statgenHTP tutorial

Emilie Millet, Bart-Jan van Rossum

2019-12-17

1 The statgenHTP Package

The statgenHTP package is developed as an easy-to-use package for analyzing data coming from high throughput phenotyping (HTP) platform experiments. The package provides many options for plotting and exporting the results of the analyses. It was developed within the EPPN^{2020} project to meet the needs for automated analyses of HTP data.

New phenotyping techniques enable measuring traits at high throughput, with traits being measured at multiple time points for hundreds or thousands of plants. This requires automatic modeling of the data (F. Tardieu et al. 2017) with a model that is robust, flexible and has easy selection steps.

Phenotyping facilities display spatial heterogeneity. For example, the spatial variability of incident light can go up to 100% between pots within a greenhouse (Cabrera-Bosquet et al. 2016). Taking into account these spatial trends is a prerequisite for precise estimation of genetic and treatment effects. In the same way as in field trials, platform experiments should obey standard principles for experimental design and statistical modeling.

Popular mixed models to separate spatial trends from treatment and genetic effects, rely on the use of autoregressive correlation functions defined on rows and columns (AR1×AR1) to model the local trends (B. R. Cullis, Smith, and Coombes 2006). These models are sometimes difficult to fit and the selection of a best model is complicated, therefore preventing an automated phenotypic analysis of series of trials. An attractive alternative is the use of 2-dimensional P-spline surfaces, the SpATS model (Spatial Analysis of Trials using Splines, (M. Rodríguez-Álvarez et al. 2017)). This model corrects for spatial trends, row and column effects and has the advantage of avoiding the model selection step. It also provides the user with graphical output that are easy to interpret. It has proven to be a good alternative to the classical AR1×AR1 modeling in the field (Velazco et al. 2017). It is also suitable for phenotyping platform data and has been tested on several datasets in the EPPN^{2020} project.

The aim of this package is to accurately separate the genetic effects from the spatial effects at each time point. It will provide user with either genotypic values or corrected values that can be used for further modeling, e.g. time series modeling to extract responses to environment (Eeuwijk et al. 2019). Note that two extensions will be included later for an overall structure of the package in three main parts:
1. Correction for spatial trends - *this version of the statgenHTP*
2. Outliers detection - *a tutorial provided by the Mistea group (INRA, France)*
3. Time course modeling - *extension of this package planned for 2020*

This tutorial describes in detail how to prepare data for analysis, perform analyses to correct for spatial trends using different modeling engines and extract the results from the models.

---

2 Load the statgenHTP R package

For more information on how to install R and RStudio and the statgen packages, please read the Getting started file. Once everything is installed and before functions of the statgenHTP R package can be used within R, it needs to be made available:

```
### Load the statgenHTP package
library(statgenHTP)
```

3 Data preparation

The first step when modeling platform experiment data with the statgenHTP package is creating an object of class `TP` (Time Points). In this object, the time points are split into single data frames. It is then used throughout the statgenHTP package as input for analyses.

> *TO NOTE: It is possible to use the functions of this package with a phenotype measured at one time point only. In that case, the user need to create a column with time point containing the unique measurement time.*

3.1 Creating a `TP` object

A `TP` object can be created from a `data.frame` with the function `createTimePoints`. This function does a number of things:

- Check the input data.
- Rename columns to default column names used by the functions in the statgenHTP package. For example, the column in the data containing variety/accession/genotype is renamed to "genotype". Original column names are stored as an attribute of the individual data frames in the `TP` object.
- Convert column types to the default column types. For example, the column "genotype" is converted to a factor and "rowNum" to a numeric column.
- Convert the column containing time into time format. If needed, the time format can be provided in `timeFormat`. For example, with a date/time input of the form "day/month/year hour:minute", use %d/%m/%Y %H:%M. For a full list of abbreviations see the R package strptime. *TO NOTE* when the input time is just a numeric, the function will convert it to time from 01-01-1970 (origin time of the package lubridate).
- Add columns `check` and `checkGenotypes` when `addCheck=TRUE`.
- Split the data into separate data.frames by time points. A `TP` object is a `list` of `data.frames` where each `data.frame` contains the data for a single time point. If there is only one time points the output will be a `list` with only one item.
- Add a `data.frame` with columns `timeNumber` and `timePoint` as attribute "timePoints" to the `TP` object. This data.frame can be used for referencing the time points by their number.

3.1.1 Example 1

The first example used in this tutorial contains data from an experiment in the Phenovator platform (WUR, Netherlands, (Flood et al. 2016)) with Arabidopsis plants. It consists of one experiment with 1440 plants grown in a growth chamber. The number of tested genotypes is 192 with 6-7 replicates per genotype. Four reference genotypes were also tested with 15 or 30 replicates. The studied trait is the photosystem II efficiency ("EffpsII") extracted from the pictures over time (Rooijen et al. 2017). The dataset called "PhenovatorDat1" is included in the package.

For the example first a `TP` object is created containing all the time points.

```
## Create a TP object containing the data from the Phenovator.
data("PhenovatorDat1")
phenoTP <- createTimePoints(dat = PhenovatorDat1,
                            experimentName = "Phenovator",
                            genotype = "Genotype",
```

```
                            timePoint = "timepoints",
                            repId = "Replicate",
                            plotId = "pos",
                            rowNum = "y", colNum = "x",
                            addCheck = TRUE,
                            checkGenotypes = c("check1", "check2", "check3", "check4"))
summary(phenoTP)
#> phenoTP contains data for experiment Phenovator.
#>
#> It contains 73 time points.
#> First time point: 2018-05-31 16:37:00
#> Last time point: 2018-06-18 16:37:00
#>
#> The following genotypes are defined as check genotypes: check1, check2, check3, check4.
```

The function getTimePoints allows to generate a data frame containing the time points and their numbers in the `TP` object. Below is a example with the first 6 time points of the `phenoTP`:

```
### Extract the time points table
timepoint <- getTimePoints(phenoTP)
```

timeNumber	timePoint
1	2018-05-31 16:37:00
2	2018-06-01 09:07:00
3	2018-06-01 11:37:00
4	2018-06-01 14:37:00
5	2018-06-01 16:37:00
6	2018-06-02 09:07:00

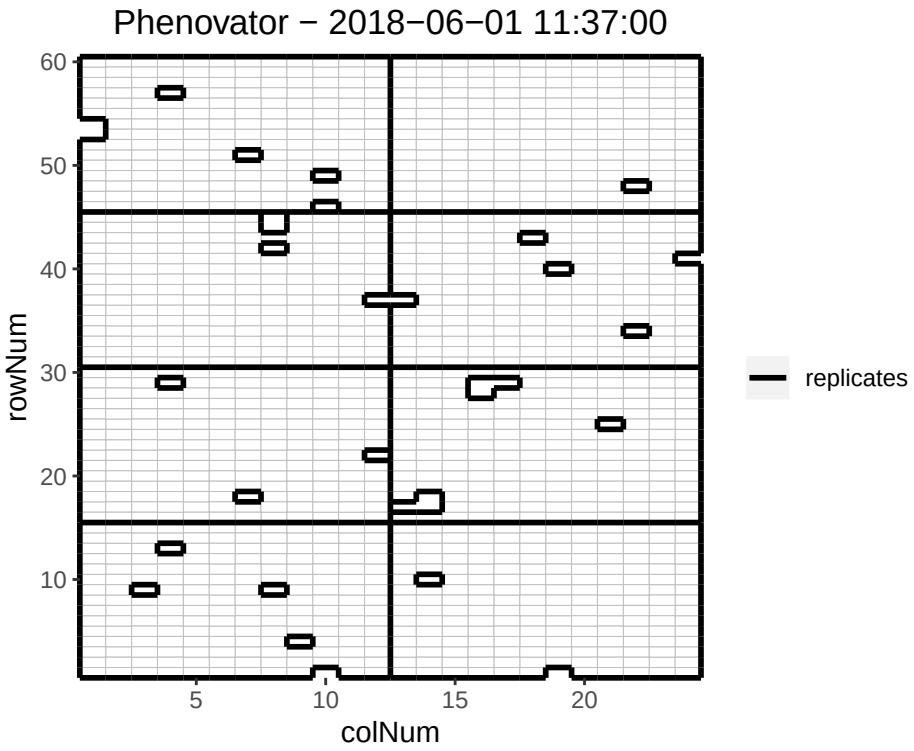
The `TP` object just created is a `list` with 73 items, one for each time point in the original `data.frame` (called "PhenovatorDat1"). The option `experimentName` is used for identifying the dataset and is a requirement. The column "Genotype" in the original data is renamed to "genotype" and converted to a factor. The columns "Replicate" and "pos" are renamed and converted likewise. The newly created column "plotID" needs to be a unique identifier for a plot or a plant. It cannot occur more than once per time point. The columns "y" and "x" are renamed to "rowNum" and "colNum" respectively. Simultaneously, two columns "rowId" and "colId" are created containing the same information converted to a factor. This seemingly duplicate information is needed for spatial analysis. The information about which columns have been renamed when creating a `TP` object is stored as an attribute of each individual `data.frame` in the object. The option `addCheck` is set as `TRUE` to specify that the genotypes listed in `checkGenotypes` are reference genotypes (or check). This option will create a column "check" with a value "noCheck" for the genotypes that are not in `checkGenotypes` and the name of the genotype for the `checkGenotypes`. Also a column "genoCheck" is added with the names of the genotypes that are not in `checkGenotypes` and `NA` for the `checkGenotypes` (see section **4.1**). These columns are necessary for fitting models on data in case of augmented design (Piepho and Williams 2016).

3.2 Plotting a `TP` object

Several plots can be made to further investigate the content of a `TP` object.

