synbreed: A Framework for the Analysis of Genomic Prediction Data Using R

Valentin Wimmer
Technische Universität München

Theresa Albrecht
Technische Universität München

Hans-Jürgen Auinger
Technische Universität München

Chris-Carolin Schön
Technische Universität München

Abstract

High-throughput genotyping and large scale phenotyping produces massive amounts of data. In this vignette, we present a novel R package named **synbreed** for the analysis of such data to derive genome-based predictions. It contains a collection of functions required to fit and validate genomic prediction models in plant and animal breeding. This covers data processing, data visualization and data analysis. Thereby a versatile analysis pipeline is established within one software package. All functions are embedded within the framework of a single, unified data object. The implementation is flexible with respect to a wide range of data formats and models. The package fills an existing gap in the availability of software for next-generation genetics research. Where necessary, the package provides gateways to other software programs to extend the field of applications. The utility of the package is demonstrated in this document using three large-scale example data sets provided by the `synbreedData` R package: a simulated data set representing a maize breeding program, a publicly available mice data set and a dairy cattle data set.

Keywords: genomic prediction, quantitative genetics, pedigree-based relatedness, marker-based relatedness, data processing, visualization.

1. Introduction

The analysis of quantitative traits is of paramount interest in agricultural genetics. For many traits such as yield, quality or resistance against diseases and environmental stress we observe continuously distributed phenotypes. According to quantitative genetic theory, these phenotypes are determined by the joint action of many genes, the so called quantitative trait loci (QTL), and the environment (**?**). To understand the inheritance of quantitative traits and to predict the unobservable genetic value of an individual are major challenges of agricultural genetics. Recently, high-throughput genotyping technology delivering tens or hundreds of thousands of single nucleotide polymorphism markers (SNPs) has become available for many crop and livestock species. The genomes of a large number of individuals can now be analyzed for their specific marker profile at high density, which allows estimating the proportion of genotype-sharing between them as well as efficient tagging of QTL in segregation analyses. In breeding, selection of the best genotypes can be conducted on high-density marker profiles once sufficiently accurate genome-based prediction models have been established. To achieve this, genomic prediction models are developed based on large training populations for which genotypic and phenotypic data are available. Once the best model is established, it can be used to predict the unobservable genetic value of selection candidates based on their marker profile.

Research on genomic prediction (GP) will be advanced through the availability of comprehensive, user-friendly software that covers a wide range of analysis steps. In a recent review, **?** state the urgent demand for such software to bring GP from theory to practice. To provide a framework for the analysis of GP data, we developed a novel add-on package named **synbreed** devised for the open-source software R (**?**). Only an open source software package is flexible enough to keep pace with advanced computational and methodological challenges. Our objectives in the design of the package were (i) to provide user-friendly algorithms for non-trivial methods required in the analysis of GP data, (ii) create an analysis framework using a single, unified data object resembling a generic data structure which is suitable for a wide range of statistical methods employing genotypic and phenotypic data such as GP, genome-wide association studies (GWAS) or QTL mapping, (iii) provide the methods within one open-source software package to avoid data conversion and transfer between software packages, (iv) to keep the implementation flexible with respect to the data structure for plant and animal genetics, and (v) to provide a gateway to other software and R packages to broaden the type of possible applications.

GP uses statistical models combining whole-genome data with phenotypic data. SNP effects are estimated from a regression of the phenotype on the marker profile. However, with a dense marker map, the model is over-parametrized. Typically, the number of SNPs p exceeds the number of observations n. A solution is the usage of mixed models (**?**). Within this framework, SNPs are used as direct predictors by modeling SNP effects or, alternatively, they are used to estimate a marker-based relationship matrix between individuals (**?**). The latter is used to model the variance-covariance structure for the genetic values. Recently, different models using Bayesian regression models have become popular (**??**). The predictive ability of a model for GP can be assessed using an out-of sample validation. If no independent test set is available, cross-validation (CV) is used to exploit the predictive ability of a model (**??**).

Several software programs for genetics research, covering parts of the required methods, have been released within the last years. The programs ASReml (**?**) and WOMBAT (**?**) provide restricted maximum likelihood (REML) estimation procedures for linear mixed models with arbitrary variance-covariance structure. The program PLINK implements algorithms for genome-wide association studies (GWAS) and identical-by-descent estimation. However, these programs are not stand-alone. Within R, different packages that tangent issues for GP are available: **qtl** for QTL analysis in experimental crosses (**?**), **GenABEL** for GWAS and effective SNP data storage and manipulation (**?**), **genetics** with classes and methods for handling genetic data (**?**) or **BLR** (**?**) for genome-based prediction models with Bayesian Ridge and Bayesian Lasso regression. However, there is no comprehensive program covering the specific needs of genetic researchers to analyze GP data.

In this article, we present how the **synbreed** package streamlines the analysis of GP data. The first part of the article summarizes the available data classes and functions. The second part shows by worked examples the application. The data management is guided by a single, unified data object. This forms the basis for all functions including the coding of the marker genotypes, algorithms to impute missing genotypes and linkage disequilibrium analysis. Moreover, we provide functions to estimate coefficients of relatedness for individuals based on both pedigree or marker data. We provide several possibilities to visualize the objects generated by the **synbreed** package. We give by simulated and real data examples for the application of these functions. Mixed Models and Bayesian Regression models are used to predict genetic effects. The predictive ability of the models is compared by CV. Both, model fit and validation using CV, can be performed using the data object including genotypes and phenotypes directly. Finally, we give the computational requirements for the analysis steps and present possible extensions of the package.

2. Statistical models

In this section, we present the statistical models used for the prediction of genetic values of individuals from a training set of individuals with phenotypes and pedigree or genotypes. We assume, that for each individual $i = 1, ..., n$ a single phenotypic record is available. Moreover, we consider a quantitative trait which can be modeled as being normal, i.e., $y_i = \mathrm{N}(\mu + g_i, \sigma^2)$. By μ, we denote the population mean and by σ^2 the residual variance. The unobservable genetic value g_i is predicted by statistical models using different data sources such as marker genotypes or pedigree.

In the mixed model "P-BLUP", the genetic values are predicted using the pedigree information to construct a variance-covariance structure for the individuals. Following **?**, this model is defined by

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{a} + \mathbf{e} \tag{1}$$

where $\mathbf{y}$ is the $n \times 1$ vector of phenotypic records, $\boldsymbol{\beta}$ is the vector of fixed effects and $\mathbf{a}$ is a $n \times 1$ vector of random effects. Observations are allocated to the fixed and random effects by the corresponding design matrices $\mathbf{X}$ and $\mathbf{Z}$. Fixed effects typically include the population mean and macro-environment effects such as location or year. Genetic values are sampled from a multivariate normal distribution

$$\mathbf{a} \sim \mathrm{N}(\mathbf{0}, \mathbf{A}\sigma_a^2)$$

where $\mathbf{A}$ is the additive numerator relationship matrix and σ_a^2 the additive genetic variance (**?**). The off-diagonal values of $\mathbf{A}$ are given by $2f_{i_1 i_2}$ for individuals i_1 and i_2 where – for a given pedigree – the *coefficient of coancastry* $f_{i_1 i_2}$ is computed by the expected probability that two alleles are identical by descent (**?**). The diagonal value for individual i_1 is $1 + F_{i_1}$ with F_{i_1} being the inbreeding coefficient. The $n \times 1$ vector $\mathbf{e}$ denotes the residuals with $\mathbf{e} \sim N(\mathbf{0}, \mathbf{I}_n\sigma^2)$ and $\mathbf{I}_n$ is the n-dimensional identity matrix. Best linear unbiased estimates (BLUE) for the fixed effects $\hat{\boldsymbol{\beta}}$ and predictions for the random effects (BLUP) $\hat{\mathbf{a}}$ are obtained by solving the mixed model equations (MME) (**?**)

$$\begin{bmatrix} \mathbf{X}^\top\mathbf{X} & \mathbf{X}^\top\mathbf{Z} \\ \mathbf{Z}^\top\mathbf{X} & \mathbf{Z}^\top\mathbf{Z} + \mathbf{A}^{-1}\frac{\sigma^2}{\sigma_\mathbf{a}^2} \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{\beta}} \\ \hat{\mathbf{a}} \end{bmatrix} = \begin{bmatrix} \mathbf{X}^\top\mathbf{y} \\ \mathbf{Z}^\top\mathbf{y} \end{bmatrix}$$

Estimates of the variance components $\hat{\sigma}_a^2$ and $\hat{\sigma}^2$ are obtained by REML estimation. A prediction for the genetic value of individual i in the training set is given by $\hat{a}_i$.

Genotypic data is incorporated in the mixed model "G-BLUP". Here, the relationship matrix based on pedigree is replaced by the genomic relationship matrix based on marker data. With the genomic relationship matrix, random deviations from the expected relationship caused by Mendelian sampling effects (**?**) can be quantified. The following equation for the genomic relationship matrix in a random mating population was proposed by **?**

$$\mathbf{U} = \frac{(\mathbf{W}-\mathbf{P})(\mathbf{W}-\mathbf{P})^\top}{2\sum_{j=1}^{p} p_j(1-p_j)} \tag{2}$$

where $\mathbf{W}$ is the marker matrix assigning p marker genotypes coded 0, 1 or 2 to n individuals. $\mathbf{P}$ is a $n \times p$ matrix with two times the minor allele frequency p_j for $j = 1, ..., p$ replicated n times within each column. The model "G-BLUP" is

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{u} + \mathbf{e} \tag{3}$$

with

$$\mathbf{u} \sim \mathrm{N}(\mathbf{0}, \mathbf{U}\sigma_u^2)$$

where σ_u^2 is the genetic variance pertaining to model "G-BLUP". The remaining parameters are defined as in model "P-BLUP". A prediction for the genetic value for individuals in the training set is given by $\hat{u}_i$.

