PPBstats

An R package for statistical analysis of unbalanced trials in decentralized participatory plant breeding programmes

version 0.11

September 16, 2016

Pierre Rivière[1,2] Olivier David[3]

[1] Réseau Semences Paysannes, 3 avenue de la gare, F-47190 Aiguillon, France
[2] INRA, UMR 0320, Génétique Quantitative et Evolution, Ferme du Moulon F-91190 Gif sur Yvette, France
[3] INRA, UR 1404 Unité Mathématiques et Informatique Appliquées du Génome à l'Environnement, F-78352 Jouy-en-Josas, France
Contact: pierre@semencespaysannes.org
Contributions: P. Rivière worte the R functions and the vignette, O. David wrote the JAGS code and review the R code and the vignette.

Le Réseau Semences Paysannes (the French Farmers' Seeds Network (RSP)), created in 2003, brings together a great diversity of collectives and people who preserve farmers' seeds in fields, orchards, vineyards and gardens. They are involved in supporting the consolidation of local initiatives to maintain and renew cultivated biodiversity through Community Seeds Systems. Over 80 organizations have come together to promote and develop farmers' seeds, and to protect farmers' rights over their seeds.
www.semencespaysannes.org (in french).

The Diversity, Evolution and Adaptation of Populations (DEAP) team led by Isabelle Goldringer is part of INRA UMR 0320 Quantitative Genetic and Evolution. Its work is based on the analysis of the genetic and evolutionary mechanisms underlying evolution and adaptation of crop populations. DEAP develops strategies for on farm management of crop genetic diversity and for plant breeding (evolutionary and/or participatory) adated to organic and low input agriculture. Assessing the benefits of in-field genetic diversity (variety mixtures, populations) and designing / breeding optimized mixtures adapted to local conditions are also key research objectives.
http://moulon.inra.fr/index.php/en/team/deap

The INRA UR1404 MaIAGE research laboratory gathers mathematicians, computer scientists, bioinformaticians and biologists to tackle problems coming from biology, agronomy and ecology; The addressed questions may concern processes at very different levels: molecular, cellular or multicellular, individual, populations, ecosystems oy landscapes. MaIAGE develops original methods in mathematics, statistics and computer science which are generic or driven by specific biological problems. A particular attention is paid to develop and make available softwares, databases, ontologies and web services so that biologists can use them easily to analyze their data or to mine the scientific literature.
http://maiage.jouy.inra.fr/?q=en/home

Contents

Wheat trials on farm within our participatory plant breeding programme, summer 2012, Auvergne, France. CC-BY-NC-SA. Pierre Rivière.

1 Philosophy of PPBstats

PPBstats aims to facilitate the implementation of statistical methods described in Rivière et al. (2015) and ?. These methods aim to analyse highly unbalanced datasets that can be found in decentralized participatory plant breeding (PPB).

It has been developped for dataset with two types of farms, regional and satellite, based on their experimental design.

Regional farms had several populations (i.e. a germplasm in an environment) in two or more blocks with populations replicated in each block. Satellite farms had no block and one germplasm replicated twice. Farmers chose the other populations to be sown that were not replicated (Figure 1). The number of populations may vary between farms.

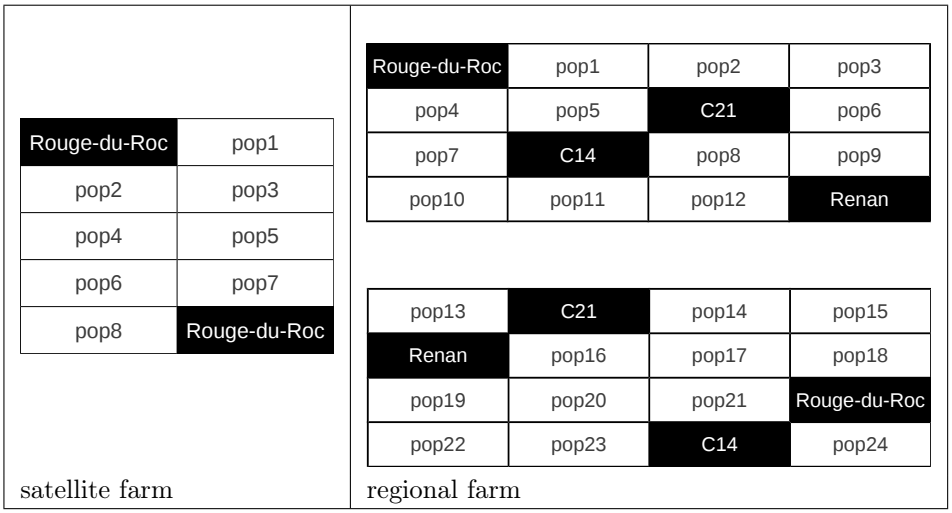


Figure 1: Experimental design of satellite and regional farms. Controls replicated are in black boxes : *Rouge-du-Roc*, *Renan*, $C14$ and $C21$. pop stands for 'tested population'.

- At the **farm level**, the residual had few degrees of freedom, leading to a poor estimation of the residual variance and to a lack of power for comparing populations. Hence, model 1 was implemented (section 2).
- At the **network level**, there is a large germplasm $\times$ environment combinaisons that are missing, leading to a poor estimation of germplasm, environment and interaction effects. Hence, model 2 was implemented (section 3).

These methods are of interest if you have a large data set with a high number of environments and germplasms. For model 1, it gave nice results with more than 20 environment (Rivière et al., 2015). For model 2, it gave nice results with 75 environments and 120 germplasms present in at least two environments (95% of missing $G \times E$ combinaisons) (Rivière et al., 2016).

This will be further explore in simulation studies.

1.1 Function relations in PPBstats

PPBstats is divided into two sets of functions:

- Hidden functions
- Used functions

In this vignette, we only used examples with used functions. Nevertheless, hidden functions could be used in other context to answer specific questions. Figure 2 displays these functions and their relations. Table 1 gives a quick description of each function. You can have more information for each function by typing `?function_name` in your R session.

function name	description
`MC`	Run model 1 to get mean comparisons (MC) on each environment of the network.
`FWH`	Run model 2 to get main germplasm, environment and sensitivity effects over the network.
`analyse.outputs`	Check with plots if the model went well based ont the Gelman-Rubin test and plots of posteriors distributions (see section 1.2). It is important to run this step before going ahead in the analysis otherwise you may make mistakes in the interpretation of the results
`get.mean.comparisons`	Get mean comparisons for a given parameter two by two or to a given threshold based on MCMC outputs
`get.parameter.groups`	Get groups of parameters based on multivariate analysis
`cross.validation.FWH`	Run complete cross vlidation with model 2
`predict.the.past`	Estimate value of a germplasm in an environment based on the FWH model.
`get.ggplot`	Get ggplot objects to visualize output from the analysis
`get.env.info`	Get regional farms data and satellite farms data
`comp.parameters`	Get parameter comparisons two by two or to a given threshold based on MCMC outputs
`get.significant.groups`	Get significant groups of differences for a set of parameters based on MCMC outputs
`get.at.least.X.groups`	Get the value of type one error needed to have X groups.

Table 1: Function descriptions in `PPBstats`.

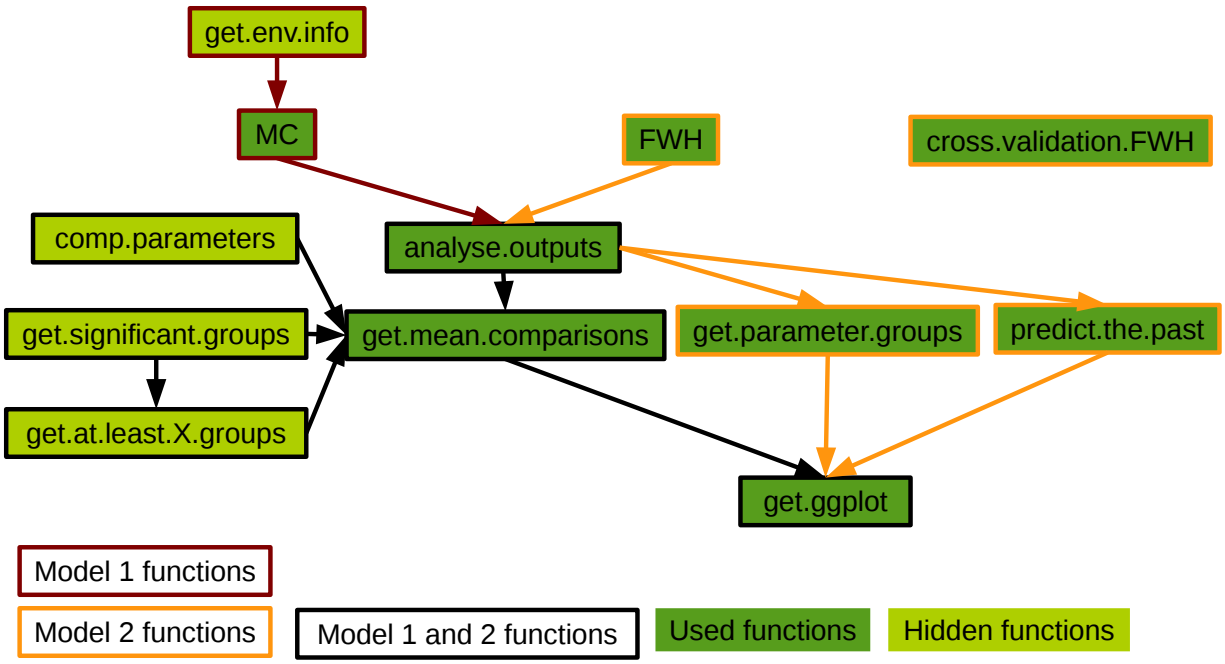


Figure 2: Function relations in `PPBstats`. Functions related to model 1 are in red. Functions related to model 2 are in orange. Functions related to both models are in black.

1.2 Bayesian statistics

The analyses performed in `PPBstats` are based on Bayesian statistics.

Bayesian statistics are based on the Bayes theorem:

$$Pr(\theta|y) \propto Pr(\theta)Pr(y|\theta)$$

with $Pr(\theta|y)$ the posterior, $Pr(y|\theta)$ the likelihood and $Pr(\theta)$ the prior.

The parameters' distribution, knowing the data (the posterior), is proportional to the distribution *a priori* (the prior) $\times$ the information brought by the data (the likelihood).

The more information (i.e. the larger the data set and the better the model fits the data), the less the prior would be of importance. If the priors equal the posteriors, it means that there is not enough data or the model does not fit the data.

Bayesian inference is based on the posterior distribution of model parameters. This distribution could not be calculated explicitely for the hierarchical model used in here (see section 2 and section 3) but could be estimated using Markov Chain and Monte Carlo (MCMC) methods.

These methods simulate values of model parameters according to a Markov chain that converges to the posterior distribution of model parameters (Robert, 2001).

MCMC methods were implemented using `JAGS` by the `R` package `rjags` that performed Gibbs sampling (Robert, 2001). Two MCMC chains were run independently to test for convergence using the Gelman-Rubin test. This test was based on the variance within and between the chains (Gelman and Rubin, 1992).

A burn-in and lots of iterations were needed in the MCMC procedure. In our case, the burn-in had 1000 iterations, then 100 000 iterations are done by default[1] with a thinning interval of 10 to reduce autocorrelations between samples, so that 10 000 samples were available for inference for each chain by default[2].

The final distribution of a posterior is the concatenation of the two MCMC chains: 20 000 samples.

1.3 Let's go!

To continue, load the package:

```
library(PPBstats)
```

and download from internet the data used in this vignette (this is useful to earn lots of time!) here : https://www.dropbox.com/sh/6qvl515k5484zg4/AADZKkaM2XZvmr9e6l5aWxN2a?dl=0 and put it in the folder `data_PPBstats` and then load() it.

2 At the farm level : model 1 to perform mean comparisons on farms

2.1 The model

We restricted ourselves to analysing plot means. The phenotypic value Y_{ijk} for variable Y, germplasm i, environment j and block k was modelled as :

$$Y_{ijk} = \mu_{ij} + \beta_{jk} + \varepsilon_{ijk}; \quad \varepsilon_{ijk} \sim \mathcal{N}(0, \sigma_j^2), \tag{1}$$

where μ_{ij} was the mean of germplasm i in environment j (note that this parameter, which corresponds to an entry, confounds the population effect and the population $\times$ environment effect); β_{jk} was the effect of block k in environment j satisfying the constraint[3] $\sum_{k=1}^{K} \beta_{jk} = 1$; ε_{ijk} was the residual error; $\mathcal{N}(0, \sigma_j^2)$ denoted normal distribution centred on 0 with variance σ_j^2, which was specific to environment j.

We took advantage of the similar structure of the trials on each environment of the network to assume that trial residual variances came from a common distribution :

$$\sigma_j^2 \sim \frac{1}{Gamma(\nu, \rho)},$$

[1] You can change it with the argument `nb_iterations` in functions `MC` and `FWH`

[2] There are `nb_iterations`/10 values for each chain. This can be changed with the `thin` argument of the functions.

[3] Note that it is quite different from Rivière et al. (2015) where the model was done only for two blocks. Here there is no restriction on the number of blocks.

where ν and ρ are unknown parameters. Because of the low number of residual degrees of freedom for each farm, we used a hierarchical approach in order to assess mean differences on farm. For that, we placed vague prior distributions on the hyperparameters ν and ρ :

$$\nu \sim Uniform(\nu_{min}, \nu_{max}); \quad \rho \sim Gamma(10^{-6}, 10^{-6}).$$

In other words, the residual variance of a trial within environment was estimated using all the informations available on the network rather than using the data from that particular trial only.

