# Decoding genotype-phenotype associations with genphen

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

October 22, 2018

## 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 continuous 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 which reaps the benefits of two approaches: i) sophisticated 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), procedure for data reduction (used for the removal of non-informative genotypes and thereby simplifying the otherwiese computationally costly GWAS), procedure for retrospective power analysis (RPA) and a set of visualization procedures which enable the user to identify the most promissing hits in the 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 containing the genotypes of $N$ individuals is to be analyzed.
  - **character matrix** with dimensions $N \times M$ whereby the $M$ columns represent different SNPs/SAAPs, and the $N$ rows represent the different individuals.
  - **AAMultipleAlignment** or **DNAMultipleAlignment** object (package `Biostrings`) - if the genotype data is a multiple sequence alignment, composed of $N$ sequences (individuals).
- phenotype data (experimental measurement made for each individual such as body mass index, immune response, survival, case-control, etc.) provided as a vector of length $N$.

## 2.2 Association Scores

Between each genotype (SNP/SAAP) and phenotype, `genphen` computes several measures of association, each of which is explained in the following paragraphs.

**Classification accuracy ($CA$)** $CA$ measures 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 $\kappa$ 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 $\kappa$ statistics can be used to estimate the degree of $CA$ above the expected accuracy ($CA_{exp}$):

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

The $\kappa$ statistics is a quality metric, which is to be used together with $CA$. Cohen defines the following meaningful $\kappa$ intervals: [$\kappa$<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 $\kappa$, we compute its mean and 95% highest density interval (95% HDI).

**Bayesian inference with hierarchical models** Given data from two groups (two allele groups in a SNP), we ask the question of how much one group differs from the other with respect to the phenotype observed in each group? We developed the following Bayesian hierarchical generalized linear model (GLM) to answer this question for **i) continuous traits**:

$$\begin{gathered}
Y_{ij} \sim \text{Student-t}(\nu, \alpha_j + \beta_j * X_{ij}, \sigma) \\
\alpha_j \sim \text{Student-t}(\nu_\alpha, \mu_\alpha, \sigma_\alpha) \\
\beta_j \sim \text{Student-t}(\nu_\beta, \mu_\beta, \sigma_\beta) \\
\nu \sim \text{Gamma}(2.0, 0.1) \\
\sigma \sim \text{Uniform}(0, \infty) \\
\mu_\alpha \sim \text{Student-t}(1, 0, 100) \\
\mu_\beta \sim \text{Student-t}(1, 0, 10) \\
\nu_\alpha \sim \text{Gamma}(2.0, 0.1) \\
\nu_\beta \sim \text{Gamma}(2.0, 0.1) \\
\sigma_\alpha \sim \text{Uniform}(0, \infty) \\
\sigma_\beta \sim \text{Uniform}(0, \infty)
\end{gathered}$$

where $i$ and $j$ index the different individuals and SNPs; $Y$ is the phenotype of each individual; $X$ is the indicator variable for the presence of a given genotype in individual $i$ of SNP $j$; The genotype and phenotype are linked via a Studen-t noise distribution, whose central tendency is described by the linear combination of the two parameters $\alpha$ and $\beta$ which are the inferred intercept and slope coefficients of the GLM, with standard deviation $\sigma$ and degrees of freedom $\nu$. The slope and intercept are each modelled with Student-t distributions with three parameters $\mu$, $\sigma$, and $\nu$ for the mean, standard deviation and degrees of freedom, with their vague priors defined by the last six lines of the above model. The overarching distributions of the mean parameters of the intercepts and slopes are modelled via Student-t distributions defined in 6th and 7th line of the above model. The hierarchical structure of the model makes automatic corrections for the multiple-comparison problem. The model was implemented in Stan [1].

We also implement a non-hierarchical (univariate) version of the model, in which the SNPs are treated as independent by assigning individual priors on their slope and intercept parameters as:

$$\alpha_j \sim \text{Student-t}(1, 0, 100)$$
$$\beta_j \sim \text{Student-t}(1, 0, 10)$$

the last six lines of the above model are therefore excluded from the simplified univariate model; and **ii) dichotomous traits**:

$$Y_{ij} \sim \text{Binomial}(\alpha_j + \beta_j * X_{ij}, N_{ij})$$
$$\alpha_j \sim \text{Student-t}(\nu_\alpha, \mu_\alpha, \sigma_\alpha)$$
$$\beta_j \sim \text{Student-t}(\nu_\beta, \mu_\beta, \sigma_\beta)$$
$$\mu_\alpha \sim \text{Student-t}(1, 0, 100)$$
$$\mu_\beta \sim \text{Student-t}(1, 0, 10)$$
$$\nu_\alpha \sim \text{Gamma}(2.0, 0.1)$$
$$\nu_\beta \sim \text{Gamma}(2.0, 0.1)$$
$$\sigma_\alpha \sim \text{Uniform}(0, \infty)$$
$$\sigma_\beta \sim \text{Uniform}(0, \infty)$$

where $i$ and $j$ index the different genotypes and SNPs; $Y$ is the number of individuals with genotype $i$ having the reference phenotype (e.g. phenotype=1); $N$ is the count of all individuals with genotype $i$; $X$ is the indicator variable for the presence of genotype $i$ in SNP $j$; The genotype and phenotype are linked with a Binomial noise distribution whose proportion parameter is described by the linear combination of the two parameters $\alpha$

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

and $\beta$, which are the inferred intercept and slope coefficients of the GLM. The slope and intercept are each modelled with a Student-t distribution with three parameters $\mu$, $\sigma$, and $\nu$ for the mean, standard deviation and degrees of freedom, with their vague priors defined by the last six lines of the above model. The overarching distributions of the mean parameters of the intercepts and slopes are modelled via Student-t distributions defined in 4th and 5th line of the above model. The hierarchical structure of the model automatically makes automatic corrections for the multiple-comparison problem. The model was implemented in Stan.

