Package: statgenHTP (via r-universe)

September 9, 2024

Title High Throughput Phenotyping (HTP) Data Analysis

Version 1.0.6.1

Date 2023-04-13

Description Phenotypic analysis of data coming from high throughput phenotyping (HTP) platforms, including different types of outlier detection, spatial analysis, and parameter estimation. The package is being developed within the EPPN2020 project (<https://eppn2020.plant-phenotyping.eu/>). Some functions have been created to be used in conjunction with the R package 'asreml' for the 'ASReml' software, which can be obtained upon purchase from 'VSN' international (<https://vsni.co.uk/software/asreml>).

License GPL

Encoding UTF-8

LazyData true

Depends R (>= 3.6)

Imports animation, factoextra, ggforce, ggplot2 (>= 3.3.0), ggnewscale, grid, gridExtra, Matrix, methods, LMMsolver, locfit, lubridate, rlang, reshape2, scales, spam, SpATS (>= 1.0-13)

Suggests asreml (>= 4.0), knitr, rmarkdown, tinytest (>= 1.2.4)

RoxygenNote 7.2.3

Roxygen list(markdown = TRUE)

VignetteBuilder knitr

URL https://biometris.github.io/statgenHTP/index.html, https://github.com/Biometris/statgenHTP/

BugReports https://github.com/Biometris/statgenHTP/issues

Repository https://biometris.r-universe.dev

RemoteUrl https://github.com/biometris/statgenhtp

RemoteRef HEAD

RemoteSha baf0d5e28f4e3be21479f8963169935ed4aa22c1

Contents

`as.data.frame.TP` *Coerce TP object to data.frame*

Description

Function for converting an object of class TP to a data.frame.

Usage

```
## S3 method for class 'TP'
as.data.frame(x, ...)
```

Arguments

`x`	An object of class TP.
`...`	Ignored.

Value

A data.frame containing the data.frames for all time points in the TP object bound together.

See Also

Other functions for data preparation: `createTimePoints()`, `getTimePoints()`, `plot.TP()`, `removeTimePoints()`, `summary.TP()`

Examples

```
## Create a TP object containing the data from the Phenovator.
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"))
## Convert phenoTP to data.frame.
phenoDat <- as.data.frame(phenoTP)
```

countValid *Count valid observations per time point for a given trait*

Description

Count valid observations per time point for a given trait.

Usage

```
countValid(TP, trait)
```

Arguments

TP	An object of class TP.
trait	A character string indicating the trait for which valid observations should be counted.

Value

A named numerical vector with he number of valid observations per time point .

Examples

```
## Create a TP object containing the data from the Phenovator.
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"))
## Count valid observations for EffpsII per time point.
validPheno <- countValid(phenoTP, trait = "EffpsII")
head(validPheno)
```

countValidPlot *Count valid observations per plotId for a given trait*

Description

Count valid observations per plotId for a given trait.

Usage

```
countValidPlot(TP, trait, plotIds = NULL)
```

Arguments

TP	An object of class TP.
trait	A character string indicating the trait for which valid observations should be counted.
plotIds	A character vector indicating the plotIds for which valid observations should be checked. If NULL valid observations are counted for all plotIds in TP.

Value

A named numerical vector with he number of valid observations per plotId.

Examples

```
## Create a TP object containing the data from the Phenovator.
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"))
## Count valid observations for EffpsII for a subset of plots.
countValidPlot(phenoTP,
               trait = "EffpsII",
               plotIds = c("c12r22", "c24r41", "c14r32"))
```

---

createTimePoints *Create an object of class TP*

Description

Convert a data.frame to an object of class TP (Time Points). The function converts a data.frame to an object of class TP in the following steps:

- Quality control on the input data. For example, warnings will be given when more than 50% of observations are missing for a plant.
- 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. When the input time is a numeric value, the function will convert it to time from 01-01-1970.
- If `addCheck = TRUE`, the genotypes listed in `checkGenotypes` are reference genotypes (or check). It will add a column check with a value "noCheck" for the genotypes that are not in `checkGenotypes` and the name of the genotypes for the `checkGenotypes`. A column genoCheck is also added with the names of the genotypes that are not in `checkGenotypes` and `NA` for the `checkGenotypes`. These columns are necessary for fitting models on data that includes check genotypes, e.g. reference genotypes that are highly replicated or in case of augmented design.
- Split the data into separate data.frames by time point. 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 point 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 time points by a unique number.

Note that `plotId` needs to be a unique identifier for a plot or a plant. It cannot occur more than once per time point.

Usage

```
createTimePoints(
  dat,
  experimentName,
  genotype,
  timePoint,
  timeFormat = NULL,
  plotId,
  repId = NULL,
  rowNum = NULL,
  colNum = NULL,
  addCheck = FALSE,
  checkGenotypes = NULL
)
```

Arguments

`dat`	A data.frame.
`experimentName`	A character string, the name of the experiment. Stored with the data and used in default plot titles.
`genotype`	A character string indicating the column in dat containing the genotypes.
`timePoint`	A character string indicating the column in dat containing the time points.

timeFormat	A character string indicating the input format of the time points. E.g. for 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 strptime. If NULL, a best guess is done based on the input.
plotId	A character string indicating the column in dat containing the plotId. This has to be a unique identifier per plot/plant per time point.
repId	A character string indicating the column in dat containing the replicates.
rowNum	A character string indicating the column in dat containing the row number of the plot.
colNum	A character string indicating the column in dat containing the column number of the plot.
addCheck	Should a column check be added to the output? If TRUE, checkGenotypes cannot be NULL.
checkGenotypes	A character vector containing the genotypes used as checks in the experiment.

Value

An object of class TP. A list with, per time point in the input, a data.frame containing the data for that time point. A data.frame with columns timeNumber and timePoint is added as attribute timePoints to the data. This data.frame can be used for referencing timePoints by their number.

See Also

Other functions for data preparation: as.data.frame.TP(), getTimePoints(), plot.TP(), removeTimePoints(), summary.TP()

Examples

```
## Create a TP object containing the data from the Phenovator.
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)
```

`detectSerieOut` *Detect outliers for series of observations*

Description

Function for detecting strange time courses. The function uses the estimates for the spline coefficients per time course (typically per plant). Correlations between those coefficient vectors are calculated to identify outlying time courses, i.e., plants. An outlying time course will have low correlation to the majority of time courses. To support the analysis by correlations, a principal component analysis is done on the plant (time course) by spline coefficient matrix. A PCA plot of the plant scores will show the outlying plants. Finally the pairwise-ratios of the slopes of a linear model fitted through the spline coefficients are computed. Plants are tagged when the average pairwise-ratio is lower the a given threshold (`thrSlope`).

Usage

```
detectSerieOut(
  corrDat,
  predDat,
  coefDat,
  trait,
  genotypes = NULL,
  geno.decomp = NULL,
  thrCor = 0.9,
  thrPca = 30,
  thrSlope = 0.7
)
```

Arguments

`corrDat`	A data.frame with corrected spatial data.
`predDat`	A data.frame with predicted data from a fitted spline.
`coefDat`	A data.frame with coefficients from a fitted spline.
`trait`	A character string indicating the trait for which to detect outliers.
`genotypes`	A character vector indicating the genotypes for which to detect outliers. If `NULL`, outlier detection will be done for all genotypes.
`geno.decomp`	A character vector indicating the variables to use to group the genotypic variance in the model.
`thrCor`	A numerical value used as threshold for determining outliers based on correlation between plots.
`thrPca`	A numerical value used as threshold for determining outliers based on angles (in degrees) between PCA scores.
`thrSlope`	A numerical value used as threshold for determining outliers based on slopes.

Value

An object of class `serieOut`, a `data.frame` with outlying series of observations.

See Also

Other functions for detecting outliers for series of observations: `plot.serieOut()`, `removeSerieOut()`

Examples

```
## The data from the Phenovator platform have been corrected for spatial
## trends and outliers for single observations have been removed.

## Fit P-splines on a subset of genotypes
subGenoVator <- c("G160", "G151")
fit.spline <- fitSpline(inDat = spatCorrectedVator,
                        trait = "EffpsII_corr",
                        genotypes = subGenoVator,
                        knots = 50)

## Extract the data.frames with predicted values and P-Spline coefficients.
predDat <- fit.spline$predDat
coefDat <- fit.spline$coefDat

## The coefficients are then used to tag suspect time courses.
outVator <- detectSerieOut(corrDat = spatCorrectedVator,
                           predDat = predDat,
                           coefDat = coefDat,
                           trait = "EffpsII_corr",
                           genotypes = subGenoVator,
                           thrCor = 0.9,
                           thrPca = 30,
                           thrSlope = 0.7)

## The `outVator` can be visualized for selected genotypes.
plot(outVator, genotypes = "G151")
```

`detectSingleOut` *Detect outliers for single observations*

Description

Detect outlying observations in a time series by modeling each plotId using a local regression.

Usage

```
detectSingleOut(
  TP,
  trait,
  plotIds = NULL,
  checkEdges = TRUE,
  confIntSize = 5,
  nnLocfit = 0.5
)
```

Arguments

TP	An object of class TP.
trait	A character vector indicating the trait to model in TP.
plotIds	A character vector of plotIds for which the outliers should be detected. If NULL, all plotIds in TP are used.
checkEdges	Before fitting the local regression should a check be done if the first and last time point for a plot are outlying observations?
confIntSize	A numeric value defining the confidence interval (see Details).
nnLocfit	A numeric value defining the constant component of the smoothing parameter nn (see Details).

