Package ‘odprism’

November 30, 2011

Type Package

Title Optimal design and performance of random intercept and slope models.

Version 1.1

Date 2011-11-07

Author Martijn van de Pol

Maintainer Martijn van de Pol <m.van.de.pol@myscience.eu>

Description Simulation functions to assess the optimal design and performance of random intercept and slope models (i.e. mixed models), which can be used to a priori determine adequate sampling designs for e.g. reaction norm studies. Functions allow users to vary the sampling design in terms of number of grouping units sampled (e.g. individuals, schools, populations,etc.) and replicates per grouping unit (unbalanced as well balanced datasets) and also allow users to vary the parameter conditions used to generate the data. Subsequently, the performance of mixed models (based on lme4 package) fitted on these datasets is assessed in terms of the accuracy and the precision of estimates of fixed and random parameter, as well as the statistical power.

License GPL-2

Encoding UTF-8

Depends R (>= 2.11.0), stats, graphics, lme4, mvtnorm

Imports Matrix, fields

Repository CRAN

Date/Publication 2011-11-08 06:54:51

R topics documented:

`odprism-package` *The package 'odprism' can be used to determine the Optimal Design and Performane of Random Intercept and Slope Models*

Description

Simulation functions to assess the optimal design for studies planning to use random intercept and/or slopes models to analyze their data, such as reaction norm studies. The functions 'odprim', 'odprismB' and odprismU' allow users to simulate many dataset while varying the sampling design in terms of number of grouping units sampled (e.g. individuals, schools, populations, etc.) and replicates per grouping unit (balanced as well as unbalanced designs) and also allow users to vary the parameter conditions used to generate the data. Subsequently, the performance of mixed models (based on package "lme4")fitted on these datasets is assessed in terms of the accuracy and the precision of estimates of fixed and random parameter, as well as the statistical power. Results can be visualised using the functions 'plot2d.prism' and 'plot3d.prism'.

Details

Package:	odprism
Type:	Package
Version:	1.0
Date:	2011-06-11
License:	GPL 2.00
LazyLoad:	yes

Author(s)

Martijn van de Pol

References

van de Pol, M. 2012 Quantifying individual reaction norms: How study design affects the accuracy, precision and power of random regression models. Methods in Ecology & Evolution, In press.

See Also

odprim, odprismB, odprismU, plot2d.prism, plot3d.prism, DataExample

---

DataExample *Example of results that can be obtained with package odprism*

Description

Example of results from a simulation obtained with the code

```
DataExample <-
   odprismB(indiv= c(10,20,30,40,50,75,100,150,200),
            repl = c(2,3,4,5,6,7,8,9,10), random = c(0.2,0.1,0.5),
            alpha= 0.05, Quant = c(0.025,0.25,0.75,0.975) )
```

Usage

```
data(DataExample)
```

Format

A data frame with 81 observations on the following 43 variables.

`G` a numeric vector describing the number of grouping units sampled

`R` a numeric vector describing the number of replicates per grouping unit sampled

`Q1Int` a numeric vector describing the lowest quantile of the distribution of the parameter estimate of intercept beta0 (e.g. the 2.5-percentile)

`Q2Int` a numeric vector describing the lower quantile of the distribution of the parameter estimate of intercept beta0 (e.g. the 25-percentile)

`EstInt` a numeric vector describing the median of the parameter estimate of intercept beta0 accross all simulated datasets (e.g. the 50-percentile)

`Q3Int` a numeric vector describing the upper quantile of the distribution of the parameter estimate of intercept beta0 (e.g. the 75-percentile)

`Q4Int` a numeric vector describing the highest quantile of the distribution of the parameter estimate of intercept beta0 (e.g. the 97.5-percentile)

`Q1Slope` a numeric vector describing the lowest quantile of the distribution of the parameter estimate of slope betaX (e.g. the 2.5-percentile)

`Q2Slope` a numeric vector describing the lower quantile of the distribution of the parameter estimate of slope betaX (e.g. the 25-percentile)

`EstSlope` a numeric vector describing the median of the parameter estimate of slope betaX accross all simulated datasets (e.g. the 50-percentile)

`Q3Slope` a numeric vector describing the upper quantile of the distribution of the parameter estimate of slope betaX (e.g. the 75-percentile)

