Covar.cpp:114: LOGGER.i(0, "Reading quantitative covariates from [" + filename + "].");

Covar.cpp:119: LOGGER.i(0, to\_string(qcovar.size()) + " covariates of " + to\_string(samples\_qcovar.size()) + " samples to be included.");

Covar.cpp:124: LOGGER.i(0, "Reading discrete covariates from [" + filename + "].");

Covar.cpp:129: LOGGER.i(0, to\_string(covar.size()) + " covariates of " + to\_string(samples\_covar.size()) + " samples to be included.");

Covar.cpp:135: LOGGER.i(0, "Reading ranked covariates from [" + filename + "].");

Covar.cpp:140: LOGGER.i(0, to\_string(rcovar.size()) + " covariates of " + to\_string(samples\_rcovar.size()) + " samples to be included.");

Covar.cpp:220: LOGGER.i(0, to\_string(qcovar.size()) + " qcovar, " + to\_string(covar.size()) + " covar and " + to\_string(rcovar.size()) + " rcovar to be included.");

Covar.cpp:226: LOGGER.i(0, to\_string(sample\_id.size()) + " common individuals among the covariate files to be included.");

Covar.cpp:543: LOGGER.w(0, "inconsistent column number in line " + to\_string(line\_number) + " from " + err\_string);

FastFAM.cpp:632: LOGGER.i(0, to\_string(remain\_index.size()) + " overlapping individuals with non-missing data to be included from the covariate file(s).");

FastFAM.cpp:683: LOGGER.i(0, "After matching all the files, " + to\_string(remain\_phenos.size()) + " individuals to be included in the analysis.");

FastFAM.cpp:718: LOGGER.w(0, "" + to\_string(num\_remain\_col - num\_remain\_col2) + " duplicated covariates are removed.");

FastFAM.cpp:790: LOGGER.i(0, "Fitting covariates jointly in the association analysis.");

FastFAM.cpp:840: LOGGER.i(0, "Estimating the genetic variance (Vg) by " + mtdString[options["VgEstMethod"]] + "...");

FastFAM.cpp:843: LOGGER.i(0, "Using related pairs only.");

FastFAM.cpp:868: LOGGER.w(0, " Forcing the program to run fastGWA MLM");

FastFAM.cpp:876: LOGGER.w(0, "Constraining Vg to 0.");

FastFAM.cpp:880: LOGGER.w(0, "Vg is larger than Vp");

FastFAM.cpp:883: LOGGER.w(0, "Constraining Vg to 0.99 \* Vp: " + to\_string(VG) + ".");

FastFAM.cpp:896: LOGGER.w(0, "the estimate of Vg is not statistically significant (i.e., p > 0.05). "

FastFAM.cpp:915: LOGGER.i(0, "Saving the inverse of V matrix for further analysis, to be used by --load-inv for further analysis");

FastFAM.cpp:936: LOGGER.i(0, "The inverse of V matrix has been saved to [" + options["out"] + ".grm.inv]");

FastFAM.cpp:940: //LOGGER.i(0, "Performing GRAMMAR-Gamma approximation...");

FastFAM.cpp:955: LOGGER.w(0, "The estimate of Vg is not statistically significant (i.e., p > 0.05). "

FastFAM.cpp:974: LOGGER.i(0, to\_string(cur\_index) + " samples are confirmed to be identical in V inverse [" + id\_file + "].");

FastFAM.cpp:981: LOGGER.i(0, "Loading the inverse of V martrix from " + in\_name + "...");

FastFAM.cpp:1005: LOGGER.i(0, "Inverse of V matrix loaded in " + to\_string(LOGGER.tp("LOAD\_INV")) + " seconds.");

FastFAM.cpp:1116: LOGGER.w(0, "Highly correlated columns are found in the covariate matrix. The results are still reliable in most cases.");

FastFAM.cpp:1376: LOGGER.w(0, "the XtX matrix is not invertible.");

FastFAM.cpp:1684: LOGGER.i(2, "Vg = " + to\_string(hsq) + ", se = " + to\_string(se) + ", P = " + to\_string(p));

FastFAM.cpp:1697: LOGGER.i(0, "Reading the sparse GRM file from [" + filename + "]...");

FastFAM.cpp:1820: LOGGER.i(0, "\nTuning parameters using " + to\_string(num\_marker\_rand) + " null SNPs...");

FastFAM.cpp:1944: LOGGER.i(0, "Mean GRAMMAR-Gamma value = " + to\_string(c\_inf));

