Supplementary Note: Comparison between bigstatsr and snpnet for fitting penalized regressions on very large genetic data

Penalized regression with L1 penalty, also known as "lasso", has been widely used since it proved to be an effective method for simultaneously performing variable selection and model fitting (Tibshirani 1996). R package glmnet is a popular software to fit the lasso efficiently (Friedman *et al.* 2010). However, glmnet cannot handle very large datasets such as biobank-scale data that are now available in human genetics, where both the sample size and the number of variables are very large. One strategy used to run penalized regressions on large datasets such as the UK Biobank (Bycroft *et al.* 2018) has been to apply a variable pre-selection step before fitting the lasso (Lello *et al.* 2018). Recently, authors of the glmnet package have developed a new R package, snpnet, to fit penalized regressions on the UK Biobank without having to perform any pre-filtering (Qian *et al.* 2020). Earlier, we developed two R packages for efficiently analyzing large-scale (genetic) data, namely bigstatsr and bigsnpr (Privé *et al.* 2018). We then specifically derived a highly efficient implementation of penalized linear and logistic regressions in R package bigstatsr, and showed how these functions were useful for genetic prediction with some applications to the UK Biobank (Privé *et al.* 2019). Here we benchmark bigstatsr against snpnet for fitting penalized regressions on large genetic data. Through some theoretical expectations and empirical comparisons, we show that package bigstatsr is generally much faster than snpnet. We also take that opportunity to provide more recommendations on how to fit penalized regressions in the context of genetic data.

Main motivation for snpnet

Before we can present the main motivation behind snpnet developed by Qian *et al.* (2020), let us recall how the lasso regression is fitted. Fitting the lasso consists in finding regression coefficients β that minimize the following regularized loss function

$$L(\lambda) = \underbrace{\sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} X_{i,j}\beta_j \right)^2}_{\text{Loss function}} + \underbrace{\lambda \sum_{j=1}^{p} |\beta_j|}_{\text{Penalization}}, \tag{1}$$

where X denotes the matrix composed of p (standardized) genotypes and possible covariates (e.g. sex, age and principal components) for n individuals, y is the (continuous) trait to predict, λ (> 0) is a regularization hyper-parameter that control the strength of the penalty. For a sequence of λ, one can find $\text{argmin}_\beta L(\lambda)$ using cyclical coordinate descent (Friedman *et al.* 2010). To speed up the coordinate descent, one can use sequential strong rules for discarding lots of variables, i.e. setting lots of β_j to 0, a priori (Tibshirani *et al.* 2012). Therefore the cyclical coordinate descent used to solve the lasso can be performed in a subset of the data only thanks to these strong rules. However, the main drawback of these strong rules is that they require checking Karush-Kuhn-Tucker (KKT) conditions a posteriori, usually in two phases. These KKT conditions are first checked in the ever-active set, i.e. the set of all variables j with $\beta_j \neq 0$ for any previous λ. Then, the cyclical coordinate descent has to be rerun while adding the new variables that do not satisfy these KKT conditions (if any). In a second phase, the KKT conditions are also checked for all the remaining variables, i.e. the ones not in the ever-active set. This last step requires to pass over the whole dataset at least once again for every λ tested. Then, when the available random access memory (RAM) is not large enough to cache the whole dataset, data has to be read from disk, which can be extremely time consuming. To alleviate this particular issue, Qian *et al.* (2020) have developed a clever approach called batch screening iterative lasso (BASIL) to be able to check these KKT conditions on the whole dataset only after having fitted solutions for many λ, instead of performing this operation for each λ. Hence, for very large datasets, the BASIL procedure enables to fit the exact lasso solution faster than when checking the KKT conditions for all variables at each λ, as performed in e.g. R package biglasso (Zeng and Breheny 2017).

