Decoding genotype-phenotype associations with genphen

Simo Kitanovski, Daniel Hoffmann,
Bioinformatics and Computational Biophysics,
University of Duisburg-Essen, Essen, Germany

March 11, 2019

Contents

This tutorial gives you some of the technical background underlying genphen that should enable you to understand and use this tool.

1 genphen quantifies genotype-phenotype associations

Genome wide association studies (GWAS) have become an important tool for understanding the association between genotypes and phenotypes. With GWAS we try to answer questions such as "what are the genotypes in the human genome which predispose to a disease?" or "what are the genotypes in certain strains of mice which confer resistance against a specific virus?". The advances in high- throughput sequencing technology (HTSeq) have provided massive genetic data and thus potentially countless applications for GWAS,

with genotypes representing single nucleotide polymorphisms (SNPs) or single amino acid polymorphisms (SAAPs) identified in a group of individuals, whereas the phenotype can be any quantiative or discrete trait or charactersitic measured in each individual.

The classical (frequentist) statistical methods for GWAS rely on simple and often inadequate methods to capture complex and potentially non-linear genotype- phenotype associations. By quantifying the strength of the association using P-values, the frequentist methods often run into the multiple-comparison problem, which when countered with a rigorous P-value correction results in large amounts of false-negatives. Additional disadvantages of the P-values include the facts that they are difficult to interpret and compare between studies.

With `genphen` we provide a hybrid method for the joint analysis of multiple traits of different typies which reaps the benefits of two approaches: i) statistical learning approaches such as random forest (RF) and support vector machine (SVM) to capture complex associations; ii) Bayesian inference using hierarchical models for accurate quantification of the strength of association, whereby the models are robust to outliers, consistent with the data and automatically deal with the multiple-hypothesis problem.

Furthermore, `genphen` provides a set of additional procedures, including a test for phylogenetic bias (used to discover biases in the data due to the population structure) and procedure for data reduction (used for the removal of non-informative genotypes and thereby simplifying the otherwiese computationally costly GWAS). Future updates will include procedures for data augmentation (to augment small/noisy datasets) and methods for gene prioritization based on network diffusion algorithms using functional network data.

2 Methods

2.1 Input

Two types of data are necessary to perform a genetic association study:

- genotype data (e.g. set of 1,000 SNPs found along the aligned genomes of 10 individuals), provided in one of three possible input types:
  - **character vector** of length N (if only a single SNP/SAAP is provided), containing the genotypes of N individuals.
  - **character matrix** with dimensions $N \times S$ (with N = individuals, S = SNPs/SAAPs)
  - **AAMultipleAlignment** or **DNAMultipleAlignment** object (package `Biostrings`) - if the genotype data is a multiple sequence alignment, composed of N sequences (individuals).
- phenotype data which can include a combination of dichotomous and quantitative traits is allowed (experimental measurement made for each individual such as body mass index, immune response, survival, case-control, etc.) provided as:
  - numerical vector of length N if only a single phenotype is analyzed
  - numerical matrix $N \times P$, if P phenotypes are provided.

2.2 Association Scores

With `genphen` we quantify the association between each genotype (SNP/SAAP) and phenotype using multiple metrics of association, each of which is explained in the following paragraphs.

Classification accuracy (CA) CA quantifies the degree of accuracy with which one can classify (predict) the alleles of a SNP from the phenotype. If there exists a strong association between a particular SNP and the phenotype, one should be able to train a statistical model (using RF or SVM) which accurately classifies the two alleles of that SNP solely from the phenotype data (with $CA \approx 1$). Otherwise, the model should perform poorly, with the classification accuracy of the model being approximately similar to that of simple guessing ($CA \approx 0.5$).

To estimate CA, `genphen` uses RF and SVM in a cross-validation (CV) mode, computing a distribution of possible CAs for each SNP. During

each iteration of the CV procedure, a subset (e.g. 66%) of the genotype-phenotype data is selected at random (with replacement) and used to train a classifier, followed by testing (prediction) based on the remaining data. To summarize CA, we compute its mean and 95% highest density interval (95% HDI), which is defined as the interval that covers 95% of the CA distribution, with every point inside the interval having higher credibility than any point outside it. The SNPs with $CA \approx 1$, and a narrow HDI have a strong association with the phenotype.

Cohen's κ statistic There is one pitfall where the CA estimate can be misleading, and this is the case when the analyzed SNP is composed of unevenly represented genetic states (alleles). For instance, the allele A of a given SNP is found in 90% of the individuals, while the other allele T in only 10%. Such an uneven composition of the alleles can lead to a misleading CA, i.e. even without learning, the algorithm can produce a high $CA \approx 0.9$ by always predicting the dominant label. The Cohen's κ statistics can be used to estimate the degree of CA above the expected accuracy (CA_{exp}):

$$\kappa = (CA - CA_{exp})/(1 - CA_{exp})$$

The κ statistics is a quality metric, which is to be used together with CA. Cohen defines the following meaningful κ intervals: [κ<0]: "no agreement", [0.0-0.2]: "slight agreement", [0.2-0.4]: "fair agreement" , [0.4-0.6]: "moderate agreement", [0.6-0.8]: "substantial agreement" and [0.8-1.0]: "almost perfect agreement". To summarize the Cohen's κ, we compute its mean and 95% highest density interval (95% HDI).

