Published in final edited form as:
Pac Symp Biocomput. 2020 ; 25: 523–534.

Hadoop and PySpark for reproducibility and scalability of genomic sequencing studies

NICHOLAS R. WHEELER,
Cleveland Institute for Computational Biology, Department of Population and Quantitative Health Sciences, Case Western Reserve University, Wolstein Research Building, 2103 Cornell Road Cleveland OH 44106, USA

PENELOPE BENCHEK,
Cleveland Institute for Computational Biology, Department of Population and Quantitative Health Sciences, Case Western Reserve University, Wolstein Research Building, 2103 Cornell Road Cleveland OH 44106, USA

BRIAN W. KUNKLE,
John P. Hussman Institute for Human Genomics, Miller School of Medicine, University of Miami, 1501 NW 10th Ave, Miami, FL 33136, USA

KARA L. HAMILTON-NELSON,
John P. Hussman Institute for Human Genomics, Miller School of Medicine, University of Miami, 1501 NW 10th Ave, Miami, FL 33136, USA

MIKE WARFE,
Cleveland Institute for Computational Biology, Center for Advanced Research Computing, University Technology, Case Western Reserve University, Wolstein Research Building, 2103 Cornell Road Cleveland OH 44106, USA

JEREMY R. FONDRAN,
Cleveland Institute for Computational Biology, Center for Advanced Research Computing, University Technology, Case Western Reserve University, Wolstein Research Building, 2103 Cornell Road Cleveland OH 44106, USA

JONATHAN L. HAINES,
Cleveland Institute for Computational Biology, Department of Population and Quantitative Health Sciences, Case Western Reserve University, Wolstein Research Building, 2103 Cornell Road Cleveland OH 44106, USA

WILLIAM S. BUSH
Cleveland Institute for Computational Biology, Department of Population and Quantitative Health Sciences, Case Western Reserve University, Wolstein Research Building, 2103 Cornell Road Cleveland OH 44106, USA

nrw16@case.edu.

6. Availability
Code and examples are available at http://www.icompbio.net/resources/software-and-downloads/

Abstract

Modern genomic studies are rapidly growing in scale, and the analytical approaches used to analyze genomic data are increasing in complexity. Genomic data management poses logistic and computational challenges, and analyses are increasingly reliant on genomic annotation resources that create their own data management and versioning issues. As a result, genomic datasets are increasingly handled in ways that limit the rigor and reproducibility of 1many analyses. In this work, we examine the use of the Spark infrastructure for the management, access, and analysis of genomic data in comparison to traditional genomic workflows on typical cluster environments. We validate the framework by reproducing previously published results from the Alzheimer's Disease Sequencing Project. Using the framework and analyses designed using Jupyter notebooks, Spark provides improved workflows, reduces user-driven data partitioning, and enhances the portability and reproducibility of distributed analyses required for large-scale genomic studies.

Keywords

Big Data; Spark; Whole-genome Sequence; Rare-variants

1. Introduction

1.1. The Rapid Scale-up of Genomic Data

The scale of modern genomic studies has shifted from the early days of genome-wide association studies with 500,000 to 1 million genetic variants on a few thousand people (IMSGC, 2007) to imputed studies capturing tens of millions of variants (Lambert et al., 2013), to whole-genome sequencing studies that routinely capture in excess of 100 million genetic variants on several thousand people (C Yuen et al., 2017). While many Genome-Wide Association Study (GWAS) style analyses of this data are conceptually straightforward (Bush & Moore, 2012), the practical implementation of quality control procedures and basic regression analyses often increase in complexity with this scale of data due to computing requirements. For example, custom scripts are often needed to partition data across multiple nodes of a computing cluster, and the creation/destruction of many temporary files is often necessary which increases the analysis workload and the number of points of manipulation of the data. These practical details of data handling and processing are often omitted from methods sections of genomics publications, but this general problem is often addressed in descriptions of data workflows. Verma et al. nicely outline multi-step, parallelized imputation and quality control (QC) workflows used within the eMERGE network (Verma et al., 2014), and Reed et al. specifically outline the need for parallel processing and distributed algorithms for basic GWAS processing within an R framework (Reed et al., 2015). The practical issues of data partitioning and manipulation often slow the pace of analyses, complicate the code needed to complete analyses, and increase the likelihood of data handling errors, thus reducing the rigor and reproducibility of many modern genomic analyses.

1.2. Increasing Dependencies on External Information

GWAS-style analyses are typically performed on the variant level, examining the independent effect of each non-reference allele in the dataset. Genomic sequencing studies, however, are designed to test the ‘rare variant hypothesis’ – that a series of low frequency, dominantly and independently acting variants across the genome each confer a moderate but readily detectable increase in disease risk (Bodmer & Bonilla, 2008; Schork, Murray, Frazer, & Topol, 2009). Because they have low frequency, there is limited power to see frequency differences between cases and controls in a population-based study. In fact, studies to date suggest that one-third of variants identified will be singletons (occurring in only one person) and doubletons (occurring in two people) (Bush et al., 2016; Butkiewicz, Blue, et al., 2017).