Details

See locfit() help function from the locfit R library. The user can act on:

nnLocfit the constant of the smoothing parameter. Increase nnLocfit to have a very smooth curve

confIntSize the level to calculate the confidence interval. Increase confIntSize to exclude less outliers

Value

An object of class singleOut, a `data.frame` with the following columns.

plotId plotId

timePoint time point

trait modeled trait

yPred prediction from the local regression

sd_yPred standard deviation of the prediction

lwr lower bound of the confidence interval

upr upper bound of the confidence interval

outlier flag for detected outlier (a value of 1 indicates the observation is an outlier)

See Also

Other functions for detecting outliers for single observations: `detectSingleOutMaize()`, `plot.singleOut()`, `removeSingleOut()`

Examples

```
## Create a TP object containing the data from the Phenovator.
PhenovatorDat1 <- PhenovatorDat1[!PhenovatorDat1$pos %in%
                                   c("c24r41", "c7r18", "c7r49"), ]
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"))

## First select a subset of plants, for example here 9 plants
plantSel <- phenoTP[[1]]$plotId[1:9]
# Then run on the subset
resuVatorHTP <- detectSingleOut(TP = phenoTP,
                                trait = "EffpsII",
                                plotIds = plantSel,
                                confIntSize = 3,
                                nnLocfit = 0.1)
```

`detectSingleOutMaize` *detectSingleOutMaize*

Description

Function to detect plant outliers in a temporal lattice experiment on Maize which can be extended to other experiment types. The criteria needs three phenotypes (ex for maize: the estimated biomass, plant height and phyllocron)

plants are identified as "small outlier plant" if for biomass AND phyllocron $res_i < \mu_{res} - qnorm(threshold) * sd_{res}$

plants are identified as "big outlier plant" if for biomass AND plant height $res_i > \mu_{res} + qnorm(threshold) * sd_{res}$

Usage

```
detectSingleOutMaize(
  TP,
  timeBeforeTrt,
  trait1 = "Biomass",
  trait2 = "PlantHeight",
  trait3 = "phyllocron",
  thr = 0.95
)
```

Arguments

TP	An object of class TP.
timeBeforeTrt	A character or numeric value indicating the date just before treatment in the experiment. When using a character string to reference a time point, the value has to be an exact match to one of the existing timePoints. When using a number it will be matched by its number ("timeNumber") in the timePoints attribute of the TP object.
trait1	A character vector indicating the first trait to model in TP.
trait2	A character vector indicating the second trait to model in TP.
trait3	A character vector indicating the third trait to model in TP.
thr	A numeric value indicating the threshold.

Value

A list with three data.frames, modDat containing the data used for fitting the models, smallPlants containing the plants identified as small plants and bigPlants containing the plants identified as big plants.

See Also

Other functions for detecting outliers for single observations: detectSingleOut(), plot.singleOut(), removeSingleOut()

Examples

```
## Create a TP object containing the data from the PhenoArch.
phenoTParch <- createTimePoints(dat = PhenoarchDat1,
                                experimentName = "Phenoarch",
                                genotype = "Genotype",
                                timePoint = "Date",
                                plotId = "pos",
                                rowNum = "Row",
                                colNum = "Col")
singleOutMaize <- detectSingleOutMaize(phenoTParch,
                                       timeBeforeTrt = "2017-04-27",
                                       trait1 = "Biomass",
                                       trait2 = "PlantHeight",
                                       trait3 = "phyllocron",
                                       thr = 0.95)
```

`estimateSplineParameters` *Extract estimates from fitted splines.*

Description

Function for extracting parameter estimates from fitted splines on a specified interval.

Usage

```
estimateSplineParameters(
  x,
  estimate = c("predictions", "derivatives", "derivatives2"),
  what = c("min", "max", "mean", "AUC", "p"),
  AUCScale = c("min", "hour", "day"),
  timeMin = NULL,
  timeMax = NULL,
  genotypes = NULL,
  plotIds = NULL,
  fitLevel = c("geno", "plot", "genoDev", "plotDev")
)
```

Arguments

`x`	An object of class HTPSpline, the output of the `fitSpline` function, or class splineHDm, the output of the `fitSplineHDM` function
`estimate`	The P-Spline component for which the estimate should be extracted, the predictions, the first derivatives or the second derivatives ("derivatives2")
`what`	The types of estimate that should be extracted. Either minimum ("min"), maximum ("max"), mean, area under the curve ("AUC") or a percentile. Percentiles should be given as p + percentile. E.g. for the 10th percentile specify what = "p10". Multiple types of estimate can be extracted at once.
`AUCScale`	The area under the curve is dependent on the scale used on the x-axis. By default the area is computed assuming a scale in minutes. This can be changed to either hours or days.
`timeMin`	The lower bound of the time interval from which the estimates should be extracted. If `NULL` the smallest time value for which the splines were fitted is used.
`timeMax`	The upper bound of the time interval from which the estimates should be extracted. If `NULL` the largest time value for which the splines were fitted is used.
`genotypes`	A character vector indicating the genotypes for which estimates should be extracted. If `NULL`, estimates will be extracted for all genotypes for which splines where fitted.
`plotIds`	A character vector indicating the plotIds for which estimates should be extracted. If `NULL`, estimates will be extracted for all plotIds for which splines where fitted.

`fitLevel` A character string indicating at which level of the data the parameter estimates should be made. Only used for splines fitted using `fitSplineHDM`.

Value

An object of class splineEst, a data.frame containing the estimated parameters.

See Also

Other functions for spline parameter estimation: `plot.splineEst()`

Examples

```
### Estimate parameters for fitted P-splines.

## Run the function to fit P-splines on a subset of genotypes.
subGeno <- c("G160", "G151")
fit.spline <- fitSpline(inDat = spatCorrectedVator,
                        trait = "EffpsII_corr",
                        genotypes = subGeno,
                        knots = 50)

## Estimate the maximum value of the predictions at the beginning of the time course.
paramVator <- estimateSplineParameters(x = fit.spline,
                                       estimate = "predictions",
                                       what = "max",
                                       timeMin = 1527784620,
                                       timeMax = 1528500000,
                                       genotypes = subGeno)
head(paramVator)

## Create a boxplot of the estimates.
plot(paramVator, plotType = "box")

## Estimate the minimum and maximum value of the predictions.
paramVator2 <- estimateSplineParameters(x = fit.spline,
                                        estimate = "predictions",
                                        what = c("min", "max"),
                                        genotypes = subGeno)
head(paramVator2)


### Estimate parameters for fitted HDM-splines.

## The data from the Phenovator platform have been corrected for spatial
## trends and outliers for single observations have been removed.

## We need to specify the genotype-by-treatment interaction.
## Treatment: water regime (WW, WD).
spatCorrectedArch[["treat"]] <- substr(spatCorrectedArch[["geno.decomp"]],
                                       start = 1, stop = 2)
spatCorrectedArch[["genoTreat"]] <-
```

```
  interaction(spatCorrectedArch[["genotype"]],
              spatCorrectedArch[["treat"]], sep = "_")

## Fit P-Splines Hierarchical Curve Data Model for selection of genotypes.
fit.psHDM  <- fitSplineHDM(inDat = spatCorrectedArch,
                           trait = "LeafArea_corr",
                           genotypes = c("GenoA14_WD", "GenoA51_WD",
                                         "GenoB11_WW", "GenoB02_WD",
                                         "GenoB02_WW"),
                           time = "timeNumber",
                           pop = "geno.decomp",
                           genotype = "genoTreat",
                           plotId = "plotId",
                           difVar = list(geno = FALSE, plot = FALSE),
                           smoothPop = list(nseg = 4, bdeg = 3, pord = 2),
                           smoothGeno = list(nseg = 4, bdeg = 3, pord = 2),
                           smoothPlot = list(nseg = 4, bdeg = 3, pord = 2),
                           weights = "wt",
                           trace = FALSE)

## Estimate minimum, maximum, and mean for predictions at the genotype level.
paramArch <- estimateSplineParameters(x = fit.psHDM,
                                      what = c("min", "max", "mean"),
                                      fitLevel = "geno",
                                      estimate = "predictions",
                                      timeMax = 28)
head(paramArch)

## Create a boxplot of the estimates.
plot(paramArch, plotType = "box")

## Estimate area under the curve for predictions at the plot level.
paramArch2 <- estimateSplineParameters(x = fit.psHDM,
                                       what = "AUC",
                                       fitLevel = "plot",
                                       estimate = "predictions")
head(paramArch2)
```

`fitModels` *Fit spatial models per time point*

Description

Perform REML analysis at each time point using either SpATS or asreml. The idea is to is to accurately separate the genetic effects from the spatial effects at each time point. SpATS is used as a default method. See details for the exact models fitted.

Usage

```
fitModels(
  TP,
  trait,
  timePoints = names(TP),
  extraFixedFactors = NULL,
  geno.decomp = NULL,
  what = c("random", "fixed"),
  useCheck = FALSE,
  useRepId = FALSE,
  engine = c("SpATS", "asreml"),
  spatial = FALSE,
  quiet = FALSE
)
```

Arguments

TP	An object of class TP.
trait	A character string indicating the trait used as response variable in the model.
timePoints	A character or numeric vector indicating the time points to be modeled. When using a character string to reference a time point, the value has to be an exact match to one of the existing time points. When using a number it will be matched by its number ("timeNumber") in the timePoints attribute of the TP object.
extraFixedFactors	A character vector indicating the variables to use as extra fixed effects in the model.
geno.decomp	A character vector indicating the variables to use to group the genotypic variance in the model.
what	A character vector specifying whether "genotype" should be fitted as "random" or "fixed" effect. Note that when using geno.decomp, fitting a model with genotype as "fixed" effect is not possible.
useCheck	Should check genotypes be used as an extra factor in the model?
useRepId	Should repId be used as a fixed effect in the model? When fitting a spatial model rowId and colId are also nested within repId in the random part of the model.
engine	A character string indicating the engine used to fit the models.
spatial	Should a spatial model be fitted for asreml?
quiet	Should printed progress messages be suppressed?

Details

The actual model fitted depends on the function parameters specified. The basic model is the following:
trait = **genotype** + e
In case useCheck = TRUE, instead of genotype, genoCheck is used as genotype and check is used as an extra fixed effect. So then the model becomes:

trait = *check* + **genoCheck** + e
Variables in `extraFixedFactors` are fitted as extra fixed effects.