In the random regression model "RR-BLUP", the phenotype is modeled as a function of the individual SNP effects

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{W}\mathbf{m} + \mathbf{e} \tag{4}$$

where $\mathbf{W}$ is the $n \times p$ marker matrix and $\mathbf{m}$ the p-dimensional vector of SNP effects. We assume that $\mathbf{m} \sim \mathrm{N}(\mathbf{0}, \mathbf{I}\sigma_{\mathbf{m}}^{\mathbf{2}})$ where σ_m^2 denotes the proportion of the genetic variance contributed by each individual SNP. A prediction for the genetic value for individuals in the training set is given by $\mathbf{w_i}^\top \hat{\mathbf{m}}$, where $\mathbf{w_i}$ is the p-dimensional vector of marker genotypes of individual i and $\hat{\mathbf{m}}$ the p-dimensional vector of estimated marker effects. Predicted genetic values and variance components from model "RR-BLUP" are predictable from model "G-BLUP" (**?**). It is computationally advantageous to use model "G-BLUP" when $n < p$ because computation times are of order $O(n)$ and $O(p)$, respectively.

The aforementioned models assume marker-homogeneous shrinkage of SNP effects. **?** suggested alternative models with marker-specific shrinkage. In this spirit, **?** used Bayesian Lasso to predict SNP effects. Their model, denoted by "BL", is given by

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{W}\mathbf{m} + \mathbf{e} \tag{5}$$

All elements but $\mathbf{m}$ are defined as in "RR-BLUP". The SNP effects $\mathbf{m}$ are modeled by marker-specific prior distributions

$$\mathbf{m} \sim \mathrm{N}(\mathbf{0}, \mathbf{T}\sigma^{\mathbf{2}})$$

with $\mathbf{T} = \mathrm{diag}(\tau_1^2, ..., \tau_j^2, ..., \tau_p^2)$ and the following model hierarchy

$$\begin{aligned}
\tau_j^2 &\sim \mathrm{Exp}(\lambda^2),\ j = 1, ..., p \\
\lambda^2 &\sim \mathrm{Ga}(\alpha, \beta) \\
e_i &\sim \mathrm{N}(0, \sigma^2),\ i = 1, ..., n \\
\sigma^2 &\sim \chi^{-2}(\nu, S^2)
\end{aligned}$$

For details of the hyperparameters see **?**. Parameter inference is performed within a Bayesian framework. The joint posterior distribution cannot be evaluated analytically in general. Hence

Markov chain Monte Carlo (MCMC) methods are used to generate samples from the full-conditional posterior distributions. Since all full-conditional distributions are well-known distributions, a Gibbs-Sampler can be utilized to generate a Markov chain (**?**). Prediction for the genetic value of individuals are obtained like in model "RR-BLUP" as $\mathbf{w_i}^\top \hat{\mathbf{m}}$.

The implementation of GP requires the prediction of the genetic performance $\mathbf{g}^* = (g_1, ..., g_{n^*})^\top$ of n^* unphenotyped individuals. For model "RR-BLUP" and "BL", the predicted genetic performance is given by $\hat{\mathbf{g}}^* = \mathbf{X}^* \hat{\boldsymbol{\beta}} + \mathbf{W}^{*\top} \hat{\mathbf{m}}$ where $\mathbf{X}^*$ denotes the design matrix for the fixed effects and $\mathbf{W}^*$ denotes the marker matrix for the unphenotyped individuals and $\hat{\mathbf{m}}$ the prediction of the SNP effects obtained from the training set. For "P-BLUP", the joint relationship matrix must be defined for all individuals in the training set and the prediction set. Predictions are obtained by solving the mixed model equation for the genetic values $\mathbf{a}^*$ of the unphenotyped individuals using the estimates of the variance-components of the training set

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + (\mathbf{Z}, \mathbf{Q_0}) \begin{pmatrix} \mathbf{a} \\ \mathbf{a}^* \end{pmatrix} + \mathbf{e} \quad \text{and} \quad \begin{pmatrix} \mathbf{a} \\ \mathbf{a}^* \end{pmatrix} \sim \mathrm{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \mathbf{A} & \mathbf{A}^* \\ \mathbf{A}^{*\top} & \mathbf{A}^{**} \end{bmatrix} \sigma^2_{\mathbf{a}} \right) \tag{6}$$

with $\mathbf{Q_0}$ being a $n \times n^*$ matrix with zeros and $\mathbf{A}$ the $n \times n$ additive numerator relationship matrix for the individuals in the training set, $\mathbf{A}^{**}$ the $n^* \times n^*$ additive numerator relationship matrix for the unphenotyped individuals and $\mathbf{A}^*$ the $n \times n^*$ additive numerator relationship matrix of the individuals in the training set with the unphenotyped individuals. Predictions for the genetic performance are obtained by $\hat{\mathbf{g}}^* = \mathbf{X}\hat{\boldsymbol{\beta}} + \mathbf{a}^*$. The same prediction scheme is employed for "G-BLUP". Here, the genomic relationship matrix substitutes the additive numerator relationship matrix in Equation 6 and σ^2_a is replaced by σ^2_u.

The out-of sample performance of a GP model determines the predictive ability. Cross-validation is an assumption-free method to investigate the predictive ability of different models (**?**). The data set is divided into k mutually exclusive subsets, $k-1$ of them form the estimation set (ES) for model training. The kth subset is used as independent test set (TS) for prediction. The predictive ability of a model is the correlation $r(\hat{\mathbf{g}}_{TS}, \mathbf{y}_{TS})$ of the vector of predicted genetic values $\hat{\mathbf{g}}_{TS}$ and the vector of observed phenotypes $\mathbf{y}_{TS}$ of the individuals in the TS. Typically individuals are assigned randomly to TS and ES. However, different sampling strategies can be employed to account for population stratification. **?** used within and across family-sampling for biparental families. The prediction bias can be assessed from a regression of the observed phenotype on the predicted genetic value (**?**). A regression coefficient of 1 indicates an unbiased prediction, a coefficient smaller than 1 implies inflation, a coefficient greater than 1 deflation of predicted genetic values compared to the observed phenotypes.

3. The class `gpData`

Data for GP consists of multiple data sources. To simplify the flow and use of data, we devised a data object named `gpData` ("**g**enomic **p**rediction **Data**"). Any object of class `gpData` includes all data required for the analysis. The first step in an analysis using the **synbreed** package is to create an object of class `gpData`. All additional functions utilize the predefined structure. Thus, it is sufficient to create once an object with the appropriate structure and use this for further analysis. The elements are phenotypes, genotypes and pedigree for a set of individuals. In addition, further elements for meta information on markers and individuals can be defined. The different elements are concatenated by common names for individuals and markers. They are used for data queries like in a data base. It is no prerequisite that all elements comprise the same subset of individuals or markers. Hence it is possible, e.g., to include individuals with genotypes but no phenotypes in an object of class `gpData`. This general structure is suitable for a wide range of data and models as employed by GP, GWAS or QTL studies.

The advantage of a single, unified data object are manifold. When data is shared, a single data file is easier to transfer than multiple files. By using basic summary methods, a first overview over all elements is forthcoming. Within the facilities of R, an object of class `gpData` is stored within the sparse binary format. For example, the claimed storage space for the mice data, described later in this article, was reduced from 95Mb in `ASCII` format to 8Mb in binary format. An object of class `gpData` can also be used as storage for multi-year experiments. New phenotypic data can be added over years or locations. In the following sections, the different elements of an object of class `gpData` and their required structure are described in more detail.

3.1. Phenotypic data

Phenotypic data is the outcome of performance tests for a set of individuals evaluating one or more traits. In plant breeding this can be the yield of a line evaluated in a field trial or in dairy cattle breeding the milk fat content measured in progeny testing. The phenotypic data are stored within the element `pheno`. This is an `array` with individuals organized in the first dimension and the traits organized in the second dimension. In case of repeated measurements for individuals, a third dimension can be used. For input, either a `data.frame` or an `array` can be used. In all cases, the phenotypic data will be converted to an `array`.

3.2. Genotypic data

The genotypic data is stored in the element `geno`. This is the marker matrix with individuals organized in rows and markers in columns. In the era of SNPs, functions for data processing are limited to biallelic markers. Each entry of the marker matrix depicts the observed genotype of an individual for a marker. Marker genotypes can either be distinguished by their names (e.g., `"AA"`, `"BB"` and `"AB"`) or by the observed alleles (e.g., `"A/C","A/T",...`).

3.3. Pedigree

Pedigree information for individuals is stored in the element `pedigree`. The pedigree is a table of individuals of the current generation and their ancestors. The pedigree is sorted by generation, beginning with the individuals with unknown parents (coded as `"0"`). The content is an object of class `pedigree` created by the function `create.pedigree`. An object of class `pedigree` is a `data.frame` with at least three/four variables, `ID` (required), `Par1` (required), `Par2` (required), and `gener` for names of individuals, Parent 1, i.e., sire, and Parent 2, i.e., dam, and generation, respectively. For animals, an optional variable `sex` can be added.