To address the issue of statistical power, rare-variants are often grouped into ‘functional’ units to generate a test statistic. These tests often rely on external data sources to define units of analysis (Lee, Abecasis, Boehnke, & Lin, 2014); for example, burden and collapsing tests group low-frequency variants together typically to perform a gene-based test. Gene databases -- even the concept of a gene -- have changed substantially over the last 20 years (Gerstein et al., 2007), and the choice of gene and transcript definitions have an impact on gene-based tests (McCarthy et al., 2014). Similarly, criteria for defining “loss of function” variants (Butkiewicz, Haines, & Bush, 2017; MacArthur et al., 2012), and methods for assessing and quantifying variant impact can vary (Kircher et al., 2014). These new biology-driven analyses create dependencies on external information (e.g. transcript reference, annotation database versions, scoring approaches, etc). The ability to manage, store, and provide version control for these external resource dependencies has now become critical to reproduce published genomic analyses.

1.3. Limitations of Replication Analyses make Rigor and Reproducibility Critical

The reproducibility of associations from GWAS has relied on strict control of type I error rates and replication of initial findings in an independent dataset. Rare variant studies have adopted similar corrections for multiple hypothesis testing (often for the number of genes used in a burden test), but replication of rare variant associations using gene-based approaches is not as straightforward (Auer & Lettre, 2015; Liu & Leal, 2010). Due to their rarity, alleles present in the discovery samples may not exist in the replication samples and vice-versa. While under some disease models, this fact improves the ability to replicate a gene-level association (Liu & Leal, 2010), it also points out the need for consistency of analyses across multiple genomic datasets. Given that some rare alleles will be population-specific, as in the discovery of an LDLR variant unique to Sardinians (Sanna et al., 2011), and the potential to identify globally unique alleles in association with disease, investigators can no longer rely on strict replication criteria to judge the reproducibility of genomic findings.

Due to the potential increases in data handling and manipulation required for genomic studies of scale, increases in data management for information beyond the primary dataset, and the limited ability to replicate certain findings, having a computational framework for managing analyses on this scale is more critical than ever before. While many new statistical approaches for the analysis of rare-variant datasets have been developed, little effort has been made to address issues of data scaling and management for sequence-based studies. In this paper, we describe an evaluation of the Apache Spark framework for supporting scalable, reproducible analyses of rare variant datasets. To our knowledge, this is the first application of Spark that allows the use of R and Python-based functions for genome-wide unit-based testing.

2. Datasets and Methods

2.1. Study Samples, Variants, and Data Scaling

Workflows developed in this paper are motivated by analyses of data generated by the Alzheimer's disease Sequencing Project (ADSP). For the Discovery Phase of this project, details of the study design (Beecham et al., 2017) and genotype quality control (Naj et al., 2018) have been previously described. From whole-genome sequencing of 578 individuals from 111 densely affected late-onset Alzheimer's Disease families, a dataset containing 27,896,774 distinct variants was generated (Butkiewicz, Blue, et al., 2017). Expanding this dataset from 578 to 1005 individuals increased the variant count to 53,041,134, and a further expansion to 4795 individuals increases the variant count again to 123,739,190 – an increase of approximately 21,000 variants per sample added. With additional multi-ethnic samples being sequenced as part of the ADSP Follow-up Study, we anticipate dataset sizes approaching 500 million variants from approximately 20,000 whole-genome sequences. For our evaluations, we accessed the ADSP Discovery Whole-Exome Sequencing dataset consisting of 5,740 late-onset Alzheimer's disease cases, and 5,096 cognitively normal controls with calls for 1,586,703 variants.

2.2. Variant Annotation Resources

Variants identified by the ADSP are annotated using a custom annotation pipeline (Butkiewicz, Blue, et al., 2017), which is a modification of the Ensembl Variant Effect Predictor (VEP) (Yourshaw, Taylor, Rao, Martín, & Nelson, 2015). Information about variant frequency (Glusman, Caballero, Mauldin, Hood, & Roach, 2011) and scores predicting variant functional impact (Kircher et al., 2014; Maurano et al., 2015; Xiong et al., 2015) are also annotated. For gene-based tests, annotations are critical for assigning variants to genes, and for providing classifications of variant impact (i.e. high, moderate, low, or modifier). As variant annotations are relative to specific *transcripts* rather than genes, we collapse multiple transcript-specific variant annotations to a *most damaging consequence* on a gene level. While this practice is the most canonical approach, we have previously shown that approximately 25% of gene unit tests would be influenced by using variant annotations relative to transcripts expressed in disease-relevant tissues (Butkiewicz, Blue, et al., 2017). With the expansion from whole-exome to whole-genome sequencing, annotations for non-genic regions have become more important.

3. Workflow

3.1. The Apache Spark Ecosystem

Given the increasing scale of genomic datasets, the increasing reliance on external annotation resources, and the need for streamlined and reproducible analyses, we explored an analysis workflow within the Apache Spark ecosystem (Zaharia et al., 2016). Spark provides an interface for data analysis and programming that utilizes an entire computing cluster with built-in data parallelism and fault tolerance. Data is ingested into the Hadoop Distributed File System (HDFS), which automatically partitions large files into redundant segments over multiple nodes of a computing cluster. The entire dataset, seamlessly partitioned across the cluster's nodes, can be accessed programmatically as a single Spark DataFrame instance. A variety of Spark functions can then be applied to the DataFrame for data processing, which is inherently parallelized so that each computing node has local access to its own partitions of the complete dataset. These data processing operations are compatible with traditional Structured Query Language (SQL), more sophisticated machine learning and graph-based operations, or custom functions. For genomic data storage and analysis, ingesting a large variant call format (VCF) file through a single command accomplishes the equivalent of scripts that segment the VCF into individual files by chromosome or individual. The complete dataset is then programmatically accessible by issuing a single function call.