When `SpATS` is used for modeling, an extra spatial term is always included in the model. This term is constructed using the function `PSANOVA` from the SpATS package as
`PSANOVA(colNum, rowNum, nseg = nSeg, nest.div = 2)` where
`nSeg = c(number of columns, number of rows)`.

When `asreml` is used for modeling and `spatial = TRUE`, four models are fitted with different covariance structures. The best model is determined based on a goodness-of-fit criterion, AIC, on 20% of the time points or at least 10 time points. The best model is then run on all time points. The following combinations of random and spatial terms are fitted

- random = repId:rowId + repId:colId, spatial = NULL
- random = repId:rowId + repId:colId, spatial = ar1(rowId):colId
- random = repId:colId + repId:colId, spatial = rowId:ar1(colId)
- random = repId:rowId + repId:colId, spatial = ar1(rowId):ar1(colId)

If there are no replicates in the model, repId is left out from the random parts above.

When `geno.decomp` is specified, the genotypic variance is decomposed following the variable(s) chosen. For example, when a treatment is used in `geno.decomp`, the initial model becomes:
trait = *treatment* + **treatment:genotype** + e

Value

An object of class `fitMod`, a list of fitted models.

References

Maria Xose Rodriguez-Alvarez, Martin P. Boer, Fred A. van Eeuwijk, Paul H.C. Eilers (2017). Correcting for spatial heterogeneity in plant breeding experiments with P-splines. Spatial Statistics doi:10.1016/j.spasta.2017.10.003 Butler, D. G., et al. (2018). ASReml-R Reference Manual Version 4. VSN International Ltd, http://asreml.org

See Also

Other functions for spatial modeling: `getCorrected()`, `getEffDims()`, `getGenoPred()`, `getHerit()`, `getVar()`, `plot.fitMod()`, `summary.fitMod()`

Examples

```
## Using the first example dataset (PhenovatorDat1):
## Fit a model using SpATS on few time points:

## Create an object of class TP.
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"))
## Fit a model with SpATS for three time points.
modPhenoSp <- fitModels(TP = phenoTP,
                        trait = "EffpsII",
                        timePoints = c(3, 6, 20))
summary(modPhenoSp)

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


## Fit a model with asreml on few time points with a spatial function:
if (requireNamespace("asreml", quietly = TRUE)) {
  modPhenoSpAs <- fitModels(TP = phenoTP,
                            trait = "EffpsII",
                            timePoints = c(1, 6, 20),
                            engine = "asreml",
                            spatial = TRUE)
}

## Using the second example dataset (PhenoarchDat1):
## Fit a model with SpATS on one time points with two variables for
## geno.decomp:
data("PhenoarchDat1")
phenoTParch <- createTimePoints(dat = PhenoarchDat1,
                                experimentName = "Phenoarch",
                                genotype = "Genotype",
                                timePoint = "Date",
                                plotId = "pos",
                                rowNum = "Row",
                                colNum = "Col")

modPhenoSpGD <- fitModels(TP = phenoTParch,
                          trait = "LeafArea",
                          geno.decomp = c("Scenario", "population"),
                          timePoints = 16)
```

`fitSpline` *Fit Splines*

Description

Fit P-Splines on corrected or raw data. The number of knots is chosen by the user. The function outputs are predicted P-Spline values and their first and second derivatives on a dense grid. The outputs can then be used for outlier detection for time series (see `detectSerieOut`) and to estimate relevant parameters from the curve for further analysis (see `estimateSplineParameters`).

Usage

```
fitSpline(
  inDat,
  trait,
  genotypes = NULL,
  plotIds = NULL,
  knots = 50,
  useTimeNumber = FALSE,
  timeNumber = NULL,
  minNoTP = NULL
)
```

Arguments

`inDat`	A data.frame with corrected spatial data.
`trait`	A character string indicating the trait for which the spline should be fitted.
`genotypes`	A character vector indicating the genotypes for which splines should be fitted. If `NULL`, splines will be fitted for all genotypes.
`plotIds`	A character vector indicating the plotIds for which splines should be fitted. If `NULL`, splines will be fitted for all plotIds.
`knots`	The number of knots to use when fitting the spline.
`useTimeNumber`	Should the timeNumber be used instead of the timePoint?
`timeNumber`	If `useTimeNumber = TRUE`, a character vector indicating the column containing the numerical time to use.
`minNoTP`	The minimum number of time points for which data should be available for a plant. Defaults to 80% of all time points present in the TP object. No splines are fitted for plants with less than the minimum number of timepoints.

Value

An object of class `HTPSpline`, a list with two `data.frames`, `predDat` with predicted values and `coefDat` with P-Spline coefficients on a dense grid.

See Also

Other functions for fitting splines: plot.HTPSpline()

Examples

```
## The data from the Phenovator platform have been corrected for spatial
## trends and outliers for single observations have been removed.

## Fit P-Splines on a subset of genotypes
subGeno <- c("G070", "G160")
fit.spline <- fitSpline(inDat = spatCorrectedVator,
                        trait = "EffpsII_corr",
                        genotypes = subGeno,
                        knots = 50)

## Extract the data.frames with predicted values and P-Spline coefficients.
predDat <- fit.spline$predDat
head(predDat)

coefDat <- fit.spline$coefDat
head(coefDat)

## Visualize the P-Spline predictions for one genotype.
plot(fit.spline, genotypes = "G160")

## Visualize the P-Spline predictions and first derivatives for one plant.
plot(fit.spline, plotIds = "c10r29", plotType = "predictions")
plot(fit.spline, plotIds = "c10r29", plotType = "derivatives")
```

---

fitSplineHDM *Fit P-Spline Hierarchical Curve Data Models*

Description

Fit the P-spline Hierarchical Curve Data Model used in the second stage of the two-stage approach proposed by Pérez-Valencia et al. (2022). This model assumes a three-level hierarchical structure in the data, with plants nested in genotypes, genotypes nested in populations. The input for this function is the spatially corrected data, as obtained from the first stage of the approach (see fitModels and getCorrected). The number of segments is chosen by the user, as well as the B-spline degree, and the penalty order for the three-levels of the hierarchy. The user can also decide if different variances for random effects at genotype (separately for each population) and plant (separately for each genotype) levels are desired. The function outputs are estimated curves (time series of trajectories and deviations) and their first and second derivatives for the three-levels of the hierarchy. The outputs can then be used to estimate relevant parameters from the curves for further analysis (see estimateSplineParameters).

Usage

```
fitSplineHDM(
  inDat,
  genotypes = NULL,
  plotIds = NULL,
  trait,
  useTimeNumber = FALSE,
  timeNumber = NULL,
  pop = "pop",
  genotype = "genotype",
  plotId = "plotId",
  weights = NULL,
  difVar = list(geno = FALSE, plot = FALSE),
  smoothPop = list(nseg = 10, bdeg = 3, pord = 2),
  smoothGeno = list(nseg = 10, bdeg = 3, pord = 2),
  smoothPlot = list(nseg = 10, bdeg = 3, pord = 2),
  offset = NULL,
  family = gaussian(),
  maxit = 200,
  trace = TRUE,
  thr = 0.001,
  minNoTP = NULL
)
```

Arguments

inDat	A data.frame with corrected spatial data.
genotypes	A character vector indicating the genotypes for which hierarchical models should be fitted. If NULL, splines will be fitted for all genotypes.
plotIds	A character vector indicating the plotIds for which hierarchical models should be fitted. If NULL, splines will be fitted for all plotIds.
trait	A character string indicating the trait for which the spline should be fitted.
useTimeNumber	Should the timeNumber be used instead of the timePoint?. If useTimeNumber = FALSE, inDat should contain a column called timePoint of class POSIXct.
timeNumber	If useTimeNumber = TRUE, a character vector indicating the column containing the numerical time to use.
pop	A character string indicating the the populations to which each genotype/variety belongs. This variable must be a factor in the data frame.
genotype	A character string indicating the populations to which each genotype/variety belongs. This variable must be a factor in the data frame.
plotId	A character string indicating the genotypes/varieties to which each plant/plot/individual belongs. This variable must be a factor in the data frame.
weights	A character string indicating the column in the data containing the weights to be used in the fitting process (for error propagation from first stage to second stage). By default, when weights = NULL, the weights are considered to be one.

difVar	Should different variances for random effects at genotype (separately for each population) and plant level (separately for each genotype) be considered?.
smoothPop	A list specifying the P-Spline model at the population level (nseg: number of segments; bdeg: degree of the B-spline basis; pord: penalty order).
smoothGeno	A list specifying the P-Spline model at the genotype level.
smoothPlot	A list specifying the P-Spline model at the plant level.
offset	A character string indicating the column in the data with an a priori known component to be included in the linear predictor during fitting. By default, when offset = NULL, the offset is considered to be zero.
family	An object of class family specifying the distribution and link function. The default is gaussian().
maxit	An optional value that controls the maximum number of iterations of the algorithm. The default is 200.
trace	An optional value that controls the function trace. The default is TRUE.
thr	An optional value that controls the convergence threshold of the algorithm. The default is 1.e-03.
minNoTP	The minimum number of time points for which data should be available for a plant. Defaults to 60% of all time points present in the TP object. No splines are fitted for plants with less than the minimum number of timepoints.