3.4. Covariate information

Additional information on individuals is stored in the element `covar`. This is a `data.frame` with a column `id` with the names of individuals that appear in `geno`, `pheno` or `pedigree`. Two automatically generated columns are added named `phenotyped` and `genotyped`. Both are logical and identify the individuals that are phenotyped (observations in `pheno`) or genotyped (observations in `geno`), respectively.

3.5. Marker map

The marker map is stored in element `map` of an object of class `gpData`. The map contains meta information about the genetic or physical positions of the marker on the genome. More precisely, the `map` is a `data.frame` with two columns named `"chr"` and `"pos"`. The first column identifies the chromosome (`character` or `numeric`) and the second is `numeric` indicating the position of the marker on the genome. This can be the genetic position within the chromosomes measured in centimorgan (cM) or the physical position relative to the reference genome in base pairs.

3.6. Info

Element `info` is used internally for additional information such as the map unit.

Beside class `gpData`, several further object classes are defined within the **synbreed** package. An overview over all available classes together with their elements as well as methods and functions are given in Figure 1. More detailed information on the functions is given in the next section.

4. Summary of functions

In this section, we present the main features of the **synbreed** package. Their application is demonstrated by examples in Section 5. The preliminary step is to read-in raw data files in the workspace, e.g., using function `read.table`.

4.1. Data processing

In all analyses using the **synbreed** package, the first step is to create an object of class `gpData`. The function `create.gpData` merges the different raw data sources. The return value is a list with elements `covar`, `pheno`, `geno`, `map`, `pedigree` and `info`. The function `create.gpData` performs consistency checks on the data and returns an object of class `gpData` with the data taken from the arguments. The basic call to create an object of class `gpData` is

```
R> gp <- create.gpData(covar,pheno,geno,map,pedigree)
```

An object of class `gpData` can be used as a data base with queries on individuals and markers. With function `discard.individuals`, a subset of individuals can be excluded from the object by removing the observations from `covar`, `pheno`, `geno` and `pedigree`. To add markers or individuals to an object of class `gpData`, the functions `add.markers` and `add.individuals` can be used. For subsequent analysis, other R packages often require a `data.frame` combining response variables, i.e., the trait and the marker genotypes. An object of class `gpData` can be converted to a `data.frame` using function `gpData2data.frame`. The function merges phenotypic and genotypic data. Multiple records for each individual result in additional rows. The `data.frame` can be extended by ungenotyped or unphenotyped individuals. This

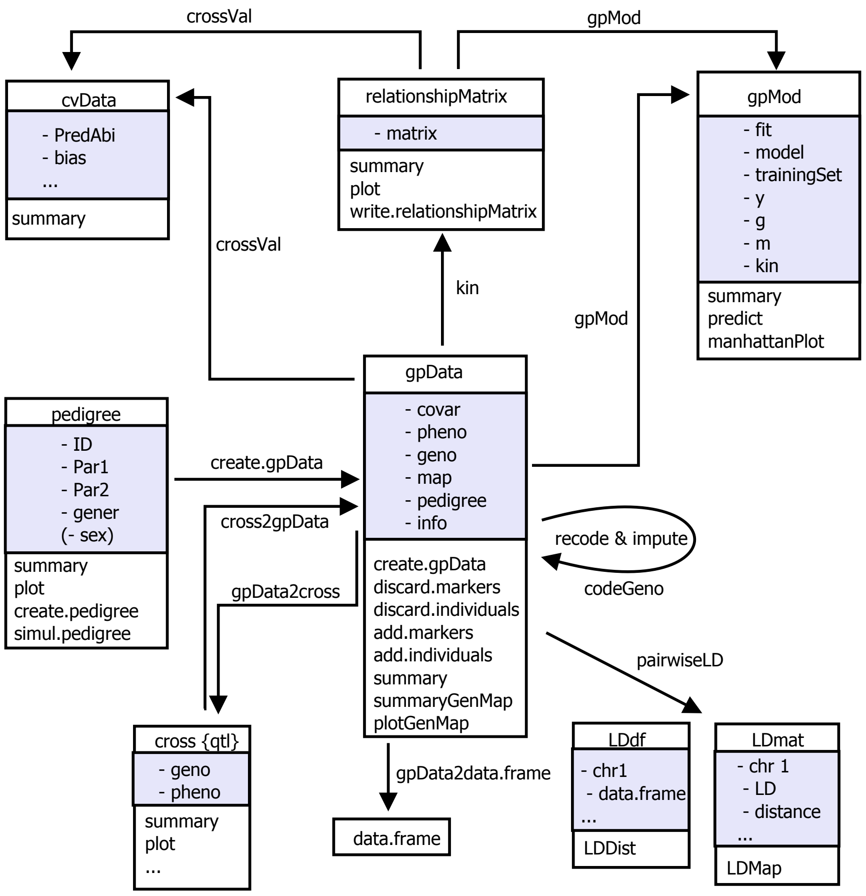


Figure 1: Overview of object classes, methods and functions within the **synbreed** package. Each box indicates a class together with the class name, the elements and the available functions and methods. The arrows indicate the data flow. The origin indicates the input argument and the head is the return value of the function.

format applies to a variety of different functions in R. Moreover, we included functions for the conversion from and to class `cross` in package **qtl** (?), see Figure 1.

The package **synbreed** provides several algorithms for the preprocessing of genotypic data. The algorithms are condensed in the function `codeGeno`. The function evaluates the following steps for the element `geno` within an object of class `gpData`.

1. Discard markers that exceed a given threshold for the fraction of missing values.

2. Recode marker genotypes from arbitrary coding into the number of copies of the minor allele, i.e., 0, 1 and 2 (see below).

3. Impute missing values according to different algorithms (see below).

4. Change the coding of minor and major allele whenever allele frequencies changed after step 3.

5. Discard markers with a minor allele frequency (MAF) below a given threshold.

6. Discard duplicated markers, retain the first copy.

Depending on choice of arguments, not all steps are performed. A report can be printed on the screen using argument `verbose=TRUE`. Below, we give more details on steps 2 and 3.

Step 2: Recode marker genotypes

Genotypic data in objects of class `gpData` can be raw marker data or output of other software, e.g., from Illumina `GenomeStudio` (`http://www.illumina.com/software/`). Typically raw genotypic data is either coded by marker genotypes, e.g., `AA` and `BB` for the homozygous genotypes and `AB` for heterozygous genotypes for a locus phase or as pair of observed nucleotides, e.g., `A/A`, `A/T`, `G/T`, ... The order of the alleles is not of interest in GP. Nevertheless a common coding is required. This is in the **synbreed** package the number of copies of the minor allele, i.e., 0, 1 and 2, for a single locus. Thus, homozygous marker genotypes are coded by 0 and 2 and heterozygous genotypes by 1. With this coding scheme the MAF for a marker j can be computed as

$$p_j = \frac{\sum\limits_{i=1}^{n} w_{ij}}{2n}, \; j = 1, ..., p$$

where w_{ij} is the marker genotype for individual i and marker j. The heterozygous genotype must be labeled unambiguously. This can either be done by a character which clearly defines the heterozygous state (e.g., "AB" for genotypes "AA", "AB" and "BB") or a function to identify them if multiple labels declare a heterozygous genotype (e.g., first allele $\neq$ second allele for `A/A`, `A/T`, `G/T`, ...). With this algorithm, it is straightforward to translate every coding scheme into the number of copies of the minor allele. Some examples will be presented in Section 5.

Step 3: Imputing of missing marker genotypes

Imputing of missing marker genotypes is often a necessary data preprocessing step. Indeed, many methods used in the analysis of GP data require a marker matrix without missing values. In function `codeGeno` missing values in the marker matrix can be replaced by one of the following algorithms (controlled by the argument `impute.type`):

1. Missing values are replaced using family information for fully homozygous individuals (**?**). This requires a biparental family structure with a uniform S_0 generation and a family size of at least 6 individuals. In the algorithm, a missing observation i for a marker j in family l is replaced according to the following rules: If marker j is monomorphic in family l, the imputed value will be the observed allele. The assumption in this case is, that both parents are homozygous for the same allele. If marker j is polymorphic in family l, the missing value is replaced from the following distribution $\mathsf{P}(x_j^{NA} = 0) = \mathsf{P}(x_j^{NA} = 2) = 0.5$.

2. The software Beagle (?) can be used to infer missing genotypes. Beagle uses a Hidden Markov Model to reconstruct missing genotypes based on flanking markers. Function `codeGeno` creates a directory `beagle` for Beagle input and output files, prepares input files and runs Beagle with default settings. The information on marker position is taken from element `map`. By default, three genotypes 0, 1, 2 are imputed using the allele with the highest posterior probability. This information is obtained from the Beagle output. In the special case of only homozygous genotypes, values are 0 and 2 according to their "dosage". The dosage is the estimated number of copies of the minor allele by the Beagle software (?).

3. A combination of the algorithms above can be used. In the first step, missing genotypes are imputed according to the family information. Here, the algorithm 1 is used for monomorphic markers and those with unknown position on the genome. In the second step, the remaining genotypes are imputed using Beagle as in algorithm 2.

