SEQSpark for the Analysis of Exome and Whole Genome Sequence Data

This exercise will demonstrate how to perform annotation, data quality control (QC) and association analysis of your whole genome or exome sequence data using SEQSpark.[1] SEQSpark can also be used to perform annotation, QC and analysis of imputed data sets such as UK Biobank. SEQSpark can greatly increase the speed of analyzing large scale genetic epidemiological studies through parallelization be implementing Apache Spark and the Hadoop file system. In this exercise, we will concentrate on analyzing exome data and will only demonstrate a few of the commands. To learn more about the extensive capabilities of SEQSpark please refer to Zhang et al. 2017[1] as well as online documentation (https://github.com/statgenetics/seqspark). For information on how to install Spark and Hadoop on your workstation or cluster please see https://statgenetics.github.io/seqspark/spark

For this exercise, we simulated 619 exonic chromosome 22 variants for 500 individuals using non-Finnish European allele frequencies obtained from 33,370 individuals from ExAC. Genotype data were generated assuming Hardy-Weinberg equilibrium (HWE). To generate realistic sequence data to analyze and perform data QC, we generated sequence data with variant sites, variable read depths and genotype quality scores. For each genotype, read depth information was generated by drawing from a $negative\ binomial\ (r,p)$ where $r = 7$ is the number of failures until the experiment is stopped and $p = 0.2$ is the probability of failure, that led to ~2% of the genotypes having a read depth (DP) of <8×. GQ scores were also generated using a two-step sampling procedure: first for 95% of the genotypes the GQ was set to 99; for the remaining genotypes the GQ scores is obtained by drawing from a $uniform\ (a,b)$ distribution where $a = 0$ is the starting value and $b = 100$ is the ending value, leading to ~1% of the GQ scores being less than 20. Missing genotypes were also introduced using a probability of 5%.

None of the phenotype data were generated to be in association with the genotypes, i.e., data are generated under the null hypothesis of no association. For each sample that exome data were generated, age, sex, a quantitative trait, and a dichotomous trait were assigned. Age was generated using a $negative\ binomial\ (r,p)$ distribution where $r = 20$ and $p = 0.3$, while the quantitative trait body mass index (BMI) was simulated using a $normal(\mu,\sigma^2)$ distribution where the mean $\mu = 25$ and the variance $\sigma^2 = 3$. Fifty percent of the samples were assigned to be male and the other half were assigned to be females. For the qualitative (disease) trait, 50% of the samples were assigned to be cases and the other half controls.

1. Introduction to the configuration file

To run SEQSpark you will need to create a configuration file to specify the parameters, e.g. the locations of the input files, the quality control criteria, and the methods to be used for the association analysis. Placing the parameters into a configuration file aids the user in checking the analysis steps which have been performed and also allows for data reanalysis with or without changes to parameters. The format of the configuration file is called Human-Optimized Config Object Notation (HOCON), which is a user-friendly superset of the more commonly used JavaScript Object Notation (JSON), and can also be used with SEQSpark. Here we cover some basics of the HOCON format to help you complete the exercise and also to use SEQSpark for your projects.

The HOCON configuration file is a tree of key-value pairs. The keys are the names of parameters to which we want to assign values. The possible value types include: null, string, number, boolean, array, and object. Null represents an unset value while string, number, and boolean are basic value types. An array, which are listed in a pair of square brackets, are values of the same type. An object, in a pair of curly brackets, is a list of key-value pairs. Because an object contains key-value pairs, the structure of the whole file can be viewed as a tree, with the keys as inner nodes types and the objects as leaves.

Here is an example:

```
example1 = {
      a = 42 #number
      b = [ "foo", "bar" ] #array of strings
      c = { /** Anything between two slashes is also comment */
            d = true #boolean
      } #object
} #object
```

Please note that, "=" is the key-value separator, "#" and "/" are markers to start comments. Can you find all the keys in this file? ("example1", "a", "b", "c", and "d"). The value associated with the key "example1" is an object containing everything else. Inside the object, there are three key-value pairs. The types of the three values are number, array, and object, respectively. The third value is again an object containing one key-value pair.