The parameters μ_{ij} and β_{j1} were assumed to follow vague prior distributions :

$$\mu_{ij} \sim \mathcal{N}(\mu_{.j}, 10^6); \quad \beta_{j1} \sim \mathcal{N}(0, 10^6).$$

The inverse gamma distribution has a support bounded by 0 (consistent with the definition of a variance) and may have various shapes including asymmetric distributions. From an agronomical point of view, the assumption that trial variances were heterogeneous was consistent with organic farming: there were as many environments as farmers leading to a high heterogeneity. Environment was here considered in a broad sense: practices (sowing date, sowing density, tilling, etc.), pedo climatic conditions, biotic and abiotic stress, ... (Desclaux et al., 2008). Moreover, the inverse gamma distribution had conjugate properties that facilitated MCMC convergence. This model was therefore a good choice based on both agronomic and statistical criteria.

The residual variance estimated from the controls was assumed to be representative of the residual variance of the other entries. Blocks were included in the model only if the trial had blocks.

2.2 With `PPBstats`

For model 1, you can follow these steps (Figure 2):

1. Run the model with `MC`
2. Analyse model outputs with graphs to know if you can continue the analysis with `analyse.outputs`
3. Get mean comparisons for each factor with `get.mean.comparisons` and `get.ggplot`

Let's get the data. The values for μ_{ij}, β_{jk}, ϵ_{ijk} and σ_j are the real value taken to create the dataset. This dataset is representative of data you can get in a PPB programme.

```
data(PPBdata)
head(PPBdata)
```

```
##   year location germplasm block X Y      tkw    mu_ij beta_jk
## 1 2010   env1-1     tem-1     1 1 a 72.09900 73.37224       0
## 2 2010   env1-1     tem-2     1 2 b 61.05274 61.61823       0
## 3 2010   env1-1     tem-3     1 3 c 62.99350 64.31830       0
## 4 2010   env1-1     tem-4     1 4 d 65.10909 62.57840       0
## 5 2010   env1-1     tem-1     2 5 e 77.01361 73.37224       0
## 6 2010   env1-1     tem-2     2 6 f 64.10541 61.61823       0
##   epsilon_ijk  sigma_j
## 1  -1.2732421 1.622339
## 2  -0.5654918 1.622339
## 3  -1.3248006 1.622339
## 4   2.5306946 1.622339
## 5   3.6413666 1.622339
## 6   2.4871807 1.622339
```

2.2.1 Run the model

To run model 1 on the dataset, used the function `MC`. You can run it on one variable. Here it is thousand kernel weight (tkw).

By default, `MC` returns posteriors for μ_{ij} (`return.mu = TRUE`), β_{jk} (`return.beta = TRUE`), σ_j (`return.sigma = TRUE`), ν (`return.nu = TRUE`) and ρ (`return.rho = TRUE`). You can also get ϵ_{ijk} value with `return.espilon = TRUE`.

By default, DIC is not displayed, you may want this value to compare to other model (`DIC = TRUE`). DIC criterion is a generalization of the AIC criterion that can be used for hierarchical models (Spiegelhalter et al., 2002). The smaller the DIC value, the better the model (Plummer, 2008).

```
# out.model1 = MC(data = PPBdata, variable = "tkw", return.epsilon = TRUE)
#Compiling model graph
#   Resolving undeclared variables
#   Allocating nodes
#   Graph Size: 7662
#
#Initializing model
#
#  |++++++++++++++++++++++++++++++++++++++++++++++++++| 100%
#  |**************************************************| 100%
#  |**************************************************| 100%
#  |**************************************************| 100%

load("./data_PPBstats/out.model1.RData") # To save time
```

You can get informations of the environments in the dataset :

```
out.model1$vec_env_with_no_data

## [1] "env4:2011"

out.model1$vec_env_with_no_controls

## [1] "env5:2010"

out.model1$vec_env_with_controls

##  [1] "env1-1:2010"  "env1-1:2011"  "env1-1:2012"  "env1-2:2010"
##  [5] "env1-2:2011"  "env1-2:2012"  "env1-3:2010"  "env1-3:2011"
##  [9] "env1-3:2012"  "env1-4:2010"  "env1-4:2011"  "env1-4:2012"
## [13] "env1-5:2011"  "env2-10:2010" "env2-10:2011" "env2-10:2012"
## [17] "env2-11:2011" "env2-11:2012" "env2-1:2010"  "env2-1:2011"
## [21] "env2-1:2012"  "env2-12:2011" "env2-12:2012" "env2-13:2011"
## [25] "env2-13:2012" "env2-14:2011" "env2-14:2012" "env2-15:2011"
## [29] "env2-15:2012" "env2-2:2010"  "env2-2:2011"  "env2-2:2012"
## [33] "env2-3:2010"  "env2-3:2011"  "env2-3:2012"  "env2-4:2010"
## [37] "env2-4:2011"  "env2-4:2012"  "env2-5:2010"  "env2-5:2011"
## [41] "env2-5:2012"  "env2-6:2010"  "env2-6:2011"  "env2-6:2012"
## [45] "env2-7:2010"  "env2-7:2011"  "env2-7:2012"  "env2-8:2010"
## [49] "env2-8:2011"  "env2-8:2012"  "env2-9:2010"  "env2-9:2011"
## [53] "env2-9:2012"  "env3-1:2011"  "env3-1:2012"  "env3-2:2011"
## [57] "env3-2:2012"  "env3-3:2011"

out.model1$vec_env_RF

##  [1] "env1-1:2010" "env1-1:2011" "env1-1:2012" "env1-2:2010"
##  [5] "env1-2:2011" "env1-2:2012" "env1-3:2010" "env1-3:2011"
##  [9] "env1-3:2012" "env1-4:2010" "env1-4:2011" "env1-4:2012"
## [13] "env1-5:2011" "env3-1:2011" "env3-1:2012" "env3-2:2011"
## [17] "env3-2:2012" "env3-3:2011"

out.model1$vec_env_SF
```

```
##  [1] "env2-10:2010" "env2-10:2011" "env2-10:2012" "env2-11:2011"
##  [5] "env2-11:2012" "env2-1:2010"  "env2-1:2011"  "env2-1:2012"
##  [9] "env2-12:2011" "env2-12:2012" "env2-13:2011" "env2-13:2012"
## [13] "env2-14:2011" "env2-14:2012" "env2-15:2011" "env2-15:2012"
## [17] "env2-2:2010"  "env2-2:2011"  "env2-2:2012"  "env2-3:2010"
## [21] "env2-3:2011"  "env2-3:2012"  "env2-4:2010"  "env2-4:2011"
## [25] "env2-4:2012"  "env2-5:2010"  "env2-5:2011"  "env2-5:2012"
## [29] "env2-6:2010"  "env2-6:2011"  "env2-6:2012"  "env2-7:2010"
## [33] "env2-7:2011"  "env2-7:2012"  "env2-8:2010"  "env2-8:2011"
## [37] "env2-8:2012"  "env2-9:2010"  "env2-9:2011"  "env2-9:2012"
```

2.2.2 Analysis of the model outputs

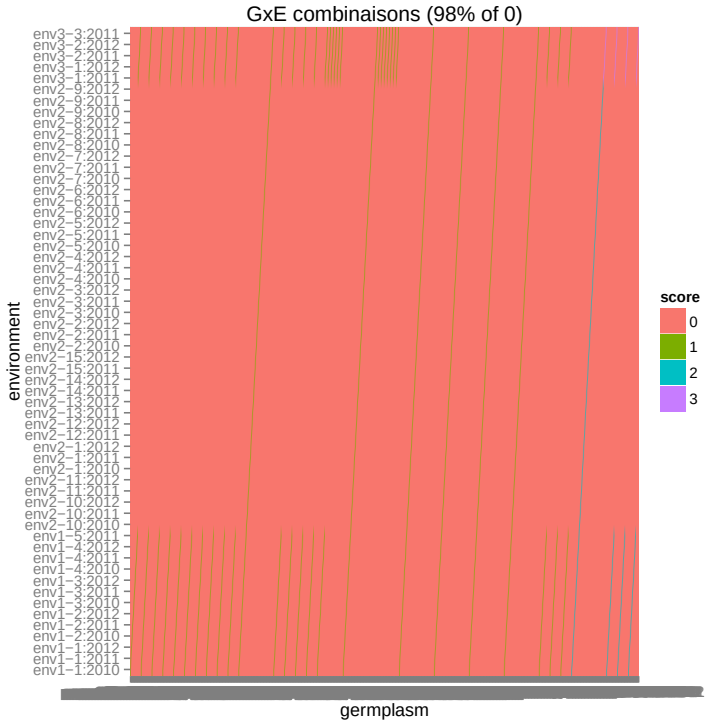
Once the model is run, it is necessary to check if the outputs can be taken with confidence. This step is needed before going ahead in the analysis (in fact, the MCMC object used in the next functions must come from `analyse.outputs`!).

```
# The experimental design plot is done.
# The Gelman-Rubin test is running for each parameter ...
# The two MCMC for each parameter converge thanks to the Gelman-Rubin test.
# The values of sigma in the inverse Gamme distribution are done.
# The mu_ij posterior distributions are done.
# The beta_jk posterior distributions are done.
# The sigma_j posterior distributions are done.
# The standardised residuals distributions are done.

load("./data_PPBstats/out1.RData")
```

`out1` is a list containing:

- "experimental_design" : a plot representing the presence/abscence matrix of G $\times$ E combinaisons. Here there are lots of 0 meaning that a lot of germplasm are no in at least two farms. A score of 1 is for a given germplasm in a given environment. A score of 2 is for a given germplasm replicated twice in a given environement. A score of 3 is for a given germplasm replicated three times in a given environement.

```
out1$data.experimental_design$plot
```

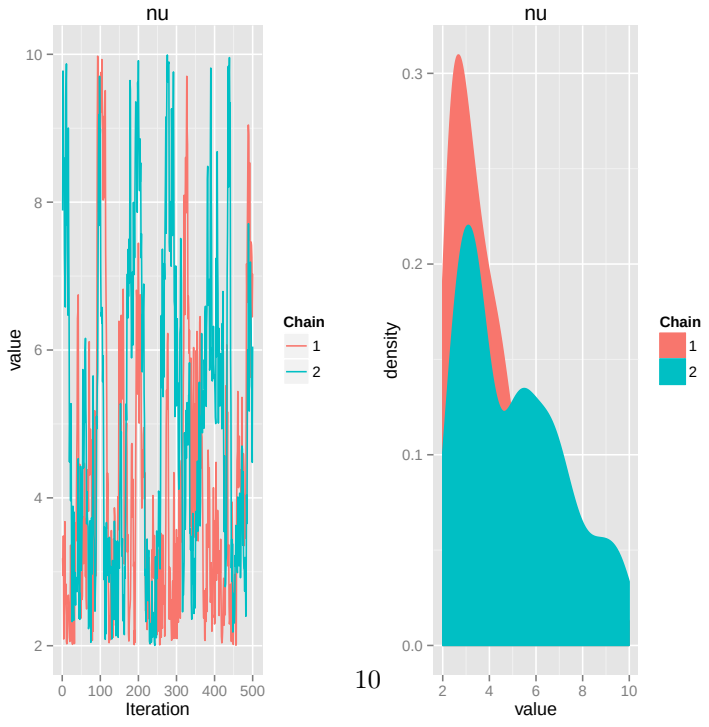


- "convergence" : a list with the plots of trace and density to check the convergence of the two MCMC only for chains that are not converging thanks to the Gelman-Rubin test (Gelman and Rubin, 1992). If all the chains converge, it is NULL

```
out1$convergence

## NULL
```

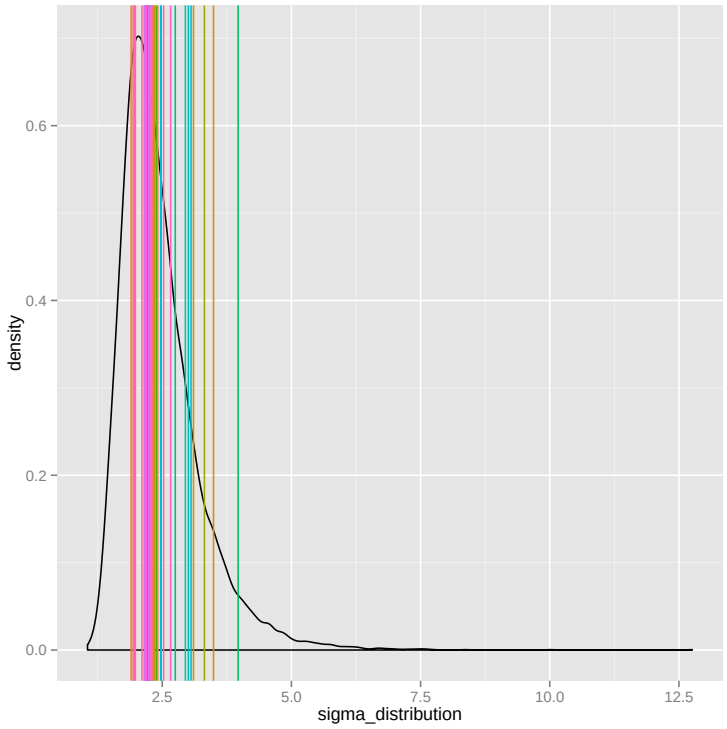
Here all the parameters converge. Below is an example where there is no convergence because the MCMC are too small.

```
# out.model1_bis = MC(data = PPBdata, variable = "tkw", nb_iteration = 5000)
#Compiling model graph
#   Resolving undeclared variables
#   Allocating nodes
#   Graph Size: 7662
#
#Initializing model
#
#  |++++++++++++++++++++++++++++++++++++++++++++++++++| 100%
#  |**************************************************| 100%
#  |**************************************************| 100%
#Warning message:
#In MC(data = PPBdata, variable = "tkw", nb_iteration = 5000) :
#  nb_iterations is below 20 000, which seems small to get convergence in the MCMC.

load("./data_PPBstats/out.model1_bis.RData") # To save time

# out1_bis = analyse.outputs(out.model1_bis)
# The experimental design plot is done.
# The Gelman-Rubin test is running for each parameter ...
# The two MCMC of the following parameters do not converge thanks to the Gelman-Rubin test :
# nu, rho, sigma[env1-1:2012], sigma[env1-2:2011], sigma[env2-12:2012], sigma[env2-6:2010].
# Therefore, they are not present in MCMC output.
# MCMC are updated, the following environment were deleted :
# env1-1:2012, env1-2:2011, env2-12:2012, env2-6:2010
# model1.data_env_whose_param_did_not_converge contains the raw data for these environments.
# The values of sigma in the inverse Gamme distribution are done.
# The mu_ij posterior distributions are done.
# The beta_jk posterior distributions are done.
# The sigma_j posterior distributions are done.

load("./data_PPBstats/out1_bis.RData") # To save time

# Get one example
toplot = out1_bis$convergence$"nu"
grid.arrange(toplot$traceplot, toplot$density, ncol=2, nrow=1)
```

