve on the frequency scale rather than the count scale. In our example we don`t have millions records we have 708, we can`t make an histogram with very small bins. So we first have to compute the frequencies then plot an histogram with appropriate bins size. So the histogram looks like this :

Histogram frequencies of 708 students’ heights



If we keep the top of the bar as points and draw a curve that goes through those points we get the smooth density curve. We can control the smoothness of the curve with the gg plot option here are two examples that show different degrees of smoothness:

Histograms of heights with different smoothness degree

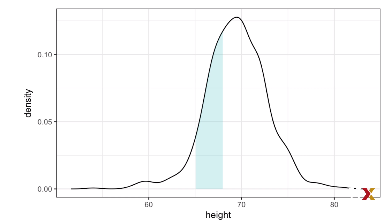


We need to make the choice of smoothness with care, as the resulting visualization can change the interpretation of the data. We should select a degree of smoothness that we can defend as being representative of the underlying data.

In this case, we do have a reason to believe that there are a proportion of people with similar heights should be about the same. For example the proportion that is 72 inches should be more similar to the proportion of the 71 inches than the proportion that is 78 or 65.

Finally, to interpret them we need to understand the unit of the Y-axis. The Y-axis is scaled so that the area under the density curve adds up to 1. So the best way to determine the proportion of data in that interval is by computing the proportion of the total area contained in that interval. Here`s an example:

Proportion of values between 65 and 68



The proportion of the area in blue is about 0.31, meaning that about 31% of our values are between 65 and 68.

* + 1. Normal Distribution

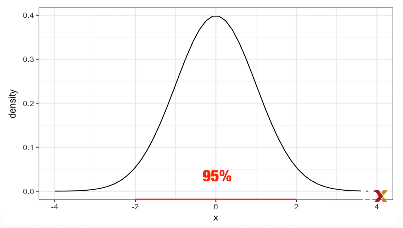
Although Histogram and density plots provide excellent summaries of a distribution, we can go further. We often see the average and standard deviation used as summary statistic for a list of numbers, a two number summary. To understand those summary we need to talk about the **normal distribution**. Also know as the Bell curve or the Gaussian distribution, is one of the most famous mathematical concepts in history. A reason for this is that approximately normal distributions occur in many situations. Example include:

* Gambling winnings
* Heights
* Weights
* Blood pressure
* Standardized test scors
* Experimental errors

Rather than using data the normal distribution is defined with a mathematical formula:

Note that the formula is composed of only 2 parameters m and s.They correspond respectively for the mean and the standard deviation. The distribution is symmetric and centered around the mean and 95% of the values are within 2 standard deviation from the mean.

Normal distribution with m=0 and s=1



So if a data set is approximated with the normal distributions, all the information coded to described the dataset are encoded in just 2 numbers the mean and the standard deviation of the data set.

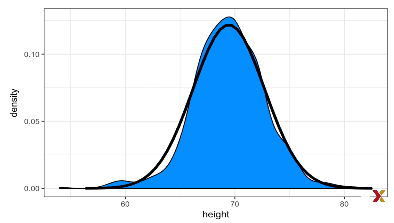
You compute those in R by:

*mean<-sum(heights$height)/length(heights$height)*

*standardDeviation<-sqrt(sum(heights$height-mean)^2)/length((heights$height))*

So, here is the plot of the smooth density of our male heights:

Smooth density of Male heights overlaid with the normal distribution



The **standard unit** of value tell us how many standard deviations away from the average this value is. Specifically for a value x, we define the standard unit as:

With standard units we can specify a value, for example a value with a z at 0 would mean a heights in the average, a person above average will have a z>0 and below average a z<0. In R we can easily obtain standard units using the function **scale()** like this:

*z<-scale(x)*

now that we have converted to standard units all we have to do is count the number of z’s that are less than 2 and bigger than negative 2 and then divide by the total i.e. we take the mean:

*mean(abs(z)<2)*

*[1] 0.9495074*

Which approximate to 95% so it is safe to assume that the data is approximately normal for those intervals.

* 1. Quantiles, Percentiles, and Boxplots
     1. Quantile-Quantile Plots

We’ve described how, if a distribution is well approximated by the normal distribution, we can have a very useful and short summary. But to check if it is we can use quantile-quantile plots or **q-q plots**. We start by defining a series of proportion for example, p = 0.05, 0.10, 0.15, up to 0.95. Then we determine the value q, so that the proportion of the values in the data below q is p. The q’s are referred to as **quantiles.** For examples:

*mean(x<=69.5)*

*[1] 0.5147783*

That means that if p=0.5, then the q associated with that p is 69.5. We can make that computation for a series of p’s. If the quantiles for the data match the quantiles for the normal distribution, then it must be because the data is approximated by a normal distribution. To have the quantiles for the data we use the **quantile()** function in R like this:

*Observed\_quantile<-quantile(x,p)*

To obtain the theoretical normal distribution quantiles with the corresponding average and standard deviation we do:

*theoretical\_quantile<-qnorm(p,mean-mean(x),SD-sd(x))*

To see if they match, we can plot them against each other, and then draw the identity line, to see if the points fall on the line, like this:

*plot(theoretical\_quantile,observed\_quantile)*

*abline(0,1)*

* + 1. Percentiles

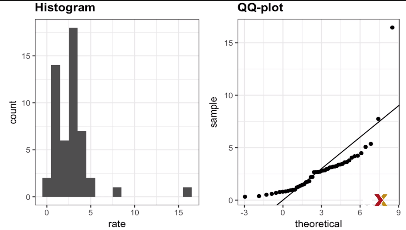
Percentiles are special quantiles that are commonly used, they are defined when p=0.01 0.02 0.03… up to 0.99 1%,2%,3%... etc . We call the case of p equals 0.25 the 25th percentiles, it give us the number for which 25% other data is below. The most famous percentile is the 50th, also know as the **median**, the median and the average are the same in a case of a normal distribution.

Another special case that receives a name are the quartiles, which are obtained when p=0.25, 0.5, 0.75

* + 1. Boxplots

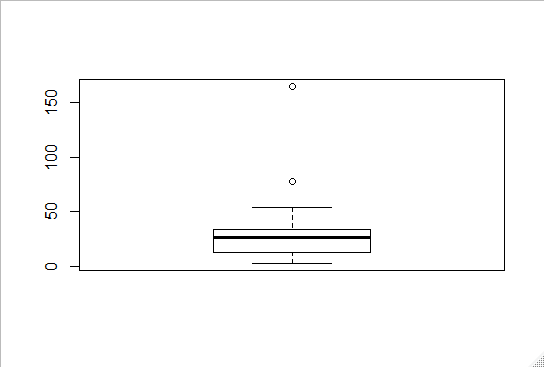
Let’s go back to our murders data in the US. Suppose we want to summarize the murder rate distribution. Using the data visualization techniques we have learned we notice that the normal approximation does not apply here, we can see it in the histogram and in the QQ plot.

Histogram and QQ plot of murder rate in the US



Now imagine you were trying to describe this data to someone who is used to receiving just two numbers, the average and the standard deviation. In this case, John Tukey offers some advice. Provide a five number summary compose of the range along the quartiles, 25th, 50th and 75th. Furthermore he suggest we ignore outliers when computing the range and instead plot these as independent points. Finally he suggest we represent this 5 numbers as a box with whiskers like this:

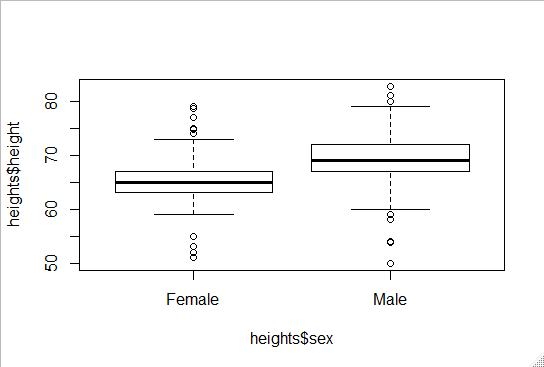
boxplot of the murder rate in the US



We can see that the majority of the data are between 0 and 50 with the exception of the 2 outliers. The thick black line represent the median.

Boxplot are even more useful when we want to quickly compare two or more distributions, here’s an example :

Boxplot of heights of male and female students

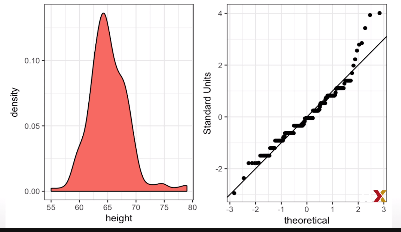


We can see from that that men are on average taller than women, we also see that the standard deviation appears to be similar.

* 1. Exploratory Data Analysis
     1. Distribution of Female Heights

So far, we only worked with male heights, but to report on the summary we need to include the summary of female heights also. We expect that will follow a normal distribution, just like men. However explanatory plots reveal that the approximation is not as useful if we look at the density plot or the qq plot:

density plot and qq plot of female heights



The density plot has a second bump around 67 also the qq plot shows highest points tend to be taller than expected for the normal distribution.

So when we have to report on female heights we might need to provide a histogram rather than the average and standard deviation. If we look at other female height distributions from other sources, we do find they are well approximated by the normal distributions so why the 2nd bump in our data? Why so many outliers? Why so many taller than expected woman? One explanation is that in the form students used to enter their heights, the internet form, female was the default sex, and some male may have forgot to changed the sex value on the form.

1. Introduction to ggplot2
   1. Basics of ggplot2
      1. ggplot

To create all the visualization technique that we learned previously, we are going to use the **ggplot2** package. We can load it and other useful packages by installing and loading the **tidyverse** library with:

*install.packages(“tidyverse”)*

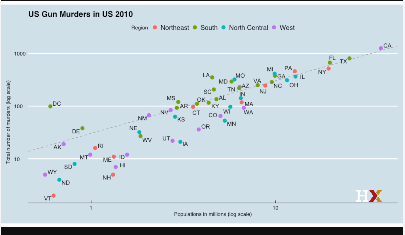
*library(tidyverse)*

There is different package to help us do graphical representation of data such as grid or lattice. Ggplot has the advantage to break plots into components in a way to create relatively complex and aesthetically pleasing plots using intuitive syntax. It uses a grammar of graphics (the “gg” in ggplot), this is similar to the way learning grammar can help you construct hundreds of different sentences by learning just a handful of verbs, nouns and adjectives without having memorize each specific sentence. One limitation of ggplot is that it works with data tables.

* + 1. Graph Components

We are going to construct a graph that summarizes the US murders data set. We are going to construct a graph like this :

Scatterplot of Us gun murders in 2010



The first step to learn ggplot2 is to be able to break graph into components. So let’s break down this plot while we introduce some of the ggplot terminology.

1. The US murders data table is being summarized (data component)
2. The plot is a scatter plot (geometry component)
3. The mappings, the x-axis value are used to display population size, the y-axis value are used to display the total number of murders, Text is used to identify the states and colors are used to denote the four different regions (aesthetic mapping)

Other components of the plot worth mentioning are:

1. The range of the x-axis and the y-axis appear to be defined by the range of the data
2. They are both on log scales (scale component)
3. There is labels, a title, a legend
4. The style looks like the style used by The Economist magazine
   * 1. Creating a New Plot

The first step in creating a ggplot graph is to define a ggplot object. We do this with:

g*gplot(data=murders)*

this will associates the data set with the plotting object we can also pipe the data to ggplot:

*murders %>% ggplot()*

Because we didn’t assign the result to a variable it was evaluated and a blank space was created (because no geometry was defined) we can define an object to store the ggplot object creation like this:

*p<-ggplot(data=murders)*

we can render the plot by simply printing p

* 1. Customizing Plots
     1. Layers

In ggplot graph are created by adding layers. We add them component by component. Layers can define geometries, compute summary statistics, define scale and even change style. To add layers we use the +. Usually the first added layer defines the geometry. We want to make a scatter plot we use the function **geom\_point**, geometry functions follow usually the patter **geom\_**name\_of\_the\_geometry\_to\_add. For geom\_point to work we need to provide data and mapping to the function, we’ve already connected an object with the murders data table, and if we add, as a layer, geom point, we will default using this data.

