

---

# Table of Contents

README.md	1.1
Syllabus	1.2
Introduction	1.3
Homework 1	1.3.1
Data Acquisition	1.4
Data Storage	1.5
Homework 2	1.5.1
Data Analysis	1.6
ElasticSearch	1.6.1
Basic Python	1.6.2
Numpy & Pandas	1.6.3
Intro to Machine Learning	1.6.4
Homework 3	1.6.5
Data Visualization	1.7
Homework 4	1.7.1
Project	1.7.2
Students Feedback	1.8

# Data Science 2016

build **passing**

Welcome to the Spring 2016 Data Science class!

As a future data analyst, you are responsible to **collect**, **store**, **analyze** and effectively **report** on data insight.

In this class, we will be learning how to clean up messy data, uncover patterns and insights to conclude or to make predictions. Moreover, we will be communicating our findings with beautiful visualization!

You are welcome to join the class [gitter chat room](#). In the interest of being efficient, we encourage everyone to ask questions publicly so that common questions can be answered once.

In this quarter, we will publish all course materials on the Github and on [Gitbook](#). This implies you **do not** need a textbook for the class as all required materials will be provided.

Please feel free to utilize Github issue or Gitbook comment to share your thoughts on the course throughout the quarter

chat **on gitter**

## Quick links

- [Announcements](#)
- [Syllabus](#)
  - [Schedule](#)
- [Gitbook](#)
- [Feedbacks](#)

# Data Science

## Course Overview

### Objectives

The primary goal for this Data Science class is to collect (acquire), store, analyze and visualize data for answering questions pertaining to *the third order of knowledge*.

Students will perform programming parts throughout the quarter to finish the final course project that achieve above requirements.

### Expectations

We expect students to have strong Java programming background with ability to manage your own programming environment such as installing new tools and maintaining your tools.

In addition, we also expect students to spend a large amount of time learning new technologies and coding outside of course schedule.

Finally, the programming assignments are **non-trivial** for a number of reasons, students are expected to understand and develop programs that have varying algorithmic complexities. To this end, leaving the homework assignments to the last minute is a surefire way to *not* pass the course.

## Logistics

### Instructor-in-charge:

- Eric Liao [rliao01@gmail.com](mailto:rliao01@gmail.com)

### Course assistant:

- John Tran [johnjtran@icloud.com](mailto:johnjtran@icloud.com)

### Schedule:

- Undergraduate: Sunday 9:10 to 13:00 at ET-A309
- Graduate: Sunday 13:10 to 17:00 at BIOS-144

## Office hours:

- Sunday 17:00 to 19:00 at BIOS-144
- Online via gitter as available (usually 24/7)

Please do ask the question early and often than leaving questions at the last moment (e.g. 1am before the homework is due). Although instructors are generally available for most of time, we are not obligated to keep up with all last moment questions with students right before due date.

## Textbook:

Students do not need to purchase a textbook for this class as this Github will provide all course materials.

## Computer:

Students are required to have a laptop computer for the course. On some assignments and certainly for the final project where a live demo is required, it is unreasonable to coordinate with anything other than a laptop.

We do not endorse a particular platform or OS; however, we encourage students to pick something that has reasonable computing horsepower. For this reason, Chromebooks or tablets are not suitable for serious software development and data processing.

## Course Objectives

- Collect data
  - Java crawler/collector
  - [Optional] Akka framework
- Store big data
  - MongoDB
  - Elastic Search
  - [Optional] HDFS
- Analyze big data
  - Elastic Search query
  - Python
  - [Optional] Hadoop MR (Map reduce)
- Visualize data
  - Basic JavaScript, HTML & CSS
  - D3.js

# Grading Policy

The purpose of this grading policy is to fairly evaluate student's performance. Please review them carefully.

## Grading Allocation

- Homework - 40 points
- Quizzes - 40 points
- Project - 20 points
- Attendance & decorum - 5 points (more below)

## Grading Scale

The final grade will be adjusted to a 100 point scale as followed.

- A: 94 to 100
- A-: 90 to 93
- B+: 85 to 89
- B: 80 to 84

Graduate students are required to get a grade 80 or above to pass the course

- B-: 77 to 79
- C+: 74 to 76
- C: 70 to 73

Undergraduate students are required to obtain a grade 70 or above to pass the course

- F: 0 to 69

Successful completion of the course project is **required** for passing this course.

## Course Decorum

We expect you to show civility and concerns for your classmates. We expect for you to approach the class with a positive attitude and professional demeanor.

This includes remaining alert (and awake!) in class, respecting and never interrupting others, limiting private conversations, and keeping phones off.

Because the class is a fast-paced and demanding class, there will be opportunities for overwhelmed feeling. The best way to address these challenges is to discuss the course on gitter and with the instructors. Please ask for help often and early.

The instructors are obligated to listen to the students' concerns and to treat each student with the needed respect and dignity. However, the instructors are not obligated to honor any and all requests. Especially in those cases when students ask for assistance during the 11th hour.

## Request for a Regrade

In those cases of arithmetic errors, please bring the issue to the instructor-in-charge immediately. We will make the necessary adjustments.

In those cases where students believe that the grade rendered does not reflect the quality of the work, students must submit in writing within 1 week a request for regrade with the following information:

- Name and CIN number
- Assignment #/Exam #/Project
- Clearly articulated rationale with supporting data to backup your claim

We will review your request for a regrade and respond accordingly. Please note that as a matter of principle the **entire** assignment, exam, or project will be evaluated. This is because, more often than none, our grading methods are generous to begin with and that we always give students the benefit of doubt.

It is entirely possible that a regrade request can result in an overall lower score for the assignment or exam. With respect to the course project or the final grade, it is possible that students can actually fail the course. Please do the proper risk reward analysis.

In those cases where **students feel that they should get more because** \_\_. Where \_\_ includes (but not limited to):

- They worked really hard on the assignment or project
- They would have to endure shame with family and friends
- They need XYZ grade to graduate

Please **do not bother**. We will not entertain such a request.

## Late Policy

Students are permitted **one** 1-week late submission or demonstration on the programming assignments provided that they coordinate *ahead of time with the instructors*. After the second week, the grade for assignment will be 0.

The same provision does not extend to the course project. The course project demonstration will take place during the university's designated final exam day and will not be offered at any other time. Please plan your study schedule and travel arrangements accordingly.

## Academic Integrity

Cheating will not be tolerated. Once adjudicated, all involved parties will receive a grade of **F for the course** and be reported to the Computer Science Department.

### Definition of Cheating

In our view, cheating is a disease in academia that undermines the value of a quality education. What's worse, it is a selfish act that is ultimately injurious to the individual and the collective.

For this, let us obviate any ambiguity with a commonly accepted definition of cheating:

Cheating is claiming someone else's work to be your own. Cheating is not just receiving unauthorized assistance on an assignment, exam, or project but also providing the solution to others.

Note that we do not discourage discussion and collaboration. However, we are very adamant that you utilize the class's discussion board and office hours for this purpose.

### Determination of a Cheating Incident

Over the years, we have developed a number of strategies and techniques to identify cheating. The biggest tell-tail sign of cheating is when students do not understand their solution and/or source code. Second, we have automated tools for source code analysis. To this extent, simply reformatting source code or renaming variables is pointless.

While we will listen to your explanation, once decided we will not change our determination. You can, however, be assured that we will treat students with dignity and respect throughout the process.

### Adjudication Process

We will inform all involved parties **in writing** of our observation, along with evidence and our intention to administer the appropriate penalty. For this course, the only available penalty is an F as the final grade.

Each student then will have 1 week to respond individually and **in writing** with his or her side of the story.

Note that an apology, recognition of mistake, and/or promise to never do it again will not be considered and will most definitely NOT have an affect on the verdict. A non-response is considered an acceptance of responsibility.

We seek out a opinion from a non-interested third party reviewer, most likely another instructor. The student's identity will be anonymized when we make a request for a third party evaluation. The review package will only include:

- Course syllabus
- Accusation
- Supporting Evidence
- Student's response

The only question that we present to the third party reviewer will be **did cheating take place?** We will be very clear that we are not interested in gradation, severity, or level of cheating nor are we interested intent or rationale.

The bottomline: don't cheat, if you cheat, you would simply get an F for course. This is not negotiable. So please do the proper risk reward analysis.

Finally, we will inform all involved of our final decision. By definition, a final decision is **final**.

## ADA statement

Reasonable accommodation will be provided to any student who is registered with the Office of Students with Disabilities and requests needed accommodation.

## Course Schedule

The schedule below is tentative and is subject to change.



Week # [date]	Topic	Notes
1 [4/3]	<a href="#">Introduction</a>	Form a team, set up environment
2 [4/10]	<a href="#">Data Acquisition</a>	<b>Homework 1 &amp; Quiz 1</b>
3 [4/17]	<a href="#">Data Storage</a>	Install MongoDB
4 [4/24]	<a href="#">Data Storage</a>	Install Elastic Search, docker <b>Homework 2</b>
5 [5/1]	<a href="#">Data Analysis</a>	Elastic Search Query <b>Quiz 2</b>
6 [5/8]	<a href="#">Data Analysis</a>	MongoDB MapReduce <b>No class but remote video and notes</b>
7 [5/15]	<a href="#">Data Analysis</a>	Install Python <b>Quiz 3</b>
8 [5/22]	<a href="#">Data Analysis</a>	Intro to Machine learning
9 [5/29]	<a href="#">Data Visualization</a>	JavaScript and D3 <b>Quiz 4 Homework 3</b>
10 [6/5]	Project	Project demo <b>Homework 4</b>
Final [6/12]	Project	Ready for demo!

You have choice of presenting the project between week 10 (June 5) and week 11 (June 12). Please select your date accordingly.

This syllabus is subjected to change. In the event of an update, we will notify students and provide a rationale for the adjustment.

# Introduction

Welcome to Data Science course again!

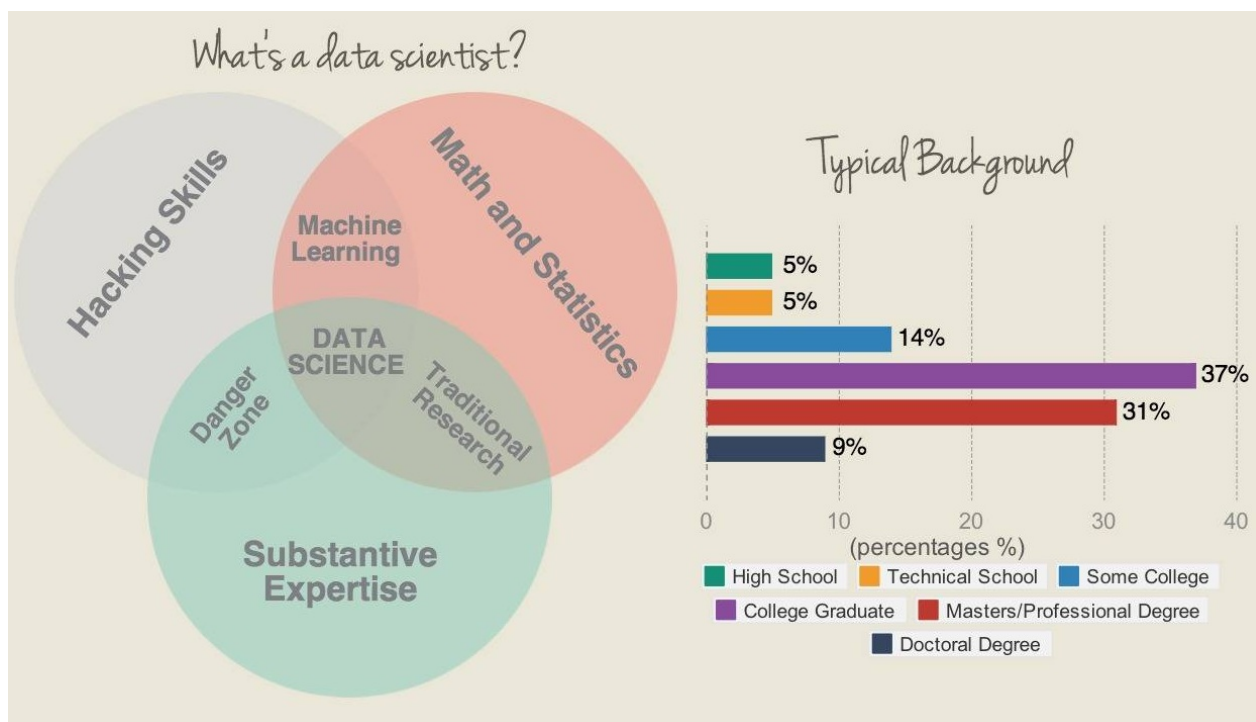
## Objectives

- [Syllabus](#)
- Introduction to Data Science
- Set up development environment (Java & Gradle)
- [Form a team & create Github team repo](#)

## Metrics

- [Github team repo](#)
- Data question - [homework 1](#)
- Pass unit tests

## What is data science?



Credit: <http://i.imgur.com/AfFMkHe.jpg>

Data scientist is to make sense of data to make conclusion or even to predict outcome based on the data sets.

Common skill sets of data scientist:

- Math Statistics
- Python / R
- D3 or some other visualization tool
- Hadoop / Elastic Search
- Database reading knowledge (like sql query, mongo db script)
- Machine learning

### Recommended readings

- [What is data science by O'reilly](#)
- [The sexiest job at 21th century by Hbr](#)

## Class Overview

### 1. Question

In this class, we will be starting by asking **question**. Your job this week is to do a lot of research on the question such as what data sets can you use to support your question/statement.

### 2. Acquiring

With a good question, you will likely want to start your project by **acquiring data**. We will be programming in our *favorite language*, Java. In other word, your job is to implement my data collector interface and implement detail to collect data.

### 3. Storage

After collecting huge amount of data, it's important to make decision on how to **store them**. To simplify our life a little bit, we will be using *MongoDB* as the primary database with Elastic Search as the secondary database for quick searching and exploring.

### 4. Explore & Analysis

Then, we will cover the most important part of the course, **analytics**. In this part, we will start by utilizing Elastic Search for quick exploring of data. In other word, you can use Elastic Search a lot to do quick searching and make sense out of this huge set of data. After having some basic knowledge of the data, we will be learning on how to process them to do analysis. We will go over some basic Python with its libraries to do some basic Machine Learning.

### 5. Communication via visualization

Once you are done analysis, we will be learning on how to create visualization based on the analysis we done earlier!

## What makes a good question?

For the purpose of this course we consider the order of knowledge:

- First order: obtaining information directly from the data or metadata
- Second order: comprehension of first order knowledge
- Third order: derive inferential information or predicting an outcome that is derived from data

A good question is aim to address third order knowledge! In other word, just download a data set and get the size of data is not consider to be good question! This video dives further indepth into how to formulate a good question for Data Science.

# Orders of Knowledge

## Some starting points of data sets

### [Awesome data sets](#)

You may use the above awesome list to find out some initial good data sets as a starting point to ask some good questions.

### [Kaggle](#)

Kaggle is awesome machine learning or data analytics competition site. It may be interesting to see if you can resolve one of their open challenge with the techniques we learn in this class.

### [Google public data set](#)

Google also provides some data set that you can use their BigQuery to do some processing.

## AWS public data set

Amazon hosts some data set as well!

## What is considered to be big data?

- Volume
- Velocity
- Variety

Above 3Vs define the properties of big data. Volume refers to the size of data (GB, TB or even PB), variety refers to the number of types of data and velocity refers to how fast slow data comes in.

### Interesting trending

Big data got started from 1990s to early 2000s when larger internet companies forced to invent new way to manage big volume of data. Today, most people think of Hadoop or NoSQL database like MongoDB when they of Big Data. However, the original core components of Hadoop, HDFS (Hadoop Distributed File System—for storage), MapReduce (the compute engine), and the resource manager now called YARN (Yet Another Resource Negotiator) are rooted in the batch-mode or offline processing commonplace ten to twenty years ago, where data is captured to storage and then processed periodically with batch jobs. Most search engines worked this way in the beginning. The data gathered by web crawlers was periodically processed into updated search results.

- [Fast Data: Big Data Evolved By Dean Wampler, PhD](#)

---

## Development Environment Setup

1. Install [Java](#) if you have not done so

Keep it in mind you have to set up Java in `Path` variable for windows user  
You should be able to do `java -version` to see 1.8 as version from here

2. Install [Gradle](#)

Remember to set up `JAVA_HOME` pointing to where you install your JDK  
You should be able to run `gradle -v` to see gradle version

3. Install [Git](#)

You should be able to find git bash under windows if you already install it

4. Clone this repository

`git clone` or use Github client or download as zip whatever you want

5. Run `gradle test`

## Java

Install [OracleJDK 8](#) if you don't already have one.