Q4Slope a numeric vector describing the highest quantile of the distribution of the parameter estimate of slope betaX (e.g. the 97.5-percentile)

Q1VI a numeric vector describing the lowest quantile of the distribution of the parameter estimate of VI (e.g. the 2.5-percentile)

Q2VI a numeric vector describing the lower quantile of the distribution of the parameter estimate of VI (e.g. the 25-percentile)

EstVI a numeric vector describing the median of the parameter estimate of VI accross all simulated datasets (e.g. the 50-percentile)

Q3VI a numeric vector describing the upper quantile of the distribution of the parameter estimate of VI (e.g. the 75-percentile)

Q4VI a numeric vector describing the highest quantile of the distribution of the parameter estimate of VI (e.g. the 97.5-percentile)

Q1VS a numeric vector describing the lowest quantile of the distribution of the parameter estimate of VS (e.g. the 2.5-percentile)

Q2VS a numeric vector describing the lower quantile of the distribution of the parameter estimate of VS (e.g. the 25-percentile)

EstVS a numeric vector describing the median of the parameter estimate of VS accross all simulated datasets (e.g. the 50-percentile)

Q3VS a numeric vector describing the upper quantile of the distribution of the parameter estimate of VS (e.g. the 75-percentile)

Q4VS a numeric vector describing the highest quantile of the distribution of the parameter estimate of VS (e.g. the 97.5-percentile)

Q1CorIS a numeric vector describing the lowest quantile of the distribution of the parameter estimate of CORIS (e.g. the 2.5-percentile)

Q2CorIS a numeric vector describing the lower quantile of the distribution of the parameter estimate of CORIS (e.g. the 25-percentile)

EstCorIS a numeric vector describing the median of the parameter estimate of CORIS accross all simulated datasets (e.g. the 50-percentile)

Q3CorIS a numeric vector describing the upper quantile of the distribution of the parameter estimate of CORIS (e.g. the 75-percentile)

Q4CorIS a numeric vector describing the highest quantile of the distribution of the parameter estimate of CORIS (e.g. the 97.5-percentile)

Q1VR a numeric vector describing the lowest quantile of the distribution of the parameter estimate of VR (e.g. the 2.5-percentile)

Q2VR a numeric vector describing the lower quantile of the distribution of the parameter estimate of VR (e.g. the 25-percentile)

EstVR a numeric vector describing the median of the parameter estimate of VR accross all simulated datasets (e.g. the 50-percentile)

Q3VR a numeric vector describing the upper quantile of the distribution of the parameter estimate of VR (e.g. the 75-percentile)

Q4VR a numeric vector describing the highest quantile of the distribution of the parameter estimate of VR (e.g. the 97.5-percentile)

PowerFixInt a numeric vector describing the proportion of simulation that rejected the null hypothesis of no effect for beta0

PowerFixSlope a numeric vector describing the proportion of simulation that rejected the null hypothesis of no effect for betaX

PowerRandInt a numeric vector describing the proportion of simulation that rejected the null hypothesis of no effect for VI

PowerRandSlope a numeric vector describing the proportion of simulation that rejected the null hypothesis of no effect for VS=CorIS=0

Int a numeric vector describing the value of intercept beta0 used to generate the datasets

Slope a numeric vector describing the value of slope betaX used to generate the datasets

VI a numeric vector describing the value of the among-grouping units variance in intercepts VI used to generate the datasets

VS a numeric vector describing the value of the among-grouping units variance in slopes VS used to generate the datasets

CorIS a numeric vector describing the value of the correlation between intercepts and slopes CorIS used to generate the datasets

VR a numeric vector describing the value of the within-grouping units (residual) variance VR used to generate the datasets

FLAG a logical vector coding for balanced datasets FLAG=FALSE and unbalanced datasets FLAG=TRUE

Source

van de Pol, M. 2012 Quantifying individual reaction norms: How study design affects the accuracy, precision and power of random regression models. Methods in Ecology & Evolution, In press.

References

van de Pol, M. 2012 Quantifying individual reaction norms: How study design affects the accuracy, precision and power of random regression models. Methods in Ecology & Evolution, In press.

