Improvement of missing genotype imputation through bi-directional parsing of large SNP panels

Christine Sinoquet

Improvement of missing genotype imputation through bi-directional parsing of large SNP panels

Christine Sinoquet

Computer Science Institute of Nantes-Atlantic (Lina), U.M.R. C.N.R.S. 6241, University of Nantes, 2 rue de la Houssinière, BP 92208, 44322 Nantes Cedex, France

— Bioinformatics —

RESEARCH REPORT

N° hal-00300596

February 2009

LINA, Université de Nantes – 2, rue de la Houssinière – BP 92208 – 44322 NANTES CEDEX 3
Tél. : 02 51 12 58 00 – Fax. : 02 51 12 58 12 – http://www.sciences.univ-nantes.fr/lina/

Christine Sinoquet

Improvement of missing genotype imputation through bi-directional parsing of large SNP panels

14 p.

Les rapports de recherche du Laboratoire d'Informatique de Nantes-Atlantique sont disponibles aux formats PostScript® et PDF® à l'URL :
http://www.sciences.univ-nantes.fr/lina/Vie/RR/rapports.html

Research reports from the Laboratoire d'Informatique de Nantes-Atlantique are available in PostScript® and PDF® formats at the URL:
http://www.sciences.univ-nantes.fr/lina/Vie/RR/rapports.html

Improvement of missing genotype imputation through bi-directional parsing of large SNP panels

Christine Sinoquet

christine.sinoquet@univ-nantes.fr

Abstract

Such difficult analyses as disease association studies, which aim at mappping genetic variants underlying complex human diseases, rely on high-throughput genotyping techniques. However, a shortcoming of these techniques is the generation of missing calls. Computational inference of missing data represents a challenging alternative to genotyping again the missing regions. In this paper, we present SNPShuttle, an algorithm designed to gain accuracy over a former method described by Roberts and co-authors [7] (NPUTE). Given an SNP panel, NPUTE algorithm infers missing data through a single parse, relying on local similarity within sliding windows. Instead, SNPShuttle scans an SNP panel in an iterative bi-directional way, to resolve missing data with more confidence.

Introduction

DNA strings consisting of billion of chemical bases A, T, C and G constitute the hereditary material stored in the pairs of chromosomes of eukaryotes. These genetic sequences contain information that influences physical traits, the likelihood of suffering from disease as well as response to pathogens, chemicals and other agents. Differences in individual bases are the most common type of genetic variation. These genetic differences, called single nucleotide polymorphisms (SNPs), are detected on chips analyzed through high-throughput genotyping techniques. In the domain of genetical epidemiology, associations studies attempt to link genetic variants to the risk for specific illnesses, with the objective of proposing new methods of preventing, diagnosing, and treating diseases. Case-control association studies are considered to be the simplest framework to help elucidate the genetic basis of complex diseases. Such studies deal with populations of unrelated individuals split into cohorts diagnosed with the disease of interest and cohorts of unaffected controls. The issue at stake is identifying genetic determinants - possibly combinations of determinants - , which should accumulate among cases. Amongst various difficulties likely to introduce a bias in the studies, not a least problem to cope with is the presence of undetermined SNPs, or "missing calls", in the data generated by genotyping techniques (approximately between 5% and 10%).

There are three alternatives to repeating the genotyping for the missing data, a prohibitive task both in terms of time and cost: (i) merely dismissing entire rows and columns of the SNP panel containing the missing calls is quite a drastic solution, with a strong impact on the power to detect disease-predisposing variants; (ii) inferring missing data prior to the task of interest (*i.e.*, disease association study, genetic mapping $\cdots$), (iii) handling missing data while the task of interest is performed. Indeed, this third category amounts to off-line or on-line inference. A peculiar case may lead to some confusion in minds: standard genotyping techniques can not distinguish the two homologous chromosomes of an individual, therefore only the "unphased" genotype (*i.e.*, the combination of the two homologous haplotypes) is directly observable. Thus, when the task of interest, genotype phasing, applies to data containing missing calls, two missing-data problems interfer. From now on, we will refer to genotypes as unphased genotypes.

