Short communication

A database for efficient storage and management of multi panel SNP data

Eildert Groeneveld and Cong VC Truong

Department of Breeding and Genetic Resources, Institute of Farm Animal Genetics (FLI), Neustadt, Germany

Abstract

The fast development of high throughput genotyping has opened up new possibilities in genetics while at the same time producing immense data handling issues. A system design and proof of concept implementation are presented which provides efficient data storage and manipulation of single nucleotide polymorphism (SNP) genotypes in a relational database. A new strategy using SNP and individual selection vectors allows us to view SNP data as matrices or sets. These genotype sets provide an easy way to handle original and derived data, the latter at basically no storage costs. Due to its vector based database storage, data imports and exports are much faster than those of other SNP databases. In the proof of concept implementation, the compressed storage scheme reduces disk space requirements by a factor of around 300. Furthermore, this design scales linearly with number of individuals and SNPs involved. The procedure supports panels of different sizes. This allows a straight forward management of different panel sizes in the same population as it occurs in animal breeding programs when higher density panels replace previous lower density versions.

Keywords: SNP, multi panel, database, compressed storage, Perl, SQL

Abbreviations: GB: gigabyte, MB: megabyte, OGPR: one genotype per row, SNP: single nucleotide polymorphism

Archiv Tierzucht 56 (2013) 103, 1023-1027
doi: 10.7482/0003-9438-56-103

Received: 17 July 2013
Accepted: 19 November 2013
Online: 20 November 2013

Corresponding author:
Eildert Groeneveld; email: eildert.groeneveld@fli.bund.de
Department of Breeding and Genetic Resources, Institute of Farm Animal Genetics (FLI), Neustadt, Germany

Introduction

High throughput single nucleotide polymorphism (SNP) genotyping is evolving at a staggering rate, developing into a powerful tool in genetic analyses in all areas of biology. Although its promises are immense, data processing issues are associated with it. Dropping genotyping costs and ever increasing marker densities result in a huge increase in data volume, which seems to develop faster than the already impressive rate at which data storage costs have come down in the past. Thus, increasing data storage requirements are an issue.

SNP data analysis workflows often result in filtering SNPs thereby creating genotype subsets for different purposes. This process can lead to a dramatic increase in disk space requirement: while each derived dataset will always be smaller than the original, their sizes will still be of the same magnitude, quickly using huge amounts of disk space.

High throughput genotyping is used across species with diverse sets of genotyping panels of different sizes ranging from a few thousand to millions. New panels of higher densities may be used to retype the same individuals as it is done in regular animal breeding programs, resulting in ever growing datasets. Accordingly, information originating from panels of different densities will have to be managed and processed.

Space requirements of SNP data may be very large, resulting in specialized hardware having to be made available for storage. Therefore, it makes sense to centralize data management in a relational database, which can be accessed by multiple users. A number of database developments try to address some of the abovementioned issues (Orro *et al.* 2008, Fong *et al.* 2010, Mitha *et al.* 2011).

However, the main shortcoming is the »one genotype per row« (OGPR) storage scheme which leads to huge storage requirements and slow export times. Rios *et al.* (2010) improved this scheme by storing one record per individual containing all genotypes together with each genotype's position. Our proposal goes well beyond this through a more efficient storage scheme and the development of the genotype set concept, which substantially enhances data manipulation.

Efficient management of SNP data has to address long term data storage, multiple genotyping panels of different sizes as are already in practical use in animal breeding and elsewhere. Further, efficient selection of SNPs and fast exports to various formats for further processing are essential. This leads us to the following design.

The design

Compressed storage

Typical files from genotyping labs easily have sizes of hundreds of megabyte (MB) per individual. We propose to store the SNP genotypes of one individual in a compressed vector using the position as determined by the panel map, which leads to one genotype record per individual. When only the biallelic state of a SNP is of interest, two bit storage is sufficient, allowing 16 SNPs to be stored in one 32 bit integer word. Accordingly, all genotypes from a 60 000 SNP panel can be stored in an integer vector of dimension 3 750 or 14.6 kb. Once a panel map is stored, any number of resulting genotypes can be loaded.

