# Data Science: Visualization

BOOK TO USE : <https://rafalab.github.io/dsbook/>

## Section 1: Introduction to Data Visualization and Distributions

## Introduction to Data Visualization

### Key points

* Plots of data easily communicate information that is difficult to extract from tables of raw values.
* Data visualization is a key component of exploratory data analysis (EDA), in which the properties of data are explored through visualization and summarization techniques.
* Data visualization can help discover biases, systematic errors, mistakes and other unexpected problems in data before those data are incorporated into potentially flawed analysis.
* This course covers the basics of data visualization and EDA in R using the **ggplot2** package and motivating examples from world health, economics and infectious disease.

### Code

library(dslabs)

data(murders)

head(murders)

## Introduction to Distributions

### Key points

* The most basic statistical summary of a list of objects is its distribution.
* We will learn ways to visualize and analyze distributions in the upcoming videos.
* In some cases, data can be summarized by a two-number summary: the average and standard deviation. We will learn to use data visualization to determine when that is appropriate.

## Data Types

### Key points

* Categorical data are variables that are defined by a small number of groups.
  + - Ordinal categorical data have an inherent order to the categories (mild/medium/hot, for example).
    - Non-ordinal categorical data have no order to the categories.
* Numerical data take a variety of numeric values.
  + - Continuous variables can take any value.
    - Discrete variables are limited to sets of specific values

## Describe Heights to ET

### Key points

* A distribution is a function or description that shows the possible values of a variable and how often those values occur.
* For categorical variables, the distribution describes the proportions of each category.
* A frequency table is the simplest way to show a categorical distribution. Use prop.table() to convert a table of counts to a frequency table. Barplots display the distribution of categorical variables and are a way to visualize the information in frequency tables.
* For continuous numerical data, reporting the frequency of each unique entry is not an effective summary as many or most values are unique. Instead, a distribution function is required.
* The cumulative distribution function (CDF) is a function that reports the proportion of data below a value a for all values of a: F(a)=Pr(x≤a).
* The proportion of observations between any two values a and b can be computed from the CDF as F(b)−F(a).
* A histogram divides data into non-overlapping bins of the same size and plots the counts of number of values that fall in that interval.

### Code

# load the dataset

library(dslabs)

data(heights)

# make a table of category proportions

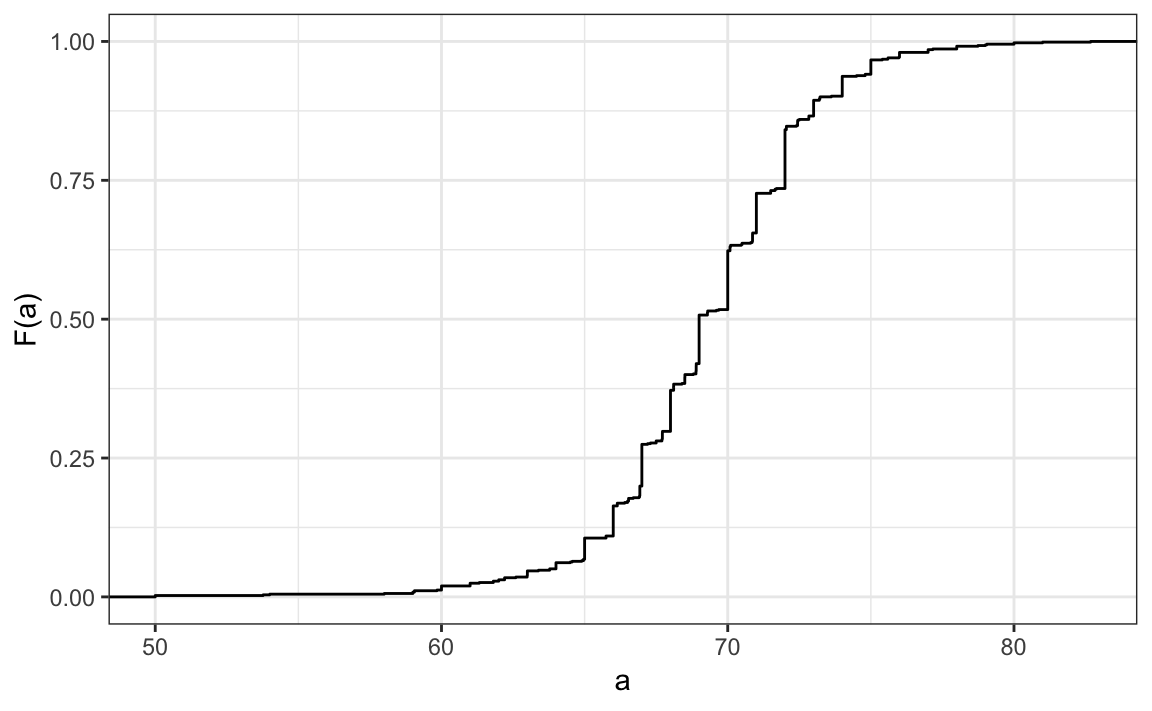
prop.table(table(heights$sex))

## Cumulative Distribution Function

Every continuous distribution has a *cumulative distribution function (CDF)*. The CDF defines the proportion of the data below a given value a for all values of a:

F(a)=Pr(x≤a)

Any continuous dataset has a CDF, not only normal distributions. For example, the male heights data we used in the previous section has this CDF:



As defined above, this plot of the CDF for male heights has height values a on the x-axis and the proportion of students with heights of that value or lower (F(a)) on the y-axis.

The CDF is essential for calculating probabilities related to continuous data. In a continuous dataset, the probability of a specific exact value is not informative because most entries are unique. For example, in the student heights data, only one individual reported a height of 68.8976377952726 inches, but many students rounded similar heights to 69 inches. If we computed exact value probabilities, we would find that being exactly 69 inches is much more likely than being a non-integer exact height, which does not match our understanding that height is continuous. We can instead use the CDF to obtain a useful summary, such as the probability that a student is between 68.5 and 69.5 inches.

For datasets that are not normal, the CDF can be calculated manually by defining a function to compute the probability above. This function can then be applied to a range of values across the range of the dataset to calculate a CDF. Given a dataset my\_data, the CDF can be calculated and plotted like this:

a <- seq(min(my\_data), max(my\_data), length = 100) # define range of values spanning the dataset

cdf\_function <- function(x) { # computes prob. for a single value

mean(my\_data <= x)

}

cdf\_values <- sapply(a, cdf\_function)  
plot(a, cdf\_values)

The CDF defines that proportion of data below a cutoff a. To define the proportion of values above a, we compute:

1−F(a)

To define the proportion of values between a and b, we compute:

F(b)−F(a)

Note that the CDF can help compute probabilities. The probability of observing a randomly chosen value between a and b is equal to the proportion of values between a and b, which we compute with the CDF.

## Smooth Density Plots

### Key points

* Smooth density plots can be thought of as histograms where the bin width is extremely or infinitely small. The smoothing function makes estimates of the true continuous trend of the data given the available sample of data points.
* The degree of smoothness can be controlled by an argument in the plotting function. (We will learn functions for plotting later.)
* While the histogram is an assumption-free summary, the smooth density plot is shaped by assumptions and choices you make as a data analyst.
* The y-axis is scaled so that the area under the density curve sums to 1. This means that interpreting values on the y-axis is not straightforward. To determine the proportion of data in between two values, compute the area under the smooth density curve in the region between those values.
* An advantage of smooth densities over histograms is that densities are easier to compare visually.

### A further note on histograms

Note that the choice of binwidth has a determinative effect on shape. There is no "correct" choice for binwidth, and you can sometimes gain insights into the data by experimenting with binwidths.

## Normal Distribution

### Key points

* The normal distribution:
  + Is centered around one value, the mean
  + Is symmetric around the mean
  + Is defined completely by its mean (μ) and standard deviation ( σ )
  + Always has the same proportion of observations within a given distance of the mean (for example, 95% within 2 σ)
* The standard deviation is the average distance between a value and the mean value.
* Calculate the mean using the mean() function.
* Calculate the standard deviation using the sd() function or manually.
* Standard units describe how many standard deviations a value is away from the mean. The z-score, or number of standard deviations an observation x is away from the mean μ:

Z=x−μσ

* Compute standard units with the scale() function.
* **Important**: to calculate the proportion of values that meet a certain condition, use the mean() function on a logical vector. Because TRUE is converted to 1 and FALSE is converted to 0, taking the mean of this vector yields the proportion of TRUE.

