APPM Seminar
April 20, 2016

Nathan Halko
https://github.com/nathanhalko
n8halko@gmail.com

All code and slides are on github MahoutApp

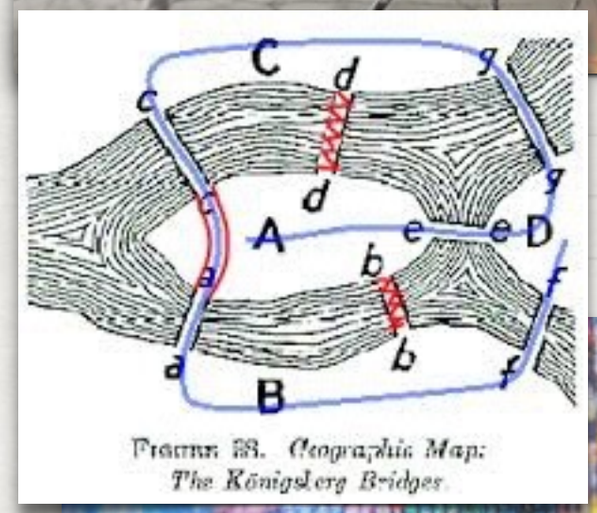
Outline

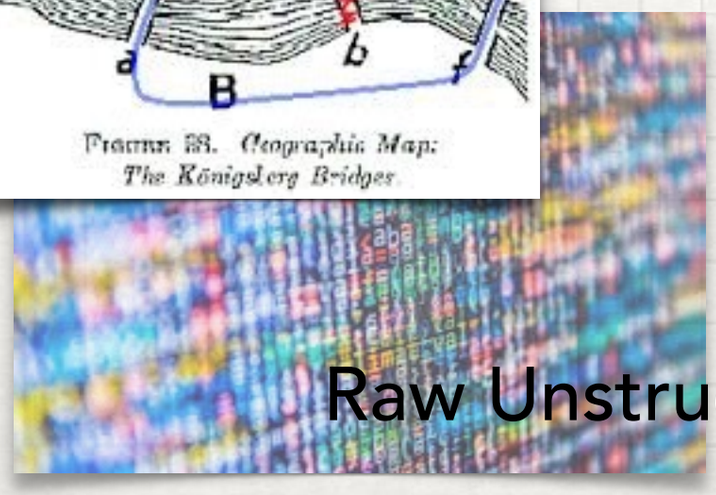
- Intro: Data science, Mahout, Spark, Scala
- Mahout setup demo
- tools demo: jq, curl, awk, sed, cut, curl from google docs, screen, csshX, sort, join,grep
- Goals:
  - generate interest in open source
  - create scalable, sharable and reusable code
- Wishlist:
  - all my code was on github
  - I had done more robust coding: used databases, used Python,R,opensource
- Aha! moments:
  - getting an idea from my head to working code (TwitterGraphBuilder)
  - understanding the full solution stack (SpotPipeline.scala)
  - setting up a new project/code/library (Giraph, Titan)

spotright

Nathan Halko, Data Scientist

n8halko@gmail.com





Data Science

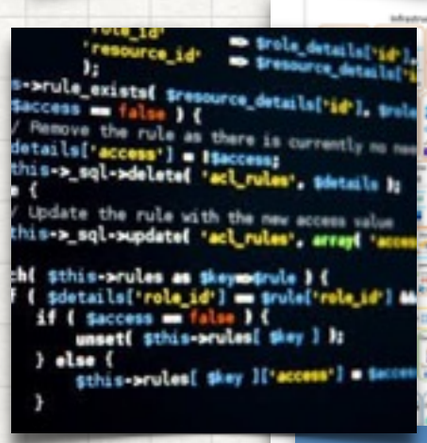
Algorithms

Clients

Sales/Marketing

Product

Engineering



Infrastructure/IT

The Stack

Apps, UserInterface

API's, webservice

Languages:
Scala, Java, Python

Client Code (libraries, packages):
Mahout, Matlab?

Framework (execution engine):
Spark, Hadoop

Database:
Cassandra, HDFS, SQL, filesystem

Hardware (servers):
Laptop, Cloud (aws)

Mahout

https://mahout.apache.org/ , https://github.com/apache/mahout, mahout-samsara-book , https://github.com/andrewpalumbo/mahout-samsara-book

Spark:: http://spark.apache.org/ , http://spark.apache.org/docs/latest/quick-start.html
Other:: https://git-scm.com/ , https://maven.apache.org/

Git Mahout and Spark (from download):

```
project> git clone https://github.com/apache/mahout.git; cd mahout
project/mahout> echo “this is your project directory”; cd ..
project> tar -xzvf ~/Downloads/spark-1.5.2-bin-hadoop2.6.tgz
project> ln -s spark-mahout spark-1.5.2-bin-hadoop2.6; cd spark-mahout
project/spark-mahout> bin/spark-shell –master local[4]
```