### Windows User

Click on the link above (OracleJDK 8) to download Java 8. Upon completion of download, please set up the `PATH` path on your advanced environment settings from right click on your computer.

Remember to set it to your JDK **bin** folder

### Mac User

You can install [brew](#) and follow the following to install Java 8.

```
brew tap caskroom/cask
brew install brew-cask
brew cask install java
```

## Gradle

Install [Gradle](#) as this will be our primary build tool.

### Windows User

Click on the link above and install Gradle accordingly. Remember to set up `PATH` variable so that your terminal knows Gradle is executable.

Also you will need to set up `JAVA_HOME` pointing to your JDK folder. In example, `C://jdk8/`

### Mac User

Install via `brew install gradle` assuming you have `brew` installed.

### Linux User

- CentOS users can follow the instruction found in [Github Gist](#).
- Ubuntu users take a look at the [Ask Ubuntu Stack Exchange Tutorial](#).

**To check Gradle is installed**

Please run `gradle -v` anywhere from terminal. You should see Gradle version as 2.12.

## Wrap Up Java Review Exercise

Clone/download the course repository, run `gradle hello` after you are done. You should see `Hello Data Science` as the console output.

Once you have above environment set up, please remove all the `@Ignore` from `src/test/edu/csula/datascience/examples/SimpleStatsTest.java` and pass all the test from there.

What you want out of this class is `gradle test` passes.

## Eclipse Gradle plugins

With above being done, you can start modifying your project in Eclipse. However, you are still not able to run the Gradle tasks. Therefore, you will also need this *Eclipse Gradle Plugin* to run the Gradle tasks (e.g. hello)

### Instructions to install Gradle plugins in Eclipse

1. In Eclipse Open Help >> Install New Software
2. Paste a Gradle update site link -- <http://dist.springsource.com/release/TOOLS/gradle> -- into the "Work with" text box.
3. Click the Add button at the top of the screen.
4. Ensure that the option "Group Items by Category" is enabled.
5. Select the top-level node 'Extensions / Gradle Integration'.
6. Click "Next". This may take a while.
7. Review the list of software that will be installed. Click "Next" again.
8. Review and accept license agreements and Click "Finish".

### Instructions to run Gradle tasks in Eclipse

1. import this repository as gradle project
2. Right click and run gradle task

Although I do suggest all of you to run tasks from terminal/cmd.

## Recommended readings for Gradle

- <http://gradle.org/getting-started-gradle-java/>
- <http://www.vogella.com/tutorials/Gradle/article.html>

## Recommended reading for Git/Github

If you have trouble with Git/Github, you can look through [this document](#) as quick tutorial.

Still have trouble? Please feel free to raise your hand and I'll be walking around to help.

## Unit Testing

Test Driven Development or Behavior Driven Development gives you a lot more confidence of refactoring in future. Moreover, testing is often being adapted at more popular Open Source projects. Why? Because testing gives the confidence of merging codes from unknown developers.

How do we measure unit test?

In this class, I'll set up Coverall as the code coverage tool to measure how much unit tests students implement. This will give me fair amount of testing you implement for your project. Example can be seen in [this repo](#).

## So how do you test?

- Dependency Injection
- Avoid static state
- Keep each unit small

### Dependency Injection

Dependency injection doesn't need to always be done by framework like [Guice](#). Put it simple, you can define dependency in your constructors. If you want to get fancy, you might want to use `Factory pattern` to protect your constructors logic being exposed.

All in all, you want to keep your module dependency being defined in clear way so you can mock them.

In example, if you have a piece of code need to take object from database. Instead of:

```
public class Test {
    public Map<String, Integer> countNumberOfWords() {
        try (Connection c = getConnection()) {
            String sql = "SELECT * FROM test";

            // use connection and get list of object out
        }
    }
}
```

to:



```
public class Test {  
    public Map<String, Integer> countNumberOfWords(List<Test> tests) {  
        // count number of words using plain old java object  
        // this way, code becomes easily testable and mockable  
    }  
}
```

Why testings need to be done at design phase?

When designing your functions/methods, you have to think about how to test it. What dependencies do you need for object and so on. If you do testing afterward, it simply becomes impossible to mock any dependency because they are too deep into your code.

## Further reading

- <https://github.com/okulbilisim/awesome-datascience>
- <http://matrturck.com/2016/02/01/big-data-landscape/>
- <http://stackoverflow.com/research/developer-survey-2016>

# Homework 1

Please keep it in mind that homework1 can still be modified until the mid of this upcoming week. So stay tuned!

## Description

This homework is a individual homework; thus, please finish this homework by yourself.

In data science, we start the process by asking a good question. In homework 1, your job is to ask a specific question with some of your initial research. In other word, you will need turn in a short write up for your question as homework 1. Be sure to review the video on the [orders of knowledge](#) before completing this assignment.

You will need to answer the followings in your write up:

- What domain (area) will you explore for your project?
- What is the question you wish to answer?
- What data do you plan to use to support your statement above?
- What do you know about the data so far?
- Why did you choose this question?

A few characteristics of a good question:

- Clearly defined: question should be able to summarized in single statement.
- As focus as possible: question should have a narrow focus rather than broad approach.
- Reasonable data available: you should be able to find data set to support up your question.
- Should be able to reach the third level knowledge - predict the future.

## Tasks

Your task is to put your write up individually on a [secret gist](#). Once you are done writing, please submit your gist address to CSNS homework 1.

[Example can be found here](#)

## Question description

- What is the question you wish to answer?
- What data do you plan to use to support your statement above?
- What do you know about the data so far?
- Why did you choose this question?

question.md hosted with ♥ by [GitHub](#)

[view raw](#)

## Deliverables

- Gist link on CSNS - homework 1

To submit a link on CSNS, you can simply attach a plain text file `homework1.txt` with your name, CIN, and gist URL.

## Grading Rubric

- Answer 5 questions above [10 pts].

Please note that if you do not submit this assignment via [secret gist](#) you will **not** get credit.

# Data Acquisition

To collect data ... in any format your question may require from any source.

But ... most of time, data comes in with some variety.

How do we handle with this messy data? After cleaning up data, how do we store this data?

## Objectives

- Data variety
- Common data acquisition problems
- Data quality
- Java data collector
- Data cleaning
- Git tutorial
- Set up team repository
- [Homework 2](#)

## Metrics

- Java data collector
- Team repo
- Initial storage concern
- Test test test

## Data variety (problems)

When acquiring data, they usually come in with some variety (including good and bad!).

Therefore, it is important for data scientists to have ability to work with this diversity. In other word, data scientists need to be able to:

- Test about assumptions about data
  - Value
  - Data type
  - Shape
- Identify errors or outliers
- Finding missing values

## How do we acquire data?

To start with, we will talk about some common ways to get data. This includes:

- Http request

The most common and low level approach to collect data. In example, you may be making http request to a url to download data (this could be in various format such as JSON, binary file or HTML).

- Web scraping

You are scraping HTML content and use it to download further data. In example, you may use HTML parser to get meta information like image url in a HTML page and use this meta information to download the source (image itself).

- Api calls

You may be working with a data provider like Twitter. Then, you will need to use their API client to collect data (like using Twitter client to get streaming tweets).

## Common data format

We are now knowing ways to download data. However, what kind of data format we may be dealing with?

Here are a few common data format but not limited:

- Tabular data

Think about excel spreadsheet or sql table.

- CSV(Comma Separate Values)

Very easy to read by software

- JSON

Probably by far most common data format because it is readable by both human and machine

- Binary files

Sometimes, you may store the binary file like videos, audios or images. Then you may extract meta information out of this binary file.

## Extract data

Upon download data, you will usually need to extra data out of its raw format so your program can start to use this information to do processing.

In example, you might need to deserialize from JSON to your Java object. You might need to extract HTML and read its text or even to download more content from this extracted content.

In nutshell, your program need to be able to load this data from its source to your memory so you can start doing some data cleaning.

## Data cleaning

The process of cleaning up acquired data is often referred to as `data munging` . On the surface, data munging is simply trying to make sense and organize acquired data into structured data.

After initial extraction, you will find out not all data are ideal sooner or later. For instance, there may be missing records. Some data might have inconsistency issues and so on.

This issue is what I called dirty data.

How do we deal with dirty data?

We need perform data cleaning (e.g. removing duplicated entity).

Keep it in mind that data cleaning is an **iterative** process. In other word, you might need to perform multiple iterations in order to get to your ideal data format. Sometimes, this may involve changing the *shape* of the data.

## Data quality

Before we start to talk about the process, we need a way to measure data.

Here are 5 qualities of data:

- Validity

| Conforms a schema

- Accuracy

| Conforms to gold standard

- Completeness

| Contains **all** records

- Consistency

Same data may show up twice with different values

- Uniformity

Same unit

## Blue print of cleaning

Once we can measure data, we want to set up a plan to clean data.

Here is a blue print of data cleaning:

1. Audit your data
2. Create data cleaning plans
  - Identify causes
  - Define operations
  - Test your theory
3. Execute the plan
4. Manually correct

## Data storage concern

Follow up with discussion, data comes in with variety. This implies traditional relational database may have hard time to store the data due to unknown shape. Therefore, it is common to store the data in **plain file** or to store in **NoSQL database**.

Since we will be learning *MongoDB* in next class. I'd suggest you to store all data in plain file format without worry too much on MongoDB storage. In specific, you can be just store your data in CSV or JSON format as long as you can read the file later.

But it's just a plain file we are writing. What about performance?

Plain file format is not that bad. Think about HDFS(Hadoop Distributed File System), it is just plain file from your program point of view! But by underlying implementation, HDFS partitions files and do parallel processing on top. In short, plain file may not be the worst in performance.

## Programming time

- Cli
- Git
- Collector Interface

## Why use terminal?

I will be strongly encourage students to use terminal rather than relying on the Eclipse right click menu to run any task. Reason being in terminal you know it will work regardless of the environment.

Think of this case, if you are working with me in a team. If you say your code works only with Eclipse, this implies an enforcement for me to use Eclipse.

Secondly, you may not be able to use this knowledge of Eclipse outside of Java language. Consider we will be working with Python later, please learn how to use terminal to run bash commands.

## Git cheatsheet

To follow up the strong suggestion on using terminal, the following session on git will be only for terminal. This way, I ensure everyone who has git bash can run the script without problem.

In git, there are few commands you absolutely have to know:

- `git add {files}`

Before you are going to commit the file, you will have to *add* file to index so git know which file to commit.

- `git commit`

This will bring up a editor for you to add commit message. In short, commit is to save changes in git.

- `git fetch`

Fetch is like to get updates from *remote*. In a sense, this tells git what changes are on the remote.

- `git push`

This is how you *upload* your *changes* from your computer to *remote* Github server.

- `git pull`

This is how you *download changes* from *remote* Github server.

A common workflow may be like:



```
# This add all files under current directory
$ git add .

# This commit the changes
$ git commit -m "Commit message"

# This push changes
$ git push
```

# Data Storage

Volume and variety of data is always a the primary concern of big data. How should we store them?

What about data comes in with high velocity? How do we store them and analyze them in real time?

## Objectives

- 4/17
  - Homework 2 clarification
  - Live demo data acquisition
  - [Set up MongoDB](#)
  - [Java MongoDB Driver](#)
  - Store data from acquisition to MongoDB
  - [Docker](#)
- 4/24
  - [Docker](#)
  - Introduction of Elastic Search
  - Set up Elastic Search on local dev machine
  - Learn Elastic Search
  - Set up Elastic Search Java Client
  - Set up Elastic Search on docker container
  - Initial exploration with Elastic Search and Kibana
  - [Optional] HDFS

## Metrics

- 4/17
  - [MongoDB installation](#)
  - [Java MongoDB Driver](#)
  - MongoDB CRUD
  - Docker set up
- 4/24
  - Elastic Search installation
  - Docker provision

## Homework 2

Many students asked questions regarding homework 1 during office hour -- **What is considered to be big data?**

Big data in this class needs to meet at least 2 of the following:

- Volume
- Velocity
- Variety

What does it mean by volume? How large is considered large enough?

To that, lets be more specific on our requirements:

- Volume

Text based data at least **3GB**

Non-text based data, please discuss with me before you move on to acquire it.

- Velocity

Real time or at least near real time data ... constantly getting new values from data source.

- Variety

More than 3 data sources. e.g. Twitter, open weather, CNN news ... etc.

To sum up:

In homework 2, data needs to meet at least two requirements above; or it is not considered to be valid big data.

## Data Collector Demo

Another common questions regarding homework 2 from office hour:

How do I implement this *Collector* and *Source* interface? What does each interface do? How do I collect data from {insert your data source}.

First, we will go over what is our Collector and Source and implement an example of getting data from Twitter in class.

Regarding your question on how do you get data from {insert your data source}:

I will not be answering domain specific questions, such as how do I get data from Twitter.

Why?

This is your choice of your data source and you should be doing a lot of research on how you get data as well as how you are going to use data. In other word, these domain specific questions are your responsibility to answer them to your best effort.

I'd suggest you to Google your question before you ask me any further detail.

Without further due, demo time.

---

## Set up MongoDB

Go to [MongoDB official site](#) and download and install MongoDB on your host machine.

Use `mongod` to start MongoDB server. Once started, you should see something like below:

```
# eric @ Erics-MacBook-Pro-2 in ~/Downloads [20:14:24]
$ mongod
2016-04-12T20:14:25.325-0700 I JOURNAL [initandlisten] journal dir=/data/db/journal
2016-04-12T20:14:25.326-0700 I JOURNAL [initandlisten] recover : no journal files pre
sent, no recovery needed
2016-04-12T20:14:25.342-0700 I JOURNAL [durability] Durability thread started
2016-04-12T20:14:25.342-0700 I CONTROL [initandlisten] MongoDB starting : pid=78049 p
ort=27017 dbpath=/data/db 64-bit host=Erics-MacBook-Pro-2.local
2016-04-12T20:14:25.342-0700 I CONTROL [initandlisten]
2016-04-12T20:14:25.342-0700 I CONTROL [initandlisten] ** WARNING: soft rlimits too l
ow. Number of files is 256, should be at least 1000
2016-04-12T20:14:25.342-0700 I CONTROL [initandlisten] db version v3.0.1
2016-04-12T20:14:25.342-0700 I JOURNAL [journal writer] Journal writer thread started
2016-04-12T20:14:25.342-0700 I CONTROL [initandlisten] git version: nogitversion
2016-04-12T20:14:25.342-0700 I CONTROL [initandlisten] build info: Darwin miniyosemit
e.local 14.1.0 Darwin Kernel Version 14.1.0: Thu Feb 26 19:26:47 PST 2015; root:xnu-27
82.10.73~1/RELEASE_X86_64 x86_64 BOOST_LIB_VERSION=1_49
2016-04-12T20:14:25.342-0700 I CONTROL [initandlisten] allocator: system
2016-04-12T20:14:25.342-0700 I CONTROL [initandlisten] options: {}
2016-04-12T20:14:26.180-0700 I NETWORK [initandlisten] waiting for connections on por
t 27017
```

With above, we will start learning MongoDB by Mongo shell commands for your own debugging. To do this, run `mongo` . This will start your *mongo* shell. You should see something like below:

```
# eric @ Erics-MacBook-Pro-2 in ~/Downloads [20:15:47]
$ mongo
MongoDB shell version: 3.0.1
connecting to: test
Server has startup warnings:
2016-04-12T20:15:46.930-0700 I CONTROL [initandlisten]
2016-04-12T20:15:46.930-0700 I CONTROL [initandlisten] ** WARNING: soft rlimits too low. Number of files is 256, should be at least 1000
>
```

## Commonly used MongoDB shell commands

- `show dbs`
- `use {db name}`
- `show collections`
- `db.{collectionName}.find().pretty()`
- `db.{collectionName}.insert({data})`
- `db.{collectionName}.drop()`

## Example of using commands above

Import example JSON data from <https://raw.githubusercontent.com/mongodb/docs-assets/primer-dataset/primer-dataset.json> as `primer-dataset.json`

You can import data using following command:

```
mongoimport --db test --collection restaurants --drop --file primer-dataset.json
```

Make sure you have `mongod` (mongo instance) running before you run `mongoimport` above

## MongoDB terminology

### MySQL terminology and comparison

MySQL	MongoDB
Database	Database
Table	Collection
Row	Document
Column	Field
Joins	Embedded documents, linking

## Query comparison

### Query Language

Both MySQL and MongoDB have a rich query language. A comprehensive list of statements can be found in the [MongoDB documentation](#).