FastFAM.cpp:1973: LOGGER.i(0, "\nInverting the variance-covariance matrix by " + options["inv\_method"] + " (This may take a long time)...");

FastFAM.cpp:2000: LOGGER.i(0, "The V matrix inverted in " + to\_string(LOGGER.tp("INVERSE\_FAM")) + " sec.");

FastFAM.cpp:2711: LOGGER.i(0, "Use --load-inv to load the V inverse file for fastGWA");

FastFAM.cpp:2715: LOGGER.i(0, "Use \"--load-model " + options["out"] + "\" to load the model file for further analysis.");

FastFAM.cpp:2725: LOGGER.i(0, "\nPerforming fastGWA generalized linear mixed model association analysis...");

FastFAM.cpp:2729: LOGGER.i(0, "\nPerforming fastGWA mixed model association analysis (exact test)...");

FastFAM.cpp:2732: LOGGER.i(0, "\nPerforming fastGWA mixed model association analysis...");

FastFAM.cpp:2737: LOGGER.i(0, "\nPerforming fastGWA linear regression analysis...");

FastFAM.cpp:2769: LOGGER.i(0, "Reading gene list file from [" + gfile + "]...");

FastFAM.cpp:2892: LOGGER.i(1, ss.str());

FastFAM.cpp:2901: LOGGER.i(0, ss.str());

FastFAM.cpp:2999: LOGGER.w(0, "GLM didn't converge.");

FastFAM.cpp:3278: LOGGER.w(0, "fastGWA-BB-REML did not converge, although the results are reliable in most cases.");

FastFAM.cpp:3422: LOGGER.i(0, "\nTuning parameters using null SNPs...");

FastFAM.cpp:3518: LOGGER.i(0, "Mean GRAMMAR-Gamma value = " + to\_string(c\_inf));

GRM.cpp:132: LOGGER.i(0, "Reading [" + files[0] + ".grm.id]...");

GRM.cpp:135: LOGGER.i(0, "Reading [" + files[1] + ".grm.id]...");

GRM.cpp:142: LOGGER.i(2, "Saving " + to\_string(common\_id.size()) + " individual IDs");

GRM.cpp:176: LOGGER.i(0, "Subtracting GRMs...");

GRM.cpp:197: LOGGER.i(0, "The subtracted GRM has been written to [" + out\_file + ".grm.bin, .grm.N.bin].");

GRM.cpp:237: LOGGER.i(0, "Reading [" + files[0] + ".grm.id]...");

GRM.cpp:246: LOGGER.i(0, "Reading [" + cur\_file + "]...");

GRM.cpp:258: LOGGER.i(0, "Keeping individuals listed in [" + options["keep\_file"] + "]...");

GRM.cpp:270: LOGGER.i(0, "Excluding individuals listed in [" + options["remove\_file"] + "]...");

GRM.cpp:306: LOGGER.i(0, "Unified individual IDs have been saved to [" + id\_file\_name + "].");

GRM.cpp:309: LOGGER.i(0, "Writing unified GRM in binary format...");

GRM.cpp:371: LOGGER.i(0, "GRM has been written to [" + wfile\_name + "].");

GRM.cpp:377: LOGGER.i(0, "Pruning the GRM to a sparse matrix with a cutoff of " + to\_string(thresh) + "...");

GRM.cpp:378: LOGGER.i(0, "Total number of parts to be processed: " + to\_string(index\_grm\_pairs.size()));

GRM.cpp:398: LOGGER.i(2, "Saving " + to\_string(keep\_ID.size()) + " individual IDs");

GRM.cpp:420: LOGGER.i(2, "Processing part " + to\_string(part\_index + 1));

GRM.cpp:480: LOGGER.i(0, "Saving the sparse GRM (" + to\_string(rm\_grm.size()) + " pairs) to [" + options["out"] + ".grm.sp]");

GRM.cpp:488: LOGGER.i(0, "Success:", "finished generating a sparse GRM");

GRM.cpp:491: LOGGER.i(0, "GRM has been saved to [" + options["out"] + ".grm.bin]");

GRM.cpp:502: LOGGER.w(0, "Reading GRM N failed between line " + to\_string(index\_grm\_pairs[part\_index].first + 1) + " and "

GRM.cpp:504: LOGGER.i(0, "Stop pruning the GRM N");

GRM.cpp:508: LOGGER.i(2, "Processing part " + to\_string(part\_index + 1));