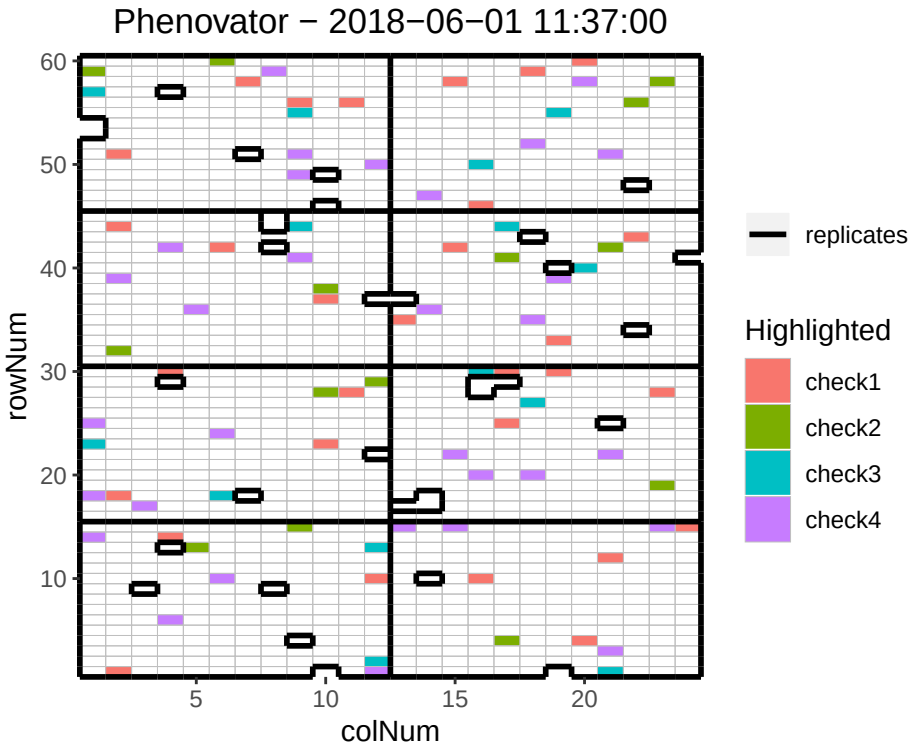
The first type of plot displays the layout of the experiment as a grid using the row and column coordinates in "plotID". The default option creates plots of all time points in the `TP` object. This can be restricted to selected time points using their number in the option `timePoints`. If replicates ("repId") are available, a black line is plotted between them. Missing plots are indicated in white circled with a bold black line.

```
## Plot the layout for the third time point.
plot(phenoTP,
     plotType = "layout",
     timePoints = 3)
```



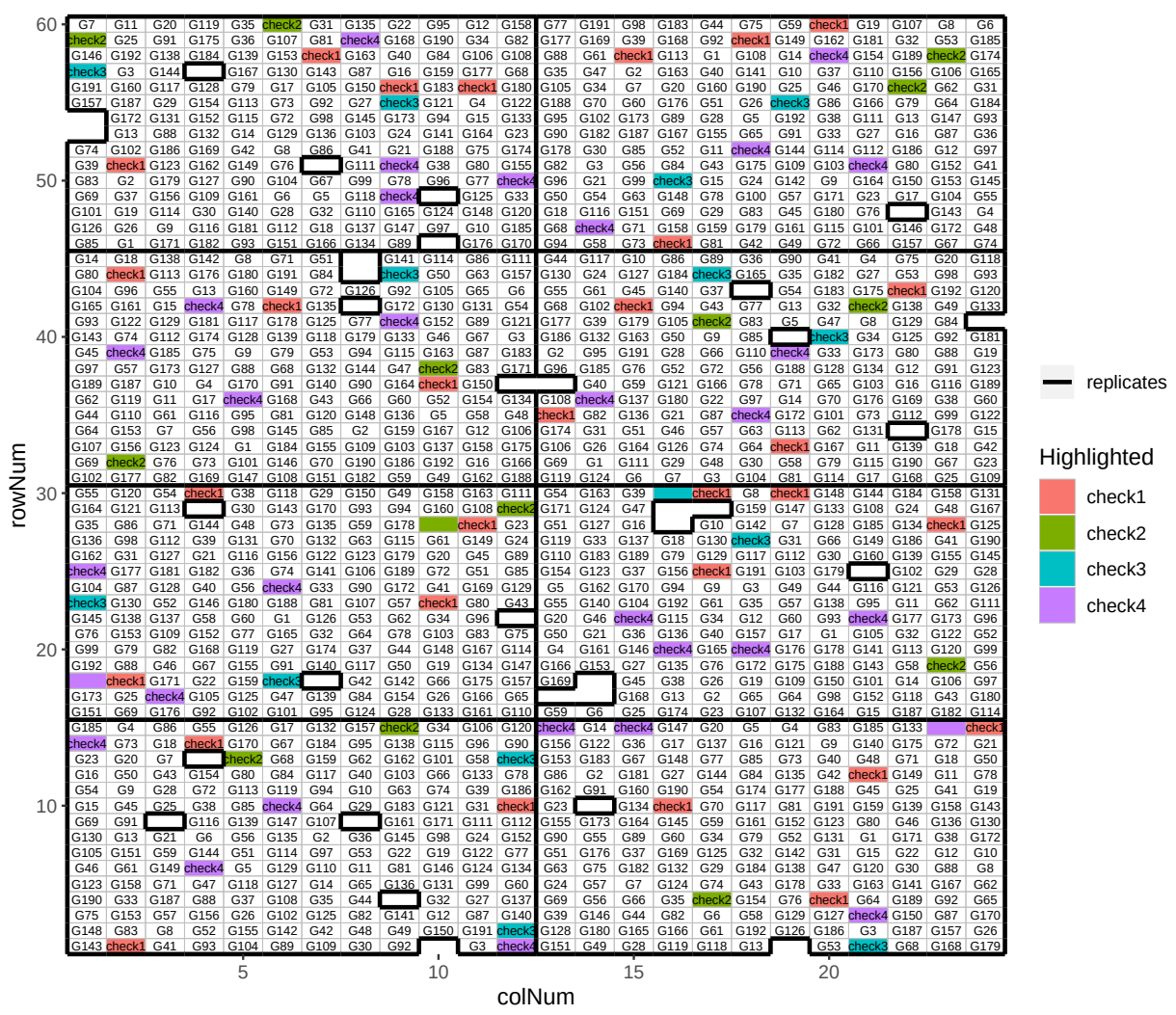
Here, the third time point is displayed which corresponds to the 1st of June at 11:37. Note that the title can be manually changed using the `title` option. This plot can be extended by highlighting interesting genotypes in the layout. Hereafter the check genotypes are highlighted:

```
## Plot the layout for the third time point with the check genotypes highlighted.
plot(phenoTP,
     plotType = "layout",
     timePoints = 3,
     highlight = c("check1", "check2", "check3", "check4"))
```



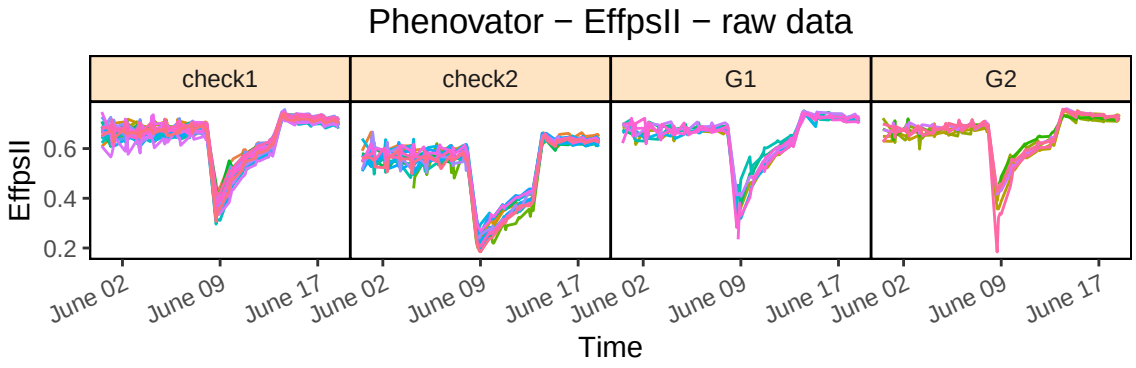
It is possible to add the names of the genotypes to the layout.

```
## Plot the layout for the third time point.
plot(phenoTP,
     plotType = "layout",
     timePoints = 3,
     highlight = c("check1", "check2", "check3", "check4"),
     showGeno = TRUE)
```



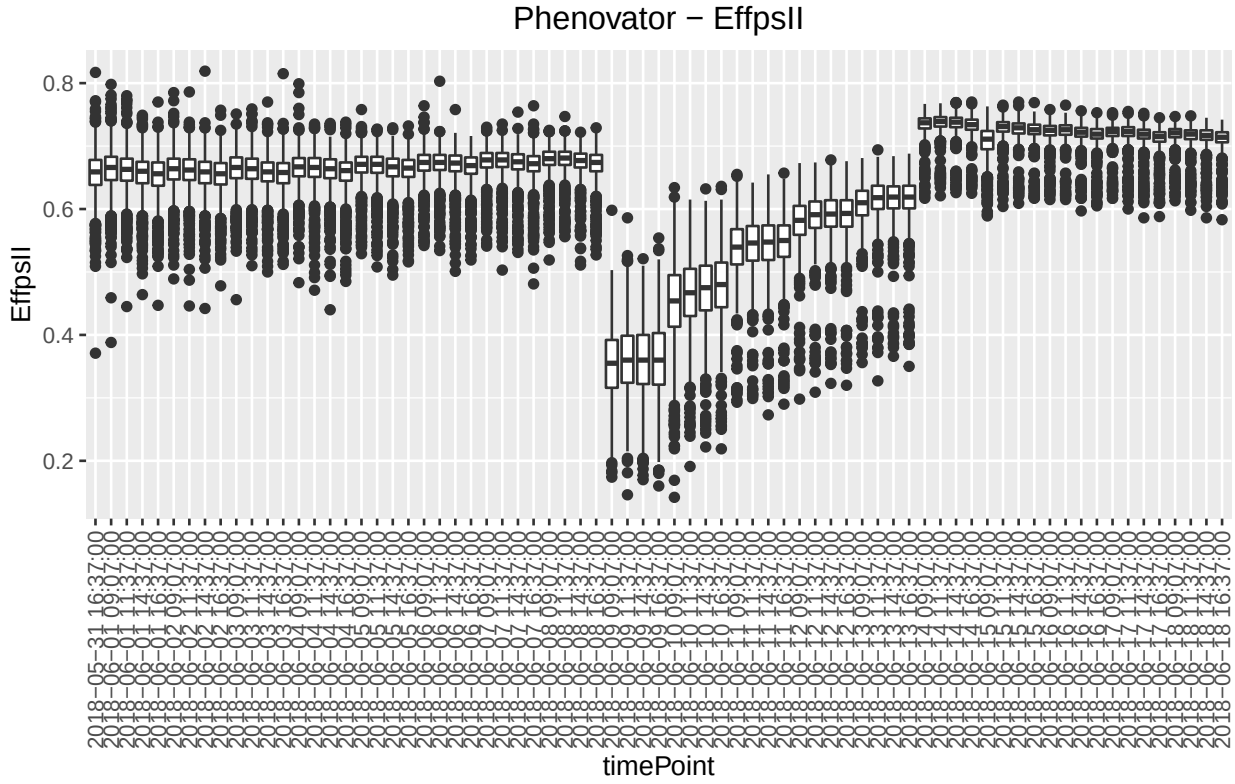
Raw data can be displayed per genotype. By default all genotypes are used but it can be restricted using `genotypes` and a subset of genotypes.

```
## Create the raw data time courses for four genotypes.
plot(phenoTP,
     traits = "EffpsII",
     plotType = "raw",
     genotypes = c("G1", "G2", "check1", "check2"))
```



Boxplots can be made to get an idea of the total variability of the trait in the `TP` object. By default a box is plotted per time point for the specified trait using all time points.