A more pragmatic approach in bigstatsr

In our R package bigstatsr, we proposed a different strategy. We also check the KKT conditions for variables in the ever-active set, i.e. for a (small) subset of all variables only; this first checking is therefore fast. However, KKT conditions almost always hold when $p > n$ (Tibshirani *et al.* 2012), which is particularly the case for the remaining variables in the second phase of checking. Because of this, we decided in Privé *et al.* (2019) to skip this second checking when designing functions `big_spLinReg` and `big_spLogReg` for fitting penalized regression on very large datasets in R package bigstatsr. Thanks to this approximation, these two functions effectively access all variables only once at the very beginning to compute the statistics used by the strong rules, and then access a subset of variables only (the ever-active set). As we show later, this means that fitting penalized regressions using the approximation we proposed in Privé *et al.* (2019) is computationally more efficient than using the BASIL procedure proposed by Qian *et al.* (2020), and yet provides equally accurate predictors. Moreover, as bigstatsr uses memory-mapping, data that resides on disk is accessed only once from disk to memory and then stays in memory while there is no need to free memory. Only when the ever-active set becomes very large, e.g. for very polygenic traits, memory can become an issue, but this extreme case would become a problem for package snpnet as well. Please refer to the Discussion section of Privé *et al.* (2019) for more details on these matters. In summary, bigstatsr effectively performs only one pass on the whole dataset while snpnet performs many passes, even though the number of passes in snpnet is reduced thanks to

the BASIL approach. Moreover, bigstatsr still uses a single pass even when performing CMSA (a variant of cross-validation (CV), see figure 1) internally, whereas performing CV with snpnet would multiply the number of passes to the data by the number of folds used in the CV.

