ImputeZScore.jl

September 12, 2017

1 Imputation of association statistics in GWAS

1.1 Background

In a traditional GWAS, one collects genotype data at a small subset of SNPs over some individuals, then imputes genotypes across the entire genome, and finally computes association statistics (e.g. Z-scores) for each genotyped and imputed SNPs. This procedure can take tremendous amount of time as genotype imputation is computationally extensive.

Here, we impute Z-scores of ungenotyped SNPs directly from Z-scores of genotyped SNPs, without first performing genotype imputation, saving hundreds of hours of CPU time. The idea behind this approach is that Z-scores of genotyped and ungenotyped SNPs follow a multivariate normal distribution with LD matrix, which can be estimated from a reference panel, as the covariance structure -- one can impute the Z-scores of ungenotyped SNPs as the expectation of Z-scores of ungenotyped SNPs conditional on the Z-scores of genotyped SNPs.

In detail, let $Z = (Z_t, Z_u)$ be the Z-score vector partitioned into two components, genotyped (Z_t) and ungenotyped (Z_u). It has been previously shown that Z has the following distribution,

$$\begin{bmatrix} Z_t \\ Z_u \end{bmatrix} \sim MVN\left(\begin{bmatrix} \Lambda_t \\ \Lambda_u \end{bmatrix}, \begin{bmatrix} \Sigma_{tt} & \Sigma_{tu} \\ \Sigma_{ut} & \Sigma_{uu} \end{bmatrix}\right),$$

where $\Lambda = (\Lambda_t, \Lambda_u)$ is the non-centrality parameter, Σ_{tt} the LD between genotyped SNPs, Σ_{tu} the LD between genotyped and ungenotyped SNPs, Σ_{uu} the LD between ungenotyped SNPs.

The conditional expectation of Z_u given Z_t is then

$$Z_u | Z_t \sim MVN\left(\Lambda_u + \Sigma_{ut}\Sigma_{tt}^{-1} Z_t, \Sigma_{uu} - \Sigma_{ut}\Sigma_{tt}^{-1}\Sigma_{tu}\right).$$

We impute the Z-scores of ungenotyped SNPs as $\hat{Z}_u = E[Z_u|Z_t] = \Sigma_{ut}\Sigma_{tt}^{-1} Z_t$ under the null assumption that $\Lambda_u = 0$. Let $W = \Sigma_{ut}\Sigma_{tt}^{-1}$. This can be viewed as the weights on Z-scores of genotyped SNPs in the imputation of Z-scores of ungenotyped SNPs. Then

$$\hat{Z}_u \sim MVN(0, \Sigma_{ut}\Sigma_{tt}^{-1}\Sigma_{tu}),$$

where each entry $\hat{Z}_{u,i}$ of $\hat{Z}_u$ follows

$$\hat{Z}_{u,i} \sim N(0, \Sigma_{ut,i*}\Sigma_{tt}^{-1}\Sigma_{tu,*i}).$$

Here, $\Sigma_{ut,i*}$ denotes the i-th row of Σ_{ut} and $\Sigma_{tu,*i}$ the i-th column of Σ_{tu}. To obtain a associations statistics that has mean 0 and variance 1, we standardize $\hat{Z}_{u,i}$ by $\sqrt{\Sigma_{ut,i*}\Sigma_{tt}^{-1}\Sigma_{tu,*i}}$. More specifically, the final imputated association statistics of each SNP is

$$\hat{Z}_{imp,i} = \frac{\hat{Z}_{u,i}}{\sqrt{\Sigma_{ut,i*}\Sigma_{tt}^{-1}\Sigma_{tu,*i}}} \sim N(0,1).$$

In practice, inverting a large matrix can be time-consuming. Instead, we adopt a window-based approach, i.e. we impute Z-scores of ungenotyped SNPs one window at a time.

