MSCBIO/CMPBIO 2065: Scalable Machine Learning for Big Data Biology, Fall 2016

A1: Population Scale Clustering from Genome Variant Data

Date: Sep 20, 2016
Due: Oct 5, 2016 by email
This assignment is worth 10% ***of your grade.***

Background: *The material here draws heavily on the Genomic Medicine Primers by Collins et al, 2002 and 2010 in NEJM.* Human body is made of cells. All cells have 23 pair of chromosomes. Chromosomes are made of deoxyribonucleic acid (DNA) molecules. DNA stores biological information in a three-dimensional, double-stranded biopolymer coiled around each other in the well-known double helix form. DNA carries genetic instructions needed for growth, development, functioning and reproduction of all known living organisms and many viruses. This information is replicated as and when the two strands separate.

Dogma: The central hypothesis (otherwise called the "dogma") of molecular biology is a description of the normal flow of biological information between the three major classes of biopolymers: DNA and RNA (both nucleic acids) and proteins. In particular, it states that DNA can be copied into DNA (replication), DNA information can be copied into messenger RNA and proteins can be synthesized using the information in mRNA as a template (translation).

The two strands of DNA run in opposite directions to each other and are thus antiparallel. There are four types of nucleic acids with the acronyms: A, C, G, T (see below). It is the sequence of these four nucleobases along the backbone that encodes biological information. RNA strands are created using DNA strands as a template in a process called transcription. Under the genetic code, these RNA strands are translated to specify the sequence of amino acids within proteins in a process called translation. The DNA backbone is resistant to cleavage, and both strands of the double-stranded structure store the same biological information. A large part of DNA (more than 98% for humans) is non-coding, meaning that these sections do not serve as patterns for protein sequences.

Within eukaryotic cells (animals, plants, fungi), DNA is organized into long structures called chromosomes. During cell division these chromosomes are duplicated in the process of DNA replication, providing each cell its own complete set of chromosomes. Eukaryotic organisms store most of their DNA inside the cell nucleus and some of their DNA in organelles, such as mitochondria or chloroplasts. In contrast, prokaryotes (bacteria and archaea) store their DNA only in the cytoplasm. Within the eukaryotic chromosomes, chromatin proteins such as histones compact and organize DNA. These compact structures guide the interactions between DNA and other proteins, helping control which parts of the DNA are transcribed.

Definitions: For convenience we will use the following definitions:

- Base pair (bp): Two nitrogenous bases paired together in double-stranded DNA by weak bonds; specific pairing of these bases (adenine with thymine and guanine with cytosine) facilitates accurate DNA replication; when quantified (e.g., 8 bp), bp refers to the physical length of a sequence of nucleotides.
- Gene: The fundamental physical and functional unit of heredity. A gene is an ordered sequence of nucleotides located in a particular position on a particular chromosome that encodes a specific functional product (i.e., a protein or an RNA molecule).
- Allele: One of two or more versions of a genetic sequence at a particular location in the genome.

- Genome: The entire set of genetic instructions found in a cell. In humans, the genome consists of 23 pairs of chromosomes, found in the nucleus, as well as a small chromosome found in the cells? mitochondria.
- Genotype: A person's complete collection of genes. The term can also refer to the two alleles inherited for a particular gene.
- Phenotype: The observable traits of an individual person, such as height, eye color, and blood type. Some traits are largely determined by genotype, whereas others are largely determined by environmental factors.
- Single-nucleotide polymorphism (SNP): A single-nucleotide variation in a genetic sequence; a common form of variation in the human genome.

Genome Variant Data: Specific locations in the human genome where differences between individual people are found are generally referred to as variations, and the term "normal" or "wild type" is often used to refer to the most common variant at a location in a given population group. In its simplest form, the variation has two different spellings, referred to as "alleles." If the frequency of the minor allele is greater than 1%, such variants are called polymorphisms. The word "mutation" is generally reserved for changes in DNA that are believed or known to be pathologic (e.g., the mutations in the gene that causes cystic fibrosis, which is cystic fibrosis transmembrane conductance regulator, or CFTR) or for changes that are recent (e.g., a DNA base change in a cancer that is not present in the patient's germ-line DNA). A person's complete genomic sequence (genotype), acting in concert with environmental influences, creates individuality (phenotype). A collection of alleles arranged linearly along a person's DNA molecule is known as a haplotype. Humans are very similar at the DNA sequence level; about 99.6% of base pairs are identical from person to person. Given the size of the genome (approximately 6 billion bp), there is substantial latitude for individual genetic variation, since the difference between any two people is about 24 million bp. It is of great interest to catalogue these variants, commonly referred to as "single-nucleotide polymorphisms" (SNPs), and to correlate these specific genotypic variations with specific phenotypic variations relevant to health. **You will be tapping into the 1000 Genomes Project data to begin answering some of these questions.**

Your goals:

1. learn to use Apache Spark and Spark's Scala bindings
2. learn to perform genomic analysis using ADAM
3. work with chromosome sequence data and variant information from the 1000 Genomes Project
4. perform population-level clustering of the genome variant data

What to hand in: You will submit your code through `http://bits.csb.pitt.edu/mscbio2065/` (This website will be setup "shortly").