We also implement a non-hierarchical (univariate) version of the model, in which the SNPs are treated as independent by assigning individual priors on their slope and intercept parameters as:

$$\alpha_j \sim \text{Student-t}(1, 0, 100)$$
$$\beta_j \sim \text{Student-t}(1, 0, 10)$$

the last six lines of the above model are therefore excluded from the simplified univariate model.

We summarize each association using the mean of its slope coefficient ($\beta$) 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.

Potential scale reduction factor (PSRF), number of effective samples (Neff) and information about divergences during the MCMC sampling are used to check for a successful convergence and are presented to the user. Built-in poterior predictive checks are also run to test the validity of our models.

**Bhattacharyya coefficient ($BC$)** The posterior distributions of the infered parameters of the GLMs introduced above, can be used to perform predictions, i.e. to simulate distributions of phenotype under the different genetic states of each SNP. The simulated data for each genotype can then be used to estimate the degree of overlap between the simulated phenotype distributions in two genetic states of the SNPs. We quantified the overlap using the Bhattacharyya coefficient ($BC$):

$$BC(p_1, p_2) = \int_x \sqrt{(p_1(x) \cdot p_2(x))} dx$$

where $p_1$ and $p_2$ are the simulated phenotype distributions in both genetic states of a SNP. For a complete overlap $BC = 1$ (i.e. no difference between the phenotype distributions in the two genetic state), and $BC = 0$ for no overlap (significant difference).

### 2.3 Retrospective power analysis

Assuming the true effect (e.g. of a SNP) is decently explained by the effect estimated from the data, retrospective power analysis (contentious idea [2]) (RPA) can be conducted to compute the statistical power of each SNP. `genphen` implements the following procedure for RPA:

1. Draw parameter estimates from the posterior probabilities and use them to generate new data (simulated genotype), with an equal sample size as the input
2. Analyze the simulated data with the univariate version of the models introduced in the methods section 2, and summarize each association according to the mean of the inferred slope coefficient ($\beta$) and the appropriate HDI (e.g. 95% HDI)
3. Repeat steps 1-2 multiple times, after which we estimate the following statistics for each SNP:
    - Power-error - Ratio of RPA iterations in which a statistically insignificant effect, i.e. slope coefficient with HDI containing the the null effect. To estimate the power of a SNP: Power = 1 - Power-error
    - Sign-error - Ratio of RPA runs in which the slope coefficient has a different sign compared to the originally estimated slope
    - slope-mean - The mean of the slope coefficients computed during the RPA
    - slope-sd - The standard deviation of the slope coefficients computed during the RPA

### 2.4 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 ($N \times N$ dissimilarity matrix for 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

[2] Gelman, Andrew, and John Carlin. "Beyond power calculations: Assessing type S (sign) and type M (magnitude) errors." Perspectives on Psychological Science 9.6 (2014): 641-651.

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 *continuous* 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 7 amino acids (positions), and a continuous phenotype measured for each individual.

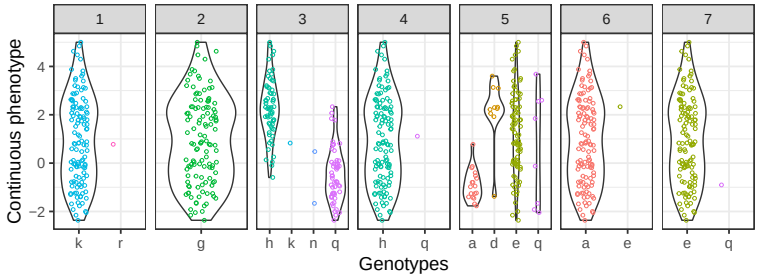
```
> require(genphen)
> require(ggplot2)
> require(knitr)
> require(ggrepel)
> require(reshape2)
> require(ape)
> require(xtable)
> options(xtable.floating = FALSE)

> # 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 7 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[, 80:86],
+                  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 = 1, scales = "free_x")+
+   geom_violin(aes(x = genotype, y = phenotype))+
+   ylab(label = "Continuous 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 continuous 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 continuous phenotypes with `genphen` using the following settings:

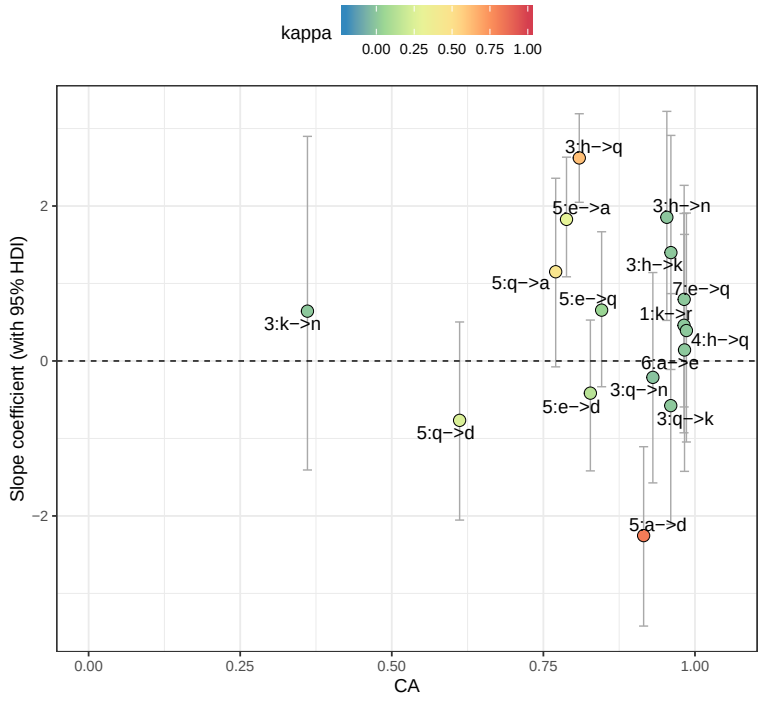
- hierarchical Bayesian model will be run with 2 MCMC chains composed of 2500 iterations each, including 500 warmup iterations.
- Random forest was selected as for the statistical learning, which will be run in a cross-validation mode with 300 iterations.
- All estimates will be reported according to their mean and 95% HDI
- Retrospective power analysis with 10 iterations will be performed.
- The resulting stan object will be returned for debugging.
- Whenever possible, 2 cores will be used.