Effect size Given N individuals, each having genotype values for S refSNPs, we generate the genotype matrix $X^{N\times S}$. The genotype matrix can also be refereed to as a design matrix in the genetic association study, with X_{ij} set to 1 if an individual has the first allele, and X set to -1 for the second allele. For multi-allelic genotypes, the genotype matrix is expanded (colmn-wise) to include each bi-allelic permutation. The phenotypes P can also be grouped to form the phenotype matrix $Y^{N\times P}$. We model the effect of each SNP/SAAP on each phenotype using the following Bayesian model:

$$Y_{ik} \sim \begin{cases} \text{Student-t}\left(\nu_k, \alpha_{jk} + \beta_{jk} \cdot X_{ij}, \sigma_k\right), & \text{if } k \text{ quant.} \\ \text{Bernoulli}\left(\text{logit}^{-1}(\alpha_{jk} + \beta_{jk} \cdot X_{ij})\right), & \text{if } k \text{ dich.} \end{cases}$$

where i and j and k index different individuals, SNPs and phenotypes; For quantitative trait, the model assumes Student-t distributed measurement errors with phenotype-specific standard deviation (σ) and degrees of freedom (ν), with central tendency defined by $\alpha_k + \beta_{jk} \cdot X_{ij}$, where α and β are

the inferred intercept and slope (effect size) coefficients of the model. For dichotomous traits, the model assumes Bernoulli distributed measurement errors. Assuming that all SNPs are independent, we can use the univariate model setting in `genphen` to place independent vague priors on all slope and intercept coefficients as:

$$\beta_{jk} \sim \text{Student-t}(1, 0, 10)$$
$$\alpha_{jk} \sim \text{Student-t}(1, 0, 100)$$

On the other hand, if we assume that the estimated slopes are not completely independent and originate from a common overarching distribution, we can use the hierarchical model setting in `genphen` to model the hierarchy as:

$$\beta_{jk} \sim \text{Student-t}(\nu_{\beta_k}, \mu_{\beta_k}, \sigma_{\beta_k})\alpha_{jk} \sim \text{Student-t}(\nu_{\alpha_k}, \mu_{\alpha_k}, \sigma_{\alpha_k})$$

The remaining lines of the model describes the priors of the remaining parameters defined in either model type:

$$\mu_{\beta_k} \sim \text{Student-t}\,(1, 0, 10)$$
$$\mu_{\alpha_k} \sim \text{Student-t}\,(1, 0, 100)$$
$$\nu_k, \nu_{\beta_k} \sim \text{Gamma}\,(1, 2)$$
$$\sigma_k, \sigma_{\beta_k} \sim \text{Half-Cauchy}\,(0, 5)$$

Importantly, the hierarchical version of the model performs partial-pooling, and therefore does an automatic correction for multiple-comparison. The model was implemented in Stan [1].

We summarize each association using the mean of its slope coefficient (β) and 95% (for instance) highest density interval (HDI), which is defined as the interval that covers a 95% of the posterior distribution, with every point inside the interval having a higher credibility than any point outside it. Thus we can define an association as significant if the null-effect, i.e. $\beta = 0$ lies outside the 95% HDI. The complete posterior is provided to the user, enabling checks for MCMC convergence and posterior prediction using built-in routines for poterior predictive checks.

2.3 Phylogenetic Bias (B)

To control for potential phylogenetic biases (population structure), we devised the following procedure. First, we use the complete genotype data (all SNPs) to compute a kinship matrix ($K^{N\times N}$ - dissimilarity matrix for

[1] Stan Development Team. 2017. Stan Modeling Language Users Guide and Reference Manual, Version 2.17.0. http://mc-stan.org

the N individuals). Alternatively, the users can provide their own kinship matrix (e.g. kinship estimated using more accurate phylogenetic methods). For a group of individuals which belong to a group defined by an alleles of a given SNP, we next compute their mean kinship distance using the kinship matrix data. If the individuals in the group are related, the compute mean kinship distance must be significantly lower than the mean kinship distance computed from the complete kinship matrix. We define the phylogenetic bias as:

$$B = 1 - \hat{d}_g/\hat{d}_t$$

where $\hat{d}_g$ is the mean kinship distance between the individuals who share the genotype g; $\hat{d}_t$ is the mean kinship distance of the complete kinship matrix. For a complete phylogenetic bias, $B = 1$ ($\hat{d}_g << \hat{d}_t$), and $B = 0$ (or slightly negative) for no bias. This estimate is computed for each SNP and genotype group within each SNP.