For the mapping there is 2 required arguments x and y (the x and y of the plot). aes is a function that connects data with what we see on the graph. We refer to this connection as the aesthetic mappings. The outcome of this function is often used as the argument of a geometry function for example:

*p+geom\_point(aes(x=population/10^6,y=total))*

Wil create a scatter plot with population in millions on x\*axis and total murder rate on y-axis. AES recognizes the variable names from the data object, this behavior is specific to aes.

A second layer we want to add involves adding a label to each point, to identify which point goes with which state. **The geom\_label and geom\_points**  permit us to add text to the plot.

Geom\_label adds a label with a little rectangle and geom text simply add the text. Because each state, each point, has a label we need an aesthetic mapping to make this connection. We supply the mapping between point and label through the label argument of aes, like this:

*p+geom\_point(aes(x=population/10^6,y=total))+geom\_text(aes(population/10^6,total,label=abb))*

* + 1. Tinkering

By exploring the documentation we note that the geometry function has many arguments other than aes and data. They tend to be specific to the function. For example in the plot we wish to make, the points are larger than the default ones. We can change it using this code:

*p+geom\_point(aes(x=population/10^6,y=total),****size=3****)+geom\_text(aes(population/10^6,total,label=abb))*

Note that size is not a mapping, it affects all the points the same, it was outside aes. Unfornunately, now we can’t read the labels anymore. By exploring the documentation, we discover that there’s an argument nudge\_x which lets us move the label just a little bit. We do this with:

p*+geom\_point(aes(x=population/10^6,y=total),size=3)+geom\_text(aes(population/10^6,total,label=abb),****nudge\_x = 1****)*

The previous code work well but can be rendered more efficient by adding a global aesthetic mapping. We can achieve this by declaring a mapping aes inside the call of ggplot liket this:

*p<-murders%>%ggplot(aes(population/10^6,total,label=abb))*

By doing this we don’t need to add the aes function for each geom function it transform the previous code to this:

*p+geom\_point(size=3)+geom\_text(nudge\_x = 1.5)*

Now if we need to override the global mapping, we can do this thanks to the local mappings that override the global mapping.

* + 1. Scales, Labels, and Colors

In our example we desire our scale to be in a log scale, which is not the default, so the change needs to be added through a scales layer. We can use the function scale\_x\_continuous to change the scale on the plot here is how you do it:

*p+geom\_point(size=3)+geom\_text(nudge\_x = 1.5)+scale\_x\_continuous(trans="log10")+scale\_y\_continuous(trans= “log10")*

Note that because we are in log scale now the nudge needs to be smaller. This transformation is so common that ggplot provides specialized functions. We can use the function **scale\_x\_log10** to achieve the same.

Now that the plot is at the correct scale, we are ready to add some labels and titles. By exploring the documentation we discovert that the function **xlab()** and **ylab()** add a label on the axis and **ggtitle** adds a title to the plot, here are to use them:

*p*

*+geom\_point(size=3)*

*+geom\_text(nudge\_x = log10(1.5))*

*+scale\_x\_log10()+scale\_y\_log10()*

*+xlab("Populations in millions (log scale)")*

*+ylab("Total number of murders (log scale)")*

*+ggtitle("US Gun murders in 2010")*

To get to the goal plot we are still missing color, legend and some optional changes to the style. We can change the color of the points using the col argument in the geom\_point function, to make the point change color depending on the region we need to assign ,inside the aes function of geom\_point , the region variable to the col argument like this:

*p+geom\_point(aes(col=region),size=3)+geom\_text(nudge\_x = log10(1.5))+scale\_x\_log10()+scale\_y\_log10()+xlab("Populations in millions (log scale)")+ylab("Total number of murders (log scale)")+ggtitle("US Gun murders in 2010")*

We want to add a line that represents the average murder rate for the entire country. Note that once we determine the per million rate to be r for the entire country, this line is defined by the formula y=rx because if a state has a population x and it has the same murder rate as the US, which is r it has r\*x total number of murders. To compute the average for the country we need to apply some dplyr skills we’ve learned:

r*<-murders%>%summarize(rate=sum(total)/sum(population)\*10^6)%>%.$rate*

To add the line we use the **geom\_abline** function (ab stands for intercept a and the slope b), the default line for geom\_abline has slope 1 and intercept 0. So we only define the intercept to add the line here is the code:

*p+geom\_point(aes(col=region),size=3)+geom\_text(nudge\_x = log10(1.5))+scale\_x\_log10()+scale\_y\_log10()+xlab("Populations in millions (log scale)")+ylab("Total number of murders (log scale)")+ggtitle("US Gun murders in 2010")+geom\_abline(intercept = log10(r))*

Finally, to reach the original plot, we need to change the line type from solid to dashed, change the color from black to grey and also, we need to draw the line before the points otherwise the lines goes over the point. To do this we are going to redefine p using some arguments in abline:

*p<-p+geom\_abline(intercept=log10(r),lty=2,color="darkgrey")+geom\_point(aes(col=region),size=3)*

One last thing we need to do to match our fist plot is to change the word region to be capitalize. To do this we use the function scale\_color\_discrete like this:

*p<-p+scale\_color\_discrete(name="Region")*

* + 1. Add-on Packages

To make the final touch on our plot we need the package ggthemes and ggrepel packages. Ggthemes, as the name suggest, is a package allowing you to change the theme of the plot. You do it by :

*library(ggthemes)*

*p+theme\_economist()*

The final difference between our plot and our goal plot has to do with the positions of the labels, some of them fall on the top of each other, making hard to read. The package ggrepel includes a geometry that adds labels ensuring they don’t fall on the top of each other to do this we do:

*p+geom\_text\_repel()*

* + 1. Other Examples

Now that we know how to use ggplot, we can try to make some of the summary plots we have previously described, let’s start with the histogram by doing the histogram for the male heights. At first we need to filter the height for the males:

*heights\_m<-heights%>%filter(sex=="Male")*

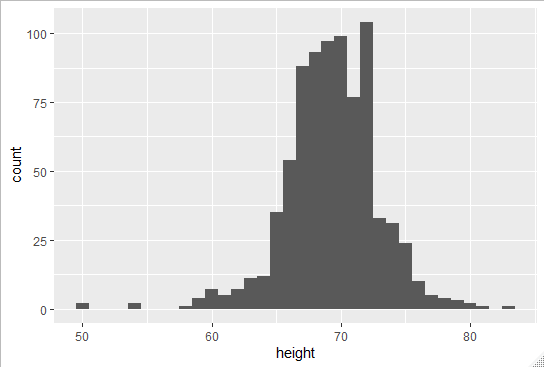
The next step is deciding on the geometry, geom\_histogram is the correct one here, by looking at the documentation we learned that geom\_histogram need only one argument x the data we want to build the histogram for. So the code looks like this:

*p<-heights%>%filter(sex=="Male")%>%ggplot(aes(x=height))*

*p+geom\_histogram(binwidth=1 )*

and here is the result:

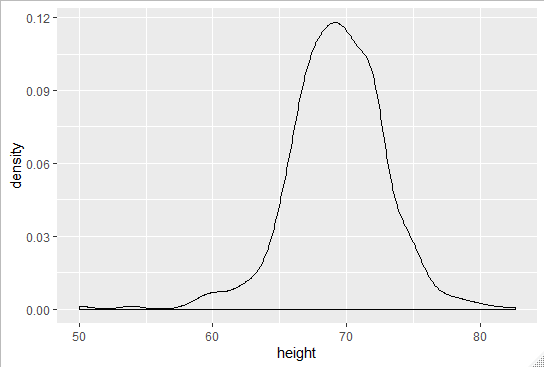
Histogram of Male heights with ggplot



We can create smooth densities using another geometry. The geometry in this case is called geom\_density. We’ve already defined an object p, so we can, instead iof adding a histogram layer, we add a geom\_density layer. Like this:

*p+geom\_density()*

Density functions of male heights

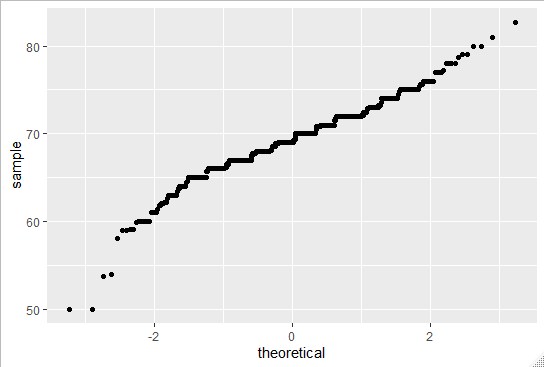


For qqplot we use the geom\_qq geometry, we need to supply a sample argument. So we need to redefine p to have a sample instead of x and add the geometry layer like this:

p*<-heights%>%filter(sex==”Male”)%>%ggplot(aes(sample=height))*

p+geom\_qq()

QQ plot of Male heights



By default the qq plot is compared to the normal distribution with the average=0 and standard deviation=1. To change this we need to use the dparams argument. We need to define an object params that will have the mean and standard deviation of our data. We do this with:

*params<-heights%>%filter(sex=="Male")%>%summarize(mean=mean(height),sd=sd(height))*

and then add the geometry by assigning this new object that we created to the dparams argument.

*p+geom\_qq(dparams=params)*

Finally, let’s learn how to make grid of plots. To put plots next to each other we can use the grid.arrange from the gridExtra package. To do this we first define plots that we assign to objects:

*p<-heights%>%filter(sex=="Male")%>%ggplot(aes(x=height))*

*p1<-p+geom\_histogram(binwidth = 1,fill="blue",col="black")*

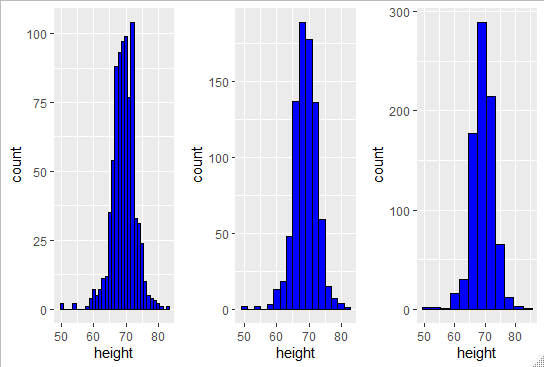
*p2<-p+geom\_histogram(binwidth = 2,fill="blue",col="black")*

*p3<-p+geom\_histogram(binwidth = 3,fill="blue",col="black")*

*grid.arrange(p1,p2,p3,ncol=3)*

We get this:

3 histogram on the same graphic



1. Section 3: Summarizing with dplyr
   1. Summarizing with dplyr
      1. Dplyr

One important aspect of exploratory data analysis is summarizing data. We’ve already discussed about the average and the standard deviation that provides all the necessary information needed to summarize normally distributed data. We’ve also seen that better summaries can be achieved by splitting data into groups before using normal approximation. For example, in our heights dataset, we describe the height of men and women separately. In this section we going to see two new dplyr verbs to make computations easier summarize() and groub\_by(). We will also learn to access resulting values using the dot placeholder. And Finally we will learn to use arrange() that help us examine data after sorting them.

Once the tidyverse is loaded we’re ready to use **summarize().** The function provides a way to compute summary statistics with intuitive and readable code. Let’s start with an example, we want to compute the average and the standard deviation for males to do this we write:

*s<-heights%>%filter(sex=="Male")%>%summarize(average=mean(height),standard\_deviation=sd(height))*

*s*

*average standard\_deviation*

*1 69.31475 3.611024*

Because the result of summarize is a dataframe we can access its components with the $ so if we type :

*s$average*

*[1] 69.31475*

Note that we can compute any summary that operates on vector and returns a single value, for example we can write code that computes the median, the mean and the max like this:

*heights%>%filter(sex=="Male")%>%summarize(median=median(height),minimum=min(height),maximum=max(height))*

*median minimum maximum*

*1 69 50 82.67717*

Note that the function summarize can only work with functions that return a single value.

* + 1. The Dot Placeholder

Here we are going to learn how to make dplyr functions return vectors instead of data frames. Let’s take an example with the US murders data. Our data table includes total murders and population size for each state and we already added a murder rate column to the data.