Various computational methods have been proposed to infer - or impute - missing genotypes. Assignment of the most frequent allele identified to the missing call concerned and k-nearest neighbor voting methods (KNN) are the most simple. A review of eight methods has been more specifically dedicated to such previous methods, as well as various regression methods [13]. Other methods implement expectation maximization [6], Bayesian approaches [4], Decision Forest pattern recognition [12], neural networks [13, 11], as well as Gibbs sampling combined with tree-based approach [1].

Other methods explicitly cope with the haplotype block structure of eukaryotic genomes. Empirical studies have confirmed that over short regions (a few kilobases in human genome), haplotypes tend to cluster into groups [5, 14], which entails interesting constraints for the corresponding genotypes. To inpute missing calls, this feature is exploited in various ways: entropy measure combined with dynamic programming to partition haplotypes into blocks [10], cluster membership allowed to change continuously along the chromosome according to a hidden Markov model [8]. In this line, Roberts and co-authors designed a new algorithm, NPUTE, which performs KNN imputation in the context of sliding windows modelling haplotype blocks [7]. Their algorithm deals with SNP panels where the number of markers is much higher than the number of individuals (up to 10^4 in the case of some chromosomes). The very point central to NPUTE is efficient knowledge management from current window to next one.

Finally, among specific softwares yet also able to handle missing data, we mention for illustration methods dedicated to genotype phasing [2, 8] and detection of causal variants that have not been directly genotyped, in the framework of association studies [3, 9].

Thoroughly examining NPUTE, we identified in dependencies between sliding windows a promising lead to infer missing calls with more confidence, therefore expecting a gain of accuracy.

Methods

Foreword

Beside gaining accuracy due to bi-directional parsing, we are also resolute to design a method which would not depend on memory availability constraints.To meet this second purpose, we implemented successive loadings of consecutive "small" SNP blocks during each parsing of the SNP panel. Thus, contrary to NPUTE, memory needs only be allocated for a SNP block rather than for the whole SNP panel. Therefore, we wish to design a variant of algorithm NPUTE, KNNWinOpti, together with a novel algorithm SNPShuttle. In the following, we will first briefly describe the original method of Roberts and co-authors. In this occasion, we will highlight the existence of calculation dependencies between blocks of the SNP panel. Moreover, these SNP block dependencies are either backward or forward dependencies, which makes the design of software also managing SNP block loading a delicate task. Then we will present KNNWinOpti. Finally, the scheme of SNPShuttle will be depicted.

The common basic concepts of NPUTE, KNNWInOpti and SNPShuttle

The input parameters for NPUTE algorithm are the SNP panel, loaded as the matrix $snp[0\ ..\ M-1][0\ ..\ N-1]$ of M rows (markers) and N columns (individuals) and L, the "half-size" of any sliding window. The elements of the SNP matrix belong to $\{0,\ 1,\ 2\}$, where 0 denotes the allele with major frequency for each genetic marker, 1 is that of the least frequent allele and 2 is the label for "missing data". The key idea of NPUTE is performing fast imputation over overlapping sliding windows. In the sequel, we will respectively denote sub-matrices and rows as $snp[m\ ..\ m'][n\ ..\ n']$ and $snp[m\ ..\ m'][n]$. $snp^*[m-L\ ..\ m+L][n\ ..\ n']$ will refer to a matrix deprived of row m. Specifically, $snp^*[m-L\ ..\ m+L][0\ ..\ N-1]$ will be named $\mathcal{W}_m$, for conciseness.

The idea central to NPUTE is to infer the missing marker m of some individual n, $snp[m][n]$, "copying" it from the marker m of the nearest neighbour of individual n, say individual n_n, in the context $\mathcal{W}_m$. Such contexts roughly model the concept of haplotype block. Namely, n_n is identified as the individual minimizing a distance criterion, denoted Δ, over $\mathcal{W}_m$. Computing the distance $\Delta(snp^*[m-L\ ..\ m+L][n], snp^*[m-L\ ..\ m+L][n'])$ involves comparing the projection of individuals n and n' onto current window $\mathcal{W}_m$:

$$\Delta(snp^*[m-L\ ..\ m+L][n_n], snp^*[m-L\ ..\ m+L][n]) = \Sigma_{i=m-L, i\neq m}^{i=m+L}\ \Delta_m(snp[i][n], snp[i][n']).$$