Value

An object of class psHDM, a list with the following outputs: time, a numeric vector with the timepoints. popLevs, a data.frame with the names of the populations genoLevs, a factor with the names of the genotypes. plotLevs, a factor with the names of the plants nPlotPop, a numeric vector with the number of plants per population. nGenoPop, a numeric vector with the number of genotypes per population. nPlotGeno, a numeric vector with the number of plants per genotype. MM, a list with the design matrices at plant, genotype and population levels. ed, a numeric vector with the estimated effective dimension (or effective degrees of freedom) for each random component of the model (intercept, slope and non-linear trend) at each level of the hierarchy (population, genotype and plant) tot_ed, a numeric value with the sum of the effective dimensions for all components of the model. vc, a numeric vector with the (REML) variance component estimates for each random component of the model (intercept, slope and non-linear trend) at each level of the hierarchy (population, genotype and plant) phi, a numeric value with the error variance estimate. coeff, a numeric vector with the estimated fixed and random effect coefficients. popLevel, a data.frame with the estimated population trajectories and first and second order derivatives. genoLevel, a data.frame with the estimated genotype-specific deviations and trajectories, and their respective first and second order derivatives. plotLevel, a data.frame with the estimated plant-specific deviations and trajectories, and their respective first and second order derivatives. deviance, the (REML) deviance at convergence. convergence, a logical value indicating whether the algorithm managed to converge before the given number of iterations. dim, a numeric vector with the (model) dimension of each model component (fixed and/or random) at each level of the hierarchy (population, genotype, and plant). These values correspond to the number of parameters to be estimated. family, an object of class family specifying the distribution and link function. Vp, the variance-covariance matrix for the coefficients. smooth, a list with the information about number of segments (nseg), degree of the B-spline basis (bdeg) and penalty order (pord) used for the three levels of the hierarchy.

References

Pérez-Valencia, D.M., Rodríguez-Álvarez, M.X., Boer, M.P. et al. A two-stage approach for the spatio-temporal analysis of high-throughput phenotyping data. Sci Rep 12, 3177 (2022). doi:10.1038/s41598022069359

See Also

Other functions for fitting hierarchical curve data models: `plot.psHDM()`, `predict.psHDM()`

Examples

```
## The data from the Phenovator platform have been corrected for spatial
## trends and outliers for single observations have been removed.
head(spatCorrectedArch)
ggplot2::ggplot(data = spatCorrectedArch,
                ggplot2::aes(x= timeNumber, y = LeafArea_corr, group = plotId)) +
  ggplot2::geom_line(na.rm = TRUE) +
  ggplot2::facet_grid(~geno.decomp)

## We need to specify the genotype-by-treatment interaction.
## Treatment: water regime (WW, WD).
spatCorrectedArch[["treat"]] <- substr(spatCorrectedArch[["geno.decomp"]],
                                       start = 1, stop = 2)
spatCorrectedArch[["genoTreat"]] <-
  interaction(spatCorrectedArch[["genotype"]],
              spatCorrectedArch[["treat"]], sep = "_")

## Fit P-Splines Hierarchical Curve Data Model for selection of genotypes.
fit.psHDM  <- fitSplineHDM(inDat = spatCorrectedArch,
                           trait = "LeafArea_corr",
                           useTimeNumber = TRUE,
                           timeNumber = "timeNumber",
                           genotypes = c("GenoA14_WD", "GenoA51_WD",
                                         "GenoB11_WW", "GenoB02_WD",
                                         "GenoB02_WW"),
                           pop = "geno.decomp",
                           genotype = "genoTreat",
                           plotId = "plotId",
                           weights = "wt",
                           difVar = list(geno = FALSE, plot = FALSE),
                           smoothPop = list(nseg = 4, bdeg = 3, pord = 2),
                           smoothGeno = list(nseg = 4, bdeg = 3, pord = 2),
                           smoothPlot = list(nseg = 4, bdeg = 3, pord = 2),
                           trace = FALSE)

## Visualize the data.frames with predicted values at the three levels of
## the hierarchy.

# Population level
head(fit.psHDM$popLevel)

# Genotype level
```

```
head(fit.psHDM$genoLevel)

# Plot level
head(fit.psHDM$plotLevel)
```

`getCorrected` *Extract corrected phenotypic values*

Description

Extract corrected phenotypic values from an object of class fitMod. After fitting a spatial model at each time point, the raw phenotypic data is corrected by subtracting the (estimated) sources of variation (environmental, design effect) that are of no interest (nuisances). This allows keeping the data resolution at the plot/plant level.

Usage

```
getCorrected(fitMod, timePoints = names(fitMod), outFile = NULL)
```

Arguments

`fitMod`	An object of class `fitMod`.
`timePoints`	A character or numeric vector indicating the time point(s) for which the corrected values should be extracted. When using a character string to reference a time point, the value has to be an exact match to one of the existing time points. When using a number it will be matched by its number ("timeNumber") in the timePoints attribute of the TP object.
`outFile`	A character string indicating the .csv file to which the results should be written. If `NULL` no file is written.

Value

A data.frame with spatially corrected values per time point.

See Also

Other functions for spatial modeling: `fitModels()`, `getEffDims()`, `getGenoPred()`, `getHerit()`, `getVar()`, `plot.fitMod()`, `summary.fitMod()`

Examples

```
## Using the first example dataset (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"))

## Fit a SpATS model on few time points.
modPhenoSp <- fitModels(TP = phenoTP,
                        trait = "EffpsII",
                        timePoints = c(1, 6, 20))

## Extract the corrected values for one time point:
spatCorrSp <- getCorrected(modPhenoSp,
                           timePoints = 6)
head(spatCorrSp)
```

`getEffDims` *Extract effective dimensions*

Description

Extract effective dimensions from an object of class fitMod. The table below gives an overview of the effective dimensions and an explanation of their meaning.

Effective Dimension	Explanation
colId	Linear trend along columns
rowId	Linear trend along rows
fCol	Smooth trend along columns
fRow	Smooth trend along rows
fColRow	Linear trend in rows changing smoothly along cols
colfRow	Linear trend in cols changing smoothly along rows
fColfRow	Smooth-by-smooth interaction trend over rows and cols
surface	Sum of smooth trends

Usage

```
getEffDims(
  fitMod,
  timePoints = names(fitMod),
  EDType = c("dimension", "ratio"),
  outFile = NULL
)
```

Arguments

fitMod	An object of class fitMod.
timePoints	A character or numeric vector indicating the time point(s) for which the effective dimension should be extracted. When using a character string to reference a time point, the value has to be an exact match to one of the existing time points. When using a number it will be matched by its number ("timeNumber") in the timePoints attribute of the TP object.
EDType	A character string specifying if the effective dimension ("dimension") or the ratio of effective dimensions ("ratio") should be returned.
outFile	A character string indicating the .csv file to which the results should be written. If NULL no file is written.

Value

A data.frame with effective dimensions per time point.

See Also

Other functions for spatial modeling: fitModels(), getCorrected(), getGenoPred(), getHerit(), getVar(), plot.fitMod(), summary.fitMod()

Examples

```
## Using the first example dataset (PhenovatorDat1):
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"))

## Fit a SpATS model on few time points:
modPhenoSp <- fitModels(TP = phenoTP,
                        trait = "EffpsII",
                        timePoints = c(1, 6, 20))

## Extract the effective dimensions for all available time points in the
## model object:
effDimSp <- getEffDims(modPhenoSp)
```

getGenoPred *Extract predicted genotypic values*

Description

Extract predictions of the genotypic value from an object of class fitMod.

Usage

```
getGenoPred(
  fitMod,
  timePoints = names(fitMod),
  predictChecks = FALSE,
  outFile = NULL
)
```

Arguments

fitMod	An object of class fitMod.
timePoints	A character or numeric vector indicating the time point(s) for which the predictions should be extracted. When using a character string to reference a time point, the value has to be an exact match to one of the existing time points. When using a number it will be matched by its number ("timeNumber") in the timePoints attribute of the TP object.
predictChecks	Should predictions of the check genotypes be included in the ouptut. If TRUE a list of two data.frames is returned from the function, one with the predictions for the regular genotypes and one with the predictions for the checks.
outFile	A character string indicating the .csv file to which the results should be written. If NULL no file is written.

Value

A list of two data.frames with predicted genotypic values per time point. genoPred with the predicted values for the genotypes and checkPred with the predicted values for the checks. If predictChecks = FALSE the latter will be NULL.

See Also

Other functions for spatial modeling: fitModels(), getCorrected(), getEffDims(), getHerit(), getVar(), plot.fitMod(), summary.fitMod()

Examples

```
## Using the first example dataset (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"))

## Fit a SpATS model on few time points.
modPhenoSp <- fitModels(TP = phenoTP,
                        trait = "EffpsII",
                        timePoints = c(1, 6, 20))

## Extract the genotypic predictions for one time point:
genoPredSp <- getGenoPred(modPhenoSp,
                          timePoints = 6)
head(genoPredSp)
```

`getHerit` *Extract heritabilities*

Description

Extract heritabilities from an object of class fitMod. When `geno.decomp` is used, the heritabilities of each level of geno.decomp are stored in separate columns.

Usage

```
getHerit(fitMod, timePoints = names(fitMod), outFile = NULL)
```

Arguments

`fitMod`	An object of class `fitMod`.
`timePoints`	A character or numeric vector indicating the time point(s) for which the heritabilities should be extracted. When using a character string to reference a time point, the value has to be an exact match to one of the existing time points. When using a number it will be matched by its number ("timeNumber") in the timePoints attribute of the TP object.
`outFile`	A character string indicating the .csv file to which the results should be written. If `NULL` no file is written.

Value

A data.frame with heritabilities per time point.

See Also

Other functions for spatial modeling: fitModels(), getCorrected(), getEffDims(), getGenoPred(), getVar(), plot.fitMod(), summary.fitMod()

Examples

```
## Using the first example dataset (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"))

## Fit a SpATS model on few time points.
modPhenoSp <- fitModels(TP = phenoTP,
                        trait = "EffpsII",
                        timePoints = c(1, 6, 20))

## Extract the heritabilities for all available time points.#'
getHerit(modPhenoSp)
```

---

getTimePoints *Extract time points*

Description

Function for extracting a data.frame with timeNumbers and timePoints from an object of class TP or fitMod.

Usage

```
getTimePoints(x)
```

Arguments

x	An object of class TP or fitMod

Value

A data.frame with columns timeNumber and timePoint listing the time points in x

See Also

Other functions for data preparation: `as.data.frame.TP()`, `createTimePoints()`, `plot.TP()`, `removeTimePoints()`, `summary.TP()`

Examples

```
## Create an object of class TP.
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"))

## Extract the time points from the object.
head(getTimePoints(phenoTP))
```

`getVar` *Extract variances*

Description

Extract variances from an object of class fitMod.