However, note that the US murder rate is not the average of the state murder rates. If you do that you get 2.78 which not the US murder rate, which is closer to 3. This because in the computation the small states count as the same as the large states. When we compute the average US murder rate it needs to take into account bigger states more than smaller states. So the correct computation of US murder rate is as follows:

*us\_murder\_rate<-murders%>%summarize(rate=sum(total)/sum(population)\*100000)*

*us\_murder\_rate*

*rate*

1. *3.034555*

Now the computation we just made represents just one number. So suppose we want to use a function that requires just a numeric value, we can’t use the US murder rate object because it’s a data frame. We can access the elements of the data frame with the dot here’s an example:

*us\_murder\_rate%>% .$rate*

*[1] 3.034555*

We get the a numeric value. It is the value in the rate column of the us\_murder\_rate variable, to understand this code you need to think of the dot as a placeholder for the data that is being passed through the pipe. Because it’s a data frame we can use the $ and the name of the column to access its value. To get a number from the original data we can type this:

*us\_murder\_rate<-murders%>%summarize(rate=sum(total)/sum(population)\*100000)%>%.$rate*

*> class(us\_murder\_rate)*

*[1] "numeric"*

*us\_murder\_rate*

*[1] 3.034555*

Note that now us\_murder\_rate is a numeric value not a data frame anymore.

* + 1. Group By

One of the common operations in data exploration is to first split data into groups and then compute summaries for each group. For example, we may want to compute the average and standard deviation for men and women heights separately. To achieve this we use the **group\_by()** function here is an example:

heights%>%group\_by(sex)

# A tibble: 1,050 x 2

# Groups: sex [2]

sex height

<fct> <dbl>

1 Male 75

2 Male 70

3 Male 68

4 Male 74

5 Male 61

6 Female 65

7 Female 66

8 Female 62

9 Female 66

10 Male 67

# ... with 1,040 more rows

Although not obvious from its appearance this now a special data frame called a “group data frame” and dplyr functions, in particular summarize, will behave differently when acting on this object. You can think of this table as many tables with the same columns but not necessarily the same rows that are stacked together as one object. So when you apply a summary function to get mean and standard deviation for example:

heights%>%group\_by(sex)%>%summarize(average=mean(height),standard\_deviation=sd(height))

# A tibble: 2 x 3

sex average standard\_deviation

<fct> <dbl> <dbl>

1 Female 64.9 3.76

2 Male 69.3 3.61

It returns a table with the average and the standard deviation for each group.

For another example we can write a code that compute the median murder\_rate in the four regions of the country:

murders%>%group\_by(region)%>%summarize(median\_murder\_rate=median(total/population\*100000))

# A tibble: 4 x 2

region median\_murder\_rate

<fct> <dbl>

1 Northeast 1.80

2 South 3.40

3 North Central 1.97

4 West 1.29

* + 1. Sorting Data Tables

We already talk about the order and sort functions, but for ordering entire tables, the dplyr function **arrange()** is very useful. For example, if we want to order the states by their population size, all we have to do is:

*murders%>%arrange(population)%>%head()*

*state abb region*

*1 Wyoming WY West*

*2 District of Columbia DC South*

*3 Vermont VT Northeast*

*4 North Dakota ND North Central*

*5 Alaska AK West*

*6 South Dakota SD North Central*

*population total murder\_rate*

*1 563626 5 0.8871131*

*2 601723 99 16.4527532*

*3 625741 2 0.3196211*

*4 672591 4 0.5947151*

*5 710231 19 2.6751860*

*6 814180 8 0.9825837*

Note that we decide which column to sort by, to see the states ordered by murder rate instead of population:

*state abb region*

*1 Vermont VT Northeast*

*2 New Hampshire NH Northeast*

*3 Hawaii HI West*

*4 North Dakota ND North Central*

*5 Iowa IA North Central*

*6 Idaho ID West*

*Population total murder\_rate*

*1 625741 2 0.3196211*

*2 1316470 5 0.3798036*

*3 1360301 7 0.5145920*

*4 672591 4 0.5947151*

*5 3046355 21 0.6893484*

*6 1567582 12 0.7655102*

Note that the default behavior of arrange is to sort in ascending order, the function desc() transforms a vector to be in descending order.

We can also do nested sorting. Let’s say we’re ordering by a column and there’s a ties. We can break the ties with a the second column, or a third of a fourth. In this example we order by region, then within each region we order by murder rate:

*murders%>%arrange(region,murder\_rate)%>%head()*

*state abb region population*

*1 Vermont VT Northeast 625741*

*2 New Hampshire NH Northeast 1316470*

*3 Maine ME Northeast 1328361*

*4 Rhode Island RI Northeast 1052567*

*5 Massachusetts MA Northeast 6547629*

*6 New York NY Northeast 19378102*

*total murder\_rate*

*1 2 0.3196211*

*2 5 0.3798036*

*3 11 0.8280881*

*4 16 1.5200933*

*5 118 1.8021791*

*6 517 2.6679599*

Another useful function is the top\_n function, above we’ve just seen the function head to avoid having the page fill with the entire data table. It show us the first 6 entries, if we want to see the top 10 we can do it like this:

*murders%>%top\_n(10,murder\_rate)*

We can combine it with arrange to automatically have the top 10 murder rate like this:

*murders%>%arrange(desc(murder\_rate))%>%top\_n(10)*

1. Section 4: Gapminder
   1. Introduction to Gapminder
      1. Case Study: Trends in World Health and Economics

In this section, we are going to apply at that we learned so far on the **Gapminder** data. Hans Rosling, cofounder of the Gapminder Foundation, created the foundation with the idea of educating the people using data to dispel common myths about the so-called developing world. The organization uses data to show how actual trends in health and economics contradict the narratives of emanate sensationalist media. We are going to answer 2 questions with this data

**Is it fair to say the world is divided into rich and poor ?**

**Has income inequality has worsened during the last 40 years ?**

* + 1. Gapminder Dataset

To load the gapminder data sets we do:

*data("gapminder")*

country year infant\_mortality

1 Albania 1960 115.40

2 Algeria 1960 148.20

3 Angola 1960 208.00

4 Antigua and Barbuda 1960 NA

5 Argentina 1960 59.87

6 Armenia 1960 NA

life\_expectancy fertility population

1 62.87 6.19 1636054

2 47.50 7.65 11124892

3 35.98 7.32 5270844

4 62.97 4.43 54681

5 65.39 3.11 20619075

6 66.86 4.55 1867396

Gdp continent region

1 NA Europe Southern Europe

2 13828152297 Africa Northern Africa

3 NA Africa Middle Africa

4 NA Americas Caribbean

5 108322326649 Americas South America

If we compare the infant mortality rate of Sri Lanka and Turkey:

# compare infant mortality in Sri Lanka and Turkey

*gapminder %>% filter(year == 2015 & country %in% c("Sri Lanka", "Turkey")) %>%select(country, infant\_mortality)*

*country infant\_mortality*

*1 Sri Lanka 8.4*

*2 Turkey 11.6*

* + 1. Life Expectancy and Fertility Rates

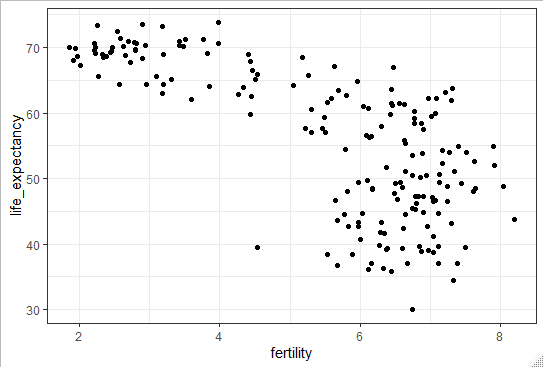
Our misconception comes from the preconceived notion that the world is divided into two groups, the Western world, composed of Western Europe and North America, which is characterized by long lifespans and small families versus the developing world , Africa, Asia and Latin America, characterized by short lifespans and large families. But does the data support this idea ?

The first plot we need to do to answer that question is the scatter plot of life expectancy versus fertility rate. Fertility rates are defined as the average number of children per woman. We will start looking at data from 50 year ago. We do it like this:

*ds\_theme\_set()*

*> filter(gapminder,year==1962)%>%ggplot(aes(fertility,life\_expectancy))+geom\_point()*

Scatter plot of fertility against life expectancy in 1962

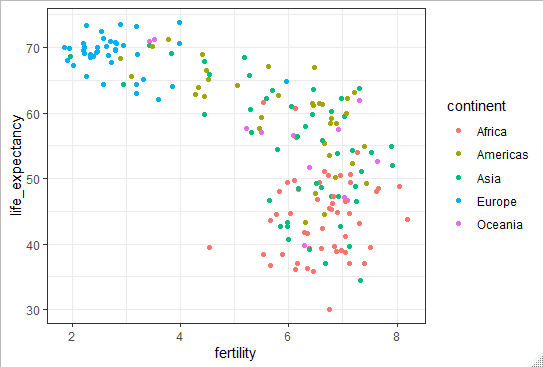


Note that most points do in fact, fall into two distinct categories, one with life expectancies around 70 years and 3 or less children per family and the other with life expectancies lower than 65 years and with more than 5 children per family.

To confirm that these countries are from the regions we expect, we can use color to represent document, so we change the code like this:

*filter(gapminder,year==1962)%>%ggplot(aes(fertility,life\_expectancy,color=continent))+geom\_point()*

Fertility against life expectancy in 1962 with continent differenciation



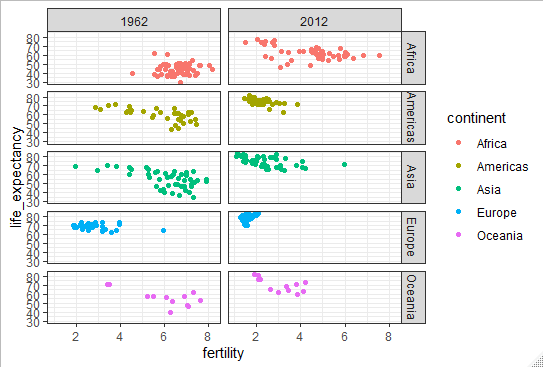
So in 1962, the West versus the developed country reality was true. Does it still hold ground 50 years later ?

* 1. Using the Gapminder Dataset
     1. Faceting

We can do the same plot with the data from 2012 but for comparison side by side plots are preferable. We can achieve this by **faceting variables**. We stratify the data by some variable and make the same plot for each strata. Here is a example of faceting by year:

filter(gapminder,year==1962)%>%ggplot(aes(fertility,life\_expectancy,color=continent))+geom\_point()+facet\_grid(contin ent~year)

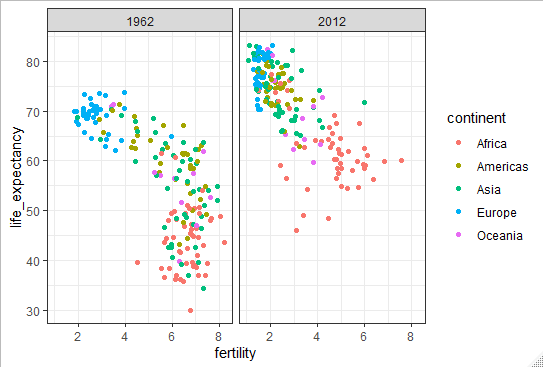
Scatter plot, Facetted by year and continent, of fertility versus life expectancy



We can replace one of the argument in facet\_grid by a . to get the facet just by one variable, so for example if we wanted to facet just by year :

*filter(gapminder,year%in%c(1962,2012))%>%ggplot(aes(fertility,life\_expectancy,color=continent))+geom\_point()+facet \_grid(.~year)*

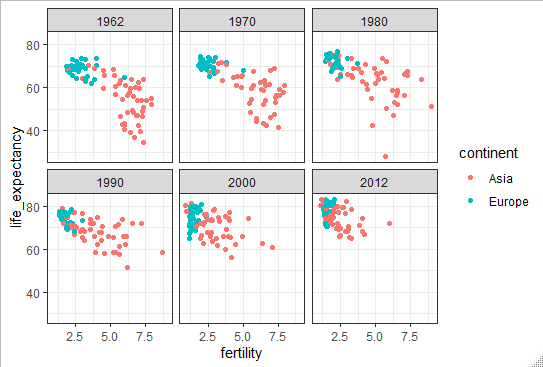
Scatter plot, Facetted by year, of fertility versus life expectancy