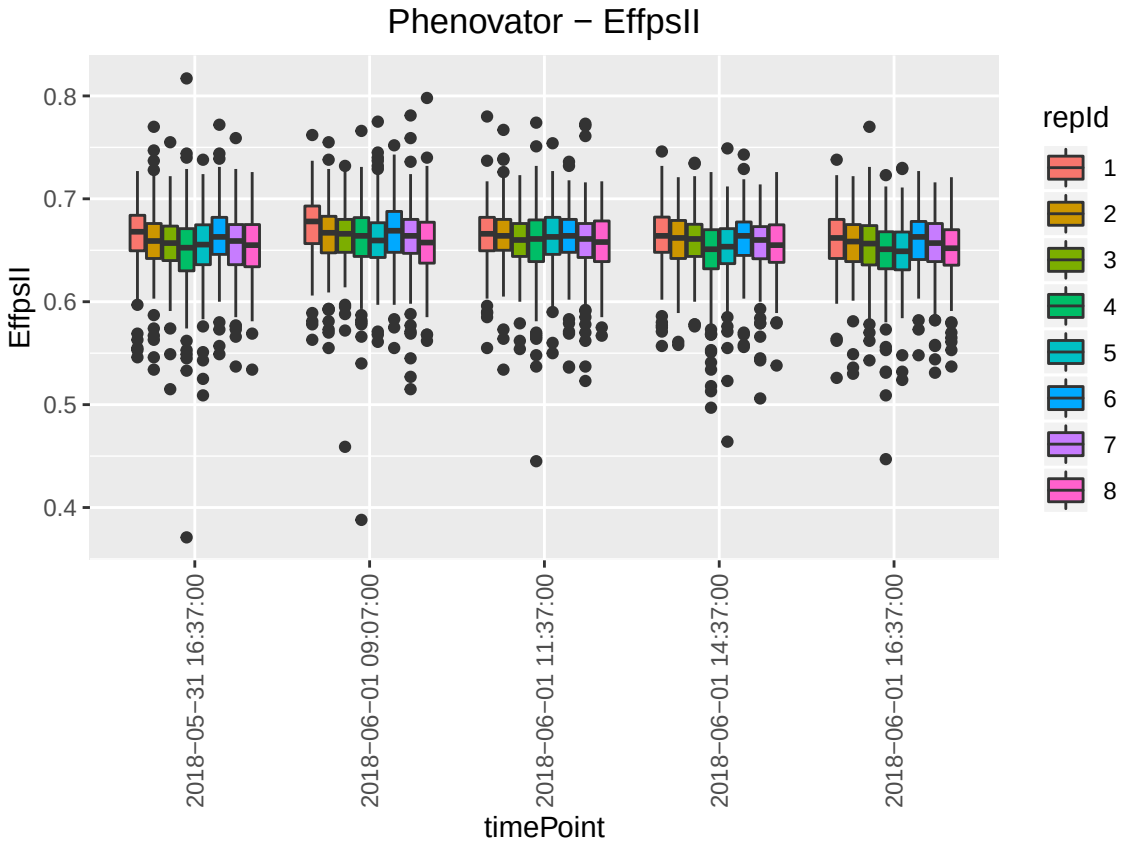
```
## Create a boxplot for "EffpsII" using the default all time points.
plot(phenoTP,
     plotType = "box",
     traits = "EffpsII")
```



Colors can be applied to groups within time points using the option `colorBy`. The boxes for the (groups of) time points can be ordered using `orderBy`. Boxes can be ordered by alphabetical order ("alphabetic") or by the group mean ("ascending","descending").

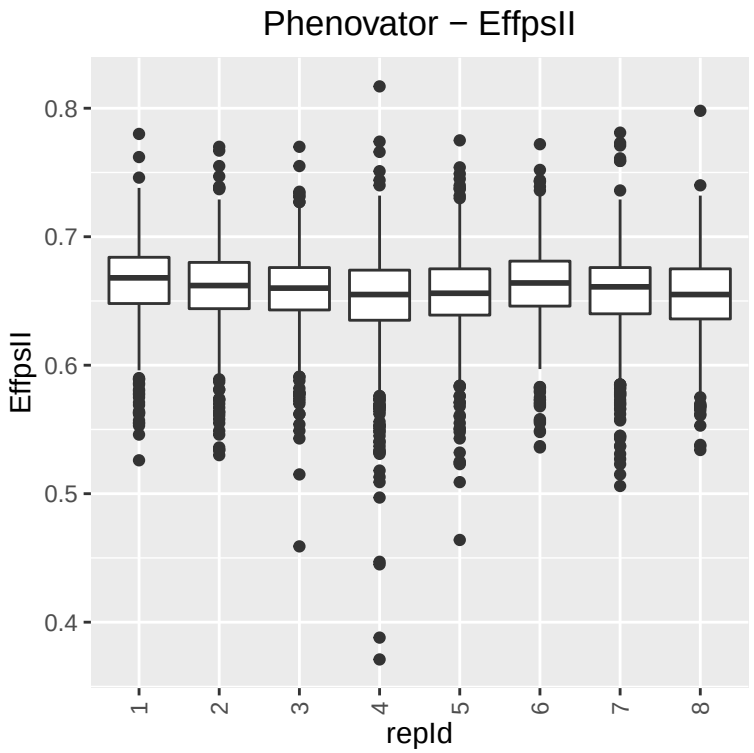
```
## Create a boxplot for "EffpsII" with 5 time points and boxes colored by "repId" within
## time point.
plot(phenoTP,
     plotType = "box",
```

```
     traits = "EffpsII",
     timePoints = 1:5,
     colorBy = "repId")
```



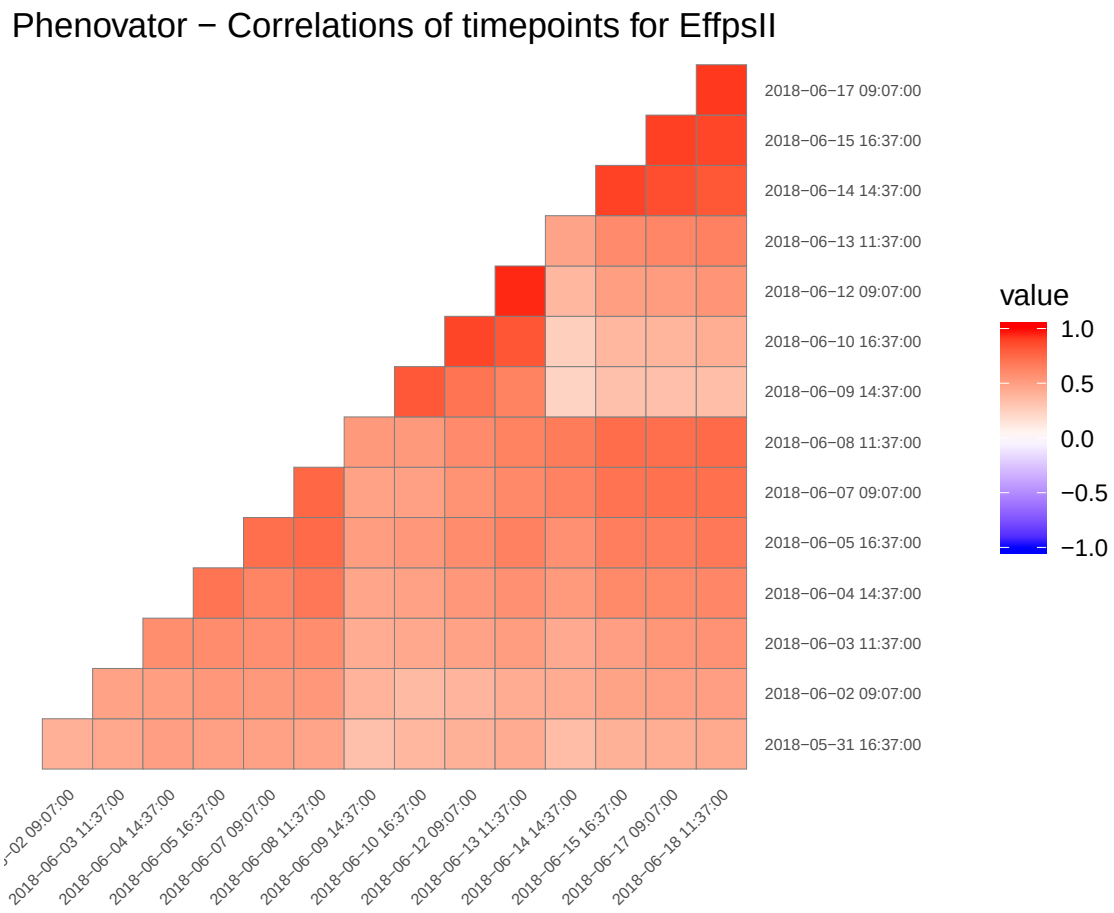
The time points in the boxplot can be grouped using the option `groupBy`.

```
## Create a boxplot for "EffpsII" with 5 time points and boxes grouped by "repId".
plot(phenoTP,
     plotType = "box",
     traits = "EffpsII",
     timePoints = 1:5,
     groupBy = "repId")
```



The last plot that can be made is a plot of the correlations between the time points for a specified trait. The order of the plot is the time and by default all time points are used.

```
## Create a correlation plot for "EffpsII" for a selection of time points.
plot(phenoTP,
     plotType = "cor",
     traits = "EffpsII",
     timePoints = seq(from = 1,to = 73,by = 5))
```



Eeach plot can be exported to a pdf document by using the `outFile` option containing the name of the document.

---

4 Modeling

After creating a `TP` object, a model can be fitted on the data. This is done using the function `fitModels`, which uses two different engines for fitting the models, namely *SpATS* (M. Rodríguez-Álvarez et al. 2017) and *ASReml* (Butler et al. 2017). For models with row and column coordinates, SpATS is the default engine (see part **4.1**). This can be overruled by specifying the function parameter `engine` and using asreml for spatial models (see part **4.2**). When the row and column coordinates are not available, asreml is used for modeling without spatial components (see part **4.3**). Finally, it is possible to decompose the genotypic variance using, for example, a treatment effect (see part **4.4**).