Usage

```
getVar(fitMod, timePoints = names(fitMod), outFile = NULL)
```

Arguments

`fitMod`	An object of class `fitMod`.
`timePoints`	A character or numeric vector indicating the time point(s) for which the variances should be extracted. When using a character string to reference a time point, the value has to be an exact match to one of the existing time points. When using a number it will be matched by its number ("timeNumber") in the timePoints attribute of the TP object.
`outFile`	A character string indicating the .csv file to which the results should be written. If `NULL` no file is written.

Value

A data.frame with variances per time point.

See Also

Other functions for spatial modeling: fitModels(), getCorrected(), getEffDims(), getGenoPred(), getHerit(), plot.fitMod(), summary.fitMod()

Examples

```
## Using the first example dataset (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"))

## Fit a SpATS model on few time points.
modPhenoSp <- fitModels(TP = phenoTP,
                        trait = "EffpsII",
                        timePoints = c(1, 6, 20))

## Extract the variances for all available time points.
getVar(modPhenoSp)
```

---

noCorrectedRoot *Root data corrected for outliers for single observations.*

Description

This dataset contains the corrected data obtained by removing outliers for single observations on the RootDat1 dataset. See the vignettes for details.

Usage

```
noCorrectedRoot
```

Format

A data.frame with 15,934 rows and 8 columns:

timePoint Original time points, date and time

Date Date

thermalTime Thermal time cumulated

Exp Experiment number

genotype Genotypes

Tank Tank in the greenhouse

plotId Unique pot ID using rowcol coordinates

rowId Row coordinate

colId Column coordinate

tipPos_y Position of the root tip in y axis

`PhenoarchDat1` *Greenhouse data for a maize experiment in the PhenoArch platform.*

Description

A dataset containing data from a greenhouse experiment with maize in the Phenoarch platform (INRAE, France, Cabrera-Bosquet et al. 2016). It consists of one experiment with 1,671 plants grown in a greenhouse under two water scenarios, well-watered (WW) and water deficit (WD). There are two populations of genotypes, panel 1 and panel 2. Panel 1 contains 60 genotypes with 14 replicates: 7 in WW and 7 in WD. Panel 2 contains 30 genotypes with 8 replicates, 4 in WW and 4 in WD. The studied trait is the leaf area extracted from the pictures over time (LeafArea). Plants were pictured every day for 33 days. This dataset was kindly provided by Llorenç Cabrera-Bosquet and Claude Welcker.

Usage

```
PhenoarchDat1
```

Format

A data.frame with 42,536 rows and 14 columns:

Date Date of measurement

pos Unique pot using rowcol coordinate

Genotype Genotype

Scenario Water regime, WW or WD

population Panel 1 or 2

Row Pot position on the conveyor belt (i.e. row coordinate)

Col Line of conveyor belt (i.e. column coordinate)

Biomass Biomass from the picture

LeafArea Leaf area from the picture

PlantHeight Plant height from the picture

LeafCount Number of leaves manually scored

phyllocron Leaf emission rate

References

Cabrera-Bosquet, Llorenç, Fournier Christian, Brichet Nicolas, Welcker Claude, Suard Benoît, and Tardieu François. 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

`PhenovatorDat1` *Growth chamber data for an Arabidopsis experiment in the Phenovator platform.*

Description

A dataset containing data from a growth chamber experiment with Arabidopsis in the Phenovator platform (WUR, Netherlands, Flood et al. 2016). It consists of one experiment with 1,440 plants grown in a growth chamber. The number of tested genotypes is 192 with 6 to 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 (van Rooijen et al. 2017). This dataset was kindly provided by René Boesten and Mark Aarts.

Usage

```
PhenovatorDat1
```

Format

A data.frame with 103,839 rows and 10 columns:

Genotype Genotypes

Basin Table of experiment

Replicate Block define after sowing for post-blocking. They are not full-resolvable blocks.

Image_pos Position of the camera

x Row coordinate

y Column coordinate

Sowing_Position Unique pot ID

timepoints time of picture

EffpsII Efficiency of the photosystem II

pos Unique pot ID using rowcol coordinates

References

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/s41467017-015763

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/s130070160113y

---

`plot.fitMod` *Plot function for class fitMod*

Description

Plotting function for objects of class `fitMod`. Seven different types of plots can be made for an object of class `fitMod`. A detailed description and optional extra parameters for the different plots is given in the sections below.

Usage

```
## S3 method for class 'fitMod'
plot(
  x,
  ...,
  plotType = c("rawPred", "corrPred", "herit", "effDim", "variance", "timeLapse",
    "spatial"),
  timePoints = names(x),
  title = NULL,
  output = TRUE,
  outFile = NULL,
  outFileOpts = NULL
)
```

Arguments

`x`	An object of class fitMod.
`...`	Extra plot options. Described per plotType in their respective section.
`plotType`	A single character string indicating which plot should be made. See the sections below for a detailed explanation of the plots.
`timePoints`	A character or numeric vector indicating the time points to be plotted. When using a character string to reference a time point, the value has to be an exact match to one of the existing timePoints. When using a number it will be matched by its number ("timeNumber") in the timePoints attribute of the TP object.

title	A character string used as title for the plot. If NULL a default title is added to the plot depending on plotType.
output	Should the plot be output to the current device? If FALSE only a (list of) ggplot object(s) is invisibly returned. Ignored if outFile is specified.
outFile	A character string indicating the .pdf file or .gif file (for plotType = "timeLapse") to which the plots should be written.
outFileOpts	A named list of extra options for the pdf outfile, e.g. width and height. See pdf for all possible options.

Value

Depending on the plot type either a ggplot object or a list of ggplot objects is invisibly returned.

rawPred plot

Plots the raw data (colored dots) overlayed with the predicted values from the fitted model (black dots). For each genotype a plot is made per plot/plant over time. These plots are put together in a 5x5 grid. By using the parameter genotypes a selection of genotypes can be plotted. Extra parameter options:

genotypes A character vector indicating the genotypes to be plotted.

plotChecks Should the check genotypes be included in the plot?

plotLine Should the data be displayed as lines? Default is FALSE.

corrPred plot

Plots the spatially corrected data (colored dots) overlayed with the predicted values from the fitted model (black dors). For each genotype a plot is made per plot/plant over time. These plots are put together in a 5x5 grid. By using the parameter genotypes a selection of genotypes can be plotted. Extra parameter options:

genotypes A character vector indicating the genotypes to be plotted.

plotChecks Should the check genotypes be included in the plot?

plotLine Should the data be displayed as lines? Default is FALSE.

herit plot

Plots the heritability over time. This plot is only available when genotype is fitted as random factor in the model. If geno.decomp is used when fitting the model, heritabilities are plotted for each level of geno.decomp in a single plot. Extra parameter options:

yLim A numerical vector of length two, used for setting the limits of the y-axis of the plot. If values outside of the plotting range are given, then these are ignored.

effDim plot

Plots the effective dimension over time for models fitted using SpATS. Extra parameter options:

whichED A character vector indicating which effective dimensions should be plotted. This should be a subset of "colId", "rowId", "fCol", "fRow", "fColRow", "colfRow", "fColfRow" and "surface". When `useRepId = TRUE`, the effective dimensions of "colId" and "rowId" become "RepId:colId" and "RepId:rowId". Default all effective dimensions are plotted.

EDType A character string specifying if the effective dimension ("dimension") or the ratio of effective dimensions ("ratio") should be plotted. Default the dimensions are plotted.

yLim A numerical vector of length two, used for setting the limits of the y-axis of the plot. If values outside of the plotting range are given, then these are ignored.

variance plot

Plots the residual, column and row variances over time for the fitted models. Extra parameter options:

yLim A numerical vector of length two, used for setting the limits of the y-axis of the plot. If values outside of the plotting range are given, then these are ignored.

timeLapse plot

Creates a time lapse of the spatial trends of models fitted using SpATS over time.

spatial plot

Creates five plots per time point, spatial plots of the raw data, fitted values, residuals and either BLUEs or BLUPs, and a histogram of the BLUEs or BLUPs. When SpATS was used for modeling an extra plot with the fitted spatial trend is included Extra parameter options:

spaTrend A character string indicating how the spatial trend should be displayed. Either "raw" for raw values, or "percentage" for displaying as a percentage of the original phenotypic values.

See Also

Other functions for spatial modeling: `fitModels()`, `getCorrected()`, `getEffDims()`, `getGenoPred()`, `getHerit()`, `getVar()`, `summary.fitMod()`

Examples

```
## Using the first example dataset (PhenovatorDat1):
## Create an object of class TP.
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"))

## Fit a SpATS model on three points:
modPhenoSp <- fitModels(TP = phenoTP,
                        trait = "EffpsII",
                        timePoints = c(1, 6, 36))

## Plot the spatial trends for one time point:
plot(modPhenoSp,
     timePoints = 36,
     plotType = "spatial",
     spaTrend = "percentage")


## Not run:
## Create a time lapse of all available time points:
plot(modPhenoSp,
     plotType = "timeLapse",
     outFile = "TimeLapse_modPhenoSp.gif")

## End(Not run)


## Plot the corrected values for a subset of four genotypes:
plot(modPhenoSp,
     plotType = "corrPred",
     genotypes = c("check1", "check2", "G007", "G058") )

## Plot the effective dimensions of all available time points in the model
## for a subset of effective dimensions:
plot(modPhenoSp,
     plotType = "effDim",
     whichED = c("colId", "rowId", "fColRow","colfRow"),
     EDType = "ratio")
```

`plot.HTPSpline` *Plot the results of a fitted spline.*

Description

Plot the results of a fitted spline.