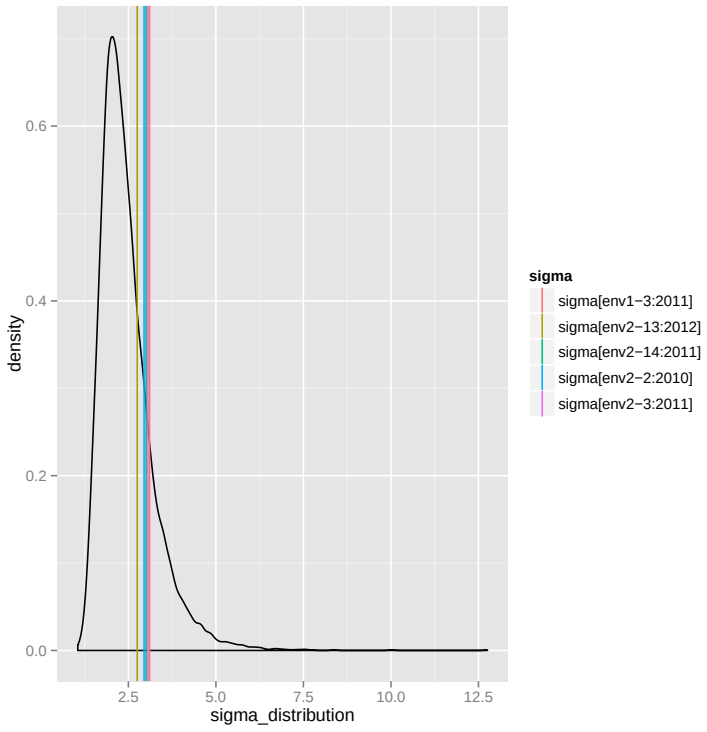


- "parameter_posteriors" : a list with
  - "sigma_distribution" : the distribution of the sigma is displayed on the Inverse Gamma distribution

```
out1$posteriors$sigma_distribution[[1]] # All the values
```

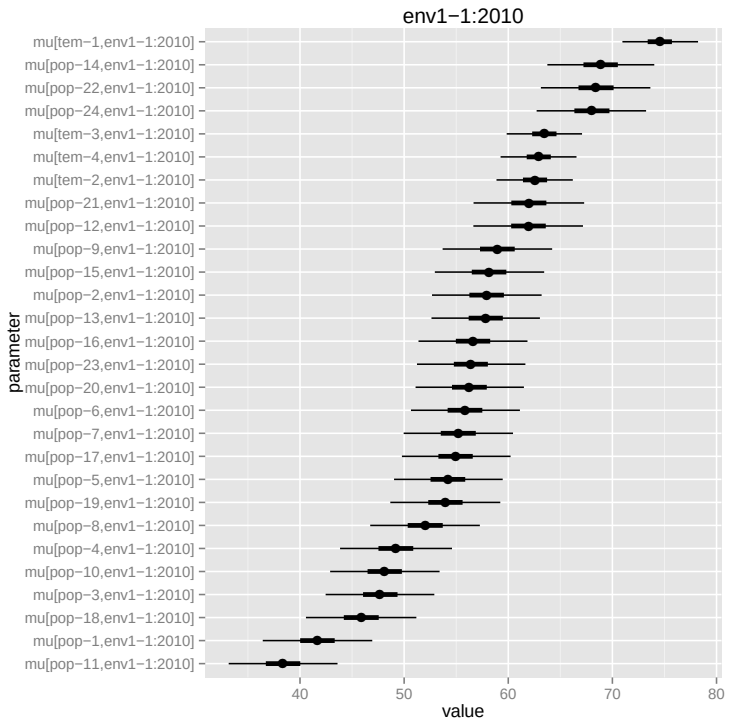



```
out1$posteriors$sigma_distribution[[12]] # A subset of values
```

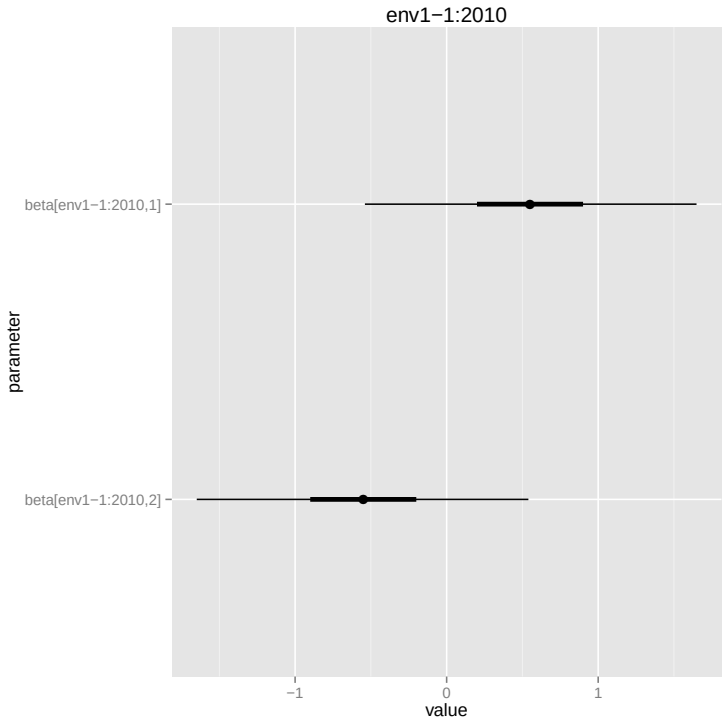


– "parameter_posteriors" : a caterpillar plot is display for each μ_{ij}, β_{jk} for a each environment and for σ_j. Below is an example for environment env1-1:2010. It is important to see it the values are coherent with your a priori knowledge. Indeed, a model can converge and estimate parameters'value that are not coherent!

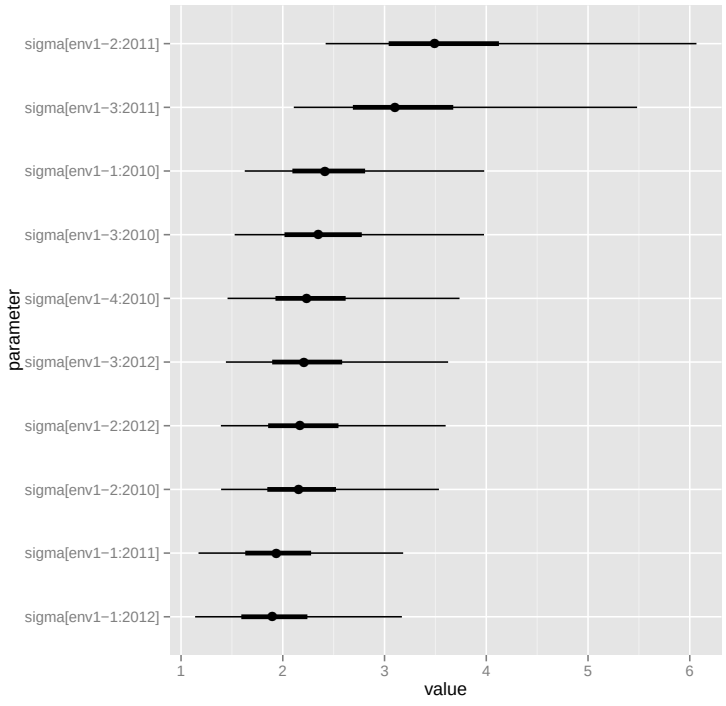
```
out1$posteriors$parameter_posteriors$mu_posteriors$"env1-1:2010"
```



```
out1$posteriors$parameter_posteriors$beta_posteriors$"env1-1:2010"
```

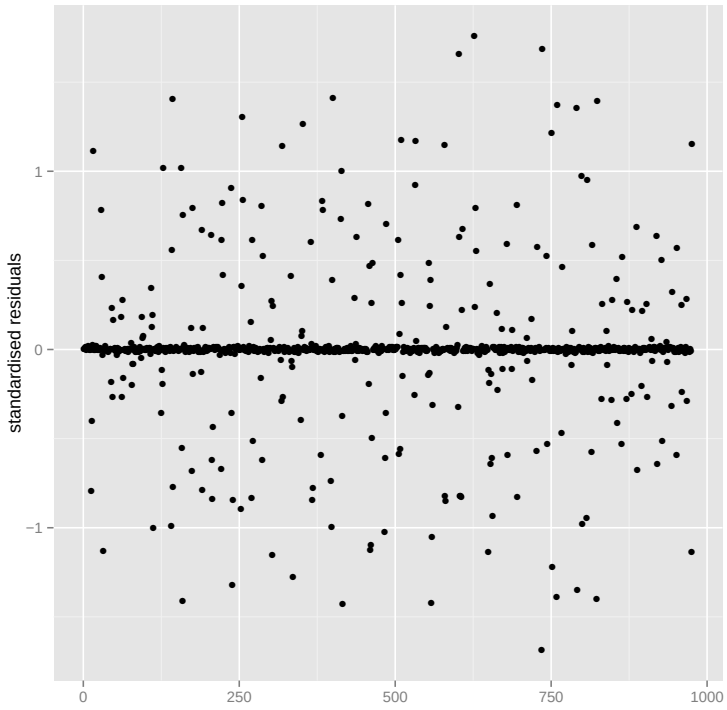



```
out1$posteriors$parameter_posteriors$sigma_posteriors[[1]]
```



– "standardized_residuals" : a plot to check the normality of the residuals. If the model went well it should be between -2 and 2.

```
out1$posteriors$standardized_residuals
```



- "MCMC" : a data fame resulting from the concatenation of the two MCMC for each parameter. This object can be used for further analysis. There are as many columns than parameters and as many rows than iterations//thin (the thin value is 10 by default in the models).

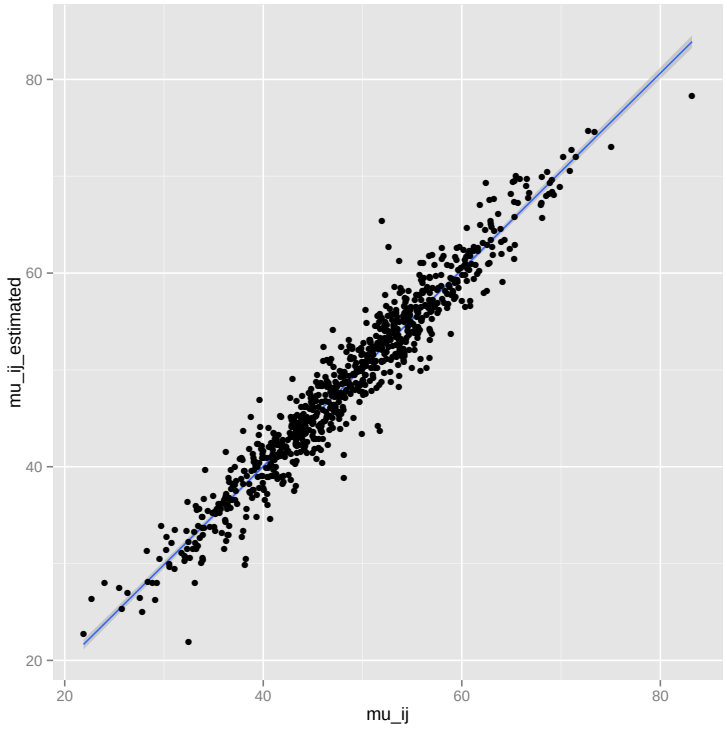
```
dim(out1$MCMC)

## [1] 20000   945
```

Just for fun, you can compare the posterior medians and the arithmetic means for the μ_{ij}.

```
MCMC = out1$MCMC
effects = apply(MCMC, 2, median)
mu_ij_estimated = effects[grep("mu",names(effects))]
names(mu_ij_estimated) = sapply(names(mu_ij_estimated),
                                function(x){  sub("\\]", "", sub("mu\\[", "", x)) }
                                )

d = filter(PPBdata, location != "env4")
d = filter(d, location != "env5")
d = droplevels(d)
environment = paste(as.character(d$location), as.character(d$year), sep = ":")
d$entry = as.factor(paste(as.character(d$germplasm), environment, sep = ","))
mu_ij = tapply(d$mu_ij, d$entry, mean, na.rm = TRUE)

check = cbind.data.frame(mu_ij, mu_ij_estimated[names(mu_ij)])
```

Let's have a look on the relation between the posterior medians and the arithmetic means. It goes pretty well!

```
p = ggplot(check, aes(x = mu_ij, y = mu_ij_estimated))
p + stat_smooth(method = "lm") + geom_point()
```



2.2.3 Get mean comparisons

In this part, the mean of each entry is compared to the mean of each other entry. Let H_0 and H_1 denote the hypotheses:

$$H_0 : \mu_{ij} \geq \mu_{i'j},\ H_1 : \mu_{ij} < \mu_{i'j}.$$

The difference $\mu_{ij} - \mu_{i'j}$ between the means of germplasm i and population i' in environment j was considered as significant if either H_0 or H_1 had a high posterior probability, that is if $Pr\{H_0|y\} > 1 - \alpha$ or $Pr\{H_1|y\} > 1 - \alpha$, where α was some specified threshold. The difference was considered as not significant otherwise. The posterior probability of a hypothesis was estimated by the proportion of MCMC simulations for which this hypothesis was satisfied (Figure 3).