MySQL	MongoDB
<pre>INSERT INTO users (user_id, age, status) VALUES ("bcd001", 45, "A")</pre>	<pre>db.users.insert({   user_id: "bcd001",   age: 45,   status: "A" })</pre>
<pre>SELECT * FROM users</pre>	<pre>db.users.find()</pre>
<pre>UPDATE users SET status = "C" WHERE age &gt; 25</pre>	<pre>db.users.update(   { age: { \$gt: 25 } },   { \$set: { status: "C" } },   { multi: true } )</pre>

Credit: <https://www.mongodb.com/compare/mongodb-mysql>

## Java MongoDB Driver

Check out the latest changes from this repository, you should have the changes below in your `build.gradle` :

```
dependencies{
  // ...
  compile 'org.mongodb:mongodb-driver:3.2.2'
}
```

Check out `MongoExampleApp.java` and continue reading from there.

## Docker

What is docker?

Docker allows you to package an application with all of its dependencies into a standardized unit for software development.

Why do we need to learn this docker?

In data science, you are likely to use a lot of dependencies that is outside of just Java. In example, you may choose to use MongoDB and other tools to support your own data analysis. You cannot simply get a server and manually SSH into server to install these dependencies one by one.

Thus, docker becomes a great solution for these tools management. In example, we will be using docker in this class to use MongoDB, Elastic Search and further to use Python.

Further, Docker makes it easier to deploy something from your development environment to cloud like Google Cloud Platform or Digital Ocean.

## Install Docker

Please direct yourself to [Docker](#) official site to install docker.

Install docker toolbox if you are using Mac or Windows. If you are using Linux, use your packaging system to install it.

After installation, you should be able to do `docker -v` to see docker version like below:

```
# eric @ Erics-MacBook-Pro in ~/Developments/csula/datascience-spring-2016 on git:mast
er x [16:00:04]
$ docker -v
Docker version 1.11.0, build 4dc5990
# eric @ Erics-MacBook-Pro in ~/Developments/csula/datascience-spring-2016 on git:mast
er x [16:00:04]
$ docker-machine -v
docker-machine version 0.7.0, build a650a40
# eric @ Erics-MacBook-Pro in ~/Developments/csula/datascience-spring-2016 on git:mast
er x [16:00:32]
$ docker-compose -v
docker-compose version 1.7.0, build 0d7bf73
```

In this class, we will be using `docker` , `docker-machine` and `docker-compose` . Make sure you can see the version number above before we move on.

## Using docker to run this repository

### Setting up Docker Machine

Once you have docker-machine installed above, run the following command:

```
# if you don't docker-machine already
# for linux, you don't need docker-machine
$ docker-machine create --driver virtualbox default
$ docker-machine start
```

You should be able to do `docker-machine status` to see status like below:

```
# eric @ Erics-MacBook-Pro in ~/Developments/csula/datascience-spring-2016 on git:mast
er x [16:27:36]
$ docker-machine status
Running
```

Now you have to use `docker-machine env` and run **the last line** to set up your environment variable like below:

```
# eric @ Erics-MacBook-Pro in ~/Developments/csula/datascience-spring-2016 on git:mast
er x [16:31:04]
$ docker-machine env
# ...
# Run this command to configure your shell:
# eval $(docker-machine env)
##### for windows, you have a different command above
```

With above setting to be done, your docker-machine is now ready to run docker!

### Docker Engine

When docker machine is ready, you can run `docker build --rm=true -t big-data .` and you should see something like below:



```
$ docker build -rm=true -t big-data .  
# ...  
--> Running in f40fce6f0d24  
  
--> 9d07afd612cd  
Removing intermediate container f40fce6f0d24  
Successfully built 9d07afd612cd
```

Above command will build up the **docker image** for your docker to run later with command

```
docker run --rm=true big-image .
```

```
$ docker run --rm=true big-data  
Hello Data Science
```

## Docker terminology

- build

An process of building Docker image based on Dockerfile

- Dockerfile

A Dockerfile is a text document that contains all the commands you would normally execute manually in order to build a Docker image. Docker can build images automatically by reading the instructions from a Dockerfile.

You can check out our course repository `Dockerfile` for example

- Image

An Image is an ordered collection of root filesystem changes and the corresponding execution parameters for use within a container runtime. An image typically contains a union of layered filesystems stacked on top of each other. An image does not have state and it never changes.

Use `docker images` to find out more about images we have built so far

- Container

Runtime instance of docker image

Use `docker ps` to find out what is current running

- Compose

With compose, you define a multi-container application in a single file, then spin your application up in a single command which does everything that needs to be done to get it running.

## Why Docker now?

Starting from this point on, we will be using more and more tools in this class like `MongoDB` or `Elastic Search` . It will get harder to maintain all these dependencies on your host machine as well as all your teammates machine.

It would be nicer if we can build our application once and run it everywhere.

Thus, it is great time to introduce docker!

For example, we ran MongoDB earlier with only on our host machine. Lets run the same MongoDB instance on Docker with `docker-compose`

## Git detour

Some students asked about how do you get the latest code from this repository while on the same time pushing code to your own team repository.

Well, one way to do it is through manual copy and paste the code. This way you don't need to learn Git but you have to do it all manually and can be prone to errors.

I'd suggest to use the following approach:

```
# list a remote you have
$ git remote -v

# add the other remote for your local git directory
# lets assume we have course repo url listed above
$ git remote add project {url}

# When you are done adding the remote, check it with
$ git remote -v

# And then you can get the latest course code by
$ git fetch && git pull origin master

# there might be some merge-conflicts happening after pull
# When in doubt, you can checkout course repo side of code only by
# $ git checkout --theirs .

# To push to your team repository
$ git push project {fromBranchName:toBranchName}
```

## Docker Compose

Alright, we have build docker successfully now. What is next?

The utility of docker starts only now. In our project, we probably need to include a couple services like MongoDB or Elastic Search. It's very hard to ask all developers in your team to set up programming environment at all time.

Using docker, we can set up the tools with our application in one command `docker-compose up`

This `docker-compose up` will build and run multiple contains with the images we defined in the `docker-compose.yml`

## Gradle shadow jar

Wait, you meant Gradle can does more things than dependencies management!? What else can Gradle provide to us?

Gradle can also help to build the jar file just like Maven. In this course repository, we are using ShadowJar plugin to build our one-fat-jar.

Once we finish building this one fat jar, our Dockerfile becomes trivial to run the jar file. (See last line of Dockerfile).

For example, if we want to run MongoDB along with our example MongoDB example app, we will need to change the following line to:

```
MongoClient mongoClient = new MongoClient();
```

to

```
MongoClient mongoClient = new MongoClient("db");
```

Why?

We have to tell our application to connect to MongoDB running in different container.

In usual dev ops world, you will need to do your own manual networking. With docker, the networking process will be simplified by the docker-compose.

Now you can change our main class under `build.gradle` line 21 to MongoDB example app and build and run using docker.

## Elastic Search

Elastic search ... you know. For searching.

## Introduction

- Open source search engine
- Built on top of Apache Lucene
- JSON based
- Scheme free
- Distributed
- Multi-tenancy
- API Centric & REST

## What can it do?

- Unstructured & structured search
- Analytics
- Combine

## Get started

- Download [Elastic Search distribution](#)

```
.
├─ LICENSE.txt
├─ NOTICE.txt
├─ README.textile
├─ bin                                # executable scripts
│   └─ elasticsearch
│   └─ plugin
│   └─ ...
├─ config                            # node configuration
│   └─ elasticsearch.yml
│   └─ logging.yml
├─ lib
│   └─ elasticsearch-2.2.0.jar
│   └─ ...
└─ ...
```

- Run `elasticsearch` executable

```
# eric at Erics-MacBook-Pro.local in ~/Downloads/elasticsearch-2.2.0 [22:37:14]
$ ./bin/elasticsearch
[2016-03-02 22:37:22,354][INFO ][node                               ] [NFL Superpro] version[2.2
.0], pid[66651], build[8ff36d1/2016-01-27T13:32:39Z]
[2016-03-02 22:37:22,354][INFO ][node                               ] [NFL Superpro] initializin
g ...
[2016-03-02 22:37:22,937][INFO ][plugins                         ] [NFL Superpro] modules [la
ng-expression, lang-groovy], plugins [], sites []
[2016-03-02 22:37:22,967][INFO ][env                             ] [NFL Superpro] using [1] d
ata paths, mounts [/ (/dev/disk1)], net usable_space [166gb], net total_space [464.7
gb], spins? [unknown], types [hfs]
[2016-03-02 22:37:22,967][INFO ][env                             ] [NFL Superpro] heap size [
989.8mb], compressed ordinary object pointers [true]
[2016-03-02 22:37:24,923][INFO ][node                               ] [NFL Superpro] initialized
[2016-03-02 22:37:24,923][INFO ][node                               ] [NFL Superpro] starting ..
.
[2016-03-02 22:37:25,020][INFO ][transport                       ] [NFL Superpro] publish_add
ress {127.0.0.1:9300}, bound_addresses {[fe80::1]:9300}, {[::1]:9300}, {127.0.0.1:9300
}
[2016-03-02 22:37:25,028][INFO ][discovery                       ] [NFL Superpro] elasticsear
ch/RMR814SUQ-SVatuaP1i31A
[2016-03-02 22:37:28,057][INFO ][cluster.service             ] [NFL Superpro] new_master
{NFL Superpro}{RMR814SUQ-SVatuaP1i31A}{127.0.0.1}{127.0.0.1:9300}, reason: zen-disco-j
oin(elected_as_master, [0] joins received)
[2016-03-02 22:37:28,075][INFO ][http                       ] [NFL Superpro] publish_add
ress {127.0.0.1:9200}, bound_addresses {[fe80::1]:9200}, {[::1]:9200}, {127.0.0.1:9200
}
[2016-03-02 22:37:28,075][INFO ][node                               ] [NFL Superpro] started
[2016-03-02 22:37:28,099][INFO ][gateway                     ] [NFL Superpro] recovered [
0] indices into cluster_state
```

- confirm working by opening in browser under <http://localhost:9200>

```
# eric at Erics-MacBook-Pro.local in ~/Downloads/elasticsearch-2.2.0 [22:38:19]
$ curl http://localhost:9200/_?pretty
{
  "name" : "NFL Superpro",
  "cluster_name" : "elasticsearch",
  "version" : {
    "number" : "2.2.0",
    "build_hash" : "8ff36d139e16f8720f2947ef62c8167a888992fe",
    "build_timestamp" : "2016-01-27T13:32:39Z",
    "build_snapshot" : false,
    "lucene_version" : "5.4.1"
  },
  "tagline" : "You Know, for Search"
}
```

## Debugging in Sense

I suggested you to install sense so that we can walk through the tutorial of Elastic Search without worry of learning curl.

- Install [Kibana](#)
- Run `./bin/kibana plugin --install elastic/sense` under kibana folder to install sense
- Confirm Sense working by [http://localhost:5601/app/sense?  
load\\_from=http://www.elastic.co/guide/en/elasticsearch/guide/current/snippets/010\\_Intro/10\\_Info.json](http://localhost:5601/app/sense?load_from=http://www.elastic.co/guide/en/elasticsearch/guide/current/snippets/010_Intro/10_Info.json)

## Communicate with ES

- [Java Client](#)
- REST

We will focus on the REST to explain Elastic Search briefly first and then implement in Java Client later

## Quick Review on RESTful

- CRUD

```
* POST    => Create
* GET     => Read
* PUT     => Update
* DELETE  => DELETE
```

## ES Terminologies from RDBS point of view

```
Relational DB  => Databases => Tables => Rows      => Columns
Elasticsearch => Indices   => Types  => Documents => Fields
```

## Learn by examples using Sense

### Example of write

```
http://localhost:5601/app/sense?load_from=http://www.elastic.co/guide/en/elasticsearch/guide/current/snippets/010_Intro/25_Index.json
```

### Example of read

```
http://localhost:5601/app/sense?load_from=http://www.elastic.co/guide/en/elasticsearch/guide/current/snippets/010_Intro/30_Get.json
```

## Example of search lite

```
http://localhost:5601/app/sense?load_from=http://www.elastic.co/guide/en/elasticsearch/guide/current/snippets/010_Intro/30_Simple_search.json
```

## More complex search

```
http://localhost:5601/app/sense?load_from=http://www.elastic.co/guide/en/elasticsearch/guide/current/snippets/010_Intro/30_Query_DSL.json
```

## Searching

While many searches will just work out of the box, to use Elasticsearch to its full potential, you need to understand three subjects:

- Mapping  
How the data in each field is interpreted
- Analysis  
How full text is processed to make it searchable
- Query DSL  
The flexible, powerful query language used by Elasticsearch

## Java Elastic Search Client

Go to [Java Client](#) to get started.

# Homework 2

This is your first team project. Please do form a team of 2 to 3 before you start your homework 2!

## Due data

**April 24th midnight**

## Updated requirements after 4/17

Make sure your data satisfies at least 2Vs out of 3Vs of Big Data below:

- Volume

At least **500MB** worth of total data size at this stage

If you successfully meet the requirements we have earlier as **3GB**, you get bonus 2 *points* in this homework

- Velocity

Real time data. Usually you can only accomplish this requirement through API client like Twitter streaming API.

- Variety

At least have **3 data source** to provide data to you (e.g. Twitter, Facebook, CNN News ... etc. -- each of them counts toward one data source)

## Description

In this homework, your job is to acquire data by implementing our data collector interface with your own detail. This may involves you to download data using Http request library, to do web scraping or to make some API calls.

This implementation may vary a lot between all teams depending on the data source you are getting from. For example, a team who wants to get Twitter tweets to compute trending will have different implementation than a team who is getting data from bunch of sensors!



Since this homework is your first **team homework**, you will need to use Github by starting your team repository. This implies you will need to be able to use Git effectively enough to commit and push code to collaborate with your teammate(s) and instructor(s). If you find yourself having trouble of doing git, please utilize [this article](#) for your benefit or play with it on [this link](#)!

This homework will involve you to program your own acquiring process, which is important for you to follow up with your questions from homework 1. You will need to choose the question(s) to work for this team project with your team members. Once your team decided the question(s) from homework 1, please answer the questions below so we know what type of project your team is working on.

### Questions

- What question(s) did you decide to work on as a team?
- What is your data source(s)?
- How long does it take for you to download data? Have you download complete data set?

Please provide a screenshot on your total data size

- How large is your data (size wise and number of records wise)?
- Do you face any dirty data issue? If you do, how did you clean up your data?
- How do you store the data you downloaded?

### Testing

Since we are downloading data from internet, how in the world we suppose to test our process!?

You don't have to test the whole process from end to end. Instead, think about what process can you simplify out to just Java programming and test only that part. In example, you may have a *model* to represent data like below:

```
// import statements
public class Model {
    private final String id;
    private final String content;
    private final String city;
    private final int area;
    private final LocalDateTime timestamp;
    // constructors, getters, setters and other model related methods
}
```

Then my first question to you is to test data validity. Lets say if you read the data online as a String and construct your own data model. How do you handle the case of missing fields? Do you throw exceptions?

All these question can be tested with test case like:

```
// import statements
public class ModelTest {
    @Test(expected=CityNotFoundException.class)
    public void TestCity() {
        String city = null;
        Model.setCity(city);
    }
}
```

In other word, think about all the edge cases and your Java collector implementation detail behavior before you start working on the code. Design an module that has clear defined dependencies so that it is testable.

But how do instructors measure your testing? I will set up the test coverage on each of your repository and see your test coverage number. I expect test coverage to be at least 30%. As long as the test coverage shows higher than 30%, you will get the test coverage points below.

## Tasks

- Create team repo
- Push commits to your own repo
- Implement your data collector
- Acquire data
- Store data in local file storage
- Answer questions in description of Pull Request

## Deliverables

- CSNS submission including team info and link to your repo
- Github pull request
  - Answer questions in the description of pull request
  - Implement data collector interfaces
  - Test your own data collector

## Grading rubric

- Github pull request [1 pt]
- Answer questions in description [2 pts]
- Satisfy at least 2Vs out of 3Vs [2 pts]
- Implement data collector [3 pts]
  - Implements **Collector** interface to mungee data and save data
  - Implements **Source** interface to download/parse data to memory
  - Save data into a file/data storage

You should be able to run this data collector yourself to download data.
- Testing your own data collector [2 pts]
  - Have at least 30% code coverage

P.s.

I will randomly ask a few teams to explain your code to me.