The Spark framework is deployed on a dedicated Hadoop Cluster consisting of 16 data/compute nodes, with an overall total of 1.28 TB of RAM and 144 CPU-cores for parallel processing. This cluster is configured with standard Apache Spark (version 2.1.0) and Hadoop features, including Cloudera server manager (version 5.7.0), HDFS with a capacity of 288 TB, and the YARN/MapReduce platform for large scale data processing.

3.2. Genotype Storage and Retrieval

To provide a set of genotype storage and quality control operations, we used the open source software Hail (version 0.1–74bf1eb) developed by the Neale Lab at the Broad Institute (Hail, https://github.com/hail-is/hail/tree/0.1). Hail operates on top of Spark, and provides extensive, efficient functions for processing genomic data. Genomic data is imported from VCF and Plink-compatible files stored on the HDFS, and are converted to the Hail Variant Dataset (VDS) representation. Both sample and variant attributes can be easily assigned to VDS objects, allowing rapid retrieval of data subsets. For example, storing genomic data from the 1000 Genomes project (Consortium, 2012) provides the capability to extract VCF and Plink-compatible files that contain both sample and variant subsets within minutes.

Hail provides functions for analysis, quality control, and data manipulation, including single-variant statistical analyses, principal component analysis for adjusting for population stratification, among others. Most critically, Hail also provides interoperability with Python and Spark libraries, allowing the generation of Spark DataFrames from collections of genetic variants. These capabilities provide the ability to use R and Python packages for analysis of segmented genomic data, along with all functions available to Spark DataFrames.

3.3. Annotation Storage and Processing

Hail provides an extensive collection of annotation resources that can be applied to genomic datasets. These resources are instantiated as a Hail KeyTable, and when applied add fields to the original VDS files. Depending on the size and scale of the variant annotation, directly annotating the VDS can inflate file sizes resulting in less efficient operations. Given the need for the most current genomic annotation information, and the desire for state-of-the-art definitions of regulatory (and other) genomic elements, we used an annotation processing pipeline (external to Hail) within the Spark framework.

Spark provides built-in support for creating DataFrames from comma-/tab-separated value or JavaScript Object Notation (JSON) text files. JSON-formatted files are especially useful in the Spark framework as the information structure is preserved and accessible in query operations without additional data parsing operations. Based on our published annotation pipeline (Butkiewicz, Blue, et al., 2017), we first generated variant-level annotations using VEP, creating a JSON file containing variant consequence predictions relative to all Ensembl transcripts. This ‘everything’ annotation was instantiated as a Spark DataFrame, which was then subsequently processed to produce derived annotations. For example, a User-Defined Function (UDF) was developed to create a ‘most damaging consequence per gene’ for each variant, and also supports producing tissue-specific variant consequence predictions using tissue-transcript reference sets like those from the Genotype-Tissue Expression project (GTEx) (Mele et al., 2015). The DataFrame resulting from the UDF is then used to generate a Hail KeyTable by generating a primary key column corresponding to the variant ID. This Hail KeyTable can be used to identify variants meeting annotation criteria and extract genotypes needed for a given analysis.

3.4. Flexible Gene-based Analyses using Spark User Defined Aggregation Functions

While basic burden tests are supported within Hail, most rare-variant tests, such as the sequence kernel association test (SKAT) (Wu et al., 2011) and family-based rare-variant tests (Svishcheva, Belonogova, & Axenovich, 2014), are implemented in either the R statistical environment or Python. These tests are typically factored to accept a single unit’s worth of genomic data as input (e.g. all variants within a single gene). Currently, the Hail framework does not support flexible user-defined code for gene- or unit-based analyses, limiting analyses to those explicitly implemented in Hail.

To support the broad array of unit-based statistical approaches implemented in R and Python within the Spark framework, we created a User Defined Aggregate Function (UDAF) to generate unit-level genotype datasets that can be passed to R and Python functions (Figure 1). Our UDAF can be used to conduct gene-based analyses using the SeqMeta package employed by the ADSP analyses of the WES data. We have also tested analyses using elastic net regression, and in general our UDAF can be easily adapted to support any R or Python function that accepts a genotype matrix as input. Genotype data stored within Hail was first processed to group data for variants associated by an aggregating factor, in our case a gene identifier. Once stored in this way, the entire genome’s collection of genes can be processed in parallel, applying custom code across each group of associated variants’ genotypes. While the initial data aggregation step can be time intensive, running subsequent analytics steps on the grouped data proceeds rapidly. This works well for the development and tuning of analytical methods, and enables truly interactive dataset exploration to proceed genome-wide.

3.5. Increasing Reproducibility of Analyses

The quality control (QC) workflows outlined in figure 5 of Verma et al. enumerate seven steps from raw data to analysis, each of which likely required writing and reading new Plink-compatible files, and a change to any one of the QC criteria would require regeneration of each file, requiring file naming strategies, scripts for file cleanup, and careful versioning. Similarly, the analysis steps outlined in figure 1 of Reed et al. nicely document the R code needed for four analysis steps which rely on the *doParallel* package to distribute association tests over multiple cores. This step was specifically modified by the authors to allow for parallel processing, and other steps of the workflow would require similar efforts for larger scales of data. Each these steps involves file read and write operations, file labeling/tracking, and could involve several thousand of temporary sub-files (totaling more than the original gigabyte/terabyte scale data), all to be managed by the user and their developed scripts. Each of these file operations represents a potential point of failure. In contrast, the Spark-based workflow provides seamless dataset partitioning, and computation occurs in parallel across the cluster. This creates a fast and more straightforward single step process for users, which allows easier development, testing, and verification.