Try it out!

```
spark> val rdd = sc.parallelize(List.fill(1000000)(scala.util.Random.nextDouble))
spark> rdd.count()

SparkUI -> localhost:4041
```

Setup Mahout (from project/mahout directory):

```
- create a file named setup with:
  export MAHOUT_HOME=/project/mahout
  export MAHOUT_LOCAL=true
  export MAHOUT_OPTS=“-Xmx4g"
  export SPARK_HOME=/project/spark-mahout
> source setup # activate the environment variables
> mvn clean install -DskipTests
> MASTER=local[4] bin/mahout spark-shell
```

Try it out!

```
mahout> val mtxA = Matrices.symmetricUniformView(5000, 5000, 1234)
mahout> mtxA.rowSums
```

Creating a project

https://www.jetbrains.com/idea/ , http://www.scala-sbt.org/

Create the bones:

```
> mkdir MahoutApp; cd MahoutApp
MahoutApp> mkdir -p src/{main,test}/scala/com/nhalko/mahoutapp
```

Create a *build.sbt* file with contents:

```
lazy val root = (project in file(".")).
settings(
  name := "mahoutapp",
  version := "0.1",
  scalaVersion := “2.10.4"
 ))
```

Open the project in IDEA and follow the wizard.

Add dependencies and some code:

Update build.sbt to look like this

Create file src/test/scala/com/nhalko/mahoutapp/RidgeRegression.scala

Run it or play with it in the repl!

```
> sbt “testOnly *RidgeRegression”
> sbt test:console
```

Create and run (from the shell) .mscala scripts

```
mahout> :load my_script.mscala
```

Use your .jar in Mahout shell

```
MahoutApp> sbt package # create a .jar in target/scala-2.10 dir of your project
mahout> :cp path/to/your/MahoutApp.jar // access your project’s classes and methods from mahout shell
```

CSV data from Google Docs

Find some data: *https://catalog.data.gov/dataset?res_format=CSV*
Download it:

```
> curl -o myDataSet.csv -s 'https://data.consumerfinance.gov/api/views/s6ew-h6mp/rows.csv?accessType=DOWNLOAD'

> curl … | cut -d, -f1-3,7-10,13
> cat myDataSet.csv | awk -F, '{print $2,$3,$5}'
```

JSON Data with **jq**:

```
> curl -o demographics.json 'https://data.cityofnewyork.us/api/views/kku6-nxdu/rows.json?accessType=DOWNLOAD'
> cat demographics.json | jq '.'
> cat … | jq '.data[100]'
> cat … | jq '.meta.view.description'
```

Upload to Google Docs: follow the import wizard
Get it from Google Docs:

```
> curl -s -d "format=csv&gid=934754936" https://docs.google.com/spreadsheets/d/1oyAEF1TfAFdW2f9XEnxLpv9XDrJzdqwRSj7HLhEmSxM/export

mahout> scala.io.Source.fromURL("https://docs.google.com/spreadsheets/d/1oyAEF1TfAFdW2f9XEnxLpv9XDrJzdqwRSj7HLhEmSxM/export?format=csv&gid=934754936")
res20: scala.io.BufferedSource = non-empty iterator
```

Working the data example:

```
mahout> val raw = scala.io.Source.fromURL("https://docs.google.com/spreadsheets/d/
1oyAEF1TfAFdW2f9XEnxLpv9XDrJzdqwRSj7HLhEmSxM/export?format=csv&gid=934754936").getLines
raw: Iterator[String] = non-empty iterator

mahout> val header = raw.next.split(",")
header: Array[String] = Array(Date received, Product, Sub-product, Issue, Sub-issue, Consumer …

mahout> val data = raw.toList.map(_.split(","))
data: List[Array[String]] = List(Array(8/30/2013, Student loan, Non-federal student loan,…

mahout> val row0 = header.zip(data(0)).map {case (h,d) => h -> d}.toMap
row0: scala.collection.immutable.Map[String,String] = Map(Sub-product -> Non-federal student …

mahout> row0(“Sub-Product”)
```

Few more bash tricks (wc, grep, join, sort, screen, csshX):

```
> wc -l Consumer_Complaints.csv
  708280 Consumer_Complaints.csv
> grep -c Credit Consumer_Complaints.csv
195689
> head -1 Consumer_Complaints_sample.csv | sed -e 's/,/;;/g'
Date received;;Product;;Sub-product;;Issue;;Sub-issue;;Consumer complaint
narrative;;Company public response;;Company;;State;;ZIP code;;Tags;;Consumer consent
provided?;;Submitted via;;Date sent to company;;Company response to consumer;;Timely
response?;;Consumer disputed?;;Complaint ID
```