4. Missing values are sampled from the marginal allele distribution for each marker. This algorithm does not take into account data stratification. Missing values are sampled from $\{0,1,2\}$ assuming a population in Hardy-Weinberg equilibrium (?). Thus $\mathsf{P}(x_j^{NA} = 0) = (1-p_j)^2$, $\mathsf{P}(x_j^{NA} = 1) = 2p_j(1-p_j)$ and $\mathsf{P}(x_j^{NA} = 2) = p_j^2$. In the special case of only homozygous genotypes, a missing value x_j^{NA} for marker j is replaced by a random draw from $\{0,2\}$ using the probabilities $\mathsf{P}(x_j^{NA} = 0) = 1 - p_j$ and $\mathsf{P}(x_j^{NA} = 2) = p_j$.

5. Replacing missing values by a given value.

We recommend the use of Beagle whenever a dense marker map is available and neighboring markers can be exploited for imputation. This software package is state-of-the-art in plant and animal breeding. However, if information from sufficiently sized families is available, the accuracy of imputing can be increased by the usage of algorithm 3. This algorithm is also suitable for genotypic data where the marker map is sparse.

4.2. Data analysis

Linkage disequilibrium

Linkage disequilibrium (LD) is defined as the non-random association of alleles at different loci. The extent of LD is determined by recombination, mutation, random drift and selection in population history. With genotypic data, LD can be calculated as the difference between the observed and expected (assuming random distributions) allele frequencies. There are many possibilities to compute LD from genotypic data, see ? using the **genetics** package. In the **synbreed** package, we use the measure r^2 (?)

$$r^2 = \frac{D_{vw}^2}{p_v(1-p_v)p_w(1-p_w)},$$

where $D_{vw} = p_{vw} - p_v p_w$ and p_{vw}, p_v and p_w are the frequencies of haplotype vw and allele v at one locus and allele w at the other locus. Pairwise LD between markers on the same chromosome can be computed using function `pairwiseLD`. For the general case, a gateway to the software PLINK is established to estimate the LD (argument `use.plink=TRUE`).

This requires that PLINK is available (e.g. in the working directory). When using PLINK, arguments `ld.threshold` and `ld.window` can be used the prune the output A fast within-R solution is available for marker data with only 2 marker genotypes. The function has two different return value types: `matrix` and `data.frame`. The first is a pairwise LD matrix and the latter a list with one row for each marker pair. In both cases, results are returned chromosome-wise. Moreover, the Euclidean distance of the markers based on the `map` is computed.

Estimation of relatedness

The **synbreed** package provides a unified function `kin` to compute pedigree-based (expected) and marker-based (realized) coefficients of relatedness. As shown in Figure 1, coefficients are estimated for a set of individuals within an object of class `gpData`. The return value is an object of class `relationshipMatrix`. This is a symmetric matrix with pairwise coefficients of relatedness for a set of individuals.

The computation of the pedigree-based relationship matrix in **synbreed** starts with the gametic relationship matrix. This approach requires only few assumptions and is very flexible with respect to special cases, i.e., homozygous inbred lines. Moreover, this computation can be extended to include two populations and dominance effects. The gametic relationship matrix is of dimension $2n \times 2n$ with n being the number of individuals in the pedigree. The matrix contains the probability that two gametes are identical by descent (IBD), denoted by the equivalent symbol $\equiv$. The gametes of an individual i_1 are denoted by X_1 and X_2, the gametes of an individual i_2 are denoted by Y_1 and Y_2. All diagonal values equal 1. The secondary diagonals contain the inbreeding coefficients $F_{i_1} = \mathsf{P}(X_2 \equiv X_1)$. Both additive and dominance relationships may be obtained from the gametic relationship matrix according to the rules in **?**.

Pedigree-based coefficients of relatedness include additive (argument `ret="add"`), dominance (argument `ret="dom"`) and kinship (argument `ret="kin"`). The additive relationship coefficient between two individuals is obtained by the sum of the four corresponding entries of the allele combinations in the gametic relationship matrix divided by 2. The additive relatedness between individuals i_1 and i_2 is given by (**?**)

$$2f_{i_1i_2} = \frac{1}{2}\left[\mathsf{P}(X_1 \equiv Y_1) + \mathsf{P}(X_1 \equiv Y_2) + \mathsf{P}(X_2 \equiv Y_1) + \mathsf{P}(X_2 \equiv Y_2)\right].$$

The numerator relationship matrix described in Section 2 is constructed using all pairwise coefficients $2f_{i_1i_2}$ on the off-diagonal and $1 + F_{i_1}$ on the diagonal. The kinship between i_1 and i_2 is defined as $f_{i_1i_2}$.

For non-inbred individuals, the dominance coefficients are

$$t_{i_1i_2} = \mathsf{P}(X_1 \equiv Y_1) \cdot \mathsf{P}(X_2 \equiv Y_2) + \mathsf{P}(X_1 \equiv Y_2) \cdot \mathsf{P}(X_2 \equiv Y_1).$$

If genotypic data is available, a genomic relationship can be constructed based on marker data. The matrix of genomic relationship coefficients is computed using function `kin` with argument `ret="realized"`. This requires an object of class `gpData` where the data in `geno` is coded by the number of copies of the minor allele, i.e., by using function `codeGeno`. The genomic relationship matrix is computed according to Equation 2. Negative values indicate pairs of individuals sharing fewer alleles than expected based on the allele frequencies and positive

Model	Argument			Variance components [1]	
	`kin`	`markerEffects`	`model`	`kinTS`	`In/VarE`
RR-BLUP	`NULL`	`TRUE`	`BLUP`	σ^2_m	σ^2
G-BLUP	marker-based	`FALSE`	`BLUP`	σ^2_u	σ^2
P-BLUP	pedigree-based	`FALSE`	`BLUP`	σ^2_a	σ^2
BL	`NULL`	`FALSE`	`BL`		σ^2

Table 1: Overview over the possible models which can be fitted using `gpMod` depending on the choice of arguments `kin`. [1]: from the output using `summary.gpMod`, `kinTS`= Genetic variance component modeled with kinship/relationship matrix of all individuals in the training set (TS), `In`= Residual variance component modeled with n-dimension identity matrix **I**.

values indicate pairs of individuals sharing more alleles than expected from the marginal allele distributions. The denominator in Equation 2 is the correction term for a random mating population (**?**). Corrections for other populations may be obtained by multiplying the resulting matrix by a constant. See **?** for an example with homozygous inbred lines.

The genomic relationship can also be computed by the simple matching coefficient (**?**). This is implemented with the argument `ret="sm"`, but only for homozygous inbred lines. To account for IBS but not IBD, **?** proposed the correction $(\mathtt{s} - s_{min})/(1 - s_{min})$ where `s` is the matrix of all pairwise simple matching coefficients multiplied by 2 and s_{min} the minimum of all $\frac{n}{2}(n-1)$ values in `s` (argument `ret="sm-smin"` within function `kin`).

Any object of class `relationshipMatrix` could easily be passed for further analysis to other software packages. The function `write.relationshipMatrix` creates a table which could be stored in an external file. Every row in the table depicts one element of the lower triangle. The table is ordered by column names within row names which is the required format for ASReml or by row names within column names which is the required format for WOMBAT.

Statistical models

The function `gpMod` provides a general interface to fit all models presented in Section 2. The input argument is an object of class `gpData`. The training set for the model are all individuals with phenotypes and genotypes (or pedigree for model “P-BLUP”). The mixed models “P-BLUP” and “G-BLUP” are fitted using the REML algorithm in function `regress` implemented in the package **regress** (**?**). Model “RR-BLUP” is derived from model “G-BLUP” using the functional relationship between the models. In this case, in the first step “G-BLUP” is fitted and then the mixed model equations are solved for the SNP effects. The Bayesian regression models are fitted using the Gibbs-Sampler implemented in the **BLR** package (**?**). The variance-covariance structure in the BLUP models “P-BLUP” and “G-BLUP” is defined within the argument `kin` by an object of class `relationshipMatrix`, see Figure 1. An overview over the possible arguments and the resulting models is given in Table 1. The data for model training consists of all individuals with phenotypes and genotypes. All data are restricted to individuals from the training set. We include the `"..."` argument to allow a subtle adjustment of the model control. This includes the choice of hyper-parameters and MCMC options for “BL”.

For all models, the function `gpMod` reports the names of all individuals in the training set together with their predicted genetic performance and phenotype. For the models "RR-BLUP" and "BL" additional predicted SNP effects are returned. Moreover, the model output of the used function, i.e., `regress` or `BLR` is returned. A `summary` method is available for objects of class `gpMod`. It summarizes the number of observations in the training set and a summary for the predicted genetic performance of the individuals in the training set. Furthermore, we provide a `predict` method for the class `gpMod`. If all individuals are phenotyped, it returns the predicted genetic performance of the individuals in the training set. To predict the genetic performance of unphenotyped individuals with the models "RR-BLUP" or "BL", an object of class `gpData` containing the genotypic data must be passed to the `predict` method. To predict unphenotyped individuals with "P-BLUP" or "G-BLUP", their coefficients of relatedness must be included in the variance-covariance structure. Alternatively, the argumetn `predict=TRUE` in `gpMod` may be used to predict genotyped but not phenotyped individuals based on marker data.