### Equation for the normal distribution

The normal distribution is mathematically defined by the following formula for any mean μ and standard deviation σ:

Pr(a<x<b)=∫ba12π√σe−12(x−μσ)2dx

### Code

# define x as vector of male heights

library(tidyverse)

library(dslabs)

data(heights)

index <- heights$sex=="Male"

x <- heights$height[index]

# calculate the mean and standard deviation manually

average <- sum(x)/length(x)

SD <- sqrt(sum((x - average)^2)/length(x))

# built-in mean and sd functions - note that the audio and printed values disagree

average <- mean(x)

SD <- sd(x)

c(average = average, SD = SD)

# calculate standard units

z <- scale(x)

# calculate proportion of values within 2 SD of mean

mean(abs(z) < 2)

### Note about the sd function

The built-in R function sd() calculates the standard deviation, but it divides by length(x)-1 instead of length(x). When the length of the list is large, this difference is negligible and you can use the built-in sd() function. Otherwise, you should compute σ by hand. For this course series, assume that you should use the sd() function unless you are told not to do so.

## Normal Distribution: Standard Units and Z-scores

### Standard units

For data that are approximately normal, standard units describe the number of standard deviations an observation is from the mean. Standard units are denoted by the variable z and are also known as z-scores.

For any value x from a normal distribution with mean μ and standard deviation σ, the value in standard units is:

z=x−μσ

Standard units are useful for many reasons. Note that the formula for the normal distribution is simplified by substituting z in the exponent:

Pr(a<x<b)=∫ba12π√σe−12z2dx

When z=0, the normal distribution is at a maximum, the mean μ. The function is defined to be symmetric around z=0.

The normal distribution of z-scores is called the standard normal distribution and is defined by μ=0 and σ=1.

Z-scores are useful to quickly evaluate whether an observation is average or extreme. Z-scores near 0 are average. Z-scores above 2 or below -2 are significantly above or below the mean, and z-scores above 3 or below -3 are extremely rare.

We will learn more about benchmark z-score values and their corresponding probabilities below.

### Code: Converting to standard units

The scale function converts a vector of approximately normally distributed values into z-scores.

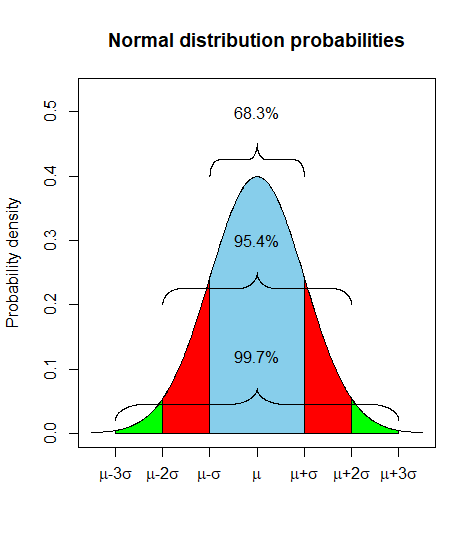
z <- scale(x)

You can compute the proportion of observations that are within 2 standard deviations of the mean like this:

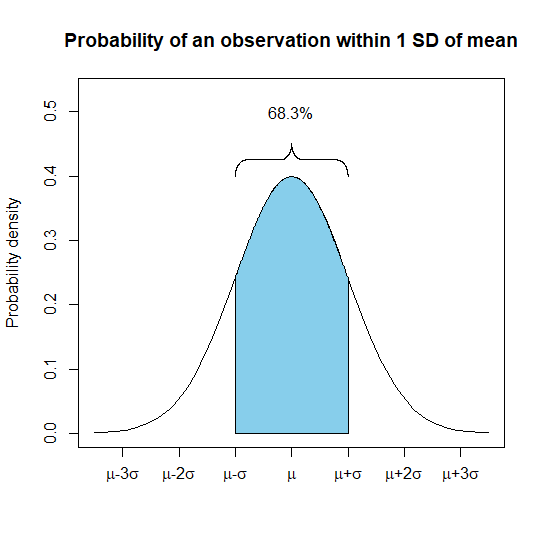
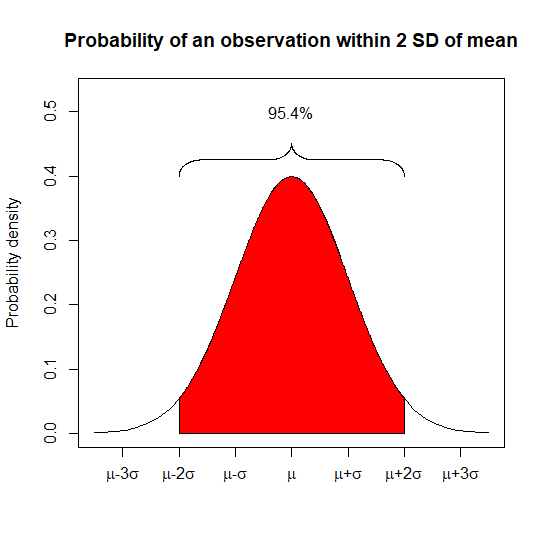
mean(abs(z) < 2)

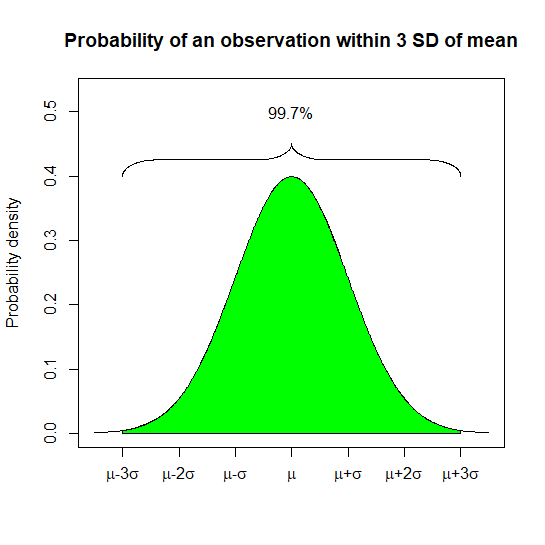
### The 68-95-99.7 Rule

The normal distribution is associated with the 68-95-99.7 rule. This rule describes the probability of observing events within a certain number of standard deviations of the mean.



The probability distribution function for the normal distribution is defined such that:

* About 68% of observations will be within one standard deviation of the mean (μ±σ). In standard units, this is equivalent to a z-score of ∣z∣≤1.
* About 95% of observations will be within two standard deviations of the mean (μ±2σ). In standard units, this is equivalent to a z-score of ∣z∣≤2.
* About 99.7% of observations will be within three standard deviations of the mean (μ±3σ). In standard units, this is equivalent to a z-score of ∣z∣≤3.



* We will learn how to compute these exact probabilities in a later section, as well as probabilities for other intervals.

## The Normal CDF and pnorm

### Key points

* The normal distribution has a mathematically defined CDF which can be computed in R with the function pnorm().
* pnorm(a, avg, s) gives the value of the cumulative distribution function F(a) for the normal distribution defined by average avg and standard deviation s.
* We say that a random quantity is normally distributed with average avg and standard deviation s if the approximation pnorm(a, avg, s) holds for all values of a.
* If we are willing to use the normal approximation for height, we can estimate the distribution simply from the mean and standard deviation of our values.
* If we treat the height data as discrete rather than categorical, we see that the data are not very useful because integer values are more common than expected due to rounding. This is called discretization.
* With rounded data, the normal approximation is particularly useful when computing probabilities of intervals of length 1 that include exactly one integer.

### Code: Using pnorm to calculate probabilities

Given male heights x:

library(tidyverse)

library(dslabs)

data(heights)

x <- heights %>% filter(sex=="Male") %>% pull(height)

We can estimate the probability that a male is taller than 70.5 inches with:

1 - pnorm(70.5, mean(x), sd(x))

### Code: Discretization and the normal approximation

# plot distribution of exact heights in data

plot(prop.table(table(x)), xlab = "a = Height in inches", ylab = "Pr(x = a)")

# probabilities in actual data over length 1 ranges containing an integer

mean(x <= 68.5) - mean(x <= 67.5)

mean(x <= 69.5) - mean(x <= 68.5)

mean(x <= 70.5) - mean(x <= 69.5)

# probabilities in normal approximation match well

pnorm(68.5, mean(x), sd(x)) - pnorm(67.5, mean(x), sd(x))

pnorm(69.5, mean(x), sd(x)) - pnorm(68.5, mean(x), sd(x))

pnorm(70.5, mean(x), sd(x)) - pnorm(69.5, mean(x), sd(x))

# probabilities in actual data over other ranges don't match normal approx as well

mean(x <= 70.9) - mean(x <= 70.1)

pnorm(70.9, mean(x), sd(x)) - pnorm(70.1, mean(x), sd(x))

## Definition of quantiles

### Definition of quantiles

Quantiles are cutoff points that divide a dataset into intervals with set probabilities. The qth quantile is the value at which q% of the observations are equal to or less than that value.