Usage

```
## S3 method for class 'HTPSpline'
plot(
  x,
```

```
  ...,
  plotType = c("predictions", "derivatives", "derivatives2"),
  genotypes = NULL,
  plotIds = NULL,
  title = NULL,
  output = TRUE,
  outFile = NULL,
  outFileOpts = NULL
)
```

Arguments

x	An object of class HTPSpline.
...	Ignored.
plotType	A character string indicating which spline component should be plotted, either predictions, derivatives or second derivatives ("derivatives2").
genotypes	A character vector indicating the genotypes for which spline components should be plotted.
plotIds	A character vector indicating the plotIds for which spline components should be plotted.
title	A character string used as title for the plot. If NULL a default title is added to the plot depending on plotType.
output	Should the plot be output to the current device? If FALSE only a (list of) ggplot object(s) is invisibly returned. Ignored if outFile is specified.
outFile	A character string indicating the .pdf file to which the plots should be written. If NULL, no file is written.
outFileOpts	A named list of extra options for the pdf outfile, e.g. width and height. See pdf for all possible options.

Value

A list of object of class ggplot is invisibly returned.

See Also

Other functions for fitting splines: fitSpline()

Examples

```
## The data from the Phenovator platform have been corrected for spatial
## trends and outliers for single observations have been removed.

## Fit P-Splines on a subset of genotypes
subGeno <- c("G070", "G160")
fit.spline <- fitSpline(inDat = spatCorrectedVator,
                        trait = "EffpsII_corr",
                        genotypes = subGeno,
                        knots = 50)
```

```
## Visualize the P-Spline predictions for one genotype.
plot(fit.spline, genotypes = "G160")

## Visualize the first and second derivatives of the predictions for one plant.
plot(fit.spline, plotIds = "c10r29", plotType =  "derivatives")
plot(fit.spline, plotIds = "c10r29", plotType =  "derivatives2")
```

`plot.psHDM` *Plot function for class psHDM*

Description

This plot function provides five plots for objects of the class `psHDM` after fitting (`fitSplineHDM`) or predicting (`predict.psHDM`): (1) Population-specific growth curves (popTra), (2) Population and genotype-specific growth curves (for all genotypes, popGenoTra), (3) First-order derivative of the population and genotype-specific growth curves (for all genotypes, popGenoDeriv), (4) Genotype-specific deviations (for all genotypes, genoDev), and (5) Genotype- and plot-specific growth curves (for a selection of genotypes, genoPlotTra). If standard errors are available, 95% point wise confidence intervals are depicted.

Usage

```
## S3 method for class 'psHDM'
plot(
  x,
  ...,
  plotType = c("popTra", "popGenoTra", "popGenoDeriv", "genoDev", "genoPlotTra"),
  genotypes = NULL,
  genotypeNames = NULL,
  genotypeOrder = NULL,
  xlab = "Time",
  ylab = expression(tilde(y)[pgi](t)),
  title = NULL,
  themeSizeHDM = 15,
  output = TRUE,
  outFile = NULL,
  outFileOpts = NULL
)
```

Arguments

`x`	An object of class "psHDM" as obtained after fitting (`fitSplineHDM`) or predicting (`predict.psHDM`),
`...`	Not used.
`plotType`	A character string indicating which plot should be made.

genotypes	A character vector with the genotypes for which plots at plot level are desired. Only used when plotType == "genoPlotTra".
genotypeNames	A character vector with alternative names for the plotted genotypes (genotypes). If NULL the names of the genotypes are used. Only used when plotType == "genoPlotTra".
genotypeOrder	A vector with the order of the selected genotypes (genotypes). If NULL then the order in the data is preserved. Only used when plotType == "genoPlotTra".
xlab	The x-axis label of the plot.
ylab	The y-axis label of the plot.
title	A character string used as title for the plot. If NULL a default title is added to the plot depending on plotType.
themeSizeHDM	Reference size for the theme
output	Should the plot be output to the current device? If FALSE only a (list of) ggplot object(s) is invisibly returned. Ignored if outFile is specified.
outFile	A character string indicating the .pdf file to which the plots should be written. If NULL, no file is written.
outFileOpts	A named list of extra options for the pdf outfile, e.g. width and height. See pdf for all possible options.

References

Pérez-Valencia, D.M., Rodríguez-Álvarez, M.X., Boer, M.P. et al. A two-stage approach for the spatio-temporal analysis of high-throughput phenotyping data. Sci Rep 12, 3177 (2022). doi:10.1038/s41598022069359

See Also

Other functions for fitting hierarchical curve data models: fitSplineHDM(), predict.psHDM()

Examples

```
## The data from the Phenovator platform have been corrected for spatial
## trends and outliers for single observations have been removed.

## We need to specify the genotype-by-treatment interaction.
## Treatment: water regime (WW, WD).
spatCorrectedArch[["treat"]] <- substr(spatCorrectedArch[["geno.decomp"]],
                                       start = 1, stop = 2)
spatCorrectedArch[["genoTreat"]] <-
  interaction(spatCorrectedArch[["genotype"]],
              spatCorrectedArch[["treat"]], sep = "_")

## Fit P-Splines Hierarchical Curve Data Model for selection of genotypes.
fit.psHDM  <- fitSplineHDM(inDat = spatCorrectedArch,
                           trait = "LeafArea_corr",
                           genotypes = c("GenoA14_WD", "GenoA51_WD",
                                         "GenoB11_WW", "GenoB02_WD",
                                         "GenoB02_WW"),
```

```
                            time = "timeNumber",
                            pop = "geno.decomp",
                            genotype = "genoTreat",
                            plotId = "plotId",
                            difVar = list(geno = FALSE, plot = FALSE),
                            smoothPop = list(nseg = 4, bdeg = 3, pord = 2),
                            smoothGeno = list(nseg = 4, bdeg = 3, pord = 2),
                            smoothPlot = list(nseg = 4, bdeg = 3, pord = 2),
                            weights = "wt",
                            trace = FALSE)

## Plot the P-Spline predictions at the three levels of the hierarchy

## Population-specific growth curves.
plot(fit.psHDM,
    plotType = "popTra")

## Population and genotype-specific growth curves.
plot(fit.psHDM,
    plotType = "popGenoTra")

## First-order derivative of the population- and genotype-specific growth curves.
plot(fit.psHDM,
    plotType = "popGenoDeriv")

## Genotype-specific deviations.
plot(fit.psHDM,
    plotType = "genoDev")

## Genotype- and plot-specific growth curves.
plot(fit.psHDM,
    plotType = "genoPlotTra")
```

`plot.serieOut` *Plot outliers for series of observations*

Description

Plot the fitted spline, correlation matrix and PCA biplot for each of the genotypes. Outlying series of observations are shown as filled dots in the fitted spline plot, other observations are shown as open dots.

Usage

```
## S3 method for class 'serieOut'
plot(
  x,
  ...,
  reason = c("mean corr", "angle", "slope"),
```

```
  genotypes = NULL,
  geno.decomp = NULL,
  useTimeNumber = FALSE,
  timeNumber = NULL,
  title = NULL,
  output = TRUE
)
```

Arguments

x	An object of class serieOut.
...	Ignored.
reason	A character vector indicating which types of outliers should be plotted.
genotypes	A character vector indicating which genotypes should be plotted. If NULL all genotypes are plotted.
geno.decomp	A character vector indicating which levels of geno.decomp should be plotted. If NULL all levels are plotted. Ignored if geno.decomp was not used when fitting models.
useTimeNumber	Should the timeNumber be used instead of the timePoint in the labels on the x-axis?
timeNumber	If useTimeNumber = TRUE, a character vector indicating the column containing the numerical time to use.
title	A character string used as title for the plot. If NULL a default title is added to the plot depending on plotType.
output	Should the plot be output to the current device? If FALSE only a (list of) ggplot object(s) is invisibly returned. Ignored if outFile is specified.

Value

A list of ggplot objects is invisibly returned.

See Also

Other functions for detecting outliers for series of observations: detectSerieOut(), removeSerieOut()

Examples

```
## The data from the Phenovator platform have been corrected for spatial
## trends and outliers for single observations have been removed.

## Fit P-splines on a subset of genotypes
subGenoVator <- c("G160", "G151")
fit.spline <- fitSpline(inDat = spatCorrectedVator,
                        trait = "EffpsII_corr",
                        genotypes = subGenoVator,
                        knots = 50)

## Extract the data.frames with predicted values and P-Spline coefficients.
```

```
predDat <- fit.spline$predDat
coefDat <- fit.spline$coefDat

## The coefficients are then used to tag suspect time courses.
outVator <- detectSerieOut(corrDat = spatCorrectedVator,
                           predDat = predDat,
                           coefDat = coefDat,
                           trait = "EffpsII_corr",
                           genotypes = subGenoVator,
                           thrCor = 0.9,
                           thrPca = 30,
                           thrSlope = 0.7)

## The `outVator` can be visualized for selected genotypes.
plot(outVator, genotypes = "G151")

## Only visualize outliers tagged because of low correlation
## between slopes of the regression.
plot(outVator, genotypes = "G151", reason = "slope")
```

`plot.singleOut` *Plot outliers for single observations*

Description

Plot the fitted local regression, confidence intervals and detected outliers for each plotId.

Usage

```
## S3 method for class 'singleOut'
plot(x, ..., plotIds = NULL, outOnly = TRUE, output = TRUE)
```

Arguments

`x`	An object of class singleOut.
`...`	Ignored.
`plotIds`	A character vector of plotIds for which the outliers should be detected. If `NULL`, all plotIds in `TP` are used.
`outOnly`	Should only plots containing outliers be plotted?
`output`	Should the plot be output to the current device? If `FALSE` only a (list of) ggplot object(s) is invisibly returned. Ignored if `outFile` is specified.

Value

A list of ggplot objects is invisibly returned.