If you cannot explain your code (aka. answer my questions on how you implemented), I will take points off. Or worse, I will count you as copying code from someone else for not able to explain and thus giving you a **zero**.

The random team list will be announced at the class right before the homework submission deadline.

- Eric

# Data Analysis

Now we have a large amount of data. How do we make sense out of this set of data?

## Objectives

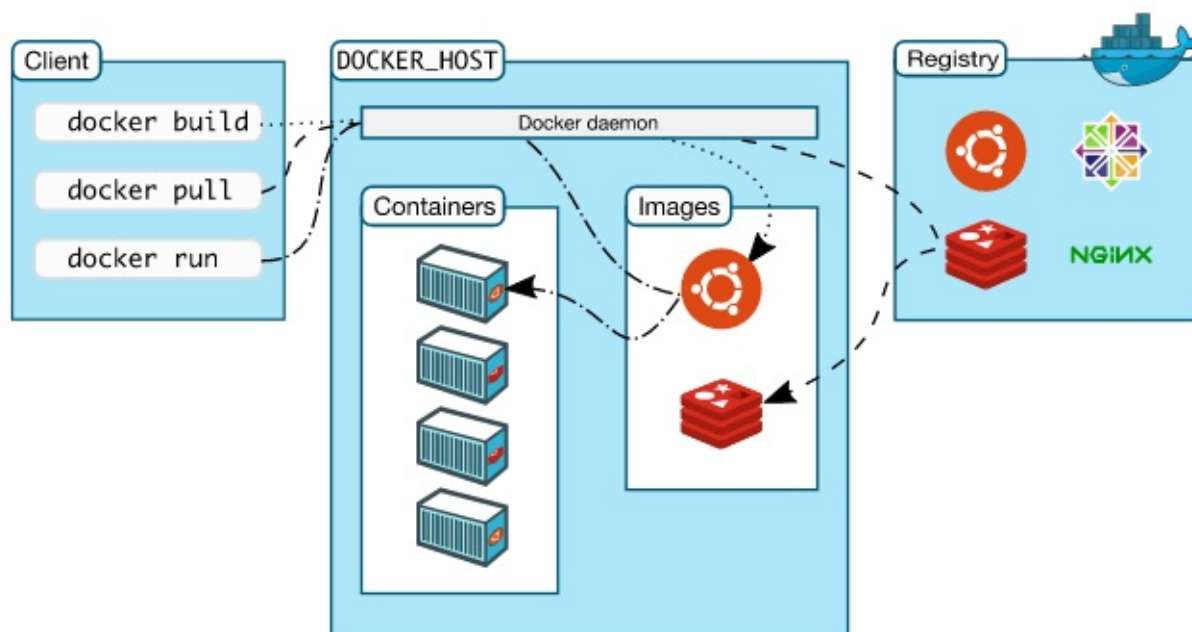
- 5/1
  - Elastic Search Query/Aggregation
- 5/8 -- No class
  - MongoDB Aggregation
    - Sum, count, avg ...
    - Map Reduce
- 5/15, 5/22
  - Python
    - Common math libraries
    - Machine learning
- [Optional knowledge] Hadoop

## Metrics

- Elastic Search query/aggregation
- Kibana visualization
- MongoDB aggregation
- Map reduce concept
- Machine learning algorithms

## Docker Architecture

Due to some questions from students that still don't quite get what Docker is, I want to clarify docker a bit on architecture stand point.



Credit: <https://docs.docker.com/engine/understanding-docker/>

## Exploratory Data Analysis

So far, we have been doing some initial data analysis(EDA) with our data munging process. We tested some of our assumptions on the data shape and clean up accordingly.

Many of you asked me if you data munge process is right. In example, is it right to just clean up the null values? What else should we clean up?

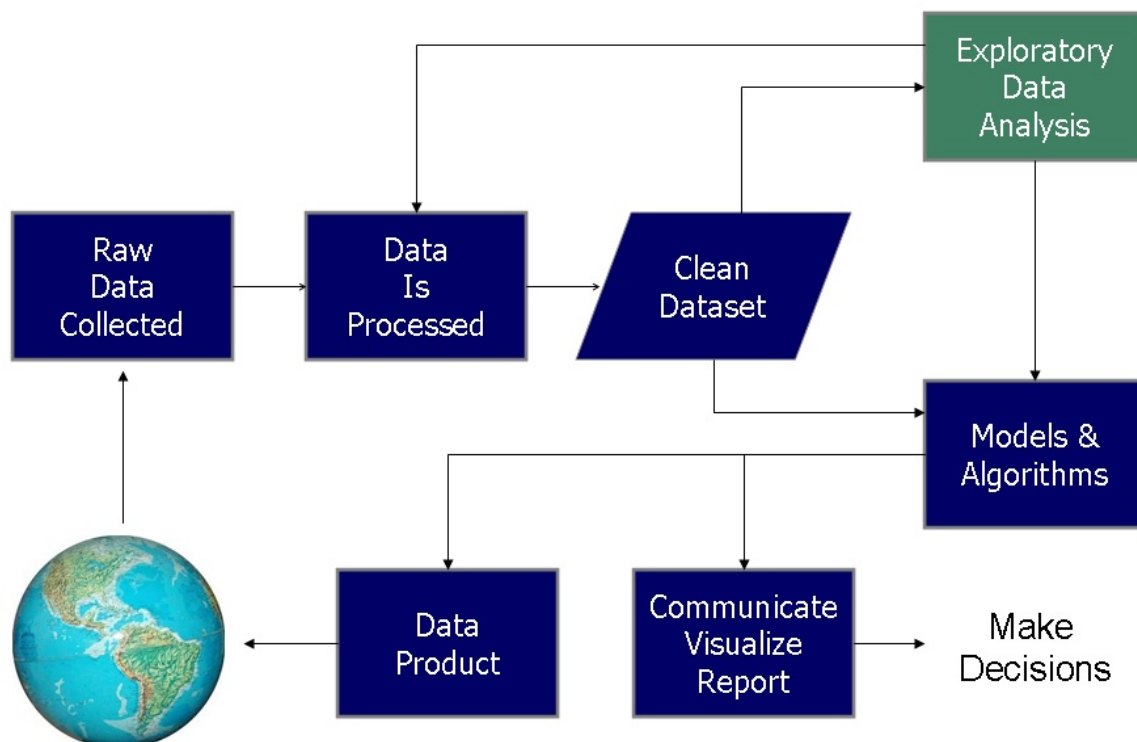
My answer is always "it depends on your question(s) and what you are trying to accomplish".

For instance, some of you are saying you don't need timestamp data because it doesn't fit into your question. So you remove timestamp data out.

This may not be the right move to do just because you think you are may not be using it without going through the process of EDA.

So what is this EDA?

## Data Science Process



credit: [https://en.wikipedia.org/wiki/Exploratory\\_data\\_analysis](https://en.wikipedia.org/wiki/Exploratory_data_analysis)

[EDA] focuses more narrowly on checking assumptions required for model fitting and hypothesis testing, and handling missing values and making transformations of variables as needed. EDA encompasses IDA.

EDA is an examination among data and relationship among variables. EDA is usually the first part of the big analysis process to build the predict model. It is an opportunity to check your assumption about data and to build your intuition about data set.

Many times, business decisions are made by the visualization result from EDA. Other times, you may polish the visualization before you present your finding of data to public.

EDA is about an approach of understanding data using visualization or some statistic model (like what we did in week 1 ... mean, min, max ... etc.).

The objectives of EDA are to:

- Suggest hypotheses about the causes of observed phenomena
- Assess assumptions on which statistical inference will be based
- Support the selection of appropriate statistical tools and techniques
- Provide a basis for further data collection through surveys or experiments

In this course, we will continue with Elastic Search to perform aggregations and use Kibana to do some initial visualization without too much programming behind the scene.

Later on, we will be using MongoDB to learn map reduce concept (again to perform aggregations in scalable way).

Map reduce concept is quite important in a sense that Hadoop is all about map reduce. Even though MongoDB map reduce is different from Hadoop, I think it's a great starting point to see what map reduce can do.

Further, we will also spend about two weeks learning Python and how to use Python to do more analysis or machine learning for better predict model.

John, by next week (May 8th), will talk more in depth about EDA in his video. Stay tuned!

## Elastic search and kibana in depth

[Detail goes here](#)

## Python

### Basic Python

[Detail goes here](#)

### Python with NumPy and Pandas

[Detail goes here](#)

## Machine Learning

[Detail goes here](#)

## Resources/references

- <https://github.com/vinta/awesome-python>
- <https://developers.google.com/edu/python/>
- [Numpy Tutorial](#)
- [Scipy tutorial](#)
- [Pandas in 10 minutes](#)





# Elastic Search Query In Depth

Lets do some searching on the weather data we have earlier from last class.

Note that in the following examples, I'm assuming you all have the Sense with Kibana set up properly.

In other word, these code only work within **Sense**.

We will start by talking about `filtering queries` (structured search)

<https://www.elastic.co/guide/en/elasticsearch/guide/current/structured-search.html>

When working with exact values, you will be working with non-scoring, filtering queries. Filters are important because they are very fast. They do not calculate relevance (avoiding the entire scoring phase) and are easily cached.

```
GET /bd-data/city-temperatures/_search
{
  "query" : {
    "constant_score" : {
      "filter" : {
        "term" : {
          "state" : "Washington"
        }
      }
    }
  }
}
```

This finds nothing!? But we looked back to the csv file ourselves, we are able to find the state "Washington". Why?

Elastic search by default will analyze the string fields for full text searching. For instance, if we type in below:

```
GET /bd-data/city-temperatures/_search?q=Washington
```

And we are able to find some records.

Going back to the source code for Elastic Search example. We have to use specific `matchPhrase` as to `match` to look for "Washington" data. This is due to the same reason -- Elastic Search automatically analyze string field.

How do we turn this analyzer off?

```
# delete the entire index is required because we cannot change existing
# mapping
DELETE /bd-data

# This add up the index mapping before we add the data
PUT /bd-data
{
  "mappings" : {
    "city-temperatures" : {
      "properties" : {
        "state" : {
          "type" : "string",
          "index" : "not_analyzed"
        },
        "date": {
          "type": "date"
        }
      }
    }
  }
}
```

With above mapping defined, we now can go ahead and re-insert data back to this index by running out elastic search example again.

After data now is inserted back to the index, we can check the mapping by:

```
GET /bd-data/city-temperatures/_mapping
```

And now you should be able to perform normal string search as above:

```
GET /bd-data/city-temperatures/_search
{
  "query" : {
    "constant_score" : {
      "filter" : {
        "term" : {
          "state" : "Washington"
        }
      }
    }
  }
}
```

Take away?

1. In order to change mapping, you will have to delete entire index first. Therefore, make sure you have your original set of data somewhere ... like in CSV or in your primary

database (MongoDB).

2. Be aware of string type mapping

## Internal Filter Operationedit

Internally, Elasticsearch is performing several operations when executing a non-scoring query:

1. Find matching docs.

The term query looks up the term XHDK-A-1293-#fJ3 in the inverted index and retrieves the list of documents that contain that term. In this case, only document 1 has the term we are looking for.

2. Build a bitset.

The filter then builds a bitset—an array of 1s and 0s—that describes which documents contain the term. Matching documents receive a 1 bit. In our example, the bitset would be [1,0,0,0]. Internally, this is represented as a "roaring bitmap", which can efficiently encode both sparse and dense sets.

3. Iterate over the bitset(s)

Once the bitsets are generated for each query, Elasticsearch iterates over the bitsets to find the set of matching documents that satisfy all filtering criteria. The order of execution is decided heuristically, but generally the most sparse bitset is iterated on first (since it excludes the largest number of documents).

4. Increment the usage counter.

Elasticsearch can cache non-scoring queries for faster access, but its silly to cache something that is used only rarely. Non-scoring queries are already quite fast due to the inverted index, so we only want to cache queries we know will be used again in the future to prevent resource wastage.

To do this, Elasticsearch tracks the history of query usage on a per-index basis. If a query is used more than a few times in the last 256 queries, it is cached in memory. And when the bitset is cached, caching is omitted on segments that have fewer than 10,000 documents (or less than 3% of the total index size). These small segments tend to disappear quickly anyway and it is a waste to associate a cache with them.

## Combine filters

Recall boolean operation from elasticsearch:

```

{
  "bool" : {
    "must" :    [], // All of these clauses must match. The equivalent of AND.
    "should" :  [], // At least one of these clauses must match. The equivalent of
OR.
    "must_not" : [], // All of these clauses must not match. The equivalent of NOT.
    "filter":   [] // Clauses that must match, but are run in non-scoring, filterin
g mode.
  }
}

```

Learn by example:

```

GET /bd-data/city-temperatures/_search
{
  "query" : {
    "constant_score" : {
      "filter" : {
        "bool" : {
          "should" : [
            { "term" : { "state" : "Washington"}},
            { "term" : { "state" : "California"}}
          ]
        }
      }
    }
  }
}

```

Above is the same as the following sql statement:

```

SELECT *
FROM city-temperatures
WHERE state = "Washington" OR state = "California";
# or
SELECT *
FROM city-temperatures
WHERE state IN ("Washington", "California");

```

Does elasticsearch has something like `IN` sql operator? Sometimes we just need to find multiple exact values.

```
GET /bd-data/city-temperatures/_search
{
  "query" : {
    "constant_score" : {
      "filter" : {
        "terms" : {
          "state" : ["Washington", "California"]
        }
      }
    }
  }
}
```

Consider the below sql query:

```
SELECT *
FROM `city-temperatures`
WHERE averageTemperature BETWEEN 20 AND 40
```

How do we translate this query to elasticsearch query?

```
GET /bd-data/city-temperatures/_search
{
  "query" : {
    "constant_score" : {
      "filter" : {
        "range" : {
          "averageTemperature" : {
            "gte" : 20,
            "lt" : 40
          }
        }
      }
    }
  }
}
```

Cool, that explains the numerical number query. What about date/time? Same idea.

```
GET /bd-data/city-temperatures/_search
{
  "query" : {
    "constant_score" : {
      "filter" : {
        "range" : {
          "date" : {
            "gte" : "2012-01-01",
            "lt" : "2013-01-01"
          }
        }
      }
    }
  }
}
```

## Full text search example

<https://www.elastic.co/guide/en/elasticsearch/guide/current/full-text-search.html>

```
http://localhost:5601/app/sense?load_from=https://www.elastic.co/guide/en/elasticsearch/guide/current/snippets/100_Full_Text_Search/05_Match_query.json
```

Elasticsearch executes the preceding match query as follows:

1. Check the field type.

The title field is a full-text (analyzed) string field, which means that the query string should be analyzed too.

2. Analyze the query string.

The query string QUICK! is passed through the standard analyzer, which results in the single term quick. Because we have just a single term, the match query can be executed as a single low-level term query.

3. Find matching docs.

The term query looks up quick in the inverted index and retrieves the list of documents that contain that term—in this case, documents 1, 2, and 3.

4. Score each doc.

The term query calculates the relevance `_score` for each matching document, by combining the term frequency (how often quick appears in the title field of each document), with the inverse document frequency (how often quick appears in the title

field in all documents in the index), and the length of each field (shorter fields are considered more relevant).

## Aggregation

Quick aggregation types:

<https://www.elastic.co/guide/en/elasticsearch/reference/current/search-aggregations.html>

To master aggregations, you need to understand only two main concepts:

- Buckets

Collections of documents that meet a criterion

- Metrics

Statistics calculated on the documents in a bucket

### Buckets

A bucket is simply a collection of documents that meet certain criteria:

- An employee would land in either the male or female bucket.
- The city of Albany would land in the New York state bucket.
- The date 2014-10-28 would land within the October bucket.

### Metrics

Most metrics are simple mathematical operations (for example, min, mean, max, and sum) that are calculated using the document values. In practical terms, metrics allow you to calculate quantities such as the average salary, or the maximum sale price, or the 95th percentile for query latency.