The output of `fitModels` is an object of class `fitMod`, a list of fitted models with one item for each time point the model was fitted for.

4.1 Spatial model using SpATS

When SpATS is used for modeling, an extra spatial term is included in the model. This spatial component is composed using the `PSANOVA` function in the SpATS package which uses 2-dimensional smoothing with

P-splines as described in Lee, Durbán, and Eilers (2013) and in M. Rodríguez-Álvarez et al. (2017). See `help(PSANOVA, SpATS)` for a detailed description. Extra fixed effects may be fitted using the option `extraFixedFactors`. The model can also be fitted following a resolvable row-column design setting `useRepId` as `TRUE`. The model specifications are listed in the table below.

option	model fitted	spatial term
default	trait = **genotype** + ϵ	PSANOVA
extraFixedFactors = c("A","B")	trait = A + B + **genotype** + ϵ	PSANOVA
useRepId = TRUE	trait = *repId* + **genotype** + **repId:rowId** + **repId:colId** + ϵ	PSANOVA
useCheck = TRUE	trait = *check* + **genoCheck** + ϵ	PSANOVA

In the models above, fixed effects are indicated in *italics* whereas random effects are indicated in **bold**. "genotype" can be fitted as **random** or *fixed* effect using the option `what`. The same model will run with genotype as fixed. The option `useCheck` allows treating some genotypes as check: it splits the column "genotype" into two columns as follows:

genotype	check	genoCheck
G_1	noCheck	G_1
G_2	noCheck	G_2
...	noCheck	...
G_{n-1}	noCheck	G_{n-1}
G_n	noCheck	G_n
$check_1$	$check_1$	NA
$check_2$	$check_2$	NA
...	...	...
$check_{m-1}$	$check_{m-1}$	NA
$check_m$	$check_m$	NA

Note that it is only possible to use the combination of check and genotype as random.

4.1.1 Calling SpATS

Using the `TP` object "phenoTP" from the previous section, a model for a few time points and trait "EffpsII" can now be fitted on the data as follows. Since `engine` is not supplied as an option, SpATS is used for fitting the following model:
EffpsII = **genotype** + **rowId** + **colId** + ϵ

```
## Fit a model for a few time points.
modPhenoSp <- fitModels(TP = phenoTP,
                        trait = "EffpsII",
                        timePoints = seq(from = 1, to = 73, by = 5))
summary(modPhenoSp)
#> Models in modPhenoSp where fitted for experiment Phenovator.
#>
#> It contains 15 time points.
#> The models were fitted using SpATS.
```

The output is a `fitMod` object, a list containing one fitted model per time point. Note that by not supplying the `what` argument to the function, genotype is set as random.

It can be run again with genotype as fixed using `what`:
EffpsII = *genotype* + **rowId** + **colId** + ϵ

```
## Fit a model for a single time point.
modPhenoSpFix <- fitModels(TP = phenoTP,
                           trait = "EffpsII",
                           timePoints = 3,
                           what = "fixed")
```

The model can be extended by including extra main fixed effects:
EffpsII = *repId* + *Image_pos* + **genotype** + **rowId** + **colId** + ϵ

```
## Fit a model for a single time point with extra fixed factors.
modPhenoSpCov <- fitModels(TP = phenoTP,
                           trait = "EffpsII",
                           extraFixedFactors = c("repId", "Image_pos"),
                           timePoints = 3)
```

It can also be extended by including check genotypes:
EffpsII = *repId* + *Image_pos* + *check* + **genoCheck** + **rowId** + **colId** + ϵ

```
## Fit a model for a single time point with extra fixed effects and check   genotypes.
modPhenoSpCheck <- fitModels(TP = phenoTP,
                             trait = "EffpsII",
                             extraFixedFactors = c("repId", "Image_pos"),
                             useCheck = TRUE,
                             timePoints = 3)
```

Finally, a model following a resolvable row-column design can be fitted: including the interactions between replicate and row-col.
EffpsII = *repId* + **genoCheck** + **repId:rowId** + **repId:colId** + ϵ

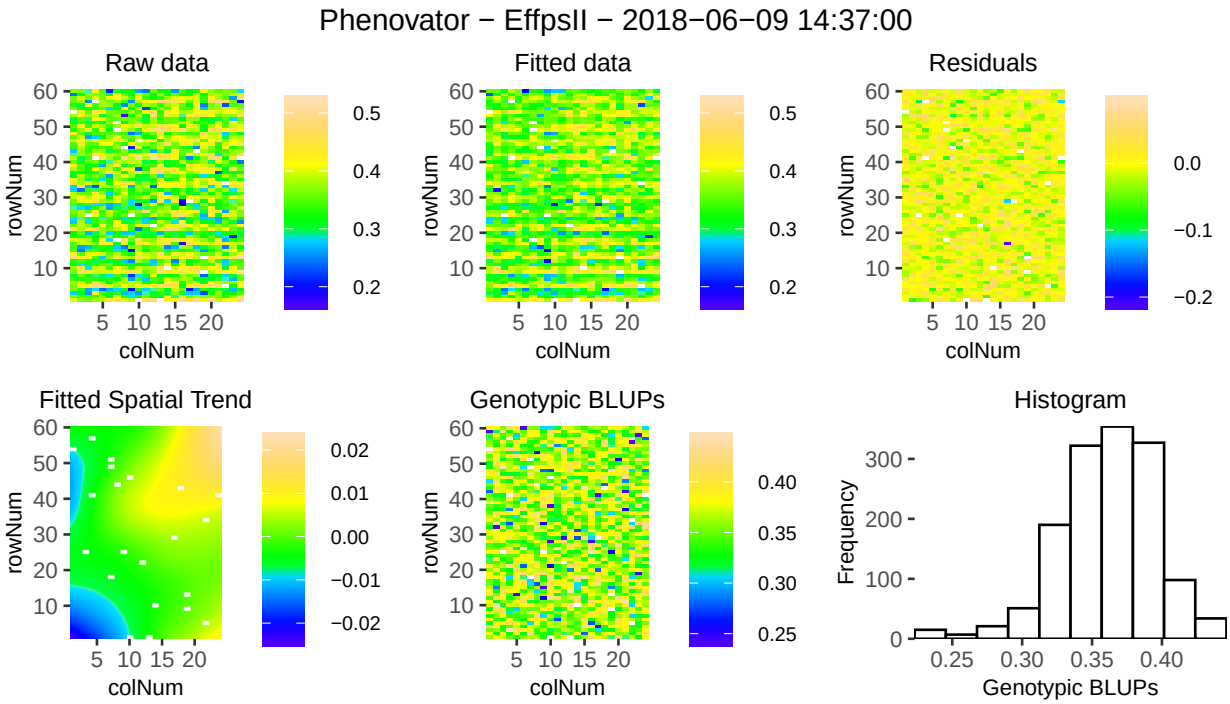
```
## Fit a model for a single time point.
modPhenoSpRCD <- fitModels(TP = phenoTP,
                           trait = "EffpsII",
                           timePoints = 3,
                           useRepId = TRUE)
```

4.1.2 Model plots

The first type of plot that can be made for fitted models, is a spatial plot per time point using `plotType = "spatial"`. It consists of five plots, spatial plots of the raw data, fitted values, residuals and either BLUEs or BLUPs, and a histogram of the BLUEs or BLUPs. When SpATS is used for modeling an extra plot with the fitted spatial trend is included.

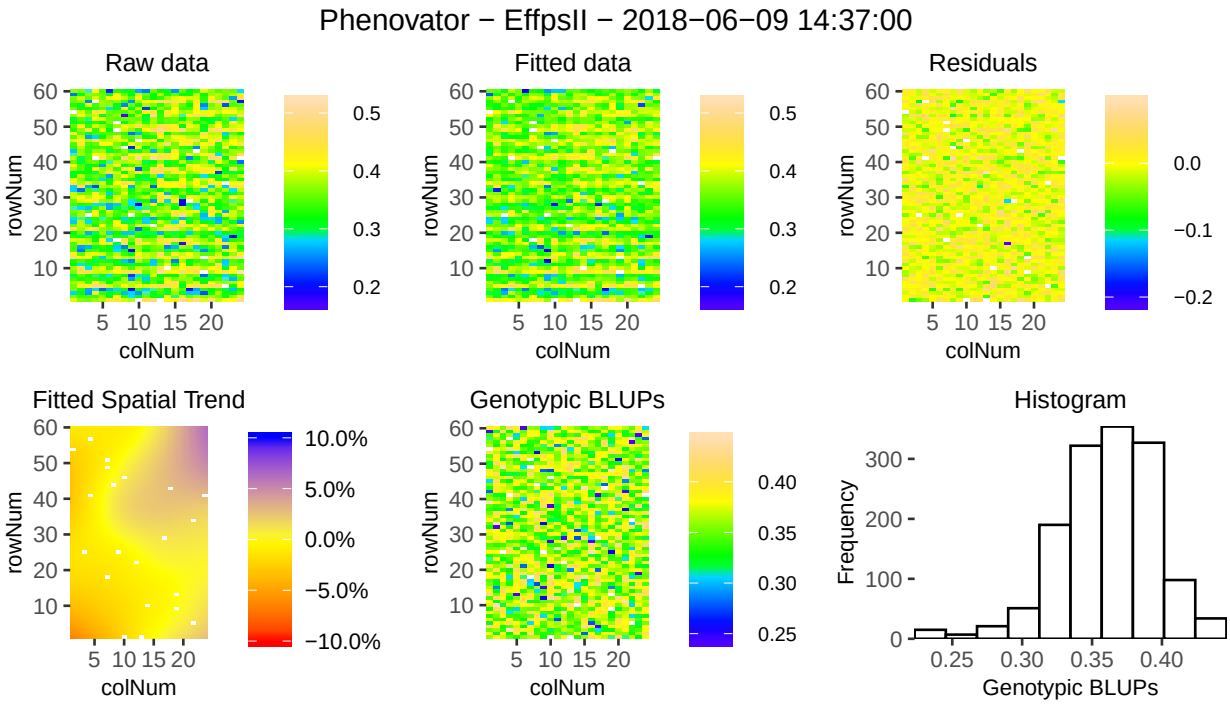
Note that spatial plots can only be made if spatial information, i.e. "rowNum" and "colNum", is available in the `TP` object.

```
plot(modPhenoSp,
     timePoints = 36,
     plotType = "spatial",
     spaTrend = "raw")
```



For a straighforward interpretation of the fitted spatial trends and for comparison between time points, the plot of the fitted spatial trend can be displayed as a ratio of the raw phenotypic mean: SpatTrend(proportion) = Estimated SpatTrend / mean(raw EffpsII). In this case, the scale will be in percentage and the min/max will be adjusted based on all the time points used but will be at least 10%. This empirical threshold allows visualising fitted trends that have a relatively small to large importance.