### Using the quantile function

Given a dataset data and desired quantile q, you can find the qth quantile of data with:

quantile(data,q)

### Percentiles

Percentiles are the quantiles that divide a dataset into 100 intervals each with 1% probability. You can determine all percentiles of a dataset data like this:

p <- seq(0.01, 0.99, 0.01)

quantile(data, p)

### Quartiles

Quartiles divide a dataset into 4 parts each with 25% probability. They are equal to the 25th, 50th and 75th percentiles. The 25th percentile is also known as the 1st quartile, the 50th percentile is also known as the median, and the 75th percentile is also known as the 3rd quartile.

The summary() function returns the minimum, quartiles and maximum of a vector.

### Examples

Load the heights dataset from the **dslabs** package:

library(dslabs)  
data(heights)

Use summary() on the heights$height variable to find the quartiles:

summary(heights$height)

Find the percentiles of heights$height:

p <- seq(0.01, 0.99, 0.01)

percentiles <- quantile(heights$height, p)

Confirm that the 25th and 75th percentiles match the 1st and 3rd quartiles. Note that quantile() returns a named vector. You can access the 25th and 75th percentiles like this (adapt the code for other percentile values):

percentiles[names(percentiles) == "25%"]

percentiles[names(percentiles) == "75%"]

## Finding quantiles with qnorm

### Definition of qnorm

The qnorm() function gives the theoretical value of a quantile with probability p of observing a value equal to or less than that quantile value given a normal distribution with mean mu and standard deviation sigma:

qnorm(p, mu, sigma)

By default, mu=0 and sigma=1. Therefore, calling qnorm() with no arguments gives quantiles for the standard normal distribution.

qnorm(p)

Recall that quantiles are defined such that p is the probability of a random observation less than or equal to the quantile.

### Relation to pnorm

The pnorm() function gives the probability that a value from a standard normal distribution will be less than or equal to a z-score value z. Consider:

pnorm(-1.96) ≈0.025

The result of pnorm() is the quantile. Note that:

qnorm(0.025) ≈−1.96

qnorm() and pnorm() are inverse functions:

pnorm(qnorm(0.025)) =0.025

### Theoretical quantiles

You can use qnorm() to determine the theoretical quantiles of a dataset: that is, the theoretical value of quantiles assuming that a dataset follows a normal distribution. Run the qnorm() function with the desired probabilities p, mean mu and standard deviation sigma.

Suppose male heights follow a normal distribution with a mean of 69 inches and standard deviation of 3 inches. The theoretical quantiles are:

p <- seq(0.01, 0.99, 0.01)

theoretical\_quantiles <- qnorm(p, 69, 3)

Theoretical quantiles can be compared to sample quantiles determined with the quantile function in order to evaluate whether the sample follows a normal distribution.

## Quantile-Quantile Plots

### Key points

* Quantile-quantile plots, or QQ-plots, are used to check whether distributions are well-approximated by a normal distribution.
* Given a proportion p, the quantile q is the value such that the proportion of values in the data below q is p.
* In a QQ-plot, the sample quantiles in the observed data are compared to the theoretical quantiles expected from the normal distribution. If the data are well-approximated by the normal distribution, then the points on the QQ-plot will fall near the identity line (sample = theoretical).
* Calculate sample quantiles (observed quantiles) using the quantile() function.
* Calculate theoretical quantiles with the qnorm() function. qnorm() will calculate quantiles for the standard normal distribution (μ=0,σ=1) by default, but it can calculate quantiles for any normal distribution given mean() and sd() arguments. We will learn more about qnorm() in the probability course.
* Note that we will learn alternate ways to make QQ-plots with less code later in the series.

### Code

# define x and z

library(tidyverse)

library(dslabs)

data(heights)

index <- heights$sex=="Male"

x <- heights$height[index]

z <- scale(x)

# proportion of data below 69.5

mean(x <= 69.5)

# calculate observed and theoretical quantiles

p <- seq(0.05, 0.95, 0.05)

observed\_quantiles <- quantile(x, p)

theoretical\_quantiles <- qnorm(p, mean = mean(x), sd = sd(x))

# make QQ-plot

plot(theoretical\_quantiles, observed\_quantiles)

abline(0,1)

# make QQ-plot with scaled values

observed\_quantiles <- quantile(z, p)

theoretical\_quantiles <- qnorm(p)

plot(theoretical\_quantiles, observed\_quantiles)

abline(0,1)

## Percentiles

### Key points

* Percentiles are the quantiles obtained when defining p as 0.01,0.02,...,0.99. They summarize the values at which a certain percent of the observations are equal to or less than that value.
* The 50th percentile is also known as the median.
* The quartiles are the 25th, 50th and 75th percentiles.

## Boxplots

### Key points

* When data do not follow a normal distribution and cannot be succinctly summarized by only the mean and standard deviation, an alternative is to report a five-number summary: range (ignoring outliers) and the quartiles (25th, 50th, 75th percentile).
* In a boxplot, the box is defined by the 25th and 75th percentiles and the median is a horizontal line through the box. The whiskers show the range excluding outliers, and outliers are plotted separately as individual points.
* The interquartile range is the distance between the 25th and 75th percentiles.
* Boxplots are particularly useful when comparing multiple distributions.

## Section 2: Introduction to ggplot2

## ggplot

### Key points and notes

* Throughout the series, we will create plots with the **ggplot2** package. **ggplot2** is part of the **tidyverse** suite of packages, which you can load with library(tidyverse).
* Note that you can also load **ggplot2** alone using the command library(ggplot2), instead of loading the entire **tidyverse**.
* **ggplot2** uses a grammar of graphics to break plots into building blocks that have intuitive syntax, making it easy to create relatively complex and aesthetically pleasing plots with relatively simple and readable code.
* **ggplot2** is designed to work exclusively with tidy data (rows are observations and columns are variables).

## Graph Components

### Key points

* Plots in **ggplot2** consist of 3 main components:
  + - Data: The dataset being summarized
    - Geometry: The type of plot (scatterplot, boxplot, barplot, histogram, qqplot, smooth density, etc.)
    - Aesthetic mapping: Variables mapped to visual cues, such as x-axis and y-axis values and color
* There are additional components:
  + - Scale
    - Labels, Title, Legend
    - Theme/Style

### Code

library(dslabs)

data(murders)

## Creating a New Plot

### Key points

* You can associate a dataset x with a ggplot object with any of the 3 commands:
  + ggplot(data = x)
  + ggplot(x)
  + x %>% ggplot()
* You can assign a ggplot object to a variable. If the object is not assigned to a variable, it will automatically be displayed.
* You can display a ggplot object assigned to a variable by printing that variable.

### Code

library(tidyverse)

library(dslabs)

data(murders)

ggplot(data = murders)

murders %>% ggplot()

p <- ggplot(data = murders)

class(p)

print(p) # this is equivalent to simply typing p

p

### Layers

### Key points

* In **ggplot2**, graphs are created by adding layers to the ggplot object:  
  DATA %>% ggplot() + LAYER\_1 + LAYER\_2 + ... + LAYER\_N
* The geometry layer defines the plot type and takes the format geom\_X where X is the plot type.
* Aesthetic mappings describe how properties of the data connect with features of the graph (axis position, color, size, etc.) Define aesthetic mappings with the aes() function.
* aes() uses variable names from the object component (for example, total rather than murders$total).
* geom\_point() creates a scatterplot and requires x and y aesthetic mappings.
* geom\_text() and geom\_label() add text to a scatterplot and require x, y, and label aesthetic mappings.
* To determine which aesthetic mappings are required for a geometry, read the help file for that geometry.
* You can add layers with different aesthetic mappings to the same graph.