Aggregation example in sense:

```
http://localhost:5601/app/sense?load_from=https://www.elastic.co/guide/en/elasticsearch/guide/current/snippets/300_Aggregations/20_basic_example.json
```

```
GET /cars/transactions/_search
{
  "size" : 0,
  "aggs" : { # aggregation syntax
    "popular_colors" : {
      "terms" : { # bucket by color
        "field" : "color"
      }
    }
  }
}
```

You can also combine bucket with metrics like below:

```
GET /cars/transactions/_search
{
  "size" : 0,
  "aggs": {
    "colors": {
      "terms": {
        "field": "color"
      },
      "aggs": { # with another aggregation within aggregation!
        "avg_price": {
          "avg": {
            "field": "price"
          }
        }
      }
    }
  }
}
```

Bucket in bucket with metric aggregation!



```
GET /cars/transactions/_search
{
  "size" : 0,
  "aggs": {
    "colors": {
      "terms": {
        "field": "color"
      },
      "aggs": {
        "avg_price": { "avg": { "field": "price" } }
      },
      "make" : {
        "terms" : {
          "field" : "make"
        },
        "aggs" : {
          "min_price" : { "min": { "field": "price" } },
          "max_price" : { "max": { "field": "price" } }
        }
      }
    }
  }
}
```

## Histogram

```
GET /cars/transactions/_search
{
  "size" : 0,
  "aggs":{
    "price":{
      "histogram":{ # using histogram we can build many graph with it (bar, pie or
        histogram!)
        "field": "price",
        "interval": 20000
      },
      "aggs":{
        "revenue": {
          "sum": {
            "field" : "price"
          }
        }
      }
    }
  }
}
```

What about time series variables?

```
http://localhost:5601/app/sense?load_from=https://www.elastic.co/guide/en/elasticsearch/guide/current/snippets/300_Aggregations/35_date_histogram.json
```

Sky's the limit!



Sometimes, you want to scope your aggregation. Such as searching only ford cars rather than searching everything all together.

This case you can do query/filter before doing aggregation.

```
http://localhost:5601/app/sense?load_from=https://www.elastic.co/guide/en/elasticsearch/guide/current/snippets/300_Aggregations/40_scope.json
```

Furthermore, if your data has geo location related data. You may want to consider reading through this document:

<https://www.elastic.co/guide/en/elasticsearch/guide/current/geoloc.html>

## Modeling your data for elasticsearch

Well, elasticsearch provides so many great benefits especially on its performance.

Why don't we use elasticsearch for everything (to replace traditional relational database)?

On one hand, elasticsearch does various things very well. It still has a few points it doesn't scale well.

For instance, handling relationships in elasticsearch is non-trivial.

In short, you want to model your data in elasticsearch or nosql database as flat as possible as opposite to having more relationships in RDBMS.

However, if you still need relationships for your data. Elasticsearch suggests the following approaches:

1. Application side join
2. Data de-normalization
3. Nested objects
4. Parent/child relationships

### **Application side join**

Application side join basically have your applicaiton to simulate joins. Consider the following:

```
PUT /my_index/user/1
{
  "name":      "John Smith",
  "email":     "john@smith.com",
  "dob":       "1970/10/24"
}

PUT /my_index/blogpost/2
{
  "title":     "Relationships",
  "body":      "It's complicated...",
  "user":      1
}
```

Then your application would take blogpost and use user as the id to look for data at the `/my_index/user/ + index`

### **Data de-normalization**

In short, you intentionally store the same data in multiple places like the following:

```
PUT /my_index/user/1
{
  "name":      "John Smith",
  "email":     "john@smith.com",
  "dob":      "1970/10/24"
}

PUT /my_index/blogpost/2
{
  "title":     "Relationships",
  "body":      "It's complicated...",
  "user":      {
    "id":       1,
    "name":     "John Smith"
  }
}
```

This may require you to do locking on your application side in order to update multiple fields at once. I'll be skipping this part of document. If you are interested of learning out more on your own and wanting to apply this knowledge to production. Please look up above on your own.

## Nested objects

```
PUT /my_index/blogpost/1
{
  "title": "Nest eggs",
  "body":  "Making your money work...",
  "tags":  [ "cash", "shares" ],
  "comments": [
    {
      "name":      "John Smith",
      "comment":   "Great article",
      "age":       28,
      "stars":     4,
      "date":      "2014-09-01"
    },
    {
      "name":      "Alice White",
      "comment":   "More like this please",
      "age":       31,
      "stars":     5,
      "date":      "2014-10-22"
    }
  ]
}
```

The primary difference between denormalization and nested objects is array vs objects.

You can still perform the search like `comments.name` (this is why you cannot have `.` in your attribute name).

Please note that if you want to use nested object. Don't rely on the dynamic mapping provided by elastic search. Create the mapping ahead of time before you insert your data.

## Parent/child relationship

Due to the time constraint, we will skip this part of notes.

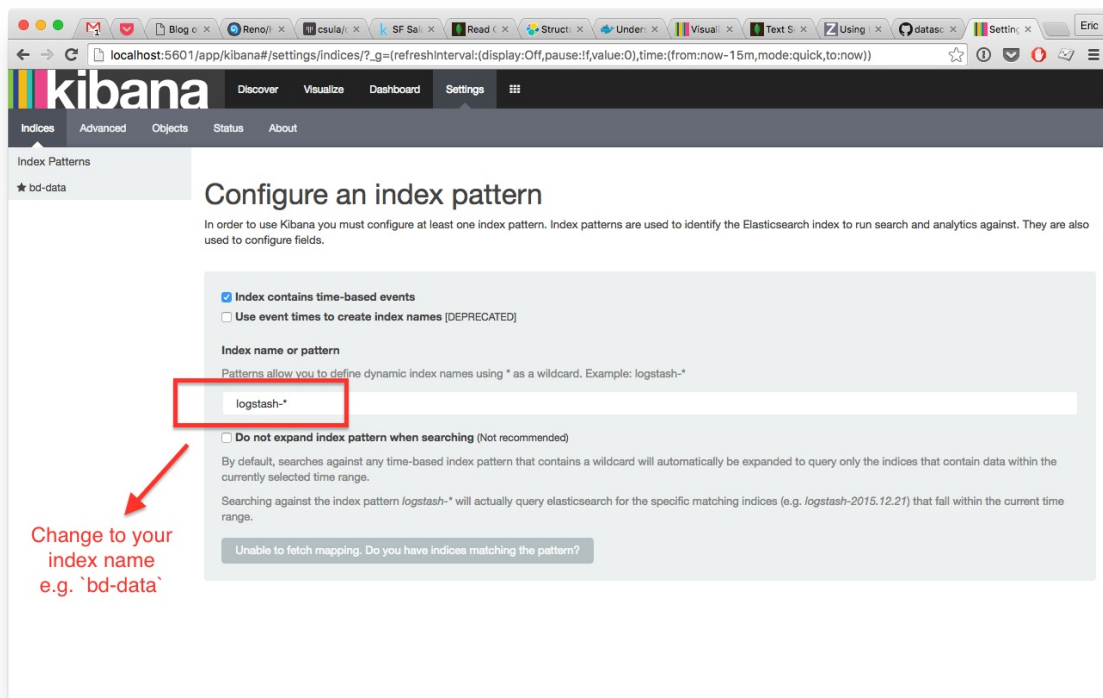
# Kibana in action

## Demo

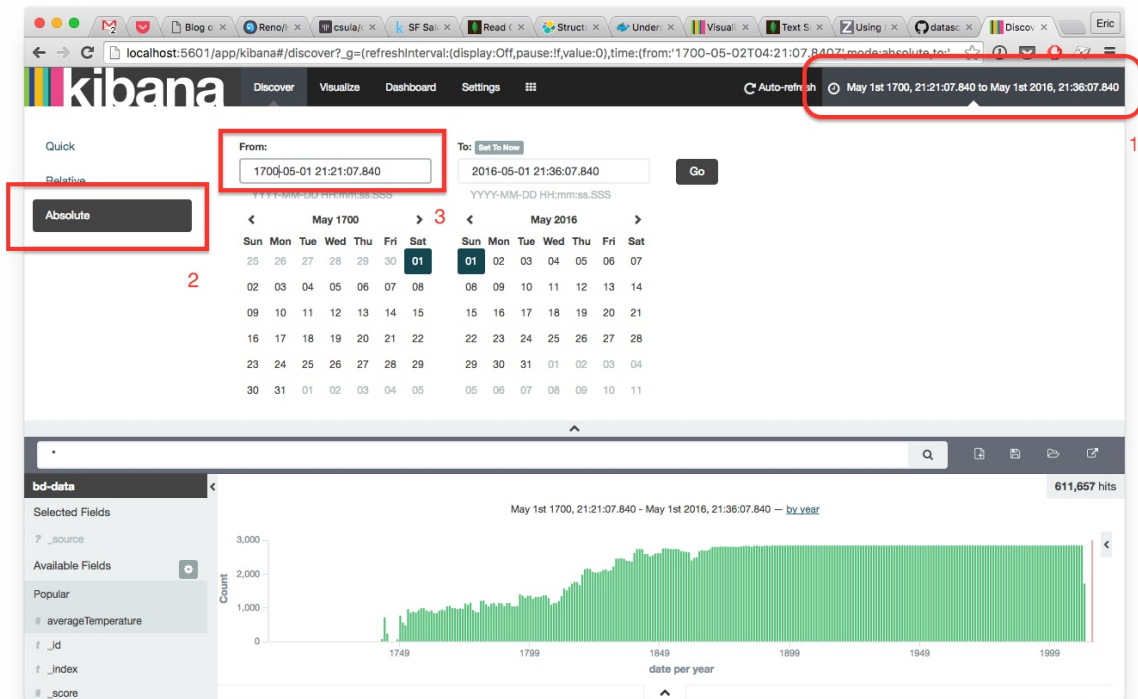
To get Kibana working as what we demo today. Please follow the following guidelines:

Before we started, make sure you have your own unique `cluster.name` set up if you are running elastic search in class with all of us.

1. Start elastic search
2. Start Kibana
3. Run `gradle esExample` to insert data into elastic search
4. Open Kibana



5. Upon above, you want to start by by changing the timestamp



And with above settings, you will be able to start using Discover mode to start searching your data.

For instance, you can type in `state: "Washington"` to search for Washington data. Or you can do `averageTemperature: [10 TO 20]` to do range search. To read more, follow here: <https://www.elastic.co/guide/en/kibana/current/discover.html>

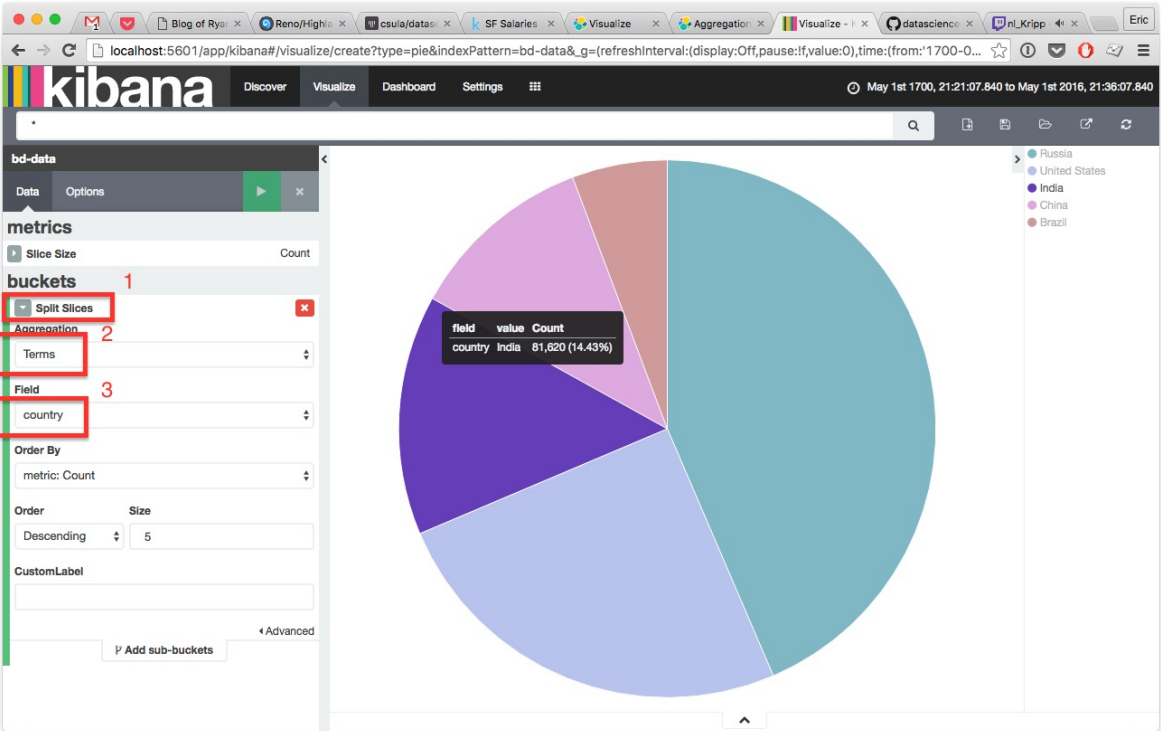
From Discovery mode, you can start to do some initial search to see how data flows through time.

But if you want to start your initial EDA, I suggest you to take a look at the visualization tab.

Make sure you start creating your own visualization to look at your data from different angle and ask yourself hard question.

To create visualization, you will need to think in aggregation. If you are not sure how to create visualization or which graph to use, I suggest you to review through the aggregations above.

For example, you can start by creating a pie chart with aggregation of terms like below:



# Python

Please install [Python 2](#). Please just download 2.7 for this class not version 3.

For Mac user, use `brew install python`

Python, one of the top two common languages choice for data science. Why?

Many libraries are written in python. This includes machine learning libraries like [Sci-kit](#), [NumPy](#) or [Pandas](#)

So why is Python popular?

- Easy to learn
- Easy to get it up and running
- Syntax is very close to pseudocode
- Interpreted languages (have interpreter built in)

In the following session, we will introduce python from the ground up.

## Syntax

Not like Java, python doesn't have bracket. Python uses **indentation** to detect code block.

What do I mean by indentation? Consider the following:

```
x = 1

if x > 0:
    print "x is positive"
```

Now when the print statement is not indented properly. it will not be executed in the if statement.

Therefore, it is recommended for you to display all the invisible characters in your favorite text editor.

Don't ask why your code doesn't work later because your code contain both hard tab and soft tab (4 spaces or 2 spaces)

## Variables and types

In python, to define variable, you don't need to specify types not like Java.



In other word, you can simply assign variable just like below:

```
# Number
x = 1 + 1
# Boolean (keep it in mind that first character is capitalized)
t = True
# String
y = "Hello, world"
z = "Hello\nworld!"
multiline_string = """
this is multiline string
if you need to print out multiline
you may use three "
"""
concat_str = "Hello" + "," + "world!"
# You can treat word like list and get character(s) out this way
y[1] # e
y[0:2] # He

# List
li = [1, 2, 3]
li[0] # 1
li[:] # return shallow copy of the list
li[1] = 4 # list is now [1, 4, 3]
li.append(5) # list is now [1, 4, 3, 5]
li[2:3] = [] # list is now [1, 4, 5]
len(li) # 3

# nested list
li = [[1, 2], [2, 3], [3, 4]]
```

## Control flow

Control flows includes if, for, while and function.

```
# if statement doesn't need to be surrounded by parathesis
x = 1

if x > 0:
    print "x is positive"

# for
li = [1, 2, 3]
for i in li:
    print i
# range
for i in range(10):
    print i # 0 ... 9
# range can also be specific range with lower-bound(inclusive) to upper-bound(exclusive)
for i in range(5, 10):
    print i # 5, 6, 7, 8, 9
# break and continue
for i in range(100):
    if i > 10:
        break
    elif i % 2 == 0:
        continue
    else:
        print i
```

```
# def is the keyword to define a function
def f(x, y):
    return x + y

f(1, 2) # 3

# python not like java can have default argument value
def g(x, y=1):
    return x + y

g(1) #2

# when calling function, it can provide keyword as argument
def h(x, y, z):
    print x
    print y
    print z
    return x + y - z

h(y=2, x=6, z=10)

# lambda
def l(x):
    return lambda n: n + x

f = l(42)
f(8) # 50

# documentation for function
def messy(z):
    """
    this is a documentation example
    """
    return z + 42

print messy.__doc__
```

## Data structure

### List

- `list.append(x)`
  - Add an item to the end of list
- `list.extend(anotherList)`
  - Extend list from another list
- `list.insert(i, x)`
  - Insert item x into position i e.g. insert to front of list `list.insert(0, item)`
- `list.remove(x)`

- remove the **first** item of value x
- list.pop([i])
  - pop certain item from the list of index i. If no argument is given, pop the last item
- list.index(x)
  - return index of item x
- list.sort()
- list.reverse()

### List as stack

```
stack = [3, 4, 5]
stack.append(6)
stack.append(7)
stack
[3, 4, 5, 6, 7]
stack.pop()
7
stack
[3, 4, 5, 6]
stack.pop()
6
stack.pop()
5
stack
[3, 4]
```

### List as queue

```
from collections import deque
queue = deque(["Eric", "John", "Michael"])
queue.append("Terry")           # Terry arrives
queue.append("Graham")         # Graham arrives
queue.popleft()                # The first to arrive now leaves
'Eric'
queue.popleft()                # The second to arrive now leaves
'John'
queue                          # Remaining queue in order of arrival
deque(['Michael', 'Terry', 'Graham'])
```

### Functional programming with list like filter, map and reduce

- filter

```
def f(x): return x % 3 == 0 or x % 5 == 0
...
filter(f, range(2, 25))
[3, 5, 6, 9, 10, 12, 15, 18, 20, 21, 24]
```

- map

```
def cube(x): return x*x*x
...
map(cube, range(1, 11))
[1, 8, 27, 64, 125, 216, 343, 512, 729, 1000]
```

- reduce

```
seq = range(8)
def add(x, y): return x+y
...
map(add, seq, seq)
[0, 2, 4, 6, 8, 10, 12, 14]
```

## Tuples

We saw that lists and strings have many common properties, such as indexing and slicing operations. They are two examples of sequence data types (see Sequence Types — str, unicode, list, tuple, bytearray, buffer, xrange). Since Python is an evolving language, other sequence data types may be added. There is also another standard sequence data type: the tuple.