2 Example

```
In [1]: # load in required packages
        include("../src/ImputeZScore.jl")
        using DataFrames, .ImputeZScore, SnpArrays
```

```
In [2]: # read in z-scores of genotyped SNPs on chromosome 22 as a data frame
        zsc_t = readtable("./hdl_chr22_typed.zsc", separator=' ')
```

```
Out[2]: 3000Œ5 DataFrames.DataFrame
         Row    rsID           pos        A0   A1   Z

         1      "rs5746647"    17057138   "G"  "T"  1.9927
         2      "rs5747968"    17067504   "G"  "T"  3.07937
         3      "rs2236639"    17072483   "A"  "G"  2.16346
         4      "rs5746679"    17080378   "A"  "G"  2.58537
         5      "rs11089263"   17087656   "C"  "A"  0.5
         6      "rs2096537"    17094749   "A"  "C"  -0.291139
         7      "rs4819849"    17152611   "A"  "G"  2.12903
         8      "rs2845379"    17202602   "T"  "C"  -0.0188679
         9      "rs2845346"    17214252   "C"  "T"  -0.144928
         10     "rs1807512"    17221495   "T"  "C"  0.190476
         11     "rs5748593"    17227461   "T"  "C"  -0.028169

         2989   "rs6010023"    51028202   "C"  "T"  -0.731959
         2990   "rs140510"     51052379   "T"  "C"  0.0384615
         2991   "rs131729"     51053719   "T"  "C"  0.581818
         2992   "rs3091400"    51059118   "G"  "T"  -0.344828
         2993   "rs8142033"    51062832   "G"  "A"  1.18462
         2994   "rs9616812"    51105556   "C"  "T"  -1.625
         2995   "rs2341009"    51133580   "C"  "A"  -1.13433
         2996   "rs736334"     51139178   "C"  "T"  0.278689
         2997   "rs6010063"    51156933   "A"  "G"  0.44
         2998   "rs8137951"    51165664   "G"  "A"  -0.368421
         2999   "rs3810648"    51175626   "A"  "G"  0.415842
         3000   "rs2238837"    51212875   "A"  "C"  0.448718
```

```
In [3]: # there are only 3000 genotyped SNPs
        println(size(zsc_t))

(3000,5)


In [4]: # read in reference panel from 1000 genomes project for chromosome 22
        refpanel = SnpData("./1000G.EUR.22");

In [5]: # the reference panel has 17,489 SNPs on chromosome 22
        println(size(refpanel.snpmatrix))

(489,17489)


In [6]: # perform imputation using the method described above
        # this will take less than 20 seconds
        @time zsc_imp = impute_zscore(zsc_t, refpanel)

 12.159418 seconds (56.25 M allocations: 1.758 GB, 3.29% gc time)


Out[6]: 10268×6 DataFrames.DataFrame
         Row    rsID          pos       A0   A1   Z           r2pred

         1      "rs2379981"   17030792  "G"  "A"  1.92311     0.705895
         2      "rs4535153"   17031072  "C"  "T"  1.92311     0.705895
         3      "rs9605903"   17054720  "C"  "T"  2.07014     0.740814
         4      "rs5747988"   17073066  "A"  "G"  2.14202     0.888847
         5      "rs5746664"   17074622  "A"  "C"  2.23801     0.914545
         6      "rs2070501"   17084609  "A"  "G"  0.345276    0.915018
         7      "rs16984366"  17096864  "C"  "T"  -1.94536    0.721418
         8      "rs8137637"   17103717  "G"  "T"  -1.98102    0.729256
         9      "rs4410381"   17107266  "A"  "G"  -2.10321    0.73318
         10     "rs5993671"   17116398  "G"  "T"  -0.246478   0.894049
         11     "rs5992472"   17132490  "G"  "A"  -0.269963   0.886824

         10257  "rs762672"    51064818  "T"  "C"  -0.0281581  0.859197
         10258  "rs10854884"  51101899  "A"  "C"  1.52138     0.818583
         10259  "rs8138460"   51103692  "G"  "A"  1.68241     0.837635
         10260  "rs9616906"   51104680  "A"  "G"  1.64511     0.851399
         10261  "rs9628185"   51109992  "C"  "T"  1.55364     0.908213
         10262  "rs9616915"   51117580  "T"  "C"  -1.57798    0.820481
         10263  "rs9616816"   51123505  "A"  "G"  -0.350208   0.805102
         10264  "rs739365"    51140316  "T"  "C"  -0.516951   0.679435
         10265  "rs10451"     51162059  "A"  "G"  0.18436     0.904821
         10266  "rs715586"    51163138  "T"  "C"  -0.778781   0.658237
         10267  "rs2285395"   51178090  "A"  "G"  -0.374061   0.855836
         10268  "rs3865766"   51186228  "T"  "C"  -0.473425   0.735647
```

```
In [7]: # the imputed data set has much more SNPs
        # the last column r2pred is a measure of imputation accuracy
        # note that this module filters out poorly imputed SNPs (r2pred < 0.6) by default
        println(size(zsc_imp))
```

```
(10268,6)
```