GRM.cpp:535: LOGGER.i(0, "GRM N has been saved to [" + options["out"] + ".grm.N.bin]");

GRM.cpp:541: LOGGER.i(0, "Pruning the GRM with a cutoff of " + to\_string(thresh) + "...");

GRM.cpp:542: LOGGER.i(0, "Total number of parts to be processed: " + to\_string(index\_grm\_pairs.size()));

GRM.cpp:577: LOGGER.i(2, "Processing part " + to\_string(part\_index + 1));

GRM.cpp:604: LOGGER.i(0, "Related family pairs have been saved to " + options["out"] + ".family.txt");

GRM.cpp:648: LOGGER.i(0, "After pruning the GRM, there are " + to\_string(keep\_ID.size()) + " individuals (" + to\_string(removed\_ID.size()) + " individuals removed).");

GRM.cpp:652: LOGGER.i(2, "Pruned unrelated IDs have been saved to " + options["out"] + ".grm.id");

GRM.cpp:658: LOGGER.i(2, "Pruned singleton IDs have been saved to " + options["out"] + ".singleton.txt");

GRM.cpp:666: LOGGER.i(0, "Pruning GRM values, total parts " + std::to\_string(index\_grm\_pairs.size()));

GRM.cpp:675: LOGGER.i(2, "GRM values have been saved to [" + options["out"] + ".grm.bin]");

GRM.cpp:677: LOGGER.i(0, "Pruning number of SNPs to calculate GRM, total parts " + std::to\_string(index\_grm\_pairs.size()));

GRM.cpp:680: LOGGER.w(2, "There is no [" + grm\_file + ".grm.N.bin]");

GRM.cpp:685: LOGGER.w(2, "can't open [" + options["out"] + ".grm.N.bin] to write. Ignore this step");

GRM.cpp:692: LOGGER.i(2, "Number of SNPs has been saved to [" + options["out"] + ".grm.N.bin]");

GRM.cpp:706: LOGGER.i(2, "Processing part " + to\_string(part\_index + 1));

GRM.cpp:752: LOGGER.w(0, "cannot divide into " + to\_string(num\_parts) + ". Use " + to\_string(parts.size()) + " instead.");

GRM.cpp:827: LOGGER.w(0, "cannot run in " + to\_string(num\_thread) + " threads. Use " + to\_string(thread\_parts.size()) + " instead");

GRM.cpp:847: LOGGER.i(0, com\_string);

GRM.cpp:848: LOGGER.i(0, "Subset " + to\_string(part) + "/" + to\_string(num\_parts) + ", no. subject " + to\_string(part\_keep\_indices.first + 1) + "-" + to\_string(part\_keep\_indices.second + 1));

GRM.cpp:849: LOGGER.i(1, to\_string(num\_individual) + " samples, " + to\_string(marker->count\_extract()) + " markers, " + to\_string(num\_grm) + " GRM elements");

GRM.cpp:875: LOGGER.i(0, "IDs for the GRM file have been saved in the file [" + o\_grm\_id + "]");

GRM.cpp:1331: //LOGGER.i(0, out\_message.str() + "% has been finished");

GRM.cpp:1357: LOGGER.i(0, "The GRM computation is completed.");

GRM.cpp:1363: LOGGER.i(0, "Saving sparse GRM with a cutoff " + to\_string(thresh) + "...");

GRM.cpp:1365: LOGGER.i(0, "Saving GRM...");

GRM.cpp:1514: LOGGER.i(0, "GRM has been saved in the file [" + o\_name + ".grm.bin]");

GRM.cpp:1515: LOGGER.i(0, "Number of SNPs in each pair of individuals has been saved in the file [" + o\_name + ".grm.N.bin]");

GRM.cpp:1517: LOGGER.i(0, "GRM has been saved in the file [" + o\_name + ".grm.sp]");

GRM.cpp:2048: LOGGER.i(0, "Note: GRM is computed using the SNPs on the autosomes.");

GRM.cpp:2057: LOGGER.i(0, "Note: this function takes X chromosome as non-PAR region.");

Geno.cpp:316: LOGGER.i(0, "Computing allele frequencies...");

Geno.cpp:367: LOGGER.i(0, to\_string(extract\_index.size()) + " SNPs remain from --maf or --max-maf, ");

Geno.cpp:386: LOGGER.i(0, "Reading frequencies from [" + alleleFileName + "]...");