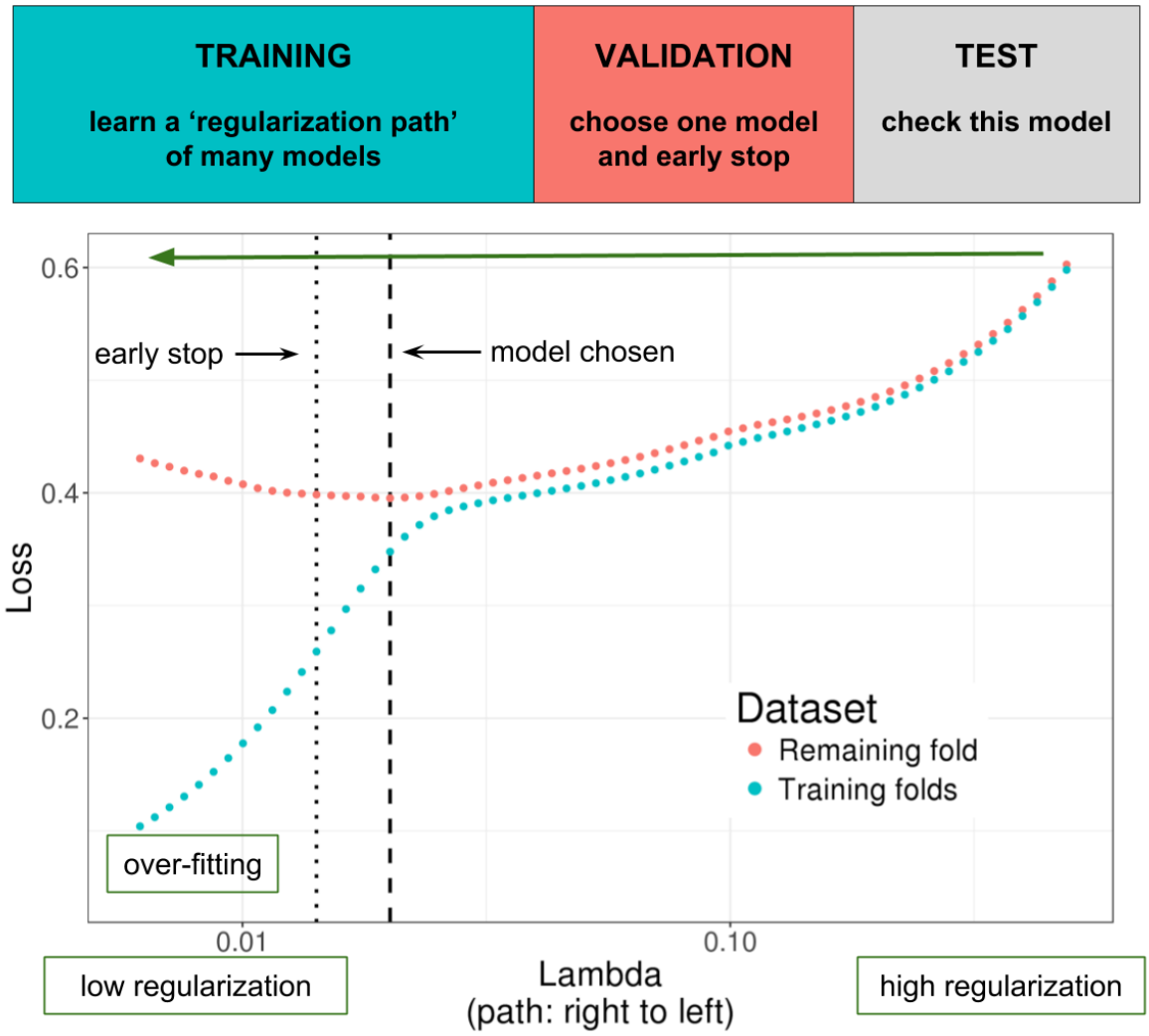

Figure 1: Illustration of one turn of the Cross-Model Selection and Averaging (CMSA) procedure. This figure comes from Privé *et al.* (2019); the Genetics Society of America has granted us permission to re-use it. First, this procedure separates the training set in K folds (e.g. 10 folds). Secondly, in turn, each fold is considered as an inner validation set (red) and the other $(K-1)$ folds form an inner training set (blue). A "regularization path" of models is trained on the inner training set and the corresponding predictions (scores) for the inner validation set are computed. The model that minimizes the loss on the inner validation set is selected. Finally, the K resulting models are averaged; this is different to standard cross-validation where the model is refitted on the whole training set using the best-performing hyper-parameters. We also use this procedure to derive an early stopping criterion so that the algorithm does not need to evaluate the whole regularization paths, making this procedure much faster.

Benchmark

Before, we have presented why we expect bigstatsr to be more efficient than snpnet. To practically support this claim, we perform comparisons for the four real traits used in the UK Biobank analyses of Qian *et al.* (2020). We compare R package snpnet (v0.3) with bigstatsr (v1.3) and bigsnpr (v1.5). We use similar quality controls as Qian *et al.* (2020) (see "Data & Methods"). We also use the same splitting strategy: 20% test, 20% validation and 60% training. To use the same sets for bigstatsr as for snpnet, we use the same test set, use K=4 folds for training with bigstatsr while making sure the first split is composed of the same 20% of the data used for validation in snpnet. Moreover, we use penalty factors to effectively use unscaled genotypes in bigstatsr (see "Conclusion & further recommendations"), as performed by default in snpnet. This enables us to compare predictions from snpnet and bigstatsr using the exact same model and the same single validation fold. Note that, to make the most of the training set, bigstatsr uses CMSA (Figure 1) while Qian *et al.* (2020) propose to refit the model (on the whole training + validation) using the best λ identified using the validation set in snpnet. Also note that the parallelism used by snpnet and bigstatsr is different; snpnet relies on PLINK 2.0 to check KKT conditions in parallel, while bigstatsr parallelizes fitting of models from different folds and hyper-parameters. Because bigstatsr uses memory-mapping, the data is shared across processes and therefore it can fit these models in parallel without multiplying the memory needed. We allow for 16 cores to be used in these comparisons; bigstatsr effectively uses only 4 here (the number of folds). We allow for 128 GB of RAM for bigstatsr, but allow for 500 GB of RAM for snpnet because we had memory issues running it with only 128 GB or 256 GB.

