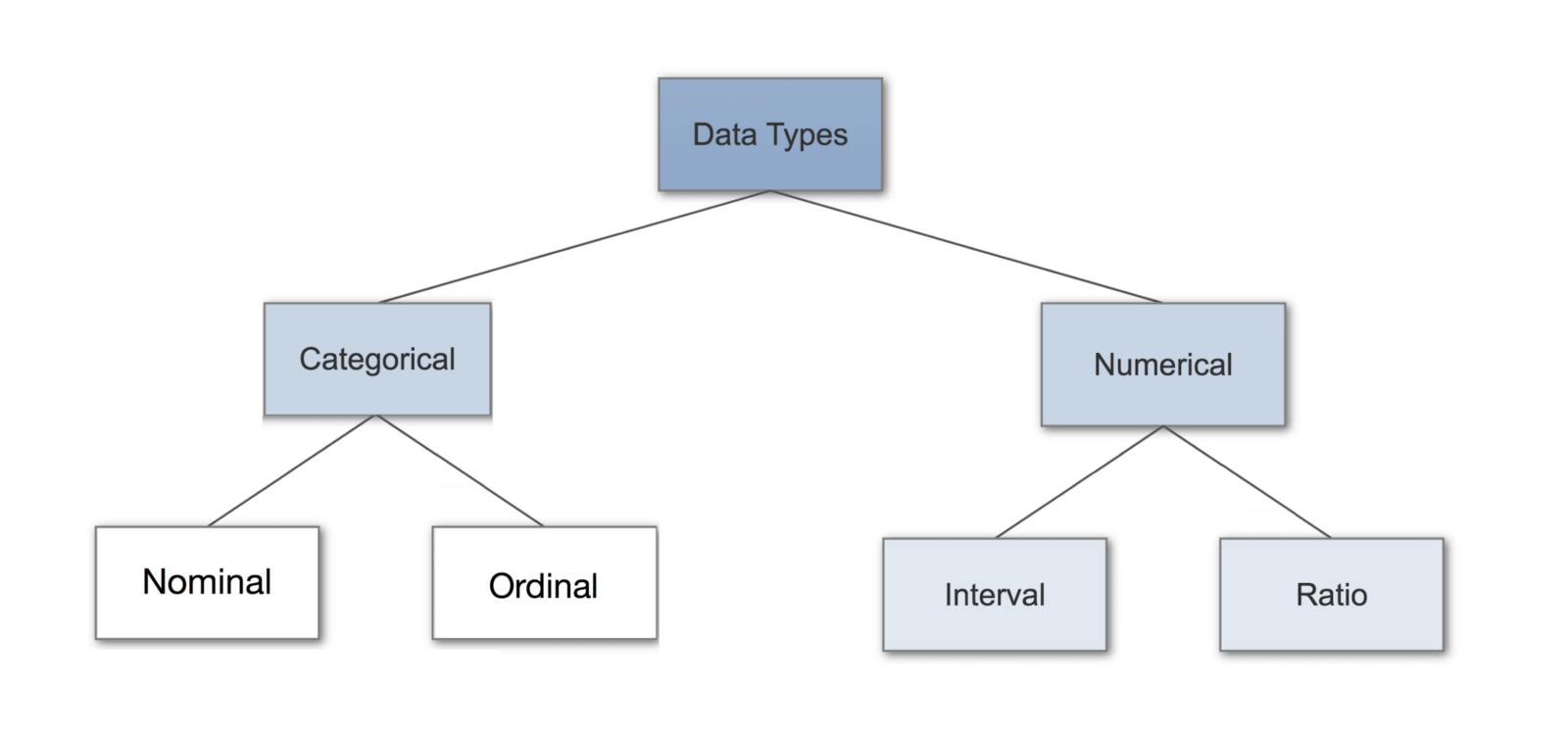
**DATA TYPES**

Data Types are an important concept of statistics, which needs to be understood, to correctly apply statistical measurements to your data and therefore to correctly conclude certain assumptions about it. This blog post will introduce you to the different data types you need to know, to do proper exploratory data analysis (EDA), which is one of the most underestimated parts of a machine learning project.