Groups are made based on the probabilites. Germplasms which share the same group are not different. Germplasms which do not share the same groupe are different.

The threshold α that depends on agronomic objectives. This threshold is set by default to $\alpha = 0.1/I$ (with I the number of entries in a given environnement). It corresponded to a 'soft' Bonferroni correction, the Bonferroni correction being very conservative.

As one objective of this PPB programme is that farmers (re)learn selection, the threshold could be adjusted to allow the detection of at least two groups instead of having farmers choose at random. The initial value could be set to $\alpha = 0.1/I$ and if only one group is obtained, then this value could be adjusted to allow the detection of two groups. In this cases, the farmers should be informed of the lower degree of confidence that there are significant differences among entries.

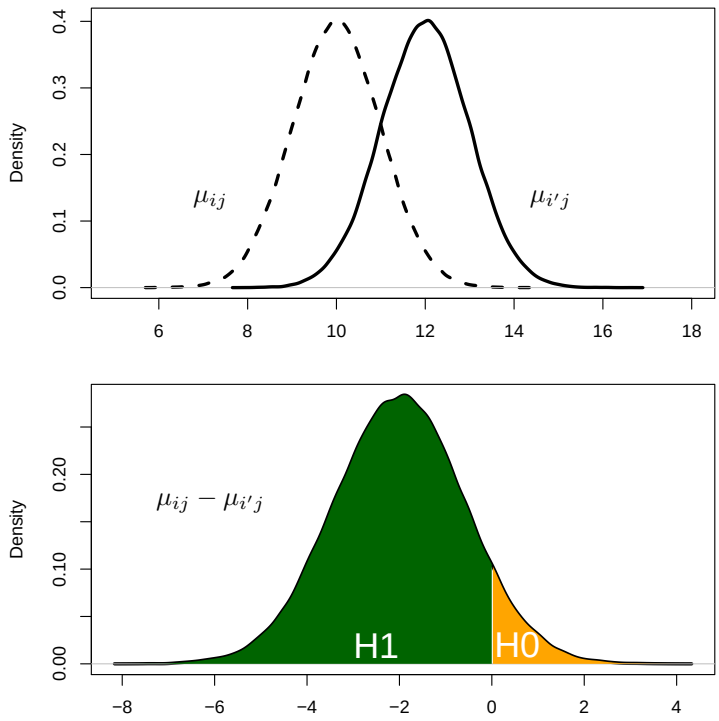


Figure 3: Mean comparison between μ_{ij} (dash line) and $\mu_{i'j}$ (plain line).

Computation In `PPBstats`, mean comparisons are done with `get.mean.comparisons`. You can choose on which parameters to run the comparison (`parameter` argument) and the α type one error (`alpha` argument). The soft Bonferonni correction is applied by default (`p.adj` argument). More informations on this function by typing `?get.mean.comparisons`.

```
# comp.mu = get.mean.comparisons(out1$MCMC, "mu")
# Get at least X groups for env2-1:2011. It may take some time ...
# Get at least X groups for  env2-1:2011 is done.
# Get at least X groups for env2-13:2011. It may take some time ...
# Get at least X groups for  env2-13:2011 is done.
# Get at least X groups for env2-3:2012. It may take some time ...
# Get at least X groups for  env2-3:2012 is done.
# Get at least X groups for env2-9:2010. It may take some time ...
# Get at least X groups for  env2-9:2010 is done.

load("./data_PPBstats/comp.mu.RData") # To save time
```

Plots

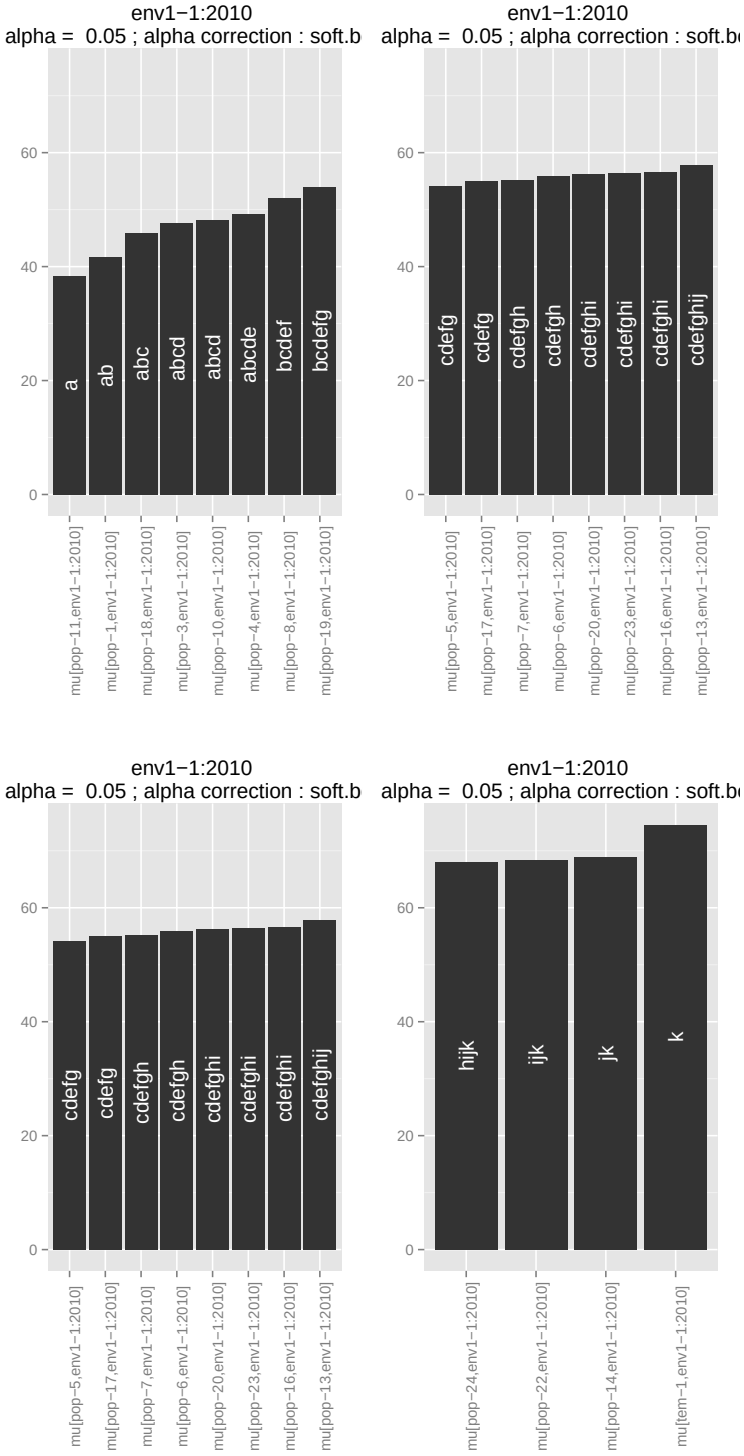
All entries in a given environment To see the output, use `get.ggplot`. On each plot, the `alpha` (type one error) value and the alpha correction are displayed. `alpha = Imp` means that no differences were possible to find. For `ggplot.type = "interaction"` and `ggplot.type = "score"`, it is display under the form: `alpha | alpha correction`.

```
p_barplot = get.ggplot(comp.mu, ggplot.type = "barplot")
length(p_barplot)

## [1] 58
```

```
names(p_barplot)
```

```
##  [1] "env1-1:2010"  "env1-1:2011"  "env1-1:2012"  "env1-2:2010"
##  [5] "env1-2:2011"  "env1-2:2012"  "env1-3:2010"  "env1-3:2011"
##  [9] "env1-3:2012"  "env1-4:2010"  "env1-4:2011"  "env1-4:2012"
## [13] "env1-5:2011"  "env2-10:2010" "env2-10:2011" "env2-10:2012"
## [17] "env2-11:2011" "env2-11:2012" "env2-1:2010"  "env2-1:2011"
## [21] "env2-1:2012"  "env2-12:2011" "env2-12:2012" "env2-13:2011"
## [25] "env2-13:2012" "env2-14:2011" "env2-14:2012" "env2-15:2011"
## [29] "env2-15:2012" "env2-2:2010"  "env2-2:2011"  "env2-2:2012"
## [33] "env2-3:2010"  "env2-3:2011"  "env2-3:2012"  "env2-4:2010"
## [37] "env2-4:2011"  "env2-4:2012"  "env2-5:2010"  "env2-5:2011"
## [41] "env2-5:2012"  "env2-6:2010"  "env2-6:2011"  "env2-6:2012"
## [45] "env2-7:2010"  "env2-7:2011"  "env2-7:2012"  "env2-8:2010"
## [49] "env2-8:2011"  "env2-8:2012"  "env2-9:2010"  "env2-9:2011"
## [53] "env2-9:2012"  "env3-1:2011"  "env3-1:2012"  "env3-2:2011"
## [57] "env3-2:2012"  "env3-3:2011"
```

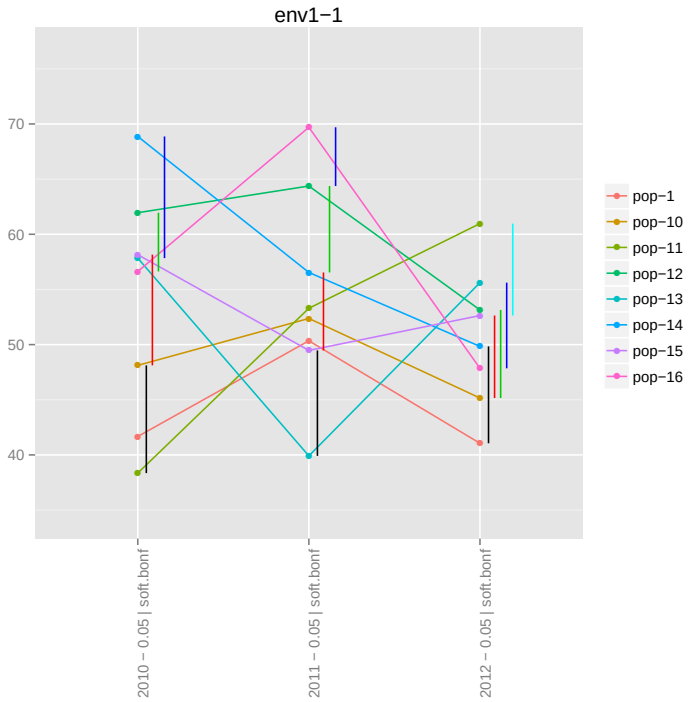
```
# For environment env-1-1:2010
grid.arrange(p_barplot$"env1-1:2010"[[1]], p_barplot$"env1-1:2010"[[2]] , ncol = 2, nrow = 1)
grid.arrange(p_barplot$"env1-1:2010"[[2]], p_barplot$"env1-1:2010"[[4]], ncol = 2, nrow = 1)
```



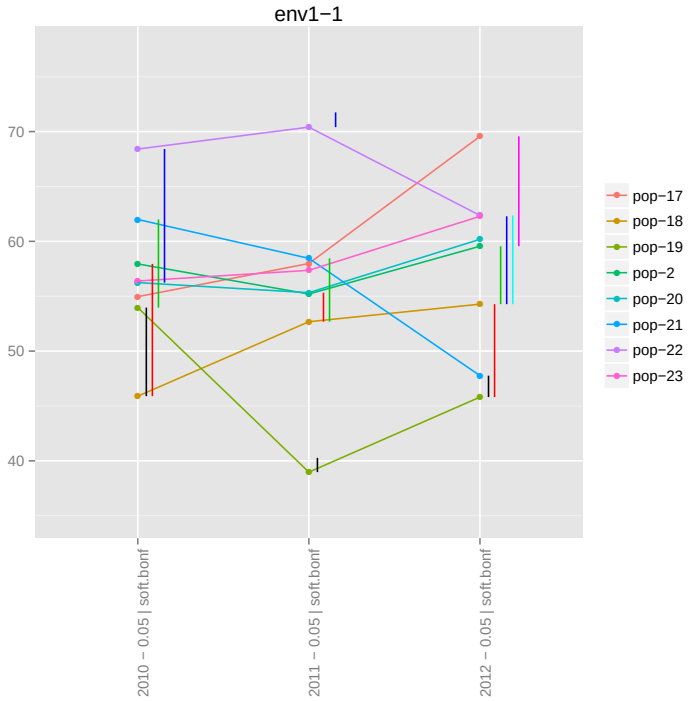
With `ggplot.type = "interaction"`, you can display the year effect as well as detect groups. One group is represented by one dashed line. Germplasms which share the same group are not different. Germplasms which do not share the same groupe are different (section 2.2.3).

```
p_interaction = get.ggplot(comp.mu, ggplot.type = "interaction")
```