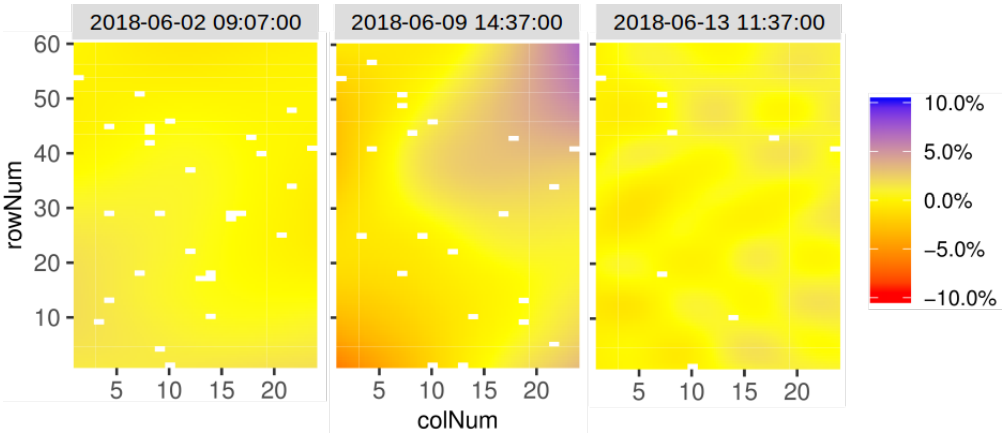
```
plot(modPhenoSp,
     timePoints = 36,
     plotType = "spatial",
     spaTrend = "percentage")
```



It is also possible to create a time lapse of the ratio of spatial trends over time. The scale is the same as previoulsy described. The time lapse is always written to an ouput file.

```
plot(modPhenoSp,
     plotType = "timeLapse",
     outFile = "TimeLapse_modPhenoSp.gif")
```

Here is an illustration with three time points:



4.1.3 Extracting model results

All results that can be extracted are shown in the table below. The first column shows what function needs to be called in order to be able to extract the result. The second column gives a short description of the result that will be extracted and, where needed, also states for which modeling engines it can be extracted.

function	description
getGenoPred	Best Linear Unbiased Predictions (BLUPS, genotype as random) or Estimators (BLUEs, genotype as fixed)
getCorrected	Spatially corrected values at the experimental unit level
getVar	Variance components
getHerit	Generalised heritabilities - only when genotype is random
getEffDims	Extract effective dimensions - only for SpATS engine

By default, all the functions run for all the time points. It is possible to select some of them using `timePoints`. The ratio of the effective dimensions can also be extracted using `EDType = "ratio"` in the `getEffDims` function.

The output of the function `getGenoPred` is a list of two dataframes: "genoPred" which contains the predicted values for all tested genotypes and "checkPred" which contains the predicted values of the check genotypes, when `useCheck = TRUE` in the model. "checkPred" is empty when `useCheck = FALSE`.

```
## Extract the genotypic predictions for one time point:
genoPredSp <- getGenoPred(modPhenoSp, timePoints = 6)
## Extract the corrected values for one time point:
spatCorrSp <- getCorrected(modPhenoSp, timePoints = 6)
## Extract model components:
varianceSp <- getVar(modPhenoSp)
heritSp    <- getHerit(modPhenoSp)
effDimSp   <- getEffDims(modPhenoSp)
```

The genotypic predictions of the test genotypes for one time point are displayed in a table like the following:

timeNumber	timePoint	genotype	predicted.values	standard.errors
6	2018-06-02 09:07:00	check1	0.6659389	0.0043077
6	2018-06-02 09:07:00	check2	0.5895457	0.0059189
6	2018-06-02 09:07:00	check3	0.6674739	0.0057894
6	2018-06-02 09:07:00	check4	0.6745296	0.0042475
6	2018-06-02 09:07:00	G1	0.6664631	0.0077036
6	2018-06-02 09:07:00	G10	0.6517828	0.0077096

The corrected values are obtained by subtracting the estimated sources of variation which are of no interest to the raw data:
$CorrData = RawData - EstSpatial - EstRow - EstCol - EstCovar$
This allows keeping the data at the experimental unit level (plants) and having more degrees of freedom for further modeling (e.g. time course modeling and estimation of the time course parameter(s)).

> *TO NOTE: all the estimated fixed effects included in* ***`extraFixedFactors`*** *are removed from the corrected phenotype (EstCovar).*

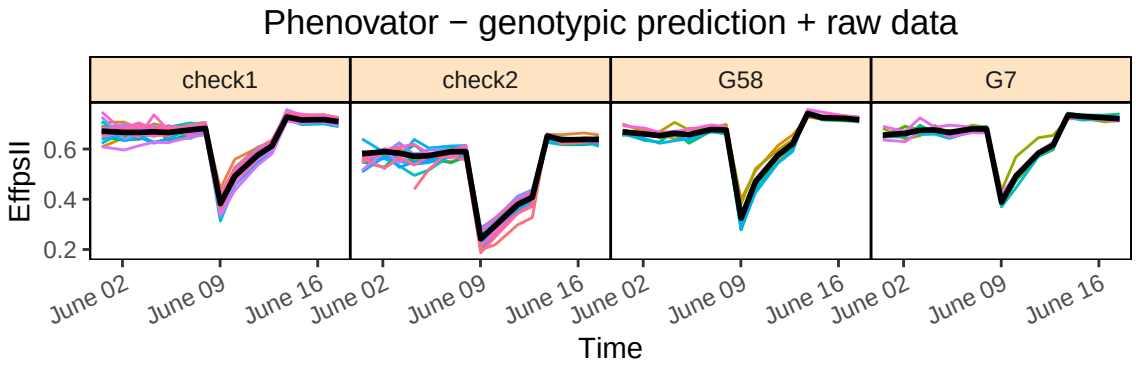
The corrected values of one time point are displayed in a table like the following:

timeNumber	timePoint	EffpsII_corr	EffpsII	genotype	colId	rowId	plotId
6	2018-06-02 09:07:00	0.6549852	0.673	G15	1	10	c1r10
6	2018-06-02 09:07:00	0.6783737	0.691	G121	10	10	c10r10
6	2018-06-02 09:07:00	0.6573910	0.669	G31	11	10	c11r10
6	2018-06-02 09:07:00	0.6952197	0.707	check1	12	10	c12r10
6	2018-06-02 09:07:00	0.6869579	0.698	G23	13	10	c13r10
6	2018-06-02 09:07:00	0.6735733	0.684	G134	15	10	c15r10

4.1.4 Plotting model results

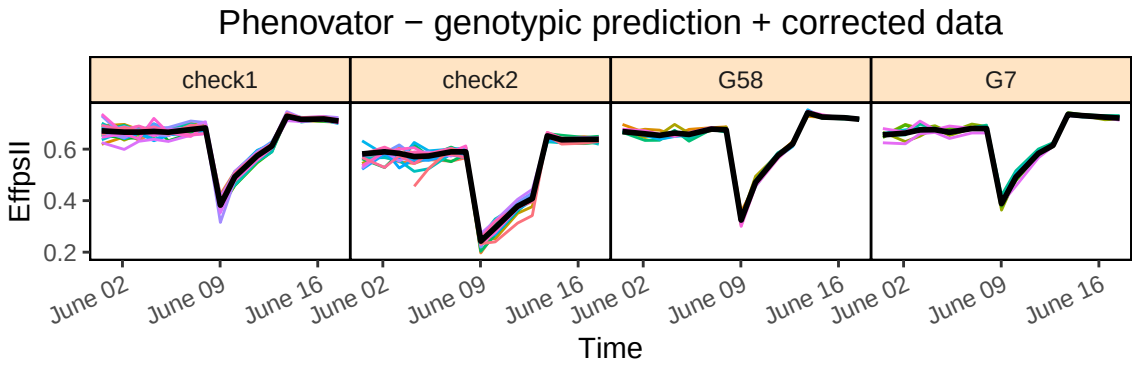
Different plots can be displayed for the `fitMod` object. The first one is "rawPred", it plots the raw data (colored lines) overlayed with the predicted values (black line) from the fitted model. For each genotype a plot is made per plot/plant over time. These plots are put together in a 5×5 grid.
By using the option `genotypes`, a subset of genotypes will be plotted.

```
plot(modPhenoSp,
     plotType = "rawPred",
     genotypes = c("check1", "check2", "G7", "G58"))
```



The second one is "corrPred", it plots the spatially corrected data (colored lines) overlayed with the predicted values from the fitted model (black line). For each genotype a plot is made per plot/plant over time. These plots are put together in a 5×5 grid. By using the option `genotypes`, a subset of genotypes will be plotted.

```
plot(modPhenoSp,
     plotType = "corrPred",
     genotypes = c("check1", "check2", "G7", "G58") )
```

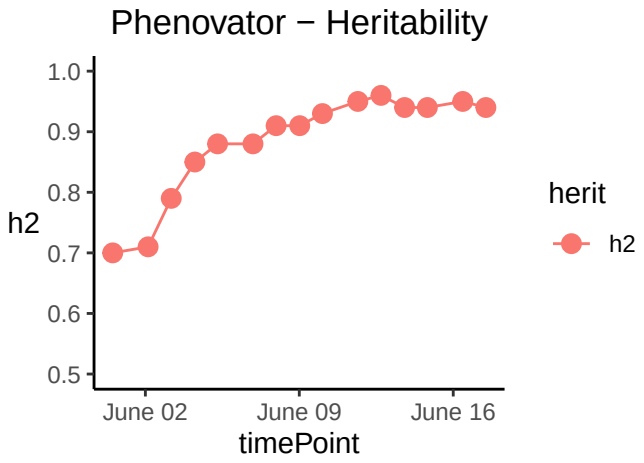


Note that when check genotypes are used for modelling, for the two previous plotType ("rawPred" and "corrPred"), the option `plotChecks` is required to display check genotypes.

```
plot(modPhenoSpCheck,
     plotType = "rawPred",
     plotChecks = TRUE,
     genotypes = c("check1", "check2", "G7", "G58"))
```