### Code: Adding layers to a plot

library(tidyverse)

library(dslabs)

data(murders)

murders %>% ggplot() +

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

# add points layer to predefined ggplot object

p <- ggplot(data = murders)

p + geom\_point(aes(population/10^6, total))

# add text layer to scatterplot

p + geom\_point(aes(population/10^6, total)) +

geom\_text(aes(population/10^6, total, label = abb))

### Code: Example of aes behavior

# no error from this call

p\_test <- p + geom\_text(aes(population/10^6, total, label = abb))

# error - "abb" is not a globally defined variable and cannot be found outside of aes

p\_test <- p + geom\_text(aes(population/10^6, total), label = abb)

## Tinkering

### Key points

* You can modify arguments to geometry functions other than aes() and the data. Additional arguments can be found in the documentation for each geometry.
* These arguments are not aesthetic mappings: they affect all data points the same way.
* Global aesthetic mappings apply to all geometries and can be defined when you initially call ggplot(). All the geometries added as layers will default to this mapping. Local aesthetic mappings add additional information or override the default mappings.

### Code

# change the size of the points

p + geom\_point(aes(population/10^6, total), size = 3) +

geom\_text(aes(population/10^6, total, label = abb))

# move text labels slightly to the right

p + geom\_point(aes(population/10^6, total), size = 3) +

geom\_text(aes(population/10^6, total, label = abb), nudge\_x = 1)

# simplify code by adding global aesthetic

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

p + geom\_point(size = 3) +

geom\_text(nudge\_x = 1.5)

# local aesthetics override global aesthetics

p + geom\_point(size = 3) +

geom\_text(aes(x = 10, y = 800, label = "Hello there!"))

## Scales, Labels, and Colors

### Key points

* Convert the x-axis to log scale with scale\_x\_continuous(trans = "log10") or scale\_x\_log10(). Similar functions exist for the y-axis.
* Add axis titles with xlab() and ylab() functions. Add a plot title with the ggtitle() function.
* Add a color mapping that colors points by a variable by defining the col argument within aes(). To color all points the same way, define col outside of aes().
* Add a line with the geom\_abline() geometry. geom\_abline() takes arguments slope (default = 1) and intercept (default = 0). Change the color with col or color and line type with lty.
* Placing the line layer after the point layer will overlay the line on top of the points. To overlay points on the line, place the line layer before the point layer.
* There are many additional ways to tweak your graph that can be found in the **ggplot2** documentation, cheat sheet, or on the internet. For example, you can change the legend title with scale\_color\_discrete().

### Code: Log-scale the x- and y-axis

# define p

library(tidyverse)

library(dslabs)

data(murders)

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

# log base 10 scale the x-axis and y-axis

p + geom\_point(size = 3) +

geom\_text(nudge\_x = 0.05) +

scale\_x\_continuous(trans = "log10") +

scale\_y\_continuous(trans = "log10")

# efficient log scaling of the axes

p + geom\_point(size = 3) +

geom\_text(nudge\_x = 0.075) +

scale\_x\_log10() +

scale\_y\_log10()

### Code: Add labels and title

p + geom\_point(size = 3) +

geom\_text(nudge\_x = 0.075) +

scale\_x\_log10() +

scale\_y\_log10() +

xlab("Population in millions (log scale)") +

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

ggtitle("US Gun Murders in 2010")

### Code: Change color of the points

# redefine p to be everything except the points layer

p <- murders %>%

ggplot(aes(population/10^6, total, label = abb)) +

geom\_text(nudge\_x = 0.075) +

scale\_x\_log10() +

scale\_y\_log10() +

xlab("Population in millions (log scale)") +

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

ggtitle("US Gun Murders in 2010")

# make all points blue

p + geom\_point(size = 3, color = "blue")

# color points by region

p + geom\_point(aes(col = region), size = 3)

### Code: Add a line with average murder rate

# define average murder rate

r <- murders %>%

summarize(rate = sum(total) / sum(population) \* 10^6) %>%

pull(rate)

# basic line with average murder rate for the country

p + geom\_point(aes(col = region), size = 3) +

geom\_abline(intercept = log10(r)) # slope is default of 1

# change line to dashed and dark grey, line under points

p +

geom\_abline(intercept = log10(r), lty = 2, color = "darkgrey") +

geom\_point(aes(col = region), size = 3)

### Code: Change legend title

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

## Add-on Packages

### Key points

* The style of a ggplot graph can be changed using the theme() function.
* The **ggthemes** package adds additional themes.
* The **ggrepel** package includes a geometry that repels text labels, ensuring they do not overlap with each other: geom\_text\_repel().

### Code: Adding themes

# theme used for graphs in the textbook and course

library(dslabs)

ds\_theme\_set()

# themes from ggthemes

library(ggthemes)

p + theme\_economist() # style of the Economist magazine

p + theme\_fivethirtyeight() # style of the FiveThirtyEight website

### Code: Putting it all together to assemble the plot

# load libraries

library(tidyverse)

library(ggrepel)

library(ggthemes)

library(dslabs)

data(murders)

# define the intercept

r <- murders %>%

summarize(rate = sum(total) / sum(population) \* 10^6) %>%

.$rate

# make the plot, combining all elements

murders %>%

ggplot(aes(population/10^6, total, label = abb)) +

geom\_abline(intercept = log10(r), lty = 2, color = "darkgrey") +

geom\_point(aes(col = region), size = 3) +

geom\_text\_repel() +

scale\_x\_log10() +

scale\_y\_log10() +

xlab("Population in millions (log scale)") +

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

ggtitle("US Gun Murders in 2010") +

scale\_color\_discrete(name = "Region") +

theme\_economist()

## Other Examples

### Key points

* geom\_histogram() creates a histogram. Use the binwidth argument to change the width of bins, the fill argument to change the bar fill color, and the col argument to change bar outline color.
* geom\_density() creates smooth density plots. Change the fill color of the plot with the fill argument.
* geom\_qq() creates a quantile-quantile plot. This geometry requires the sample argument. By default, the data are compared to a standard normal distribution with a mean of 0 and standard deviation of 1. This can be changed with the dparams argument, or the sample data can be scaled.
* Plots can be arranged adjacent to each other using the grid.arrange() function from the **gridExtra** package. First, create the plots and save them to objects (p1, p2, ...). Then pass the plot objects to grid.arrange().

### Code: Histograms in ggplot2

# load heights data

library(tidyverse)

library(dslabs)

data(heights)

# define p

p <- heights %>%

filter(sex == "Male") %>%

ggplot(aes(x = height))

# basic histograms

p + geom\_histogram()

p + geom\_histogram(binwidth = 1)

# histogram with blue fill, black outline, labels and title

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

xlab("Male heights in inches") +

ggtitle("Histogram")

### Code: Smooth density plots in ggplot2

p + geom\_density()

p + geom\_density(fill = "blue")

### Code: Quantile-quantile plots in ggplot2

# basic QQ-plot

p <- heights %>% filter(sex == "Male") %>%

ggplot(aes(sample = height))

p + geom\_qq()

# QQ-plot against a normal distribution with same mean/sd as data

params <- heights %>%

filter(sex == "Male") %>%

summarize(mean = mean(height), sd = sd(height))

p + geom\_qq(dparams = params) +

geom\_abline()

# QQ-plot of scaled data against the standard normal distribution

heights %>%

ggplot(aes(sample = scale(height))) +

geom\_qq() +

geom\_abline()

### Code: Grids of plots with the gridExtra package

# define plots p1, p2, p3

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")

# arrange plots next to each other in 1 row, 3 columns

library(gridExtra)

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

## Section 3: Summarization with dpylr

## dplyr

### Key points

* summarize() from the **dplyr**/**tidyverse** package computes summary statistics from the data frame. It returns a data frame whose column names are defined within the function call.
* summarize() can compute any summary function that operates on vectors and returns a single value, but it cannot operate on functions that return multiple values.
* Like most **dplyr** functions, summarize() is aware of variable names within data frames and can use them directly.

### Code

library(tidyverse)

library(dslabs)

data(heights)

# compute average and standard deviation for males

s <- heights %>%

filter(sex == "Male") %>%

summarize(average = mean(height), standard\_deviation = sd(height))

# access average and standard deviation from summary table

s$average

s$standard\_deviation

# compute median, min and max

heights %>%

filter(sex == "Male") %>%

summarize(median = median(height),

minimum = min(height),

maximum = max(height))

# alternative way to get min, median, max in base R

quantile(heights$height, c(0, 0.5, 1))

# generates an error: summarize can only take functions that return a single value

heights %>%

filter(sex == "Male") %>%

summarize(range = quantile(height, c(0, 0.5, 1)))

## The Dot Placeholder

### Key points

* The dot operator allows you to access values stored in data that is being piped in using the %>% character. The dot is a placeholder for the data being passed in through the pipe.
* The dot operator allows **dplyr** functions to return single vectors or numbers instead of only data frames.
* us\_murder\_rate %>% .$rate is equivalent to us\_murder\_rate$rate.
* Note that an equivalent way to extract a single column using the pipe is us\_murder\_rate %>% pull(rate). The pull() function will be used in later course material.