You can use dotted-notation for keys inside an object, so it is the same to write the above file in the following way:

```
example1.a = 42
example1.b = [ "foo", "bar" ]
example1.c.d = true
```

You can use the above two styles in one file:

```
example1 = {
      a = 42
      b = [ "foo", "bar" ]
      c.d = true #notice this line
}
```

You can also use ":" as the key-value separator, and a separator before the left curly bracket can be omitted.

```
example1 {
      a = 42
      b : [ "foo", "bar" ]
      c { d : true }
}
```

Please check https://goo.gl/X3iks2 and https://goo.gl/4EifCn for more explanations and examples of the HOCON format.

In SEQSpark, we organize parameters in this way. Please check the file `demo.conf`, in which we have made comments explaining the meaning of every parameter used in the exercise. For details about all the SEQSpark parameters, please visit the online documentation https://statgenetics.github.io/seqspark/reference

2. Data Preparation

In your SEQSpark directory you will find the following files, `demo.vcf.bz2`, which is a compressed vcf file which provides variant calls, read depths and GQ scores. Please note for this example only a limited amount of variant annotation is provided. Additionally, the `demo.tsv` file is provided which provides the simulated quantitative trait (BMI), qualitative trait (disease) and covariates (age and sex).

Please first upload the `vcf` and `tsv` files into the Hadoop file system (HDFS) using the following commands.

```
hdfs dfs -put demo.vcf.bz2
hdfs dfs -put demo.tsv
```

3. Annotation with GnomAD

Next you will create a configuration file to specify the parameters. We provide the files that contains the data for the example analysis here:

File	URL	Comment
annotation.conf	https://goo.gl/9XZGHb	for annotation
qc.conf	https://goo.gl/jeJqc2	for annotation and QC
demo.conf	https://goo.gl/TQR8tp	for the full analysis (annotation, QC, association)

You can download these files instead of cutting and pasting them from what is written below. Because we will use the allele frequency information form GnomAD in the analysis, let's first set up how to annotate the variants with this database. In this exercise, we only use a subset of the database to keep the size small.

```
#file annotation.conf
seqspark {
  project = demo
  partitions = 10
  pipeline = [ "annotation" ]
  input.genotype.path = "demo.vcf.bz2"
  annotation {
    addInfo {
      gnomAD = "gnomad"
      Total_AF = "gnomad.AF"
      SS_AF = "gnomad.AC_NFE/gnomad.AN_NFE"
    }
    gnomad = {
      path = "seqspark/gnomad.exome.vcf.bz2"
      format = "vcf"
    }
  }
}
```

Explanation of parameters:

seqspark {} is mandated and anything outside this object will be ignored by the program.
project is the name of the run, and it will be used as a suffix when creating output folder in the local filesystem and cached dataset in HDFS.
partitions the number of partitions of genotype and database datasets. The default it is 1024, which would be too large for this toy exercise, so it is set to 10.
pipeline is an array of procedures to be performed, here we will only perform the annotation procedure which is described below.
input.genotype.path gives the path of the input vcf file. By default, the location of the files is within HDFS.
annotation {} specifies the parameters that control which databases will be used and the information which will be extracted. Inside this object, there are two objects addInfo and gnomad.

addInfo specifies what information will be included to our dataset. Here the AF value from the INFO field of the gnomAD VCF file will be extracted and stored as refAF, and another two values AC_NFE and AN_NFE will also be extracted from gnomAD and the ratio will be calculated and stored as SS_AF. Notice that the format of a reference to a value in a database is db.key.

gnomad specifies the path of the database and the format is vcf. Other database formats, i.e. tsv and csv, are also supported.

Export the annotated VCF file

You may want to check the annotation or use the annotated VCF file with other software. Simply add the following line to the seqspark object:

```
output.genotype.export = true
```

This will cause SEQSpark to store the genotype file after the pipeline. Variant and sample filters can be applied before exportation, i.e. only export a subset of variants and/or samples. For example, if we only need to check the annotation, we don't have to copy the genotypes from the VCF file. We can only store variant level information, i.e. the first eight columns of a VCF file. In this case we need no sample data. We can modify the above output.genotype object as follows:

```
output.genotype {
    export = true
    samples = none
  }
```

If you only want to analyze or annotate a subset of variants, there are several options. You can use an expression for regions, a file containing regions, or just "exome" for all the exons defined by the RefSeq database. For more examples of variant filters, please check the demo.conf file which we provide. Please note that the variant and sample filters can also be used in the input.genotype object, to import only a subset of variants and or samples. You can now run the pipeline using the following command"

```
seqspark annotation.conf
```

4. Data quality control (QC)

Next you will create a configuration file to specify the desired QC pipeline. You can either open a new file and type in the configurations, or download a copy of this file here https://goo.gl/jeJqc2.

