

Coursera - Data Science & R Course

Daniel O.

15/12/2020

Types of Data Science Questions

Descriptive

Goal: To describe or summarize a set of data

- Early analysis when receiving new data
- Generate simple summaries about the samples and their measurements
- **Not** for generalizing the results of the analysis to a larger population

Exploratory

Goal: To examine the data and find relationships that weren't previously known

- Explore how different variables might be related
- Useful for discovering new connections
- Help to formulate hypotheses and drive the design of future studies and data collection

Inferential

Goal: Use a relatively small sample of data to say something about the population at large

- Provide your estimate of the variable for the population and provide your uncertainty about your estimate
- Ability to accurately infer information about the larger population depends heavily on sampling scheme

Predictive

Goal: Use current and historical data to make predictions about future data

- Accuracy in predictions is dependent on measuring the right variables
- Many ways to build up prediction models with some being better or worse for specific cases

Causal

Goal: See what happens to one variable when we manipulate another variable

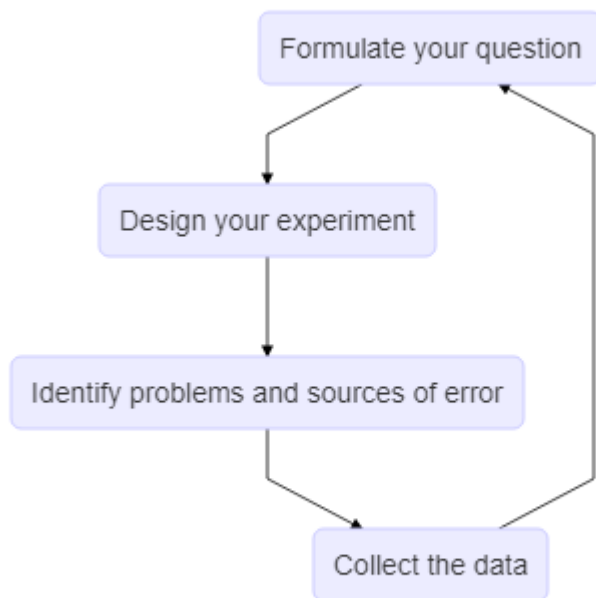
- Gold standard in data analysis
- Often applied to the results on randomized studies that were designed to identify causation
- Usually analyzed in aggregate and observed relationships are usually average effects

Mechanistic

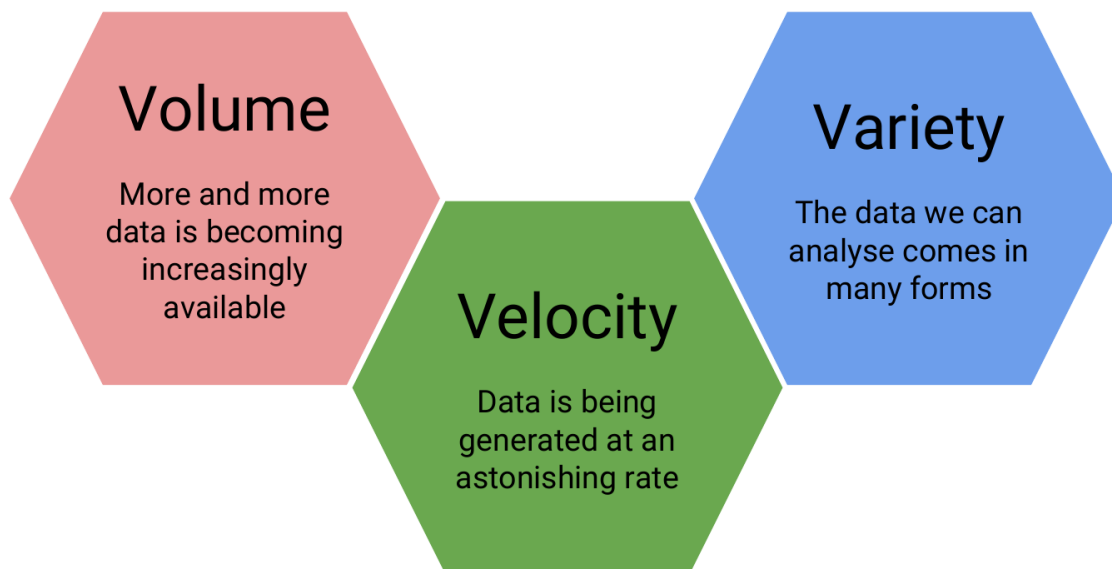
Goal: Understand the exact changes in variables that lead to exact changes in other variables

- Applied to simple situations or those that are nicely modeled by deterministic equations
- Commonly applied to physical or engineering sciences
- Often, the only noise in the data is measurement error

Experimental design



Big Data



Practical R Exercises in swirl

- `install.packages("swirl")`
- `library(swirl)`
- `install_from_swirl("R Programming")`
- `swirl()`
- `getwd()`
- `setwd()`
- `ls()`
- `dir()`
- `dir.create()`
- `file.create()`
- `file.exists()`
- `file.infor()`
- `file.path()`
- `file.rename()`

- `file.copy()`

—

- `length(x)`
- `x <- dim(x,y)` » Creates matrix
- `attributes(x)`

—

- `matrix(data, nrow,ncol)`
- `cbind({vector with row names}, matrix)` » Combine columns
- `colnames(data.frame) = {vector with column names}` » Names of columns
- `data.frame({vector with row names}, matrix)` » Allows Text and Numbers

Types of Data Science Questions

Control Structures - if/else

```
if(condition) {
```

```
  <do something>
```

```
} else("condition2") {
```

```
  <do something else>
```

```
}
```

—

```
if(condition) {
```

```
  <do something>
```

```
{ else if(condition2) {
```

```
  <do something different>
```

```
} else {
```

```
  <do something else>
```

```
}
```

Control Structures - for loop

```
for(i in vect){  
  
  <function>[i]  
  
}
```

Control Structures - while

```
while(finite condition){  
  
  <do something>  
  
}
```

Control Structures - repeat, next, break, return

repeat is a construct that basically initiates an infinite loop. The only way to exit a **repeat** loop is to call **break**

next is basically used in any time of looping construct when you want to skip an iteration.

return signals that a function should exit and return a given value

Functions

Basic format:

```
f <- function(x, y =  
) {  
  x+y  
}
```

User-defined binary operators have the following syntax:

```
%[whatever]%
```

where [whatever] represents any valid variable name.

Dates and Time

Dates are represented by the **Date** class and times are represented by the **POSIXct** and **POSIXlt** classes. Internally, dates are stored as the number of days since 1970-01-01 and times are stored as either the number of seconds since 1970-01-01 (for **POSIXct**) or a list of seconds, minutes, hours, etc. (for **POSIXlt**).

strptime() converts character vectors to **POSIXlt**.

If you want more control over the units when finding the above difference in times, you can use **difftime()**, which allows you to specify a 'units' parameter.

Loop functions

`lapply`: Loop over a list and evaluate a function on each element

`sapply`: Same as `lapply` but try to simplify the result

`apply`: Apply the function over the margins of an array

`tapply`: Apply a function over the subsets of a vector

`mapply`: Multivariate function of `lapply`