A tuple consists of a number of values separated by commas, for instance:

```
>>>
t = 12345, 54321, 'hello!'
t[0]
12345
t
(12345, 54321, 'hello!')
# Tuples may be nested:
... u = t, (1, 2, 3, 4, 5)
u
((12345, 54321, 'hello!'), (1, 2, 3, 4, 5))
# Tuples are immutable:
... t[0] = 88888
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
# but they can contain mutable objects:
... v = ([1, 2, 3], [3, 2, 1])
v
([1, 2, 3], [3, 2, 1])
```

## Set

Similar to list but with keyword `set`

```
li = [1, 2, 3, 1]
se = set(li) # set([1, 2, 3])
1 in se # True
4 in se # False

# operation on sets
li2 = set([2, 3, 4])
# in se but not in li2
se - li2 # 1

# union
se | li2 # 1234

# interception
se & li2

# in a or b but not both
se ^ li2
```

## Dictionary

Dictionary is like map in Java.

```
tel = {'jack': 4098, 'sape': 4139}
tel['guido'] = 4127
tel

tel['jack']

del tel['sape']
tel['irv'] = 4127
tel

tel.keys()

'guido' in tel

# to loop through
for i, v in enumerate(['tic', 'tac', 'toe']):
    print i, v
```

## Modules

Each python file is essentially a module.

For instance,

```
# my_math.py

def add(x, y):
    return x + y
```

```
import my_math

my_math.add(1, 2)
```

### Python main method

if you have a if statement like below, that block of code will not be executed when you import to other module.

```
if __name__ == "__main__":
    import sys
    print sys.argv[1]
```

### Packages

When your code base grows larger, you will need to think about how to organize your code.

Python will treat folder with `__init__.py` as a package. Think of below:

```
main.py
acquisition/
    __init__.py
    collector.py
    source.py
storage/
    __init__.py
    mongo.py
```

Then in your `main.py` you can start import like below:

```
# main.py
import acquisition.collector
import acquisition.source

while source.hasNext():
    collector.save(collector.munge(source.next()))
```

### File read/write

```
f = open('file_name.extension', 'r') # second argument is for mode read or write or both!

f.read() # reads out the entire content
f.readline() # reads line by line

# below also works
for line in f:
    print line

f.write('Hello world!\n') # to write to file
```

## json

Json is built-in with Python!

```
# write json string
json.dumps([1, 'simple', 'list'])

# dump json to a file
json.dumps([1, 'simple', 'list'], f)

# read json string from file
json.load(f)
```

## Exceptions

```
while True:
    try:
        x = int(raw_input("Please enter a number: "))
        break
    except ValueError:
        print "Oops! That was no valid number. Try again..."
```

## Classes

```
class MyClass: # can defined inheritance MyClass(ParentClass, ParentClass2)
    """A simple example class"""
    i = 12345

    def __init__(self):
        print 'no arg constructor is called'

    def f(self): # first argument is always self
        return 'hello world'
```



## CSV

```
import csv

with open('example.csv', 'rb') as csvfile:
    spamreader = csv.reader(csvfile)
    for row in spamreader:
        print ', '.join(row)
```

## Package manager -- pip

<https://pip.pypa.io/en/stable/installing/>

## Numpy

Provides various utilities functions on top of N-dimensional array such statistical calculation (.mean(), .std()) and vectorized operation.

## Pandas

Provides high-performance, easy to use data structures and data analytics tools in Python.

## One dimensional data structure comparison

Python	NumPy	Pandas
List	Array	Series
Simple	More features	Even more features
Access by index, range of indexes, use loop	each element has same data type, vectorized operation, .mean() and .std()	.describe()

## NumPy examples

```
>>> from numpy import *
>>> a = arange(15).reshape(3, 5)
>>> a
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14]])
>>> a.shape
(3, 5)
>>> a.ndim
2
>>> a.dtype.name
'int32'
>>> a.itemsize
4
>>> a.size
15
>>> type(a)
numpy.ndarray
>>> b = array([6, 7, 8])
>>> b
array([6, 7, 8])
>>> type(b)
numpy.ndarray
```

## Array creation

```
>>> from numpy import *
>>> a = array( [2,3,4] )
>>> a
array([2, 3, 4])
>>> a.dtype
dtype('int32')
>>> b = array([1.2, 3.5, 5.1])
>>> b.dtype
dtype('float64')
```

## Array operations

```
>>> a = array( [20,30,40,50] )
>>> b = arange( 4 )
>>> b
array([0, 1, 2, 3])
>>> c = a-b
>>> c
array([20, 29, 38, 47])
>>> b**2
array([0, 1, 4, 9])
>>> 10*sin(a)
array([ 9.12945251, -9.88031624,  7.4511316 , -2.62374854])
>>> a<35
array([True, True, False, False], dtype=bool)
```

## Linear algebra review:

Review [https://en.wikipedia.org/wiki/Matrix\\_\(mathematics\)](https://en.wikipedia.org/wiki/Matrix_(mathematics)) for vectorized operations.

```
>>> A = array( [[1,1],
...            [0,1]] )
>>> B = array( [[2,0],
...            [3,4]] )
>>> A*B                                     # elementwise product
array([[2, 0],
       [0, 4]])
>>> dot(A,B)                               # matrix product
array([[5, 4],
       [3, 4]])
```

```
>>> a = ones((2,3), dtype=int)
>>> b = random.random((2,3))
>>> a *= 3
>>> a
array([[3, 3, 3],
       [3, 3, 3]])
>>> b += a
>>> b
array([[ 3.69092703,  3.8324276 ,  3.0114541 ],
       [ 3.18679111,  3.3039349 ,  3.37600289]])
>>> a += b                                     # b is converted to integer type
>>> a
array([[6, 6, 6],
       [6, 6, 6]])
```

```
>>> a = random.random((2,3))
>>> a
array([[ 0.6903007 ,  0.39168346,  0.16524769],
       [ 0.48819875,  0.77188505,  0.94792155]])
>>> a.sum()
3.4552372100521485
>>> a.min()
0.16524768654743593
>>> a.max()
0.9479215542670073
```

## Show histogram

```
import numpy
import pylab
# Build a vector of 10000 normal deviates with variance 0.5^2 and mean 2
mu, sigma = 2, 0.5
v = numpy.random.normal(mu, sigma, 10000)
# Plot a normalized histogram with 50 bins
pylab.hist(v, bins=50, normed=1)           # matplotlib version (plot)
pylab.show()
# Compute the histogram with numpy and then plot it
(n, bins) = numpy.histogram(v, bins=50, normed=True) # NumPy version (no plot)
pylab.plot(.5*(bins[1:]+bins[:-1]), n)
pylab.show()
```

## Pandas

Series = 1 dimensional data DataFrame = N dimensional data

Example

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Create series
s = pd.Series([1,3,5,np.nan,6,8])
dates = pd.date_range('20130101', periods=6)

# Create DataFrame
df = pd.DataFrame(np.random.randn(6,4), index=dates, columns=list('ABCD'))
df2 = pd.DataFrame({ 'A' : 1.,
                      'B' : pd.Timestamp('20130102'),
                      'C' : pd.Series(1,index=list(range(4)),dtype='float32'),
                      'D' : np.array([3] * 4,dtype='int32'),
                      'E' : pd.Categorical(["test","train","test","train"]),
                      'F' : 'foo' })

# you can check type here
df2.dtypes

# Viewing data
df.head()
df.tail(3)

# showing indexes names
df.index

# quick statistics
df.describe()

# sorting by indexes
df.sort_index(axis=1, ascending=False)

# or by values
df.sort_values(by='B')

# select by position
df.iloc[3]

# use boolean to filter out rows
df[df.A > 0]

# Missing data? Pandas has way to deal with it
df1 = df.reindex(index=dates[0:4], columns=list(df.columns) + ['E'])
df1.loc[dates[0]:dates[1], 'E'] = 1
df1.dropna(how='any') # drop any rows having NaN in value
df1.fillna(value=5)  # or fill with default value for NaN

# Stats
df.mean() # for the entire two dimensional data
df.mean(1) # or just one column
```

```

# apply() method is the same as map method,
# you transform each element in DataFrame by calling apply()
df.apply(np.cumsum)

# histogram
data = np.random.randint(0, 7, size=50)
s = pd.Series(data)
s.value_counts()

# merge two DataFrames
df = pd.DataFrame(np.random.randn(10, 4))

pieces = [df[:3], df[3:7], df[7:]]

# can even join
left = pd.DataFrame({'key': ['foo', 'foo'], 'lval': [1, 2]})
right = pd.DataFrame({'key': ['foo', 'foo'], 'rval': [4, 5]})
pd.merge(left, right, on='key')

# group by? Panda got you
df = pd.DataFrame({'A' : ['foo', 'bar', 'foo', 'bar',
                          'foo', 'bar', 'foo', 'foo'],
                  'B' : ['one', 'one', 'two', 'three',
                          'two', 'two', 'one', 'three'],
                  'C' : np.random.randn(8),
                  'D' : np.random.randn(8)})

df.groupby('A').sum()
df.groupby(['A', 'B']).sum()

# Time series data
rng = pd.date_range('1/1/2012', periods=100, freq='S')
ts = pd.Series(np.random.randint(0, 500, len(rng)), index=rng)
ts.resample('5Min').sum()
ts.head()

# Change frequency of time series data
converted = ts.asfreq('45Min', method='pad')
converted.head()
# get mean by day
ts.resample('D').mean()

# plotting graph
ts = pd.Series(np.random.randn(1000), index=pd.date_range('1/1/2000', periods=1000))
ts = ts.cumsum()
ts.plot()

df = pd.DataFrame(np.random.randn(1000, 4), index=ts.index,
                  columns=['A', 'B', 'C', 'D'])
df = df.cumsum()
plt.figure(); df.plot(); plt.legend(loc='best')

# i/o from/to csv

```

```
df.to_csv('foo.csv')  
pd.read_csv('foo.csv')
```

# Intro to Machine learning

Please note this class is not intended to learn Machine Learning in detail. If you want to learn more on Machine Learning, please take the appropriate class like CS-461.

Above disclaimer doesn't mean we will not talk about Machine Learning at all. Instead, we will cover the Machine Learning algorithms like blackbox and use them as what they are intended for.

In short, if you want to learn the implementation for the Machine Learning, go to CS-461!

## Problems to solve

So what kind of problems does machine learning algorithms solve?

In general, a learning problem considers a set of  $n$  samples of data and then tries to predict properties of unknown data. If each sample is more than a single number and, for instance, a multi-dimensional entry (aka multivariate data), is it said to have several attributes or features.

from <http://scikit-learn.org/stable/tutorial/basic/tutorial.html>

In short, you have a set of training data and you train your algorithms with them. After training your algorithm, you can feed some unknown data (which is not presented in the training data set) to predict the outcome.

## Decisions on algorithms

I found [this cheat sheet](#) to be very effective to select an algorithms on your problem. Please utilize it!

Cheat sheet:

[http://scikit-learn.org/stable/tutorial/machine\\_learning\\_map/](http://scikit-learn.org/stable/tutorial/machine_learning_map/)

Briefly speaking in selection algorithms, you want to start by highest level with below two categories: *supervised* and *unsupervised* learning.

Supervised learning indicates your training set of data has a expected labels as outcome which we want to predict.

Within supervised learning algorithms, you will face another decision on either *classification* or *regression*.



**Classification** means you want to predict samples into two or more classes. Such as handwritten digit recognition. You can think of this as discrete form of supervised learning where one has a limited number of categories and for each of the  $n$  samples provided, one is to try to label them with the correct category or class.

**Regression** if the desired outcome contains one or more continuous variables. An example of a regression problem would be the prediction of the length of a salmon as a function of its age and weight. (as you cannot discrete limit the number of length)

**Unsupervised** learning usually involves grouping similar data together like *k-nearest neighbors*.

## Training data set and testing data set

When feeding algorithms data, you want to split your data into training set and testing data set. They should be different so that your algorithms don't get overfitting into certain outcome.

But how much data should I provide as training set and test data set?

This is where the art of machine learning engineer lives, you have to play with experiments and figure out the proper number.

Briefly speaking, you can start 10% of data as testing set and test your accuracy accordingly.

## Brief workflow for sklearn

In this class, we will be using Python sklearn(scikit) for all Machine Learning algorithms. Please get used to their official site and their documentation -- <http://scikit-learn.org/stable/index.html>

Without going through the detail of implementation, your sklearn code will mostly look like below:

```
# get the training set of data and test set of data
from sklearn import datasets
# luckily sklearn provides some sample data to start with
iris = datasets.load_iris()
digits = datasets.load_digits()

# import sklearn libraries
from sklearn import svm # for example

# instantiate your algorithm
clf = svm.SVC(gamma=0.001, C=100.)

# learn by `fit` or train your algorithm by providing features_test and features_labels
clf.fit(digits.data[:-1], digits.target[:-1])

clf.predict(digits.data[-1:])