```
> # Run genphen
> c.out <- genphen::runGenphen(genotype = genotype.saap[, 80:86],
+                              phenotype = phenotype.saap,
+                              phenotype.type = "continuous",
+                              model.type = "hierarchical",
+                              mcmc.chains = 2,
+                              mcmc.iterations = 1500,
+                              mcmc.warmup = 500,
+                              cores = 2,
+                              hdi.level = 0.95,
+                              stat.learn.method = "rf",
+                              cv.iterations = 300,
+                              rpa.iterations = 10,
+                              with.stan.obj = TRUE)
```

Typical way of visualizing the `genphen` results is with the following plot, where each point represents a polymorphism (here SAAP) plotted according to x = classification accuracy ($CA$), y = slope ($\beta$), color = Cohen's $\kappa$. The

most promising SAAPs have $CA$ and $\kappa$ close to 1, with a non-null $\beta$, i.e. $\beta$ with 95% HDI that does not overlap with 0 (shown as a dashed line in the figure). The labels show the SNP numbers (sites in the genotype data) and the type of the polymorphism (e.g. nucleotide mutation T->A).

```
> # 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$site, ":", c.score$g1,
+                        "->", c.score$g0, sep = '')
> # Visualization
> g <- ggplot(data = c.score)+
+   geom_errorbar(aes(x = ca, ymin = beta.L, ymax = beta.H),
+                 width = 0.015, col = "darkgray")+
+   geom_point(aes(x = ca, y = beta.mean, fill = kappa), shape = 21, size = 4)+
+   geom_text_repel(aes(x = ca, y = beta.mean, label = label), size = 5)+
+   theme_bw(base_size = 14)+
+   ylab(label = "Slope coefficient (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$beta.hdi <- apply(X = c.score[, c("beta.L", "beta.H")],
+                           MARGIN = 1, getHdiPretty, digits = 2)
> c.score$ca.hdi <- apply(X = c.score[, c("ca.L", "ca.H")],
+                           MARGIN = 1, getHdiPretty, digits = 2)
> c.score$kappa.hdi <- apply(X = c.score[, c("kappa.L", "kappa.H")],
+                           MARGIN = 1, getHdiPretty, digits = 2)
> # Print table
> print(xtable(c.score[, c("label", "beta.mean", "beta.hdi", "ca",
+                          "ca.hdi", "kappa", "kappa.hdi", "bc"), ],
+              align = rep(x = "c", times = 9, digits = 2)),
+       include.rownames = FALSE, size = "scriptsize")
```

| label | beta.mean | beta.hdi | ca | ca.hdi | kappa | kappa.hdi | bc |
|---|---|---|---|---|---|---|---|
| 1:k->r | 0.46 | (-0.93, 1.9) | 0.98 | (0.95, 1) | 0.00 | (0, 0) | 0.99 |
| 3:h->k | 1.40 | (-0.11, 2.91) | 0.96 | (0.9, 1) | 0.00 | (0, 0) | 0.91 |
| 3:h->n | 1.85 | (0.52, 3.22) | 0.95 | (0.9, 1) | -0.01 | (-0.04, 0) | 0.85 |
| 3:h->q | 2.62 | (2.05, 3.19) | 0.81 | (0.74, 0.89) | 0.62 | (0.47, 0.77) | 0.71 |
| 3:k->n | 0.64 | (-1.41, 2.9) | 0.36 | (0, 1) | 0.00 | (0, 0) | 0.97 |
| 3:q->k | -0.58 | (-2.11, 0.87) | 0.96 | (0.88, 1) | 0.00 | (0, 0) | 0.98 |
| 3:q->n | -0.21 | (-1.57, 1.14) | 0.93 | (0.87, 1) | -0.01 | (-0.05, 0) | 0.99 |
| 4:h->q | 0.39 | (-1.05, 1.91) | 0.99 | (0.97, 1) | 0.00 | (0, 0) | 0.99 |
| 5:a->d | -2.25 | (-3.42, -1.11) | 0.92 | (0.75, 1) | 0.82 | (0.46, 1) | 0.78 |
| 5:e->a | 1.83 | (1.09, 2.63) | 0.79 | (0.7, 0.88) | 0.29 | (0.02, 0.56) | 0.84 |
| 5:e->d | -0.42 | (-1.42, 0.53) | 0.83 | (0.74, 0.92) | 0.13 | (-0.14, 0.39) | 0.99 |