Filter genotypes and variants

First, let's set up a basic pipeline that filter genotypes and variants. Genotypes (variant calls) will be filtered based on read depth (DP) and genotype quality (GQ) score where genotypes with a DP value of less than 8 and GQ score less than 20 will be removed, i.e. replaced with missing values. Then variants with missing rate higher than 10% will also be removed.

```
#file qc.conf
seqspark {
  project = demo
  partitions = 10
  pipeline = [ "annotation", "qualityControl" ]
  input {
    genotype.path = "demo.vcf.bz2"
    phenotype.path = "demo.tsv"
  }
  output.genotype {
    export = true
    samples = none
  }
  annotation {
    addInfo {
      gnomAD = "gnomad"
      Total_AF = "gnomad.AF"
      SS_AF = "gnomad.AC_NFE/gnomad.AN_NFE"
    }
    gnomad = {
      path = "/user/student/ref/demo.gnomad.vcf.bz2"
      format = "vcf"
    }
  }
  qualityControl {
    genotypes = ["DP >= 8 and GQ >= 20"]
    variants = [ "missingRate <= 0.1", "gnomAD" ]
  }
}
```

Explanation of the parameters:

Comparing with the `annotation.conf`, several changes were made here: **`pipeline`** now contains another procedure "qualityControl"; **`input`** now contains another path for phenotype data; and a new object `qualityControl` was also added to `seqspark`. The first two are trivial but we will explain the `qualityControl` object in detail.

`qualityControl` contains the parameters for the quality control procedures. Here we have two filters for genotypes and variants. Each filter is an array of logical expressions.

> **`Genotypes:`** specifies the criteria for filtering genotypes. The logical expressions in this array should be in the form "`key cmp value [and/or key cmp value ...]`". `key` must be a field of the genotype FORMAT. `cmp` should be one of >, <, >=, <=, ==. `value`

must be a string or number. The items in this array will be joined using the "and" logic. Our setting above is equivalent to `genotypes = ["DP >= 8", "GD >= 20"]`.

`Variants:` specifies the criteria for filtering variants. The format is similar to the genotypes. For information about possible keys and examples, please check the `demo.conf` file we provided.

Generate summaries for genotype data

The above configuration will filter genotypes and variants for exportation or subsequent analysis, but how can the data QC be assessed? We should be able to generate some summary information, or compute some interesting statistics based on the data before and after QC to evaluate how well data QC is performing. In this exercise, we provide two examples.

For principal components analysis (PCA), by default variants after QC from the autosomes with minor allele frequency (MAF)>=1% will be selected, and LD-based SNP pruning will be performed. This parameter can be changed for example if you wish to use a lower MAF cut-off. The generated PC terms can be used for identifying outliers and adjusting for population stratification and admixture in the association. The file `demo/pca.txt` stores the PC terms.

For Ti/Tv calculations, variants will be categorized as transition (Ti), transversion (Tv), or neither. The Ti/Tv ratios can be calculated by samples and variant groups defined by user. The results will be in `demo/qc.txt`

These analyses are performed using the following lines which are in qc.conf.

```
summaries = ["pca", "titv"]
titv.by = ["samples", "QC", "MAF"]
group.variants {
  QC {
    pass = ["SS_PASS"]
    dicard = ["! SS_PASS"]
  }
  MAF {
    rare = ["maf < 0.01 or maf > 0.99", "SS_PASS"]
    common = ["maf >= 0.01 and maf <= 0.99", "SS_PASS"]
  }
}
```

Now you can run SEQSpark to perform annotation and QC with the following command:

```
seqspark qc.conf
```

In theory, you could continue to set up the association analysis and perform all the analyses in a single run. However, for real data analysis, investigators tend to run the QC pipeline first and even multiple times to make sure that proper data QC has been performed.