To compute the phylogenetic bias associated with a SNP we compute:

$$B = 1 - min(\hat{d}_{g_1}, \hat{d}_{g_2})/\hat{d}_t$$

where $\hat{d}_{g_1}$ and $\hat{d}_{g_2}$ represent the mean kinship distance between the individuals who share the genotype (allele) g_1 and g_2 or a given SNP; $\hat{d}_t$ is the mean kinship distance in the complete kinship matrix. For a complete phylogenetic bias, $B = 1$ and $B = 0$ (or slightly negative) for no bias. This estimate is computed for each SNP and each pair of genotypes.

3 Case studies

3.1 I: Association between SNPs and a *quantiative* phenotype

In the first case study, we show a typical genotype-phenotype analysis, whereby the genotype is a multiple sequence alignment containing of 120 protein sequences (individuals), each composed of 8 amino acids (positions), and a quantiative phenotype measured for each individual.

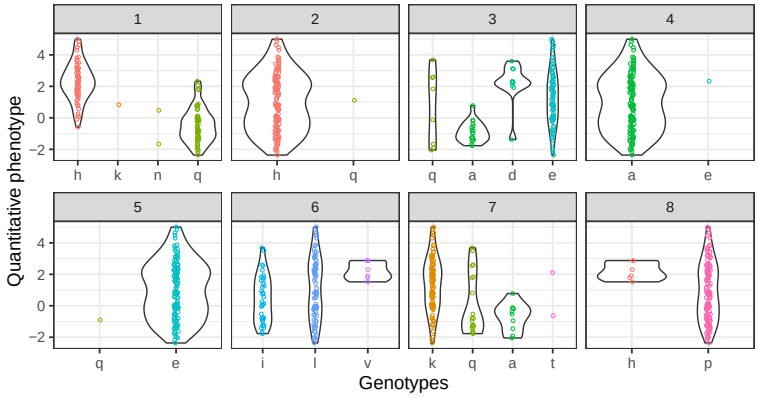
```
> require(genphen)
> require(ggplot2)
> require(knitr)
> require(ggrepel)
> require(reshape)
> require(ape)
> require(xtable)
> require(gridExtra)
> options(xtable.floating = FALSE)

> # genotype as matrix (120x154), we will subset part of it:
> data(genotype.saap)
> # phenotype as vector (120 measurements):
> data(phenotype.saap)
```

Genotype-phenotype data First we show an overview of the distribution of the phenotype across the genetic states found at each of the 8 studied positions in the multiple sequence alignment.

```
> # Format the genotype-phenotype data, such that it can then
> # be visualized with ggplot
> df <- data.frame(genotype.saap[, 82:89],
+                  phenotype = phenotype.saap,
+                  stringsAsFactors = FALSE)
> df <- melt(data = df, id.vars = "phenotype")
> colnames(df) <- c("phenotype", "site", "genotype")
> df$site <- gsub(pattern = "X", replacement = '', x = df$site)
> df$site <- factor(x = df$site, levels = unique(df$site))

> # Visualization
> g <- ggplot(data = df)+
+   facet_wrap(facets = ~site, nrow = 2, scales = "free_x")+
+   geom_violin(aes(x = genotype, y = phenotype))+
+   ylab(label = "Quantitative phenotype")+
+   xlab(label = "Genotypes")+
+   geom_point(aes(x = genotype, y = phenotype, col = genotype),
+              size = 1, shape = 21, position = position_jitterdodge())+
+   scale_color_discrete(name = "genotype")+
+   theme_bw(base_size = 14)+
+   theme(legend.position = "none")
> g
```



Important remark: We recommended that the quantiative phenotypes are roughly normally (or T) distributed. While our models are designed to be robust against outliers, we advise you to perform the data transformations of skewed phenotypes (e.g. log-transformations) before the analysis. Here the phenotype has already been log10-transformed and is normally distributed.

Association analysis Next, we perform the genetic association study for a single quantitative ('Q') phenotype with `genphen` using the following settings:

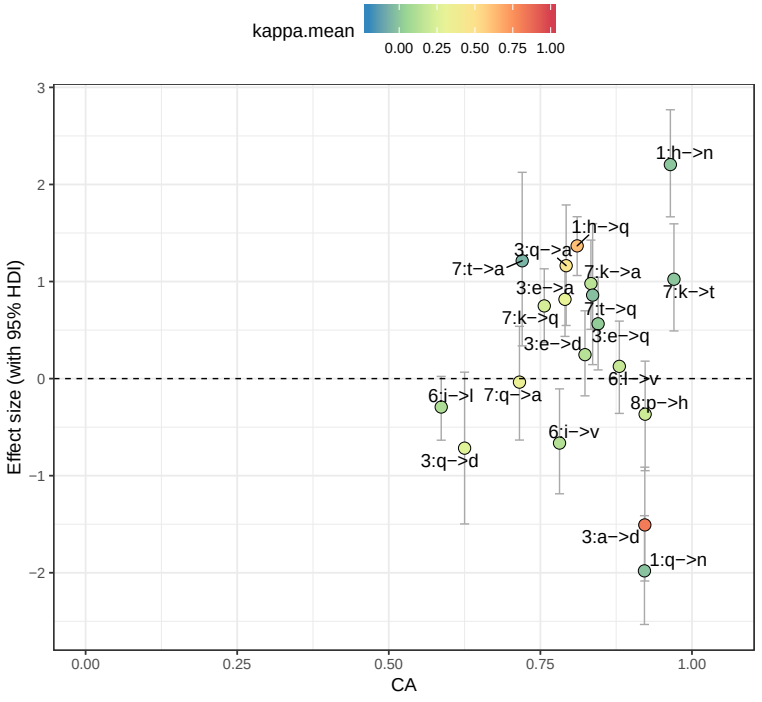
- hierarchical Bayesian model will be run with 2 MCMC chains composed of 1,500 iterations each, including 500 warmup iterations.
- Random forest was selected for the statistical learning, which will be run in a cross-validation mode with 200 iterations, whereby in each iteration 66% of the data (default: cv.fold = 0.66) will be used to train the model.
- We report for each metrics its mean and 95% HDI
- Whenever possible, 2 cores will be used.

```
> # Run genphen
> c.out <- genphen::runGenphen(genotype = genotype.saap[, 82:89],
+                              phenotype = phenotype.saap,
+                              phenotype.type = "Q",
+                              model.type = "hierarchical",
+                              mcmc.chains = 2,
+                              mcmc.steps = 1500,
+                              mcmc.warmup = 500,
+                              cores = 2,
+                              hdi.level = 0.95,
+                              stat.learn.method = "rf",
+                              cv.steps = 200)
```

Typical way of visualizing the `genphen` results is with the following plot, where each point represents a SAAP plotted according to x = classification

accuracy (CA), y = effect slize (β), color = Cohen's κ. The most promising SAAPs have CA and κ close to 1, with a non-null β, i.e. β with 95% HDI that does not overlap with 0 (shown as a dashed line in the figure). The labels show the SAAP site in the genotype data, and its constituting genotypes.

```
> # Get the scores data
> c.score <- c.out$scores
> # Some optional formatting for the SNPs (label = site : genotype1 -> genotype2)
> c.score$label <- paste(c.score$site, ":", c.score$ref,
+                        "->", c.score$alt, sep = '')
> # Visualization
> g <- ggplot(data = c.score)+
+   geom_errorbar(aes(x = ca.mean, ymin = beta.hdi.low, ymax = beta.hdi.high),
+                 width = 0.015, col = "darkgray")+
+   geom_point(aes(x = ca.mean, y = beta.mean, fill = kappa.mean),
+              shape = 21, size = 4)+
+   geom_text_repel(aes(x = ca.mean, y = beta.mean, label = label), size = 5)+
+   theme_bw(base_size = 14)+
+   ylab(label = "Effect size (with 95% HDI)")+
+   scale_x_continuous(name = "CA", limits = c(0, 1.05))+
+   geom_hline(yintercept = 0, linetype = "dashed")+
+   theme(legend.position = "top")+
+   scale_fill_distiller(palette = "Spectral", limits = c(-0.2, 1))+
+   guides(fill = guide_colorbar(barwidth = 10, barheight = 1.5))
> g
```



The association scores are also shown in the following table:

```
> # Description:
> # Rounds digits to 2-decimal points, and concatinates the lower and upper
> # limits of the HDI to have a simpler visualization
> getHdiPretty <- function(x, digits = 2) {
+   x[1] <- round(x = x[1], digits = digits)
+   x[2] <- round(x = x[2], digits = digits)
+   return(paste("(", x[1], ", ", x[2], ")", sep = ''))
+ }
> c.score$beta.hdi <- apply(X = c.score[, c("beta.hdi.low", "beta.hdi.high")],
+                           MARGIN = 1, getHdiPretty, digits = 2)
> c.score$ca.hdi <- apply(X = c.score[, c("ca.hdi.low", "ca.hdi.high")],
+                         MARGIN = 1, getHdiPretty, digits = 2)
> c.score$kappa.hdi <- apply(X = c.score[, c("kappa.hdi.low", "kappa.hdi.high")],
+                            MARGIN = 1, getHdiPretty, digits = 2)
> # Print table
> print(xtable(c.score[, c("label", "beta.mean", "beta.hdi", "ca.mean",
+                          "ca.hdi", "kappa.mean", "kappa.hdi"), ],
+              align = rep(x = "c", times = 8, digits = 2)),
+       include.rownames = FALSE, size = "scriptsize")
```