| 5:e->q | 0.66 | (-0.33, 1.67) | 0.85 | (0.77, 0.92) | 0.04 | (-0.13, 0.3) | 0.97 |
| 5:q->a | 1.15 | (-0.08, 2.36) | 0.77 | (0.54, 0.93) | 0.46 | (-0.13, 0.84) | 0.93 |
| 5:q->d | -0.77 | (-2.05, 0.5) | 0.61 | (0.33, 0.82) | 0.24 | (-0.25, 0.62) | 0.96 |
| 6:a->e | 0.14 | (-1.42, 1.63) | 0.98 | (0.94, 1) | 0.00 | (0, 0) | 0.99 |
| 7:e->q | 0.80 | (-0.59, 2.27) | 0.98 | (0.94, 1) | 0.00 | (0, 0) | 0.97 |

**MCMC convergence** Next, we want to check the validity our Bayesian inference model 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/).

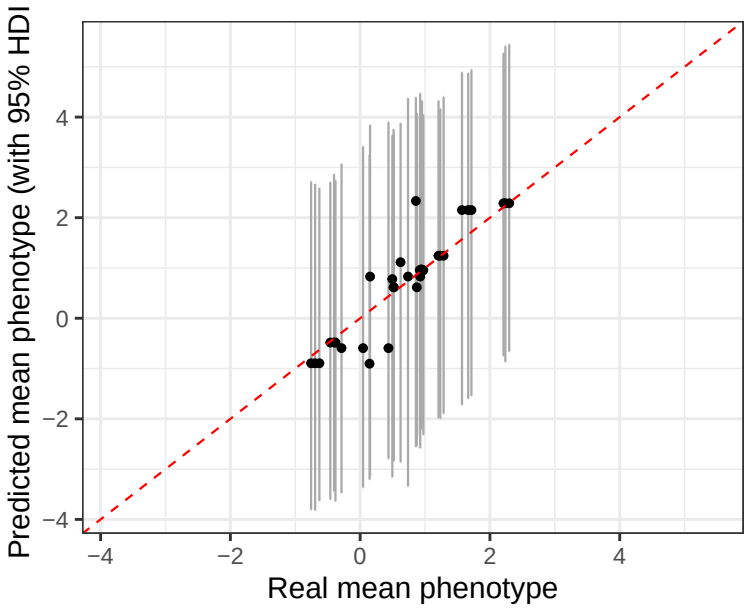
```
> # Get the convergence output
> c.convergence <- c.out$convergence
> c.convergence$label <- paste(c.convergence$site, ":",
+                              c.convergence$g1, "->",
+                              c.convergence$g0, sep = '')
> c.convergence$label[c.convergence$site == ''] <- ''
> # Print table
> print(xtable(c.convergence[, c("label", "par", "Rhat", "n_eff")],
+              align = rep(x = "c", times = 5, digits = 2)),
+       include.rownames = FALSE, size = "scriptsize")
```

| label | par | Rhat | n_eff |
|---|---|---|---|
| | sigma | 1.00 | 2000.00 |
| | nu | 1.00 | 2000.00 |
| | mu_alpha | 1.00 | 1150.65 |
| | mu_beta | 1.00 | 2000.00 |
| | sd_alpha | 1.00 | 2000.00 |
| | sd_beta | 1.00 | 2000.00 |
| | nu_alpha | 1.00 | 2000.00 |
| | nu_beta | 1.00 | 2000.00 |
| 1:k->r | alpha | 1.00 | 1633.55 |
| 1:k->r | beta | 1.00 | 2000.00 |
| 3:h->q | alpha | 1.00 | 2000.00 |
| 3:h->q | beta | 1.00 | 2000.00 |
| 3:h->k | alpha | 1.00 | 2000.00 |
| 3:h->k | beta | 1.00 | 2000.00 |
| 3:h->n | alpha | 1.00 | 1631.82 |
| 3:h->n | beta | 1.00 | 1618.73 |
| 3:q->k | alpha | 1.00 | 2000.00 |
| 3:q->k | beta | 1.00 | 2000.00 |
| 3:q->n | alpha | 1.00 | 2000.00 |
| 3:q->n | beta | 1.00 | 2000.00 |
| 3:k->n | alpha | 1.00 | 2000.00 |
| 3:k->n | beta | 1.00 | 2000.00 |
| 4:h->q | alpha | 1.00 | 1847.44 |
| 4:h->q | beta | 1.00 | 1790.37 |
| 5:e->q | alpha | 1.00 | 1643.90 |
| 5:e->q | beta | 1.00 | 1611.05 |
| 5:e->a | alpha | 1.00 | 1730.12 |
| 5:e->a | beta | 1.00 | 1907.45 |
| 5:e->d | alpha | 1.00 | 2000.00 |
| 5:e->d | beta | 1.00 | 2000.00 |
| 5:q->a | alpha | 1.00 | 2000.00 |
| 5:q->a | beta | 1.00 | 2000.00 |
| 5:q->d | alpha | 1.00 | 2000.00 |
| 5:q->d | beta | 1.00 | 2000.00 |
| 5:a->d | alpha | 1.00 | 1550.50 |
| 5:a->d | beta | 1.00 | 1398.07 |
| 6:a->e | alpha | 1.00 | 2000.00 |
| 6:a->e | beta | 1.00 | 2000.00 |
| 7:e->q | alpha | 1.00 | 1592.34 |
| 7:e->q | beta | 1.00 | 1556.79 |

**MCMC sampling issues** Please refer to the Stan documentation to understand the content of the stan object (mc-stan.org/users/documentation/).

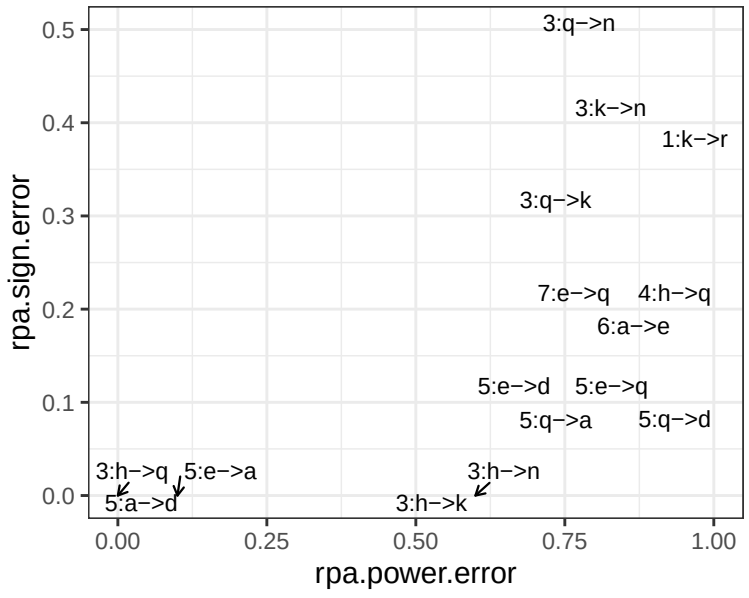
```
> # We can visualize the different MCMC properties as follows:
> s1 <- rstan::stan_diag(object = c.out$stan.obj, information = "sample")
> s2 <- rstan::stan_diag(object = c.out$stan.obj, information = "stepsize")
> s3 <- rstan::stan_diag(object = c.out$stan.obj, information = "treedepth")
> s4 <- rstan::stan_diag(object = c.out$stan.obj, information = "divergence")
```

**Posterior predictive checks** To test the validity of the Bayesian models, we perform posterior predictive checks by predicting new data using the inferred parameters for each SNPs. This is done automatically by `genphen`. We can evaluate the goodness of the model by comparing the predicted and observed data.




### Retrospective power analysis