Cross-validation

CV for different GP models is implemented in function `crossVal`. Possible models include mixed models and Bayesian regression models. To account for population stratification, different sampling strategies may be employed. Thereby in most cases the population structure will represent families or the age-group in animal breeding. The sampling strategies are

random Random sampling of the complete data set.

within popStruc Each group is splitted into k subsets to investigate within group predictive ability.

across popStruc The ES and TS contains only complete groups for across group predictive ability.

The function `crossVal` returns an object of class `cvData`. This is a `list` of predicted genetic performance and observed phenotypic values for all $k \cdot r$ test sets. Moreover, the list includes individuals, size, predictive abilities and bias for each corresponding TS.

4.3. Data summary and visualization

In this section we present the different possibilities to visualize objects in the **synbreed** package and provide graphical or textual summaries. Thus key features of the high-dimensional objects can be assessed. For objects of class `gpData`, the `summary` method reveals general information about all elements. This includes the number of genotyped and phenotyped individuals, summary statistics (0, 25, 50, 75 and 100% quantiles and mean) for the traits, number of markers, marker genotype frequencies and the summary for the pedigree. The `summary` method for an object of class `pedigree` includes number of individuals, parents and generations. The `summary` method for objects of class `relationshipMatrix` indicates the dimension, rank, range and number of unique values. The `summary` method for objects of class `cvData` provides information on the employed sampling scheme. Results are summarized by the minimum, mean and maximum of the predictive ability and bias across all test sets together with an estimated standard error for the mean pooled over the k subsets.

The LD is visualized chromosome-wise. A LD heatmap is available using function `LDMap`. This function is a wrapper to apply the function `LDheatmap` of the package **LDheatmap** (**?**)

for an object of class `gpData`. The LD must be organized in the `matrix`-format of function `pairwiseLD`. The function `LDDist` visualizes LD decay as scatterplot or stacked histogram. This requires the LD organized in the `data.frame` format. In the stacked histogram, the LD is visualized with bars representing fractions of SNP pairs. Breaks for the distance and LD level are controlled by the user. Moreover, a nonlinear regression curve according to **?** can be added to the scatterplot.

A `plot` method is available for objects of class `pedigree` and `relationshipMatrix`. The pedigree structure is visualized by a directed graph. One line is representing a generation with a vertex for each individual. Objects of class `relationshipMatrix` are visualized using a heatmap representation. Estimated SNP effects of an object of class `gpMod` can be visualized using a "Manhattan" plot. This is implemented in function `manhattanPlot`.

Genome-wide dense marker maps require new types of visualization. The function `plotGenMap` provides layouts for small ($<$ 200 markers/chromosome) and high numbers of markers. In both cases, each chromosome is represented as a vertical bar. The length of the bar is given by the distance between the first and the last marker on the chromosome. For a sparse map, each marker is plotted as a horizontal bar. With a dense map, only the marker density is evaluated within equally-spaced intervals. The bandwidth for the intervals is controlled by the user. Within each interval, the number of markers is counted and visualized by a color image. In addition, function `summaryGenMap` returns a summary of the marker distances. This includes the number of markers, minimum, mean and maximum distances between markers by chromosome and over all chromosomes.

5. Examples

5.1. Data sets

In this section, the application of the **synbreed** package is demonstrated using the data from the **synbreedData** package. The **synbreedData** is included as a dependency in the `synbreed` package.

Simulated maize data

This simulated data resembles the output from a typical maize breeding program based on doubled haploid (DH) lines. A DH is produced by sampling gametes from heterozygous S_0 plants. Thus, a DH line received both gametes from a single parent and is fully homozygous.

Ten chromosomes of length 160 cM are simulated. True genetic values are sampled from $K = 500$ segregating, biallelic QTL with equal additive effects assuming absence of dominance and epistasis

$$g_i = \sum_{k=1}^{K} QTL_{ik},\ i = 1, ..., n$$

where QTL_{ik} is the effect of the k-th QTL allele for individual i. Phenotypic values are simulated by adding a random environmental residual. Thus the underlying genetic model is

$$y_i = g_i + e_i,\ i = 1, ..., n$$

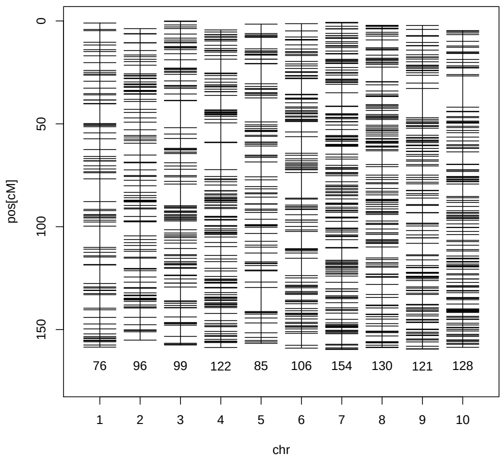
where $e_i = N(0, \sigma^2)$. Genotypic and phenotypic data was simulated for 1250 DH lines. Moreover, pedigree for additional 360 ancestor individuals is available. These ancestors can be heterozygous. The 1250 DH lines belong to 25 biparental families with 50 lines in each family. The `maize` data can be loaded within the **synbreed** package using

```
R> library(synbreed)
R> data(maize)
```

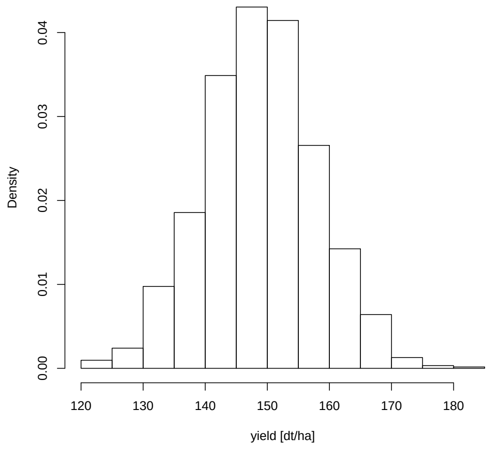
A basic overview over the contents of the `gpData` is available through

```
R> summary(maize)

object of class 'gpData'
covar
         No. of individuals 1610
                 phenotyped 1250
                  genotyped 1250
pheno
         No. of traits:                   1

     Trait
 Min.   :120.7
 1st Qu.:142.8
 Median :148.9
 Mean   :148.9
 3rd Qu.:154.9
 Max.   :181.8

geno
         No. of markers 1117
         genotypes 0 1
         frequencies 0.339995 0.660005
         NA's 0.000 %
map
         No. of mapped markers  1117
         No. of chromosomes     10

         markers per chromosome

  1   2   3   4   5   6   7   8   9  10
 76  96  99 122  85 106 154 130 121 128

pedigree
Number of
         individuals  1610
         Par 1        219
         Par 2        221
         generations  15
```

Genotypic data in element `geno` comprises 1117 polymorphic SNP markers. The distribution of markers across the chromosomes is visualized in Figure 2(a) using function `plotGenMap(maize)`. Phenotypic data was simulated for the quantitative trait yield [dt/ha], evaluated in testcrosses of DH lines in 3 locations. Figure 2(b) shows a histogram of average testcross performance of the 1250 DH lines.



(a) Marker map



(b) Phenotypic trait values

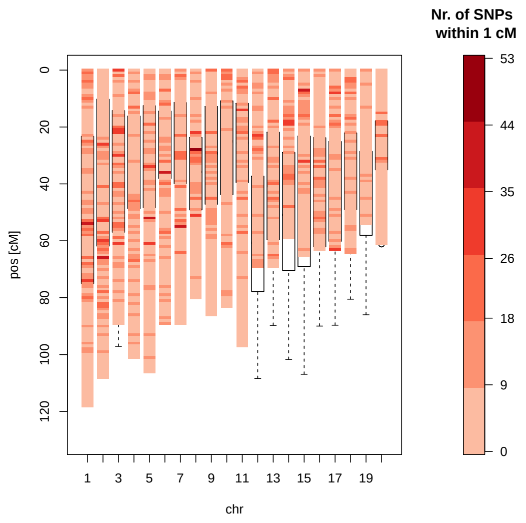
Figure 2: Simulated `maize` data with 1250 DH lines and 1117 polymorphic SNP markers. (a) Marker map with marker positions in cM and number of markers below chromosome bars. The plot is created using `plotGenMap`. (b) Histogram of phenotypic trait values for testcrosses of DH lines.

Mice data

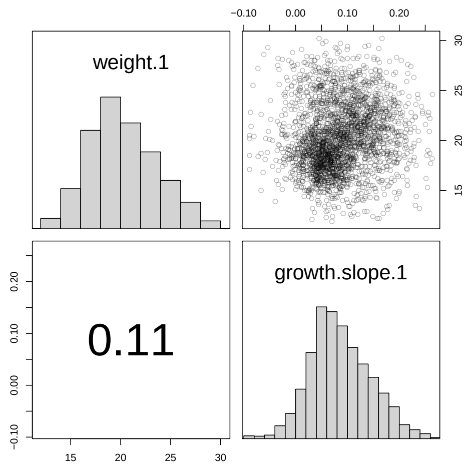
The mice data set was recently used to illustrate prospects of GP (**??**). The data is publicly available from http://gscan.well.ox.ac.uk and comprises data of 2527 mice. They are progenies from eight inbred strains followed by 50 generations of pseudorandom mating. Original data comprises several qualitative and quantitative traits. See the website or **?** for more details on this data set. The data can be loaded using

```
R> data("mice")
```

Genotypic data in element `geno` consists of 12545 SNP markers. Marker data is available for a subset of 1940 individuals. The marker map in element `map` is the sex-averaged genetic map with distances given in cM. See Figure 3(a) for a visualization of the marker map using `plotGenMap(mice,TRUE,FALSE,ylab="pos [cM]")`. Element `pheno` comprises two quantitative traits for 2527 mice: the body weight at 6 weeks age [g] and growth slope between 6 and 10 weeks age [g/day]. The distribution of both traits is shown in Figure 3(b). Pedigree information is not yet available from the website. The element `covar` includes the variables sex (females=0, males=1), month of birth (1-12), birth year, coat color, cage density and litter.