1. **[0 points] Spark and ADAM installations**

   Spark 2.0 In order to get Spark working on the cluster, it was necessary to upgrade to using Spark 2.0. The installation instructions are slightly different. Note the expectation is that you will be working on Linux or OS X (but it's possible the new Windows 10 Bash commandline will work as well). Instructions for setting up a virtual machine can be found here: http://www.psychocats.net/ubuntu/virtualbox

   (a) Download Apache Spark 2.0 from: `http://spark.apache.org/downloads.html`
   The Pre-built binaries are fine - you need only unpack them. Once they are unpacked set your `SPARK_HOME` environment variable to point to this directory. Edit `.bashrc` appropriately, e.g.: `export SPARK_HOME=/home/dkoes/build/spark-2.0.0-bin-hadoop2.7`

   (b) Install Maven 3.3. Ubuntu 14.04 has version 3.0.5 while Ubuntu 16.04 has 3.3.9. MacPorts on OS X has version 3.3, but you need to install the maven3 package and use the mvn3 command. If you don't have the latest version, download the binary package (`https://maven.apache.org/download.cgi`) and install so that the 3.3.9 mvn binary is in your path.

   (c) ADAM is a genomics analysis platform that provides both an application programming and command line interface for analyzing genomics data in a distributed fashion. ADAM reads data stored in bdg-format and ADAM Parquet files supports parallel distribution of sequence data. You will need to have Maven (version 3) installed before you can build ADAM (`sudo apt-get install maven`). To install ADAM with Spark 2.0:

   ```
   $ git clone https://github.com/bigdatagenomics/adam.git
   $ cd adam
   $ ./scripts/move_to_spark_2.sh
   $ ./scripts/move_to_scala_2.11.sh
   $ mvn clean package -DskipTests
   ```

   (d) Give a quick read of the 1000 Genomes Project
   `http://www.1000genomes.org`

2. **[30 points] Reading 1000 Genomes Project Data with ADAM/Spark**

   (a) **Process panel file.** In this task you will gain familiarity with Scala (no need to use Spark RDDs yet) to read in and process a tab-delimited file that describes the population membership of each person in the 1000 Genomes project.

   i. Download the file:
   `ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/release/20130502/integrated_call_samples_v3.20130502.ALL.panel`

   ii. Write a Scala script that identifies the sub-populations with more than 90 members. This is a chance to get familiar with using idioms like `tmap` and `filter` instead of loops to process data.

   ```
   val panelfile = "integrated_call_samples_v3.20130502.ALL.panel"
   import scala.io.Source
   val biggroups = Source.fromFile(panelfile).getLines().drop(1)...
   ```

   iii. Create a set of all the people (sample ids) that are contained within these large populations.

   ```
   val people = ....toSet
   ```

   iv. Print out the sizes of these two groups.

```
println("Populations with more than 90 individuals: "+biggroups.size)
println("Individuals from these populations: "+people.size)
```

v. Get the right answer.

```
Populations with more than 90 individuals: 21
Individuals from these populations: 2123
```

(b) **Create test examples.** In this task we will download and process small slices of the full variant call information available.

i. Create a two subsamples of Chromosome 1 data using the DataSlicer tool: `http://browser.1000genomes.org/Homo_sapiens/UserData/SelectSlice`. For our small example, slice the region "1:10000-80000", and for our medium example slice the region "1:10000-800000"

ii. Convert the VCF format to ADAM format.

```
$ adam-submit vcf2adam 1.10000-80000.ALL.chr1.
    phase3_shapeit2_mvncall_integrated_v5a.20130502.genotypes.vcf.gz small.adam
$ adam-submit vcf2adam 1.10000-800000.ALL.chr1.
    phase3_shapeit2_mvncall_integrated_v5a.20130502.genotypes.vcf.gz medium.adam
```

iii. Inspect the variants file using the ADAM shell.

```
scala> import org.bdgenomics.adam.rdd.ADAMContext._
```