Examples

```
## see ?odprism  for more examples, and above how DataExample is produced

data(DataExample)

## plot DataExample in various ways:
plot2d.prism(x=DataExample, variable="VS", xvar="G", cons=10)
plot3d.prism(x=DataExample, variable="VS")
```

`odprim` *Simulation function to asses the optimal design (balanced datasets) for studies planning to use random intercept models to analyze their data.*

Description

`odprim` is a simulation functions that allows users to vary the sampling design in terms of number of grouping units sampled (e.g. individuals, schools, populations, etc.) and replicates per grouping unit (limited to balanced datasets; all individuals have the same number of replicates) and also allow users to vary the parameter conditions used to generate the data. Subsequently, the performance of random intecept models of the form `Y~X+X2+(1|Individual)` fitted on these datasets is assessed in terms of the accuracy and the precision of estimates of fixed and random parameter, as well as the statistical power. Y values are generated using the formula Y=b0+bX*X+bX2*X2+u+e, where residual noise e is generated by a Gaussian random variable with mean zero and residual variance VR and the variation between grouping units in their intercept u is generated by a normal distribution with mean zero and varaince VI. X values are generated by a random normal variable with zero mean and variance VX, where X is a variable that varies both within and among grouping units, while X2 only varies among grouping units.

Usage

```
odprim(indiv, repl, sims = 1000, fixed, random,
  Xvar = c(1, 0, FALSE), Xvar2 = 1, alpha = 0,
  Quant = c(0.025, 0.25, 0.75, 0.975), Verbal = TRUE)
```

Arguments

`indiv` Number of individuals (or any other type of grouping units) sampled

`repl` Number of replicates per individual

`sims` Number of different simulated dataset that are generated. Default: `sims=1000`

`fixed` Vector of parameter values for b0, bX, and optionally bX2 which are used to generate trait values (i.e. mean intercept and mean slopes of X and X2). If fixed[3] is not provided (e.g. when `fixed=c(0,1)`) then the varaible X2 is removed from the model.

`random` Vector of parameter values VI and VR, which are used to generate trait values (residual varaince and variance in intercepts). If no value is given for VR (e.g. when `random=0.2`) it is assumed VR=1-VI.

`Xvar` Vector describing properties used to generate X values. The first entry gives the value of VX, the second value the autocorrelation in X between subsequent samping occasions, if the third entry is `TRUE` all individuals sampled at a specific occasion are given the same X value, while if `FALSE` all individuals are given different values at a specific occasion. Default: `Xvar=c(1,0,FALSE)`.

`Xvar2` Vector describing properties used to generate X2 values. The first entry gives the value of VX2, such that X2~Normal(0,VX2).

`alpha` If `alpha!=0` then the statistical power is calculated, defined as the proportion of analysed simulated dataset that resulted in P-value < `alpha`. P-values of fixed effects (b0 & bX) were generated using Wald-tests, while P-value of random effects were generated using likelihood rate tests. Specifically, the P-value for the random slopes term was obtained by comparing the deviance of full model `Y~X+(X|Individual)` with the reduced model `Y~X+(1||Individual)`, while the P-value for the random intercept term was obtained by comparing the deviance of full model `Y~X+(1|Individual)` with the reduced model `Y~X`. Default: `alpha=0`.

`Quant` Each sampling design is simulated sims times. The argument `Quant` is used to describe which percentiles of the distribution of parameter estimates should be stored. Default: `Quant=c(0.025,0.25,0.75,0.975)`.

`Verbal` If `TRUE`, the function prints to the screen which combination of indiv and repl it is currently evaluating. Default: `Verbal=TRUE`

Details

Please be aware that statistical inference in terms of P-values for fixed and especially for random parameters is a contentious topic. The methods used to calculate P-values here (Wald for fixed effects and LRT test for random effects) are widely used, but not necessarily the best. More generally, instead of focussing on statisical power I would advice to focus more on the acuracy and precision of parameter estimates. Therefore the default options (`alpha=0`) do not generate descriptions of statistical power.

Value

Dataframe of class ’odprism’ that contains the chosen values used to generate the data as the distribution of estimates derived from the random regression models applied to these datasets. For example the column ’VR’ gives the value used to generate residual noise, while the columns ’Q1VR’, ’Q2VR’, ’EstVR’, ’Q3VR’ and ’Q4VR’ give respectively the 2.5, 25, 50 (i.e. median), 75 and 97.5 percentiles of the parameters estimates of VR in all simulated dataset. See the help file of DataExample for more details.