(a) Marker map



(b) Phenotypic trait values

Figure 3: Publicly available `mice` data. (a) Visualization of the marker map comprising in total 12545 SNP markers along 19 autosomes and the X-chromosome using `plotGenMap`. (b) Histograms for univariate distribution of traits weight and growth slope for 2527 mice and scatterplots for the bivariate distribution and the correlation coefficient.

Dairy cattle data

This data set contains genotypic, phenotypic, map and pedigree data of 500 bulls. Genotypic and pedigree data are taken from real cattle data while phenotypes are simulated. The `cattle` data was provided by the Animal Breeding and Genetics group of Henner Simianer, Georg-August-Universität Göttingen. The data can be loaded using

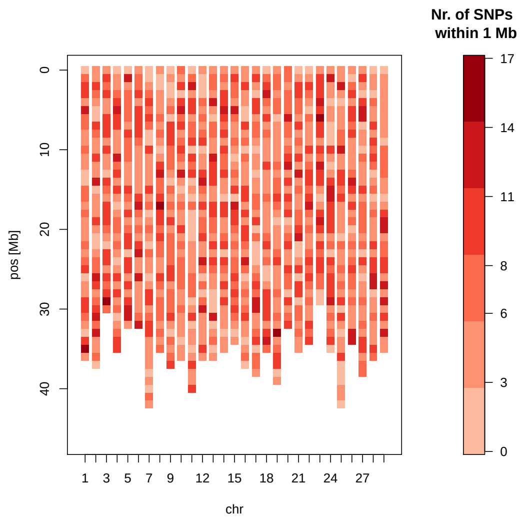
```
R> data("cattle")
```

Two quantitative traits are available with simulated heritabilities of 0.41 and 0.66, respectively. The distribution of both traits is shown in Figure 4(b). Genotypic data consists of 7250 biallelic SNP markers for every phenotyped individual with missing data included. SNPs are mapped across all 29 autosomes. Distances in the SNP map are given in mega bases (Mb). See Figure 4(a) for a visualization of the marker map using function `plotGenMap`. The pedigree information is available at least on parents and grandparents of the phenotyped individuals.

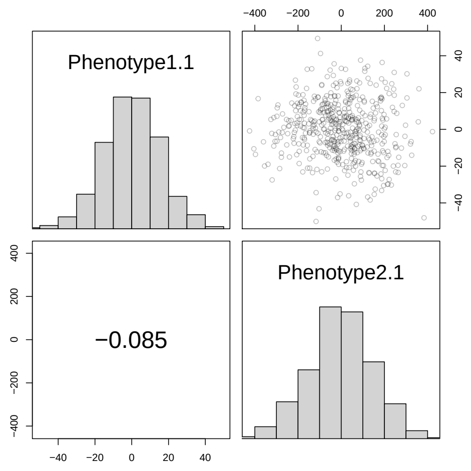
5.2. Data processing and visualization

All data sets are already given as objects of class `gpData`. Hence the next step is the processing of the genotypic data in element `geno`. The `maize` data includes no missing values. Thus data processing only involves the recoding of the alleles into the number of copies of the minor allele. There are only homozygous genotypes. Hence no heterozygous genotype must be declared and genotypic data is recoded by

```
R> maizeC <- codeGeno(maize)
```



(a) Marker map



(b) Phenotypic trait values

Figure 4: Description of the `cattle` data. (a) Visualization of the marker map comprising in total 7250 SNP markers along 29 autosomes using `plotGenMap`. (b) Histograms for univariate distribution of traits 1 and 2 for 500 bulls and scatterplots for the bivariate distribution and the correlation coefficient.

Resulting object `maizeC` is again of class `gpData` but with recoded genotypic data. This information is stored for further analysis as `codeGeno` sets `maizeC$info$codeGeno=TRUE`. When there are heterozygous genotypes, the argument `label.heter` is required. This is shown for the `mice` data, where genotypes are coded by the observed alleles separated by a slash symbol. A heterozygous genotype is identified whenever the first allele differs from the second allele. This can be generalized by the function

```
R> is.heter <- function(x) {
+       substr(x, 1, 1) != substr(x, 3, 3)
+  }
```

The function `is.heter` checks for unequal alleles at a locus to identify heterozygous genotypes. It is passed to the function `codeGeno` by the argument `label.heter`.

```
R> miceC <- codeGeno(mice, label.heter = is.heter)
```

The function `codeGeno` enables a preselection of markers and the imputing of missing values. For the following analysis, we choose a threshold for the MAF of 0.05 and of 0.01 for the fraction of missing values. All SNPs failing these criteria are discarded from the object of class `gpData`. Missing values are sampled from the marginal marker allele distribution. The corresponding call and verbose output is

```
R> miceC <- codeGeno(mice, label.heter = is.heter, impute = TRUE,
+       impute.type = "random", maf = 0.05, nmiss = 0.01, verbose = TRUE)
```

```
step 1  : 400 marker(s) removed with > 1 % missing values
step 2  : Recoding alleles
step 2.1: No duplicated markers discarded
step 3d : Random imputing of missing values
         approximate run time  17.82  seconds
step 4  : No recoding of alleles necessary after imputation
step 5  : 2148 marker(s) removed with maf < 0.05
step 6  : No duplicated markers discarded
End      : 9997 marker(s) remain after the check

Summary of imputation
  total number of missing values                : 23527
  number of random imputations                  : 23527
```

Recoding of the `cattle` data can be performed according to the `mice` data using argument `label.heter="AB"`.

In the next step, the pairwise LD for markers on chromosome 1 of `maize` is computed using the within-R solution for homozygous genotypes. We choose `type="data.frame"` and visualize the LD with the function `LDDist`.

```
R> maizeLD <- pairwiseLD(maizeC,chr=1,type="data.frame")
```

The LD decay for the LD data in the object `maizeLD` is shown in Figure 5. This plot can be generated using

```
R> LDDist(maizeLD,type="p",xlab="dist [cM]",pch=19,col=hsv(alpha=0.075,v=0))
```

for the scatterplot and

```
R> LDDist(maizeLD,type="bars",breaks=list(dist=c(0,25,50,75,200),
+  r2=c(1,0.5,0.3,0.2,0.1,0.05,0)),xlab="dist [cM]")
```

for the stacked histograms with user-specified breaks. The graphical layout is controlled by specifying additional graphical parameters. The LD for the recoded mice data may be obtained using the PLINK software invoked by the argument `use.plink=TRUE`.

5.3. Data analysis

In the following, different coefficients of relatedness are calculated for the individuals. Pedigree-based coefficients are only inferred for the `maize` data. All genotyped individuals are DH lines having an inbreeding coefficient of 1. When using function `kin`, the special argument `DH` identifies DH lines in the data. For a DH line i_1, $F_{i_1} = \mathsf{P}(X_1 \equiv X_2)$ is set to 1 by the algorithm. In the `maize` data a variable `DH` in element `covar` indicates DH lines. The additive relationship matrix for all individuals in the pedigree is computed by

```
R> A <- kin(maize,ret="add",DH=maize$covar$DH)
```

The additive relationship matrix for the general case is obtained by omitting the argument `DH`. A summary of the relationship matrix is obtained by

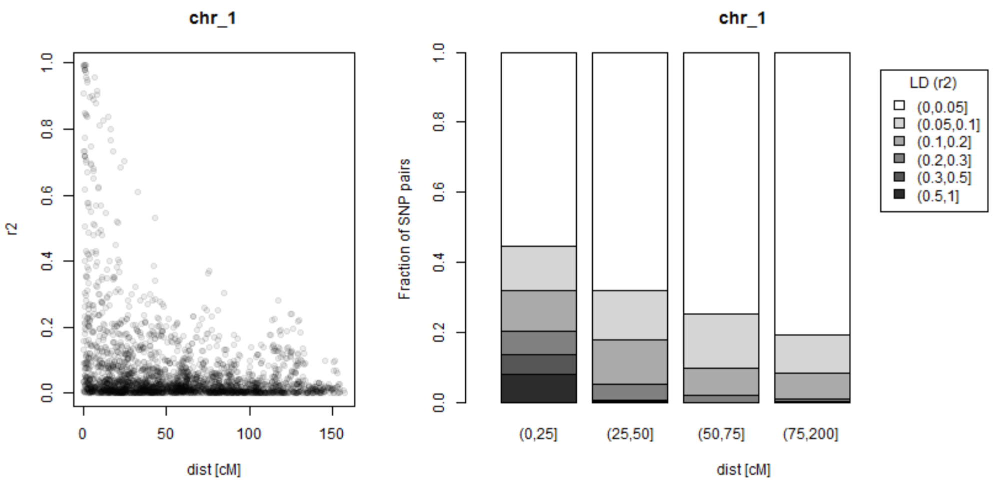


Figure 5: LD decay as a scatter plot (left side) and stacked histogram (right side) for 76 markers on the first chromosome for the `maize` data.

```
R> summary(A)

 dimension                     1610 x 1610
 rank                          1460
 range of off-diagonal values 0 -- 1.757812
 mean off-diagonal values      0.6907328
 range of diagonal values      1 -- 2
 mean diagonal values          1.891466
 number of unique values       1435
```