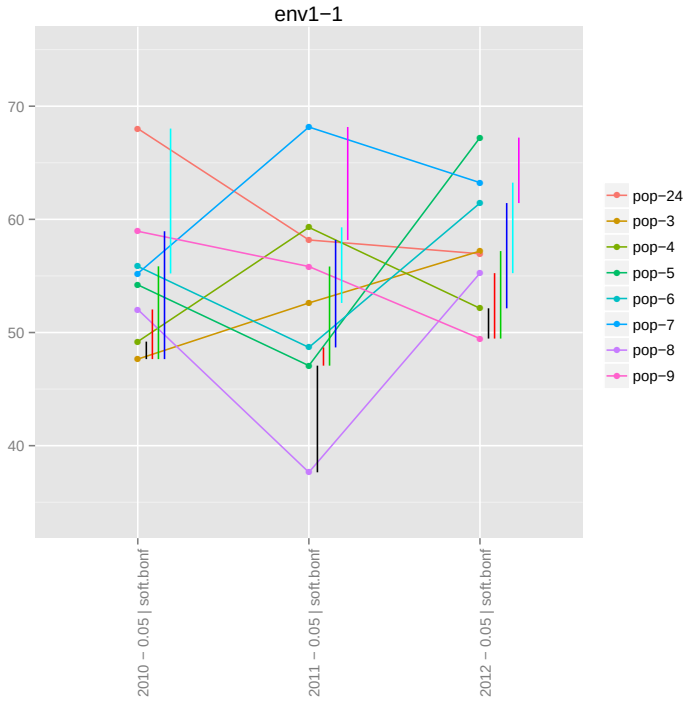
```
# For location env-1-1.
p_interaction$"env1-1"[[1]]
```



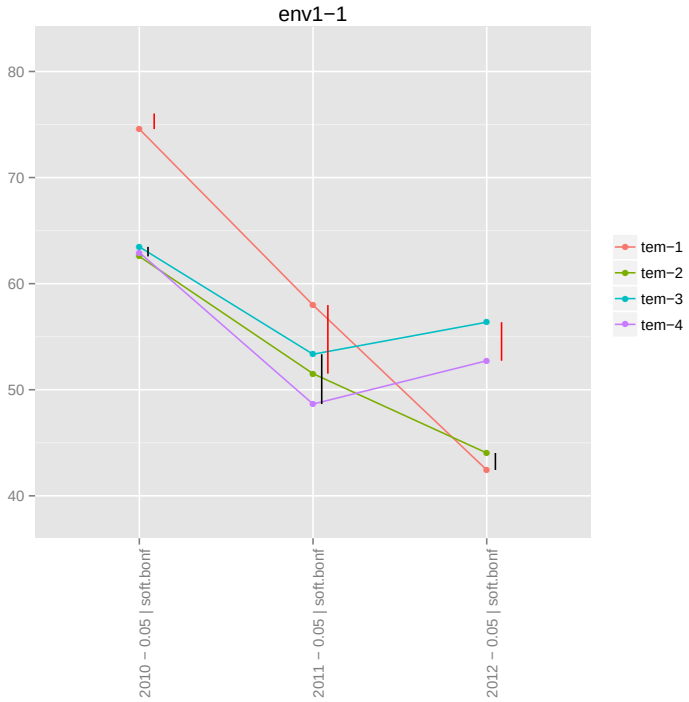

```
p_interaction$"env1-1"[[2]]
```



```
p_interaction$"env1-1"[[3]]
```



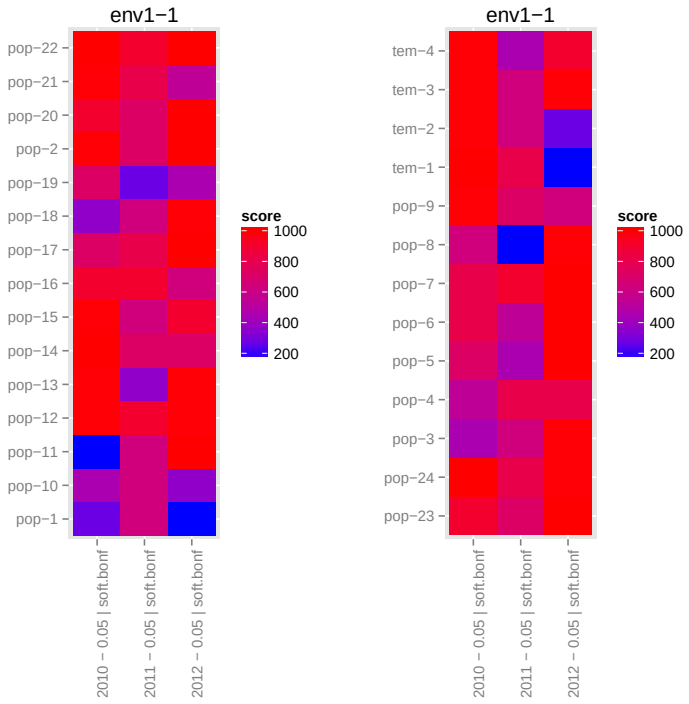

```
p_interaction$"env1-1"[[4]]
```



For the score, more entries are displayed. An high score means that the entry was in a group with an high mean. A low socre means that the entry was in a group with an low mean. This plot is useful to look

at year effects.

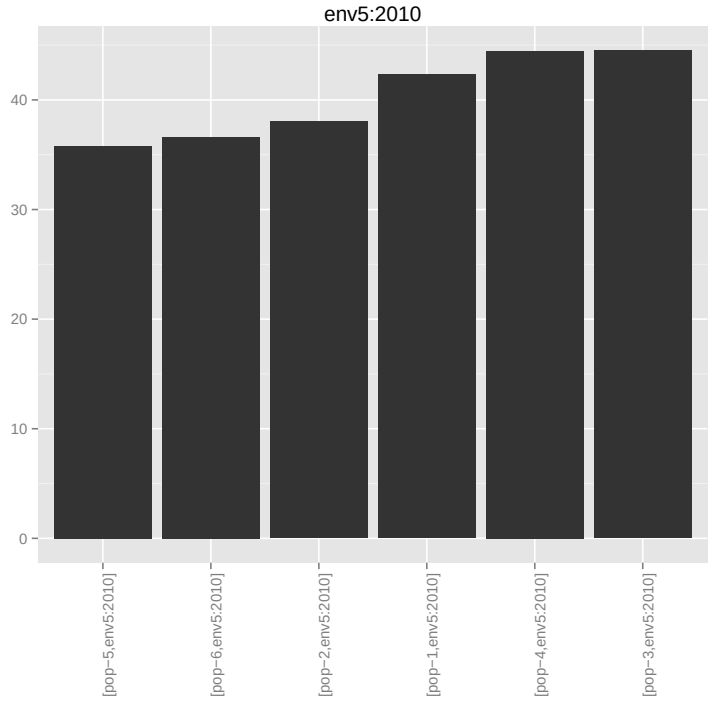
```
p_score = get.ggplot(comp.mu, ggplot.type = "score", nb_parameters_per_plot = 15)
# For location env-1-1
grid.arrange(p_score$"env1-1"[[1]], p_score$"env1-1"[[2]] , ncol = 2, nrow = 1)
```



The same method is used for each β_{jk}.

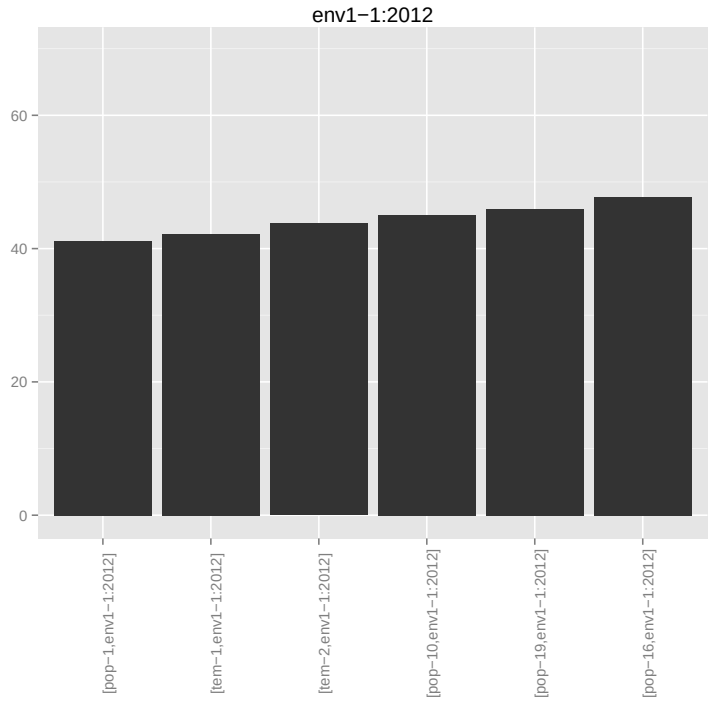
For environments with no controls or where at least one MCMC did not converge, it may be useful to get the plot as well.

```
get.ggplot(out.model1$data_env_with_no_controls, ggplot.type = "barplot")

## $`env5:2010`
## $`env5:2010`$`1`
```



You can also do a plot with interaction. Here it is not useful as there is only one year.

```
g = get.ggplot(out1_bis$model1.data_env_whose_param_did_not_converge, ggplot.type = "barplot")

names(g)

## [1] "env1-1:2012"  "env1-2:2011"  "env2-12:2012" "env2-6:2010"

g$`env1-1:2012`$`1`
```

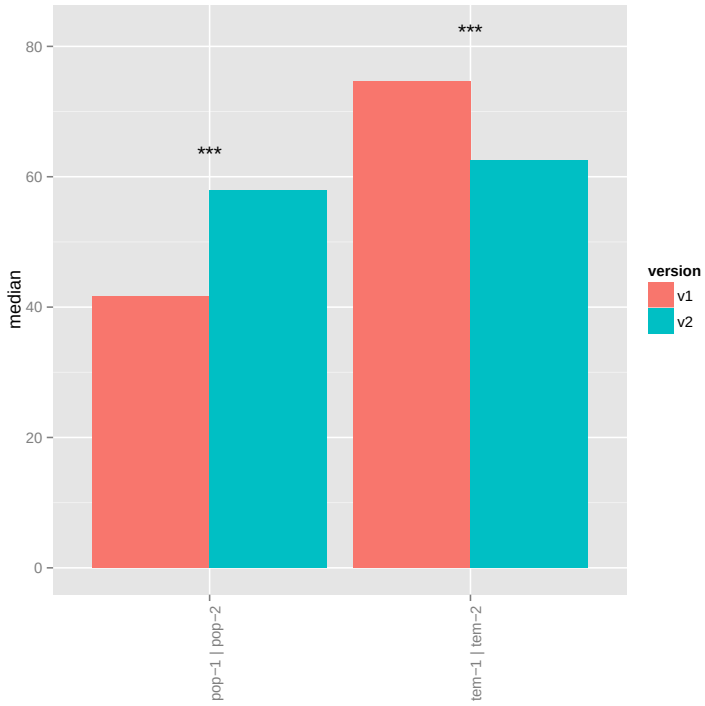


Pairs of entries in a given environment It is possible to get comparison of paris of entries in a given location. This is useful if you want to compare two versions within a group. For exemple:

```
data(data_version)
head(data_version)

##   year location germplasm group version
## 1 2010   env1-1     tem-1     1      v1
## 2 2010   env1-1     tem-2     1      v2
## 3 2010   env1-1     pop-1     2      v1
## 4 2010   env1-1     pop-2     2      v2
## 5 2010   env1-2     tem-1     3      v1
## 6 2010   env1-2     tem-2     3      v2
```

Here, in location `env1-1`, `tem-1` and `tem-2` are two version belonging to the same groupe.
Lets' make the plots:

```
g = get.ggplot(data = comp.mu, data_version = data_version, ggplot.type = "barplot")

## Warning in get.ggplot(data = comp.mu, data_version = data_version, ggplot.type = "barplot"):
The following environments in data_version are not taken:  env5:2010.

g$`env1-1:2010`$`1`
```



The stars corresponds to the pvalue:

pvalue	stars
< 0.001	***
$[0.001, 0.05]$	**
$[0.05, 0.01]$	*
> 0.01	.

The pvalue is computed as describe in section 2.2.3 if the parameters have been estimated with the model.

It is also possible to make this kind of plots for data that did not converge or without environments. In this case, it is a `t.test` which is perform.

```
g = get.ggplot(out1_bis$model1.data_env_whose_param_did_not_converge, data_version = data_version, gg

## Warning in get.ggplot(out1_bis$model1.data_env_whose_param_did_not_converge, :  The following
environments in data_version are not taken:  env1-1:2010, env1-2:2010, env5:2010.
## Warning in FUN(X[[i]], ...):  No t.test are done as there are not enough observations.

g = get.ggplot(out.model1$data_env_with_no_controls, data_version = data_version, ggplot.type = "barp

## Warning in get.ggplot(out.model1$data_env_with_no_controls, data_version = data_version,
:  The following environments in data_version are not taken:  env1-1:2010, env1-2:2010, env1-1:2012.
## Warning in FUN(X[[i]], ...):  No t.test are done as there are not enough observations.
```

3 At the network level : model 2 to analyse $G \times E$ interaction in the network of farms

3.1 The model

The phenotypic value Y_{ij} for a given variable Y, germplasm i and environment j, was modeled as :

$$Y_{ij} = \alpha_i + \theta_j + \eta_i\theta_j + \varepsilon_{ij}; \quad \varepsilon_{ij} \sim \mathcal{N}(0, \sigma_e^2),$$

for $i = 1, \ldots, I$ and $j = 1, \ldots, J$, where I was the number of germplasms, J was the number of environments, α_i was the main effect of germplasm i, θ_j was the main effect of environnment j, ε_{ij} was the residual and $\mathcal{N}(0, \sigma_e^2)$ was the normal distribution with mean 0 and variance σ_e^2. The interaction between germplasm i and environment j was divided into a multiplicative term $\eta_i\theta_j$ and a remaining term that contributed to the residual ε_{ij}.

