Manuscript Title

This manuscript (permalink) was automatically generated from related-sciences/gwas-analysis-manuscript@e30095e on February 6, 2020.

Authors

- **John Doe**
  XXXX-XXXX-XXXX-XXXX · johndoe · johndoe
  Department of Something, University of Whatever · Funded by Grant XXXXXXXX

- **Jane Roe**
  XXXX-XXXX-XXXX-XXXX · janeroe
  Department of Something, University of Whatever; Department of Whatever, University of Something

Abstract

- UKBB
  - UKB Genotyping and Imputation Data Release March 2018 – FAQ
    - Imputed call set is 2.1T bgen
    - Measured call set is 92G bed
  - DETAILS AND CONSIDERATIONS OF THE UK BIOBANK GWAS
    - "Starting from the 487,409 individuals with phased and imputed genotype data, we filtered down to 337,199 QC positive individuals"
    - "We have run 1513 unique phenotypes"
    - "Over 92 million imputed autosomal SNPs were available for analysis. As of this writing, over half of these SNPs were undergoing re-imputation by the UK Biobank team due to mapping issues, leaving the ~40 autosomal million SNPs imputed from the Haplotype Reference Consortium as eligible for analysis. We further restricted to SNPs with minor allele frequency (MAF) > 0.1% and HWE p-value > 1e-10 in the 337,199 QC positive individuals, an INFO score > 0.8 (directly from UK Biobank), leaving 10.8 million SNPs for analysis."
  - http://biobank.ctsu.ox.ac.uk/crystal/label.cgi?id=100314
    - Genotypes and imputation therefrom for 488,000 participants
    - Exome sequences for 50,000 participants
    - Whole genome sequences for 50 participants
  - Quality control guide: http://www.ukbiobank.ac.uk/wp-content/uploads/2014/04/UKBiobank_genotyping_QC_documentation-web-1.pdf
    - UK Biobank team uses PLINK (for 1KG mostly), shellfish, and flashPCA for QC that is then used by NealeLab
    - This guide is about QC for interm release of 150k samples
      - Results in 152k samples by 806k variants
    - SNP QC is first performed only on European cohort
      - Using a homogeneous cohort is ideal just for initial SNP filters
    - SNPs are filtered based on heterozygosity (adjusted by regression using cohort since heterozygosity differs so much), missingness, and HWE
    - QC is done for on SNP in 4.8k sample batches (matching Affymetrix batches) so a SNP can fail QC in one batch and become all missing
    - SNP QC also done using ROH which is sequential heterozygous call rates
    - Sample QC adjusts for gender (finds .1% are wrong – has interesting aneuploidy figure)
- GPU Acceleration
  - Epistatic Interactions
    - SHEsisEpi, a GPU-enhanced genome-wide SNP-SNP interaction scanning algorithm, efficiently reveals the risk genetic epistasis in bipolar disorder
    - 1000× faster than PLINK: Combined FPGA and GPU accelerators for logistic regression-based detection of epistasis
      - Uses reparameterization of Newton's method in PLINK to make interaction model much faster to solve
  - IBS/IBD
    - PlinkGPU: A Framework for GPU Acceleration of Whole Genome Data Analysis
      - Ports pairwise IBS distance calculation in PLINK to GPU using block-wise calculations
      - Summary of PLINK capabilities:
        - Data management (filtering, merging, etc.)
        - Summary statistics (HWE, AF, call rates, heterozygosity, etc.)
        - Population stratification (IBS + MDS)
        - Association analysis (logreg, fisher/chisq)
        - Genetic relatedness
- Ecosystem Tools
  - bigsnpr

    - https://privefl.github.io/bigsnpr/articles/demo.html
    - Supports PLINK and UKBB BGEN reading
    - Efficient analysis of large-scale genome-wide data with two R packages: bigstatsr and bigsnpr
    - bigstatsr is related but more generic library for out-of-core matrix ops, PCA, and linear models
    - Supports imputation using custom XGBoost model as well as wrappers for calls out to PLINK + Beagle
- FaST-LMM
    - From 2011 Nature Methods paper FaST linear mixed models for genome-wide association studies
    - PySnpTools
        - https://fastlmm.github.io/PySnpTools/#snpreader-snpreader
        - Can read and write PLINK bed/ped data (it appears to be designed specifically for working efficiently with PLINK data)
        - Supports "DistributedBed" files with chunked PLINK datasets
        - Supports slicing of PLINK datasets (i.e. random IO)
        - Supports IO with npz, hdf5, and memmap files
        - Has efficient c++ reader/writer implementations
- scikit-allel
    - In memory EDA for genotyping data
    - Supports vcf, plink,
    - Has vcf_to_{npz,zarr,recarray} methods
    - Represents genotype calls as 3D uint8 arrays
        - GenotypeArray > This class represents data on discrete genotype calls as a 3-dimensional numpy array of integers. By convention the first dimension corresponds to the variants genotyped, the second dimension corresponds to the samples genotyped, and the third dimension corresponds to the ploidy of the samples.

          Each integer within the array corresponds to an allele index, where 0 is the reference allele, 1 is the first alternate allele, 2 is the second alternate allele, … and -1 (or any other negative integer) is a missing allele call. A single byte integer dtype (int8) can represent up to 127 distinct alleles, which is usually sufficient. The actual alleles (i.e., the alternate nucleotide sequences) and the physical positions of the variants within the genome of an organism are stored in separate arrays, discussed elsewhere.