Author(s)

Martijn van de Pol

References

van de Pol, M. 2012 Quantifying individual reaction norms: How study design affects the accuracy, precision and power of random regression models. Methods in Ecology & Evolution, In press.

See Also

`odprismB`, `odprismU`, `plot2d.prism`, `plot3d.prism`, `DataExample`

Examples

```
## Note that this example (model is Y~X+(1|Individual)) has only few sims
results<-odprim(indiv=c(10,25,50,75,100), repl=c(2,4,6,8,10),
  fixed=c(0,1), random=0.2, sims=10, alpha=0.05)
results
plot2d.prism(x=results, variable="bX", xvar="G", cons=2)

## The difference wht the above model is that here X2 is included,
## which is a covariate that varies only among grouping units.
results<-odprim(indiv=c(10,25,50,75,100), repl=c(2,4,6,8,10),
  fixed=c(0,1,1), random=0.2, sims=10, alpha=0.05)
results
plot3d.prism(x=results, variable="VI")
```

---

odprismB — *Simulation function to asses the optimal design (balanced datasets) for studies planning to use random intercept and slopes models to analyze their data.*

Description

odprismB is a simulation functions that allows users to vary the sampling design in terms of number of grouping units sampled (e.g. individuals, schools, populations, etc.) and replicates per grouping unit (limited to balanced datasets; all individuals have the same number of replicates) and also allow users to vary the parameter conditions used to generate the data. Subsequently, the performance of random regression models of the form `Y~X+(X|Individual)` fitted on these datasets is assessed in terms of the accuracy and the precision of estimates of fixed and random parameter, as well as the statistical power. Y values are generated using the formula Y=b0+bX*X+u+e, where residual noise e is generated by a Gaussian random varibale with mean zero and variance VR and the variation between grouping units in their intercept and slopes u is generated by a multivariate normal distribution with mean zero and covariance matrix (VI, CorIS, VS). X values are generated by a random normal variable with zero mean and varaince VX.

Usage

```
odprismB(indiv, repl, sims = 1000, fixed = c(0, 0), correl = TRUE,
  random, Xvar = c(1, 0, FALSE), alpha = 0,
  Quant = c(0.025, 0.25, 0.75, 0.975), Verbal = TRUE)
```

Arguments

`indiv`	Number of individuals (or any other type of grouping units) sampled
`repl`	Number of replicates per individual
`sims`	Number of different simulated dataset that are generated. Default: `sims=1000`
`fixed`	Vector of parameter values b0 and bX, which are used to generate trait values (i.e. mean intercept and mean slope). Default `fixed=c(0,0)`.

`correl` If `correl=TRUE` then the third entry in the vector of argument `random` is taken to be CorIS (correlation intercept and slopes), if `FALSE` then it is taken to be CovIS (covariance intercept and slopes). Default: `correl=TRUE`.

`random` Vector of parameter values VI, VS, CorIS/CovIS and VR, which are used to generate trait values (residual varaince and (co)variance in intercepts and slopes). If no value is given for VR (e.g. when `random=c(0.2,0.1,0.5)`) it is assumed VR=1-VI.

`Xvar` Vector describing properties used to generate X values. The first entry gives the value of VX, the second value the autocorrelation in X between subsequent samping occasions, if the third entry is `TRUE` all individuals sampled at a specific occasion are given the same X value, while if `FALSE` all individuals are given different values at a specific occasion. Default: `Xvar=c(1,0,FALSE)`.

`alpha` If `alpha!=0` then the statistical power is calculated, defined as the proportion of analysed simulated dataset that resulted in P-value < `alpha`. P-values of fixed effects (b0 & bX) were generated using Wald-tests, while P-value of random effects were generated using likelihood rate tests. Specifically, the P-value for the random slopes term was obtained by comparing the deviance of full model `Y~X+(X|Individual)` with the reduced model `Y~X+(1||Individual)`, while the P-value for the random intercept term was obtained by comparing the deviance of full model `Y~X+(1|Individual)` with the reduced model `Y~X`. Default: `alpha=0`.

`Quant` Each sampling design is simulated sims times. The argument `Quant` is used to describe which percentiles of the distribution of parameter estimates should be stored. Default: `Quant=c(0.025,0.25,0.75,0.975)`.