Code for these Spark-based workflows is developed and deployed within Jupyter Notebooks, which provides features for documentation and ease of readability. Jupyter Notebooks are an open-source web application that allows users to create and share documents that contain live code, equations, and figures alongside formatted narrative text. As a hybrid of a script and a document, Jupyter Notebooks allow entire sections of code to be re-executed on the Spark cluster, typically in real-time. Analysis results can be written to files, or passed as DataFrames to R and Python libraries for immediate visualization. The use of Jupyter Notebooks is generally accepted as a way to share and duplicate analysis workflows, and can interoperate with R and Python packages for custom code execution. These highly portable approaches to utilizing code are well-suited to scientific collaboration and reproducibility.

4. Results

4.1. Validation of the Implementation

To validate the Spark framework, we accessed the final QC+ version of the ADSP Discovery Whole-Exome Sequencing dataset, as described in (Bis et al., 2018). Following steps outlined in figure 2, we ingested VCF files containing SNV and short INDEL calls, along with their variant annotations; we then partitioned this collection of genetic variants using our custom UDF, and performed gene-wise seqMeta analyses. All analyses were conducted with software packages and settings congruent to Bis et al, and a Manhattan plot of the published results alongside results from the Spark implementation are shown in figure 3.

4.2. Approximate Runtimes and Scalability

We compared the Spark implementation to the actual analysis workflow used by Bis et al. Using a traditional approach of splitting jobs over a scheduled cluster environment, the entire job from start to finish has an estimated max wall time of approximately 19 hours, assuming hardware equivalent to our cluster environment (number of nodes, etc). The Spark approach as currently configured requires a maximum of approximately 40 hours to execute from start to finish on the Bis et al dataset. The creation of a Spark dataframes from annotation files requires 18 minutes, and creation of Hail VDS files from VCFs requires 10 minutes. The lengthy step in the process is the generation of aggregated data tables from the genotypes, which with the current configuration requires 39 hours to process. Once performed, this step returns a single Spark object containing all genotypes and aggregation information written to HDFS, and can be used to support any downstream gene-based analysis, or multiple variations of a single analysis, with a vastly shorter runtime. Once aggregated, the entire seqMeta analysis of all genes requires approximately 42 minutes or less, which allows for multiple genome-wide runs using different annotation or filtering criteria. While we do not have precise timings for each step of the traditional analysis workflow, the bulk of the allocated analysis time is used for file I/O, extracting genotypes for each gene and writing these to files for subsequent analysis within R. For the traditional approach implemented in Bis et al., any repeat of the analysis requires a complete rerun of all computing steps. In contrast, the Spark-based framework is more modular, allowing selective execution of individual steps while maintaining a verified state of the analysis.

4.3. Advantages Over Traditional Implementations

While there currently is no timing advantage to the Spark approach, there are a few clear advantages in terms of the flexibility of the workflow. Unlike using Hail alone, our workflow allows users to incorporate the extensive library of existing R and Python packages that support various gene- and unit-based analyses. Because we have employed the Spark framework, the functions for performing gene-wise analyses have redundancy and operational integrity. Unlike a scheduled cluster environment where jobs are queued, must be tracked, and results must be aggregated when jobs are fully completed, the Spark environment provides process redundancy where any failed processes are tracked and dynamically re-executed in case of failure. Results are aggregated and returned as a single distributed dataframe object. The Spark workflow effectively replaces process and file tracking tasks usually performed explicitly by an end user (within scripts) with features inherent to the Spark environment.

A second advantage of the Spark approach is that aggregated variant tables can be modified by adding additional annotations or covariates. In the traditional workflow, this would require scripts to process each of the written files to add these columns post-hoc, or more likely, a revision of the analysis workflow and a complete re-execution. In fact, any change to analysis parameters, variant QC criteria, or variant annotations would require a complete re-execution of a traditional workflow, whereas with the Spark workflow only a small fraction must be rerun.

Finally, replicating the analysis implemented in the traditional workflow would require extensive customization (or a complete re-implementation) before it could be run on a local cluster environment. Adaptations are needed to account for different cluster schedulers, software package availability and version control, and file locations/disk usage. In contrast, the Spark workflow can be executed on an equivalent Hadoop environment and Spark implementation using only code contained with Juypter Notebooks and base Hail and PySpark functionality. While this does not completely eliminate the need to customize a computing environment (as Spark setup and configuration is required), it is a step closer toward easy transferability and reproducibility of analysis tasks whose size necessitate execution within a distributed processing framework. Furthermore, as large-scale studies are beginning to provide access to data exclusively through cloud computing environments, frameworks like this will be necessary for distributed analysis tasks.