The distance between markers, Δ_m, is merely computed as follows:

$$\Delta\_m(i,j) = \begin{cases} 0\ if\ i=j \\ 2\ if\ (i,j)=(0,1)\ or (i,j)=(1,0) \\ 1\ if\ i=2\ or\ j=2 \end{cases}$$

It must be highlighted that in this algorithm, the missing calls ("2") of the context participate in the distance computation. We now denote W_m the vector of the $n\ (n-1)/2$ pairwise mismatch distances between individual projections onto current window $\mathcal{W}_m$. Given W_m, the Pairwise Mismatch Vector (PMV) related to $\mathcal{W}_m$, inference for any missing $snp[m][n]$ is straightforward: considering only the $N-1$ relevant entries (n,n') $(n<n')$ and (n',n) $(n'<n)$ in W_m, the smallest distance with individual n is identified, say, for individual n_n. Provided that $snp[m][n_n]$ is not missing itself, it is assigned to $snp[m][n]$. From now on, we will name $PMV(m)$ the Pairwise Mismatch Vector with $n(n-1)/2$ entries such that $PMV(m)[(n,n')] = \Delta_m(snp[m][n], snp[m][n'])$, $(n<n')$.

For tractability over large SNP panels, NPUTE fully exploits window sliding. The parsing of the SNP panel is implemented shifting the current window one range further at each step. Thus, the PMV relative to a window centered on range m is merely the PMV relative to the previous overlapping window centered on range $m-1$, from which the contribution of range $m-L-1$ must be substracted and that of range $m+L$ must be added (see Figure 2). In addition, note that there are $2\times L+1$ non symmetric windows to be specially processed, among which $2\times L$ are not symmetric.

However, in the simple case of a symmetric window, W_m is not merely computed as $W_{m-1} - PMV(m-L-1) + PMV(m+L)$. Row m itself does not contribute to the calculation of W_m, which is indeed computed as follows

$$W_m = W_{m-1} + PMV(m-1) - PMV(m) - PMV(m-L-1) + PMV(m+L). \quad (1)$$

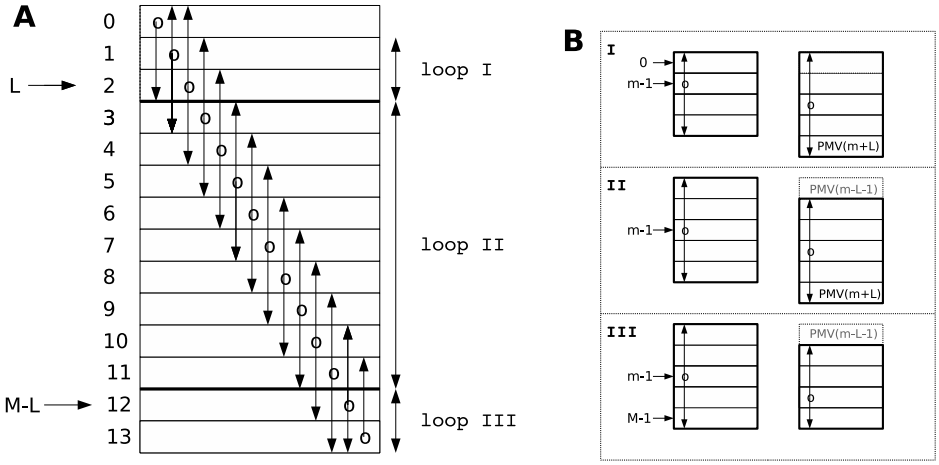

Figure 1: Sliding windows. $M = 14$; $L = 2$. Depending on the row m of the SNP inferred, the sliding window is either centered on m or is non symmetric.

Now all basic concepts common to NPUTE, KNNWinOpti and SNPShuttle are settled. However, even the original algorithm, NPUTE, is not so simple since special care must be taken regarding non symmetric windows. In the sequel, we will proceed gradually in the presentation of the sketch of the original algorithm, the adaptations implemented to meet the memory sparing purpose and the innovative bi-directional variant.

Sketch of the original method

The sketch for SNP panel processing is reminded in Algorithm 1. The algorithm processes apart the non symmetric window 0 (lines 1 to 3), then runs three loops. First loop (line 5) processes non symmetric windows 1 through $L-1$ and first symmetric window L. Third loop (line 18) processes only non symmetric windows $M-L$ through $M-1$. General formula (1) (line 14) is adapted to the case of the L windows successively encompassing an increasing number of rows (line 7), whereas it is symmetrically tuned to the case of the L last windows (line 20).