Often the call rates i.e. the GCscore need to be stored for each SNP, typically a floating point number in the range of 1 to 100. Here, a flexible scheme is proposed, allowing the user to determine the number of bits to be used for the score: 4 bits accommodate a resolution of 100/15. Using the same four bits on the range of 51 % to 100 % will double the resolution. As with the genotpyes, the GCscore is also designed as one vector per individual.

Set based manipulation

For easy data manipulation, we introduce the concept of effectively spaceless genotype sets. Each SNP panel comprises a specified set of SNPs. Using the SNP's position in the panel as its position in the genotype bit vector enables access to each SNP without explicitly having to state the SNP name. Once genotypes are treated as vectors, a SNP dataset can be viewed as a matrix of genotypes with the SNPs constituting the columns while each genotyped individual leads to one row.

Often, only subsets of SNPs are used in analyses, perhaps only those from a particular chromosome, or SNPs with a minimum frequency. Using a bit vector snp_sel_vec of the dimension of the SNP panel provides a generalized approach: a .TRUE. or .FALSE. decides if a SNP is a member of a particular subset. A genotype set is then completely defined by adding a vector indiv_sel_vec which contains the individuals of the subset. Finally, a set name for the combination of a particular snp_sel_vec and indiv_sel_vec, uniquely specifies a genotype set. Thus, creating new derived genotype sets amounts to creating two new lists: one for the SNPs and the other for the individuals and giving a genotype set name for easy access. The storage implications are clear: instead of having to store a matrix of dimension nSNP * nIndividual only two vectors of dimension nSNP and nIndividual need to be stored for each derived matrix which is thus effectively spaceless.

It should be noted that the concept of genotype sets allows rapid implementation of general set operations like unions or intersections on the basis of any genotype set, original or derived.

Being a central repository for all data to be analysed, fast exports are critical. A data analysis step typically starts with an export using an appropriate format for downstream processing. Often exports are performed by costly SQL selects on the basis of SNP names (Orro *et al.* 2008, Fong *et al.* 2010, Mitha *et al.* 2011).

However, export speed is a function of both the data storage and the retrieval scheme. Storage overhead of our approach is minimal: the indiv_sel_vec is an integer vector with as many 32 bit words as there are individuals in the genotype sample, while snp_sel_vec has the dimension of the number of SNPs in the panel divided by 16 (for two bit storage). Thus, all genotypes from a one million panel will occupy 244 kb in snp_sel_vec. Clearly, on this data volume basis our design will have a performance advantage over the OGPR paradigm which needs to process one million records per individual.

For data retrieval, the genotype set approach replaces SQL based SNP selection by much faster vector operation. An export of a genotype set amounts to these actions: firstly, snp_sel_vec and indiv_sel_vec are fetched for the chosen set. Secondly, for each individual the compressed genotype vector is retrieved by one SQL select and shrunk on the basis of the snp_sel_vec which can be implemented as fast shifts. Thus, the extraction speed is largely independent from the subset selection.

Performance of proposed design

The proof of concept implementation was done in Perl and PostgreSQL as the backend database server. Apart from conceptual simplicity, performance in terms of storage efficiency and speed of data import and export is of critical importance. This was investigated by a number of genotype datasets of very different sizes (Table 1).