5. Conclusions

We have explored the use of Spark for implementing a typical analysis of genomic sequence data, and found the implementation to be a flexible approach for analyses of large-scale genomic data that provides built-in parallelization, minimizes data handling, and improves reproducibility. By storing variant annotation information alongside study participant genotypes, the Spark framework provides a route by which custom variant annotation information can be rapidly integrated to support new variant filters, or variant groupings to test within unit-based association analyses. For many existing datasets, the distributed Spark infrastructure described here is not necessary; however as sequencing studies continue to add samples and expand the scope of variant capture, and the breadth of genotype imputation panels grow, future genomic studies will require a reproducible parallel processing framework for statistical analysis.

7. Acknowledgements

This work was supported by the NIH (U54 AG052427, UF01 AG07133). Support for computational infrastructure was provided by the CWRU University Technology division.

References

Auer PL, & Lettre G (2015). Rare variant association studies: considerations, challenges and opportunities. Genome Medicine, 7(1), 16 10.1186/s13073-015-0138-2 [PubMed: 25709717]

Beecham GW, Bis JC, Martin ER, Choi S-H, DeStefano AL, van Duijn CM, … Schellenberg G (2017). The Alzheimer's Disease Sequencing Project: Study design and sample selection. Neurology Genetics, 3(5), e194 10.1212/NXG.0000000000000194 [PubMed: 29184913]

Bis JC, Jian X, Kunkle BW, Chen Y, Hamilton-Nelson KL, Bush WS, … Farrer LA (2018). Whole exome sequencing study identifies novel rare and common Alzheimer's-Associated variants involved in immune response and transcriptional regulation. Molecular Psychiatry. 10.1038/s41380-018-0112-7

Bodmer W, & Bonilla C (2008). Common and rare variants in multifactorial susceptibility to common diseases. Nature Genetics, 40(6), 695–701. 10.1038/ng.f.136. [PubMed: 18509313]

Bush WS, Crosslin DR, Obeng AO, Wallace J, Almoguera B, Basford MA, … Ritchie MD (2016). Genetic Variation among 82 Pharmacogenes: the PGRN-Seq data from the eMERGE Network. Clinical Pharmacology and Therapeutics. 10.1002/cpt.350

Bush WS, & Moore JH (2012). Chapter 11: Genome-Wide Association Studies. PLoS Computational Biology, 8(12), e1002822. 10.1371/journal.pcbi.1002822

Butkiewicz M, Blue E, Leung F, Jian X, Marcora E, Renton A, … Bush WS (2017). Functional Annotation of Genomic Variants in studies of Late-Onset Alzheimer's Disease. Bioinformatics (Oxford, England), In Press.

Butkiewicz M, Haines JL, & Bush WS (2017). Introducing COCOS: codon consequence scanner for annotating reading frame changes induced by stop-lost and frame shift variants. Bioinformatics, 33(10), btw820. 10.1093/bioinformatics/btw820

C Yuen RK, Merico D, Bookman M, L Howe J, Thiruvahindrapuram B, Patel RV, … Scherer SW (2017). Whole genome sequencing resource identifies 18 new candidate genes for autism spectrum disorder. Nature Neuroscience, 20(4), 602–611. https://doi.org/10.1038/nn.4524 [PubMed: 28263302]

Consortium T 1000 G. P. (2012). An integrated map of genetic variation from 1,092 human genomes. Nature, 135(V), 0–9. 10.1038/nature11632

Gerstein MB, Bruce C, Rozowsky JS, Zheng D, Du J, Korbel JO, … Snyder M (2007). What is a gene, post-ENCODE? History and updated definition. Genome Research, 17(6), 669–681. 10.1101/gr.6339607 [PubMed: 17567988]

Glusman G, Caballero J, Mauldin DE, Hood L, & Roach JC (2011). Kaviar: an accessible system for testing SNV novelty. Bioinformatics, 27(22), 3216–3217. 10.1093/bioinformatics/btr540 [PubMed: 21965822]

IMSGC. (2007). Risk alleles for multiple sclerosis identified by a genomewide study. The New England Journal of Medicine, 357(9), 851–862. 10.1056/NEJMoa073493 [PubMed: 17660530]

Kircher M, Witten DM, Jain P, O'Roak BJ, Cooper GM, & Shendure J (2014). A general framework for estimating the relative pathogenicity of human genetic variants. Nature Genetics, 46(3), 310–315. 10.1038/ng.2892 [PubMed: 24487276]

Lambert J-C, Ibrahim-Verbaas CA, Harold D, Naj AC, Sims R, Bellenguez C, … Amouyel P (2013). Meta-analysis of 74,046 individuals identifies 11 new susceptibility loci for Alzheimer's disease. Nature Genetics, 45(12), 1452–1458. 10.1038/ng.2802 [PubMed: 24162737]

Lee S, Abecasis GR, Boehnke M, & Lin X (2014). Rare-Variant Association Analysis: Study Designs and Statistical Tests. The American Journal of Human Genetics, 95(1), 5–23. 10.1016/j.ajhg.2014.06.009 [PubMed: 24995866]

Liu DJ, & Leal SM (2010). Replication strategies for rare variant complex trait association studies via next-generation sequencing. American Journal of Human Genetics, 87(6), 790–801. 10.1016/j.ajhg.2010.10.025 [PubMed: 21129725]

MacArthur DG, Balasubramanian S, Frankish A, Huang N, Morris J, Walter K, … Tyler-Smith C (2012). A systematic survey of loss-of-function variants in human protein-coding genes. Science (New York, N.Y.), 335(6070), 823–828. 10.1126/science.1215040