We can clearly see that the cluster with large family low life expectancy moved up to join the cluster of low family high life expectancy. To explore how this transformation happened through the years, we can make the plot for several years, for example 1970, 1980, 1990 and 2000. If we do this, we will not want all the plots on the same row. This is the default behavior of facet\_grid. If we do this the plot will be too thin. Instead we want to space the plots around different rows and columns. For this we use the **facet\_warp()** function like this:

*gapminder%>%filter(year %in% years & continent %in% contients)%>%ggplot(aes(fertility, life\_expectancy,col=continent))+geom\_point()+facet\_wrap(.~year)*

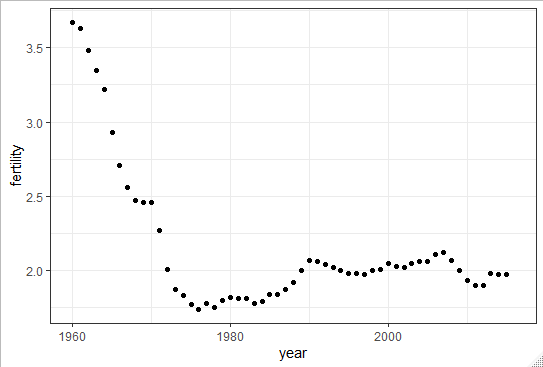
Scatter plot of life expectancy versus fertility from 1962 to 2012



* + 1. Time Series Plots

The visualizations we have just seen effectively illustrate that data no longer supports the Western versus developing worldview. Once this question is answer other emerges. Which countries are improving more? which are improving less? Was the improvement constant during the 50 years? For a closer look that may help answer this question we introduce **time series plot.** They have the time in the x-axis and an outcome or measurement of interest on the y-axis. For example here is the time series of the US fertility rate over the years in our data:

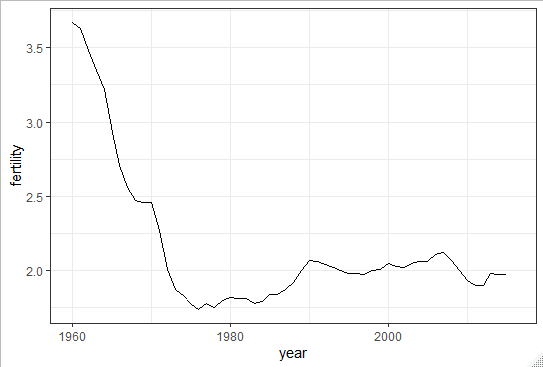
Time series of US fertility rate



We you look at the data, we can clearly see that the trend is no linear at all. Instead you see a sharp drop after the 60s and 70s to below 2%.Then the trend comes up and stabilize over 2%. When the points are regulary spaced and densely packed as they are here, we can create curves by joining the lines. To do this we use the geom\_line function instead of geom\_point(). We write a code like this:

*gapminder%>%filter(country=="United States")%>%ggplot(aes(x=year,y=fertility))+geom\_line()*

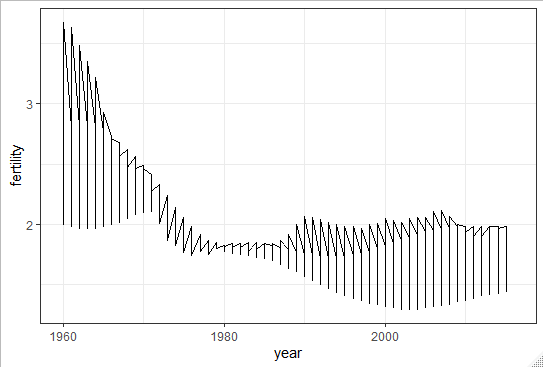
Time series curve of the US fertility



This is particular useful when comparing data, let’s subset the data and look at one from Europe and one from Asia:

*gapminder%>%filter(country%in%c("United States","France","Japan"))%>%ggplot(aes(x=year,y=fertility))+geom\_line()*

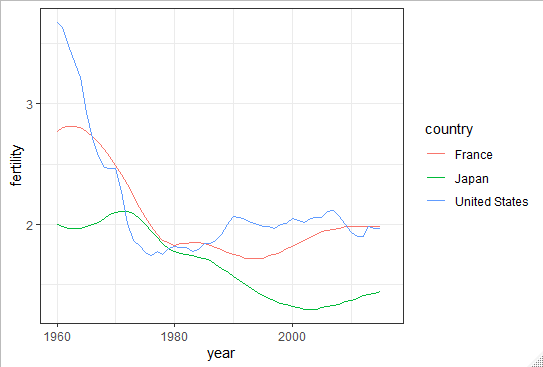
Time series of US, Japan and France fertility between 1960 and 2012



Note that this is not what we want, rather than a line for each country, this code has produced a line that goes through the point for all countries, they’re both joined. This was to be expected because , nowhere in our code do we ask for separate lines. To let ggplot know to separate the lines we assign each point to a group by mapping the country to the group argument in aes like this:

*gapminder%>%filter(country%in%c("United States","France","Japan"))%>%ggplot(aes(x=year,y=fertility,group=country,color=country))+geom\_line()*

fertility rate comparison of France, Japan and US between 1960 and 2012

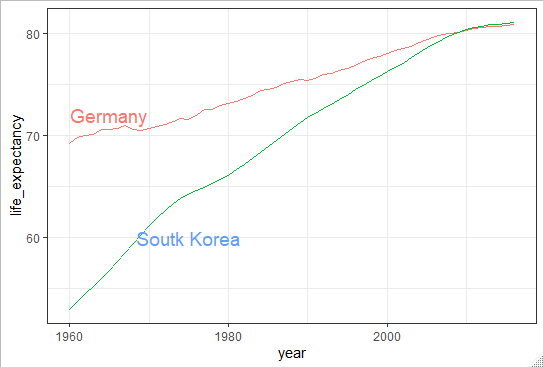


For time series plots, it is recommended to label the curves rather than using legends, this apply to most plots. To label the plots we do:

*labels<-data.frame(country=c("Germany","South Korea"),x=c(1975,1965),y=c(60,72))*

*gapminder%>%filter(country%in%c("Germany","South Korea"))%>%ggplot(aes(x=year,y=life\_expectancy,col=country))+geom\_ line()+geom\_text(data=labels,aes(x,y,label=country),size=5)+theme(legend.position = "none")*

Comparison of Germany and South Korea life expctency over they years



* + 1. Transformations

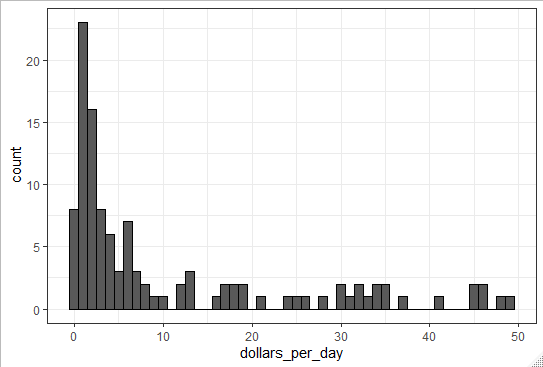
Transformations can be very useful to better understand distributions. As an example let’s look at the income. The gapminder data table includes a column with the country’s gross domestic product GDP. Here we divide the quantity by 365 to obtain a the more interpretable measure dollar/day. Using current US dollars as a unit, a person surviving on an income of less than $2 a day, for example is defined to be living is absolute poverty. So we are going to add this variable to our data table.

*gapminder<-gapminder%>%mutate(dollars\_per\_day=gdp/population/365)*

let’s look at the distribution if you type:

*gapminder%>%filter(year==past\_year & !is.na(gdp))%>%ggplot(aes(dollars\_per\_day))+geom\_histogram(binwidth = 1,color="black")*

Histogram of per day incomes from 1970

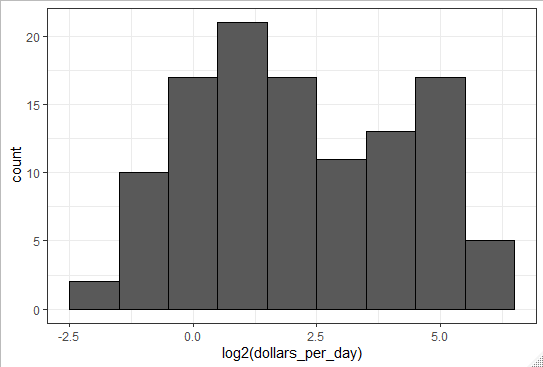


We see that most countries averages are below $10 a day. However, the majority of the x-axis is dedicated to 35 countries with average above 10. It might be more informative to quickly be able to see how many countries make on average about $1 a day, $2 a day, $3 day, $4 a day… these changes are multiplicative, so we introduce log transformations.

**Log transformations** changes multiplicative changes into additive ones. Using base 2 for, example, means that every time a value doubles the transformation increases by one. So to get the distribution of the log base 2 transformed values, we simply transform the data and use the same code like this:

*gapminder%>%filter(year==past\_year & !is.na(gdp))%>%ggplot(aes(log2(dollars\_per\_day)))+geom\_histogram(binwidth = 1,color="black")*

Histogram of the log2 dollar per day in 1970



Here, we see something new, we have 2 clear bumps. In statistics, these bumps are sometimes referred to as modes. The mode of a distribution is the value with the highest frequency. The mode of a normal distribution is the average. But if the mode is a value with the highest frequency, how can we have more than one? When a distribution doesn’t monotically decrease from the mode, we call the location where it goes up and down again as local modes, and we say that the distribution has multiple modes.

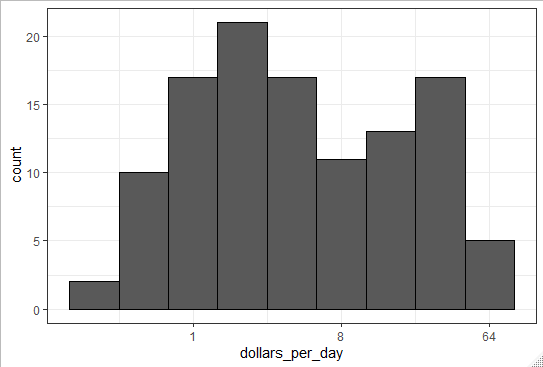
The previous histogram shows that we have 2 modes, one at about $2 a day (1 in log2 scale) and another at about $32(5 in log2 scale), which is consistent with the dichotomic world of 1970.

We choose the base of our log depending on the easiest way to calculate it.

There is 2 way to use log transformations in a plot. We can log the values before plotting them or we can use log scales in the axis. Both approaches are useful and have different strengths. If we log the data we can more easily interpret intermediate values in the scale. The advantage of using log scales is that we see the original values on the axis. To remake the previous histogram with scale transformed we do:

*gapminder%>%filter(year==past\_year & !is.na(gdp))%>%ggplot(aes(dollars\_per\_day))+geom\_histogram(binwidth = 1,color="black")+scale\_x\_continuous(trans="log2")*

Histogram of dollar per day in 1970 with logged scale



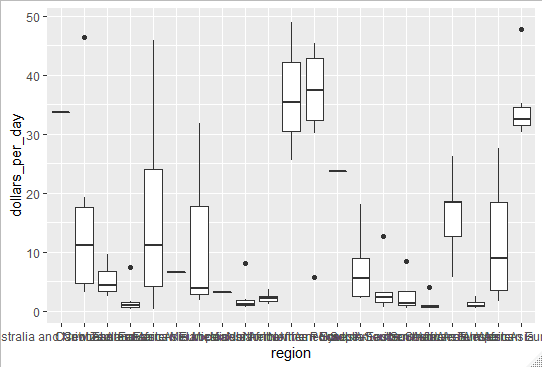
* + 1. Stratify and Boxplot

The histogram showed us that income distribution values show a dichotomy. However we don’t know the 2 groups. To see distributions by geographical region, we first stratify the data into regions and then examine the distribution for each. Because of the high number of regions (22 in our case) looking at histograms or smooth densities will not be useful. Instead we can stack box plots next to each other with this code:

*p<-gapminder%>%filter(year==past\_year & !is.na(gdp))%>%ggplot(aes(region,dollars\_per\_day))*

*p+geom\_boxplot()*

Boxplots of dollar per day by region in 1970

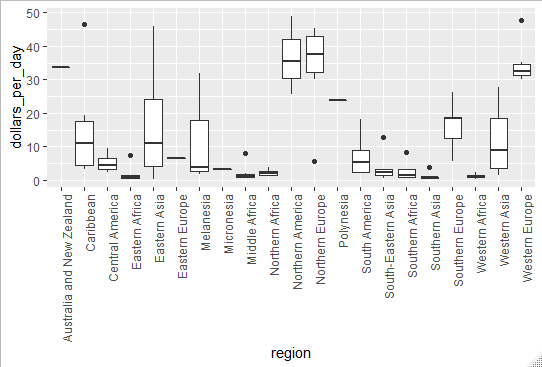


As you can see, there is some small problem with this plot. First we cannot read the region names, the default behavior is to write labels horizontally and we run out of space to write the name regions, we should turn them vertically.