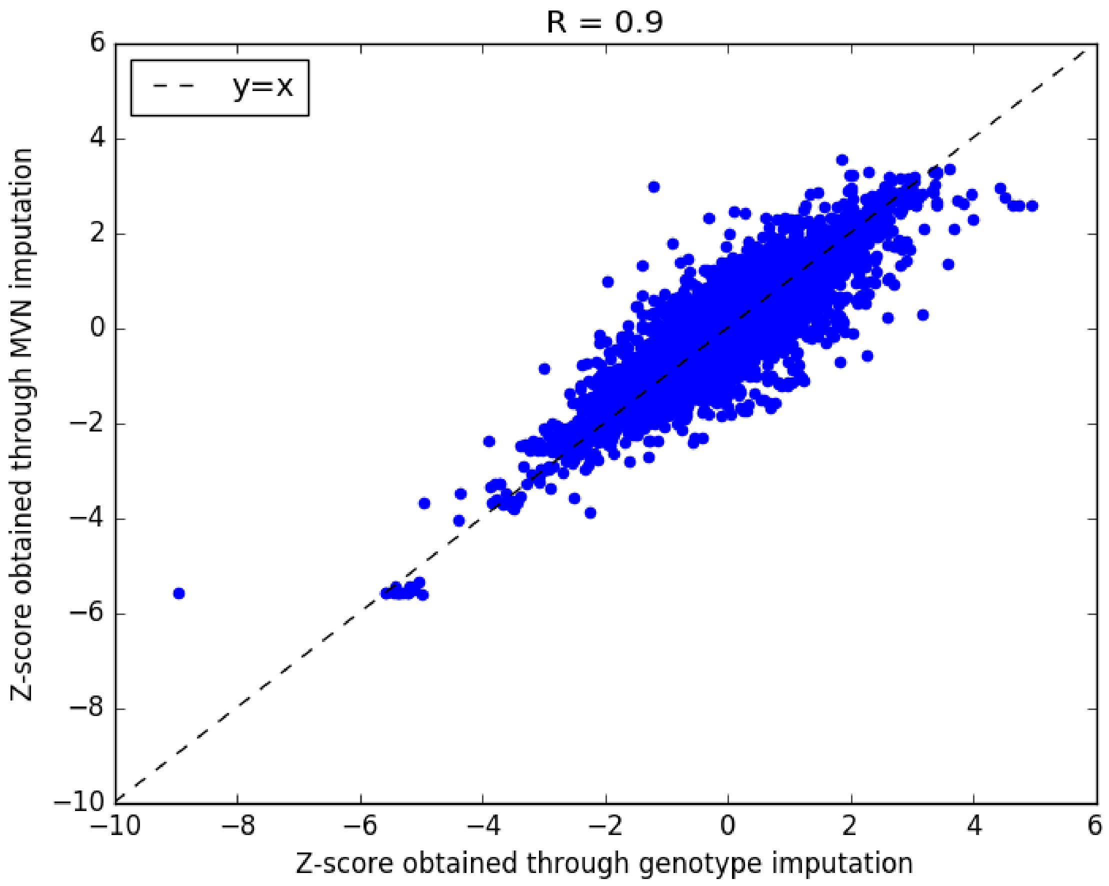
```
In [8]: # now let's compare the imputed z-scores with the z-scores obtained by first
        # performing a genotype imputation
        zsc_full = readtable("./hdl_chr22_full.zsc", separator=' ');

        # this removes SNPs with duplicated SNP ID and position and makes the
        # sign consistent
        snp_legend = DataFrame(rsID = refpanel.snpid, pos = refpanel.basepairs,
            A0 = refpanel.allele1, A1 = refpanel.allele2)
        filter_input!(zsc_full, snp_legend)
```

```
In [9]: # match z-scores based on SNP ID
        zsc_matched = join(zsc_full, zsc_imp, on=:rsID)
        println(zsc_matched)
```

```
7444×10 DataFrames.DataFrame
 Row   rsID         pos       A0   A1   Z           pos_1     A0_1

 1     "rs1000427"  36890105  "G"  "A"  -0.697368   36890105  "A"
 2     "rs1000815"  26831077  "A"  "G"  0.166667    26831077  "A"
 3     "rs1001021"  26403599  "G"  "A"  -0.290598   26403599  "A"
 4     "rs1001213"  34131736  "G"  "A"  1.66346     34131736  "A"
 5     "rs1001586"  42670293  "G"  "T"  0.516667    42670293  "T"
 6     "rs1001587"  42670111  "C"  "T"  0.516667    42670111  "T"
 7     "rs1001794"  32850930  "C"  "T"  -0.163636   32850930  "T"
 8     "rs1001896"  19032215  "G"  "A"  -0.580645   19032215  "A"
 9     "rs1002048"  34253393  "G"  "T"  1.2         34253393  "G"
 10    "rs1002189"  30771458  "T"  "C"  -1.69643    30771458  "C"
 11    "rs1003480"  31346752  "A"  "G"  -0.1875     31346752  "A"

 7433  "rs9941935"  22015144  "A"  "G"  2.8         22015144  "G"
 7434  "rs9941962"  40172198  "A"  "G"  -0.525641   40172198  "G"
 7435  "rs9941971"  26051000  "T"  "C"  1.15238     26051000  "C"
 7436  "rs9956"     32015450  "T"  "G"  1.31481     32015450  "G"
 7437  "rs9967"     18211205  "T"  "C"  1.08333     18211205  "C"
 7438  "rs997120"   33108536  "C"  "T"  -0.102564   33108536  "T"