Introduction to Data Types

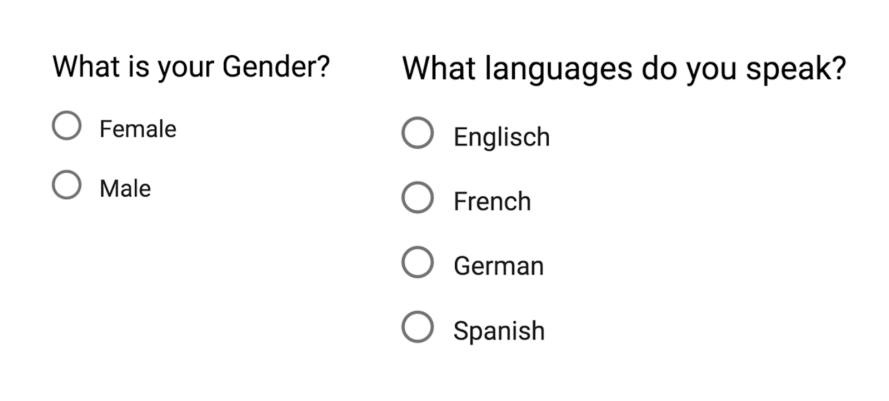
* Having a good understanding of the different data types, also called measurement scales, is a crucial prerequisite for doing Exploratory Data Analysis (EDA), since you can use certain statistical measurements only for specific data types.
* You also need to know which data type you are dealing with to choose the right visualization method. Think of data types as a way to categorize different types of variables. We will discuss the main types of variables and look at an example for each. We will sometimes refer to them as measurement scales.

**Categorical Data**

* Categorical data represents characteristics. Therefore it can represent things like a person’s gender, language etc. Categorical data can also take on numerical values (Example: 1 for female and 0 for male).
* Note that those numbers don’t have mathematical meaning.

**Nominal Data**

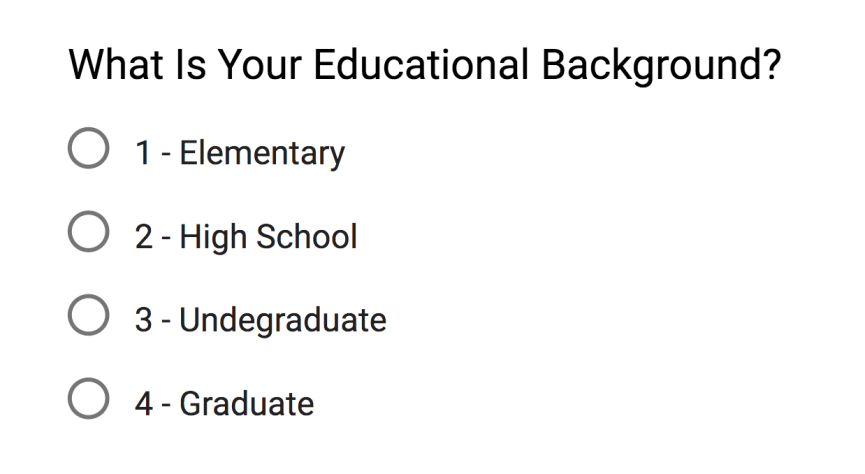
* Nominal values represent discrete units and are used to label variables, that have no quantitative value. Just think of them as „labels“. Note that nominal data that has no order.
* Therefore if you would change the order of its values, the meaning would not change. You can see two examples of nominal features below:



The left feature that describes a persons gender would be called „dichotomous“, which is a type of nominal scales that contains only two categories.

**Ordinal Data**

* Ordinal values represent discrete and ordered units. It is therefore nearly the same as nominal data, except that it’s ordering matters. You can see an example below:



* Note that the difference between Elementary and High School is different than the difference between High School and College. This is the main limitation of ordinal data, the differences between the values is not really known.
* Because of that, ordinal scales are usually used to measure non-numeric features like happiness, customer satisfaction and so on.

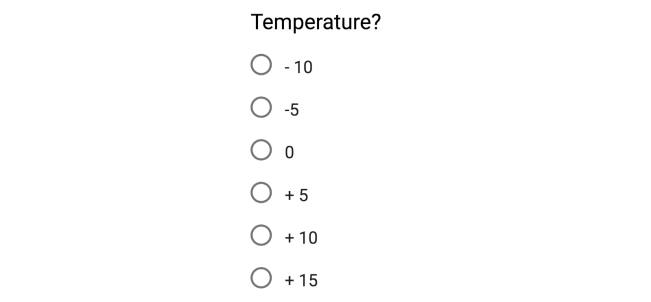
**Numerical Data**

**1. Discrete Data**

* We speak of discrete data if its values are distinct and separate. In other words: We speak of discrete data if the data can only take on certain values. This type of data can’t be measured but it can be counted. It basically represents information that can be categorized into a classification. An example is the number of heads in 100 coin flips.
* You can check by asking the following two questions whether you are dealing with discrete data or not: Can you count it and can it be divided up into smaller and smaller parts?