`Verbal` If `TRUE`, the function prints to the screen which combination of indiv and repl it is currently evaluating. Default: `Verbal=TRUE`

Details

Please be aware that statistical inference in terms of P-values for fixed and especially for random parameters is a contentious topic. The methods used to calculate P-values here (Wald for fixed effects and LRT test for random effects) are widely used, but not necessarily the best. More generally, instead of focussing on statisical power I would advice to focus more on the acuracy and precision of parameter estimates. Therefore the default options (`alpha=0`) do not generate descriptions of statistical power.

Value

Dataframe of class 'odprism' that contains the chosen values used to generate the data as the distribution of estimates derived from the random regression models applied to these datasets. For example the column 'VR' gives the value used to generate residual noise, while the columns 'Q1VR', 'Q2VR', 'EstVR', 'Q3VR' and 'Q4VR' give respectively the 2.5, 25, 50 (i.e. median), 75 and 97.5 percentiles of the parameters estimates of VR in all simulated dataset. See the help file of DataExample for more details.

Author(s)

Martijn van de Pol

References

van de Pol, M. 2012 Quantifying individual reaction norms: How study design affects the accuracy, precision and power of random regression models. Methods in Ecology & Evolution, In press.

See Also

odprim, odprismU, plot2d.prism, plot3d.prism, DataExample

Examples

```
## Note that this example has only few sims
results<-odprismB(indiv=c(10,25,50,75,100), repl=c(2,4,6,8,10),
  random=c(0.2,0.1,0.5), sims=10, alpha=0.05)
results
plot2d.prism(x=results, variable="VS", xvar="R", cons=10)

## Alternatively look at the example datafile DataExample,
## which  is available with the package as an example)
## ensure that class(DataExample) = c("odprism", "data.frame")
## then run: plot3d.prism(x=DataExample, variable="VS")
```

odprismU

Simulation function to asses the optimal design (unbalanced datasets) for studies planning to use random intercept and slopes models to analyze their data consiting of annually measured traits.

Description

In cases where traits are only measured or expressed once a year, the number of replicates per individual one can obtain is limited by their lifespan. odprismU is a simulation function that allows users to vary the sampling design in terms of population size being monitored and the number of years (occasions) this population is being sampled. The mean annual survival of individuals can be varied too (dead individuals are replaced), which together with the population size and sampling duration directly affects how many individuals are being sampled and how many replicates per individuals (i.e. unbalanced data). Finally, one can choose to expose the trait of interest Y to viability selection, to determine whether this will bias the outcome of parameter estimates obtianed using standard random regression models.

Usage

```
odprismU(pops, years, sims = 1000, fixed = c(0, 0), correl = TRUE,
  random, Survival, Xvar = c(1, 0, FALSE), alpha = 0,
  Quant = c(0.025, 0.25, 0.75, 0.975), ViabilitySelection = 0,
  Verbal = TRUE)
```

Arguments

`pops` Population size being sampled at each given occasion.

`years` The number of sampling occasions ('years') the population is being sampled

`sims` Number of different simulated dataset that are generated. Default: `sims=1000`

`fixed` Vector of parameter values b0 and bX, which are used to generate trait values (i.e. mean intercept and mean slope). Default `fixed=c(0,0)`.

`correl` If `correl=TRUE` then the third entry in the vector of argument random is taken to be CorIS (correlation intercept and slopes), if `FALSE` then it is taken to be CovIS (covariance intercept and slopes). Default: `correl=TRUE`.

`random` Vector of parameter values VI, VS, CorIS/CovIS and VR, which are used to generate trait values (residual varaince and (co)variance in intercepts and slopes). If no value is given for VR (e.g. when `random=c(0.2,0.1,0.5)`) it is assumed VR=1-VI.

`Survival` The mean annual survival of individuals, determining how many of the individuals measured at a given occasion can be sampled again at the next occasion. All dead individuals are replaced with new indviduals (without inheritance of trait values).

`Xvar` Vector describing properties used to generate X values. The first entry gives the value of VX, the second value the autocorrelation in X between subsequent samping occasions, if the third entry is `TRUE` all individuals sampled at a specific occasion are given the same X value, while if `FALSE` all individuals are given different values at a specific occasion. Default: `Xvar=c(1,0,FALSE)`.