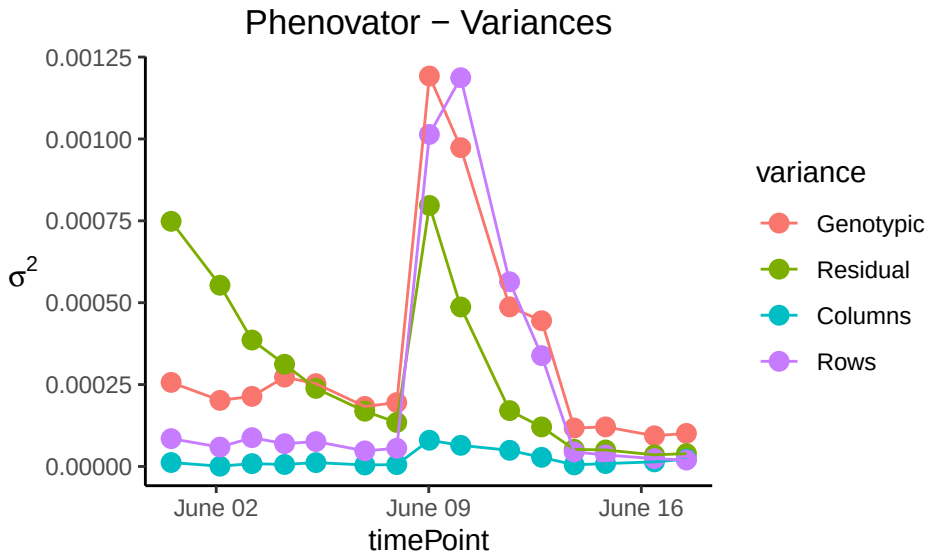
The three last types of plot display different model parameters over time. Plot type "herit" plots the heritability over time. If `geno.decomp` is used when fitting the model, heritabilities are plotted for each level of variance decomposition in a single plot (see section **3.4**). The scale of the plot can be adjusted using `yLim`.

```
plot(modPhenoSp,
     plotType = "herit",
     yLim = c(0.5,1))
```



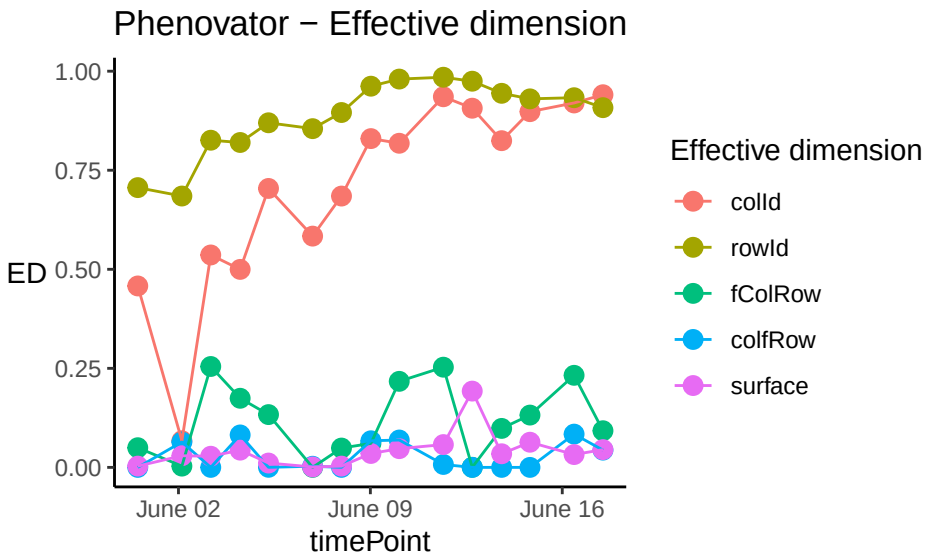
Plot type "variance" plots the residual, column and row variances over time.

```
plot(modPhenoSp,
     plotType = "variance")
```



Plot type "effDim" plots the effective dimension from model fitted using SpATS over time. By default, all the spatial components are plotted. This can be restricted using the option `whichED`.

```
plot(modPhenoSp,
     plotType = "effDim",
     whichED = c("colId", "rowId", "fColRow","colfRow", "surface"),
     EDType = "ratio")
```



The effective dimensions are also known as the effective degrees of freedom. They can be interpreted as a measure of the complexity of the corresponding component: if the effective dimension of one component is large, it indicates that there are strong spatial trends in this direction. Here, for better comparison between components, the ratio of effective dimensions vs. total dimensions can be used. It has a value between 0, no spatial trend, and 1, strong spatial trend (almost all the degrees of freedom are used to model it).

4.2 Spatial model with ASReml

When asreml is used for modeling and `spatial = TRUE`, four models are fitted with different random terms and covariance structures. The best model is determined based on a goodness-of-fit criterion, AIC, on 20% on the time points or at least 10 time points. The best model is then run on all time points. As for SpATS, all the asreml models can be extended by fitting extra fixed factors using the option `extraFixedFactors`.

Note that for the moment, it is only running with asreml-R version 4 only.

option	model fitted	spatial term
spatial = TRUE	trait = **genotype** + **row** + **col** + ϵ	ar1(rowId):ar1(colId)
	trait = **genotype** + **row** + ϵ	ar1(rowId):colId
	trait = **genotype** + **col** + ϵ	rowId:ar1(colId)
	trait = **genotype** + **row** + **col** + ϵ	-
spatial = TRUE, extraFixedFactors = c("A","B")	trait = A + B + **genotype** + **row** + **col** + ϵ	ar1(rowId):ar1(colId)
	trait = A + B + **genotype** + **row** + ϵ	ar1(rowId):colId
	trait = A + B + **genotype** + **col** + ϵ	rowId:ar1(colId)
	trait = A + B + **genotype** + **row** + **col** + ϵ	-
spatial = TRUE, repID = TRUE	trait = *repId* + **genotype** + **repId:row** + **repId:col** + ϵ	ar1(rowId):ar1(colId)
	trait = *repId* + **genotype** + **repId:row** + ϵ	ar1(rowId):colId
	trait = *repId* + **genotype** + **repId:col** + ϵ	rowId:ar1(colId)
	trait = *repId* + **genotype** + **repId:row** + **repId:col** + ϵ	-

In the models above, fixed effects are indicated in *italics* whereas random effects are indicated in **bold**.

"genotype" can be fitted as **random** or *fixed* effect using the option `what`. The same model will run with genotype as fixed. The option `useCheck` is not displayed in the table but works the same as for SpATS: treating some genotypes as check (see section **3.1** for details).

Calling asreml is done by changing the `engine` option in the `fitModels` function.

```
if (requireNamespace("asreml", quietly = TRUE)) {
## Fit a model on few time points with spatial function:
modPhenoSpAs <- fitModels(TP = phenoTP,
                          trait = "EffpsII",
                          timePoints = seq(from = 1, to = 73, by = 5),
                          engine = "asreml",
                          spatial = TRUE)
summary(modPhenoSpAs)
}
#> Models in modPhenoSpAs where fitted for experiment Phenovator.
#>
#> It contains 15 time points.
#> The models were fitted using asreml.
#>
#> The selected spatial model is AR1(x)AR1.
#> 10 time points were used to select the best spatial model.
```

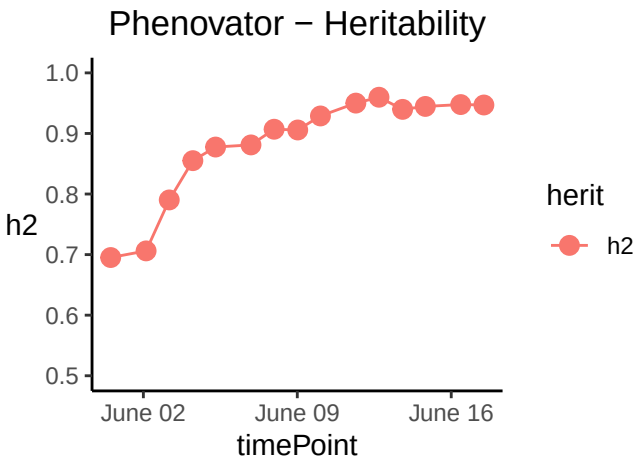
Here the best spatial model is: trait = **genotype** + **row** + **col** + ϵ, with a spatial component: **ar1(rowId):ar1(colId)**. It has been selected using 10 time points.

Plotting and extracting results is then done the same way as for SpATS. Below are few examples.

```
if (requireNamespace("asreml", quietly = TRUE)) {
 spatCorrSpAs <- getCorrected(modPhenoSpAs, timePoints = 6)
}
```

timeNumber	timePoint	EffpsII_corr	EffpsII	genotype	rowId	colId	plotId
6	2018-06-02 09:07:00	0.6605319	0.673	G15	10	1	c1r10
6	2018-06-02 09:07:00	0.6781434	0.691	G121	10	10	c10r10
6	2018-06-02 09:07:00	0.6573399	0.669	G31	10	11	c11r10
6	2018-06-02 09:07:00	0.6939129	0.707	check1	10	12	c12r10
6	2018-06-02 09:07:00	0.6858002	0.698	G23	10	13	c13r10
6	2018-06-02 09:07:00	0.6719760	0.684	G134	10	15	c15r10

```
if (requireNamespace("asreml", quietly = TRUE)) {
plot(modPhenoSpAs,
     plotType = "herit",
     yLim = c(0.5, 1))
}
```



Note that when the engine is asreml, the heritability is calculated using the formula provided in (B. R. Cullis, Smith, and Coombes 2006).

4.3 Modeling without spatial terms with ASReml

When the row and column coordinates are not available, only asreml can be used for modeling. In that case, the model simply uses the genotype and the extraFixedFactors, if any.

option	model fitted	spatial term
spatial = FALSE	trait = **genotype** + ϵ	-
spatial = FALSE, extraFixedFactors = c("A","B")	trait = A + B + **genotype** + ϵ	-

In the models above and below, fixed effects are indicated in *italics* whereas random effects are indicated in **bold**. "genotype" can be fitted as **random** or *fixed* effect using the option `what`. The same model will run with genotype as fixed. The option `useCheck` is not displayed in the table, but works the same as for SpATS (see section **3.1**).

```
## Fit a model on few time points without spatial function.
modPhenoAs <- fitModels(TP = phenoTP,
                        trait = "EffpsII",
                        timePoints = seq(from = 1, to = 73, by = 5),
                        engine = "asreml",
                        spatial = FALSE)
```

4.4 Modeling with variance decomposition

When an experimental treatment is applied, for example, a water scenario, it is required to decompose the genotypic variance into the levels of the treatment. In the following example, two water scenarios, well-watered (WW) and water deficit (WD), were applied. Thus, instead of modeling one genotypic variance, there will be one genotypic variance per treatment. In this part, we will describe briefly what is specific of such modeling with the statgenHTP package using a another example dataset.