```
> # Get the RPA output
> c.rpa <- c.out$rpa
> c.rpa$label <- paste(c.rpa$site, ":", c.rpa$g1, "->", c.rpa$g0, sep = '')
> g <- ggplot(data = c.rpa)+
+   geom_text_repel(aes(x = rpa.power.error, y = rpa.sign.error, label = label),
+                   arrow = arrow(length = unit(0.02, "npc")), size = 4)+
+   theme_bw(base_size = 14)
> g
```



```
> print(xtable(c.rpa[, c("label",  "rpa.power.error", "rpa.sign.error",
+                        "rpa.beta.mean", "rpa.beta.sd", "rpa.N")],
+              align = rep(x = "c", times = 7, digits = 2)),
+       include.rownames = FALSE, size = "small")
```

| label | rpa.power.error | rpa.sign.error | rpa.beta.mean | rpa.beta.sd | rpa.N |
|---|---|---|---|---|---|
| 1:k->r | 1.00 | 0.40 | 0.41 | 1.40 | 10 |
| 3:h->q | 0.00 | 0.00 | 2.36 | 0.37 | 10 |
| 3:h->k | 0.60 | 0.00 | 2.69 | 1.54 | 10 |
| 3:h->n | 0.60 | 0.00 | 2.58 | 1.78 | 10 |
| 3:q->k | 0.70 | 0.30 | -1.05 | 2.55 | 10 |
| 3:q->n | 0.70 | 0.50 | 0.02 | 2.34 | 10 |
| 3:k->n | 0.90 | 0.40 | 0.02 | 2.41 | 10 |
| 4:h->q | 0.90 | 0.20 | 0.74 | 1.28 | 10 |
| 5:e->q | 0.80 | 0.10 | 0.55 | 0.74 | 10 |
| 5:e->a | 0.10 | 0.00 | 1.54 | 0.52 | 10 |
| 5:e->d | 0.70 | 0.10 | -0.55 | 0.56 | 10 |
| 5:q->a | 0.70 | 0.10 | 0.98 | 0.81 | 10 |
| 5:q->d | 0.90 | 0.10 | -0.91 | 0.76 | 10 |
| 5:a->d | 0.10 | 0.00 | -2.51 | 0.82 | 10 |
| 6:a->e | 0.90 | 0.20 | 0.80 | 1.94 | 10 |
| 7:e->q | 0.80 | 0.20 | 1.13 | 2.02 | 10 |

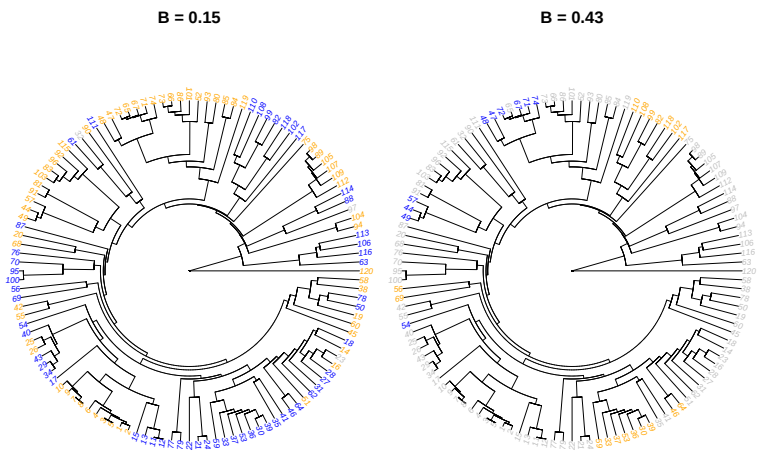
**Phylogenetic bias control** Next, we compute the phylogenetic bias of each mutation, shown in the table below:

```
> # 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 - 79
> mutation.bias <- merge(x = c.score, y = mutation.bias,
+                        by = c("site", "g1", "g0"))
> # Show the bias table