Geno.cpp:418: LOGGER.i(0, "Frequencies of " + to\_string(AFA1.size()) + " SNPs are updated.");

Geno.cpp:454: LOGGER.i(0, "Saving allele frequencies...");

Geno.cpp:466: LOGGER.i(0, "Allele frequencies of " + to\_string(AFA1.size()) + " SNPs have been saved in the file [" + name\_frq + "]");

Geno.cpp:517: LOGGER.i(0, "BED file(s) check OK.");

Geno.cpp:1549: LOGGER.i(1, ss.str());

Geno.cpp:1558: LOGGER.i(1, ss.str());

Geno.cpp:2968: LOGGER.i(1, ss.str());

Geno.cpp:3258: LOGGER.i(0, "Reading PLINK BED file(s) in SNP-major format...");

Geno.cpp:3338: LOGGER.i(1, ss.str());

Geno.cpp:3346: LOGGER.i(1, ss.str());

Geno.cpp:3478: LOGGER.w(0, "Geno: multiple " + key\_name + ", use the first one only" );

Geno.cpp:3583: LOGGER.w(0, "--freq should not be followed by any parameter, if you want to calculate the allele frequencies in founders only, "

Geno.cpp:3596: LOGGER.w(0, "--freq should not be followed by any other parameter, if you want to calculate the allele frequencies in founders only, "

Geno.cpp:3723: LOGGER.i(1, ss.str());

Geno.cpp:3731: LOGGER.i(1, ss.str());

Geno.cpp:3782: LOGGER.i(1, ss.str());

Geno.cpp:3790: LOGGER.i(1, ss.str());

Geno.cpp:3971: LOGGER.i(0, "Computing allele frequencies...");

Geno.cpp:4028: LOGGER.i(0, "Allele frequencies of " + to\_string(nValidMarker) + " SNPs have been saved in the file [" + name\_frq + "]");

Geno.cpp:4094: LOGGER.i(0, "Computing allele frequencies...");

Geno.cpp:4111: LOGGER.i(0, "Saving genotype to PLINK binary PED format [" + filename + ".bed]...");

Geno.cpp:4115: LOGGER.i(0, "Genotype has been saved.");

Geno.cpp:4125: LOGGER.i(0, "Converting bgen to PLINK binary PED format [" + filename + ".bed]...");

Geno.cpp:4127: LOGGER.i(0, "Genotype has been saved.");

Geno.cpp:4136: LOGGER.i(0, "Summing up genotypes based on sex");

Geno.cpp:4139: LOGGER.i(0, "Summary has been saved.");

LD.cpp:195: LOGGER.i(0, "Generating LD matrix...");

Marker.cpp:109: LOGGER.i(0, "Reading reference alleles of SNPs from [" + options["update\_ref\_allele\_file"] + "]...");

Marker.cpp:115: LOGGER.i(0, to\_string(index\_extract.size()) + " reference alleles are updated.");

Marker.cpp:210: LOGGER.w(0, to\_string(rm\_snps) + " SNPs are removed due to mismatched SNP IDs or alleles.");

Marker.cpp:418: LOGGER.i(0, "Saving SNP information to [" + filename + "]...");

Marker.cpp:425: LOGGER.i(0, to\_string(index\_extract.size()) + " SNPs saved.");

Marker.cpp:430: LOGGER.i(0, "Reading PLINK2 PVAR file from [" + pvar\_file + "]...");

Marker.cpp:530: LOGGER.i(0, to\_string(num\_marker) + " SNPs to be included from PVAR file(s).");

Marker.cpp:532: LOGGER.i(0, to\_string(num\_extract) + " SNPs to be included on valid chromosomes");

Marker.cpp:547: LOGGER.i(0, "Reading PLINK BIM file from [" + bim\_file + "]...");

Marker.cpp:571: LOGGER.w(0, "the bim file [" + bim\_file + "], line " + to\_string(line\_number) +

Marker.cpp:605: LOGGER.i(0, to\_string(num\_marker) + " SNPs to be included from BIM file(s).");

Marker.cpp:607: LOGGER.i(0, to\_string(num\_extract) + " SNPs to be included on valid chromosomes");

Marker.cpp:786: LOGGER.w(0, "GCTA reads sample information from '--sample' input, but ignores the built-in sample data.");

Marker.cpp:805: LOGGER.i(0, "Loading bgen index from [" + index\_fname + "]...");

Marker.cpp:906: LOGGER.w(0, "No biallelic variants.");