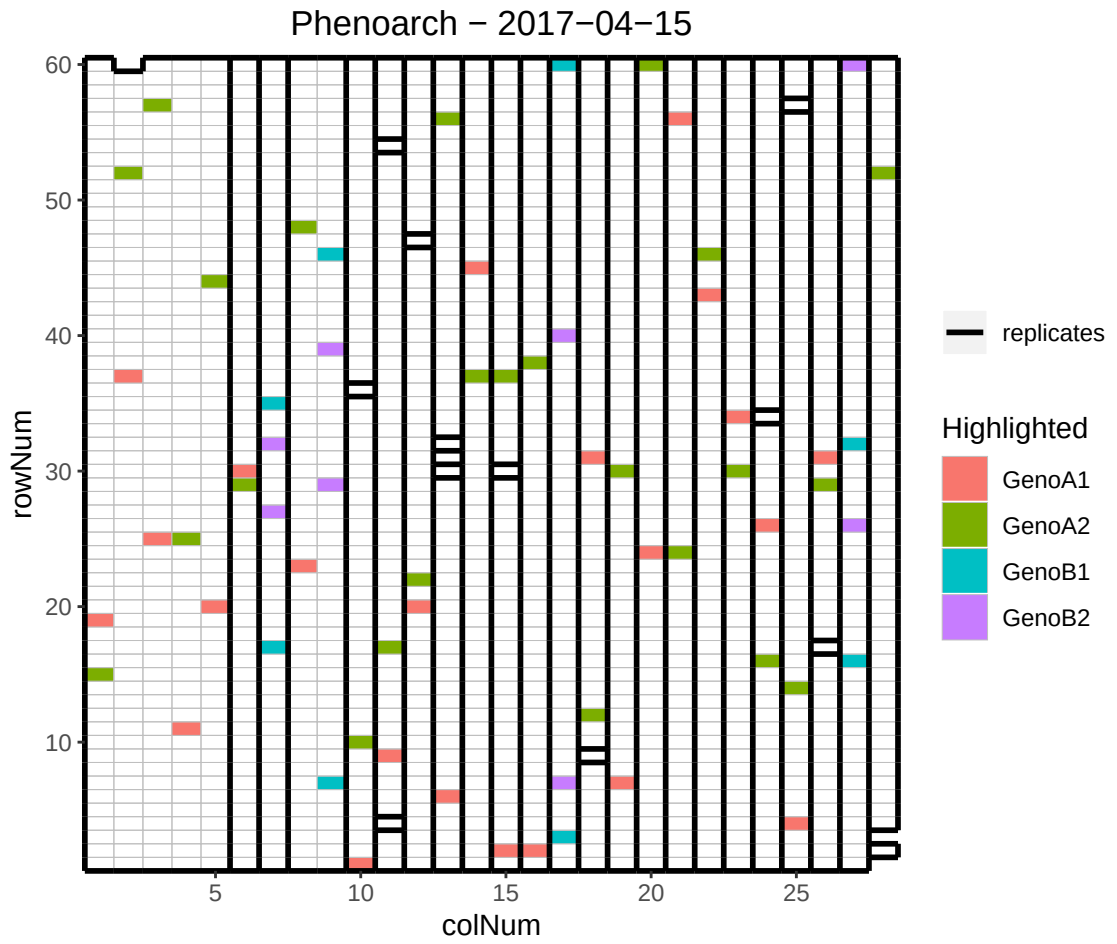
4.4.1 Example 2

The second example used in this tutorial contains data from an experiment in the Phenoarch platform with maize plants (INRA, France, (Cabrera-Bosquet et al. 2016)). It consists of one experiment with 1671 plants grown in a greenhouse. There are two populations of genotypes and two water scenarios, well-watered (WW) and water deficit (WD). The first panel contains 60 genotypes with 14 replicates: 7 in WW and 7 in WD. Note that there are more plants per replicate than one for the first panel (about 24 plants per genotype). The second population contains 30 genotypes with 8 replicates, 4 in WW and 4 in WD. The studied trait is the leaf area ("LA_Estimated") extracted from the pictures over time. Plants were pictured every day for 35 days.

A second `TP` object is created containing all the time points:

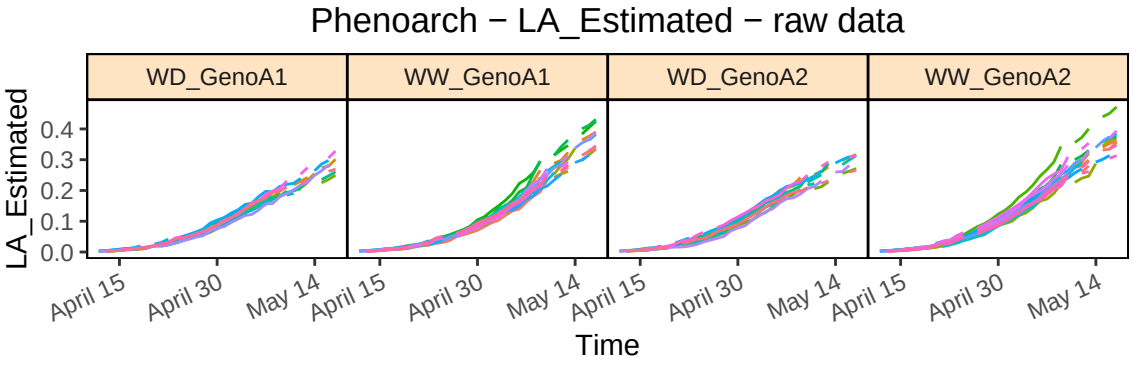
```
data("PhenoarchDat1")
phenoTParch <- createTimePoints(dat = PhenoarchDat1,
                                experimentName = "Phenoarch",
                                genotype = "geno",
                                timePoint = "Date",
                                repId = "Rep",
                                plotId = "pos",
                                rowNum = "Position",
                                colNum = "Line")
summary(phenoTParch)
#> phenoTParch contains data for experiment Phenoarch.
#>
#> It contains 35 time points.
#> First time point: 2017-04-13
#> Last time point: 2017-05-17
#>
#> No check genotypes are defined.
```

The "phenoTParch" object just created is a `list` with 35 items, one for each time points in the original `data.frame` (called "PhenoarchDat1"). We can visualize the layout and the raw data the same way as for the Phenovator data.



Note that for the raw data, we can already use the geno.decomp option to split the genotypes using the water scenario:

```
plot(phenoTParch,
     traits = "LA_Estimated",
     plotType = "raw",
     genotypes = c("GenoA1", "GenoA2"),
     geno.decomp = c("Scenario"))
```



4.4.2 Modeling

Using the `geno.decomp` option when modeling will have two main consequences in the model: the `geno.decomp` variable(s) will be added as main fixed effect(s) and also as interaction(s) with the genotype effect (and the extraFixedFactors and/or check, if any, with spats only). The table below provides one example with three models:

option	engine	model fitted	spatial term
geno.decomp = c("treatment")	spats	trait = *treatment* + **treatment:genotype** + **row** + **col** + ϵ	PSANOVA
geno.decomp = c("treatment"), useCheck = TRUE, extraFixedFactors = "covar1"	spats	trait = *treatment* + *treatment:covar1* + *treatment:check* + **treatment:genoCheck** + **row** + **col** + ϵ	PSANOVA
spatial = TRUE, geno.decomp = c("treatment")	asreml	trait = *treatment* + **treatment:genotype** + **row** + **col** + ϵ	ar1(row):ar1(col)

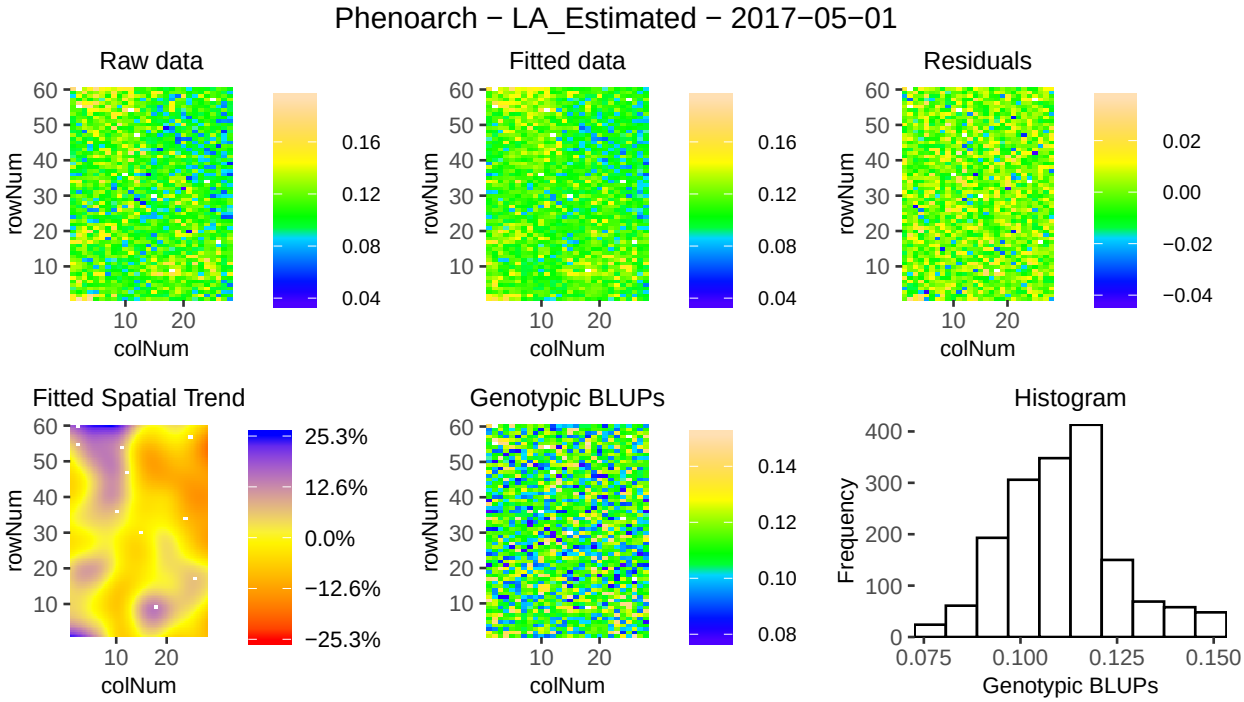
Note that for the moment, it is not possible to use the combination of geno.decomp and genotype as fixed.