label	beta.mean	beta.hdi	ca.mean	ca.hdi	kappa.mean	kappa.hdi
1:h->q	1.37	(1.06, 1.67)	0.81	(0.72, 0.9)	0.62	(0.43, 0.79)
1:h->n	2.20	(1.67, 2.77)	0.96	(0.9, 1)	-0.00	(-0.05, 0)
1:q->n	-1.98	(-2.53, -1.41)	0.92	(0.84, 1)	-0.02	(-0.08, 0)
3:e->q	0.56	(0.09, 1.04)	0.85	(0.74, 0.94)	0.02	(-0.14, 0.35)
3:e->a	0.82	(0.43, 1.2)	0.79	(0.68, 0.88)	0.29	(-0.01, 0.6)
3:e->d	0.25	(-0.18, 0.7)	0.82	(0.71, 0.94)	0.13	(-0.15, 0.53)
3:q->a	1.16	(0.55, 1.79)	0.79	(0.5, 1)	0.47	(0, 1)
3:q->d	-0.72	(-1.5, 0.07)	0.62	(0.33, 0.83)	0.26	(-0.2, 0.67)
3:a->d	-1.51	(-2.08, -0.91)	0.92	(0.89, 1)	0.82	(0.4, 1)
6:i->l	-0.29	(-0.63, 0.02)	0.59	(0.45, 0.71)	0.10	(-0.17, 0.33)
6:i->v	-0.66	(-1.19, -0.11)	0.78	(0.6, 0.93)	0.12	(-0.19, 0.63)
6:l->v	0.13	(-0.36, 0.59)	0.88	(0.78, 0.96)	0.18	(-0.1, 0.65)
7:k->t	1.02	(0.49, 1.59)	0.97	(0.93, 1)	-0.00	(-0.03, 0)
7:k->q	0.75	(0.37, 1.13)	0.76	(0.64, 0.86)	0.23	(-0.06, 0.54)
7:k->a	0.98	(0.51, 1.43)	0.83	(0.73, 0.91)	0.14	(-0.12, 0.53)
7:t->q	0.86	(0.14, 1.6)	0.84	(0.57, 1)	-0.02	(-0.24, 0)
7:t->a	1.21	(0.34, 2.12)	0.72	(0.5, 1)	-0.06	(-0.33, 0)
7:q->a	-0.04	(-0.63, 0.54)	0.72	(0.5, 0.9)	0.34	(-0.19, 0.78)
8:p->h	-0.37	(-0.95, 0.18)	0.92	(0.85, 0.98)	0.20	(-0.06, 0.66)

MCMC convergence and sampling issues You might want to check the validity your Bayesian inference by inspecting the `genphen` output named convergence which contains information about the markov chain monte carlo (MCMC) simulation done with R package rstan including potential scale reduction factor (Rhat) and effective sampling size (ESS), as well as information concerning potential convergence issues such as divergences and tree depth exceeded warnings. For detailed information about each warning please read Stan documentation (mc-stan.org/users/documentation/).

```
> rstan::check_hmc_diagnostics(c.out$complete.posterior)
> rstan::stan_rhat(c.out$complete.posterior)
> rstan::stan_ess(c.out$complete.posterior)
> rstan::stan_diag(c.out$complete.posterior)
```

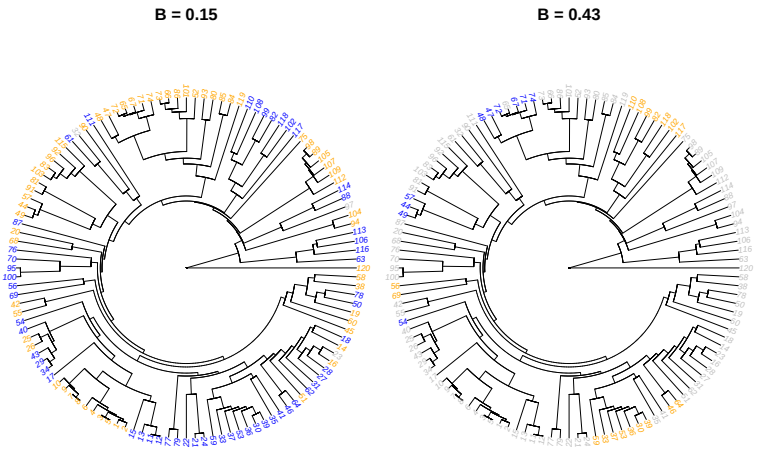
Phylogenetic bias control Next, we compute the phylogenetic bias of each mutation, shown in the table below:

```
> # Compute the phylogenetic bias
> bias <- runPhyloBiasCheck(genotype = genotype.saap,
+                           input.kinship.matrix = NULL)
> # Extract kinship matrix
> kinship.matrix <- bias$kinship.matrix
> # Extract the bias associated with mutations of the sites which
> # were included in the association analysis
> mutation.bias <- bias$bias
> # To make site id concordant with data
> mutation.bias$site <- mutation.bias$site - 81
> mutation.bias <- merge(x = c.score, y = mutation.bias,
+                        by = c("site", "ref", "alt"))
> # Show the bias table
> print(xtable(mutation.bias[, c("site", "ref", "alt", "bias.ref", "bias.alt")],
+              align = rep(x = "c", times = 6, digits = 2)),
+       include.rownames = FALSE, size = "small")
```