Question 1: In the demo/qc.txt file, find the numbers:

a. Number of genotypes before QC _____ after QC _____
b. Number of variants before QC _____ after QC _____
c. Number of rare variants defined by the gnomAD MAF _____
d. Number of common variants defined by the gnomAD MAF _____

Question 2: In the output/titv.txt file, find the numbers:

e. Ti/Tv ratio for all variants before QC _____ after QC _____

5. Single variant association testing

The association analysis is performed by the below object which is included in demo.conf.

```
#assoc.conf
seqspark {
  project = demo
  partitions = 10
  pipeline = [ "annotation", "qualityControl", "association" ]
  input {...} #check qc.conf for this object
  output {...} #check qc.conf for this object
  annotation {...} #check qc.conf for this object
  qualityControl {...} #check qc.conf for this object
  association {
    trait {
      list = ["bmi"]
      bmi {
        binary = false
        covariates = ["sex", "age", "disease"]
        pc = 0
      }
    }
    method {
      list = ["snv"]
      snv {
            type = "snv"
            maf {
                  source = "pooled"
                  cutoff = 0.01
            }
      }
    }
  }
}
```

The `association` section consists of two objects, `trait` and `method`.

`Trait:` describes the phenotypes to be analyzed and the corresponding covariates to include in the regression model. Here a simulated quantitative trait BMI is being analyzed. `pc` denotes how many principal components should be used as covariates in the analysis.
`Method:` describes the methods that will be used for association analysis, e.g. single variant test, type of rare variant aggregate association test.
`List:` catalogs the methods which will be used for analysis. Here we have **`snv`** for single variant analysis. The method **snv** is configurable. User can control the MAF source, e.g. gnomAD and cutoff, e.g. MAF<0.005.

```
method {
      list = [ snv1, snv2 ]
      snv1 {
            type = snv
            maf {
                  source = “gnomAD.NFE_AF”
                  cutoff = 0.01
            }
      }
```

```
	snv2 {
		type = snv
		maf {
			source = pooled
			cutoff = 0.05
		}
	}
}
```

`type:` describes the method which will be used. Here we are using `snv` for single variant analysis. Please note that the method name is arbitrary. It is the `type` option decides what the method will be used. This allows user to use the same type of method multiple times, with different parameters (see above). If the user uses a method name that is the same as the type name, e.g. use snv for snv type, the parameters of the method can be omitted. For instance, in the demo.conf file, the following lines could be omitted.

```
	snv {
		type = "snv"
		maf {
			source = "pooled"
			cutoff = 0.01
		}
	}
```

But if the user uses another method name, like snv1 for snv, the type option should be specified:

```
method {
	list = [snv1] #or method1, whatever names other than snv
	snv1.type = snv
}
```

`maf.source:` describes how to obtain MAFs. The MAFs can sometimes be obtained from a key in the INFO field of the original VCF, or annotation can be performed using an external database, e.g., gnomAD. MAFs can also be calculated from the data, using all samples (pooled), or only controls (control). Caution should be used doing this, e.g. using MAF cutoffs obtained from controls from a dataset can increase type I errors.
`maf.cutoff:` describes the MAF cutoff(s). For single variant analysis, the variants with MAF lower than this cutoff will be ignored.

Question 3: Suppose you have performed the analysis including first 0 PC components, 1 PC component, 1 and 2 PC components and then 1, 2 and 3 PC components. You obtain the following values for lambda. How many PC components should you use for the analysis?

_____

PCs	Lambda
0	1.085
1	1.086
2	1.084

6. Rare variant aggregate association tests

The rare variant aggregate association analysis is performed by the below object which is included in demo.conf.

```
seqspark {
  project = demo
  partitions = 10
  pipeline = [ "annotation", "qualityControl", "association" ]
  input {...}
  output {...}
  annotation {...}
  qualityControl {...}
  association {
    trait {
      list = ["bmi"]
      bmi {
        binary = false
        covariates = ["sex", "age", "disease"]
        pc = 0
      }
    }
    method {
      list = ["snv", "cmc", "brv", "skat", "skato"]
      snv.maf.source = "SS_AF"
      cmc.maf.source = "SS_AF"
      brv.maf.source = "SS_AF"
      skat.maf.source = "SS_AF"
      skato.maf.source = "SS_AF"
    }
  }
}
```