> print(xtable(mutation.bias[, c("site", "g1", "g0", "bias.g1", "bias.g0")],
+              align = rep(x = "c", times = 6, digits = 2)),
+       include.rownames = FALSE, size = "small")
```

| site | g1 | g0 | bias.g1 | bias.g0 |
|---|---|---|---|---|
| 1 | k | r | 0.01 | 1.00 |
| 3 | h | k | 0.04 | 1.00 |
| 3 | h | n | 0.04 | 0.43 |
| 3 | h | q | 0.04 | 0.15 |
| 3 | k | n | 1.00 | 0.43 |
| 3 | q | k | 0.15 | 1.00 |
| 3 | q | n | 0.15 | 0.43 |
| 4 | h | q | 0.01 | 1.00 |
| 5 | a | d | 0.29 | 0.43 |
| 5 | e | a | 0.04 | 0.29 |
| 5 | e | d | 0.04 | 0.43 |
| 5 | e | q | 0.04 | 0.19 |
| 5 | q | a | 0.19 | 0.29 |
| 5 | q | d | 0.19 | 0.43 |
| 6 | a | e | 0.01 | 1.00 |
| 7 | e | q | 0.00 | 1.00 |

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 a *dichotomous* phenotype

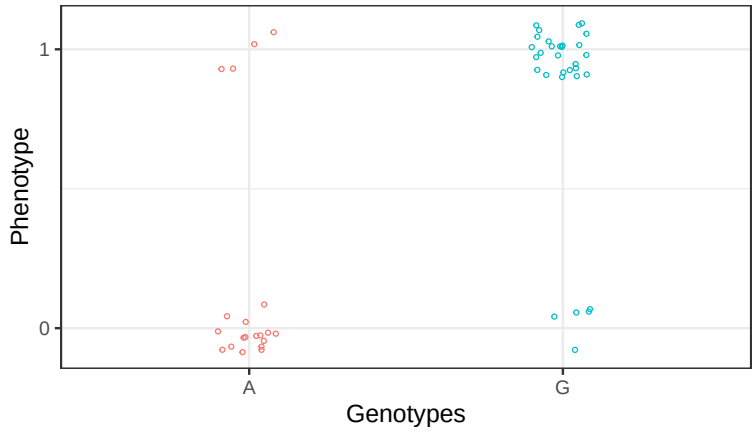
In the second case study we show you how to use `genphen` in case of dichotomous phenotypes. Here, the genotype is a single SNP in a study of 51 individuals, while the phenotype is a vector of 51 dichotomous measurements for each individual. First we show an overview of the distribution of the phenotype in the two genotypes of the studied SNP.

```
> # Genotype inputs:
> data(genotype.snp)
> # Select a single SNP
> genotype.snp <- genotype.snp[, 10]
> # Generate phenotype inputs:
```

```
> dichotomous.phenotype.snp <- c(rbinom(n = 20, size = 1, prob = 0.25),
+                                rbinom(n = 31, size = 1, prob = 0.75))

> # Format the genotype-phenotype data, such that it can then
> # be visualized with ggplot
> df <- data.frame(genotype.snp,
+                  phenotype = dichotomous.phenotype.snp,
+                  stringsAsFactors = FALSE)
> df <- melt(data = df, id.vars = "phenotype")
> colnames(df) <- c("phenotype", "site", "genotype")