We can rotate the names by changing the theme through element\_text. Like this:

*p+geom\_boxplot()+theme(axis.text.x = element\_text(angle=90,hjust = 1))*

Readable x label boxplots of dollars per day by regions in 1970



The hjust equals 1 argument justifies the text. Second it helps to order the regions in some other order that the alphabetical order, we are going to order by something more meaningful. We are going to use the **reorder()** function which let us change the order of the levels of a factor variable base on a summary computed on a numeric vector. Let’s open a parenthesis in our example to show a bit how the reorder function works. We define a factor based on a vector with 5 entries:

*fac<-factor(c("Asia","Asia","West","West","West"))*

If we turn this vector into factor the levels of this factor are ordered alphabetically:

*levels(fac)*

*[1] "Asia" "West"*

Suppose now that each of the elements are associated with a value for example:

*value<-c(10,11,12,6,4)*

let’s suppose that we want to order the levels based on the mean value of these numbers, in this case the west has a lower mean, so if we use reorder like this:

*fac<-reorder(fac,value,FUN=mean)*

*levels(fac)*

*[1] "West" "Asia"*

You can see that the levels are ordered differently, because west has a smaller mean value of the value vector.

Let’s go back to our example. In our example the regions are the different parts of a continent, we also have a variable for the continents, and then we have divided the world into West versus the rest. So we have 3 different ways of diving the data.

The first thing we’re going to do is to simply reorder the regions by their median income level, like this:

*p<-gapminder%>%*

*+filter(year==past\_year&!is.na(gdp))%>%*

*+mutate(region=reorder(region,dollars\_per\_day,FUN=median))%>%*

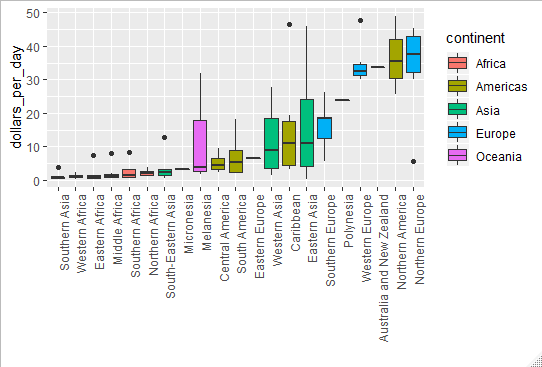
*+ggplot(aes(region,dollars\_per\_day,fill=continent))+geom\_boxplot()*

*+theme(axis.text.x = element\_text(angle=90,hjust=1))*

*+xlab("")*

*p*

Boxplots comparison of dollars per day by region in 1970 group by continent

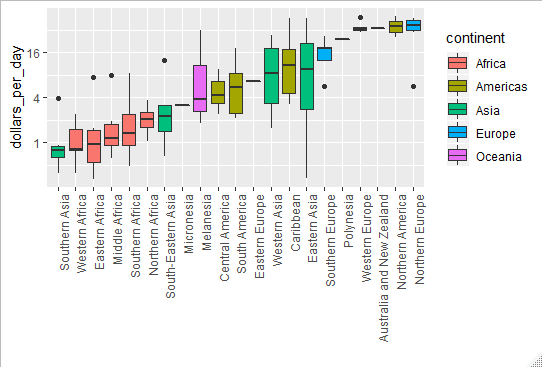


it is the same code as before, but we add to mutate that changes regions to a new factor where the levels are reordered.

The last thing we can do to have a better plot is to change the scale to a log scale. We want to change it to log2 scale, so we add the layer scale\_continuous and we use log2 transformation. Like this:

*p+scale\_y\_continuous(trans= « log2 »)*

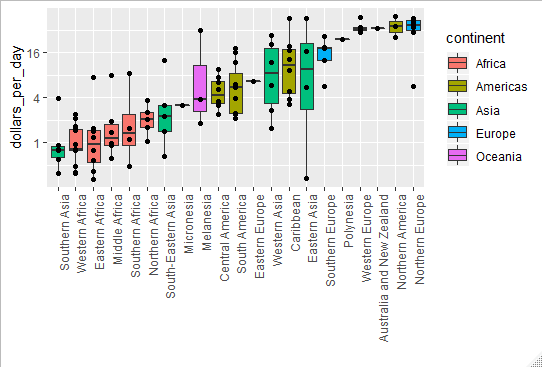
Log scaled Boxplots comparison of dollars per day by region in 1970 group by continent



We can see now a real differences between African regions and Asian regions for example. The last thing we can do is to show the data, in most case we don’t want to show the data, because it add to much clutter to the plot, but in this case we don’t have that much data so it’s a good thing to do. We do it like this:

p*+scale\_y\_continuous(trans=”log2”)+geom\_point(show.legend = FALSE)*

Log scaled Boxplots comparison of dollars per day by region in 1970 group by continent with data points



* + 1. Comparing Distributions

Now that we have confirmed that there was a differences in income in 1970 between the regions in the West and the rest. So to continue our analysis we are going to define a vector representing the West regions like this:

*West<-c(“Western Europe”,”Northern Europe”,”Southern Europe”,”Northern America”,”Australia and New Zealand”)*

Now we want to focus on comparing the differences in distribution across time. We start by confirming that the bi modality observed in 1970 is explained by a west versus developing world economy. We do this by creating a **histog**ram for the groups previously defined. Like this:

*gapminder%>%*

*+filter(year==past\_year & !is.na(gdp))%>%*

*+mutate(group=ifelse(region%in%west,"West","Developing"))%>%*

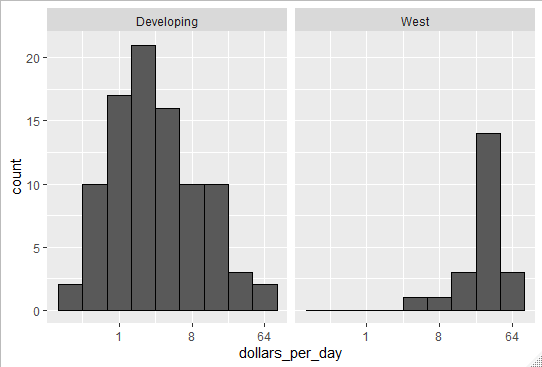
*+ggplot(aes(dollars\_per\_day))+geom\_histogram(binwidth = 1,color="black")*

*+scale\_x\_continuous(trans="log2")*

*+facet\_grid(.~group)*

We use facet\_grid to have histogram for each group

Histograms of dollar per day distribution of West and Developing region in 1970



We can see immediately that the West regions have a distribution shifted to the right, indicating higher income, that the Developing region which have a distribution shifted to the right.

Now we can check if this separation is worst today by writing the same code but faceting by both region and year. Like this:

*gapminder%>%filter(year%in%c(past\_year,present\_year) & !is.na(gdp))%>%*

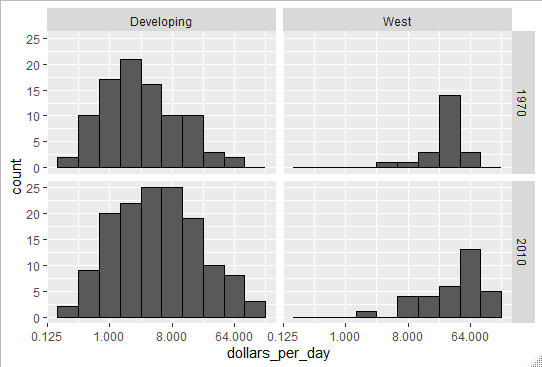
*+mutate(group=ifelse(region%in%west,"West","Developing"))%>%*

*+ggplot(aes(dollars\_per\_day))+geom\_histogram(binwidth = 1,color="black")*

*+scale\_x\_continuous(trans="log2")*

*+facet\_grid(year~group)*

Histograms comparison of dollar per day distribution of West and Developing region in 1970 and 2010



We can observe that the Developing region have shifter more to right between 1970 and 2010 than the West regions, which means that they got closer. It is important to note, before continuing commenting the plots, that there are more countries represented in the 2010 histograms that in 1970 ones. One reason for that is that many countries were founded after 1970, for example the Soviet union turned into several countries, including Russia and Ukraine in the 90’s. Another reason is that data is available for more countries in 2010 compared to 1970. So we’re going to remake the plots using only the countries with data available for both years. Like this:

*country\_list\_1<-gapminder%>%filter(year==past\_year & !is.na(dollars\_per\_day))%>%.$country*

*country\_list\_2<-gapminder%>%filter(year==present\_year & !is.na(dollars\_per\_day))%>%.$country*

*country\_list<-intersect(country\_list\_1,country\_list\_2)*

there’s 108 countries in this list, which account for 86% of the total population. So we can say that this subset should be representative of the entire world.

Let’s make the plot again but this time using only the subset of countries we just created. Like this:

*gapminder%>%*

*+filter(year%in%c(past\_year,present\_year) & country %in% country\_list)%>%*

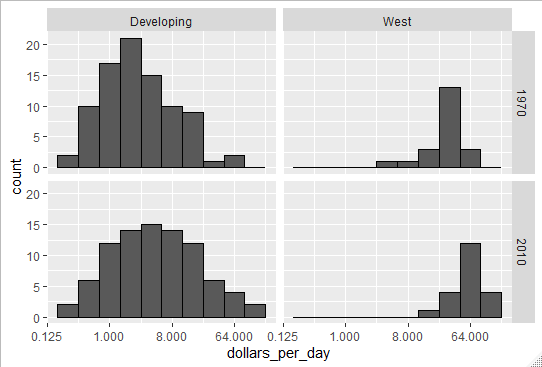
*+mutate(group=ifelse(region%in%west,"West","Developing"))%>%*

*+ggplot(aes(dollars\_per\_day))+geom\_histogram(binwidth = 1,color="black")*

*+scale\_x\_continuous(trans="log2")*

*+facet\_grid(year~group)*

Histograms comparison of dollar per day distribution of West and Developing region in 1970 and 2010 without missing data



We can observe now that, even if the West countries have shifted to the right, the proportions of developing countries that have increase their income over the years is bigger. We can specially note the amount of developing country earing more than $16 a day.