Algorithm 1 $NPUTE(M, N, SNP, L)$

Input: $\mathbf{M}$, the number of genetic markers; $\mathbf{N}$, the number of individuals; a matrix $\mathbf{SNP}[\mathbf{0} \,..\, \mathbf{M-1}][\mathbf{0} \,..\, \mathbf{N-1}]$ of known markers (belonging to $\{0,\ 1\}$) and missing markers (2), each column corresponding to a given individual; $\mathbf{L}$, half-size of a sliding window.
Output: matrix $\mathbf{SNP}$, where each previously missing value 2 is now replaced with either 0 or 1.

1: $W \leftarrow sum\_of\_PMVs\_from\_to(1, L)$
2: $inference\_of\_missing\_markers\_for\_range(0)$
3: $previous\_PMV \leftarrow compute\_PMV(0)$
4:
5: **for** $m = 1$ *to* L // LOOP I
6: $(\mathbf{1})\ current\_PMV \leftarrow compute\_PMV(m)$
7: $W \leftarrow W + previous\_PMV - current\_PMV + compute\_PMV(m + L)$
8: $(\mathbf{2})\ previous\_PMV \leftarrow current\_PMV$
9: $inference\_of\_missing\_markers(W)$
10: **endfor**
11:
12: **for** $m = L + 1$ *to* $M - L - 1$ // LOOP II
13: $process\ as\ in\ (\mathbf{1})$
14: $W \leftarrow +W + previous\_PMV - current\_PMV + \mathbf{compute\_PMV(m + L)} - \mathbf{compute\_PMV(m - L - 1)}$
15: $process\ as\ in\ (\mathbf{2})$
16: **endfor**
17:
18: **for** $m = M - L$ *to* $M - 1$ // LOOP III
19: $process\ as\ in\ (\mathbf{1})$
20: $W \leftarrow +W + previous\_PMV - current\_PMV - compute\_PMV(m - L - 1)$
21: $process\ as\ in\ (\mathbf{2})$
22: **endfor**

Management of backward and forward dependencies for the purpose of memory sparing

Before adapting SNP block loading to the previous algorithm, a remark is imperative. In line 14 of Algorithm 1, due to backward dependencies ($compute\_PMV(m-L-1)$) and forward dependencies ($compute\_PMV(m+L)$), the computation of Ws is not optimized. Indeed, $PMV(m+L)$ will be computed again as a contribution to W_{m+L} (line 13 referring to line 6). Similarly, $PMV(m-L-1)$ (line 14) has already been computed since it had to be dismissed from W_{m-L-1} as "$current\_PMV$". These remarks point out that the memorization should not restrain to the single last PMV vector calculated (lines 8, 15 and 21 in algorithm), but should extend to the latest L PMVs computed instead. Moreover, similarly, forward dependencies will be accounted for through the memorization of the L PMVs relative to the highest row numbers calculated.

It now remains to combine such dependency management with the loading of successive SNP blocks of R rows (R is an input parameter). The sketch of this novel version, KNNWinOpti, is described in Algorithm 2. Since the combination of the two modifications (dependency management, SNP block loading) brings complexity in the description of the novel version, we will carefully comment it in the following.

In the original version, two PMVs are computed for row m, $PMV(m+L)$ and $PMV(m-L-1)$ (Algo. 1, line 14). Instead, in novel Algo. 2, $PMV(m+L)$ is computed (line 12) and stored in the FIFO $PMV\_forward$ (line 13) for further reuse (line 25 referring to line 11). Besides, when it is time to infer missing data in a given row, not only is the PMV relative to this row available as head of FIFO $PMV\_forward$ (line 25 referring to line 11), its update after inference is added to FIFO $PMV\_backward$ (line 27 pointing to line 16) so that it may be reused as $PMV(m'-L-1)$, the head of the previous FIFO at the time of inference for row m' ($m'=m+L+1$), at line 26. Figure 3 A shows on a simple example how the two FIFO lists are synchronized. Finally, the SNP block loading manager (lines 23 and 32) ensures that row $i+L$ was also loaded for last row i in each newly loaded block of LOOP II. The preliminary loading of a block of $2\times L+1$ rows, required by inference of row 0 and LOOP I, is crucial to the whole synchronization of the SNP block loading manager with the inferring process under way. Note that LOOP III does not refer to forward dependencies. Figure 3 B illustrates the synchronization between inference and block loading.