### Code

library(tidyverse)

library(dslabs)

data(murders)

murders <- murders %>% mutate(murder\_rate = total/population\*100000)

summarize(murders, mean(murder\_rate))

# calculate US murder rate, generating a data frame

us\_murder\_rate <- murders %>%

summarize(rate = sum(total) / sum(population) \* 100000)

us\_murder\_rate

# extract the numeric US murder rate with the dot operator

us\_murder\_rate %>% .$rate

# calculate and extract the murder rate with one pipe

us\_murder\_rate <- murders %>%

summarize(rate = sum(total) / sum(population) \* 100000) %>%

.$rate

## Group By

### Key points

* The group\_by() function from **dplyr** converts a data frame to a grouped data frame, creating groups using one or more variables.
* summarize() and some other **dplyr** functions will behave differently on grouped data frames.
* Using summarize() on a grouped data frame computes the summary statistics for each of the separate groups.

### Code

# libraries and data

library(tidyverse)

library(dslabs)

data(heights)

data(murders)

# compute separate average and standard deviation for male/female heights

heights %>%

group\_by(sex) %>%

summarize(average = mean(height), standard\_deviation = sd(height))

# compute median murder rate in 4 regions of country

murders <- murders %>%

mutate(murder\_rate = total/population \* 100000)

murders %>%

group\_by(region) %>%

summarize(median\_rate = median(murder\_rate))

## Sorting Data Tables

### Key points

* The arrange() function from **dplyr** sorts a data frame by a given column.
* By default, arrange() sorts in ascending order (lowest to highest). To instead sort in descending order, use the function desc() inside of arrange().
* You can arrange() by multiple levels: within equivalent values of the first level, observations are sorted by the second level, and so on.
* The top\_n() function shows the top results ranked by a given variable, but the results are not ordered. You can combine top\_n() with arrange() to return the top results in order.

### Code

# libraries and data

library(tidyverse)

library(dslabs)

data(murders)

# set up murders object

murders <- murders %>%

mutate(murder\_rate = total/population \* 100000)

# arrange by population column, smallest to largest

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

# arrange by murder rate, smallest to largest

murders %>% arrange(murder\_rate) %>% head()

# arrange by murder rate in descending order

murders %>% arrange(desc(murder\_rate)) %>% head()

# arrange by region alphabetically, then by murder rate within each region

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

# show the top 10 states with highest murder rate, not ordered by rate

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

# show the top 10 states with highest murder rate, ordered by rate

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

## Section 4: Gapminder

## Case Study: Trends in World Health and Economics

### More about Gapminder

The original Gapminder TED talks are available and we encourage you to watch them.

* [The Best Stats You've Ever Seen](https://www.ted.com/talks/hans_rosling_shows_the_best_stats_you_ve_ever_seen?language=en) : <https://www.ted.com/talks/hans_rosling_the_best_stats_you_ve_ever_seen?language=en>
* [New Insights on Poverty](https://www.ted.com/talks/hans_rosling_reveals_new_insights_on_poverty?language=en) : <https://www.ted.com/talks/hans_rosling_new_insights_on_poverty?language=en>

You can also find more information and raw data (in addition to what we analyze in class) at <https://www.gapminder.org/>.

### Key points

* Data visualization can be used to dispel common myths and educate the public and contradict sensationalist or outdated claims and stories.
* We will use real data to answer the following questions about world health and economics:
  + - Is it still fair to consider the world as divided into the West and the developing world?
    - Has income inequality across countries worsened over the last 40 years?

## Gapminder Dataset

### Key points

* A selection of world health and economics statistics from the Gapminder project can be found in the **dslabs** package as data(gapminder).
* Most people have misconceptions about world health and economics, which can be addressed by considering real data.

### Code

# load and inspect gapminder data

library(dslabs)

data(gapminder)

head(gapminder)

# compare infant mortality in Sri Lanka and Turkey

gapminder %>%

filter(year == 2015 & country %in% c("Sri Lanka", "Turkey")) %>%

select(country, infant\_mortality)

## Life Expectancy and Fertility Rates

### Key points

* A prevalent worldview is that the world is divided into two groups of countries:
  + - Western world: high life expectancy, low fertility rate
    - Developing world: lower life expectancy, higher fertility rate
* Gapminder data can be used to evaluate the validity of this view.
* A scatterplot of life expectancy versus fertility rate in 1962 suggests that this viewpoint was grounded in reality 50 years ago. Is it still the case today?

### Code

# basic scatterplot of life expectancy versus fertility

ds\_theme\_set() # set plot theme

filter(gapminder, year == 1962) %>%

ggplot(aes(fertility, life\_expectancy)) +

geom\_point()

# add color as continent

filter(gapminder, year == 1962) %>%

ggplot(aes(fertility, life\_expectancy, color = continent)) +

geom\_point()

### Faceting

### Key points

* Faceting makes multiple side-by-side plots stratified by some variable. This is a way to ease comparisons.
* The facet\_grid() function allows faceting by up to two variables, with rows faceted by one variable and columns faceted by the other variable. To facet by only one variable, use the dot operator as the other variable.
* The facet\_wrap() function facets by one variable and automatically wraps the series of plots so they have readable dimensions.
* Faceting keeps the axes fixed across all plots, easing comparisons between plots.
* The data suggest that the developing versus Western world view no longer makes sense in 2012.

### Code

# facet by continent and year

filter(gapminder, year %in% c(1962, 2012)) %>%

ggplot(aes(fertility, life\_expectancy, col = continent)) +

geom\_point() +

facet\_grid(continent ~ year)

# facet by year only

filter(gapminder, year %in% c(1962, 2012)) %>%

ggplot(aes(fertility, life\_expectancy, col = continent)) +

geom\_point() +

facet\_grid(. ~ year)

# facet by year, plots wrapped onto multiple rows

years <- c(1962, 1980, 1990, 2000, 2012)

continents <- c("Europe", "Asia")

gapminder %>%

filter(year %in% years & continent %in% continents) %>%

ggplot(aes(fertility, life\_expectancy, col = continent)) +

geom\_point() +

facet\_wrap(~year)

## Time Series Plots

### Key points

* Time series plots have time on the x-axis and a variable of interest on the y-axis.
* The geom\_line() geometry connects adjacent data points to form a continuous line. A line plot is appropriate when points are regularly spaced, densely packed and from a single data series.
* You can plot multiple lines on the same graph. Remember to group or color by a variable so that the lines are plotted independently.
* Labeling is usually preferred over legends. However, legends are easier to make and appear by default. Add a label with geom\_text(), specifying the coordinates where the label should appear on the graph.

### Code: Single time series

# scatterplot of US fertility by year

gapminder %>%

filter(country == "United States") %>%

ggplot(aes(year, fertility)) +

geom\_point()

# line plot of US fertility by year

gapminder %>%

filter(country == "United States") %>%

ggplot(aes(year, fertility)) +

geom\_line()

### Code: Multiple time series

# line plot fertility time series for two countries- only one line (incorrect)

countries <- c("South Korea", "Germany")

gapminder %>% filter(country %in% countries) %>%

ggplot(aes(year, fertility)) +

geom\_line()

# line plot fertility time series for two countries - one line per country

gapminder %>% filter(country %in% countries) %>%

ggplot(aes(year, fertility, group = country)) +

geom\_line()

# fertility time series for two countries - lines colored by country

gapminder %>% filter(country %in% countries) %>%

ggplot(aes(year, fertility, col = country)) +

geom\_line()

### Code: Adding text labels to a plot

# life expectancy time series - lines colored by country and labeled, no legend

labels <- data.frame(country = countries, x = c(1975, 1965), y = c(60, 72))

gapminder %>% filter(country %in% countries) %>%

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

geom\_line() +

geom\_text(data = labels, aes(x, y, label = country), size = 5) +

theme(legend.position = "none")

## Transformations

### Key points

* We use GDP data to compute income in US dollars per day, adjusted for inflation.
* Log transformations convert multiplicative changes into additive changes.
* Common transformations are the log base 2 transformation and the log base 10 transformation. The choice of base depends on the range of the data. The natural log is not recommended for visualization because it is difficult to interpret.
* The mode of a distribution is the value with the highest frequency. The mode of a normal distribution is the average. A distribution can have multiple local modes.
* There are two ways to use log transformations in plots: transform the data before plotting or transform the axes of the plot. Log scales have the advantage of showing the original values as axis labels, while log transformed values ease interpretation of intermediate values between labels.
* Scale the x-axis using scale\_x\_continuous() or scale\_x\_log10() layers in **ggplot2**. Similar functions exist for the y-axis.
* In 1970, income distribution is bimodal, consistent with the dichotomous Western versus developing worldview.