**2. Continuous Data**

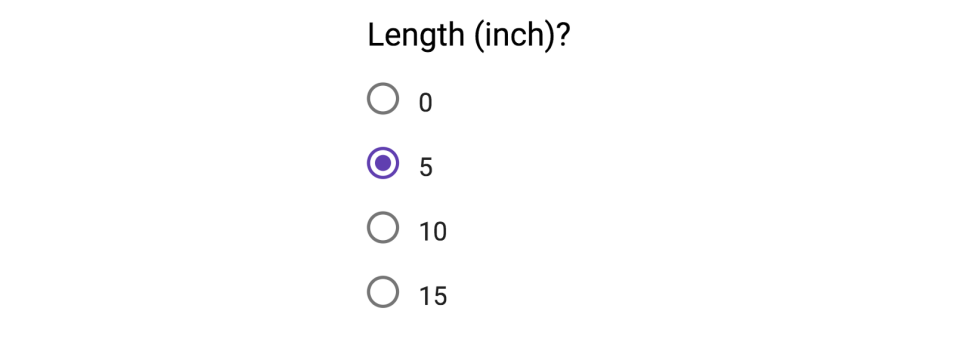
* Continuous Data represents measurements and therefore their values can’t be counted but they can be measured. An example would be the height of a person, which you can describe by using intervals on the real number line.
* Interval Data
* Interval values represent ordered units that have the same difference. Therefore we speak of interval data when we have a variable that contains numeric values that are ordered and where we know the exact differences between the values. An example would be a feature that contains temperature of a given place like you can see below:



* The problem with interval values data is that they don’t have a „true zero“. That means in regards to our example, that there is no such thing as no temperature. With interval data, we can add and subtract, but we cannot multiply, divide or calculate ratios. Because there is no true zero, a lot of descriptive and inferential statistics can’t be applied.

**Ratio Data**

Ratio values are also ordered units that have the same difference. Ratio values are the same as interval values, with the difference that they do have an absolute zero. Good examples are height, weight, length etc.



**Why Data Types are important?**

Data types are an important concept because statistical methods can only be used with certain data types. You have to analyze continuous data differently than categorical data otherwise it would result in a wrong analysis. Therefore knowing the types of data you are dealing with, enables you to choose the correct method of analysis.

Statistical Methods

Nominal Data

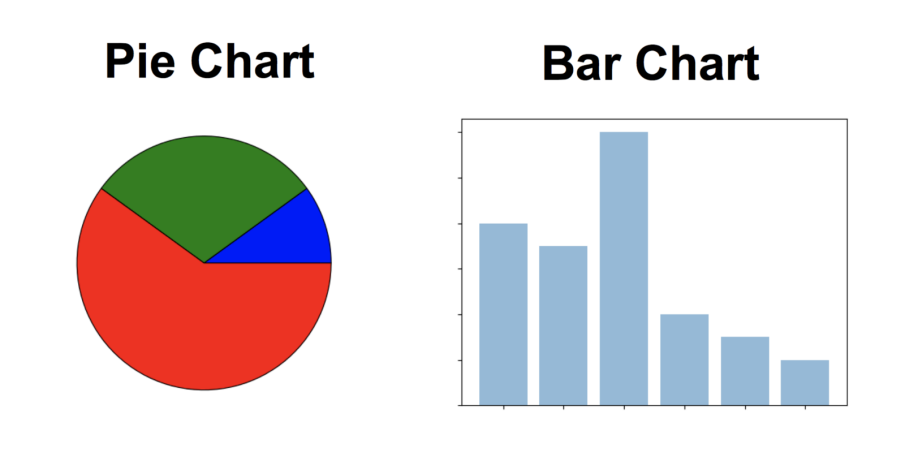
When you are dealing with nominal data, you collect information through:

Frequencies: The Frequency is the rate at which something occurs over a period of time or within a dataset.

Proportion: You can easily calculate the proportion by dividing the frequency by the total number of events. (e.g how often something happened divided by how often it could happen)

Percentage.

Visualization Methods: To visualize nominal data you can use a pie chart or a bar chart.



In Data Science, you can use one hot encoding, to transform nominal data into a numeric feature.