site	ref	alt	bias.ref	bias.alt
1	h	n	0.04	0.43
1	h	q	0.04	0.15
1	q	n	0.15	0.43
3	a	d	0.29	0.43
3	e	a	0.04	0.29
3	e	d	0.04	0.43
3	e	q	0.04	0.19
3	q	a	0.19	0.29
3	q	d	0.19	0.43
6	i	l	0.10	0.07
6	i	v	0.10	0.86
6	l	v	0.07	0.86
7	k	a	0.04	0.46
7	k	q	0.04	0.09
7	k	t	0.04	0.79
7	q	a	0.09	0.46
7	t	a	0.79	0.46
7	t	q	0.79	0.09
8	p	h	0.03	0.86

We use the kinship matrix to perform hierarchical clustering, visualizing the population strcuture and two examples (mutations) with genotype 1 marked with blue and genotype 2 marked with orange in either case. Individuals not covered by either genotype are marked with gray color. The shown examples differ in the degree of phylogenetic bias.

```
> color.a <- character(length = nrow(genotype.saap))
> color.a[1:length(color.a)] <- "gray"
> color.a[which(genotype.saap[, 82] == "h")] <- "orange"
> color.a[which(genotype.saap[, 82] == "q")] <- "blue"
> color.b <- character(length = nrow(genotype.saap))
> color.b[1:length(color.b)] <- "gray"
> color.b[which(genotype.saap[, 84] == "a")] <- "orange"
> color.b[which(genotype.saap[, 84] == "d")] <- "blue"
> c.hclust <- hclust(as.dist(kinship.matrix), method = "average")
> par(mfrow = c(1, 2), mar = c(0,0,1,0) + 0.1)
> plot(as.phylo(c.hclust), tip.color = color.a, cex = 0.6,
+      type = "fan", main = "B = 0.15")
> plot(as.phylo(c.hclust), tip.color = color.b, cex = 0.6,
+      type = "fan", main = "B = 0.43")
```



3.2 II: Association between SNP and two phenotypes (quantitative and dichotomous)

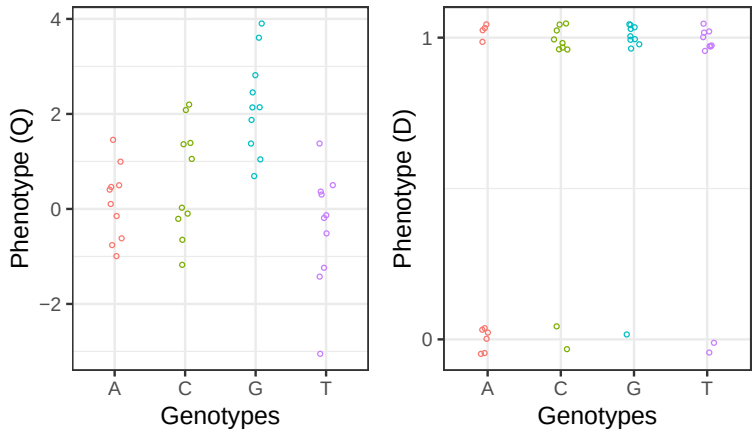
In the second case study we show you how to use `genphen` in case of two phenotypes of different types (quantitative and dichotomouse). Here, the genotype is a single simulated SNP in 40 individuals. First we show an overview of the distribution of the phenotypes in each genotype.

```
> # simulate genotype
> genotype <- rep(x = c("A", "C", "T", "G"), each = 10)
> # simulate quantitative and dichotomous phenotypes
> phenotype.Q <- c(rnorm(n = 10, mean = 0, sd = 1),
+                  rnorm(n = 10, mean = 0.5, sd = 1),
+                  rnorm(n = 10, mean = -0.5, sd = 1),
+                  rnorm(n = 10, mean = 2, sd = 1))
> phenotype.D <- c(rbinom(n = 10, size = 1, prob = 0.3),
+                  rbinom(n = 10, size = 1, prob = 0.5),
+                  rbinom(n = 10, size = 1, prob = 0.6),
+                  rbinom(n = 10, size = 1, prob = 0.7))
> phenotype <- cbind(phenotype.Q, phenotype.D)
> rm(phenotype.Q, phenotype.D)
> out <- runGenphen(genotype = genotype,
+                   phenotype = phenotype,
+                   phenotype.type = c("Q", "D"),
+                   model.type = "hierarchical",
+                   mcmc.chains = 4,
+                   mcmc.steps = 2500,
+                   mcmc.warmup = 500,
+                   cores = 2,
+                   hdi.level = 0.95,
+                   stat.learn.method = "svm",
+                   cv.steps = 500)

> # Format the genotype-phenotype data, such that it can then
> # be visualized with ggplot
> df <- data.frame(genotype = genotype,
+                  phenotype.Q = phenotype[, 1],
```

```
+                  phenotype.D = phenotype[, 2],
+                  stringsAsFactors = FALSE)

> # Visualization
> g1 <- ggplot(data = df)+
+   geom_point(aes(x = genotype, y = phenotype.Q, col = genotype), size = 1,
+              shape = 21, position = position_jitterdodge(jitter.width = 0.2,
+                                                          jitter.height = 0,
+                                                          dodge.width = 0.5))+
+   xlab(label = "Genotypes")+
+   ylab(label = "Phenotype (Q)")+
+   theme_bw(base_size = 14)+
+   theme(legend.position = "none")
> g2 <- ggplot(data = df)+
+   geom_point(aes(x = genotype, y = phenotype.D, col = genotype), size = 1,
+              shape = 21, position = position_jitterdodge(jitter.width = 0.2,
+                                                          jitter.height = 0.05,
+                                                          dodge.width = 0.5))+
+   xlab(label = "Genotypes")+
+   scale_y_continuous(name = "Phenotype (D)",
+                      breaks = c(0, 1), labels = c(0, 1))+
+   theme_bw(base_size = 14)+
+   theme(legend.position = "none")
> gridExtra::grid.arrange(g1, g2, ncol = 2)
```