Using the Phenoarch example, we will apply a variance decomposition using the variables "Scenario" and "population":

```
modPhenoSpGD <- fitModels(TP = phenoTParch,
                          trait = "LA_Estimated",
                          geno.decomp = c("Scenario", "population"),
                          timePoints = seq(from = 1, to = 35, by = 3))
```

4.4.3 Output

Extracting results and plotting from the models work exactly the same as previously. For example, the spatial plot on this dataset look like the following for one time point:



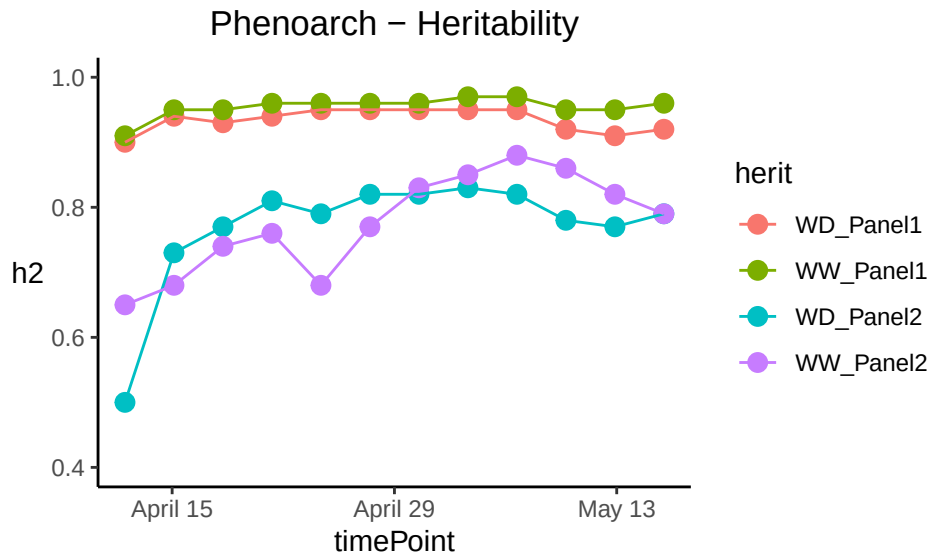
There are some significant differences in the display of some results and plots. They are highlighted below.

The predictions have two values per genotype, one per combined level of `geno.decomp`, here "Scenario_population", as illustrated in the table below for three genotypes predicted from the SpATS model `modPhenoSpGD`.

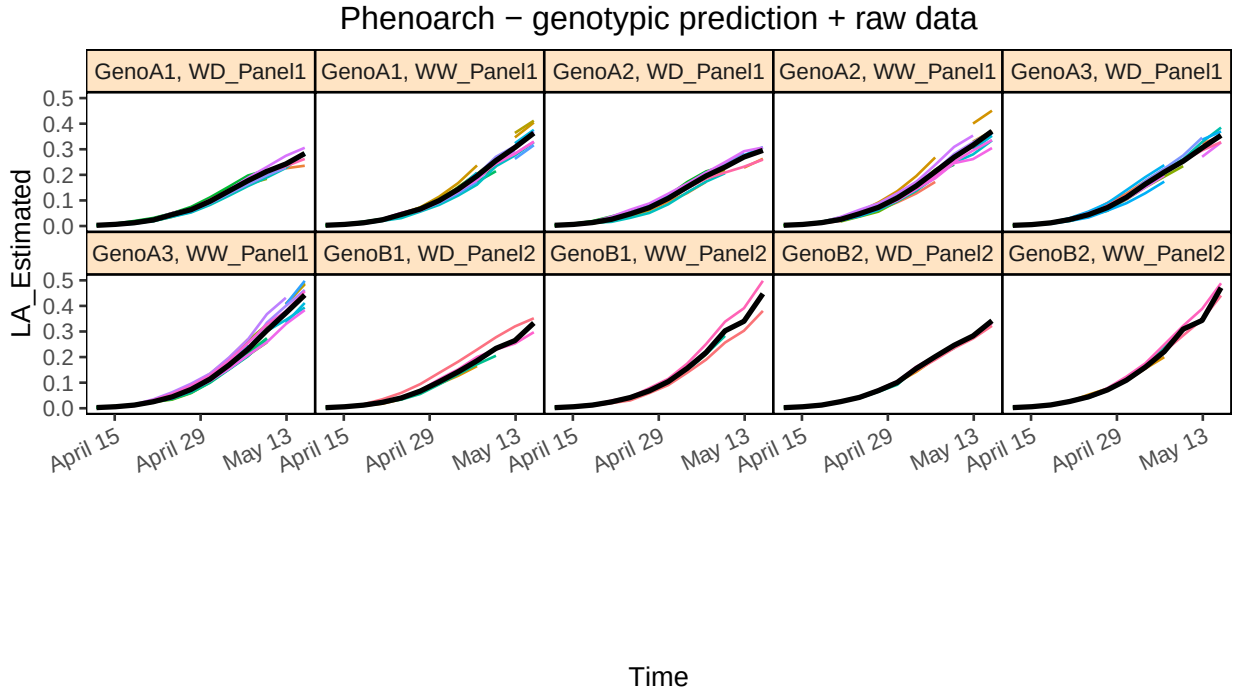
timeNumber	timePoint	geno.decomp	genotype	predicted.values	standard.errors
16	2017-04-28	WD_Panel1	GenoA1	0.0639173	0.0024161
16	2017-04-28	WD_Panel1	GenoA2	0.0719531	0.0024125
16	2017-04-28	WD_Panel2	GenoB1	0.0669098	0.0035835
16	2017-04-28	WD_Panel2	GenoB2	0.0689069	0.0035658
16	2017-04-28	WW_Panel1	GenoA1	0.0663221	0.0020659
16	2017-04-28	WW_Panel1	GenoA2	0.0728305	0.0020720
16	2017-04-28	WW_Panel2	GenoB1	0.0681970	0.0034844
16	2017-04-28	WW_Panel2	GenoB2	0.0716663	0.0035123

The heritabilities are now split into `geno.decomp` levels and their plot now displays one line per level.

timeNumber	timePoint	WD_Panel1	WW_Panel1	WD_Panel2	WW_Panel2
1	2017-04-13	0.90	0.91	0.50	0.65
4	2017-04-16	0.94	0.95	0.73	0.68
7	2017-04-19	0.93	0.95	0.77	0.74
10	2017-04-22	0.94	0.96	0.81	0.76
13	2017-04-25	0.95	0.96	0.79	0.68
16	2017-04-28	0.95	0.96	0.82	0.77



The prediction and corrected data plots display one plot per combination genotype × geno.decomp.



References

Butler, D. G., B.R. Cullis, A. R. Gilmour, B.G. Gogel, and R. Thompson. 2017. "ASReml-R Reference Manual Version 4." VSN International Ltd, Hemel Hempstead, HP1 1ES, UK. https://www.vsni.co.uk/.

Cabrera-Bosquet, Llorenç, Christian Fournier, Nicolas Brichet, Claude Welcker, Benoît Suard, and François Tardieu. 2016. "High-Throughput Estimation of Incident Light, Light Interception and Radiation-Use

Efficiency of Thousands of Plants in a Phenotyping Platform." *New Phytologist* 212 (1): 269–81. doi:10.1111/nph.14027.

Cullis, B. R., A. B. Smith, and N. E. Coombes. 2006. "On the Design of Early Generation Variety Trials with Correlated Data." *Journal of Agricultural, Biological, and Environmental Statistics* 11 (4): 381–93. doi:10.1198/108571106X154443.

Eeuwijk, Fred A. van, Daniela Bustos-Korts, Emilie J. Millet, Martin P. Boer, Willem Kruijer, Addie Thompson, Marcos Malosetti, et al. 2019. "Modelling Strategies for Assessing and Increasing the Effectiveness of New Phenotyping Techniques in Plant Breeding." *Plant Science* 282 (May): 23–39. doi:10.1016/j.plantsci.2018.06.018.

Flood, Pádraic J., Willem Kruijer, Sabine K. Schnabel, Rob van der Schoor, Henk Jalink, Jan F. H. Snel, Jeremy Harbinson, and Mark G. M. Aarts. 2016. "Phenomics for Photosynthesis, Growth and Reflectance in Arabidopsis Thaliana Reveals Circadian and Long-Term Fluctuations in Heritability." *Plant Methods* 12 (1): 14. doi:10.1186/s13007-016-0113-y.

Lee, Dae-Jin, María Durbán, and Paul Eilers. 2013. "Efficient Two-Dimensional Smoothing with P-Spline Anova Mixed Models and Nested Bases." *Computational Statistics & Data Analysis* 61 (May): 22–37. doi:10.1016/j.csda.2012.11.013.

Piepho, Hans-Peter, and Emlyn R. Williams. 2016. "Augmented Row–Column Designs for a Small Number of Checks." *Agronomy Journal* 108 (6): 2256. doi:10.2134/agronj2016.06.0325.

Rodríguez-Álvarez, María, Martin P. Boer, Fred van Eeuwijk, and Paul H.C. Eilers. 2017. "Correcting for Spatial Heterogeneity in Plant Breeding Experiments with P-Splines." *Spatial Statistics* 23 (October): 52–71. doi:10.1016/j.spasta.2017.10.003.

Rooijen, Roxanne van, Willem Kruijer, René Boesten, Fred A. van Eeuwijk, Jeremy Harbinson, and Mark G. M. Aarts. 2017. "Natural Variation of YELLOW SEEDLING1 Affects Photosynthetic Acclimation of Arabidopsis Thaliana." *Nature Communications* 8 (1). doi:10.1038/s41467-017-01576-3.

Tardieu, François, Llorenç Cabrera-Bosquet, Tony Pridmore, and Malcolm Bennett. 2017. "Plant Phenomics, from Sensors to Knowledge." *Current Biology* 27 (15): R770–R783. doi:10.1016/j.cub.2017.05.055.

Velazco, Julio G., María Xosé Rodríguez-Álvarez, Martin P. Boer, David R. Jordan, Paul H. C. Eilers, Marcos Malosetti, and Fred A. van Eeuwijk. 2017. "Modelling Spatial Trends in Sorghum Breeding Field Trials Using a Two-Dimensional P-Spline Mixed Model." *Theoretical and Applied Genetics* 130 (7): 1375–92. doi:10.1007/s00122-017-2894-4.