> # Visualization
> g <- ggplot(data = df)+
+   geom_point(aes(x = genotype, y = phenotype, col = genotype), size = 1,
+              shape = 21, position = position_jitterdodge(jitter.width = 0.2,
+                                                          jitter.height = 0.1,
+                                                          dodge.width = 0.5))+
+   xlab(label = "Genotypes")+
+   scale_y_continuous(name = "Phenotype", breaks = c(0, 1), labels = c(0, 1))+
+   theme_bw(base_size = 14)+
+   theme(legend.position = "none")
> g
```




**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** Next, we perform the genetic association study for dichotomous phenotypes with `genphen` using the following settings:

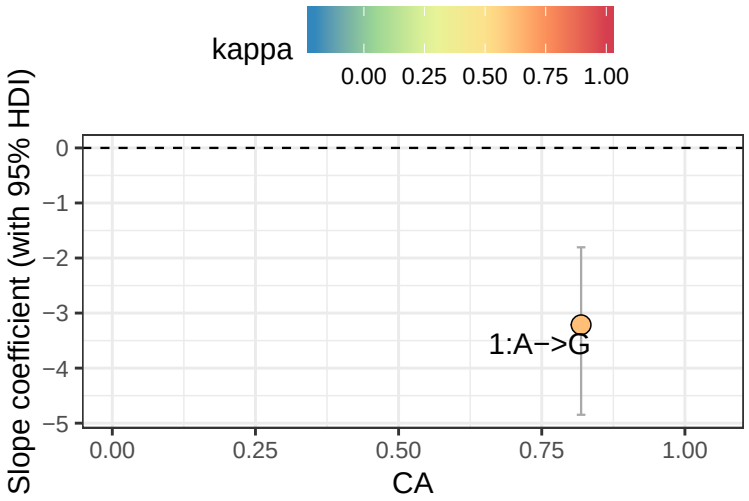
- univariate Bayesian model will be run with 2 MCMC chains composed of 2500 iterations each, including 500 warmup iterations.

- Random forest was selected as for the statistical learning, which will be run in a cross-validation mode with 500 iterations.
- All estimates will be reported according to their mean and 95% HDI
- No retrospective power analysis.
- No stan object.
- Whenever possible, 2 cores will be used.

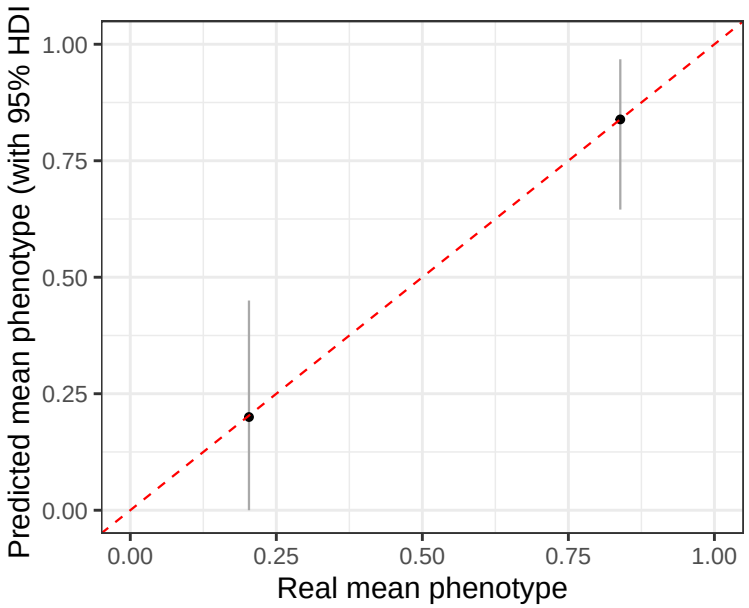
```
> # run genphen
> d.out <- genphen::runGenphen(genotype = genotype.snp,
+                              phenotype = dichotomous.phenotype.snp,
+                              phenotype.type = "dichotomous",
+                              model.type = "univariate",
+                              mcmc.chains = 2,
+                              mcmc.iterations = 2500,
+                              mcmc.warmup = 500,
+                              cores = 2,
+                              hdi.level = 0.95,
+                              stat.learn.method = "rf",
+                              cv.iterations = 500,
+                              rpa.iterations = 0,
+                              with.stan.obj = FALSE)
```

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 ($\beta$) with error bars representing the 95% HDI, color = Cohen's $\kappa$.