Depending on the number of rows involved in LOOP II, the last iteration in this loop may require the loading of less than R rows, which explains a special (trivial) treatment (lines 20 and 31 to 34).

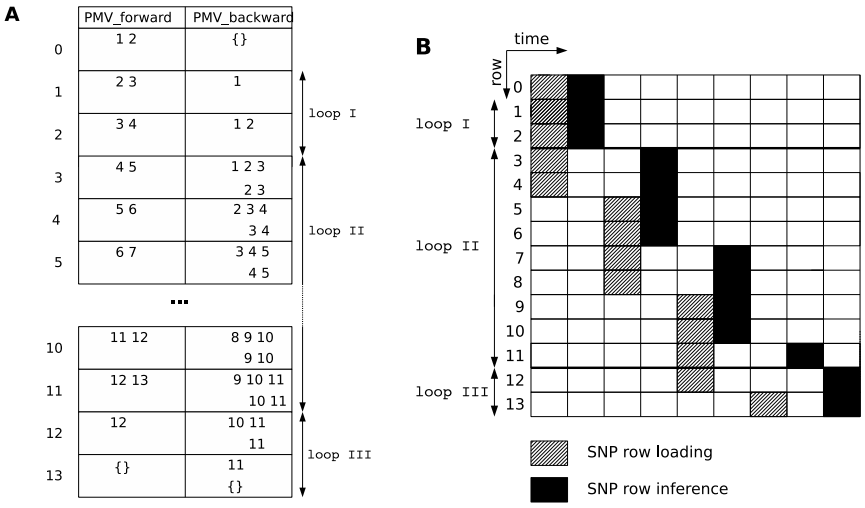

Figure 2: **A** Synchronization of forward and backward FIFO lists; **B** Synchronization of SNP block loading manager and missing data inference; $M=14$; $L=2$; $R=4$.

Algorithm 2 $KNNWinOpti(M, N, SNP, L, R)$

Input and Output: in addition to those of algorithm , input parameter R, size of the SNP blocks to be successively loaded

Auxiliary variables:
$PMV\_backward,\ PMV\_forward$: *two FIFO lists initialized as empty lists*

1: $SNP \leftarrow load\_next\_ranges(2L+1)$
2: $W \leftarrow 0$
3: **for** $i = 1$ *to* L
4: $PMV \leftarrow compute\_PMV(i)$; $add\_queue(PMV\_forward,\ PMV)$; $W \leftarrow W + PMV$
5: **endfor**
6: $current\_PMV\_inferred \leftarrow inference\_of\_missing\_markers(W)$ inference for row 0
7: **add_queue**(**PMV_backward**, **current_PMV_inferred**)
8: $previous\_PMV \leftarrow current\_PMV\_inferred$
9:
10: **for** $i = 1$ *to* L // LOOP I
11: (**1**) $current\_PMV \leftarrow remove\_head(PMV\_forward)$
12: **PMV_forward_aux** $\leftarrow$ **compute_PMV**(**m** + **L**)
13: **add_queue**(**PMV_forward**, **PMV_forward_aux**)
14: $W \leftarrow W + previous\_PMV - current\_PMV + PMV\_forward\_aux$
15: (**2**) $current\_PMV\_inferred \leftarrow inference\_of\_missing\_markers(W)$
16: **add_queue**(**PMV_backward**, **current_PMV_inferred**)
17: $previous\_PMV \leftarrow current\_PMV\_inferred$
18: **endfor**
19:
20: $(nb\_loadings,\ lastIterApart,\ rest) \leftarrow compute\_nb\_SNP\_block\_loadings()$
21:
22: **for** $i = 1$ *to* $nb\_loadings$ // LOOP II
23: $SNP \leftarrow load\_next\_ranges(\mathbf{R})$
24: **for** $j = 1$ *to* R
25: *process as in* (**1**)
26: $W \leftarrow W + previous\_PMV - current\_PMV + PMV\_forward\_aux -$ **remove_head**(**PMV_backward**)
27: *process as in* (**2**)
28: **endfor**
29: **endfor**
30:
31: **if** $lastIterApart$ // LOOP II **(end)**
32: $SNP \leftarrow load\_next\_ranges($**rest**$)$
33: *process as in lines* 23 *through* 27 (*with* $rest$ *instead of* r)
34: **endif**
35:
36: **for** $m = 1$ *to* L // LOOP III
37: $current\_PMV \leftarrow remove\_head(PMV\_forward)$
38: $W \leftarrow W + previous\_PMV - current\_PMV - remove\_head(PMV\_backward)$
39: $current\_PMV\_inferred \leftarrow inference\_of\_missing\_markers(W)$
40: $previous\_PMV \leftarrow current\_PMV\_inferred$
41: **endfor**