# to measure how good your algorithms are, you can measure accuracy by below:
clf.score(features_test, labels_test)

# what about you don't want to relearn your algorithm everytime you run it?
# you can save your algorithm using `pickle`
import pickle
s = pickle.dumps(clf)
# and load it up afterward
clf2 = pickle.loads(s)
```

## Agenda of the day

Without further due, lets get into a few selected algorithms.

In this class, we will talk very briefly on some algorithms that I think it's useful for your problem sets.

- Supervised learning
  - Naive Bayes algorithm
  - SVM (Support Vector Machine)
  - Decision tree
- Unsupervised learning
  - k-means
  - SVM (yes, this can also be used for unsupervised)

## Naive Bayes

[http://scikit-learn.org/stable/modules/naive\\_bayes.html](http://scikit-learn.org/stable/modules/naive_bayes.html)

Using Bayes rule to predict outcome.

It assumes independence between every pair of features.

You can start using this by using [http://scikit-learn.org/stable/modules/generated/sklearn.naive\\_bayes.GaussianNB.html#sklearn.naive\\_bayes.GaussianNB](http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html#sklearn.naive_bayes.GaussianNB)

#### *Strength*

- Very easy to implement
- Supported in almost all languages
- Very simple to understand

#### *Weakness*

- Can break if the features have dependencies

## **SVM (Support Vector Machine)**

<http://scikit-learn.org/stable/modules/svm.html>

Basically finding a line to maximize the margin (distance to the nearest points). This sometimes implies ignoring outliers.

#### *Strength*

- Works very well when there is clear set of boundary even in complex situation

#### *Weakness*

- Training time can be slow when the data set is large
- When there is a lot of noise(outliers), this can overfitting to the data.

## **Decision tree**

<http://scikit-learn.org/stable/modules/tree.html>

In nutshell, you can think of decision tree to generate a list of if else statements. And predict function will run through the series of if else statement to get the outcome.

In more theoretical way of speaking, decision tree algorithm maximize the information gain (entropy of parent - weighted average of children).

#### *Strength*

- Very easy to use
- Can be very easy to understand as outcome

### *Weakness*

- Can be overfitting sometimes

## **k-means**

<http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html>

You can think of k-means to split data set into k groups by grouping near data together.

Your argument will be k -- number of clusters.

It's the most common algorithm used for clustering unknown set of data.

# Homework 3

## Due Date

May 29th Midnight.

## Description

Exploratory Data Analysis (EDA). So what is EDA? Recall from the [data\\_analysis.md](#) earlier:

The objectives of EDA are to:

- Suggest hypotheses about the causes of observed phenomena
- Assess assumptions on which statistical inference will be based
- Support the selection of appropriate statistical tools and techniques
- Provide a basis for further data collection through surveys or experiments

Your job is to create a presentation containing graph about your data. Your presentation should answer each of the above.

## Tasks

In this homework, your job is to create your initial presentation on your EDA. The deliverable for this homework is to create a presentation and submit your link on CSNS.

In order to do your presentation, your team will need to:

1. Aggregate data
2. Generate graph
3. Do statistical analysis based on your aggregation

A hint about creating these graph and aggregation:

You can utilize Elastic Search and Kibana to generate such graphs. To do so, you will need to import data into Elastic Search first. Please review the `ElasticSearchExampleApp.java`.

Using Elastic Search implies your tasks would be like below:

1. Import your data into Elastic Search

When you import data, please make sure your data has timestamp or any time related data first.

2. Check if your mapping is correct by `_mapping`
3. If not correct, you should be deleting your entire set of data and generate your mapping first.

If you do change your mapping, please submit your mapping as part of description in Pull Request

4. Start up Kibana and start visualizing with your data set

From this moment, you should be thinking about what type of graph you should be using. For instance, bar chart, pie chart, line chart ... etc.

Once you are satisfied with your visualization. Save it. You will need to use these later

5. Create dashboard containing all your visualization

Make sure you have those visualization saved from 4th step

**You cannot use technology other than Kibana and Elastic Search for your programming part. This policy is to ensure that grading is consistent across the teams. Deviation from this requirement will not be accepted.**

## Presentation Guideline

In order to preserve valuable lecture time, your team will need to create an online screencast using youtube, vimeo, or any other video distribution services. Your video must be visible by a non-registered user. However, it does not have to be listed in a public directory.

Here is the guideline for the presentation:

- The screencast video must not require special pluggins (we recommend uploading to youtube)
- You **cannot** email or attach a video file, e.g. `.mov` , `.avi` , `.mpeg` etc.
- Your presentation must be between 1:40 and 2:00. Videos with duration > 2:00 or < 1:40 will not be accepted
- Your presentation must be visually clear and have good audio; we recommend that you record at HD level

Please note that the above requirements are **not** negotiable.

Some recommendations:

- Avoid verbal pauses (ah and umm's should be avoided) -- when in doubt, practice. You

should script out what you are going to say ahead of time

- The whole idea of "winging it" is not recommended
- Avoid reading code or talking too low-level details
- We recommend the following presentation structure:
  - Title slide (presentation title and team members) -- introduce yourself and your team
  - Agenda slide, `here is our agenda...`
  - Background slide(s)
    - State your hypotheses
    - State your assumptions
  - Approach and methods
    - Talk about your data
    - Tools you use
  - Findings, give a one or two sentence punchline
- If you have live or animated data, recommend that you use keyboard shortcut to switch between slides and demo.

Some thoughts:

- It is estimated that, if this the first time doing an online presentation, you'd need to spend at least 1 hour to do the presentation justice.
- You will find yourself recording about 4 to 5 times before getting it perfect.
- You should definitely watch your own video from start to finish. Anything that does not sound right or not clear, you should re-do.

## Deliverables

- CSNS submission containing the following:
  - list of team members
  - presentation title
  - link your presentation slides
  - link to the video screencast of your presentation
- Answer each statement above
- Pull request containing all the coding changes you have made

## Grading rubric

- CSNS submission [1 pts]
- Answer each statement in objectives of EDA above [2 pts]
- Presentation [7 pts]





# Data Visualization

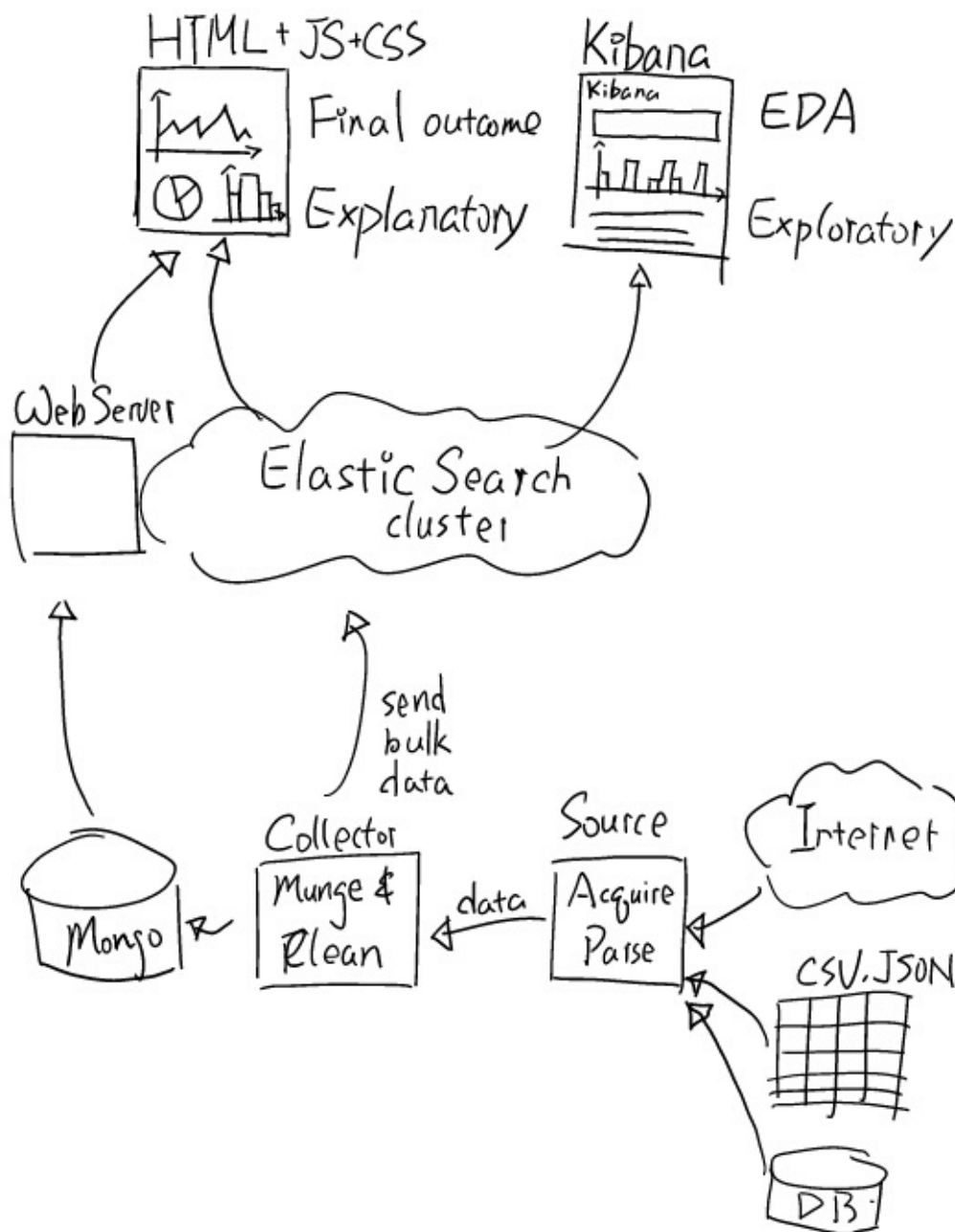
We've got some conclusions out of the data. How do we report this data to others?

## Architecture of the project

At this point, we are at the end of this lecture.

Lets spend some time to talk about out accomplishments so far:

We have built a data science project that is architected below:



1. We've built the *Source* to acquire and parse the data from internet or downloaded file
2. We've built the *Collector* to munge and clean the data
3. Also in *Collector* we send the data to our storage (primary to ElasticSearch)
4. Using ElasticSearch & Kibana, we do our EDA (Exploratory Data Analysis)
5. Using formal web graphing libraries to refine and polish our report in more of formal report

From here, we will learn to utilize some graphing libraries like D3 to refined our visualization earlier.

A quick disclaimer: this architecture is not the only architecture to do the data science project. In fact, there are many ways to do the data science project.

Moreover, you can replace any part in this architecture with a different technique.

In this class, we picked Elasticsearch as the central because it is the tool I think fits the best in our timeline.

In other word, for your future purpose. Please review the data science procedure and use that as a base instead of strictly following the architecture of this class project.

Be creative on the tools you choose and pick the right tool for the right need!

## What is visualization?

A picture worths a thousand words.

Visualization is turning numbers (data) into pictures and into stories.

But how to define a good visualization vs bad ones?

Data visualization can be briefly list as exploratory vs explanatory.

### Exploratory

In earlier lecture, we learned how to do exploratory data analysis with Kibana by making a bunch of graphs.

This, so to speak, exploratory visualization. You are playing with different types of graphs in order to figure the best visualization to represent your data.

Using metaphor to explain -- you are finding a perfect rounded rock among a thousand rocks.

### Explanatory

A good explanatory visualization should cover the following five qualities:

1. Robust understanding of the context
2. Choosing appropriate type of visualization
3. Eliminating those things that are not aiding to the goal
4. Drawing audience's attention using color, size and other design aspect

## 5. Story

In short, your explanatory visualization should tell a story in a straight forward way.

Continued from the metaphor above, you are about to present this perfect rounded rock to your audiences.

Lets give an example -- think about the visualization like notes of lecture. Exploratory is like your notes to yourself while the explanatory is like notes from you to your audience (like my notes to all of students).

In other word, EDA is between you and your data and explanatory is between data directly to audience.

# Design process of creating data visualization

<http://vizwiz.blogspot.com/2013/01/alberto-cairo-three-steps-to-become.html>

## Art & Science

Data visualization is not just science, it is combining both art and science.

A good visualization will contain design elements inside like color, size and story telling elements.

In other word, you will need to have the ability to code, design as well as the ability to communicate.

I know that us as Computer Scientist; we really don't have a background of design nor the story telling. However, I'd love all of you to try your best for these two elements as part of your visualization.

Give it a try to create your best visualization and see what kind of challenges you are facing.

In future, you may be working with others (which may cover your weakness) to create a fined graph!

In short, don't just treat data science as only software engineering. It is much larger than just coding!

We will try out best to covert both the coding and the design part of the visualization here.

## Example of good data visualization

Above talks about a few principles of good visualization.

Can you give me example on a good visualization?

I think video by Hans Rosling best represents all three parts of good visualization (design, coding and story telling).

<https://www.youtube.com/watch?v=jbkSRLYSojo>

## Data Types

At this phase of project, we have a lot of data; and we category these data into a few different data types.

These categories of data types not only helps us to model better but also helpful toward the visualization aspect.

Think about the Kibana graph, you have to tell Elasticsearch ahead of time on what *mapping* (data type) a column is even before you send the data into Elasticsearch.

An example of this is you will have to tell Elasticsearch this data is for geopoint rather than just two numbers (lat and long) in order to map these points on map!

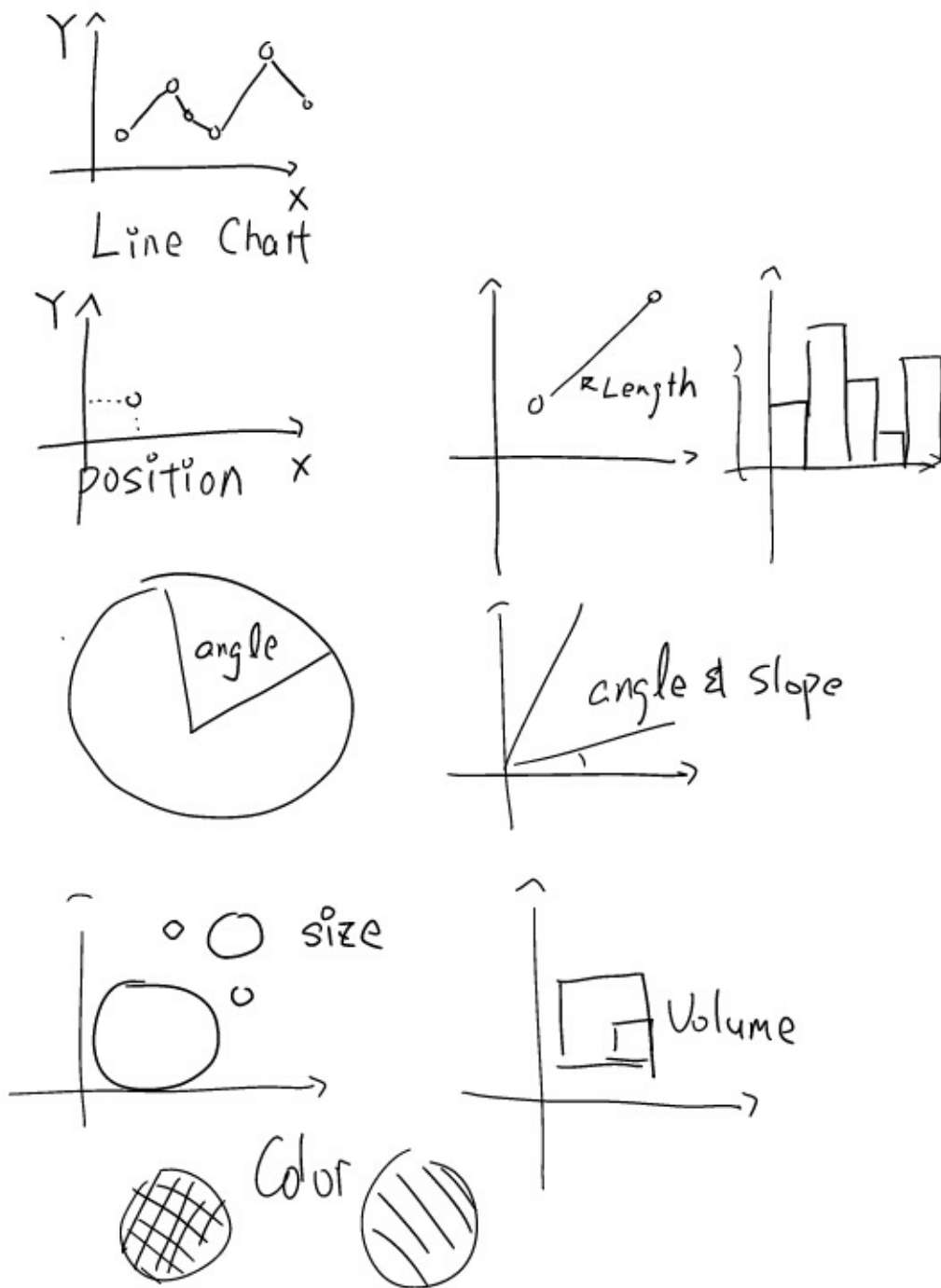
For visualization, lets consider the following few categories of data types for simplification:

1. Numerical data
  - Numbers
2. Categorical data
  - Separate data into groups
3. Time series data

## Visual encoding

Visual encoding can be like position, size, length, angle, slope, color or even density.

For example, it's easy to think about line chart like below:



Rankings of visual encoding accuracy:

1. Position (like x, y)
2. Length
3. Angle & slope
4. Area (size)

5. Volume
6. Color & density

Reference: <http://flowingdata.com/2010/03/20/graphical-perception-learn-the-fundamentals-first/>

## Visualization technologies

We have far too many choices of doing visualization in web pages now. Like D3, NvD3, Chart.js, HighChart, SVG, WebGL ... etc.

We can level the above technologies from high level (easy to use but more restrictions) to low level (harder to use but higher flexibility).

In lowest level, we have WebGL, Canvas and SVG. One can certainly create graphs in this level but it will be a lot of work to create even a simple bar chart. However, you do get the flexibility of changing colors, positions and performance (only if you do it right).

In the middle level, we see D3. D3 provides a lot of flexibility while on the same time keeping its library API simple as possible. You still need to put medium level of effort to create simple type of graph. But on the same time, you will be able to create various graphs that is not just bar chart, line chart or other common charting solution.

On the high level, we have NvD3, HighChart and other tools. In this level, you are not really creating charts from start. Instead, you are doing similar work in Kibana vs using these libraries. You are plotting data into the library and have library do the rest.

Regardless of which level of implementation you choose, the goal is to pick the right tool for the right job!

In this class, we will be learning the underlying implementation from JavaScript first and then D3 finally using the high level libraries to do the job.

Libraries urls:

- [D3.js](#)
- [NVD3.js](#)
- [HighChart](#)
- [Chart.js](#)
- [JavaScript DOM Api](#)

## Intro to JavaScript

Before we talk about D3, it would be very helpful if we know how JavaScript works.

While there are many different aspects of JavaScript that is worth mentioning, I want to summarize the JavaScript into two most important concepts that are used in browser.

1. Dom selector API
2. Callback/promise

## Dom Selector API

In JavaScript, each node in browser is a document object model (DOM). You can do many different things with this DOM element such as adding on a click event listener or even to change the attributes of node.

But you have to know how DOM selector works before you can start to use this DOM element in JavaScript.


In a nutshell, DOM selector API is very close to how CSS selector works. In this level of class, I'm expecting students to know the CSS selector. If you don't know how CSS selector works or require a review on the subject of CSS selector. Please review [this site](#)

BUT! I can hear the cynics.

Why don't we just use jQuery!? It's easier this way. Just \$ everything and we are good to go!

To that, I present you this graph.




stackoverflow

[Questions](#)
[Tags](#)
[Users](#)
[Badges](#)
[Unanswered](#)

[Ask Question](#)

---

## Add a number to another number in JavaScript

---

▲

0

▼

hallo

I have got a number in my JavaScript variable! Now how do I add another number to it? Please

javascript

☆

3 Answers

oldest newest votes

▲

22


▼

✓

You should definitely use jQuery. It's really great and does all things

link | edit | flag

answered 11 minutes ago


jQuery

1,234 ● 2 ● 13

I agree, jQuery is really the best, it solves all kinds of browser problems and is good, as well – [jsmcd0da](#) 8 mins ago

+1 jquery is best quality code ever, if you don't use your a idiot – [Werry\\_Togan](#) 4 mins ago

add comment

▲


4

▼

I think there's a jQuery plugin for that. Google for jQuery basic arithmetic plugin.

link | edit | flag

answered 5 minutes ago

 [Timothy Goatse](#)

4,321 ● 1 ● 12

yeah, jQuery is definately the way to go – [fishnipples](#) 5 mins ago

I used the jQuery diet plugin and lost 10kg in a week – [jfatty](#) 4 mins ago

add comment

▲

-2


▼

To add numbers together you should use the `+` operator, for example:

link | edit | delete | flag