Table 1 presents the results of this benchmark. Fitting lasso is 35 times faster using bigstatsr than using snpnet for high cholesterol, 29 times faster for asthma, 16 times faster for BMI, and 4.5 times faster for height. When using only one validation fold for choosing the best-performing λ and no refitting, snpnet and bigstatsr provide the same predictive performance, validating the use of the approximation in bigstatsr. When using the whole training set, i.e. when refitting in snpnet and using CMSA in bigstatsr, predictive performance is much higher than when the validation set is not used for training. For example, partial correlation for height is of 0.6116 with CMSA (i.e. using the average of 4 models) compared to 0.5856 when using only one of these models, showing how important it is to make the most of the training + validation sets. Also, CMSA can provide slightly higher predictive performance than the refitting strategy in snpnet, with e.g. a partial correlation of 0.3324 vs 0.3221 for BMI.

Conclusion & further recommendations

We have found the BASIL approach derived in Qian *et al.* (2020) to be a clever approach that alleviates the I/O problem of other penalized regression implementations for very large datasets. BASIL makes significant and valuable contributions to the important problem of fitting penalized regression models efficiently. However, we also find that the implementation of BASIL in snpnet is still an order of magnitude slower than our package bigstatsr, which uses a simpler and more pragmatic approach (Privé *et al.* 2019). Hereinafter we also come back to some statements made in Qian *et al.* (2020) and provide more recommendations on how to best use

Table 1: Benchmark of snpnet against bigstatsr in terms of predictive performance and computation time. Predictive performance is reported in terms of partial correlations between the polygenic scores and the phenotypes, residualized using the covariates. Timings are reported in minutes. Timings for snpnet report the training for 60% of the data (using the training set only) + the refitting for 80% of the data (using both the training and validation sets). Timings for bigstatsr report the time taken by the CMSA procedure (fitting K=4 models here).

	snpnet			bigstatsr		
Trait	Perf. (1 fold)	Perf. (refit)	Time	Perf. (1 fold)	Perf. (CMSA)	Time
Asthma	0.1349	0.1438	188 + 101	0.1348	0.1493	10
High cholesterol	0.1254	0.1366	101 + 146	0.1257	0.1387	7
BMI	0.3031	0.3231	161 + 893	0.3018	0.3324	65
Height	0.5871	0.6106	409 + 715	0.5856	0.6116	249

penalized regression for deriving polygenic scores based on very large individual-level genetic data. This also enables us to highlight further similarities and differences between implementations from snpnet and bigstatsr.

First, in their UK Biobank applications, Qian *et al.* (2020) have tried using elastic-net regularization (a combination of L1 and L2 penalties) instead of lasso (only L1), i.e. introducing a new hyper-parameter α ($0 < \alpha < 1$, with the special case of $\alpha = 1$ being the L1 regularization). They show that L1 regularization is very effective for very large sample sizes, and elastic-net regularization is not needed in this case, which we have also experienced. Yet, in smaller sample sizes and for very polygenic architectures, we showed through extensive simulations that using lower values for α can significantly improve predictive performance (Privé *et al.* 2019). In Qian *et al.* (2020), they tried $\alpha \in \{0.1, 0.5, 0.9, 1\}$; we recommend to use a grid on the log scale with smaller values (e.g. 1, 0.1, 0.01, 0.001, and even until 0.0001) for smaller sample sizes. Note that using a smaller α leads to a larger number of non-zero variables and therefore more time and memory required to fit the penalized regression. In functions `big_spLinReg` and `big_spLogReg` of R package bigstatsr, we allow to directly test many α values in parallel within the CMSA procedure. Therefore an optimal α value can be chosen automatically within the CMSA framework, without the need for more passes on the data.

Second, for large datasets, one should always use early-stopping. We have not found this to be emphasized enough in Qian *et al.* (2020). Indeed, while fitting the regularization path of decreasing λ values on the training set, one can monitor the predictive performance on the validation set, and stop early in the regularization path when the model starts to overfit (Figure 1). For large datasets, performance on the validation sets is usually very smooth and monotonic (before and after the minimum) along the regularization path, then one can safely stop very early, e.g. after the second iteration for which prediction becomes worse on the validation set. This corresponds to setting `n.abort=2` in bigstatsr and `stopping.lag=2` in snpnet. This is particularly useful because, when we move down the regularization path of λ values, more and more variables enter the model and the cyclical coordinate descent takes more and more time and memory. Therefore, the early-stopping criterion used in both bigstatsr and snpnet prevents from fitting very costly models, saving a lot of time and memory.