In the `list` besides `snv` is now included five rare variant aggregate association tests: `cmc` the Combined Multivariate Collapsing;[2] `brv` Burden of Rare Variants;[3,4] `vt` Variable Threshold;[5] `skat` Sequence Kernel Association Test;[6] and `skato` SKAT-optimal.[7] It should be noted that for rare variant analysis, functional annotation using RefSeq is performed before the analysis. By default, all splice site, frameshift, missense and nonsense variants for each gene are analyzed that have a MAF below a threshold, i.e., $MAF < 0.01$. The user also can specify the MAF cut-off to be used in the analysis. It is also possible to annotate using additional databases, e.g. functional annotation databases such as CADD[8] and specify which variants to analyze e.g. $c.score > 10$ or weights can be used in the analysis for example weighting can be performed using CADD scores or MAFs from gnomAD.

Like the snv method, the rare variant aggregate association methods are also configurable. You can set the MAF cutoff, MAF source, and type the same way it is performed for the snv method. Some popular rare variant aggregate association methods with defaults, which can be changed, are listed below.

`cmc` method is of **cmc** type and fixed MAF. This test using binary coding for the rare variants within a region, i.e., 1 if a variant is present or 0 if the individuals does not carry a rare variant. If set the `maf.fixed` to false, this will become a VT method with the CMC coding.

`brv` is similar to the **cmc**, except it analyzes the counts of the number of rare variants in a region instead of using binary coding. Similar to CMC, BRV coding can also be used to perform the VT

method. There are also several weighting options, e.g., equal weight, beta distribution density function (as used by SKAT). You can use annotation values from the INFO field of the VCF or weights can be obtained from external databases, e.g. functional annotation.

`wss` is a **brv** type method, i.e., uses BRV coding for the variants, except that it uses wss weights, which is that it bases the weights on variant MAFs in the controls for case-control data all the entire sample for quantitative traits. Due to the use of internal weights empirical p-values should be obtained for this test.

`vt` can use either **cmc** or **brv** variant coding with `maf.fixed` set to false. This allows the test to maximize the test statistic over allele frequencies. Since more than one test is performed for a region/gene p-values are obtained empirically to adjust for multiple testing.

`skat` has a skat type, and additional parameters are put into a **misc** section. These parameters are compatible to those used by the SKAT R package.

`skato` has a skato type, and additional parameters are also put into a misc section. The parameters are compatible to those used by the SKAT R package.

For more information, please visit https://statgenetics.github.io/seqspark

```
cmc {
      type = cmc
      maf {
            source = pooled
            cutoff = 0.01
            fixed = true}}
brv {
      type = brv
      weight = equal
      maf {
            source = pooled
            cutoff = 0.01
            fixed = true}}
wss {
      type = brv
      weight = wss
      maf {
            source = pooled
            cutoff = 0.01
            fixed = true}}
vt {
      type = brv
      weight = equal
      maf {
            source = pooled
            cutoff = 0.05
            fixed = false}}
skat {
      type = skat
      weight = skat #or equal, wss, etc.
      maf {
            source = pooled
            cutoff = 0.01
```

```
      }
      misc {
            method = "davies"
            rho = 0.0
            kernel = "linear.weighted"
            weightBeta = [1, 25]}}
skato {
      type = skato
      weight = skat
      maf {
            source = pooled
            cutoff = 0.01}
      misc {
            method = "optimal.adj"
            rhos = []
            kernel = "linear.weighted"
            weightBeta = [1, 25]}}
```

7. Running SEQSpark and collecting results

Now that the configuration file is ready, the analysis can be performed:

```
seqspark demo.conf
```

An `demo` directory will be created, and the results are inside it:

`qc.txt` summarizing the genotype counts, variants counts, and Ti/Tv ratios for the dataset.

`pca.txt` contains the principal components values for each sample's genotypes for the 10 components.

A series of **`assoc_${trait}_${method}`** files contains the results of the association analysis for each trait and method combination.