Important remark: The dichotomous phenotype can be provided as both numeric or character vector. The elements of these vectors are then encoded into two categories (1 and 0). If the user has a preference of how the encoding has to be done (which category is to be encoded to 1 or 0), the encoding should be done prior to the analysis.

Association analysis We perform a genetic association study for multiple phenotypes of different types, including one quantiative ('Q') and one dichotomous phenotype ('D'), using the following settings:

- univariate Bayesian model will be run with 4 MCMC chains composed

of 1500 iterations each, including 500 warmup iterations.

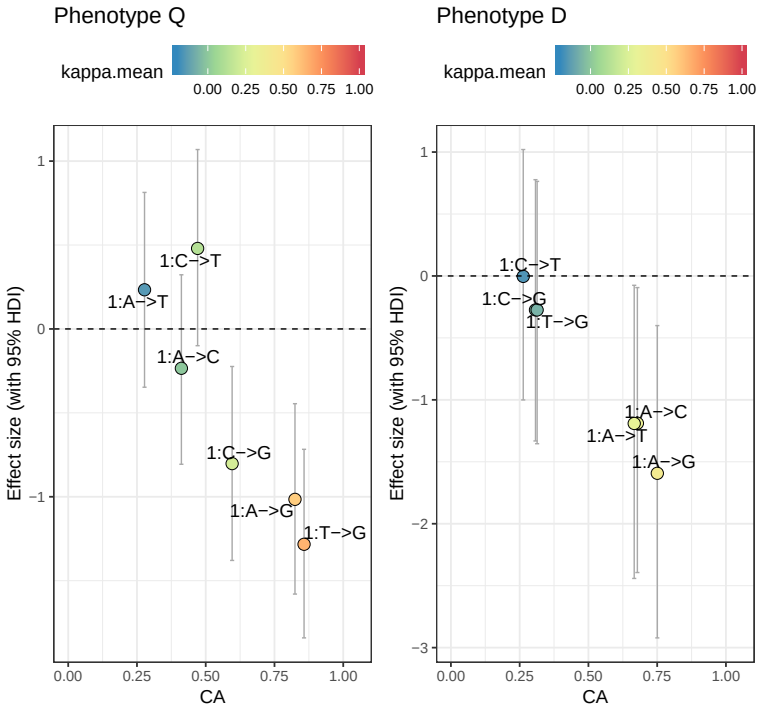
- Support vector machines was selected for the statistical learning, which will be run in a cross-validation mode with 500 iterations, whereby in each iteration 80% of the data will be used to train the model.
- All estimates will be reported according to their mean and 95% HDI
- Whenever possible, 2 cores will be used.

```
> # run genphen
> m.out <- genphen::runGenphen(genotype = genotype,
+                              phenotype = phenotype,
+                              phenotype.type = c("Q", "D"),
+                              model.type = "univariate",
+                              mcmc.chains = 4,
+                              mcmc.steps = 1500,
+                              mcmc.warmup = 500,
+                              cores = 2,
+                              hdi.level = 0.95,
+                              stat.learn.method = "svm",
+                              cv.steps = 500,
+                              cv.fold = 0.8)
```