          In many cases the number of distinct alleles for each variant is small, e.g., less than 10, or even 2 (all variants are biallelic). In these cases a genotype array is not the most compact way of storing genotype data in memory. This class defines functions for bit-packing diploid genotype calls into single bytes, and for transforming genotype arrays into sparse matrices, which can assist in cases where memory usage needs to be minimised. Note however that these more compact representations do not allow the same flexibility in terms of using numpy universal functions to access and manipulate data.
    - Supports bitpacking by collapsing the ploidy dimension (axis=2) into a single byte using 4 bits for each uint8
    - Phasing is supported by assuming the ordering in the ploidy dimension has meaning: > If the genotype calls are unphased then the ordering of alleles along the third (ploidy) dimension is

arbitrary

    - http://alimanfoo.github.io/2016/06/10/scikit-allel-tour.html > The scikit-allel genotype array convention is flexible, allowing for multiallelic and polyploid genotype calls. However, it is not very compact, requiring 2 bytes of memory for each call. A set of calls for 10,000,000 SNPs in 1,000 samples thus requires 20G of memory.

      One option to work with large arrays is to use bit-packing, i.e., to pack two or more items of data into a single byte. E.g., this is what the plink BED format does. If you have have diploid calls that are only ever biallelic, then it is possible to fit 4 genotype calls into a single byte. This is 8 times smaller than the NumPy unpacked representation.

      However, coding against bit-packed data is not very convenient. Also, there are several libraries available for Python which allow N-dimensional arrays to be stored using compression: h5py, bcolz and zarr. Genotype data is usually extremely compressible due to sparsity - most calls are homozygous ref, i.e., (0, 0), so there are a lot of zeros.

- File Formats
  - BGEN
    - BGEN: a binary file format for imputed genotype and haplotype data
    - https://www.cog-genomics.org/plink/2.0/formats#bgen
    - https://www.well.ox.ac.uk/~gav/bgen_format/
    - https://www.well.ox.ac.uk/~gav/bgen_format/spec/latest.html

Manuscript Outline

- Background
  - Scale of current genomic datasets
    - Stats on number of Biobanks
    - Stats on UKBB
    - Stats on growth from HapMap and 1KG for comparison
  - Focus of paper
    - Genotyping array data QC, control flow, and association modeling
    - Scalability problems with single-node, in-memory tools
    - Spark
  - Tutorial
    - Explain necessary GWAS toolkit operations
      - Data management (filtering, merging, etc.)
      - Liftover
      - Summary statistics (HWE, AF, call rates, heterozygosity, etc.)
      - Population stratification (IBS + MDS)
      - Association analysis (logreg, fisher/chisq)
      - Genetic relatedness
    - Introduce Marees 2018 paper and explain how tutorial is re-implemented with other tools
  - Tools
    - Primary: PLINK, Glow, Hail
    - Secondary: dask, bigsnpr, scikit-allel, pysnptools/fastlmm
      - Modin may be worth mentioning, even though out-of-core isn't really supported
  - Datasets
    - HapMap
    - 1KG
    - 3K rice genome
  - Data Formats

        - plink, bgen, parquet, Hail MT, vcf, hdf5/npz/zarr
        - Explain encoding and compression concerns
- Results
    - Code
        - Figure: flow chart of Marees analysis
            - Explain what the resulting code for this project does
        - Figure: side-by-side comparison of code examples
            - Primary differences:
                - Hail is an API over opaque data structures and implementations
                - Glow is simply a convention for representing genetic data in a Spark Dataset, with accompanying methods
                - PLINK is a gigantic list of parameterized commands for a single-core, single-node CLI
            - Operation differences
                - Operations that are similarly easy with all 3 toolkits:
                    - call rate filtering
                    - heterozygosity rate filtering
                    - HWE filtering
                    - AF filtering
                - LD Pruning: non-existent in Glow, very slow in Hail
                    - However, Glow does support inline calls to PLINK (albeit very awkwardly since PLINK is not streaming software)
                - Gender imputation: PLINK=automatic, Hail=automatic, Glow=manual
                    - PLINK and Hail both use inbreeding coefficient on X chromosome data
                    - Glow approach (and Marees 2018) look at homozygosity rate on X instead
                - Liftover
                    - Not available in PLINK
                    - Hail supports coordinate liftover only (not variant liftover)
                        - Requires a chain file for the destination reference genome
                    - Glow supports both coordinate liftover and variant liftover
                        - Requires a chain file and a reference fasta for the destination genome
                        - see the notebook at https://glow.readthedocs.io/en/latest/etl/lift-over.html for specification of chain and reference files
                - PCA for population stratification: simple in PLINK and Hail, non-existent in Glow
        - Usability:
            - Extending and learning algorithms from Glow source code is the easiest of all tools
            - Hail documentation is fairly thorough, though essentially no examples or answers are available outside of that documentation (or the discourse)
            - PLINK examples and documentation are both very extensive and easily found, however few PLINK workflows don't also include the need for some scripting language in the same pipeline for interpreting/visualizing results (those outputs are then often used to parameterize other PLINK commands, as exemplified by the tutorials in the project)
    - Data
        - Figure: File Format Comparison
            - Show comparison of file sizes by dataset and format
    - Performance
        - Figure: Times for operations by dataset and toolkit

References