Iterative bi-directional inference

In their method, Roberts and co-authors use a specific data structure, a mismatch accumulator array MMA, which is computed before performing inference. To be short, it plays the same role in W computation as the PMV vectors aforementioned. However, since the MMA is calculated off-line, the inference for row m will not benefit from the inference for rows of lower ranks. That is, in all cases where distance $\Delta_m(i,j)$ is approximated as 1 if i or j is equal to 2, an update would possibly lead to a refined distance 0 or 1. We took this remark into account when designing Algorithm 2, which is therefore not a simple transcription of Roberts and co-workers' method merely augmented with block loading and FIFO list management. Indeed, we are careful that any newly inferred row is added to the $PMV\_backward$ list (line 15), so that the inference might be more accurate.

Moreover, this update concern allows further optimization. It was of no consequence for Roberts *et al.* to scan the SNP panel from top to bottom (TB) or from bottom to top (BT) since context $\mathcal{W}_m$ did not account of the results of previous inference for rows $m-L$ to $m-1$. Nonetheless, it is attractive to confront the result of a TB scan with that of a BT scan, in order to resolve the missing SNPs with more confidence, which is implemented in our second version, SNPShuttle. Thus, any SNP inferred as the same allele identifier by successive TB and BT scans can be fixed. Any uncertain SNP will remain tagged as "missing" until a further iteration yields identical TB and BT results. The entire process is to be iterated until a minimal percentage of missing data remains uncertain or until a maximal number of iterations is reached. The SNP panel is successively cleared from its missing calls, starting with the markers easiest to guess and enriching the context of SNPs more difficult to infer.

Algorithm 3 presents the scheme of SNPShuttle. At line 4, the call to $top\_to\_bottom\_scan$ procedure is actually a call to KNNWinOpti. Similarly, call $bottom\_to\_top\_scan$ is applied on the current SNP panel inverted row per row.

Algorithm 3 $SNPShuttle(M, N, SNP, L, R, \tau)$

Input and Output: in addition to those of algorithm , input parameter τ specifies the minimal percentage of missing data allowed to remain uncertain

1: $modified \leftarrow true$; $percentage\_of\_non\_solved\_SNPs \leftarrow 100$
2: **while**$(modified\ and\ (percentage\_of\_non\_solved\_SNPs > \tau))$
3: $modified \leftarrow false$
4: $TB\_inferred\_SNPs \leftarrow top\_to\_bottom\_scan(SNP)$
5: $BT\_inferred\_SNPs \leftarrow bottom\_to\_top\_scan(SNP)$
6: $solved\_SNPs \leftarrow compare(TB\_inferred\_SNPs, BT\_inferred\_SNPs)$
7: **if** $(solved\_SNPs\ is\ not\ empty)$
8: $updateWith(SNP,\ solved\_SNPs)$
9: $update(percentage\_of\_non\_solved\_SNPs)$
10: $modified \leftarrow true$
11: **endif**
12: **endwhile**

Conclusion

Roberts and co-authors precursory work provided a promising basis to gain accuracy with a simple algorithm. In this paper, we proposed a novel algorithm, based on iterative bi-directional parsing of SNP panels. We are currently implementing the two algorithms, KNNWinOpti and SNPShuttle. Also do we have to adapt KNNWinOpti (the core of SNPShuttle) to obtain pre-processing software dedicated to the identification of the optimized window "half-size" L (the corresponding accuracy is computed for all non missing markers, temporarily considered as missing calls and inferred). As chip resolution increase will also rise the number of SNPs available for each chromosome, it is crucial to implement SNP block loading, as we plan to do, if the software is intended to run on on standard computer. Finally, one of our future tasks is more thoroughly examining the idea of benefitting from previously inferred missing calls, locally relying on regions of high quality.