To see which specific regions improves the most, we can remake the box plots that we made earlier but now adding 2010. We do it by :

*p<-gapminder%>%*

*+filter(year%in%c(past\_year,present\_year)&country%in% country\_list)%>%*

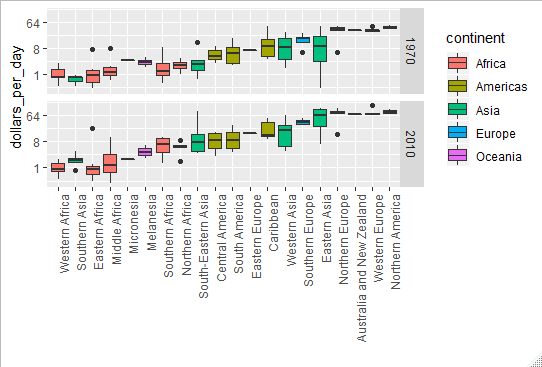
*+mutate(region=reorder(region,dollars\_per\_day,FUN=median))%>%*

*+ggplot()+theme(axis.text.x = element\_text(angle=90,hjust=1))*

*+xlab("")+scale\_y\_continuous(trans="log2")*

*p+geom\_boxplot(aes(region,dollars\_per\_day,fill=continent))+facet\_grid(year~.)*

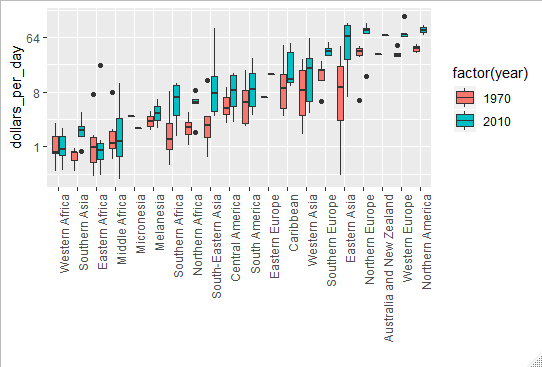
Boxplots comparison of regions income distribution between 1970 and 2010



Drawing conclusions from those plots, is hard it will be easier to have them next to each other. To do this we introduce a new ggplot function. So instead of faceting, we keep the data from each year together, but ask ggplot to color or fill the box block depending of the year. Ggplot automatically separates them and puts the two box plots next to each other. Because year is a number we turn it into a factor so that each is a category. Like this:

*p+geom\_boxplot(aes(region,dollars\_per\_day,fill=factor(year)))*

Boxplots, of 1970 and 2010, comparison of regions income distribution



Now we can see which countries have improved the most. Look at Eastern Asia, for example, it went from around $8 a day in 1970 to almost $64 a day in 2010

* + 1. Density Plots

By data exploration we discovered that the income gab between rich and poor countries has closed considerably during the last forty years. We can convey this message with just one plot, the smooth density plots. One reason that can explain this change in distribution is that poor countries became richer rather than some rich countries becoming poorer. To prove this all we need to do is we assign a color to the groups we identified. But before that we need to learn how to make these smooth densities in a way that preserves information of how many countries are in each group. If we overlay the two densities, the default is to have the area represented by each distribution add up to 1 regardless of the size of the group, which in our case make it seems like there is the same number of countries in each group which is incorrect, to change this we need to access computed variables with the **geom\_density()** function. To have the areas of the densities be proportional to the size of the groups, we can simply multiply the y-axis values by the size of the group. From the geom\_density documentation we learn that there is a variable called count that does exactly this, to access it we surround the name of the variable with “..” like this:

*aes(x=dollars\_per\_day,y=..count..)*

*Aesthetic mapping:*

*\* `x` -> `dollars\_per\_day`*

*\* `y` -> `..count..`*

Now if we change the mapping in the previous code:

*p<-gapminder%>%*

*+filter(year%in%c(past\_year,present\_year)&country%in%country\_list)%>%*

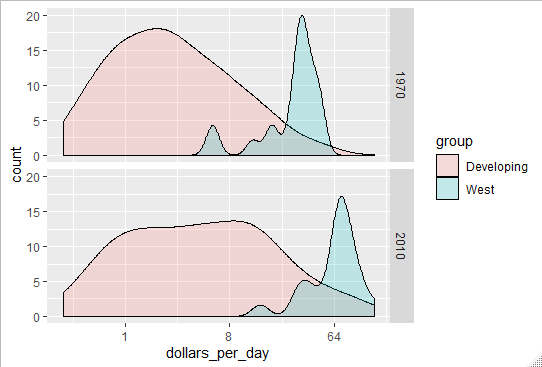
*+mutate(group=ifelse(region %in% west, "West","Developing"))%>%*

*+ggplot(aes(dollars\_per\_day,y=..count..,fill=group))*

*+scale\_x\_continuous(trans="log2")*

*> p+geom\_density(alpha=0.2)+facet\_grid(year~.)*

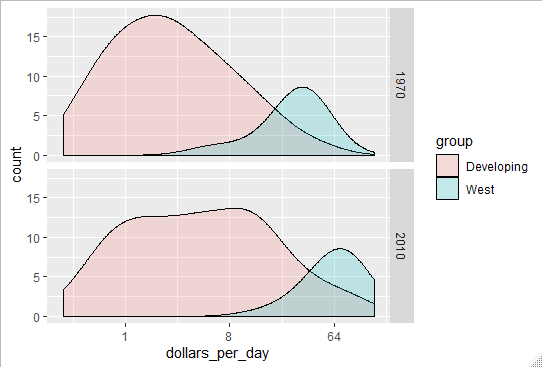
Smooth density plots comparison of Developing countries vs West countries between 1970 and 2010



Notice that now we can clearly see that the developing world has more countries. If you want the densities to be smoother, specially for the West countries where there is a lot of unsmoothness, we can change the bw argument like this:

*p+geom\_density(alpha=0.2,bw=0.75)+facet\_grid(year~.)*

Smooth density plots comparison of Developing countries vs West countries between 1970 and 2010 with smoothing



We can clearly see the developing countries distribution changing. A third mode appears consisting of the countries that most closed the gap. From our data exploration we discovered that most improved countries were from Asia, we can easily alter the plot to show key regions separately, with the **case\_when()** function here how it works:

*gapminder<-gapminder%>%*

*+mutate(group=case\_when(*

*+.$region %in% west ~ "West",*

*+.$region %in% c("Eastern Asia","South-Eastern Asia") ~ "East Asia",*

*+.$region %in% c("Caribbean","Central America","South America") ~ "Latin America",*

*+.$continent=="Africa" & .$region !="Northern Africa" ~ "Sub-Saharan Africa",*

*TRUE ~ "Others"))*

We just created a new variable describing the groups that interest us. Then we just turn this group variable into a factor, we do it like this:

*gapminder<-gapminder%>%*

*+mutate(group=factor(group,levels=c("Others","Latin America","East Asia","Sub-Saharan Africa","West")))*

Now we can modify a bit the code for the smooth density to include the new group and stacked them on the top of each other like this:

*gapminder%>%*

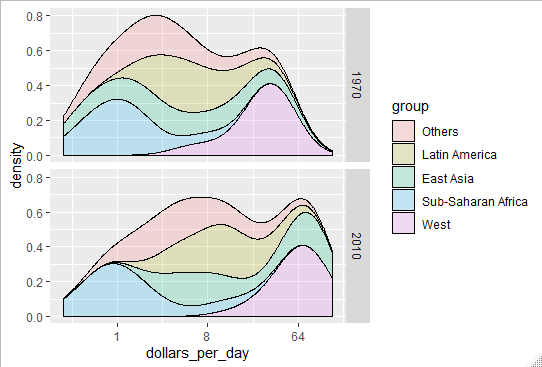
*+filter(year%in%c(past\_year,present\_year)& country %in% country\_list)%>%*

*+group\_by(year)%>%ggplot(aes(dollars\_per\_day,fill=group))*

*+scale\_x\_continuous(trans = "log2")*

*+geom\_density(alpha=0.2,bw=0.75,position="stack")+facet\_grid(year~.)*

Stacked density plots comparison of countries income between 1970 and 2010



* + 1. Ecological Fallacy

Here we are going to focus on the importance of describing the variability within the groups. While we do this, we’ll also discover some other ggplot functions as well as transformation called the **logit transformation**, which is useful for the data that we’ll be looking at. As an example we’ll look at the relationship between child survival rates and average income. We are going to define a few more regions using the case\_when like this:

*gapminder<-gapminder%>%*

*+mutate(group=case\_when(*

*+.$region %in% west ~"The West",.$region %in% "Northern Africa"~"Northern Africa",*

*+.$region %in% c("Eastern Asia","South-Eastern Asia")~"Eastern Asia",*

*+.$region=="Southern Asia"~"Southern Asia",.$region %in% c("Central America","South America","Caribbean")~"Latin America",*

*+.$continent=="Africa"&.$region!="NorthernAfrica"~"Sub-SaharanAfrica",*

*+.$region %in%c("Melanesia","Micronesia","Polynesia")~"Pacific Islands"))*

Once this is done we can compute the quantities we are interested in for each group :

*surv\_income<-gapminder%>%*

*+filter(year%in%present\_year&!is.na(gdp)&!is.na(infant\_mortality)& !is.na(group))%>%*

*+group\_by(group)%>%*

*+summarize(income=sum(gdp)/sum(population)/365,*

*+ infant\_surviva\_rate=1-sum(infant\_mortality/1000\*population)/sum(population))*

*surv\_income%>%arrange(income)*

*# A tibble: 7 x 3*

*group income infant\_surviva\_ra~*

*<chr> <dbl> <dbl>*

*1 Sub-Saharan Africa 1.76 0.936*

*2 Southern Asia 2.07 0.952*

*3 Pacific Islands 2.70 0.956*

*4 Northern Africa 4.94 0.970*

*5 Latin America 13.2 0.983*

*6 Eastern Asia 13.4 0.985*

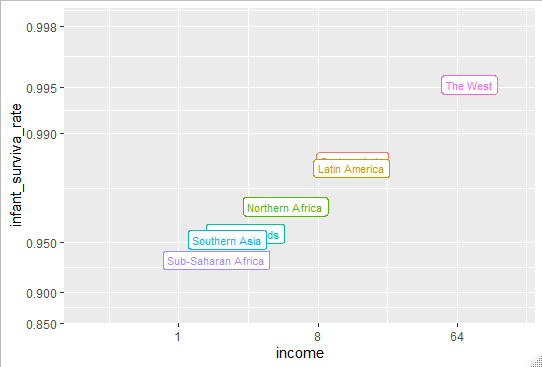
*7 The West 77.1 0.995*

This shows a dramatic difference. While in the West less than 0.5% of children die, in sub-saharan Africa the rate is higher than 6%. In fact it looks like the relationship between the two is perfectly linear.

With this command :

*surv\_income%>%ggplot(aes(income,infant\_surviva\_rate,label=group,color=group))+scale\_x\_continuous(trans="log2",li mit=c(0.25,150))+scale\_y\_continuous(trans="logit",limit=c(0.875,0.9981),breaks=c(0.85,0.90,0.95,0.99,0.995,0.998))+ geom\_label(size=3,show.legend = FALSE)*

Region average income vs infant survival rate

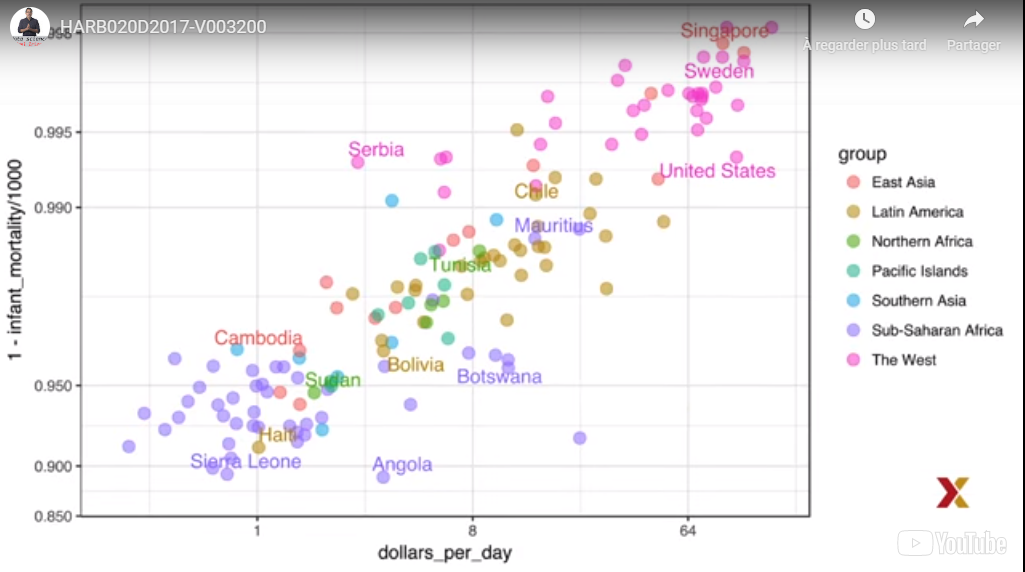


**The logistic or logit transformation** for a proportional rate p is defined as follows:

When p is a proportion or probability, the quantity that is being logged p divided by 1 minus p, is called the odds. And the case p is the proportion of children that survive, the odds tell us how many more children are expected to survive than to die. The log transformation makes this quantity symmetric. This scale is useful when **we want to highlight differences that are near 0 or near 1.** For survival rates, this important because a survival rate of 90% is unacceptable while the survival rate of 99% is relatively good. We want to highlight this difference.