Third, Qian *et al.* (2020) recommend not to use scaled genotypes when applying lasso to genetic data.

However, using scaled genotypes is common practice in genetics, and is the assumption behind models in popular software such as GCTA and LDSC (Yang *et al.* 2011; Bulik-Sullivan *et al.* 2015). Scaling genotypes assumes that, on average, all variants explain the same amount of variance and that low-frequency variants have larger effects. Speed *et al.* (2012) argued that this assumption might not be reasonable and proposed another model: $\mathbb{E}[h_j^2] \propto [p_j(1-p_j)]^\nu$, where h_j^2 is the variance explained by variant j and p_j is its allele frequency. In Speed *et al.* (2017), they estimated ν to be between -0.25 and -0.5 for most traits. Note that scaling genotypes by dividing them by their standard deviations SD_j as done by default in bigstatsr assumes $\nu = -1$ while not using any scaling as argued by Qian *et al.* (2020) assumes $\nu = 0$. Therefore, using a trade-off between these two approaches can provide higher predictive performance and is therefore recommended (Zhang *et al.* 2020). In the case of L1 regularization, using a different scaling can be obtained by using different penalty factors λ_j in equation (1), which is an option available in both bigstatsr and snpnet. For example, using $\lambda_j = 1/\text{SD}_j$ allows to effectively use unscaled genotypes. Recently, we have implemented a new parameter `power_scale` to allow for different scalings when fitting the lasso in bigstatsr. Note that a vector of values to try can be provided, and the best-performing scaling is automatically chosen within the CMSA procedure.

Fourth, Qian *et al.* (2020) stated that bigstatsr "do not provide as much functionality as needed in [their] real-data application", mainly because bigstatsr requires converting the input data and cannot handle missing values. It is true that bigstatsr uses an intermediate format, which is a simple on-disk matrix format accessed via memory-mapping. However, package bigsnpr provides fast parallel functions `snp_readBed2` for converting from '.bed' files and `snp_readBGEN` for converting from imputed '.bgen' files, the two formats used by the UK Biobank. For example, it took 6 minutes only to read from the UK biobank '.bed' file used in this paper. We then used function `snp_fastImputeSimple` to impute by the variant means in 5 minutes only, which is also the imputation strategy used in snpnet. When reading imputed dosages instead, it takes less than one hour to access and convert 400K individuals over 1M variants using function `snp_readBGEN` with 15 cores, and less than three hours for 5M variants. When available, we recommend to directly read from '.bgen' files to get dosages from external reference imputation. As for package snpnet, it uses the PLINK 2.0 '.pgen' format, which is still under active development (in alpha testing, see `https://www.cog-genomics.org/plink/2.0/formats#pgen`). This format is not currently provided by the UK Biobank, and can therefore be considered as an intermediate format as well.

Data & Methods

As in Qian *et al.* (2020), we use the UK Biobank data (Bycroft *et al.* 2018), which is a large cohort of half a million individuals from the UK, for which we have access to both genotypes and multiple phenotypes (`https://www.ukbiobank.ac.uk/`). We apply some quality control filters to the genotyped data; we remove individuals with more than 10% missing values, variants with more than 1% missing values, variants having a minor allele frequency < 0.01, variants with P-value of the Hardy-Weinberg exact test < 10^{-50}, and non-autosomal variants. We restrict individuals to the ones used for computing the principal components in the UK Biobank (Field 22020); these individuals are unrelated and have passed some quality control (Bycroft

et al. 2018). We also restrict to the "White British" group defined by the UK Biobank (Field 22006) to get a set of genetically homogeneous individuals. These filters result in 337,475 individuals and 504,139 genotyped variants.