References

1. J Y Dai, I Ruczinski, M LeBlanc, and C Kooperberg. Imputation methods to improve inference in snp association studies. *Genet Epidemiol*, 30(8):690–702, 2006.

2. E Halperin and E Eskin. Haplotype reconstruction from genotype data using imperfect phylogeny. *Bioinformatics*, 20(12):1842–1849, 2004.

3. J Marchini, B Howie, S Myers, G McVean, and P Donnelly. A new multipoint method for genome-wide association studies by imputation of genotypes. *Nature Genetics*, 39:906–913, 2007. doi:10.1038/ng2088.

4. T Niu, Z S Qin, X Xu, and J S Liu. Bayesian haplotype inference for multiple linked single-nucleotide polymorphisms. *Am J Hum Genet*, 70(1):157–169, 2002.

5. N Patil, A J Berno, D A Hinds, W A Barrett, J M Doshi, C R Hacker, C R Kautzer, D H Lee, C Marjoribanks, D P McDonough, B T Nguyen, M C Norris, J B Sheehan, N Shen, D Stern, R P Stokowski, D J Thomas, M O Trulson, K R Vyas, K A Frazer, S P Fodor, and D R Cox. Blocks of limited haplotype diversity revealed by high-resolution scanning of human chromosome 21. *Science*, 294(5547):1719–1723, 2001.

6. Z S Qin, T Niu, and J S Liu. Partition ligation expectation maximization algorithm for haplotype inference with single nucleotide polymorphisms. *Am J Hum Genet*, 71(5):1242–1247, 2002.

7. A Roberts, L McMillan, W Wang, J Parker, I Rusyn, and D Threadgill. Inferring missing genotypes in large snp panels using fast nearest-neighbor searches over sliding windows. *Bioinformatics*, 23(13):i401–i407, 2007. doi:10.1093/bioinformatics/btm220.

8. P Scheet and M Stephens. A fast and flexible statistical model for large-scale population genotype data: applications to inferring missing genotypes and haplotypic phase. *Am J Hum Genet*, 78(4):629–644, 2006.

9. B Servin and M Stephens. Imputation-based analysis of association studies: candidate regions and quantitative traits. *PLoS Genetics*, 3(7), 2007. e114.

10. S-C Su, C-C Jay Kuo, and T Chen. Inference of missing snps and information quantity measurements for haplotype blocks. *Bioinformatics*, 21(9):2001–2007, 2005.

11. Y V Sun and S L Kardia. Imputing missing genotypic data of single-nucleotide polymorphisms using neural networks. *Eur J Hum Genet*, 16(4):487–95, 2008.

12. Q Xie, L D Ratnasinghe, H Hong, R Perkins, Z-Z Tang, N Hu, P R Taylor, and W Tong. Decision forest analysis of 61 single nucleotide polymorphisms in a case-control study of esophageal cancer; a novel method. *BMC Bioinformatics*, 6(2), 2005. doi:10.1186/1471-2105-6-S2-S4.

13. Z Yu and D J Schaid. Methods to impute missing genotypes for population data. *Hum Genet*, 122(5):495–504, 2007.

14. K Zhang, J Zhu, J Shendure, G J Porreca, J D Aach, R D Mitra, and G M Church. Long-range polony haplotyping of individual human chromosome molecules. *Nature genetics*, 38(3):382–387, 2006.

Improvement of missing genotype imputation through bi-directional parsing of large SNP panels

Christine Sinoquet

Abstract

Such difficult analyses as disease association studies, which aim at mappping genetic variants underlying complex human diseases, rely on high-throughput genotyping techniques. However, a shortcoming of these techniques is the generation of missing calls. Computational inference of missing data represents a challenging alternative to genotyping again the missing regions. In this paper, we present SNPShuttle, an algorithm designed to gain accuracy over a former method described by Roberts and co-authors [7] (NPUTE). Given an SNP panel, NPUTE algorithm infers missing data through a single parse, relying on local similarity within sliding windows. Instead, SNPShuttle scans an SNP panel in an iterative bi-directional way, to resolve missing data with more confidence.

LINA, Université de Nantes
2, rue de la Houssinière
B.P. 92208 — F-44322 NANTES CEDEX 3