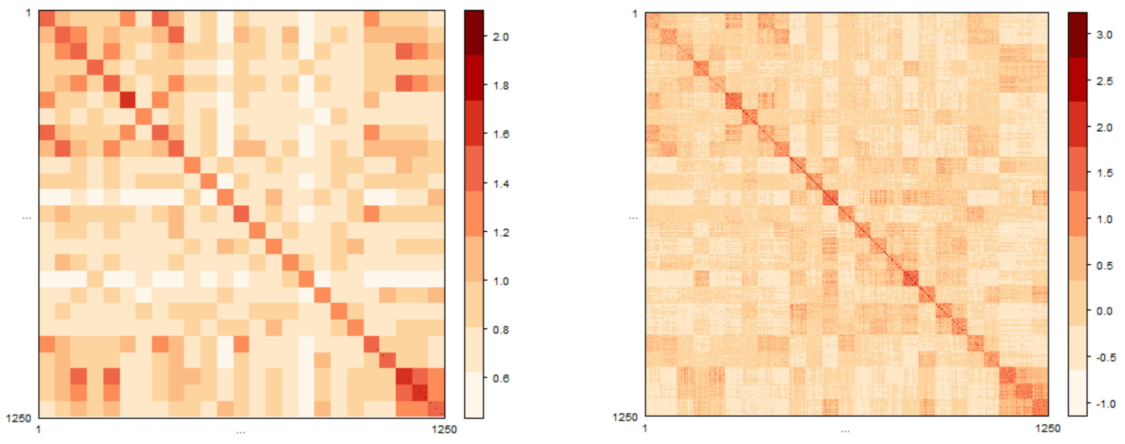
Genomic relationship coefficients can be computed using the recoded marker matrix.

```
R> U <- kin(maizeC,ret="realized")
R> summary(U)

 dimension                     1250 x 1250
 rank                          1108
 range of off-diagonal values -0.8803719 -- 2.10971
 mean off-diagonal values      -0.001601281
 range of diagonal values      1.467 -- 2.964
 mean diagonal values          2
 number of unique values       685017
```

Resulting matrix `U` is the genome-based analogon for `A`. The difference in expected and realized relationship matrices can be visualized using the heatmap visualization, see Figure 6. Note that only the 1250 genotyped individuals in `A` instead of all 1610 individuals are presented. Hence expected and realized relationship can be compared directly. The individuals were selected using the column `genotyped` in element `covar` to query only rows and columns in `A` for genotyped individuals.

```
R> plot(A[maize$covar$genotyped,maize$covar$genotyped])
R> plot(U)
```

The DH lines in the `maize` data are sorted by their family number. Thus the heatmap in Figure 6(a) is structured as 25×25 squares. Pedigree-based coefficients indicate no difference within families but differ across families. In contrast, the marker-based relationship reveals random Mendelian sampling effects within families. Thus, the heatmap provides a higher resolution for the coefficients.



(a) Pedigree-based relationship (b) Marker-based relationship

Figure 6: Comparison of pedigree-based (expected) relationship matrix and marker-based (realized) relationship for the 1250 genotyped individuals in `maize` data. Visualized are the heatmaps of pairwise relationship coefficients.

5.4. Statistical models

In this section, we fit statistical models to predict genetic values for the individuals in the `mice` and `maize` data. Model performance is assessed by cross-validation.

For the `mice` data, model "G-BLUP" is fitted for the trait `weight`. The model includes a random genetic effect and the population mean μ as fixed effect. The covariance structure for the genetic values is given by the realized relationship matrix. This is obtained by

```
R> UM <- kin(miceC,ret="realized")
```

The model is fitted using the function `gpMod`

```
R> miceGBLUP <- gpMod(miceC,model="BLUP",kin=UM,trait="weight")
```

The resulting object `miceGBLUP` is of class `gpMod`. A summary of the model fit is obtained by

```
R> summary(miceGBLUP)
```

```
Object of class 'gpMod'
Model used: BLUP
Nr. observations 1928
Genetic performances:
  Min.    1st Qu. Median  Mean    3rd Qu. Max
-4.72400 -1.01600 -0.09459 -0.01184 0.96120 5.17200
--
Model fit
Likelihood kernel: K = (Intercept)

Maximized log likelihood with kernel K is  -3178.055

Linear Coefficients:
             Estimate Std. Error
 (Intercept)   20.344      0.065

Variance Coefficients:
             Estimate Std. Error
       kinTS    3.704      0.498
       In       8.069      0.311
```

Estimates for the SNP effect may be obtained using "RR-BLUP" through

```
R> miceRRBLUP <- gpMod(miceC,model="BLUP",kin=UM,trait="weight",markerEffects=TRUE)
```

Note that in this case the variance components pertaining to model (4), i.e. σ_m^2 rather than σ_u^2 is reported. The model "RR-BLUP" uses marker-homogenous shrinkage. As a contrasting model, "BL" is employed which uses marker-specific shrinkage. Values for the prior distribution of σ^2 and λ are determined using the equations of **?**.

```
R> prior <- list(varE = list(df = 3, S = 40), lambda = list(shape = 0.8,
+      rate = 1e-04, value = 52, type = "random"))

R> miceModBL  <- gpMod(miceC,model="BL",trait="weight",
+  prior=prior,nIter=12000,burnIn=2000,thin=10)
```

A summary of the model fit may be obtained using

```
R> summary(miceModBL)
```

```
Object of class 'gpMod'
Model used: BL
Nr. observations 2511
Genetic performances:
  Min.    1st Qu. Median  Mean    3rd Qu. Max
  15.61   19.33   20.25   20.33   21.31   25.54
--
```

```
Model fit
MCMC options: nIter = 12000, burnIn = 2000, thin = 10
              Posterior mean
(Intercept)   18.91775
VarE          8.116326
lambda        128.6805
```

The summary reveals only marginal differences compared to the mixed model "G-BLUP". A "Manhattan" plot of the predicted marker effects is shown in Figure 7.

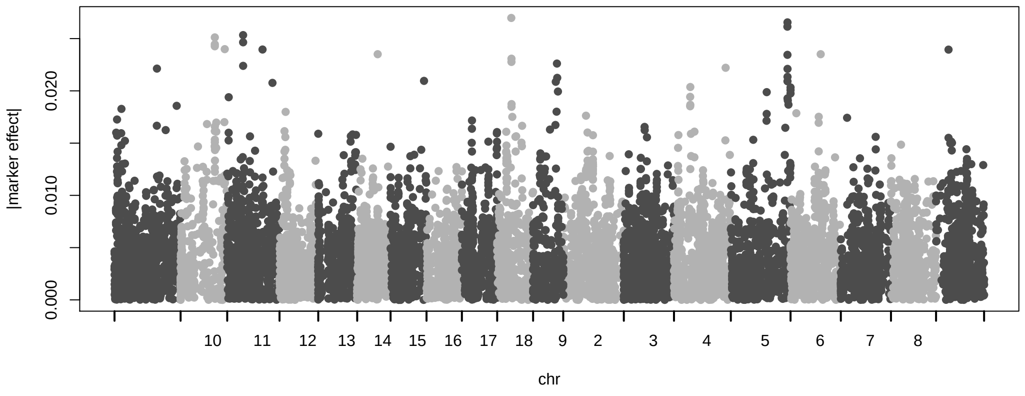


Figure 7: Predicted SNP effects $|\hat{\mathbf{m}}|$ for the trait weight in mice data using "BL".

The estimated SNP effects of "BL" are used to predict the genetic performance of the 12 individuals with a missing value for weight. First, a new object of class gpData is created including the marker genotypes of the individuals of the prediction set.

```
R> unphenotyped <- dimnames(mice$pheno)[[1]][is.na(mice$pheno[,1,])]
R> phenotyped <- mice$covar$id[!mice$covar$id %in% unphenotyped]
R> predSet <- discard.individuals(miceC,phenotyped)
```

Predictions for the genetic performance are obtained by

```
R> predict(miceModBL,predSet)
```

The predictive performance of the mixed model is judged by CV. Here, we use 2-fold CV as in ?. The splitting into TS and ES is repeated 10 times. Individuals are assigned randomly to TS and ES. For computational ease, the variance-components are estimated once for the complete data set and committed to model training in CV. This CV scheme may be employed using

```
R> cv.mice <- crossVal(miceC,cov.matrix=list(UM),k=2,Rep=10,Seed=123,
+                 sampling="random",varComp=miceGBLUP$fit$sigma,VC.est="commit")
```

A summary of the CV is obtained by

```
Object of class 'cvData'

 2 -fold cross validation with 10 replication(s)
     Sampling:                 random
     Variance components:      committed
     Number of random effects: 1
     Size of the TS:           964 -- 964

Results:
                      Min           Mean +- pooled SE           Max
 Predictive ability:  0.2916            0.3697 +- 0.009471            0.4323
 Bias (reg. slope)    0.6830            0.9927 +- 0.02621            1.2195

Seed start:  123
Seed replications:
 [1] 28758 78831 40898 88302 94047  4557 52811 89242 55144 45662
```

The mean of the predictive ability is 0.37 using random sampling. **?** reported values of 0.25 for across family sampling and 0.67 for within family sampling but of a different subset of the `mice` data. Moreover, they included a cage effect in the model.