Maurano MT, Haugen E, Sandstrom R, Vierstra J, Shafer A, Kaul R, & Stamatoyannopoulos JA (2015). Large-scale identification of sequence variants influencing human transcription factor occupancy in vivo. Nature Genetics, 47(12), 1393–1401. 10.1038/ng.3432 [PubMed: 26502339]

McCarthy DJ, Humburg P, Kanapin A, Rivas MA, Gaulton K, Cazier J-B, & Donnelly P (2014). Choice of transcripts and software has a large effect on variant annotation. Genome Medicine, 6(3), 26 10.1186/gm543 [PubMed: 24944579]

Mele M, Ferreira PG, Reverter F, DeLuca DS, Monlong J, Sammeth M, … Guigo R (2015). The human transcriptome across tissues and individuals. Science, 348(6235), 660–665. 10.1126/science.aaa0355 [PubMed: 25954002]

Naj AC, Lin H, Vardarajan BN, White S, Lancour D, Ma Y, … DeStefano AL (2018). Quality control and integration of genotypes from two calling pipelines for whole genome sequence data in the Alzheimer's disease sequencing project. Genomics. 10.1016/j.ygeno.2018.05.004

Reed E, Nunez S, Kulp D, Qian J, Reilly MP, & Foulkes AS (2015). A guide to genome-wide association analysis and post-analytic interrogation. Statistics in Medicine, 34(28), 3769–3792. 10.1002/sim.6605 [PubMed: 26343929]

Sanna S, Li B, Mulas A, Sidore C, Kang HM, Jackson AU, … Abecasis GR (2011). Fine Mapping of Five Loci Associated with Low-Density Lipoprotein Cholesterol Detects Variants That Double the Explained Heritability. PLoS Genetics, 7(7), e1002198. 10.1371/journal.pgen.1002198

Schork NJ, Murray SS, Frazer KA, & Topol EJ (2009). Common vs. rare allele hypotheses for complex diseases. Current Opinion in Genetics & Development, 19(3), 212–219. 10.1016/j.gde.2009.04.010 [PubMed: 19481926]

Svishcheva GR, Belonogova NM, & Axenovich TI (2014). FFBSKAT: fast family-based sequence kernel association test. PloS One, 9(6), e99407. 10.1371/journal.pone.0099407

Verma SS, de Andrade M, Tromp G, Kuivaniemi H, Pugh E, Namjou-Khales B, … Ritchie MD (2014). Imputation and quality control steps for combining multiple genome-wide datasets. Frontiers in Genetics, 5, 370 10.3389/fgene.2014.00370 [PubMed: 25566314]

Wu MC, Lee S, Cai T, Li Y, Boehnke M, & Lin X (2011). Rare-variant association testing for sequencing data with the sequence kernel association test. American Journal of Human Genetics, 89(1), 82–93. 10.1016/j.ajhg.2011.05.029 [PubMed: 21737059]

Xiong HY, Alipanahi B, Lee LJ, Bretschneider H, Merico D, Yuen RKC, … Frey BJ (2015). RNA splicing. The human splicing code reveals new insights into the genetic determinants of disease. Science (New York, N.Y.), 347(6218), 1254806. 10.1126/science.1254806

Yourshaw M, Taylor SP, Rao AR, Martín MG, & Nelson SF (2015). Rich annotation of DNA sequencing variants by leveraging the Ensembl Variant Effect Predictor with plugins. Briefings in Bioinformatics, 16(2), 255–264. 10.1093/bib/bbu008 [PubMed: 24626529]

Zaharia M, Franklin MJ, Ghodsi A, Gonzalez J, Shenker S, Stoica I, … Venkataraman S (2016). Apache Spark. Communications of the ACM, 59(11), 56–65. 10.1145/2934664