We use the same four phenotypes as used in Qian *et al.* (2020), namely height, body mass index (BMI), high cholesterol and asthma. We define height using field 50, BMI using field 21001, high cholesterol using field 20002 ("Non-cancer illness code, self-reported"). Asthma is defined using field 20002 as well as fields 40001, 40002, 41202 and 41204 (ICD10 codes); please see code for further details at `https://github.com/privefl/paper2-PRS/tree/master/response-snpnet/code`. For height and BMI, L1-penalized linear regressions are fitted using function `big_spLinReg` from bigstatsr and using parameter `family="gaussian"` in snpnet. For high cholesterol and asthma, L1-penalized logistic regressions are fitted using function `big_spLogReg` from bigstatsr and using parameter `family="binomial"` in snpnet. We use sex (Field 22001), age (Field 21022), and the first 16 principal components (Field 22009) as unpenalized covariates when fitting the lasso models.

References

Bulik-Sullivan, B. K., Loh, P.-R., Finucane, H. K., Ripke, S., Yang, J., Patterson, N., Daly, M. J., Price, A. L., Neale, B. M., of the Psychiatric Genomics Consortium, S. W. G., *et al.* (2015). LD score regression distinguishes confounding from polygenicity in genome-wide association studies. *Nature Genetics*, **47**(3), 291.

Bycroft, C., Freeman, C., Petkova, D., Band, G., Elliott, L. T., Sharp, K., Motyer, A., Vukcevic, D., Delaneau, O., O'Connell, J., *et al.* (2018). The UK biobank resource with deep phenotyping and genomic data. *Nature*, **562**(7726), 203–209.

Friedman, J., Hastie, T., and Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, **33**(1), 1.

Lello, L., Avery, S. G., Tellier, L., Vazquez, A. I., de los Campos, G., and Hsu, S. D. (2018). Accurate genomic prediction of human height. *Genetics*, **210**(2), 477–497.

Privé, F., Aschard, H., Ziyatdinov, A., and Blum, M. G. (2018). Efficient analysis of large-scale genome-wide data with two R packages: bigstatsr and bigsnpr. *Bioinformatics*, **34**(16), 2781–2787.

Privé, F., Aschard, H., and Blum, M. G. (2019). Efficient implementation of penalized regression for genetic risk prediction. *Genetics*, **212**(1), 65–74.

Qian, J., Tanigawa, Y., Du, W., Aguirre, M., Chang, C., Tibshirani, R., Rivas, M. A., and Hastie, T. (2020). A fast and scalable framework for large-scale and ultrahigh-dimensional sparse regression with application to the UK Biobank. *PLOS Genetics*, **16**(10), 1–31.

Speed, D., Hemani, G., Johnson, M. R., and Balding, D. J. (2012). Improved heritability estimation from genome-wide SNPs. *The American Journal of Human Genetics*, **91**(6), 1011–1021.

Speed, D., Cai, N., Johnson, M. R., Nejentsev, S., and Balding, D. J. (2017). Reevaluation of SNP heritability in complex human traits. *Nature Genetics*, **49**(7), 986–992.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, **58**(1), 267–288.

Tibshirani, R., Bien, J., Friedman, J., Hastie, T., Simon, N., Taylor, J., and Tibshirani, R. J. (2012). Strong rules for discarding predictors in lasso-type problems. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **74**(2), 245–266.

Yang, J., Lee, S. H., Goddard, M. E., and Visscher, P. M. (2011). GCTA: a tool for genome-wide complex trait analysis. *The American Journal of Human Genetics*, **88**(1), 76–82.

Zeng, Y. and Breheny, P. (2017). The biglasso package: A memory- and computation-efficient solver for lasso model fitting with big data in R. *arXiv preprint arXiv:1701.05936*.

Zhang, Q., Privé, F., Vilhjalmsson, B. J., and Speed, D. (2020). Improved genetic prediction of complex traits from individual-level data or summary statistics. *bioRxiv*.