This model was written as :

$$Y_{ij} = \alpha_i + \beta_i\theta_j + \varepsilon_{ij}; \quad \varepsilon_{ij} \sim \mathcal{N}(0, \sigma_\varepsilon), \tag{2}$$

Where $\beta_i = (1+\eta_i)$ was the sensitivity of germplasm i to environments. This model is known as the Finlay Wilkinson model or as joint regression (Finlay and Wilkinson, 1963). Germplasm sensitivities quantified the stability of germplasm performances over environments. The average sensitivity was equal to 1 so that a gemplasm with $\beta_i > 1$ ($\beta_i < 1$) was more (less) sensitive to environments than a germplasm with the average sensitivity (Nabugoomu et al., 1999).

Given the high disequilibrium of the data and the large amount of data, we decided to implement this model with a hierarchical Bayesian approach. In the following, this Hierarchical Finlay Wilkinson model was denoted by HFW.

We used hierarchical priors for α_i, β_i and θ_j and a vague prior for σ_ε.

$$\alpha_i \sim \mathcal{N}(\mu, \sigma_\alpha^2), \quad \beta_i \sim \mathcal{N}(1, \sigma_\beta^2), \quad \theta_j \sim \mathcal{N}(0, \sigma_\theta^2), \quad \sigma_\varepsilon^{-2} \sim \mathcal{G}amma(10^{-6}, 10^{-6}),$$

where μ, σ_α^2, σ_β^2 and σ_θ^2 were unknown parameters. The mean of β_i was set to 1 (Nabugoomu et al., 1999).

Then, we placed weakly-informative priors on the hyperparmeters μ, σ_α^2, σ_β^2 and σ_θ^2:

$$\mu \sim \mathcal{N}(\nu, \nu^2), \quad \sigma_\alpha \sim \mathcal{U}niforme(0, \nu), \quad \sigma_\beta \sim \mathcal{U}niforme(0, 1), \quad \sigma_\theta \sim \mathcal{U}niforme(0, \nu),$$

where ν was the arithmetic mean of the data : $\nu = \sum_{ij} Y_{ij}/n$ where n was the number of observations. Uniform priors were used for σ_α^2, σ_β^2 and σ_θ^2 to reduce the influence of these priors on posterior results (**?**). The support of these priors took account of the prior knowledge that σ_α^2, σ_β^2 and σ_θ^2 were expected to be respectively smaller than ν, 1, ν.

Initial values for each chain were taken randomly except for μ, σ_α and σ_θ whose initial values were equal to their posterior median from additive model (i.e. model 2 with $\forall i, \beta_i = 1$).

The main parameter of interest were germplasm main effects ($\alpha_i, i = 1, \ldots, I$), environment main effects ($\theta_j, j = 1, \ldots, J$) and germplasm sensitivities ($\beta_i, i = 1, \ldots, I$). For α_i, the average posterior response of each germplasm over the environments of the network was considered:

$$\gamma_i = \alpha_i + \beta_i\bar{\theta},$$

where $\bar{\theta} = \sum^J \theta_j/J$.

To simplify, the α_i notation is kept instead of γ_i (i.e. $\alpha_i = \gamma_i$). But keep in mind it has been corrected.

3.2 With `PPBstats`

For model 2, you can follow these steps (Figure 2):

1. Run the model with `FWH`
2. Analyse model outputs with graphs to kow if you can continue the analysis with `analyse.outputs`
3. Perform cross validation studies with `cross.validation.FWH` in order to assess the quality of the model
4. Get mean comparisons for each factor with `get.mean.comparisons` and `get.ggplot`
5. Get groups of parameters for α, β and θ with `get.parameters.groups` and `get.ggplot`
6. Predict the past with `predict.the.past` and `get.ggplot`

Let's get the data. The values for α_i, β_i, θ_j are the real value taken to create the dataset for y1. This dataset is representative of data you can get in a PPB programme.

```
data(PPBdata2)
head(PPBdata2)

##   germplasm location   year        y1 alpha_i-1 beta_i-1  theta_j-1
## 1     geno1   loc-35 year-1  7.926204  10.25349 2.170004 -0.7776704
## 2     geno1   loc-48 year-1  9.772076  10.25349 2.170004 -0.7531355
## 3     geno1   loc-20 year-5  9.199745  10.25349 2.170004  0.1163468
## 4     geno1   loc-33 year-5 10.131745  10.25349 2.170004  0.2755013
## 5     geno1   loc-44 year-3 14.329280  10.25349 2.170004  1.8495949
## 6     geno1   loc-34 year-1  8.709140  10.25349 2.170004 -0.5750281
##         y2       y3 block X Y
## 1 18.28223 31.57931     1 1 1
## 2 18.41129 31.44957     1 2 2
## 3 18.94209 33.19169     1 3 3
## 4 24.86338 29.34573     1 4 4
## 5 16.09421 32.36811     1 5 5
## 6 17.93222 37.85269     1 6 6
```

3.2.1 Run the model

To run model 2 on the dataset, used the function `FWH` (which stands for Finlay Wilkinson Hierarchical). You can run it on one variable. Here it is on thousand kernel weight (tkw)

By default, `FWH` returns posteriors for α_i (`return.alpha = TRUE`), σ_α (`return.sigma_alpha = TRUE`), β_i (`return.beta = TRUE`), σ_β (`return.sigma_beta = TRUE`), θ_j (`return.theta = TRUE`), σ_θ (`return.sigma_theta = TRUE`) and σ_ϵ (`return.sigma_epsilon = TRUE`). You can also get ϵ_{ij} with `return.epsilon = TRUE`.

By default, DIC is not display, you may want this value to compare to other model (`DIC = TRUE`). DIC criterion is a generalization of the AIC criterion that can be used for hierarchical models (Spiegelhalter et al., 2002). The smaller the DIC value, the better the model (Plummer, 2008).

```
# out.model2 = FWH(data = PPBdata2, variable = "y1", return.epsilon = TRUE)
#Run additive model ...
#Compiling model graph
#   Resolving undeclared variables
#   Allocating nodes
#   Graph Size: 9759
#
#Initializing model
#
#  |++++++++++++++++++++++++++++++++++++++++++++++++++| 100%
#  |**************************************************| 100%
#  |**************************************************| 100%
#Run FWH model ...
#Compiling model graph
#   Resolving undeclared variables
#   Allocating nodes
#   Graph Size: 14677
#
#Initializing model
#
#  |++++++++++++++++++++++++++++++++++++++++++++++++++| 100%
#  |**************************************************| 100%
#  |**************************************************| 100%
#  |**************************************************| 100%
```

```
load("./data_PPBstats/out.model2.RData") # To save time
```

It may be useful to see which germplasm were not use in the analysis because they were in only one environment.

```
out.model2$germplasm.not.used

## NULL
```

3.2.2 Analysis of model outputs

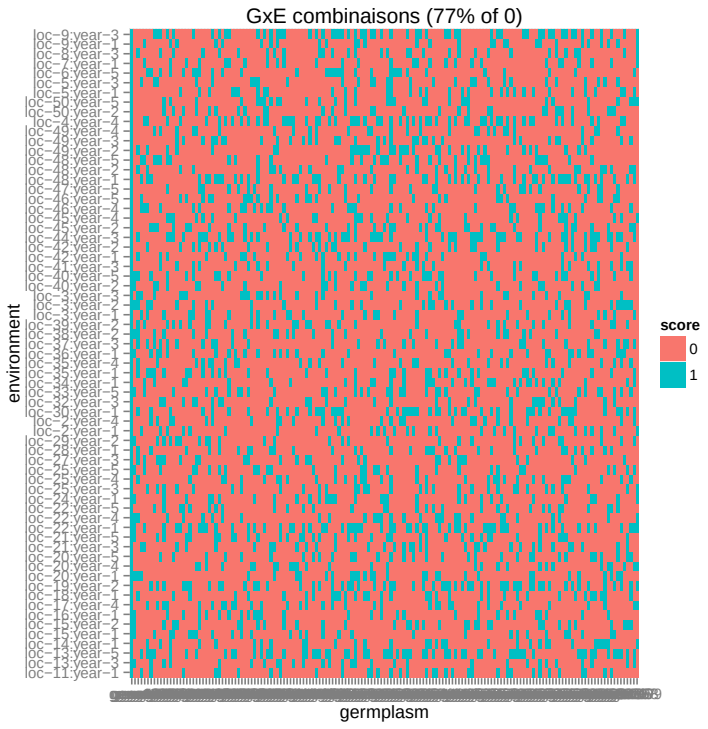
Once the model is run, it is necessary to check if the outputs can be taken with confidence. This step is needed before going ahead in the analysis (in fact, the MCMC object used in the next functions must come from `analyse.outputs`!).

```
# out2 = analyse.outputs(out.model2)
# The experimental design plot is done.
# The Gelman-Rubin test is running for each parameter ...
# The two MCMC for each parameter converge thanks to the Gelman-Rubin test.
# The alpha_i posterior distributions are done.
# The beta_i posterior distributions are done.
# The theta_j posterior distributions are done.
# The standardised residuals distributions are done.

load("./data_PPBstats/out2.RData") # To save time
```

`out2` is a list containing :

- "experimental_design" : a plot representing the presence/abscence matrix of G × E combinaisons. Note that it displays only germplasms that are on at least two environments.

```
out2$data.experimental_design$plot
```

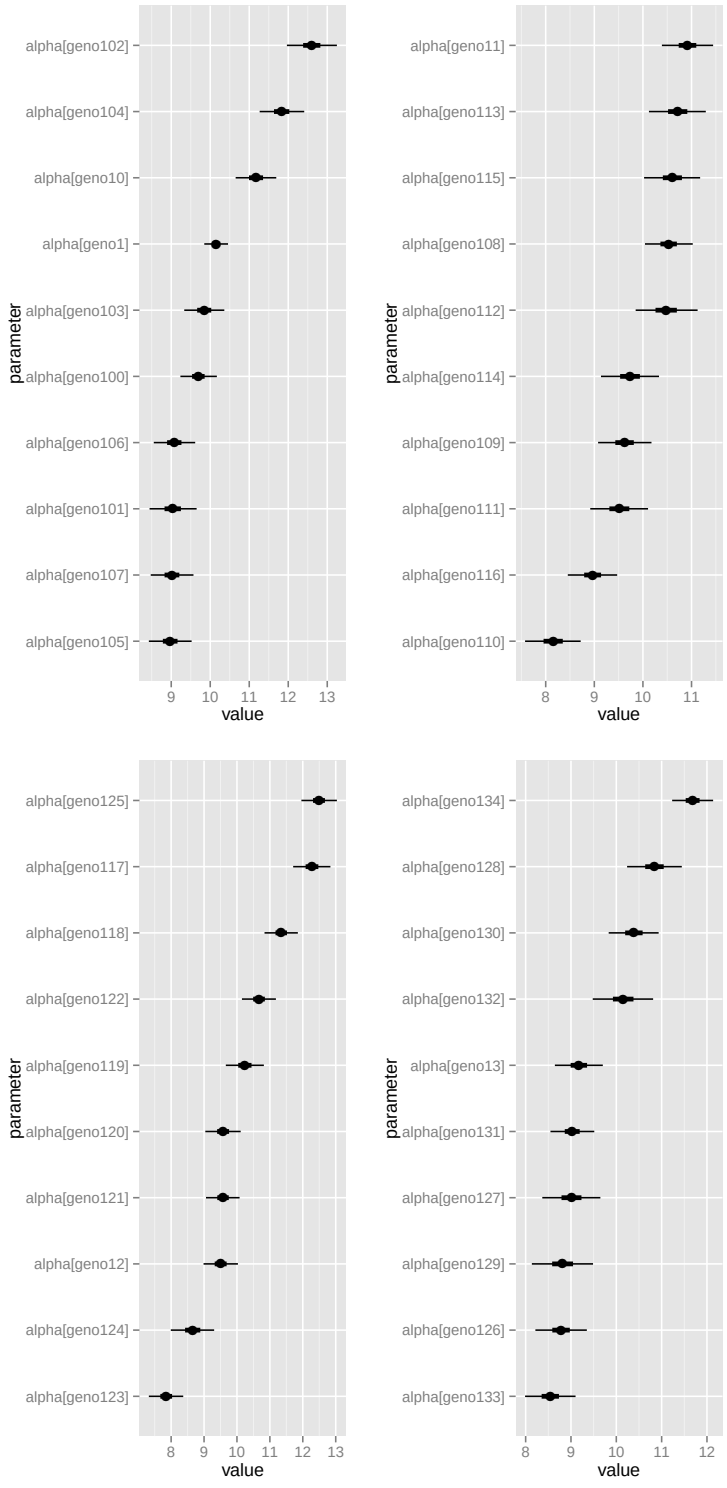


- "convergence" : a list with the plots of trace and density to check the convergence of the two MCMC only for chains that are not converging thanks to the Gelman-Rubin test (Gelman and Rubin, 1992). If all the chains converge, it is NULL

```
out2$convergence

## NULL
```