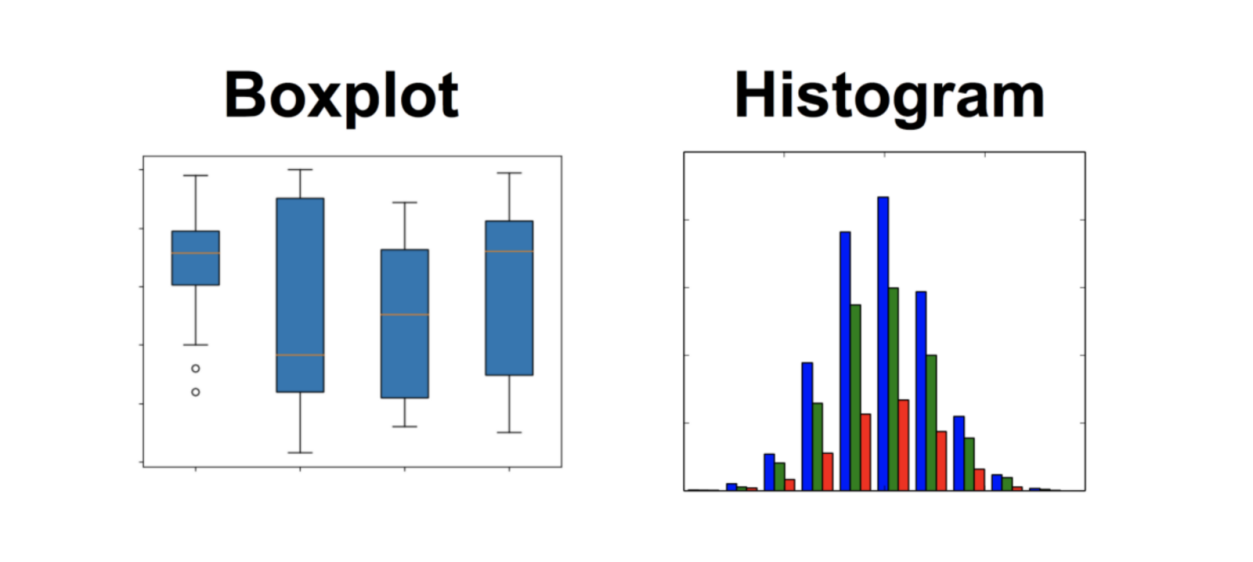
**Ordinal Data**: When you are dealing with ordinal data, you can use the same methods like with nominal data, but you also have access to some additional tools. Therefore you can summarize your ordinal data with frequencies, proportions, percentages. And you can visualize it with pie and bar charts. Additionally, you can use percentiles, median, mode and the interquartile range to summarize your data.

In Data Science, you can use one label encoding, to transform ordinal data into a numeric feature.

**Continuous Data**: When you are dealing with continuous data, you can use the most methods to describe your data. You can summarize your data using percentiles, median, interquartile range, mean, mode, standard deviation, and range.

**Visualization Methods:**

To visualize continuous data, you can use a histogram or a box-plot. With a histogram, you can check the central tendency, variability, modality, and kurtosis of a distribution. Note that a histogram can’t show you if you have any outliers. This is why we also use box-plots.



**Practical Overview**

NOTE:

Install R

Install R studio

Create new file and write in script area

Link for R: <https://cran.r-project.org/bin/windows/base/>

Link for R studio : <https://www.rstudio.com/products/rstudio/download/>

Understanding Basic Data Types in R

R has 6 (although we will not discuss the raw class for this workshop) atomic vector types.

character

numeric (real or decimal)

integer

logical

complex

Run this code in R or R Studio

num<-5.0

class(num) # numerical variable

int<-5L

class(int) # integer variable

str<-"prachi"

class(str) # string variable

com<-5+3i

class(com) # complex variable

log1<-TRUE

class(log1) # logical variable

**(NOTE: By *atomic*, we mean the vector only holds data of a single type.**

character: "a", "swc"

numeric: 2, 15.5

integer: 2L (the L tells R to store this as an integer)

logical: TRUE, FALSE

complex: 1+4i (complex numbers with real and imaginary parts)

# to generate the sequence of numbers

se<-1:10

se

# to generate the sequence of numers with step

se1<-seq(1,100,2)

se1

non- atomatic data types in R

A vector is what is called an array in all other programming languages except R — a collection of cells with a fixed size where all cells hold the same type (integers or characters or reals or whatever).

A list can hold items of different types and the list size can be increased on the fly. List contents can be accessed either by index (like mylist[[1]]) or by name (like mylist$age).

A matrix is a two-dimensional vector (fixed size, all cell types the same).

An array is a vector with one or more dimensions. So, an array with one dimension is (almost) the same as a vector. An array with two dimensions is (almost) the same as a matrix. An array with three or more dimensions is an n-dimensional array.