See Also

Other functions for detecting outliers for single observations: detectSingleOutMaize(), detectSingleOut(), removeSingleOut()

Examples

```
## Create a TP object containing the data from the Phenovator.
PhenovatorDat1 <- PhenovatorDat1[!PhenovatorDat1$pos %in%
                                   c("c24r41", "c7r18", "c7r49"), ]
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"))

## Select a subset of plants, for example here 9 plants.
plantSel <- phenoTP[[1]]$plotId[1:9]
# Then run on the subset.
resuVatorHTP <- detectSingleOut(TP = phenoTP,
                                trait = "EffpsII",
                                plotIds = plantSel,
                                confIntSize = 3,
                                nnLocfit = 0.1)

## Visualize the prediction by choosing a single plant...
plot(resuVatorHTP, plotIds = "c21r24", outOnly = FALSE)
## ...or a subset of plants.
plot(resuVatorHTP, plotIds = plantSel, outOnly = FALSE)
```

`plot.splineEst` *Plot the results of estimated spline parameters.*

Description

Plot the results of estimated spline parameters.

Usage

```
## S3 method for class 'splineEst'
plot(
  x,
  ...,
  plotType = c("box", "hist"),
```

```
  what = attr(x, "what"),
  title = NULL,
  output = TRUE,
  outFile = NULL,
  outFileOpts = NULL
)
```

Arguments

x	An object of class splineEst
...	Ignored.
plotType	A character string indicating the type of plot to be made.
what	The types of estimate that should be plotted.
title	A character string used as title for the plot. If NULL a default title is added to the plot depending on plotType.
output	Should the plot be output to the current device? If FALSE only a (list of) ggplot object(s) is invisibly returned. Ignored if outFile is specified.
outFile	A character string indicating the .pdf file to which the plots should be written. If NULL, no file is written.
outFileOpts	A named list of extra options for the pdf outfile, e.g. width and height. See pdf for all possible options.

Value

A list of objects of class ggplot is invisibly returned.

See Also

Other functions for spline parameter estimation: estimateSplineParameters()

---

plot.TP *Plot function for class TP*

Description

Plotting function for objects of class TP. Plots the layout of the platform for different time points within the TP object. Also a boxplot can be made for selected traits and time points and a plot of correlations between time points. Finally the raw data can be displayed per genotype. A detailed description and optional extra parameters for the different plots are given in the sections below.

Usage

```
## S3 method for class 'TP'
plot(
  x,
  ...,
  plotType = c("layout", "box", "cor", "raw"),
  timePoints = names(x),
  title = NULL,
  traits = NULL,
  output = TRUE,
  outFile = NULL,
  outFileOpts = NULL
)
```

Arguments

x	An object of class TP.
...	Extra plot options. Described per plotType in their respective section.
plotType	A single character string indicating which plot should be made. See the sections below for a detailed explanation of the plots.
timePoints	A character or numeric vector indicating the time points to be plotted. When using a character string to reference a time point, the value has to be an exact match to one of the existing timePoints. When using a number it will be matched by its number ("timeNumber") in the timePoints attribute of the TP object.
title	A character string used as title for the plot. If NULL a default title is added to the plot depending on plotType.
traits	A character vector indicating the traits to be plotted. If plotType = "layout" only a single trait may be plotted. For the other plotTypes, providing multiple traits will create multiple plots.
output	Should the plot be output to the current device? If FALSE only a (list of) ggplot object(s) is invisibly returned. Ignored if outFile is specified.
outFile	A character string indicating the .pdf file to which the plots should be written. If NULL, no file is written.
outFileOpts	A named list of extra options for the pdf outfile, e.g. width and height. See pdf for all possible options.

Value

Depending on the plot type, either a ggplot object or a list of ggplot objects is invisibly returned.

Layout Plot

Plots the layout of the platform for selected time points (all available time points by default). This plot can only be made for time points that contain both row (rowNum) and column (colNum) information. If either one of those is missing the timePoint is skipped with a warning. If replicates (repId) are available, a black line is plotted between diffent replicates. Missing plots are indicated

in white. This can either be single plots in a time point or complete missing columns or rows. Extra parameter options:

showGeno Should individual genotypes be labeled in the plot? Defaults to FALSE

highlight A character vector of genotypes to be highlighted in the plot.

Box Plot

Creates a boxplot per selected trait grouped by time point (all available time points by default). Extra parameter options:

groupBy A character string indicating a column in TP by which the boxes in the plot should be grouped. By default the boxes are grouped per time point.

colorBy A character string indicating a column in TP by which the boxes are colored. Coloring will be done within the groups indicated by the groupBy parameter, e.g. per replicate within each time point using repId.

orderBy A character string indicating the way the boxes should be ordered. Either "alphabetic" for alphabetical ordering of the groups, "ascending" for ordering by ascending mean, or "descending" for ordering by descending mean. By default boxes are ordered alphabetically.

Correlation Plot

Draws a heatmap of correlations of raw data between time points per selected trait for selected time points (all available time points by default).

Raw data plot

Create a plot of the raw data of the selected trait over time for selected time points (all available time points by default). Plots are grouped by genotype, or by genotype x treatment when the geno.decomp option is specified. By default, all the genotypes will be plotted which might take time and memory when the output is not saved in a file (see parameter outFile). Extra parameter options:

genotypes A character vector indicating the genotypes to be plotted.

geno.decomp A character vector indicating the grouping of the genotypes to be plotted.

plotLine Should the data be displayed as lines? Default is FALSE.

See Also

Other functions for data preparation: as.data.frame.TP(), createTimePoints(), getTimePoints(), removeTimePoints(), summary.TP()

Examples

```
## Create a TP object containing the data from the Phenovator.
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"))

## 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"))

## 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")

## 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))

## Plot the raw data of four genotypes for the trait "EffpsII":
plot(phenoTP,
     traits = "EffpsII",
     plotType = "raw",
     genotypes = c("G001","G002","check1","check2"))
```

`predict.psHDM` *Predict the P-Splines Hierarchical Curve Data Model*

Description

Function that predicts the P-spline Hierarchical Curve Data Model (see `fitSplineHDM`) on a dense grid. It provides standard errors for curves at each level of the hierarchy. User has to be aware that standard errors at the plot level demand large memory. We suggest set that option at the `FALSE` level

Usage

```
## S3 method for class 'psHDM'
predict(
  object,
  newtimes,
```

```
  pred = list(pop = TRUE, geno = TRUE, plot = TRUE),
  se = list(pop = TRUE, geno = TRUE, plot = FALSE),
  trace = TRUE,
  ...
)
```

Arguments

object	An object of class "psHDM" as obtained after fitting (fitSplineHDM) the P-spline Hierarchical Curve Data Model
newtimes	A numeric vector with timepoints at which predictions are desired
pred	A list that controls the hierarchical levels at which predictions are desired (population/genotypes/plots). The default is TRUE.
se	A list that controls the hierarchical levels at which standard errors are desired (population/genotypes/plots). The default is TRUE except at the plot level.
trace	An optional value that controls the function trace. The default is TRUE.
...	Not used.

Value

An object of class psHDM, a list with the following outputs: predict.psHDM newtimes A numeric vector with the timepoints at which predictions and/or standard errors have been obtained. popLevel A data.frame with the estimated population trajectories and first and second order derivatives, and if required their respective standard errors, at the newtimes. genoLevel A data.frame with the estimated genotype-specific deviations and trajectories and their respective first and second order derivatives, and if required their respective standard errors, at the newtimes. plotLevel A data.frame with the estimated plot-specific deviations and trajectories and their respective first and second order derivatives, and if required their respective standard errors, at the newtimes. plotObs A data.frame with the raw data at the original timepoints.

References

Pérez-Valencia, D.M., Rodríguez-Álvarez, M.X., Boer, M.P. et al. A two-stage approach for the spatio-temporal analysis of high-throughput phenotyping data. Sci Rep 12, 3177 (2022). doi:10.1038/s41598022069359

See Also

Other functions for fitting hierarchical curve data models: fitSplineHDM(), plot.psHDM()

Examples

```
## The data from the Phenovator platform have been corrected for spatial
## trends and outliers for single observations have been removed.

## We need to specify the genotype-by-treatment interaction.
## Treatment: water regime (WW, WD).
spatCorrectedArch[["treat"]] <- substr(spatCorrectedArch[["geno.decomp"]],
                                       start = 1, stop = 2)
```

```
spatCorrectedArch[["genoTreat"]] <-
  interaction(spatCorrectedArch[["genotype"]],
              spatCorrectedArch[["treat"]], sep = "_")

## Fit P-Splines Hierarchical Curve Data Model for selection of genotypes.
fit.psHDM  <- fitSplineHDM(inDat = spatCorrectedArch,
                           trait = "LeafArea_corr",
                           genotypes = c("GenoA14_WD", "GenoA51_WD",
                                         "GenoB11_WW", "GenoB02_WD",
                                         "GenoB02_WW"),
                           time = "timeNumber",
                           pop = "geno.decomp",
                           genotype = "genoTreat",
                           plotId = "plotId",
                           difVar = list(geno = FALSE, plot = FALSE),
                           smoothPop = list(nseg = 4, bdeg = 3, pord = 2),
                           smoothGeno = list(nseg = 4, bdeg = 3, pord = 2),
                           smoothPlot = list(nseg = 4, bdeg = 3, pord = 2),
                           weights = "wt",
                           trace = FALSE)

## Predict the P-Splines Hierarchical Curve Data Model on a dense grid
## with standard errors at the population and genotype levels
pred.psHDM <- predict(object = fit.psHDM,
                      newtimes = seq(min(fit.psHDM$time[["timeNumber"]]),
                                     max(fit.psHDM$time[["timeNumber"]]),
                                     length.out = 100),
                      pred = list(pop = TRUE, geno = TRUE, plot = TRUE),
                      se = list(pop = TRUE, geno = TRUE, plot = FALSE))

## Plot the P-Spline predictions at the three levels of the hierarchy

## Plots at population level.
plot(pred.psHDM,
    plotType = "popTra")

## Plots at genotype level.
plot(pred.psHDM,
    plotType = "popGenoTra")

## Plots of derivatives at genotype level.
plot(pred.psHDM,
    plotType = "popGenoDeriv")

## Plots of deviations at genotype level.
plot(pred.psHDM,
    plotType = "genoDev")

## Plots at plot level.
plot(pred.psHDM,
    plotType = "genoPlotTra")
```

removeSerieOut *Replace outliers for series of observations by NA*

Description

Function for replacing outliers for series of observations in the data by NA. The input can either be a data.frame, specified in dat, or the output of the fitSpline function, specified in fitSpline. Exactly one of these should be provided as input for the function.