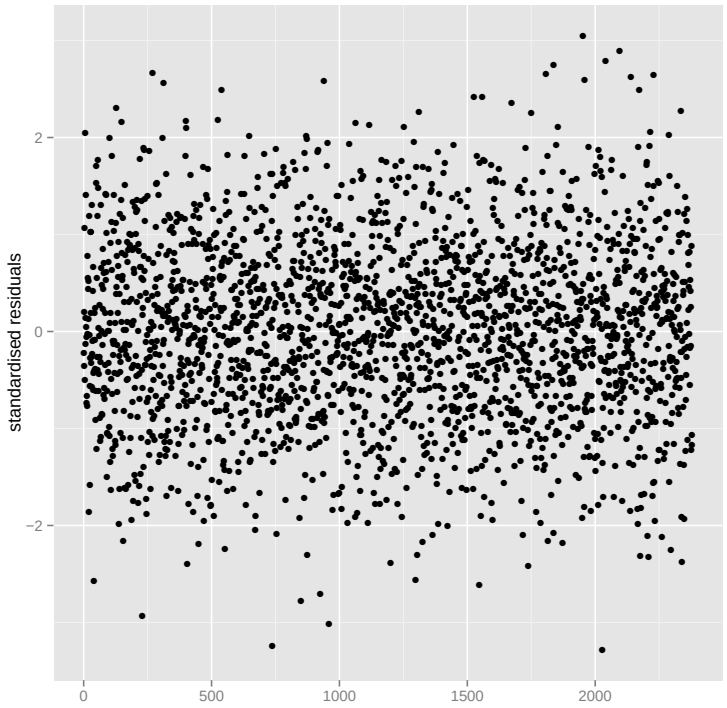
- "parameter_posteriors": a list with caterpillar plot for each α_i, β_i and θ_j.
  Below an example for α_i.

```
p = out2$posteriors$parameter_posteriors$alpha_posteriors
grid.arrange(p[[1]], p[[2]],ncol = 2, nrow = 1)
grid.arrange(p[[3]], p[[4]],ncol = 2, nrow = 1)
```



- "standardized_residuals" : a plot to check the normality of the residuals. If the model went well it should be between -2 and 2.

```
out2$posteriors$standardized_residuals
```



- "MCMC": a data fame resulting from the concatenation of the two MCMC for each parameter. This object can be used for further analysis. There are as many columns than parameters and as many rows than iterations/thin (the thin value is 10 by default in the models).

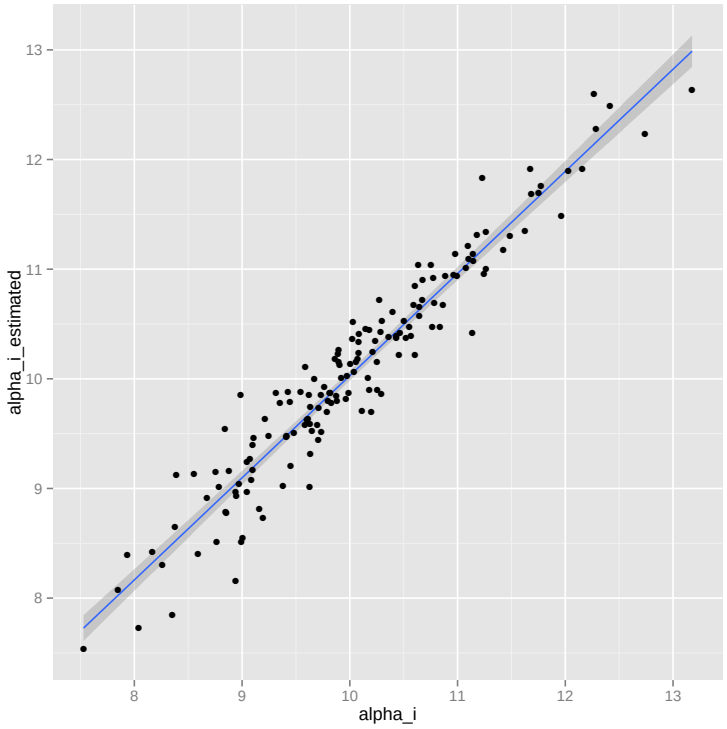
```
dim(out2$MCMC)

## [1] 20000   385
```

Just for fun, you compare the posterior medians and the arithmetic means for the α_i's.

```
MCMC = out2$MCMC
effects = apply(MCMC, 2, median)
alpha_i_estimated = effects[grep("alpha\\[",names(effects))]
names(alpha_i_estimated) = sapply(names(alpha_i_estimated), function(x){
sub("\\]", "", sub("alpha\\[", "", x)) } )

alpha_i = tapply(PPBdata2$alpha_i, PPBdata2$germplasm, mean, na.rm = TRUE)

check = cbind.data.frame(alpha_i = alpha_i, alpha_i_estimated = alpha_i_estimated[names(alpha_i)])
```

Let's have a look at the relation between both values.

```
p = ggplot(check, aes(x = alpha_i, y = alpha_i_estimated))
p + stat_smooth(method = "lm") + geom_point()
```



3.2.3 Perform cross validation studies

This step is useful to assess the quality of the model. This step is higly computing consuming as the FWH model is run as many time as there is value of Y_{ij} (i.e. number of rows of the data set).

The complete cross validation is done with `cross.validation.FWH`: each Value of Y_{ij} is estimated by the entire data set without this value.

The convergence is not check for each validation. If the parameters in the FWH converge, then it is assumed that the FWH in the cross validation converge as well.

The model is run on dataset where germplasms are in three environments at least so the smallest data set where the cross valisation is run has germplasms present in two environments at least.

You may parallelise to gain time with the `mc.cores` argument of the function.

The number of iterations is set to 100 000 but you can change it with the `nb_iterations` argument.

The percentage of confidence is calculated with a t-test:

$$t = \frac{m - 0}{s/\sqrt{N}}$$

with,

N the number of observations in the data set,

$m = \frac{1}{N} \sum_{n=1}^{N} Y_n - \hat{Y_n}$, the average bias

$s = \sqrt{\frac{1}{N-1} \sum_{n=1}^{N} (Y_n - \hat{Y_n})^2}$, the standard deviation of the bias

t follows a Student distribution with $N - 1$ degree of freedom.

The percentage of confidence (i.e. the probability $H0$: the bias is equal to zero) comes from this distribution.

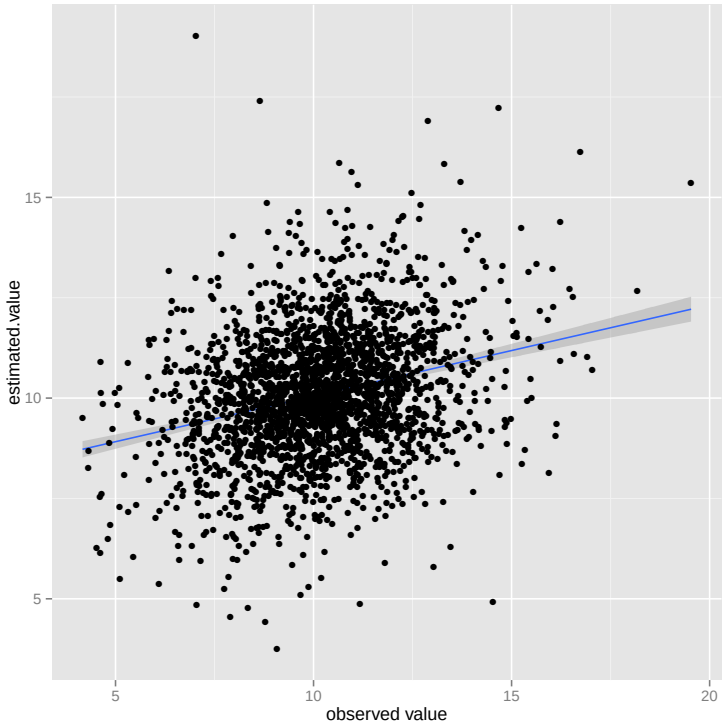
A regression is also done between estimated and observed value.

Here it is bad as only 10 iterations have been done to save computing time ...

```
# out.cv = cross.validation.FWH(data = PPBdata2, variable = "y1", nb_iterations = 10)
load("./data_PPBstats/out.cv.RData") # to save lots of time
```

```
out.cv$percentage.of.confidence

## [1] 6.7
```

```
out.cv$regression

## $plot

## Warning in loop_apply(n, do.ply): Removed 1 rows containing missing values
(stat_smooth).
## Warning in loop_apply(n, do.ply): Removed 1 rows containing missing values
(geom_point).

##
## $anova
##
## Call:
## lm(formula = real.value ~ estimated.value)
##
## Coefficients:
##     (Intercept)  estimated.value
##          6.8640           0.3275
```



3.2.4 Get mean comparisons

For mean comparisons of parameters, it is the same method that presented in section 2.2.3.

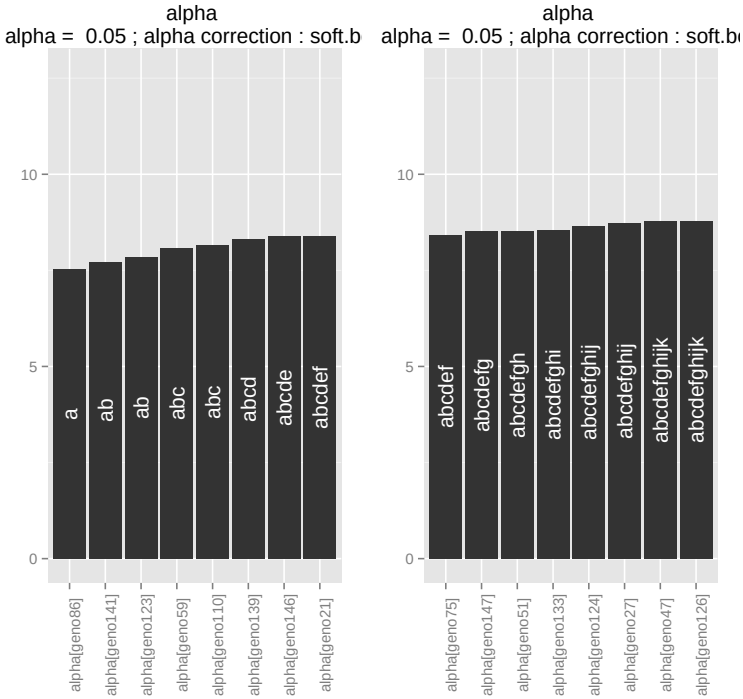
```
comp.alpha = get.mean.comparisons(out2$MCMC, "alpha")
comp.theta = get.mean.comparisons(out2$MCMC, "theta")
comp.beta = get.mean.comparisons(out2$MCMC, "beta", type = 2, threshold = 1)
```

To see the output, use `get.ggplot`.

```
p_barplot = get.ggplot(comp.alpha, ggplot.type = "barplot")
```

Lets' have a look at the firt values of α_i.

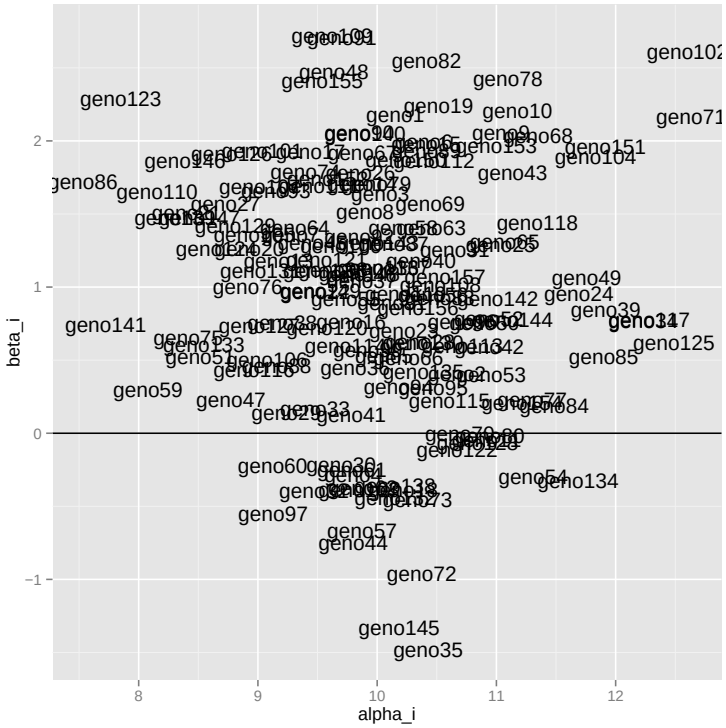
```
grid.arrange(p_barplot$"alpha"[[1]], p_barplot$"alpha"[[2]] , ncol = 2, nrow = 1)
```



3.2.5 Get biplot $\beta = f(\alpha)$

It is interessting to compare genetic effect versus sensibility to interaction. A germplasm with an high genetic effect and a low sensitivity to interaction (i.e. close to 0) may be a good candidate to sown.

```
comp.alpha = get.mean.comparisons(out2$MCMC, "alpha")
comp.beta = get.mean.comparisons(out2$MCMC, "beta")

g = get.ggplot(data = comp.alpha, data_2 = comp.beta, ggplot.type = "biplot-alpha-beta")
g$biplot
```



3.2.6 Get groups of parameters

In order to cluster environments or germplasms, you may use mulivariate analysis on a matrix with several variables in columns and parameter in rows.