`alpha` If `alpha!=0` then the statistical power is calculated, defined as the proportion of analysed simulated dataset that resulted in P-value < `alpha`. P-values of fixed effects (b0 & bX) were generated using Wald-tests, while P-value of random effects were generated using likelihood rate tests. Specifically, the P-value for the random slopes term was obtained by comparing the deviance of full model `Y~X+(X|Individual)` with the reduced model `Y~X+(1||Individual)`, while the P-value for the random intercept term was obtained by comparing the deviance of full model `Y~X+(1|Individual)` with the reduced model `Y~X`. Default: `alpha=0`.

`Quant` Each sampling design is simulated sims times. The argument `Quant` is used to describe which percentiles of the distribution of parameter estimates should be stored. Default: `Quant=c(0.025,0.25,0.75,0.975)`.

`ViabilitySelection`
Determines the strength of directional viability selection on trait Y, using the function P(survival)=1/(1+exp(-1(g0+ViabilitySelection*Y)). When `ViabilitySelection=0` there is no selection on Y. The parameter g0 is determined by the argument `Survival`, such that P(survival) at Y=0 equals the chosen values of `Survival`, see also the automatically generated plotted survival function.

`Verbal` If `TRUE`, the function prints to the screen which combination of indiv and repl it is currently evaluating. Default: `Verbal=TRUE`

Details

Please be aware that statistical inference in terms of P-values for fixed and especially for random parameters is a contentious topic. The methods used to calculate P-values here (Wald for fixed effects and LRT test for random effects) are widely used, but not necessarily the best. More generally, instead of focussing on statisical power I would advice to focus more on the acuracy and precision of parameter estimates. Therefore the default options (alpha=0) do not generate descriptions of statistical power.

Value

Dataframe of class 'odprism' that contains the chosen values used to generate the data as the distribution of estimates derived from the random regression models applied to these datasets. For example the column 'VR' gives the value used to generate residual noise, while the columns 'Q1VR', 'Q2VR', 'EstVR', 'Q3VR' and 'Q4VR' give respectively the 2.5, 25, 50 (i.e. median), 75 and 97.5 percentiles of the parameters estimates of VR in all simulated dataset. See the help file of DataExample for more details.

Author(s)

Martijn van de Pol

References

van de Pol, M. 2012 Quantifying individual reaction norms: How study design affects the accuracy, precision and power of random regression models. Methods in Ecology & Evolution, In press.

See Also

odprim, odprismU, plot2d.prism, plot3d.prism, DataExample

Examples

```
## Note example uses only few sims to speed things up, normally sims>1000
results<-odprismU(pops=c(10,25,50,75,100), years=c(2,4,6,8,10),
  random=c(0.2,0.1,0.5), Survival=0.7, sims=10, alpha=0.05)
results
plot2d.prism(x=results, variable="VS", xvar="G", cons=10)

## Alternatively look at an example datafile DataExample,
## which  is available with the package as an example.
## ensure that class(DataExample) = c("odprism", "data.frame")
## then run: plot3d.prism(x=DataExample, variable="VS")
```

`plot2d.prism` *Function to plot the performance (accuracy, precision and statistical power) as a function of the number of individuals (x-axis) and replicates (y-axis) sampled.*

Description

Function to plot the performance (accuracy, precision and statistical power) as a function of either the number of individuals or number of replicates sampled, while keeping the other constant. The function only plots dataframes of class 'odprism' that can be generated with the functions 'odprim', 'odpsrimB' and 'odprismU'. The function plots the values of the median (blue points) and the inner percentiles (green lines) and outer percentile (red lines) that are determined by the argument `Quant=c(0.025,0.25,0.75,0.975)`, while the value used to simulate the data is depicted by a horizontal straight black line. Power is calculated as the proportion of simulations in which the variable of interest was significant at the level given by `alpha` in the simulation function.

Usage

```
plot2d.prism(x, variable, xvar = "G", cons, ylim1 = c(0, 0))
```

Arguments

`x` dataframe of class 'odprism' that can be generated by the functions 'odprim', 'odprismB' and 'odprismU'.

`variable` Variable from the mixed model applied to the data, e.g. 'b0, bX, bX2, VI, VS, C (coding for either CorIS or CovIS) or VR.

`xvar` If `xvar = "G"` grouping units are plotted on the x-axis and the number of replicates per grouping units is kept constant at value of `cons`. If `xvar = "R"` number of replicates per grouping unit are plotted on the x-axis and the number of grouping units is kept constant at value of `cons`. Default: `xvar = "G"`.