```
> # Get scores data
> d.score <- d.out$scores
> # Some optional formatting
> d.score$label<-paste(d.score$site, ":", d.score$g1, "->", d.score$g0, sep = '')
> # Visualization
> g <- ggplot(data = d.score)+
+   geom_errorbar(aes(x = ca, ymin = beta.L, ymax = beta.H),
+                 width = 0.015, col = "darkgray")+
+   geom_point(aes(x = ca, y = beta.mean, fill = kappa), shape = 21, size = 4)+
+   geom_text_repel(aes(x = ca, y = beta.mean, label = label), size = 5)+
+   theme_bw(base_size = 14)+
+   ylab(label = "Slope coefficient (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
```




**Posterior predictive checks** We again evaluate the goodness of our model by comparing the predicted and observed data.



# 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. We use the complete genotype and phenotype data to train a model using RF, quantifying the association between each SNP and the phenotype using the RF's variable importance measure (impurity).
2. Next, we rank the SNPs according to their their importance. One can study the distribution of variable importances to get an insight into the structure of the importances values and potentially detect outlying patterns.
3. We create a set of 'diagnostic points', i.e. SNPs with specific ranks in terms of their importance value, which will next be evaluated with a the standard `genphen` approach.
4. By plotting the estimated association scores as a function of the rank of importance, we can roughly determine the importance rank at which the SNPs no longer carry any signal of association, and discard that portian from the data before conducting the main analysis using the standard `genphen` method.

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.

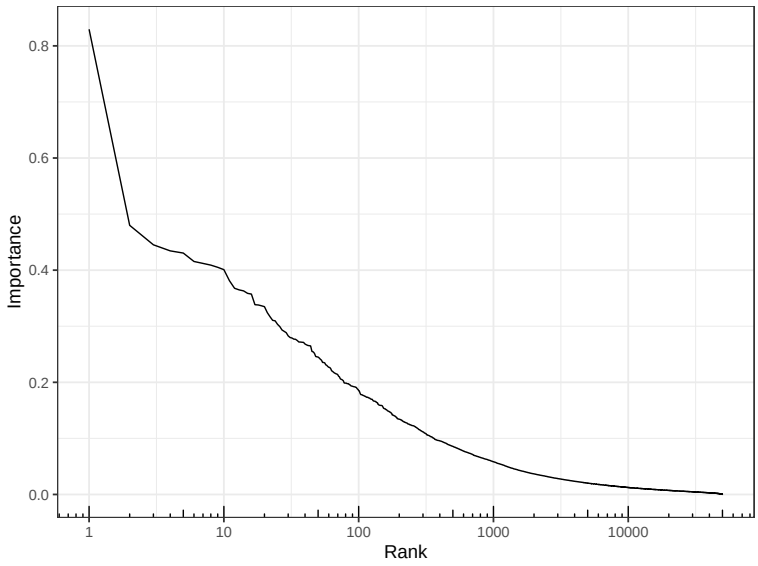
```
> # Simulate 50,000 SNPs and 60 phenotypes
> set.seed(seed = 551155)
> g1 <- replicate(n=5*10^4, expr=as.character(rbinom(n=30, size = 1,prob = 0.49)))
> g2 <- replicate(n=5*10^4, expr=as.character(rbinom(n=30, size = 1,prob = 0.51)))
> gen <- rbind(g1, g2)
> phen <- c(rnorm(n = 30, mean = 3, sd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

We then select a set of 41 'diagnostic points' (SNPs) in range 1-50,000 and run the diagnosis. One can also decide not to analyze any 'diagnostic points', in which case only the variable importance will be evaluated and returned. The typical runtime for the provided dataset ($60 \times 50,000$) is few minutes.

```
> # Set diagnostic points (in range 1-50,000)
> diagnostic.points <- c(seq(from = 0, to = 100, by = 20),
+                        seq(from = 200, to = 1000, by = 200),
+                        seq(from = 2000, to = 50000, by = 3000))
> diagnostic.points[1] <- 1
> # Run diagnostics
> diag <- genphen::runDiagnostics(genotype = gen,
+                                 phenotype = phen,
+                                 phenotype.type = "continuous",
+                                 rf.trees = 50000,
+                                 mcmc.chains = 2,
+                                 mcmc.iterations = 2500,
+                                 mcmc.warmup = 500,
+                                 cores = 2,
+                                 hdi.level = 0.95,
+                                 diagnostic.points = diagnostic.points)
> # Get diagnosis scores, and mark effects as significant|not-significant
> diag.stats <- diag$scores
> diag.stats$significant <- ifelse(test = diag.stats$beta.L <= 0
+                                  & diag.stats$beta.H >= 0,
+                                  yes = FALSE, no = TRUE)
```

We can evaluate the distribution of importance scores, and include additional 'diagnostic points' to be analyzed.

```
> # Visualization
> g <- ggplot(data = diag$importance.scores)+
+   geom_line(aes(x = importance.rank, y = importance))+
+   xlab("Rank")+
+   ylab("Importance")+
+   theme_bw(base_size = 14)+
+   scale_x_continuous(trans = "log10")+
+   annotation_logticks(base = 10, sides = "b")
> g
```



By visualizing the estimated slope coefficients of the diagnosed SNPs, we can observe an enrichment of non-informative (statistically not-significant) SNPs beyond the rank of 1000. We can thus narrow down our interval of interest to the top-ranked 2500 SNPs, yielding massive data reduction of 95%, while still retaining many SNPs with small effects.

```
> # Visualization
> g <- ggplot(data = diag.stats)+
+   geom_errorbar(aes(x = diagnostic.points, ymin = beta.L, ymax = beta.H),
+                 col = "gray", width = 0.05)+
+   geom_point(aes(x = diagnostic.points, y = beta.mean, col = significant),
+              size = 2)+
+   geom_hline(yintercept = 0, linetype = "dashed", col = "gray")+
+   theme_bw(base_size = 14)+
+   xlab(label = "Rank")+
+   ylab(label = "Slope coefficient (with 95% HDI)")+
+   scale_x_continuous(trans = "log10",
+                      breaks = c(1, 10, 100, 10^3, 10^4,5*10^4),
+                      labels = c(1, 10, 100, 10^3, 10^4, 5*10^4))+
+   annotation_logticks(sides = "b")+
+   theme(legend.position = "top")
> g
```