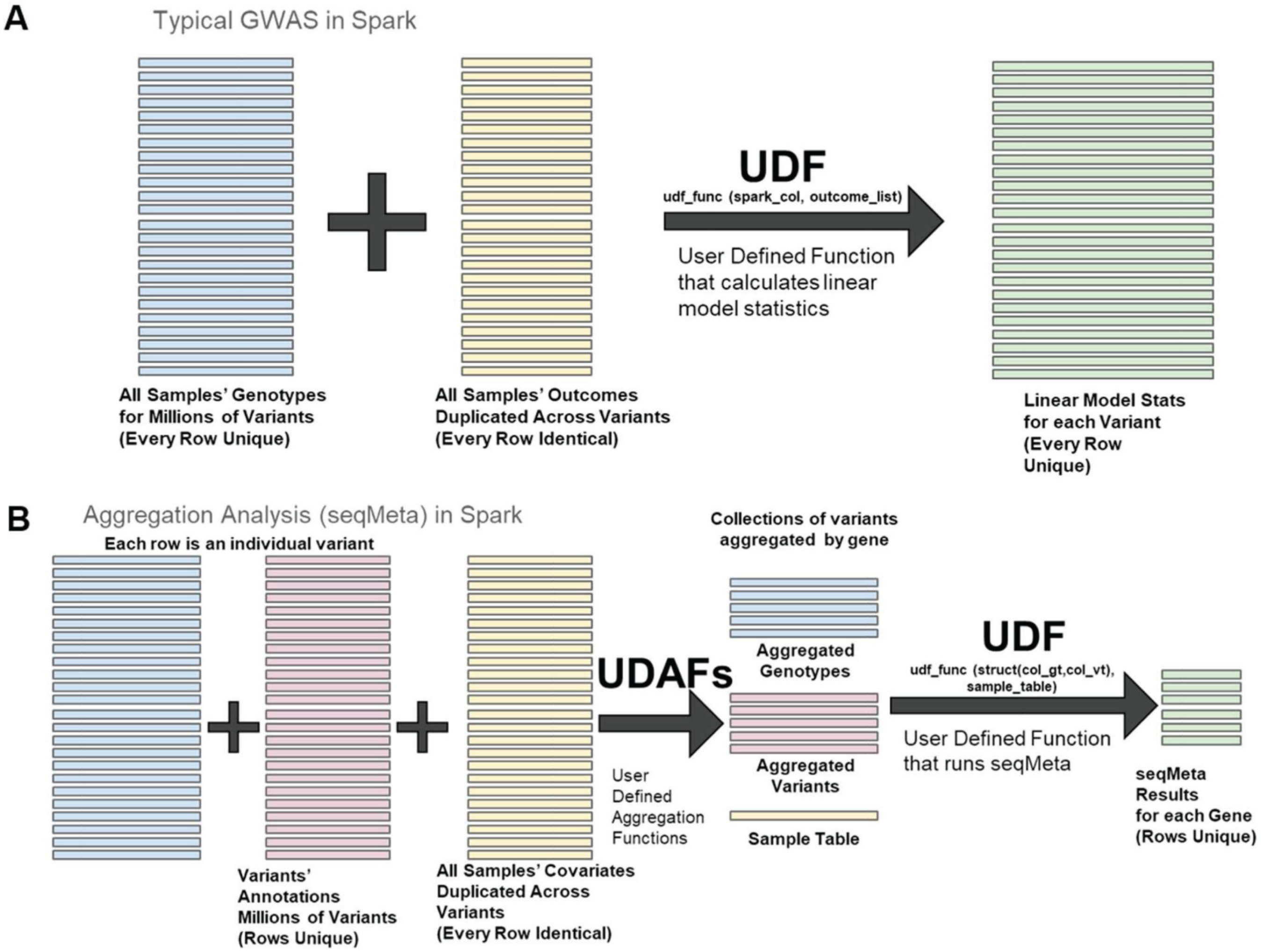
A
Typical GWAS in Spark
UDF
udf_func (spark_col, outcome_list)
User Defined Function that calculates linear model statistics
All Samples' Genotypes for Millions of Variants (Every Row Unique)
All Samples' Outcomes Duplicated Across Variants (Every Row Identical)
Linear Model Stats for each Variant (Every Row Unique)
B
Aggregation Analysis (seqMeta) in Spark
Each row is an individual variant
Collections of variants aggregated by gene
Aggregated Genotypes
UDAFs
User Defined Aggregation Functions
UDF
udf_func (struct(col_gt,col_vt), sample_table)
User Defined Function that runs seqMeta
Aggregated Variants
Sample Table
seqMeta Results for each Gene (Rows Unique)
Variants' Annotations Millions of Variants (Rows Unique)
All Samples' Covariates Duplicated Across Variants (Every Row Identical)

Figure 1.

Illustration of UDFs and UDAFs for producing typical GWAS association results (A) and gene-based aggregation test results (B). User Defined Aggregation Functions (UDAFs) partition the genotype data into frames that are programmatically accessible to User Defined Functions (UDFs), which can implement R and Python-based code. In this example, genotypes are aggregated by gene to produce results from the seqMeta R package.