`cons` Value of grouping units or replicates that is kept constant.

`ylim1` Describes the range over which values of parameter estimates are plotted on the y-axis. If no values are given than it will automatically use the entire observed range.

Details

If results were generated with the option `alpha=0` then the results for statistical power cannot be plotted.

Value

Plots one or two graphs. Left panel: the distribution of parameter estimates described by the percentile plots. Right panel: statistical power.

Author(s)

Martijn van de Pol

References

van de Pol, M. 2012 Quantifying individual reaction norms: How study design affects the accuracy, precision and power of random regression models. Methods in Ecology & Evolution, In press.

See Also

odprim, odprismU, odprismB, plot3d.prism, DataExample

Examples

```
## Note example uses only few sims to speed things up, normally sims>1000
results<-odprismB(indiv=c(10,25,50,75,100), repl=c(2,4,6,8,10),
  random=c(0.2,0.1,0.5), sims=10, alpha=0.05)
results
plot2d.prism(x=results, variable="VI", xvar="G", cons=10)

## Alternatively look at an example datafile DataExample,
## which is available with the package as an example.
## ensure that class(DataExample) = c("odprism", "data.frame")
## then run: plot2d.prism(x=DataExample, variable="VI", xvar="R", cons=100)
```

---

plot3d.prism *Function to plot the performance (accuracy, precision and statistical power) as a function of the number of individuals (x-axis) and replicates (y-axis) sampled.*

Description

Function to plot the performance (accuracy, precision and statistical power; different colors depicting different levels) as a function of the number of individuals (x-axis) and replicates (y-axis) sampled. The function only plots dataframes of class 'odprism' that can be generated with the functions 'odprim', 'odpsrimB' and 'odprismU'. Accuracy/bias is calculated as the median of all parameter estimates for a specific model variable minus the value of that variable used to generate the data. Imprecision is calculated as the difference between the upper and lower quantile (e.g. 75percentile-25percentile) of all parameter estimates from the simulated datasets. Power is calculated as the proportion of simulations in which the variable of interest was significant at the level given by alpha in the simulation function.

Usage

```
plot3d.prism(x, variable, prec = 2, absol = TRUE,
  zmax1 = 0, zmax2 = 0)
```

Arguments

`x` dataframe of class ’odprism’ that can be generated by the functions ’odprim’, ’odprismB’ and ’odprismU’.

`variable` Variable from the mixed model applied to the data, e.g. ’b0, bX, bX2, VI, VS, C (coding for either CorIS or CovIS) or VR.

`prec` Determines which quantiles are used to estimate (im)precision. For example if function ’odprismB’ was ran with `Quant=c(0.025,0.25,0.75,0.975)` then imprecision uses the inner quantiles 0.75 and 0.25 if `prec=1` and uses the outer quantiles 0.975 and 0.025 if `prec=2`.

`absol` If `absol=TRUE` the absolute value is taken of the accuracy/bias.

`zmax1` Determines the value beyond which values of accuracy/bias are truncated to improve readability of graphs. For example, if `zmax=1`, all values of bias>=1 are shown by the same colour.

`zmax2` Determines the value beyond which values of imprecision are truncated to improve readability of graphs. For example, if `zmax=1`, all values of imprecsion>=1 are shown by the same colour.

Details

If results were generated with the option `alpha=0` then the results for statistical power cannot be plotted.

Value

Plots two to three graphs. Left panel: bias-inaccuracy. Middle panel: Imprecsion. Right panel: statistical power.

Author(s)

Martijn van de Pol

References

van de Pol, M. 2012 Quantifying individual reaction norms: How study design affects the accuracy, precision and power of random regression models. Methods in Ecology & Evolution, In press.

See Also

`odprim`, `odprismU`, `odprismB`, `plot2d.prism`, `DataExample`

Examples

```
## Note example uses only few sims to speed things up, normally sims>1000
results<-odprismB(indiv=c(10,25,50,75,100), repl=c(2,4,6,8,10),
  random=c(0.2,0.1,0.5), sims=10, alpha=0.05)
results
plot3d.prism(x=results, variable="C")
```

```
## Alternatively look at an example datafile DataExample,
## which is available with the package as an example.
## ensure that class(DataExample) = c("odprism", "data.frame")
## then run plot3d.prism(x=DataExample, variable="C")
```

Index