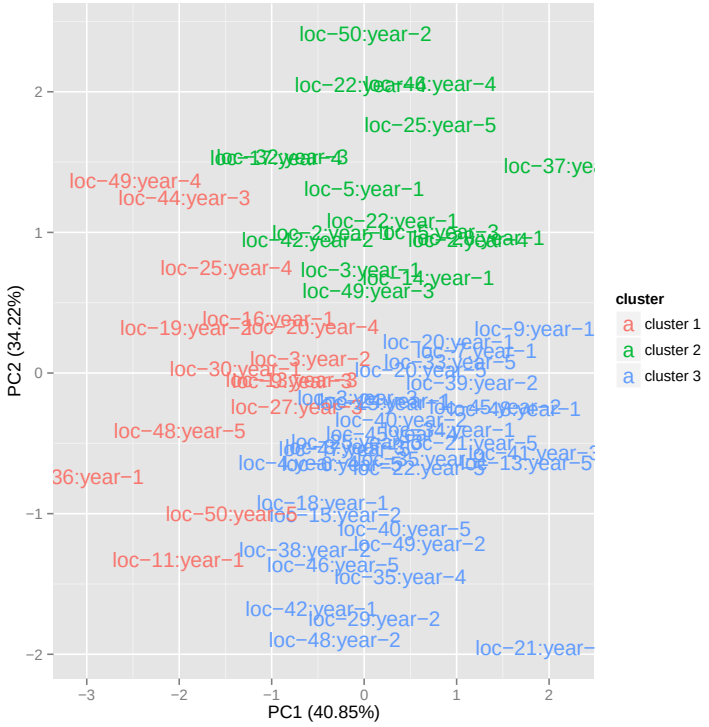
This is done with `get.parameter.groups` which do a PCA on this matrix and then find cluster with the `HCPC` procedure from package `FactoMineR`: it is a K-means clustering that creates clusters of similar parameters (Husson et al., 2010). The Kmeans clustering was done on the first two axes of the PCA that represented the main information while the last axes represented mainly noise (Husson et al., 2010). The number of clusters is choosen to maximise the variance between clusters and within clusters. For more information type `?get.parameter.groups`.

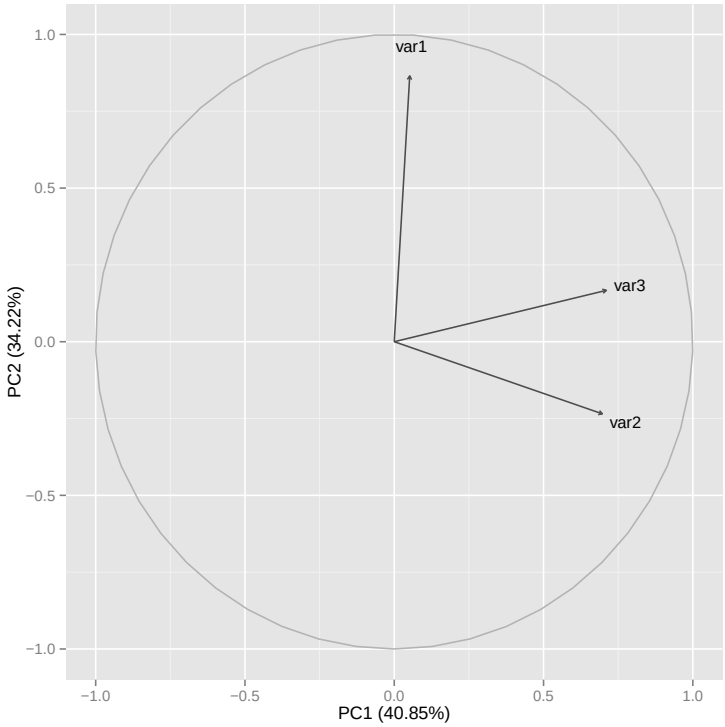
```
# out.model2_y1 = FWH(PPBdata2, variable = "y1")
load("./data_PPBstats/out.model2_y1.RData") # to save time

# out.model2_y2 = FWH(PPBdata2, variable = "y2")
load("./data_PPBstats/out.model2_y2.RData") # to save time

# out.model2_y3 = FWH(PPBdata2, variable = "y3")
load("./data_PPBstats/out.model2_y3.RData") # to save time

out2_y1 = analyse.outputs(out.model2_y1)

## The experimental design plot is done.
## The Gelman-Rubin test is running for each parameter ...
## The two MCMC for each parameter converge thanks to the Gelman-Rubin test.
## The alpha_i posterior distributions are done.
## The beta_i posterior distributions are done.
## The theta_j posterior distributions are done.

out2_y2 = analyse.outputs(out.model2_y2)

## The experimental design plot is done.
## The Gelman-Rubin test is running for each parameter ...
```

```
## The two MCMC for each parameter converge thanks to the Gelman-Rubin test.
## The alpha_i posterior distributions are done.
## The beta_i posterior distributions are done.
## The theta_j posterior distributions are done.

out2_y3 = analyse.outputs(out.model2_y3)

## The experimental design plot is done.
## The Gelman-Rubin test is running for each parameter ...
## The two MCMC for each parameter converge thanks to the Gelman-Rubin test.
## The alpha_i posterior distributions are done.
## The beta_i posterior distributions are done.
## The theta_j posterior distributions are done.

analyse.outputs.list = list(var1 = out2_y1, var2 = out2_y2, var3 = out2_y3)

clust = get.parameter.groups(analyse.outputs.list, parameter = "theta")
```

To see the output, use `get.ggplot`. A farmer may find a germplasm that behaves well according to informations from model 1 (section 2) in a farm that shares its cluster.

```
p_PCA = get.ggplot(clust, ggplot.type = "PCA")
p_PCA

## $ind
##
## $var
```





3.2.7 Predict the past

In order to choose a new germplasm to test on his farm, a farmer may choose a germplasm according to the value it would have obtained on his farm.

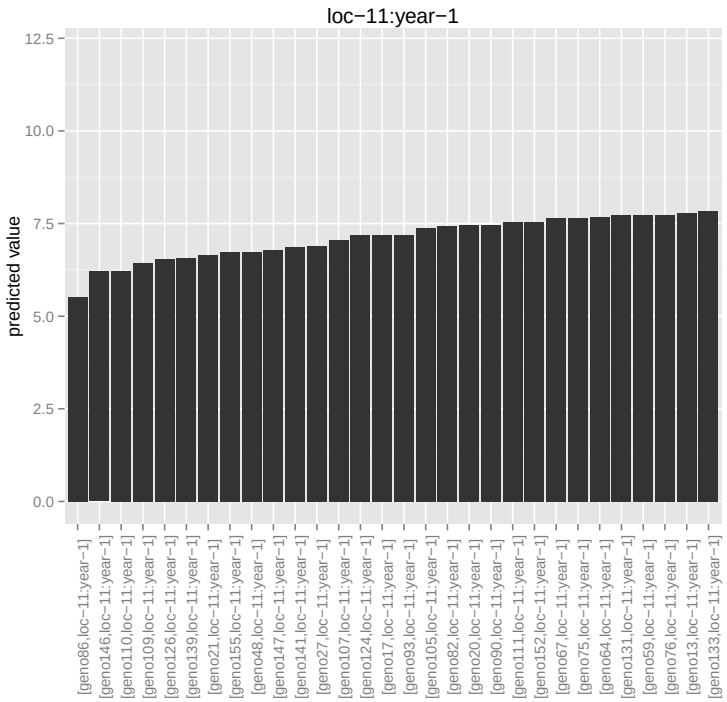
You may either get the estimated MCMC, but you will need lots of memory, or the summary statistics of the MCMC.

Due to memory issues, it may be better to choose output.format = "summary". This allows caterpillar plots but no mean comparisons that are base on the whole MCMC.

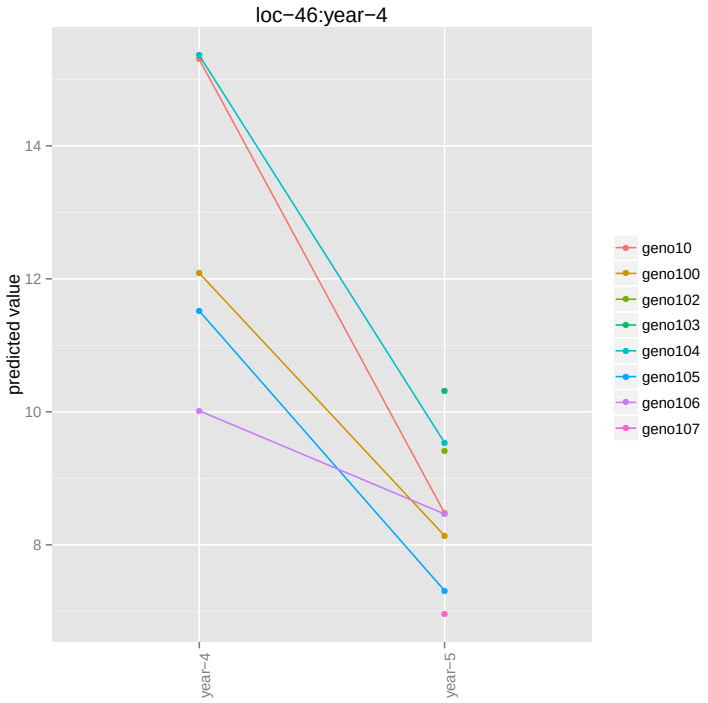
```
# out.predict.the.past = predict.the.past(out2, output.format = "summary")
# |============================================================| 100%
load("./data_PPBstats/out.predict.the.past.RData") # to save time
dim(out.predict.the.past)

## [1] 8140    8
```

If you choose `output.format = "summary"`, it is possible to look at the results with `get.ggplot`.

```
p_barplot_predict = get.ggplot(out.predict.the.past, ggplot.type = "barplot",
                               nb_parameters_per_plot = 30)
p_barplot_predict$`loc-11:year-1`$`1`
```




```
p_interaction_predict = get.ggplot(out.predict.the.past, ggplot.type = "interaction")
p_interaction_predict$`loc-46`$`1`
```


loc−46:year−4
predicted value
14
12
10
8
year−4
year−5
geno10
geno100
geno102
geno103
geno104
geno105
geno106
geno107

To cite PPBstats

To cite this package and or this vignette:

```
citation("PPBstats")

##
## To cite the PPBstats package in publications use:
##
##   Pierre Riviere and Olivier David, 2016, PPBstats: An R
##   package for statistical analysis of unbalanced trials in
##   decentralized participatory plant breeding programmes.
##   Version 0.11.0, URL: https://github.com/priviere/PPBstats
##
## A BibTeX entry for LaTeX users is
##
##   @Manual{,
##     title = {PPBstats: An R package for statistical analysis
##           of unbalanced trials in decentralized participatory plant
##           breeding programmes. Version 0.11.0},
##     author = {{Pierre Riviere and Olivier David}},
##     organisation = {{Reseau Semences Paysannes}, {INRA}},
##     year = {2016},
##     url = {https://github.com/priviere/PPBstats},
##     note = {R code is under licence GPL-3.
##           Vignette is under licence creative commons BY-NC-SA 4.0.},
##   }
```

Aknowledgement

This worked has been first funded by the European Community's Seventh Framework Programme (FP7/9 2007–2013) under the grant agreement n245058-Solibam (Strategies for Organic and Low-input Integrated Breeding and Management). It has been completed by funding from European Union's Horizon 2020 research and innovation programme under grant agreement No 633571 (DIVERSIFOOD project) and Fondation de France.

Thanks to Hadley Wickham for his web site http://r-pkgs.had.co.nz/ that help us a lot in the creation of this package.

Thanks to Isabelle Goldringer for her comments and remarks improving this vignette. Thanks to Gaelle Van Franck for her comments and remarks improving the code of the package.

References

D. Desclaux, J. M. Nolot, Y. Chiffoleau, C. Leclerc, and E. Gozé. Changes in the concept of genotype X environment interactions to fit agriculture diversification and decentralized participatory plant breeding: pluridisciplinary point of view. *Euphytica*, 163:533–546, 2008.

K.W. Finlay and G.N. Wilkinson. The analysis of adaptation in a plant-breeding programme. *Australian Journal of Agricultural Research*, 14(6):742–754, 1963.

A. Gelman and D.B. Rubin. Inference from Iterative Simulation Using Multiple Sequences (with discussion). *Statistical Science*, (7):457–511, 1992.

F. Husson, J. Josse, and J. Pagès. Principal component methods - hierarchical clustering - partitional clustering: why would we need to choose for visualizing data? Technical report, Agrocampus Ouest, 2010.

F. Nabugoomu, R.A. Kempton, and M. Talbot. Analysis of series of trials where varieties differ in sensitivity to locations. *Journal of agricultural, biological and environnmental Statistics*, 4(3):310–325, 1999.

M. Plummer. Penalized loss functions for Bayesian model comparison. *Biostatistics*, 9(3):523–539, July 2008.

P. Rivière, J.C. Dawson, I. Goldringer, and O. David. Hierarchical Bayesian Modeling for Flexible Experiments in Decentralized Participatory Plant Breeding. *Crop Science*, 55(3), 2015.

P. Rivière, J.C. Dawson, I. Goldringer, and O. David. Hierarchical multiplicative modeling of genotype × environment interaction for flexible experiments in decentralized participatory plant breeding. *In prep*, 2016.

C.P. Robert. *The Bayesian Choice*. Springer Texts in Statistics. Springer, second edition, 2001.

D. J. Spiegelhalter, N. G. Best, B. P. Carlin, and A. Van Der Linde. Bayesian measures of model complexity and fit. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 64(4):583–639, 2002.