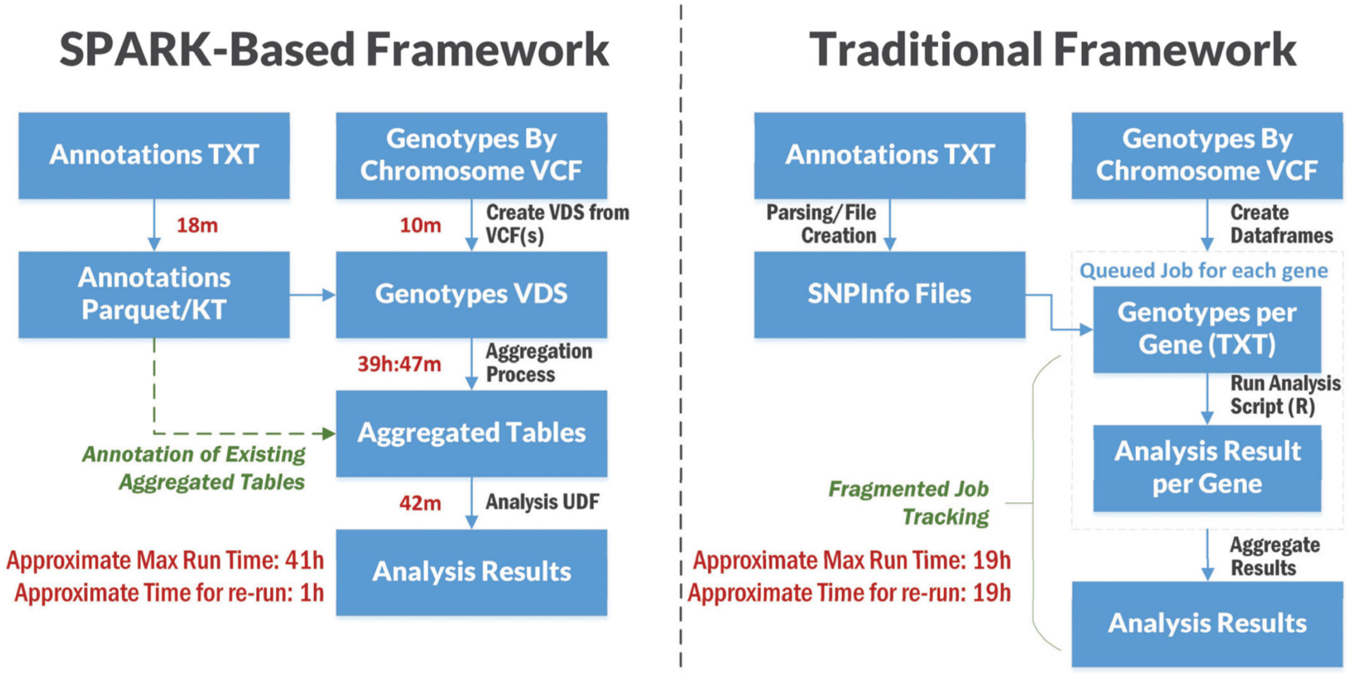

Figure 2.

Comparison of the Spark-based versus Traditional workflows for conducting rare-variant analyses. Approximate timings are noted in red (timings for individual steps of the traditional framework were unavailable). Green labels denote advantages of the Spark-based over the traditional framework.

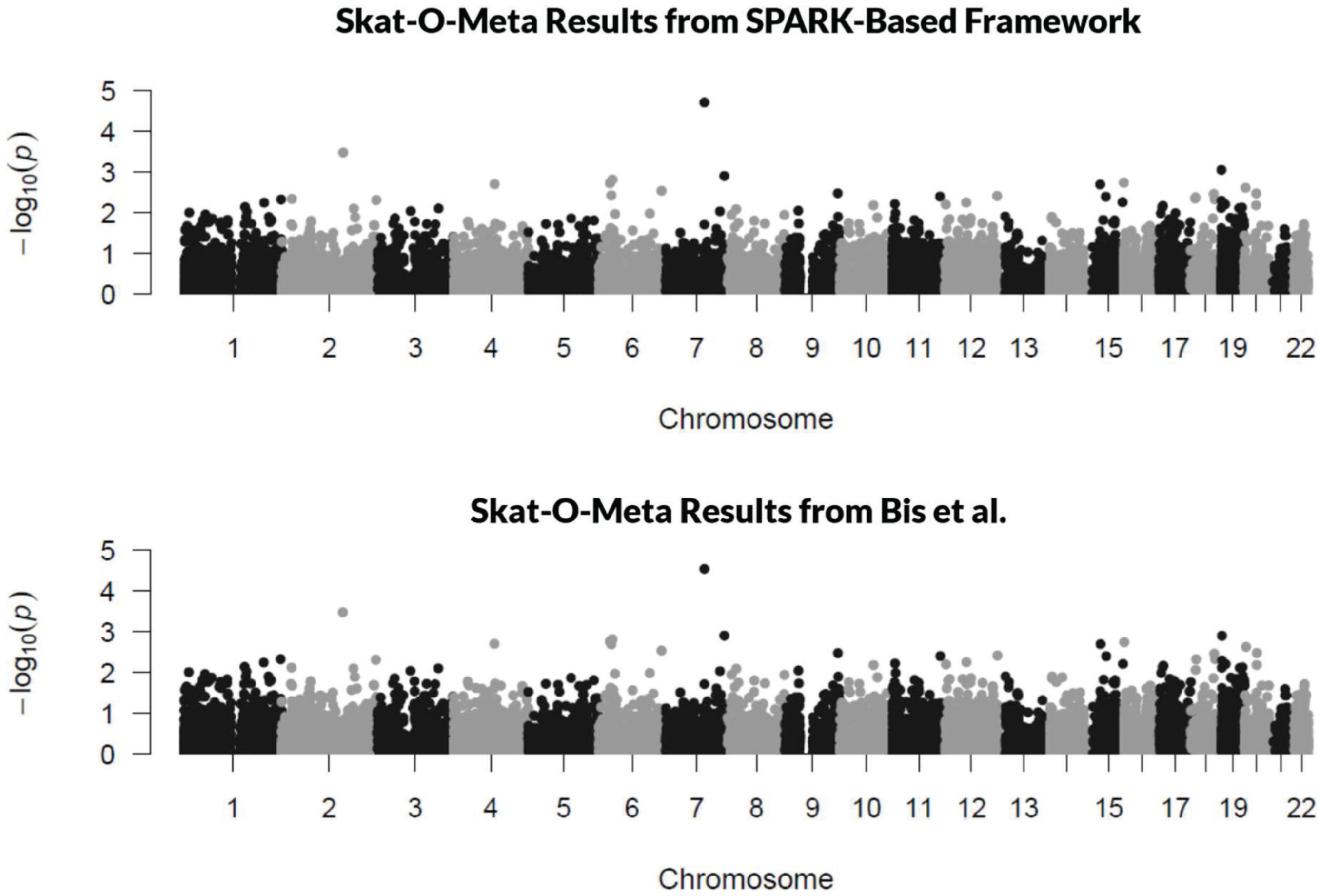

Figure 3.

Manhattan plots of Skat-O Meta-analysis results from the Spark-based implementation versus the results published in Bis et al.