Question 4: List the top 10 genes of smallest p-values using the BRV method with MAF cutoff set to <**0.005**. (Hint: You can add another `brv1` method and set `brv.maf.cutoff = 0.005` in the method object, and rerun the analysis)

Gene	P-value

Question 5: What are the p-values observed for gene *GSTT2B* when the data is analyzed using the CMC, BRV, SKAT, and SKATO rare variant association methods? For each method, what are the p-values using MAF cutoff 0.01 and 0.005? Hint: For each method, add another object in which the MAF is set to 0.005. e.g. To run CMC with two MAF cutoffs, you can use the following configuration:

```
list = [cmc, cmc2]
cmc.maf.source = "SS_AF"
cmc2 {type = cmc, maf { source = "SS_AF", cutoff = 0.005 }}
```

Method	MAF < 0.01	MAF < 0.005	Different (Yes/No)
CMC			
BRV			
SKAT			
SKATO			

Answers:

1. From the screen output, find the numbers:
    a. Number of genotypes before QC _293933_ after QC _285370___
    b. Number of variants before QC ____619___ after QC ___614____
    c. Number of rare variants defined by the gnomAD MAF __464_____
    d. Number of common variants defined by the gnomAD MAF __150___

2. From the screen output, find the numbers:
    a. Ti/Tv ratio for all variants before QC __3.49__ after QC _3.48___

3. Suppose you have performed the analysis with PC 0,1,2 term, respectively. You get the following table for the inflation coefficient lambda. How many PC should you continue to use for your analysis? Include O PC components. Including additional PC components does not change the lambda value.

PCs	Lambda
0	1.085
1	1.084
2	1.086

4. List the top 10 genes of smallest p-values using the BRV method with MAF cutoff set to <**0.005**.

Gene	P-value
SLC2A11	0.100
MIF	0.126
C22orf15	0.136
GSTT2B	0.146
DERL3	0.289
GSTT2	0.290
PIWIL3	0.303
GUCD1	0.310
GGT1	0.396
GGT5	0.443

5. What are the p-values for gene *GSTT2B* performing the analysis using CMC, BRV, SKAT, and SKATO? For every method, are the p-values different using MAF cutoffs 0.01 and 0.005?

Method	MAF < 0.01	MAF < 0.005	Different (Yes/No)
CMC	0.050	0.146	Yes
BRV	0.056	0.146	Yes
SKAT	0.691	0.703	Yes
SKATO	0.167	0.444	Yes

Reference:

1. Zhang D, Zhao L, Li B, He Z, Wang GT, Li DJ, and Leal SM SEQSpark: A complete analysis tool for large-scale rare variant association studies using whole genome and exome sequence data.

2. Li, B., and Leal, S.M. (2008). Methods for Detecting Associations with Rare Variants for Common Diseases: Application to Analysis of Sequence Data. Am. J. Hum. Genet. *83*, 311–321.

3. Morris, A.P., and Zeggini, E. (2010). An evaluation of statistical approaches to rare variant analysis in genetic association studies. Genet. Epidemiol. *34*, 188–193.

4. Auer, P.L., Wang, G., NHLBI Exome Sequencing Project, and Leal, S.M. (2013). Testing for Rare Variant Associations in the Presence of Missing Data. Genet. Epidemiol. *37*, 529–538.

5. Price, A.L., Kryukov, G.V., de Bakker, P.I.W., Purcell, S.M., Staples, J., Wei, L.-J., and Sunyaev, S.R. (2010). Pooled Association Tests for Rare Variants in Exon-Resequencing Studies. Am. J. Hum. Genet. *86*, 832–838.

6. Wu, M.C., Lee, S., Cai, T., Li, Y., Boehnke, M., and Lin, X. (2011). Rare-Variant Association Testing for Sequencing Data with the Sequence Kernel Association Test. Am. J. Hum. Genet. *89*, 82–93.

7. Lee, S., Wu, M.C., and Lin, X. (2012). Optimal tests for rare variant effects in sequencing association studies. Biostatistics *13*, 762–775.

8. Kircher, M., Witten, D.M., Jain, P., O'Roak, B.J., Cooper, G.M., and Shendure, J. (2014). A general framework for estimating the relative pathogenicity of human genetic variants. Nat. Genet. *46*, 310–315.