For the `maize` data, we compare the predictive ability of the pedigree-based model "P-BLUP" with the genome-based models "GBLUP" and "BL". Because phenotypic performance is evaluated as testcrosses of DH lines, the relationship matrix must be replaced by the kinship matrix (**?**). Thus, relationship matrices are divided by 2. The models are fitted by

```
R> PBLUP <- gpMod(maizeC, model = "BLUP", kin = A/2)
R> GBLUP <- gpMod(maizeC, model = "BLUP", kin = U/2)
R> prior <- list(varE = list(df = 3, S = 35), lambda = list(shape = 0.52,
+      rate = 1e-04, value = 20, type = "random"))
R> modBL <- gpMod(maizeC, model = "BL", prior = prior, nIter = 6000,
+      burnIn = 1000, thin = 5)
```

Convergence of the Markov chain for `modBL` was examined visually. No convergence problems were observed. The model performance is measured by the predictive ability $r(\hat{\mathbf{g}}_{TS}, \mathbf{y}_{TS})$ from CV. The accuracy is defined as the correlation of the predicted genetic value with true breeding values $r(\mathbf{g}, \hat{\mathbf{g}})$ for the whole data set. We use 5-fold CV with 10 replications. The results are summarized in Table 2. The models using marker data outperform the pedigree-based model "P-BLUP". However, the average predictive ability of "BL" using marker-specific shrinkage is similar compared to "G-BLUP" using marker-homogeneous shrinkage.

5.5. Computation times

A crucial point in GP are computation times. In this section, we evaluate the computational requirement for the algorithms implemented within the **synbreed** package. The scaling with respect to the number of data points can be roughly assessed by a comparison of the `maize` and `mice` data. The number of data points in the marker matrices are $1250 \cdot 1117 \approx 1.4 \cdot 10^7$ and $1940 \cdot 12545 \approx 2.4 \cdot 10^8$, respectively. We list in Table 3 the elapsed system times on a

Model	cross-validation		$r(\mathbf{g}, \hat{\mathbf{g}})$
	$\bar{r}(\hat{\mathbf{g}}_{TS}, \mathbf{y}_{TS})$ (s.e.)	avg. bias (s.e.)	
P-BLUP	0.22 (0.002)	1.00 (0.013)	0.587
G-BLUP	0.53 (0.002)	1.00 (0.007)	0.856
BL	0.53 (0.002)	1.01 (0.006)	0.856

Table 2: Comparison of the model performance of pedigree-based mixed model "P-BLUP", marker-based mixed model "G-BLUP", and Bayesian Lasso regression "BL" with respect to the predictive ability, the prediction bias and the accuracy.

standard PC (Intel Core 2, 2.8 Ghz, 4 GB RAM). To impute with the `maize` data, a fraction of 0.444% of the marker genotypes was masked. This corresponds to the number of missing data points in the marker matrix of the `mice` data.

		Computation Times [sec]	
Analysis step	Function	`maize` data	`mice` data
Load data	`load`	0.6	9.4
Summary	`summary.gpData`	3.0	6.2
Plot marker map	`plotGenMap`	0.4	0.2
Recoding genotypes	`codeGeno`	5.8	68.3
Imputing genotypes (`"random"`)	`codeGeno`	5.9[a]	71.9
Imputing genotypes (`"Beagle"`)	`codeGeno`	67.0[a]	2695.8
Imputing genotypes (`"family"`)	`codeGeno`	9.0[a]	NA
Imputing genotypes (`"beagleAfterFamily"`)	`codeGeno`	67.26[a]	NA
Pairwise LD (only Chr. 1)	`pairwiseLD(,chr=1)`	2.4	679.7
Pedigree-based relationship	`kin(,ret="add")`	137.8	NA
Marker-based relationship	`kin(,ret="realized")`	1.2	47.2

Table 3: Elapsed system times (seconds) for the analysis steps using the **synbreed** package. [a] after randomly masking 0.444% values, NA=computation not possible because no pedigree is available.

We observed, that the computation times for many algorithms scale linearly with the number of data points. For a 50k SNP chip and 1000 individuals the recoding of the alleles is performed in less than 5 minutes on a standard PC. The marker-based matrix is computed within the same time frame. The model "G-BLUP" can be fitted in less than a minute. Hence, predictions for the genetic value are available in only a few minutes. However, additional analysis steps such as CV - depending on the number of splits and replications - are computationally intensive. Imputing of missing values using Beagle is very accurate but slow for a dense genome-wide marker map. For a family-stratified population of homozygous inbred lines the `"family"` algorithm is a faster alternative.

6. Conclusions and future work

The package **synbreed** provides a valuable tool within the plant and animal genetics researchers software toolbox. It offers a comprehensive collection of methods required in the analysis of GP data. The data flow is guided by a single, unified data object. Once an object of class gpData is created, all analysis steps rely on its structure. Moreover, it is very convenient to share objects of class gpData. A key issue is the generality of the class gpData. We minimized the requirements concerning the data structure. This includes replicated and unreplicated trials for phenotypic data as well as arbitrary coding of marker data either by alleles or marker genotypes. Consequently, the package can also be applied in GWAS or QTL studies. For the latter, we included a gateway to the package **qtl**. The **synbreed** package is implemented as generic as possible. Hence, the package is suitable for both plant and animal genetics researchers and not limited to any species. We have successfully tested this framework for the analysis of several scenarios with simulated and experimental data from different species (maize, rye, dairy cattle, rice and Arabidopsis).

Next generation genetics research involves computer-intensive methods to analyze massive amounts of high-throughput genotyping and large scale phenotyping data. Especially in the plant breeding community, there is a strong demand for robust standard software. We designed this package to provide access to standard methods required in GP. The **synbreed** package provides an interface to fit robust standard parametric GP models being publicly available. This allows researchers to conduct standard analyses, e.g., using mixed models. Moreover, the framework of R enhances the analysis and visualization of results in one software. All code is given in the R language. This permits the users to customize the methods to specific needs. When necessary, we provide gateways to standard software such as Beagle or Plink. Beside research and routine analysis, the package is also useful in the education of young scientists and breeders. It provides fast access to a wide range of different analysis methods and a hand-on tutorial with example data sets. Most functions have default values and short argument lists. Thus graduate students can gain expeditious insight into the statistical models and apply GP methods without having profound programing skills.

One objective of the **synbreed** package was to create a gateway to related software programs. Thereby we utilize the strength of R to integrate functions by the package structure. We provide an interface to fit GP models using the mixed and Bayesian regression models. The model performance can be investigated using cross-validation. The function gpData2data.frame allows the conversion to a data.frame. This comprehensive format can be used by many other R functions. Other software packages such as ASReml or WOMBAT are frequently used in genetics and breeding research. An object of class relationship matrix can be stored as a file using function write.relationshipMatrix which meets the required formats for ASReml or WOMBAT. Thus data in **synbreed** package can be used straightforward by other software. Moreover, we included functions to prepare the input files for the programs Beagle and Plink. We provide an interface to use the methods therein for objects of class gpData. Combining the functionality of R with Sweave (**?**) and LaTeX enables to create automatic reports presenting results for e.g., predicted genetic performance and SNP effects within tables and graphics. This is a step towards automatized analysis pipelines in the analysis of next generation genotype and phenotype data.

There is no universal constraint in the **synbreed** package with respect to the number of data points. Rather computational facilities impose limitations. Table 3 gives an overview of computation times for both data sets. Various analysis steps are performed in seconds. In general, the elapsed time increases linearly with respect to the number of data points. However, as data output from next generation sequence technology is emerging faster than computational capacities, additional effort is required to improve the algorithms. The open-source structure of R allows the user to customize the algorithms. Solutions can be the use of parallel computing utilizing the power of multi-core systems (**?**). Hence, computation times are likely to be reduced by an order of magnitude, especially on a multi-core cluster. The claimed workspace can be reduced by using sparse matrix methods (e.g., R package **ff**, **?**). The current version of the package enables genotype-based analyses. However, using haplotypes is desirable, too. For the future, we are aiming to include an additional data object for haplotypes and provide tools to analyze them. As available amount of data points are expected to increase, future work will also include the embedding of foreign languages, i.e., C.

Acknowledgments

This research was funded by the German Federal Ministry of Education and Research (BMBF) within the AgroClustEr *Synbreed – Synergistic plant and animal breeding (FKZ 0315528A)*. We gratefully acknowledge Larry Schaeffer for providing us R code which is used by functions `kin` and `MME`, Michael Höhle for valuable discussions with respect to the manuscript and package design and Richard Mott for his help with the `mice` data. Henner Simianer, Malena Erbe and Christian Reimer are acknowledged for providing and preparing the `cattle` data. All users of the beta-version of the **synbreed** package are acknowledged for their suggestions and contributions.

Affiliation:

Valentin Wimmer
Lehrstuhl für Pflanzenzüchtung
Technische Universität München
85354 Freising, Germany
E-mail: `Valentin.Wimmer@wzw.tum.de`