 7439  "rs997379"   35274298  "G"  "T"  0.270833    35274298  "G"
 7440  "rs9983"     30423744  "G"  "A"  0.461538    30423744  "A"
 7441  "rs998482"   41108135  "G"  "A"  -0.179104   41108135  "A"
 7442  "rs9985182"  45539841  "G"  "T"  -0.0392157  45539841  "T"
 7443  "rs999540"   37121101  "G"  "A"  0.174603    37121101  "A"
 7444  "rs9997"     20796175  "T"  "C"  0.7         20796175  "C"
```

```
Row   A1_1  Z_1          r2pred

1     "G"   -0.728381    0.924804
2     "G"   -0.210362    0.942315
3     "G"   -0.476867    0.903856
4     "G"   1.24971      0.911577
5     "G"   0.440369     0.979745
6     "C"   0.473304     0.976182
7     "C"   0.086738     0.897738
8     "G"   -0.530442    0.894444
9     "T"   1.20897      0.905532
10    "T"   -1.8985      0.963061
11    "G"   -0.0941326   0.942817

7433  "A"   2.83774      0.817645
7434  "A"   -0.706095    0.974607
7435  "T"   1.11075      0.959307
7436  "T"   0.948379     0.956148
7437  "T"   1.30208      0.914667
7438  "C"   -1.09414     0.809205
7439  "T"   0.451928     0.957169
7440  "G"   0.416925     0.976849
7441  "G"   -0.0895031   0.971362
7442  "G"   -0.0544267   0.859937
7443  "G"   -0.00408845  0.79141
7444  "T"   0.850073     0.952408
```

```
In [11]: # make scatter plot comparing the z-scores obtained through different methods
         using PyPlot
         scatter(zsc_matched[:Z], zsc_matched[:Z_1], color="blue")
         plot([-10, 6], [-10, 6], color="black", linestyle="--", label="y=x")
         legend(loc=2)
         xlim([-10, 6]); ylim([-10, 6])
         xlabel("Z-score obtained through genotype imputation")
         ylabel("Z-score obtained through MVN imputation")
         title("R = " * string(round(cor(zsc_matched[:Z], zsc_matched[:Z_1]),2)))
```



```
Out[11]: PyObject <matplotlib.text.Text object at 0x7f90ed6fbbd0>

In [ ]:
```