### Code

# add dollars per day variable

gapminder <- gapminder %>%

mutate(dollars\_per\_day = gdp/population/365)

# histogram of dollars per day

past\_year <- 1970

gapminder %>%

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

ggplot(aes(dollars\_per\_day)) +

geom\_histogram(binwidth = 1, color = "black")

# repeat histogram with log2 scaled data

gapminder %>%

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

ggplot(aes(log2(dollars\_per\_day))) +

geom\_histogram(binwidth = 1, color = "black")

# repeat histogram with log2 scaled x-axis

gapminder %>%

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

ggplot(aes(dollars\_per\_day)) +

geom\_histogram(binwidth = 1, color = "black") +

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

## Stratify and Boxplot

### Key points

* Make boxplots stratified by a categorical variable using the geom\_boxplot() geometry.
* Rotate axis labels by changing the theme through element\_text(). You can change the angle and justification of the text labels.
* Consider ordering your factors by a meaningful value with the reorder() function, which changes the order of factor levels based on a related numeric vector. This is a way to ease comparisons.
* Show the data by adding data points to the boxplot with a geom\_point() layer. This adds information beyond the five-number summary to your plot, but too many data points it can obfuscate your message.

### Code: Boxplot of GDP by region

# add dollars per day variable

gapminder <- gapminder %>%

mutate(dollars\_per\_day = gdp/population/365)

# number of regions

length(levels(gapminder$region))

# boxplot of GDP by region in 1970

past\_year <- 1970

p <- gapminder %>%

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

ggplot(aes(region, dollars\_per\_day))

p + geom\_boxplot()

# rotate names on x-axis

p + geom\_boxplot() +

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

### Code: The reorder function

# by default, factor order is alphabetical

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

levels(fac)

# reorder factor by the category means

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

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

levels(fac)

### Code: Enhanced boxplot ordered by median income, scaled, and showing data

# reorder by median income and color by continent

p <- gapminder %>%

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

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

ggplot(aes(region, dollars\_per\_day, fill = continent)) + # color by continent

geom\_boxplot() +

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

xlab("")

p

# log2 scale y-axis

p + scale\_y\_continuous(trans = "log2")

# add data points

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

## Comparing Distributions

### Key points

* Use intersect() to find the overlap between two vectors.
* To make boxplots where grouped variables are adjacaent, color the boxplot by a factor instead of faceting by that factor. This is a way to ease comparisons.
* The data suggest that the income gap between rich and poor countries has narrowed, not expanded.

### Code: Histogram of income in West versus developing world, 1970 and 2010

# add dollars per day variable and define past year

gapminder <- gapminder %>%

mutate(dollars\_per\_day = gdp/population/365)

past\_year <- 1970

# define Western countries

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

# facet by West vs devloping  
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)

# facet by West/developing and year

present\_year <- 2010

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)

### Code: Income distribution of West versus developing world, only countries with data

# define countries that have data available in both years

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)

# make histogram including only countries with data available in both years

gapminder %>%

filter(year %in% c(past\_year, present\_year) & country %in% country\_list) %>% # keep only selected countries

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)

### Code: Boxplots of income in West versus developing world, 1970 and 2010

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 ~ .)

# arrange matching boxplots next to each other, colored by year

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

## Density Plots

### Key points

* Change the y-axis of density plots to variable counts using ..count.. as the y argument.
* The case\_when() function defines a factor whose levels are defined by a variety of logical operations to group data.
* Plot stacked density plots using position="stack".
* Define a weight aesthetic mapping to change the relative weights of density plots - for example, this allows weighting of plots by population rather than number of countries.

### Code: Faceted smooth density plots

# see the code below the previous video for variable definitions

# smooth density plots - area under each curve adds to 1

gapminder %>%  
 filter(year == past\_year & country %in% country\_list) %>%  
 mutate(group = ifelse(region %in% west, "West", "Developing")) %>% group\_by(group) %>%  
 summarize(n = n()) %>% knitr::kable()

# smooth density plots - variable counts on y-axis

p <- gapminder %>%  
 filter(year == past\_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, bw = 0.75) + facet\_grid(year ~ .)

### Code: Add new region groups with case\_when

# add group as a factor, grouping regions

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"))

# reorder factor levels

gapminder <- gapminder %>%  
 mutate(group = factor(group, levels = c("Others", "Latin America", "East Asia", "Sub-Saharan Africa", "West")))

### Code: Stacked density plot

# note you must redefine p with the new gapminder object first

p <- gapminder %>%  
 filter(year %in% c(past\_year, present\_year) & country %in% country\_list) %>%  
 ggplot(aes(dollars\_per\_day, fill = group)) +  
 scale\_x\_continuous(trans = "log2")

# stacked density plot

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

### Code: Weighted stacked density plot

# weighted stacked density plot

gapminder %>%  
 filter(year %in% c(past\_year, present\_year) & country %in% country\_list) %>%  
 group\_by(year) %>%  
 mutate(weight = population/sum(population\*2)) %>%  
 ungroup() %>%  
 ggplot(aes(dollars\_per\_day, fill = group, weight = weight)) +  
 scale\_x\_continuous(trans = "log2") +  
 geom\_density(alpha = 0.2, bw = 0.75, position = "stack") + facet\_grid(year ~ .)

## Ecological Fallacy

### Key points

* The breaks argument allows us to set the location of the axis labels and tick marks.
* The logistic or logit transformation is defined as f(p)=logp1−p, or the log of odds. This scale is useful for highlighting differences near 0 or near 1 and converts fold changes into constant increases.
* The ecological fallacy is assuming that conclusions made from the average of a group apply to all members of that group.

### Code

# define gapminder

library(tidyverse)

library(dslabs)

data(gapminder)

# add additional cases

gapminder <- gapminder %>%  
 mutate(group = case\_when(  
 .$region %in% west ~ "The West",  
 .$region %in% "Northern Africa" ~ "Northern Africa",  
 .$region %in% c("Eastern Asia", "South-Eastern Asia") ~ "East Asia",  
 .$region == "Southern Asia" ~ "Southern Asia",  
 .$region %in% c("Central America", "South America", "Caribbean") ~ "Latin America",  
 .$continent == "Africa" & .$region != "Northern Africa" ~ "Sub-Saharan Africa",  
 .$region %in% c("Melanesia", "Micronesia", "Polynesia") ~ "Pacific Islands"))

# define a data frame with group average income and average infant survival rate

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\_survival\_rate = 1 - sum(infant\_mortality/1000\*population)/sum(population))  
surv\_income %>% arrange(income)

# plot infant survival versus income, with transformed axes

surv\_income %>% ggplot(aes(income, infant\_survival\_rate, label = group, color = group)) +  
 scale\_x\_continuous(trans = "log2", limit = c(0.25, 150)) +  
 scale\_y\_continuous(trans = "logit", limit = c(0.875, .9981),  
 breaks = c(.85, .90, .95, .99, .995, .998)) +  
 geom\_label(size = 3, show.legend = FALSE)

## Section 5: Data Visualization Principles

## Introduction to Data Visualization Principles

### Key points

* We aim to provide some general guidelines for effective data visualization.
* We show examples of plot styles to avoid, discuss how to improve them, and use these examples to explain research-based principles for effective visualization.
* When choosing a visualization approach, keep your goal and audience in mind.

## Encoding Data Using Visual Cues

### Key points

* Visual cues for encoding data include position, length, angle, area, brightness and color hue.
* Position and length are the preferred way to display quantities, followed by angles, which are preferred over area. Brightness and color are even harder to quantify but can sometimes be useful.
* Pie charts represent visual cues as both angles and area, while donut charts use only area. Humans are not good at visually quantifying angles and are even worse at quantifying area. Therefore pie and donut charts should be avoided - use a bar plot instead. If you must make a pie chart, include percentages as labels.
* Bar plots represent visual cues as position and length. Humans are good at visually quantifying linear measures, making bar plots a strong alternative to pie or donut charts

## Know When to Include Zero

### Key points

* When using bar plots, always start at 0. It is deceptive not to start at 0 because bar plots imply length is proportional to the quantity displayed. Cutting off the y-axis can make differences look bigger than they actually are.
* When using position rather than length, it is not necessary to include 0 (scatterplot, dot plot, boxplot).