Table 1
Performance for five test datasets

data	n SNP[a]	n ind[b]	tot SNP[c]	imp[d]	exp[e]	DB stor[f]
set1	58	47	2.7	2.38	1.38	0.28
set2	36	403	14.7	4.37	1.02	0.27
set3	229	90	20.6	1.89	0.69	0.25
set4	4 098	270	1 106	1.60	0.53	0.25
set5	1 458	1 397	2 036	1.50	0.54	0.25

[a]panel size: number of SNPs×1 000, [b]number of individuals, [c]total number of SNPs in dataset×1000 000, [d]time to import 1 mio SNPs in s, [e]time to export 1 mio SNPs in s, [f]storage in MByte per 1 mio SNPs

The timings refer to the import of SNPs after the panel map has been loaded, as this is done only once. Further, two bit storage for the biallelic state genotypes (A, B, AB, and no call) with no GCscore is assumed. The benchmarks were executed on an iCore 5 laptop (2.7GHz / 4GB RAM).

Importing data following our design requires two steps. Firstly, the panel map is loaded which contains the panel size, the SNP names and chromosome location. A unique panel name is given, which needs to be specified whenever new genotype data are loaded in a second step. During the initial load, a snp_sel_vec is created with all bits set to 1. The individual IDs are stored in indiv_sel_vec as picked up from the data file. The combination of those two selection vectors then yields the first genotype set, uniquely identified by its symbolic name, which is the only information required for an export. Notice that our design scales very well (column 5 and 6 in Table 1), with import and export time per one million SNP even going down as panel sizes increase.

Popular SNP database management systems store OGPR (Orro *et al.* 2008, Fong *et al.* 2010, Mitha *et al.* 2011) resulting in huge storage requirements: using the HumapHap300 for 300 subjects, about 90 million of records will be generated (Mitha *et al.* 2011). In contrast, our design will create only 300 rows, which will clearly reduce both storage and processing requirements. Even the improved design presented by Rios *et al.* (2010) requires 5 gigabyte (GB) for 1.5 billion genotypes, while our design would require approximately 0.37 GB.

A direct comparison to other implementations for computing time is difficult. Based on the timings from our design (1.55 sec and .54/1 million genotypes, for import and exports as a weighted mean from Table 1), we expanded the software comparisons given in Table 1 in Mitha *et al.* (2011) for a 317K panel with 100 samples. The timings in minutes are n/a, 18, 4.2 and 0.82, for imports and 10, 93, 5, 0.29 for exports from SNPLims, GWASA, SNPpy single, and our implementation, respectively. Thus, our design seems to be an order of magnitude faster than the best of the contenders. This is not surprising, since the OGPR requires space for the sample ID, the SNP name and the chromosome location for each genotype. Using the database design given in Mitha *et al.* (2011) for the 317K dataset, we have a storage requirement of 2 721 MB versus 8 MB. Thus, our compressed bit vector storage scheme is around 300 times more efficient than the OGPR scheme of other packages.

As an example, on a 500 GB disk two million individuals genotyped with a 317K panel can be stored using our scheme. In contrast, the same disk would store 0.146 million individuals with the design from Rios at al. (2010), and only 7 000 with the common OGPR storage scheme.

Supplementary material

The proof of concept code is available under the GPL license at ftp://ftp.tzv.fal.de/pub/snp/SNP-PoC.tar.gz

References

Fong C, Ko DC, Wasnick M, Radey M, Miller SI, Brittnacher M (2010) GWAS Analyzer: integrating genotype, phenotype and public annotation data for genome-wide association study analysis. Bioinformatics 26, 560-564

F. Mitha, H. Herodotou, N. Borisov, C. Jiang, J. Yoder, K. Owzar. SNPpy-Database management for SNP data from GWAS studies. Duke Biostatistics and Bioinformatics (B&B) Working Paper Series, page 14, 2011

Orro A, Guffanti G, Salvi E, Macciardi F, Milanesi L (2008) SNPLims: a data management system for genome wide association studies. BMC Bioinformatics 9 (Suppl.), S13

Rios D, McLaren WM, Chen Y, Birney E, Stabenau A, Flicek P, Cunningham F (2010) A database and API for variation, dense genotyping and resequencing data. BMC Bioinformatics, 11, 238