Once again we visualize the `genphen` results is with a plot in which the point represents the SNP, plotted according to x = classification accuracy (CA), y = slope (β) with error bars representing the 95% HDI, color = Cohen's κ.

```
> # Get the scores data
> m.score <- m.out$scores
> # Some optional formatting for the SNPs
> # (label = site : genotype1 -> genotype2)
> m.score$label <- paste(m.score$site, ":", m.score$ref,
+                        "->", m.score$alt, sep = '')
> # Visualization
> g1 <- ggplot(data = m.score[m.score$phenotype.id == 1, ])+
+   geom_errorbar(aes(x = ca.mean, ymin = beta.hdi.low, ymax = beta.hdi.high),
+                 width = 0.015, col = "darkgray")+
+   geom_point(aes(x = ca.mean, y = beta.mean, fill = kappa.mean),
+              shape = 21, size = 4)+
+   geom_text_repel(aes(x = ca.mean, y = beta.mean, label = label), size = 5)+
+   theme_bw(base_size = 14)+
+   ylab(label = "Effect size (with 95% HDI)")+
+   scale_x_continuous(name = "CA", limits = c(0, 1.05))+
+   geom_hline(yintercept = 0, linetype = "dashed")+
+   theme(legend.position = "top")+
+   scale_fill_distiller(palette = "Spectral", limits = c(-0.2, 1))+
+   guides(fill = guide_colorbar(barwidth = 10, barheight = 1.5))+
+   ggtitle(label = "Phenotype Q")
> g2 <- ggplot(data = m.score[m.score$phenotype.id == 2, ])+
+   geom_errorbar(aes(x = ca.mean, ymin = beta.hdi.low, ymax = beta.hdi.high),
+                 width = 0.015, col = "darkgray")+
+   geom_point(aes(x = ca.mean, y = beta.mean, fill = kappa.mean),
+              shape = 21, size = 4)+
+   geom_text_repel(aes(x = ca.mean, y = beta.mean, label = label), size = 5)+
+   theme_bw(base_size = 14)+
+   ylab(label = "Effect size (with 95% HDI)")+
+   scale_x_continuous(name = "CA", limits = c(0, 1.05))+
+   geom_hline(yintercept = 0, linetype = "dashed")+
+   theme(legend.position = "top")+
```

```
+   scale_fill_distiller(palette = "Spectral", limits = c(-0.2, 1))+
+   guides(fill = guide_colorbar(barwidth = 10, barheight = 1.5))+
+   ggtitle(label = "Phenotype D")
> grid.arrange(g1, g2, ncol = 2)
```



4 Extra Utilities

4.1 Data Reduction

The methods implemented in `genphen` are statistically superior to the ones implemented by most classical (frequentist) tools for GWAS. The major challenge, however, is the substantially increased computational cost when analyzing the effects of hundreeds of thousands of SNPs. Inspired by the biological assumption that the major fraction of the studied SNPs are non-informative (genetic noise) with respect to the selected phenotype, various data reduction techniques can be implemented to quickly scan the SNP scpae and discard a substantial portion of the the SNPs deemed clearly non-informative.

Our data reduction procedure includes the following steps:

1. The complete data (genotypes and a single phenotype) is used to train a random forest (RF) model, which will quantify the importance of each SNP/SAAP in explaining the phenotypeassociation between each SNP and the phenotype.
2. We can then plot the distribution of variable importances, to get an insight into the structure of the importances values and potentially disect the signal from the noise.
3. The main analysis can then be performed with runGenphen using a subset (based on their importance) of SNPs

   Using a case study based on a simulated data of 50,000 SNPs (60 subjects), we elaborate the typical data reduction steps in more detail. The typical runtime for the provided dataset ($60 \times 50,000$) is few minutes.

```
> # Simulate 50,000 SNPs and 60 phenotypes
> set.seed(seed = 551155)
> g1 <- replicate(n=5*10^4, expr=as.character(rbinom(n=30, size = 1,prob = 0.49)))
> g2 <- replicate(n=5*10^4, expr=as.character(rbinom(n=30, size = 1,prob = 0.51)))
> gen <- rbind(g1, g2)
> phen <- c(rnorm(n = 30, mean = 3, sd = 3), rnorm(n = 30, mean = 5, sd = 3))
```

```
> # Run diagnostics
> diag <- genphen::runDiagnostics(genotype = gen,
+                                 phenotype = phen,
+                                 phenotype.type = "Q",
+                                 rf.trees = 50000)
```

   We can inspect the distribution of importances and select a set of promising SNP based on their importance score, which can then be studied in the main association study as explained previously.

```
> # Visualization
> g <- ggplot(data = diag)+
+   geom_density(aes(importance))+
```

```
+   xlab("Importance")+
+   theme_bw(base_size = 14)+
+   scale_x_continuous(trans = "log10")+
+   annotation_logticks(base = 10, sides = "b")
> g
```