## Do Not Distort Quantities

### Key points

* Make sure your visualizations encode the correct quantities.
* For example, if you are using a plot that relies on circle area, make sure the area (rather than the radius) is proportional to the quantity.

## Order by a Meaningful Value

### Key points

* It is easiest to visually extract information from a plot when categories are ordered by a meaningful value. The exact value on which to order will depend on your data and the message you wish to convey with your plot.
* The default ordering for categories is alphabetical if the categories are strings or by factor level if factors. However, we rarely want alphabetical order.

## Show the Data

### Key points

* A dynamite plot - a bar graph of group averages with error bars denoting standard errors - provides almost no information about a distribution.
* By showing the data, you provide viewers extra information about distributions.
* Jitter is adding a small random shift to each point in order to minimize the number of overlapping points. To add jitter, use the  geom\_jitter() geometry instead of geom\_point(). (See example below.)
* Alpha blending is making points somewhat transparent, helping visualize the density of overlapping points. Add an alpha argument to the geometry.

### Code

# dot plot showing the data

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

# jittered, alpha blended point plot

heights %>% ggplot(aes(sex, height)) + geom\_jitter(width = 0.1, alpha = 0.2)

## Ease Comparisons: Use Common Axes

### Key points

* Ease comparisons by keeping axes the same when comparing data across multiple plots.
* Align plots vertically to see horizontal changes. Align plots horizontally to see vertical changes.
* Bar plots are useful for showing one number but not useful for showing distributions.

## Consider Transformations

### Key points

* Use transformations when warranted to ease visual interpretation.
* The log transformation is useful for data with multiplicative changes. The logistic transformation is useful for fold changes in odds. The square root transformation is useful for count data.
* We learned how to apply transformations earlier in the course.

## Ease Comparisons: Compared Visual Cues Should Be Adjacent

### Key points

* When two groups are to be compared, it is optimal to place them adjacent in the plot.
* Use color to encode groups to be compared.
* Consider using a color blind friendly palette like the one in this video.

### Code

color\_blind\_friendly\_cols <- c("#999999", "#E69F00", "#56B4E9", "#009E73", "#F0E442", "#0072B2", "#D55E00", "#CC79A7")

p1 <- data.frame(x = 1:8, y = 1:8, col = as.character(1:8)) %>%  
 ggplot(aes(x, y, color = col)) +  
 geom\_point(size = 5)  
p1 + scale\_color\_manual(values = color\_blind\_friendly\_cols)

### Slope Charts

### Key points

* Consider using a slope chart or Bland-Altman plot when comparing one variable at two different time points, especially for a small number of observations.
* Slope charts use angle to encode change. Use geom\_line() to create slope charts. It is useful when comparing a small number of observations.
* The Bland-Altman plot (Tukey mean difference plot, MA plot) graphs the difference between conditions on the y-axis and the mean between conditions on the x-axis. It is more appropriate for large numbers of observations than slope charts.

### Code: Slope chart

library(tidyverse)  
library(dslabs)  
data(gapminder)

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")

### Code: Bland-Altman plot

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")

## Encoding a Third Variable

### Key points

* Encode a categorical third variable on a scatterplot using color hue or shape. Use the shape argument to control shape.
* Encode a continuous third variable on a using color intensity or size.

## Case Study: Vaccines

### Key points

* Vaccines save millions of lives, but misinformation has led some to question the safety of vaccines. The data support vaccines as safe and effective. We visualize data about measles incidence in order to demonstrate the impact of vaccination programs on disease rate.
* The **RColorBrewer** package offers several color palettes. Sequential color palettes are best suited for data that span from high to low. Diverging color palettes are best suited for data that are centered and diverge towards high or low values.
* The geom\_tile() geometry creates a grid of colored tiles.
* Position and length are stronger cues than color for numeric values, but color can be appropriate sometimes.

### Code: Tile plot of measles rate by year and state

# import data and inspect

library(tidyverse)  
library(dslabs)  
data(us\_contagious\_diseases)  
str(us\_contagious\_diseases)

# assign dat to the per 10,000 rate of measles, removing Alaska and Hawaii and adjusting for weeks reporting

the\_disease <- "Measles"  
dat <- us\_contagious\_diseases %>%  
 filter(!state %in% c("Hawaii", "Alaska") & disease == the\_disease) %>%  
 mutate(rate = count / population \* 10000 \* 52/weeks\_reporting) %>%  
 mutate(state = reorder(state, rate))

# plot disease rates per year in California

dat %>% filter(state == "California" & !is.na(rate)) %>%  
 ggplot(aes(year, rate)) +  
 geom\_line() +  
 ylab("Cases per 10,000") +  
 geom\_vline(xintercept=1963, col = "blue")

# tile plot of disease rate by state and year

dat %>% ggplot(aes(year, state, fill=rate)) +  
 geom\_tile(color = "grey50") +  
 scale\_x\_continuous(expand = c(0,0)) +  
 scale\_fill\_gradientn(colors = RColorBrewer::brewer.pal(9, "Reds"), trans = "sqrt") +  
 geom\_vline(xintercept = 1963, col = "blue") +  
 theme\_minimal() + theme(panel.grid = element\_blank()) +  
 ggtitle(the\_disease) +  
 ylab("") +  
 xlab("")

### Code: Line plot of measles rate by year and state

# compute US average measles rate by year

avg <- us\_contagious\_diseases %>%  
 filter(disease == the\_disease) %>% group\_by(year) %>%  
 summarize(us\_rate = sum(count, na.rm = TRUE)/sum(population, na.rm = TRUE)\*10000)

# make line plot of measles rate by year by state

dat %>%  
 filter(!is.na(rate)) %>%  
 ggplot() +  
 geom\_line(aes(year, rate, group = state), color = "grey50",   
 show.legend = FALSE, alpha = 0.2, size = 1) +  
 geom\_line(mapping = aes(year, us\_rate), data = avg, size = 1, col = "black") +  
 scale\_y\_continuous(trans = "sqrt", breaks = c(5, 25, 125, 300)) +  
 ggtitle("Cases per 10,000 by state") +  
 xlab("") +  
 ylab("") +  
 geom\_text(data = data.frame(x = 1955, y = 50),  
 mapping = aes(x, y, label = "US average"), color = "black") +  
 geom\_vline(xintercept = 1963, col = "blue")

## Avoid Pseudo and Gratuitous 3D Plots

### Avoid Too Many Significant Digits

### Key points

* In tables, avoid using too many significant digits. Too many digits can distract from the meaning of your data.
* Reduce the number of significant digits globally by setting an option. For example, options(digits = 3) will cause all future computations that session to have 3 significant digits.
* Reduce the number of digits locally using round() or signif().

## Titanic Survival Exercises

 Bookmark this page

Put all your new skills together to perform exploratory data analysis on a classic machine learning dataset: Titanic survival!

### Background

The Titanic was a British ocean liner that struck an iceberg and sunk on its maiden voyage in 1912 from the United Kingdom to New York. More than 1,500 of the estimated 2,224 passengers and crew died in the accident, making this one of the largest maritime disasters ever outside of war. The ship carried a wide range of passengers of all ages and both genders, from luxury travelers in first-class to immigrants in the lower classes. However, not all passengers were equally likely to survive the accident. We use real data about a selection of 891 passengers to learn who was on the Titanic and which passengers were more likely to survive.

### Libraries, Options, and Data

Be sure that you have installed the **titanic** package before proceeding.

Define the titanic dataset starting from the **titanic** library with the following code:

options(digits = 3) # report 3 significant digits

library(tidyverse)

library(titanic)

titanic <- titanic\_train %>%  
 select(Survived, Pclass, Sex, Age, SibSp, Parch, Fare) %>%  
 mutate(Survived = factor(Survived),  
 Pclass = factor(Pclass),  
 Sex = factor(Sex))

### **Question 1: Variable Types**

3.0/3.0 points (graded)

Inspect the data and also use ?titanic\_train to learn more about the variables in the dataset. Match these variables from the dataset to their variable type. There is at least one variable of each type (ordinal categorical, non-ordinal categorical, continuous, discrete).

Survived                                                 

correct

Pclass                                                 

correct

Sex                                                 

correct

SibSp                                                 

correct

Parch                                                 

correct

Fare                                                 

correct

Submit

You have used 2 of 3 attemptsSome problems have options such as save, reset, hints, or show answer. These options follow the Submit button.