Marker.cpp:919: LOGGER.i(0, to\_string(n\_variants) + " SNPs to be included from bgen index file.");

Marker.cpp:1220: LOGGER.i(0, to\_string(num\_marker) + " SNPs to be included from bgen file.");

Marker.cpp:1296: if(numDup != 0) LOGGER.w(0, to\_string(numDup) + " duplicated SNPs were ignored in the list." );

Marker.cpp:1317: LOGGER.i(0, string("After ") + (isExtract? "extracting" : "excluding") + " SNP, " + to\_string(num\_extract) + " SNPs remain.");

Marker.cpp:1411: LOGGER.w(0, "the SNP list file [" + snplist\_file + "], line " + to\_string(line\_number) +

Marker.cpp:1428: LOGGER.w(0, "Marker: multiple " + key\_name + ", using the first one only" );

Marker.cpp:1498: LOGGER.w(0, "One of the chromosome filtering criteria has been applied, which has been overridden by --autosome-x-y");

Marker.cpp:1514: LOGGER.w(0, "One of the CHR filtering criteria has been applied, which has been overridden by --chr");

Marker.cpp:1544: LOGGER.w(0, "One of the CHR filtering criteria has been applied, which has been overridden by --chrx");

OptionIO.cpp:18: LOGGER.w(0, "There are multiple " + key\_name + ". Only the first item will be used in the analysis." );

OptionIO.cpp:35: LOGGER.w(0, "There are multiple " + key\_name + ". Only the first item will be used in the analysis." );

OptionIO.cpp:171: LOGGER.w(0, "the file [" + fileName + "] contains extra number of elements in line " + to\_string(line\_number) + ".");

Pheno.cpp:90: LOGGER.i(0, "Reading phenotype data from [" + options["qpheno\_file"] + "]...");

Pheno.cpp:112: LOGGER.i(0, to\_string(index\_keep.size()) + " overlapping individuals with non-missing data to be included from the phenotype file.");

Pheno.cpp:118: LOGGER.i(0, "Reading gender information from [" + options["sex\_file"] + "]...");

Pheno.cpp:125: LOGGER.i(0, to\_string(index\_keep.size()) + " individuals with valid sex information to be included from the phenotype file.");

Pheno.cpp:182: LOGGER.i(0, to\_string(new\_index.size()) + " individuals have gender information.");

Pheno.cpp:277: LOGGER.w(0, errmsg);

Pheno.cpp:347: LOGGER.w(0, "Sample file [" + mfiles[i] + "] is different from the first file.");

Pheno.cpp:353: LOGGER.i(1, "All the files checked are OK.");

Pheno.cpp:359: LOGGER.i(0, "Reading PLINK sample file from [" + psam\_file + "]...");

Pheno.cpp:435: LOGGER.i(0, to\_string(num\_ind) + " individuals to be included from the sample file.");

Pheno.cpp:444: LOGGER.i(0, "Reading Oxford sample information file from [" + sample\_file + "]...");

Pheno.cpp:503: LOGGER.i(0, to\_string(num\_ind) + " individuals to be included from the sample file.");

Pheno.cpp:509: LOGGER.i(0, "Reading PLINK FAM file from [" + fam\_file + "]...");

Pheno.cpp:528: LOGGER.w(0, "the fam file [" + fam\_file + "], line " + to\_string(line\_number) + " have different elements");

Pheno.cpp:545: LOGGER.i(0, to\_string(num\_ind) + " individuals to be included from FAM file.");

Pheno.cpp:601: LOGGER.i(0, "Saving individual information to [" + filename + "]...");

Pheno.cpp:609: LOGGER.i(0, to\_string(index\_keep.size()) + " individuals have been saved.");

Pheno.cpp:669: LOGGER.w(0, to\_string(nDup) + " duplicated samples were ignored in the list.");

Pheno.cpp:687: LOGGER.i(0, string("After ") + (isKeep?"keeping":"removing") + " individuals, " + to\_string(keeps.size()) + " subjects remain.");

main.cpp:286: LOGGER.i(0, "Options: ");

main.cpp:287: LOGGER.i(0, " ");

main.cpp:295: LOGGER.i(0, "");

main.cpp:299: LOGGER.i(0, "The program will be running with up to " + std::to\_string(thread\_num) + " threads.");

main.cpp:315: LOGGER.i(0, "");

main.cpp:317: LOGGER.i(0, "at " + getLocalTime(), "Analysis finished");