Now if we conclude based on that plot that all survival rates in Sub-saharan Africa countries is lower to all Southern Asia is what we call an **ecological fallacy**. The perfect linear relationship between infant survival rate and income is only true at the region level if we do the same with the actual data points here is the result :

Region income vs infant survival rate



1. Section 5: Data Visualization Principles
   1. Data Visualization Principles, Part 1
      1. Introduction to Data Visualization Principles

Here we aim to provide some general principles we can use as guidelines for effective data visualization. We going to show some examples of plot styles we should avoid, explain how to improve them and use these as motivation for a list of principles. We are going to compare plots that follow these principles to those that don’t.

The principles are related to how humans detect patterns and make visual comparisons. The preferred approaches are those that best fit the way our brain process visual information. Furthermore, it is important to note that as a data scientist you must adapt and optimize graphs to the audience. For example, and exploratory plot made for us will be different than a chart intended to communicate a finding to a general audience.

* + 1. Encoding Data Using Visual Cues

First, we start by describing some principles for encoding data. There are several approaches to our disposal, including:

Position

Aligned lengths

Angles

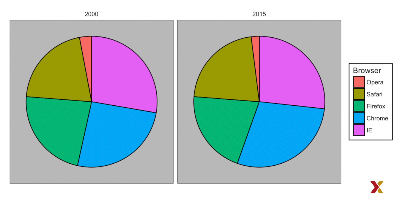
Area

Brightness

Color hue

In our first example. To illustrate how some of these strategies compare, let’s suppose we want to report the results from two hypothetical polls, asking what your browser preference is. Polls were taken in 2000 and 2015. Here, for each year, we are simply comparing four quantities, four percentage. One widely used graphical representation in this case, popularize by Microsoft Excel, is the pie chart. Here’s what it looks like:

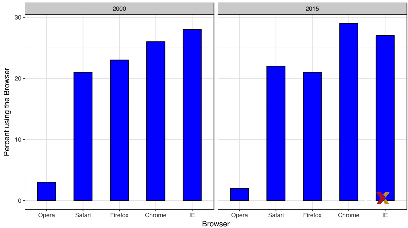
Pie chart comparison of browser preference between 2000 and 2015



Here we are representing data with both angle and area, since both the angle and area of each pie slice is proportional to the quantity it represents. This turns out to be a suboptimal choice, since as demonstrated by perceptions studies, humans are not good at precisely quantifying angles, and are even worse when only area is available. From that chart, it’s not easy to identify the percentage of each browser. One solution will be to just show the table with the data, which will save some paper and ink.

If the graphical visualization is still necessary, we will choose to use length and positions, since humans are much better at judging linear measures, which is why the bar plot is preferred in this case:

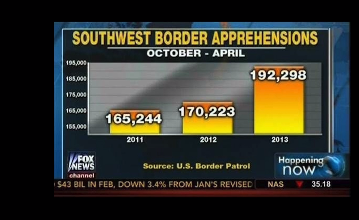
Barplot chart comparison of browser preference between 2000 and 2015



* + 1. Know When to Include Zero

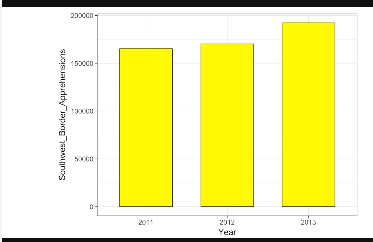
When using bar plots, not including 0 is dishonest. This is because by using bar plot, we are implying the length is proportional to the quantities being displayed. By avoiding 0, relatively small differences can be made to look much bigger that they actually look. This is often used by politicians or media organizations trying to exaggerate the difference. Here is an illustrative example:

Box plot made by FoxNews showing southwest border apprehensions in 2011, 2012 and 2013



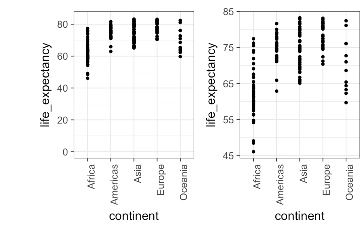
If you look at the 2013 bar compared to the 2011 it appears that apprehensions have almost tripled, when in fact, if you look at the number they only increased by 16%, if we start the plot at 0 here what it looks like:

Box plot showing southwest border apprehensions in 2011,2012 and 2013



When using position instead of length, it is not necessary to include 0. This is particularly the case when we want to compare differences between groups relative to the variability seen within the groups. Here is an example:

Country average life expectancies stratified into continents with 0 inclusion (on the left) and without 0 inclusion (on the right)

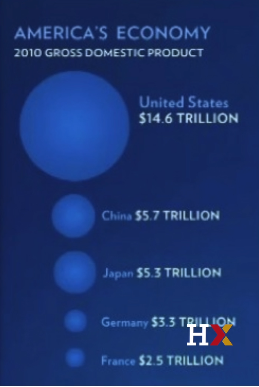


In this case the 0 inclusion does not add more information to the graph, and is actually making it harder to read.

* + 1. Do Not Distort Quantities

The next principle is to not distort quantities, for example here is a chart:

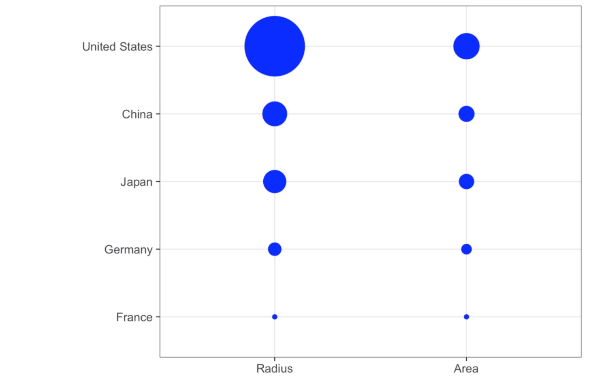
US economy comparison with 4 competing nations with distorted area



If we look at this chart, it seems that US economy is 5 larger than China and over 30 times larger than France, however when you look at the actual number the ratio is more 2.6 and 5.8 bigger than China and France respectively.

The reason for the distortion is that the chart uses the radius of the cercle instead of the area, which implies that the area value is squared. 2.6 becomes 6.5 and 5.8 becomes 34.1. Here’s the comparison with the correct area:

US economy comparison with 4 competing nations

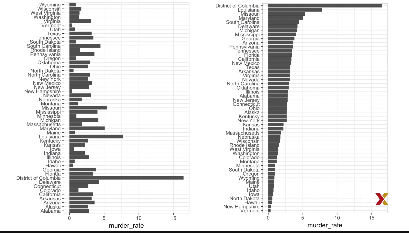


Ggplot defaults is to use area instead of radius.

* + 1. Order by a Meaningful Value

The next principle is ordering the value. When one the axes is used to show categories, as done in a bar plot, the default ggplots behavior is to order the categories alphabetically when they are defined as character strings. If they are defined by factors, they are ordered by factor levels. The default for ordering factor level is the alphabetical order. That order for the factor is arbitrary we should order by a meaningful quantity. We previously learned how to use reorder function to helps us changed the order of a factor. To appreciate how the right order can help convey message, let’s take an example, we want to create a plot to compare the murder rates across states. We’re particularly interested in the most dangerous sand the safest states.

Murder rates across states with default ordering (on the left) and ordering by the rate (on the right)

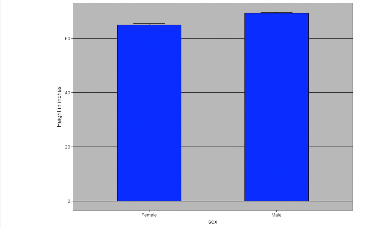


The information is much easier to read on the right that on the left.

* 1. Data Visualization Principles, Part 2
     1. Show the Data

We have focus on displaying single quantities across categories. We now shift our attention to displaying data with a focus on comparing groups. To illustrate that we go back to our artificial example, describing heights to E.T. This time, let’s assume ET is interested in the difference in heights between males an females. A commonly used plot for comparing groups, popularized by software such as Microsoft Excel, shows the average and the standard error (Not the same as standard deviation).

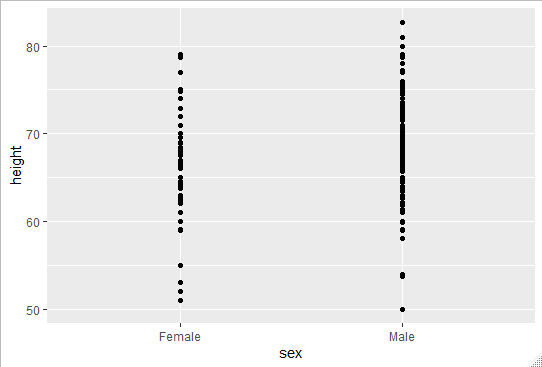
Average of male and females with standard error



The average is represented by the top of each bar and the antenna that we see that expands out is the average plus 2 standard errors. If all ET receives is this plot, he will have little information on what to expect if he meets a group of humans. The following ggplot code gives more information that that plot:

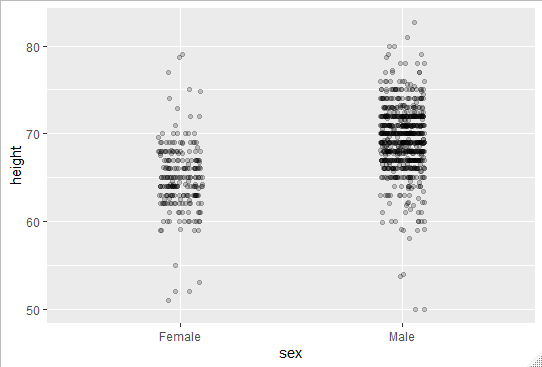
*heights%>%ggplot(aes(sex,height))+geom\_point()*

scatter point plot of male and female heights



However, this plot has limitations as well. Since we can’t really see all the 216 and 708 points plotted for females and males, furthermore a lot of points are on top of each other so it is not easy to differentiate between the plots. We can improve this plot by adding jitter (a small random shift to each point). In this case, adding horizontal jitter does not alter the interpretation since the height of the points doesn’t change. A second change is to add an alpha blending making the point a bit transparent, here’s the same plot with alpha blending and jitter:

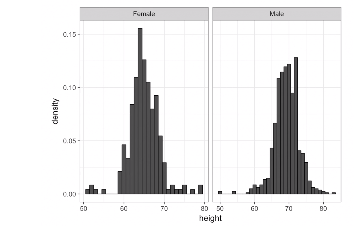
scatter point plot of male and female heights with jitter and alpha



* + 1. Ease Comparisons: Use Common Axes

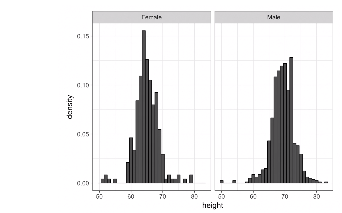
We’ve seen that the previous plot to compare male and female heights is not good because it doesn’t show all the points. In our case the most effective way to communicate the difference between the groups is looking at the distribution of each group. So let’s start by doing an histogram of each group:

Histogram of male and female heights distribution with different X-axis



From this histogram, it is not immediately clear that males are , on average, taller that women. We have to look carefully to notice that the x-axis has a higher values in the male histogram. Which lead us to a another princinple, always keep the axes the same when comparing data across plots.