SaveSave Your Answer Show Answer

### **Question 2: Demographics of Titanic Passengers**

3.5/3.5 points (graded)

Make density plots of age grouped by sex. Try experimenting with combinations of faceting, alpha blending, stacking and using variable counts on the y-axis to answer the following questions. Some questions may be easier to answer with different versions of the density plot.

Which of the following are true?

Select all correct answers.

Females and males had the same general shape of age distribution.

The age distribution was bimodal, with one mode around 25 years of age and a second smaller mode around 5 years of age.

There were more females than males.

The count of males of age 40 was higher than the count of females of age 40.

The proportion of males age 18-35 was higher than the proportion of females age 18-35.

The proportion of females under age 17 was higher than the proportion of males under age 17.

The oldest passengers were female.

correct

Submit

You have used 1 of 3 attemptsSome problems have options such as save, reset, hints, or show answer. These options follow the Submit button.

SaveSave Your Answer Show Answer

### **Question 3: QQ-plot of Age Distribution**

1/1 point (graded)

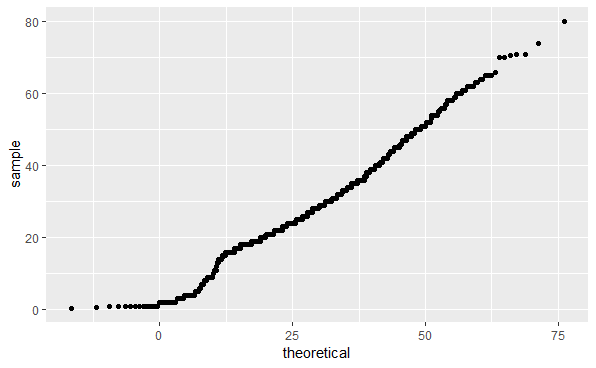
Use geom\_qq() to make a QQ-plot of passenger age and add an identity line with geom\_abline(). Filter out any individuals with an age of NA first. Use the following object as the dparams argument in geom\_qq():

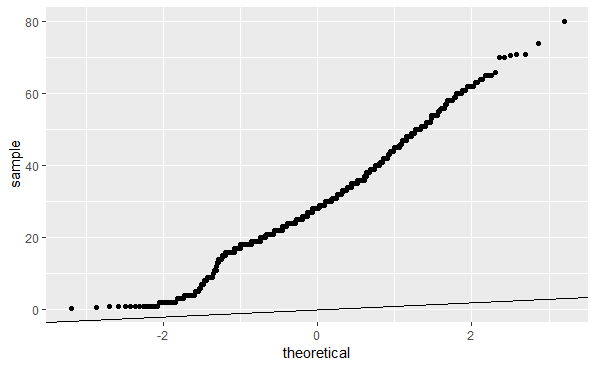
params <- titanic %>%

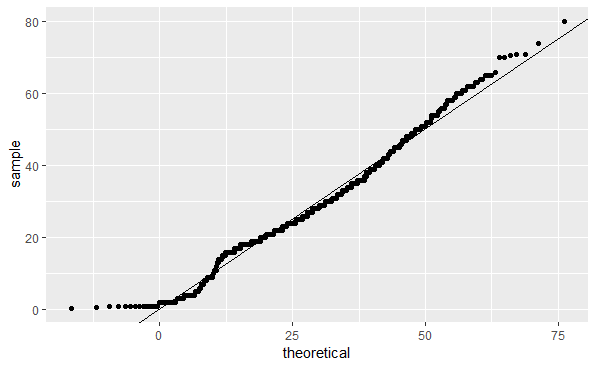
filter(!is.na(Age)) %>%

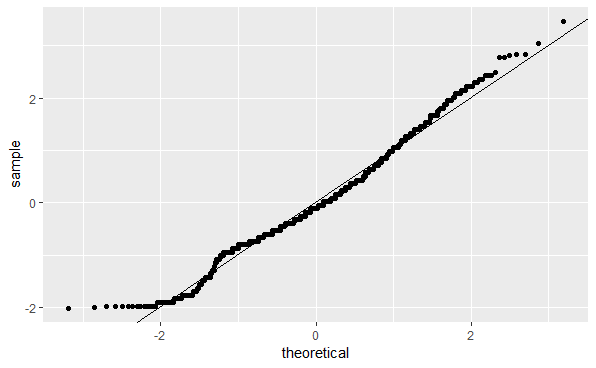
summarize(mean = mean(Age), sd = sd(Age))

Which of the following is the correct plot according to the instructions above?









correct

Submit

You have used 1 of 2 attemptsSome problems have options such as save, reset, hints, or show answer. These options follow the Submit button.

SaveSave Your Answer Show Answer

### **Question 4: Survival by Sex**

2.0/2.0 points (graded)

To answer the following questions, make barplots of the Survived and Sex variables using geom\_bar(). Try plotting one variable and filling by the other variable. You may want to try the default plot, then try adding position = position\_dodge() to geom\_bar() to make separate bars for each group.

You can read more about making barplots in the [textbook section on ggplot2 geometries](https://rafalab.github.io/dsbook/distributions.html#other-geometries).

Which of the following are true?

Select all correct answers.

Less than half of passengers survived.

Most of the survivors were female.

Most of the males survived.

Most of the females survived.

correct

Submit

You have used 2 of 2 attemptsSome problems have options such as save, reset, hints, or show answer. These options follow the Submit button.

Show Answer

### **Question 5: Survival by Age**

3.0/3.0 points (graded)

Make a density plot of age filled by survival status. Change the y-axis to count and set alpha = 0.2.

Which age group is the only group more likely to survive than die?

0-8

10-18

18-30

30-50

50-70

70-80

correct

Which age group had the most deaths?

0-8

10-18

18-30

30-50

50-70

70-80

correct

Which age group had the highest proportion of deaths?

0-8

10-18

18-30

30-50

50-70

70-80

correct

Submit

You have used 3 of 4 attemptsSome problems have options such as save, reset, hints, or show answer. These options follow the Submit button.

SaveSave Your Answer Show Answer

### **Question 6: Survival by Fare**

1.5/2.5 points (graded)

Filter the data to remove individuals who paid a fare of 0. Make a boxplot of fare grouped by survival status. Try a log2 transformation of fares. Add the data points with jitter and alpha blending.

Which of the following are true?

Select all correct answers.

Passengers who survived generally payed higher fares than those who did not survive.

The interquartile range for fares was smaller for passengers who survived.

The median fare was lower for passengers who did not survive.

Only one individual paid a fare around $500. That individual survived.

Most individuals who paid a fare around $8 did not survive.

partially correct

**Answer**

Incorrect:

Try again. The interquartile range (IQR) is the size of the box.

Try again. Make sure you add points with geom\_jitter.

Submit

You have used 2 of 2 attemptsSome problems have options such as save, reset, hints, or show answer. These options follow the Submit button.

Show Answer

### **Question 7: Survival by Passenger Class**

3.0/3.0 points (graded)

The Pclass variable corresponds to the passenger class. Make three barplots. For the first, make a basic barplot of passenger class filled by survival. For the second, make the same barplot but use the argument position = position\_fill() to show relative proportions in each group instead of counts. For the third, make a barplot of survival filled by passenger class using position = position\_fill().

You can read more about making barplots in the [textbook section on ggplot2 geometries](https://rafalab.github.io/dsbook/distributions.html#other-geometries).

Which of the following are true?

Select all correct answers.

There were more third class passengers than passengers in the first two classes combined.

There were the fewest passengers in first class, second-most passengers in second class, and most passengers in third class.

Survival proportion was highest for first class passengers, followed by second class. Third-class had the lowest survival proportion.

Most passengers in first class survived. Most passengers in other classes did not survive.

The majority of survivors were from first class. (Majority means over 50%.)

The majority of those who did not survive were from third class.

correct

Submit

You have used 1 of 2 attemptsSome problems have options such as save, reset, hints, or show answer. These options follow the Submit button.

SaveSave Your Answer Show Answer

### **Question 8: Survival by Age, Sex and Passenger Class**

2.5/2.5 points (graded)

Create a grid of density plots for age, filled by survival status, with count on the y-axis, faceted by sex and passenger class.

Which of the following are true?

Select all correct answers.

The largest group of passengers was third-class males.

The age distribution is the same across passenger classes.

The gender distribution is the same across passenger classes.

Most first-class and second-class females survived.

Almost all second-class males did not survive, with the exception of children.

correct

Submit

You have used 1 of 2 attemptsSome problems have options such as save, reset, hints, or show answer. These options follow the Submit button.