Usage

```
removeSerieOut(
  dat = NULL,
  fitSpline = NULL,
  serieOut,
  reason = c("mean corr", "angle", "slope"),
  traits = attr(x = serieOut, which = "trait")
)
```

Arguments

dat	A data.frame.
fitSpline	An object of class HTPSpline, the output of the fitSpline function.
serieOut	A data.frame with at least the column plotId with values corresponding to those in dat/fitSpline.
reason	A character vector indicating which types of outliers should be replaced by NA.
traits	The traits that should be replaced by NA. When using the output of detectSerieOut as input for serieOut this defaults to the trait used for when detecting the outliers.

Value

Depending on the input either a data.frame or an object of class HTPSpline for which the outliers specified in serieOut are replaced by NA.

See Also

Other functions for detecting outliers for series of observations: detectSerieOut(), plot.serieOut()

Examples

```
## Run the function to fit P-splines on a subset of genotypes.
subGenoVator <- c("G160", "G151")
fit.spline <- fitSpline(inDat = spatCorrectedVator,
                        trait = "EffpsII_corr",
                        genotypes = subGenoVator,
                        knots = 50)
```

```
## Extract the tables of predicted values and P-spline coefficients.
predDat <- fit.spline$predDat
coefDat <- fit.spline$coefDat

## The coefficients are then used to tag suspect time courses
outVator <- detectSerieOut(corrDat = spatCorrectedVator,
                           predDat = predDat,
                           coefDat = coefDat,
                           trait = "EffpsII_corr",
                           genotypes = subGenoVator,
                           thrCor = 0.9,
                           thrPca = 30,
                           thrSlope = 0.7)

## Replace the outliers by NA in the corrected data.
spatCorrectedVatorOut <- removeSerieOut(dat = spatCorrectedVator,
                                        serieOut = outVator)

## Only replace the slope outliers by NA in the corrected data.
spatCorrectedVatorOut2 <- removeSerieOut(dat = spatCorrectedVator,
                                        serieOut = outVator,
                                        reason = "slope")

## Replace the outliers by NA in the corrected data.
## Replace both the corrected value and the raw trait value by NA.
spatCorrectedVatorOut3 <-
  removeSerieOut(dat = spatCorrectedVator,
                 serieOut = outVator,
                 traits = c("EffpsII", "EffpsII_corr"))
```

`removeSingleOut` *Replace outliers for single observations by NA*

Description

Function for replacing outliers for single observations by NA.

Usage

```
removeSingleOut(TP, singleOut, trait = attr(x = singleOut, which = "trait"))
```

Arguments

`TP`	An object of class TP.
`singleOut`	A data.frame with at least the columns plotId and timePoint with values corresponding to those in TP. If a column outlier is present, as in the output of `detectSingleOut`, only plotId x timePoint combinations for which outlier = 1

will be set to NA. If no column outlier is present, all observations in singleOut will be set to NA.

`trait` The trait that should be set to NA. Can be ignored when using the output of `detectSingleOut` as input.

Value

An object of class TP, the input with the outlier replaced by NA.

See Also

Other functions for detecting outliers for single observations: `detectSingleOutMaize()`, `detectSingleOut()`, `plot.singleOut()`

Examples

```
## Create a TP object containing the data from the Phenovator.
PhenovatorDat1 <- PhenovatorDat1[!PhenovatorDat1$pos %in%
                                   c("c24r41", "c7r18", "c7r49"), ]
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"))

## First select a subset of plants, for example here 9 plants.
plantSel <- phenoTP[[1]]$plotId[1:9]
# Then run on the subset
resuVatorHTP <- detectSingleOut(TP = phenoTP,
                                trait = "EffpsII",
                                plotIds = plantSel,
                                confIntSize = 3,
                                nnLocfit = 0.1)

## Replace the studied trait by NA for the plants marked as outliers.
phenoTPOut <- removeSingleOut(phenoTP, resuVatorHTP)
```

---

removeTimePoints *Remove time points from an object of class TP*

Description

Function for removing selected time points from an object of class TP.

Usage

```
removeTimePoints(TP, timePoints)
```

Arguments

TP	An object of class TP.
timePoints	A character or numeric vector indicating the time points to be removed. When using a character string to reference a time point, the value has to be an exact match to one of the existing timePoints. When using a number it will be matched by its number ("timeNumber") in the timePoints attribute of the TP object.

Value

An object of class TP, the input with the selected time points removed.

See Also

Other functions for data preparation: `as.data.frame.TP()`, `createTimePoints()`, `getTimePoints()`, `plot.TP()`, `summary.TP()`

Examples

```
## Create a TP object containing the data from the Phenovator.
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"))
## Remove the first and last time point from the TP object.
phenoTPNew <- removeTimePoints(phenoTP,
                               timePoints = c(1, 73))

## Compare by looking at summaries.
summary(phenoTP)
summary(phenoTPNew)
```

RootDat1 *Greenhouse data for an experiment in the RootPhAir platform.*

Description

A dataset containing greenhouse data from the RootPhAir platform (UCLouvain, Belgium). It consists of one experiment with one aeroponic tanks with 340 maize plants. The studied traits are the root tip coordinates in y and x axis, extracted from the pictures over time. Plants were pictured every 2 hours for 10 days. This dataset was kindly provided by Xavier Draye.

Usage

```
RootDat1
```

Format

A data.frame with 16,275 rows and 10 columns:

Exp Experiment number

thermalTime Thermal time cumulated

Genotype Genotype

plantId Unique pot using tank and rowcol coordinate

Tank Tank A or B

Strip Number of strip of five plants (i.e. row coordinate)

Pos Position within th strip (i.e. column coordinate)

tipPos_x Position of the root tip in x axis

tipPos_y Position of the root tip in y axis

Time Time of measurement

---

spatCorrectedArch *Maize data corrected for spatial trends.*

Description

This dataset contains the corrected data obtained by (1) removing outliers for single observations and (2) running a spatial model on the PhenoarchDat1 dataset. See the vignettes for details.

Usage

```
spatCorrectedArch
```

Format

A data.frame with 37,038 rows and 9 columns:

timeNumber Time number obtained after formatting the original dataset with the function createTP.

timePoint Original time point.

LeafArea_corr Leaf area, corrected data

LeafArea Leaf area from the picture, raw data

wt Weight factor

genotype Genotypes

geno.decomp Combination of treatment levels to decompose the genotypic variance (see vignettes)

colId Column coordinate

rowId Row coordinate

plotId Unique pot ID using rowcol coordinates

---

spatCorrectedVator *Arabidopsis data corrected for spatial trends.*

Description

This dataset contains the corrected data obtained by (1) removing outliers for single observations and (2) running a spatial model on the PhenovatorDat1 dataset. See the vignettes for details.

Usage

```
spatCorrectedVator
```

Format

A data.frame with 103,801 rows and 11 columns:

timeNumber Time number obtained after formatting the original dataset with the function createTP.

timePoint Original time point.

EffpsII_corr Efficiency of the photosystem II, corrected data

EffpsII Efficiency of the photosystem II, raw data

genotype Genotypes

repId Block define after sowing for post-blocking.

Image_pos Position of the camera

check Status of the genotypes: check for the reference genotypes, noCheck for the others.

colId Column coordinate

rowId Row coordinate

plotId Unique pot ID using rowcol coordinates

spatPredArch *Maize data, genotypic predictions.*

Description

This dataset contains the genotypic predictions obtained by (1) removing outliers for single observations and (2) running a spatial model on the PhenoarchDat1 dataset. See the vignettes for details.

Usage

```
spatPredArch
```

Format

A data.frame with 6,120 rows and 6 columns:

timeNumber Time number obtained after formatting the original dataset with the function createTP.

timePoint Original time point.

geno.decomp Combination of treatment levels to decompose the genotypic variance (see vignettes)

genotype Genotypes

predicted.values Biomass, predicted values

standard.errors Standard errors associated with the prediction

summary.fitMod *Summary function for fitMod objects*

Description

Function for creating a short summary of the contents of a TP object. The summary consists of the name of the experiment, the number of time points, the engine used to fit the models and, in case spatial models where fitted using asreml, the selected spatial model.

Usage

```
## S3 method for class 'fitMod'
summary(object, ...)
```

Arguments

object	An object of class fitMod.
...	Ignored.

Value

No return value, a summary is printed.

See Also

Other functions for spatial modeling: `fitModels()`, `getCorrected()`, `getEffDims()`, `getGenoPred()`, `getHerit()`, `getVar()`, `plot.fitMod()`

Examples

```
## Using the first example dataset (PhenovatorDat1):
## Create an object of class TP.
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"))

## Fit a SpATS model on few time points:
modPhenoSp <- fitModels(TP = phenoTP,
                        trait = "EffpsII",
                        timePoints = c(1, 6, 36))

## Create a summary.
summary(modPhenoSp)
```

`summary.TP` *Summary function for TP objects*

Description

Function for creating a short summary of the contents of a `TP` object. The summary consists of the name of the experiment, the number of time points, the first and last time point and the genotypes defined as checks.

Usage

```
## S3 method for class 'TP'
summary(object, ...)
```

Arguments

object	An object of class TP.
...	Ignored.

Value

No return value, a summary is printed.

See Also

Other functions for data preparation: as.data.frame.TP(), createTimePoints(), getTimePoints(), plot.TP(), removeTimePoints()

Examples

```
## Create a TP object containing the data from the Phenovator.
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"))
## Create a summary.
summary(phenoTP)
```

Index