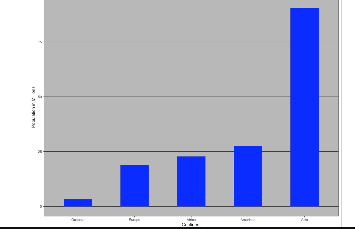
Histogram of male and female heights distribution with same X-axis



* + 1. Consider Transformations

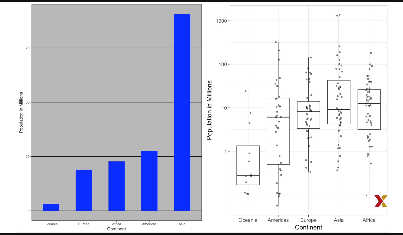
We have motivated the use of the log transformation in cases where the changes are multiplicative. Population size was an example in which we found a log transformation to yield a more informative plot. The combination of incorrectly using bar plots, when a log transformation is merited, can be particularly distorting, here is an example:

Bar plot of average population sizes for each continent in 2015



From this plot, one could conclude that countries in Asia are much more populous than other continents. Following the show-the-data principle, we quickly notice that this is due to 2 countries (India and China). Here using the log transformation provides a much more informative plot. We compare the original bar plot to a box plot using the log-scale transformation for the y-axis:

Boxplot log scaled of average population sizes for each continent in 2015

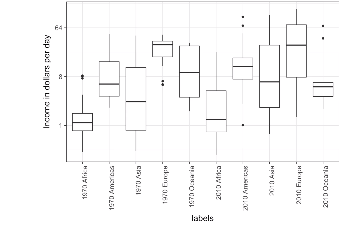


Now you can see the 2 outliers in Asia distribution and actually see that Africa has a higher median population size than Asia

* + 1. Ease Comparisons: Compared Visual Cues Should Be Adjacent

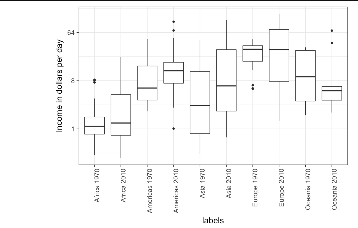
When comparing income data between 1970 and 2010, across regions, we made a figure like this one:

Comparison of income between 1970 and 2010 across continents



The default in ggplot is to order alphabetically, so the labels with 1970 come before the labels with 2010, making that comparison challenging. If we reorder the boxplots we want to compare next to each other is much easier to read, like this:

Comparison of income between 1970 and 2010 across continents with reordering of labels



* 1. Data Visualization Principles, Part 3
     1. Slope Charts

Every time we have examined the relationship between 2 variables, we have used scatter plots. In general that rules is true, there is one exception though, when you are comparing variables of the same type but at different time points and for a relatively small number of comparison. For example, comparing life expectancy between 2010 and 2015. In this case we might consider creating a **slop chart**. There`s no geometry for slope charts in ggplot2, but we can construct one using the geom\_line. Like this:

*west<-c("Western Europe","Northern Europe","Southern Europe","Northern America","Australia and New Zealand")*

*dat<-gapminder%>%*

*+filter(year%in% c(2010,2015)&region %in% west & !is.na(life\_expectancy)& population>10^7)*

*dat%>%*

*+mutate(location=ifelse(year==2010,1,2),*

*+location=ifelse(year==2015&country%in%c("United Kingdom","Portugal"),*

*+location+0.22,location),*

*+hjust=ifelse(year==2010,1,0))%>%*

*+mutate(year=as.factor(year))%>%*

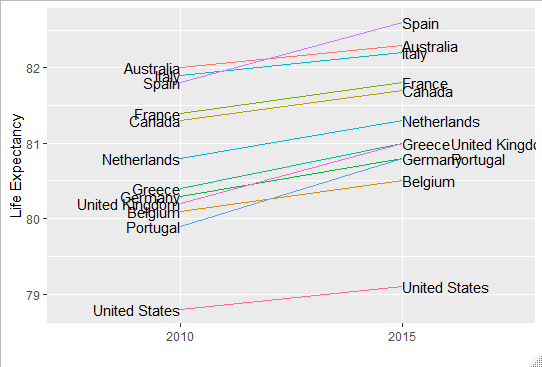
*+ggplot(aes(year,life\_expectancy,group=country))*

*+geom\_line(aes(color=country),show.legend = FALSE)*

*+geom\_text(aes(x=location,label=country,hjust=hjust),show.legend=FALSE)*

*+xlab("")+ylab("Life Expectancy")*

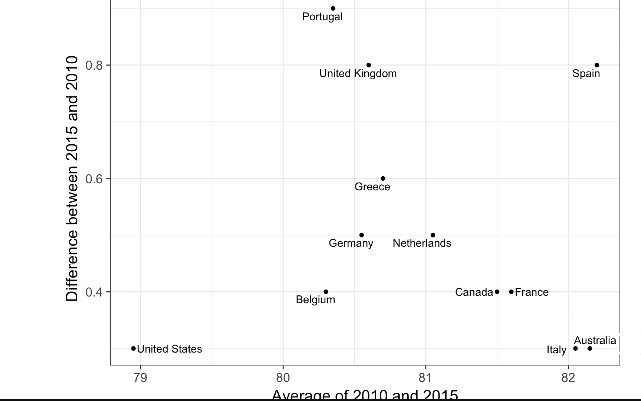
Slope Chart of life expectancy between 2010 and 2015



The advantage of those charts is that you get an idea of the evolution with the slope of the line. Note that when you have a lot of points the slope chart become useless and will prefer a scatter plot.

Finally we are going to see the Bland-Altman plot also know as the MA plots, which shows the difference versus the average. Here is an example:

Bland-Altman plot of difference between 2010 and 2015



And here is how to write it in R:

*library(ggrepel)*

*dat %>%*

*mutate(year = paste0("life\_expectancy\_", year)) %>%*

*select(country, year, life\_expectancy) %>% spread(year, life\_expectancy) %>%*

*mutate(average = (life\_expectancy\_2015 + life\_expectancy\_2010)/2,*

*difference = life\_expectancy\_2015 - life\_expectancy\_2010) %>%*

*ggplot(aes(average, difference, label = country)) +*

*geom\_point() +*

*geom\_text\_repel() +*

*geom\_abline(lty = 2) +*

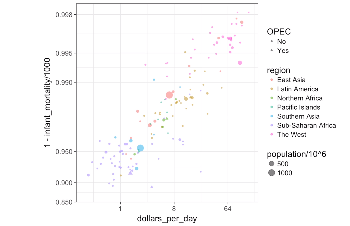
*xlab("Average of 2010 and 2015") +*

*ylab("Difference between 2015 and 2010")*

* + 1. Encoding a Third Variable

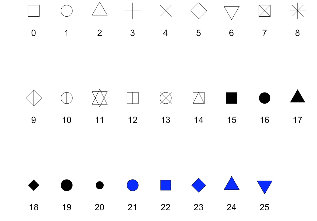
Previously we showed a scatter plot showing the relationship between infant survival rates and average income. Here’s a version where we encoded 3 more variables, OPEC membership, region and population size.

Scatter plot of income per day vs infant mortality rate , with OPEC membership, region and population size



Note that we encoded categorical variables with color hue and shape. They can be controlled with the shape argument, here is the list of all shapes available:

List of all shapes for the shape argument in R



* + 1. Case Study: Vaccines

In the 19th century, before herd immunization was achieved through vaccination programs, deaths from infectious diseases, like smallpox and polio, were common. However, today despite all the scientific evidence for their importance, vaccination programs have become somewhat controversial. The controversy started with a paper published in 1988 and led by Andrew Wakefield claiming there was a link between MMR vaccine appearance of autism and bowel disease. With effort from sensationalist media reports and fears mongering from conspiracy theorists lead parts of the public to believe vaccines were harmful. Some parents even stopped vaccinating their children. Earlier we showed an example provided by the Wall Street journal showing data related to the impact of vaccines on battling infectious diseases, here we are going to reconstruct that example.

We are going to use the **us\_contagious\_diseases** data set included in dslabs:

*> data("us\_contagious\_diseases")*

*> str(us\_contagious\_diseases)*

*'data.frame': 18870 obs. of 6 variables:*

*$ disease : Factor w/ 7 levels "Hepatitis A",..: 1 1 1 1 1 1 1 1 1 1 ...*

*$ state : Factor w/ 51 levels "Alabama","Alaska",..: 1 1 1 1 1 1 1 1 1 1 ...*

*$ year : num 1966 1967 1968 1969 1970 ...*

*$ weeks\_reporting: int 50 49 52 49 51 51 45 45 45 46 ...*

*$ count : num 321 291 314 380 413 378 342 467 244 286 ...*

*$ population : num 3345787 3364130 3386068 3412450 3444165 ...*

For the plot we want to produce we are going to create a temporary object called dat that stores all the measles data:

*the\_disease<-"Measles"*

*>dat<-us\_contagious\_diseases%>%*

*+filter(!state%in%c("Hawaii","Alaska")&disease==the\_disease)%>%*

*+mutate(rate=count/population\*10000)%>%*

*+mutate(state=reorder(state,rate))*

Now we can now easily plot the disease rates per year. Here are the data for California:

dat%>%filter(state=="California")%>%

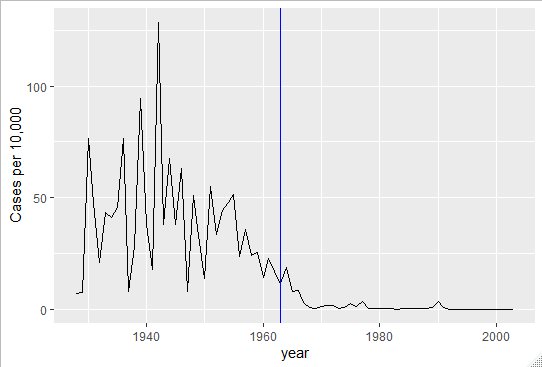
+ggplot(aes(year,rate))

+geom\_line()+ylab("Cases per 10,000")

+geom\_vline(xintercept=1963,col="blue")

We added a vertical line in 1963, the year the vaccine was introduced:

Case of measles in California over the years



Now we want to show data for all states in one plot, we have 3 variables to show, year, state and rate. In the Wall Street journal figure, they use the x-axis for year, the y-axis for state and color hue represent rates. We’re using color to represent a continuous variable. When choosing the color to quantify a numeric variable, we choose between 2 options:

Sequential: palettes are suited for data that goes from high to low

Divergent: diverging color are used to represent values that verge from center (same emphasis on both end of the data range)

In our example we want to use a sequential palette since there is no meaningful center, just low and high rates, we use the geom\_tile to tile the region with colors representing disease rates. And we use square root transformation to avoid having the high counts dominate the plot:

*library(RColorBrewer)*

*> dat%>%ggplot(aes(year,rate,fill=rate))*

*+geom\_tile(color="grey50")*

*+scale\_fill\_gradientn(colors=brewer.pal(9,"Reds"),trans="sqrt")*

*+geom\_vline(xintercept=1963,col="blue")*

*+theme\_minimal()+theme(panel.grid = element\_blank())*

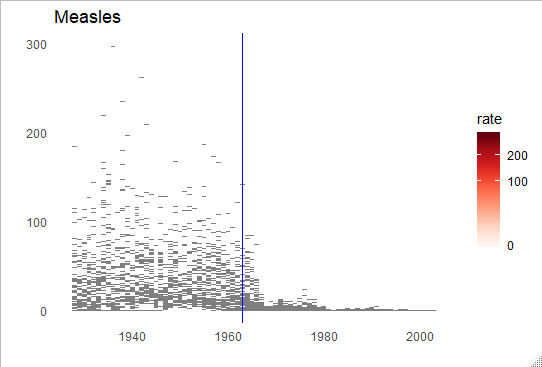
*+ggtitle(the\_disease)*

*+ylab("")*

*+xlab("")*

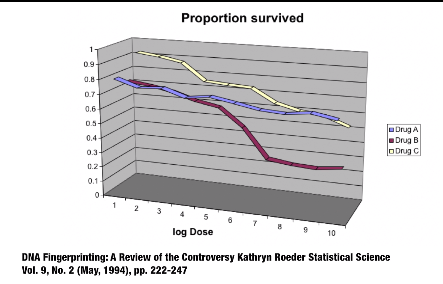
And here is the result:

Case of measles in the US over the years



* + 1. Avoid Pseudo and Gratuitous 3D Plots

Bad example of a 3D graphic

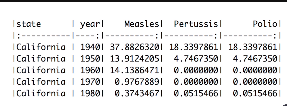


The figure above was taken from scientific literature. It shows 3 variables, dose, drug type and survival. Although when you look at a plot, you’re almost always looking at a screen or a book page, which are both flat and two dimensional, this plot tries to imitate 3 dimensions and assigns a dimension to each variable. Humans are not good at seeing in 3 dimensions and are specially not good with pseudo 3 dimensions. This is an example which it’s easy to use color to represent the categorical value. So avoid 3D plots!!!

* + 1. Avoid Too Many Significant Digits

By default R returns many significant digits, the default for R is 7 significant digits. So many does not add information and the visual clutter makes it hard for the consumer of your table to understand the message. As an example :

Per 10000 disease rates for California



We are reporting position up to 0.00001 cases per 10000 a very small value in the context of the changes that occurs across time. 2 functions to help you change the number of significant digits or to round numbers are **signif()** and **round().** You can define the number of significant digits to use globally by setting an option like this:

Options(digits=n)