A data frame is called a table in most languages. Each column holds the same type, and the columns can have header names.

R Code for understanding non- atomic data

|  |
| --- |
| "vactor data type in R"  v <- TRUE  print(class(v))  v <- 23.5  print(class(v))  v <- 2.3L  print(class(v))  v <- 2+5i  print(class(v))  v <- "TRUE"  print(class(v))  # Create a vector.  apple <- c('red','green',"yellow")  print(apple)  # Get the class of the vector.  print(class(apple))  vect1<-length(c("aa", "bb", "cc", "dd", "ee"))  class(vect1)  # to generate sequence  x1 <- 1:7  x1  y1<- 2:-2  y1  vect2<-seq(1, 3, by=0.2)  vect2  x<-c(-3,-2,-1,0,1,2)  x  x[2] <- 0 # will assign value 0 to 2nd index  x  x[x<0] <- 5 # all less than 0 will be replaced by 5  x  x<-c(-3,-2,-1,0,1,2)  vect1<-c(22,23,24,25,26)  vect1  vect2<- vect1 - c(2,1)  vect2  list1<-list("1",2,"true",T)  list1  class(list1)  class(list1[4])  list2<-list(1,"2")  list3<-c(list1,list2)  list3  class(list3)  v1<-c(2,12,34,25,65)  v2<-c(23,43,76,81,12,34,54,99)  arr<-array(c(v1,v2),dim=c(5,5,3))  arr  my\_matrix <- matrix(1:6, nrow=3, ncol=2,byrow = T)  my\_matrix  dim(my\_matrix)  my\_matrix[,2]  my\_matrix[,1]  my\_matrix[2,]  my\_matrix[1,]  x <- c(1, 2, 3, 4, 5, 6)  y <- c(10, 20, 30, 40, 50, 60)  cbind(x,y)  v1<-c(12,23,24,12,55)  v2<-c(23,24,84,54,66)  mtr<-matrix(c(v1,v2))  mtr  mtr<-matrix(c(v1,v2),5,5)  mtr  mtr<-matrix(c(v1,v2),8,6)  mtr<-matrix(c(v1,v2),3,5)  mtr<-matrix(c(v1,v2),10,15)  mtr  vect1<-c(1,2,2,3,4,5,6,6,3,4,1)  vect1  class(vect1)  fact1<-factor(vect1) # will convert vector into factor , factor has discrete levels  class(fact1)  fact1  id<-c(100:105)  names<-c("prachi","sonal","rahul","maya","rehan","ruby")  new<-data.frame(id,names) # will generate table structure  new  #operators  v <- c( 2,5.5,6)  t <- c(8, 3, 4)  print(v+t)  v <- c( 2,5.5,6)  t <- c(8, 3, 4)  print(v-t)  v <- c( 2,5.5,6)  t <- c(8, 3, 4)  print(v\*t)  v <- c( 2,5.5,6)  t <- c(8, 3, 4)  print(v/t)  v <- c( 12,5,6)  t <- c(8, 3, 4)  print(v%%t)  v <- c( 2,25,37)  t <- c(8, 3, 4)  print(v%/%t)  v <- c( 2,5.5,6)  t <- c(8, 3, 4)  print(v^t)  #relational operators  v <- c(2,5.5,6,9)  t <- c(8,2.5,14,9)  print(v>t)  v <- c(2,5.5,6,9)  t <- c(8,2.5,14,9)  print(v < t)  v <- c(2,5.5,6,9)  t <- c(8,2.5,14,9)  print(v == t)  v <- c(2,5.5,6,9)  t <- c(8,2.5,14,9)  print(v<=t)  v <- c(2,5.5,6,9)  t <- c(8,2.5,14,9)  print(v>=t)  v <- c(2,5.5,6,9)  t <- c(8,2.5,14,9)  print(v!=t)  #logical operators; All numbers greater than 1 are considered as logical value TRUE.  v <- c(3,1,TRUE,2+3i)  t <- c(4,1,FALSE,2+3i)  print(v&t)  v <- c(-3,0,TRUE,2+2i)  t <- c(-4,0,FALSE,2+3i)  print(v|t)  v <- c(3,0,TRUE,2+2i)  print(!v)  #assignment operator  v1 <- c(3,1,TRUE,2+3i)  v2 <<- c(3,1,TRUE,2+3i)  v3 = c(3,1,F,2+3i)  print(v1)  print(v2)  print(v3)  v1 <- 8  v2 <- 12  t <- 1:10  print(v1 %in% t)  print(v2 %in% t) |