```
var a= 1;
var b= a+2;
alert(b); // 3
```

answered 50 seconds ago

 [bobince](#)

some ● ● ●

-1 not enough jQuery – [jsmcd0da](#) 30 secs ago

you sunk – [Timothy Goatse](#) 3 secs ago

tagged

javascript × 18553

asked

a while ago

viewed

some times

latest activity

just now

Wanted: Yet another ASP.NET developer. See this and other great job listings at [jobs.stackoverflow.com](#).

Related

[What is the best number?](#)  
[How can I use JavaScript to parse some HTML using regex?](#)  
[JavaScript: why is my text content getting mangled when I clone nodes? Obviously I must be doing something wrong as jQuery is perfect](#)  
[Stupid JavaScript floating point numbers are broken](#)  
[How can I extract number from HTML using a regex without `parseFloat` singing the song that ends the world?](#)  
[Is there a jQuery plugin for making an HTML page appear in the browser?](#)  
[Where are my legs?](#)

Please! Native JavaScript API has gone way much better in last 5 years. Unless you have to support IE6 or something legacy like that. You don't really jQuery. You don't!

In example, you can do the DOM selector API via `document.querySelector('#id .class')` and it works the same way as the jQuery `$('#id .class')` with better performance and one less library for you to rely on.

If you want to know more on how to replace jQuery, please go to [this site](#)

In short, learn native JavaScript API. They are amazing. And if you need to look up for the API documents, please go with MDN (Mozilla Developer Network).

## Callback

In JavaScript, it only have one thread for the display. This is why most JavaScript codes are all asynchronous functions.

This is probably by far the most important gotcha for Java developers to get into JavaScript.

What is asynchronous? You asked.

Consider the following code snippet:

```
var request = new XMLHttpRequest();
request.open('GET', 'https://github.com');
request.send(null);
// this is wrong, you will get undefined
console.log(request.response);

// this is callback example
request.onreadystatechange = callback;

function callback() {
  if (request.readyState === XMLHttpRequest.DONE) {
    if (request.status === 200) {
      alert(request.responseText);
    } else {
      alert('There was a problem with the request.');
```

This is not the same as Java that you can always expect the method at line 1 always execute before line 2. Some methods are designed to execute asynchronously and you will have to apply *callback* in order to ensure method execution in synchronous order.

Therefore, when executing the JavaScript methods, make sure you know the concept of *callback*!

Another slightly more complex example

```
function a (callback) {
  return function (callback2) {
    console.log('Method a gets called');
    // I'm using fat arrow function here for my own simplicity
    setTimeout(() => {
      callback(callback2);
    }, Math.random() * 5000);
    console.log('After method a gets called');
  }
}

function b (callback) {
  console.log('Method b gets called');
  alert('Alert blocks future method execution!');
  setTimeout(() => {
    callback();
  }, Math.random() * 5000);
  console.log('After method b gets called');
}

function c () {
  console.log('Method c gets called');
  console.log('this is where method execution gets finished');
}

a(b)(c);
console.log('and we are done???');
```

In summary, when the method doesn't get executed in the order you want. Please be aware that method can be asynchronous and you will need to do callback function.

## D3 (Data Driver Document)

Once you understand above two concepts (especially on the DOM selector part), you are now ready to move onto the D3 part.

So what is different from D3 and native JavaScript API?

It allows you to bind data directly to the DOM and provides various utilities method for visualization related computation.

Lets talk about the starting point first -- document seleciton

Consider the following code snippet in native JavaScript:

```
var paragraphs = document.querySelectorAll('p');
for (var i = 0; i < paragraphs.length; i++) {
  var paragraph = paragraphs.item(i);
  paragraph.style.setProperty("color", "red", null);
}
```

Can be exchanged with D3 code below:

```
d3.selectAll("p").style("color", "red");
```

What about just want to modify single document?

```
d3.select("body").style("background-color", "pink");
```

Since we are using JavaScript to manipulate the attributes, we can certainly add some dynamic attributes like below:

```
// set random color per paragraph
d3.selectAll("p").style("color", function() {
  return "hsl(" + Math.random() * 360 + ",100%,50%)";
});
```

What makes D3 really as a solution toward visualization is its data binding:

```
d3.selectAll("p")
  .data([4, 8, 15, 16, 23, 42])
  .style("font-size", function(d) { return d + "px"; });
```

What happened when the data gets changed after the initial binding?

You can use `enter` and `exit` method to tell D3 to add or remove nodes on adding and removing data.

```
d3.select("body").selectAll("p")
  .data([4, 8, 15, 16, 23, 42])
  .enter().append("p")
  .text(function(d) { return "I'm number " + d + "!"; });
```

Why is this important?

With these few event binding, you will be able to change the graph upon user interaction. In example, you can say if user select different month of the default month, you can change the graph completely with the same code!

## How is D3 implemented behind the scene?

Or what technologies D3 is built upon?

D3 not like other visualization tools is built on top of native browser elements (HTML, SVG and CSS).

In example, you can create SVG element and style them later with external CSS stylesheet yourself.

## Animation

In addition to the static visualization, D3 also provides various animation utilities method like below:

```
d3.select("body").transition()  
  .style("background-color", "black");
```

`transition()` method will help a lot when you are using animation to tell further story!

## D3.scale

D3.scale method can help you to scale number of max and min to different scale.

This is helpful when you are graphing something on the x, y chart.

e.g. you may have a data of domain from -46 to 32. This is extremely hard to graph if you just put it into the x, y axis in the SVG (which can range from 0 to 100). In other word, you will have to do scaling of this min and max of domain to the range.

## Example coding

Check [visualizations/d3/basic.html](https://visualizations.d3.js.org/basic.html)

## Code a barchart

<https://bost.ocks.org/mike/bar/>

As reference, we are going to learn how to create a barchart in D3.

Check [visualizations/d3/barchart.html](https://visualizations.d3.js.org/barchart.html)

## The need for higher level library

Combine above examples, you start to see D3 code can slowly get more and more complex since it is, in my opinion, very low level implementation.

Therefore, for the scope of this class, we will pick some higher level library like Chart.js to do our visualization.

Example of Chart.js

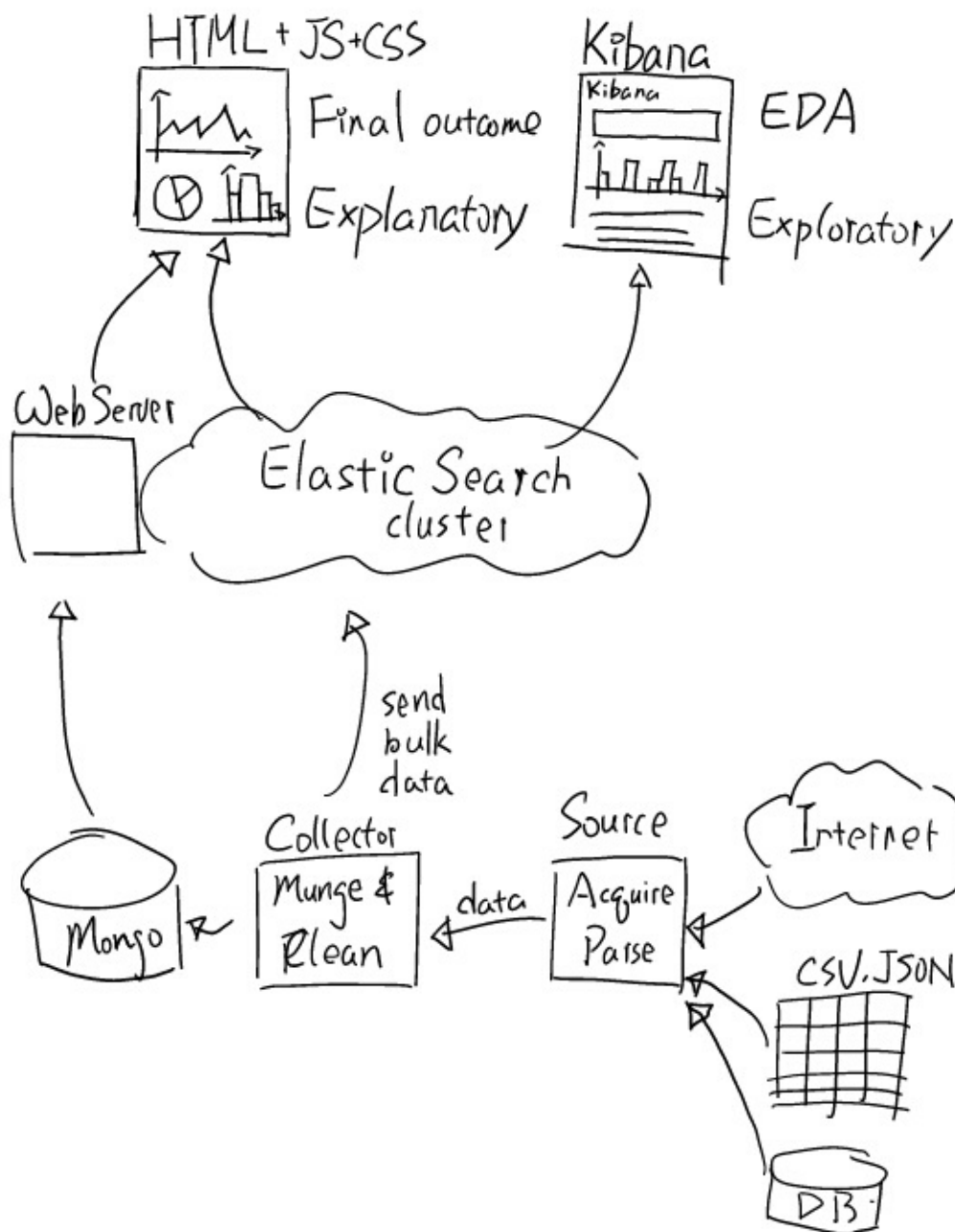
Check visualizations/chart.js/index.html

## Communicating to Server (ElasticSearch)

In traditional architecture, your visualization will be communicating to a web server (built by some developer).

To make more power to the data scientist and to be able to build quite high quality product at the end of this quarter. We decide to use ElasticSearch to replace the traditional web server layer.

Recall earlier architecture graph:



In this graph, visualization will need to make AJAX request to the server to grab data from server.

But in this class, since we are using Elasticsearch, we can simply use the Elasticsearch client to query data directly from Elasticsearch without us building any sort of web server.

To show a demonstration, open up [visualization/chart.js/with-elastic.html](http://visualization/chart.js/with-elastic.html)

Note that you will have to run `npm install` first to install required dependencies.

## Summary

From here we have learned how to use graphing library to visualize.

It is your job to put together graphs into an HTML page for your final project that talks to ElasticSearch.

Further reading: <https://www.elastic.co/blog/data-visualization-elasticsearch-aggregations>

## Objectives

- Basic chart types
  - Bar chart, pie chart, line chart, heap map, geo map ... etc
- Basic JavaScript, HTML and CSS review
- D3

## Metrics

- Visualization of data



# Homework 4

## Due Date

June 5th Midnight.

## Description

In this homework, your job is to **publish** and *polish* your visualization from homework 3.

To recall, in homework 3, you did EDA and present your own findings to us in a private video with many different graphs.

Now, you will have to publish and polish your findings to public with Elasticsearch running on the cloud.

In summary, we want you to show your work on the internet!

But how?

Here is what we suggest you to do:

- Host on AWS:
  1. Host Elasticsearch on [AWS](#)
  2. Start to build your own Kibana dashboard on AWS Elasticsearch
  3. Send us your dashboard link (should display what you have built)
- Optionally as bonus
  1. Build a static site to host your visualization

## Statements to answer for visualizations

- Within a sentence, summarize your findings
- What challenges do you have for polishing visualizations?
- Why did you choose the certain graph types over others (Pie chart, Bar chart, ... etc.)?

## Instructions

## Overview

Previously, we are able to insert data into Elasticsearch we ran locally. However, we are not able to show these visualizations to our friends cross internet. We are sad because we don't get to show our precious works :(

For this homework, we want to host our findings on the internet and want to show these results to our friends and most importantly to us (lecturers).

In order to do so, you will need to do one of the following things:

1. Get an AWS account
2. Create an Elasticsearch service on your AWS
3. Send data to Elasticsearch using JestExampleApp.java (with your modification)
4. Open Kibana to build your dashboard

And then optionally if you want bonus

- Host your visualizations on the cloud using something like [Firebase](#)

## Send data to AWS Elasticsearch

Please look into JestExampleApp.java for sending data from local storage to AWS Elasticsearch cluster.

Try to modify the JestExampleApp.java to send data to your own Elasticsearch cluster and see it showing up on the cloud first.

JestExampleApp is very similar to the ElasticsearchExampleApp.java we went through before but with different library -- Jest.

Please go here to find their API document -- <https://github.com/searchbox-io/Jest>

A quick reason why on we have to use different library rather than using Elasticsearch java.

In AWS, they don't support TCP transport (which is what Elasticsearch Java client was written in). Therefore, we have to use Jest to send data to Elasticsearch with REST only.

Based on the above restriction, you might find sending data to Cloud Elasticsearch to be noticeably longer. Please be patient while waiting data to show up on Elasticsearch cluster on AWS.

Once you started sending data to AWS Elasticsearch cluster, you should be able to query immediately on the AWS cluster.

In example, you should be able to open [http://your-elastic-search.us-west-2.es.amazonaws.com/\\_search](http://your-elastic-search.us-west-2.es.amazonaws.com/_search) and find some result out.

Similarly, you should be able to open your Kibana and start seeing graph immediately by [http://your-elastic-search.us-west-2.es.amazonaws.com/\\_plugin/kibana/](http://your-elastic-search.us-west-2.es.amazonaws.com/_plugin/kibana/)

## Tasks

1. Get an AWS account
2. Create an ElasticSearch service
3. Send data to ElasticSearch
4. Build your own Kibana dashboard
5. Submit the link to your Kibana dashboard

## Deliverables

- CSNS Submission containing:
  - link to your Kibana
  - Pull request on your repository if there is any coding changes

## Grading rubric

- CSNS Submission [1 pt]
- Hosting on cloud [2 pts]
- Visualizations in cloud (data in cloud ElasticSearch instance) [7 pts]
- Visualization using web technologies (NVD3.js, Chart.js or any other libraries you used) [2 pts]

Students' feedback on what could've done better for this class:

1. Use only one languages as to Java and Python. (aka. focus only on Python)
2. Give more time on certain technology (under-grad wants more on Kibana while grad wants more on Python & Machine Learning)
3. Give overall architecture first
4. More guided project while still keeping the certain level of freedom as to complete freedom without guidelines
5. More take home quiz (or some certain way of make up after in class quiz)

Some crazy ideas but I like:

- Make project senior design project (so having more time to finish the project)