The above automatically adds ADAM load functions to the Spark Context variable `sc`. (Type `sc.`$< TAB >$ to see the available functions).

```
scala> val data = sc.loadGenotypes("small.adam")
```

The above loads `small.adam` into a Scala Product of (rdd,samples,sequences). The RDD is what we are interested in.

```
scala> println(data.rdd.take(1)(0))
{"variant": {"variantErrorProbability": 100, "contigName": null, "start": null, "
    end": null, "referenceAllele": "A", "alternateAllele": "AC", "svAllele": null,
     "isSomatic": false}, "contigName": "1", "start": 10176, "end": 10177, "
    variantCallingAnnotations": {"variantIsPassing": true, "variantFilters": [], "
    downsampled": null, "baseQRankSum": null, "fisherStrandBiasPValue": null, "
    rmsMapQ": null, "mapq0Reads": null, "mqRankSum": null, "readPositionRankSum":
    null, "genotypePriors": [], "genotypePosteriors": [], "vqslod": null, "culprit
    ": null, "attributes": {}}, "sampleId": "HG00096", "sampleDescription": null,
    "processingDescription": null, "alleles": ["Alt", "Ref"], "
    expectedAlleleDosage": null, "referenceReadDepth": null, "alternateReadDepth":
     null, "readDepth": null, "minReadDepth": null, "genotypeQuality": null, "
    genotypeLikelihoods": [], "nonReferenceLikelihoods": [], "strandBiasComponents
    ": [], "splitFromMultiAllelic": false, "isPhased": true, "phaseSetId": null, "
    phaseQuality": null}
```

The above is a single variant call. For our purposes, all we care about are the `contigName`, `start`, and `end` fields (which uniquely define the variant position), the `sampleId` (who has this variant), and the `alleles` (what this variant is). Specifically, we are interested if the alleles match the reference ("Ref") sequence or not. Recall that humans have two copies of each chromosome, one from your father and one from your mother, hence the two allele values (other organisms, especially plants, display polyploidy). Unfortunately, the allele record is stored as a `java.util.List` and the serializer likes to crash with a NullPointerExecption if you don't convert it to a Scala type, so we use this work around:

```
import collection.JavaConverters._
def convertAlleles(x: java.util.List[org.bdgenomics.formats.avro.GenotypeAllele] )
     = { x.asScala.map(_.toString) }
```

This let's us extract just the information we care about:

```
scala> data.rdd.map(r => (r.contigName,r.start, r.end, r.sampleId, convertAlleles(
    r.alleles))).take(1)
res2: Array[(String, Long, Long, String, scala.collection.mutable.Buffer[String])]
     = Array((1,10176,10177,HG00096,ArrayBuffer(Alt, Ref)))
```

3. [**40 points**] **Clustering Variants** In this task we cluster individual genotypes with the goal of seeing how well our clusters recapitulate the known population groups of the individuals.

   (a) **Identify good variants**. We are only interested in those variants that are present exactly once in every member of our desired population groups. Count the total number of variants and the number of variants that are present exactly once (in the small data set some are duplicated, in larger data sets some may be missing).

   ```
   Total variants: 551
   Variants with right number of samples: 548
   ```

   (b) **Convert variant data into vectors** For purposes of clustering, we want to represent each of the 2123 individuals as vectors of numbers where each position of the vector corresponds to a variant and indicates whether that person has zero, one or two alternative (not "Ref") alleles. That is, you might end up with a data structure like this (HG01459 is a sample id):

   ```
   (String, Array[Float]) = (HG01459,Array(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
       0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
       0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 2.0,
   ```

   (c) **Implement K-means**

   The k-means algorithm is relatively simple. For a given K and MAX,

   0. Randomly sample K vectors to serve as cluster centers.
   1. For each vector, find closest cluster center (partitioning step)
   2. Replace each cluster center by average of data points in its partition
   3. Iterate 1 and 2 until convergence (clusters don't change) or MAX iterations.

   After performing K-means clustering with K equal to the number of populations (**K=21**), you should print the resulting clusters one-per-a-line, prefaced by the "Clusters:" keyword.

   ```
   Clusters:
   NA18549 NA18747 NA18648 NA18980 ...
   NA18644 HG00759 NA18545 NA18546 ...
   NA19067 HG03228 HG00654 HG01260 ...
   etc.
   ```

4. [**30 points**] **Scale and Improve** We will setup a website, `http://bits.csb.pitt.edu/mscbio2065/`, where you can submit your Scala code for evaluation on a ~100 core cluster of 4-core machines where each machine has 2.5GB of memory available to Spark. Your code will be evaluated for how quickly it runs and how good a clustering it finds (as determined by congruence with existing population groups).