# Package 'plsRbeta'

July 23, 2025

**Version** 0.3.0

**Date** 2023-03-12

**Depends** R (>= 3.5.0)

**Imports** mvtnorm, boot, Formula, MASS, plsRglm, betareg, methods

**Suggests** bipartite, knitr, markdown, plotrix, pls, plsdof, prettydoc, rmarkdown

**Title** Partial Least Squares Regression for Beta Regression Models

**Author** Frederic Bertrand [cre, aut] (ORCID: <https://orcid.org/0000-0002-0837-8281>), Myriam Maumy-Bertrand [aut] (ORCID: <https://orcid.org/0000-0002-4615-1512>)

**Maintainer** Frederic Bertrand <frederic.bertrand@utt.fr>

**Description** Provides Partial least squares Regression for (weighted) beta regression models (Bertrand 2013, <http://journal-sfds.fr/article/view/215>) and k-fold cross-validation of such models using various criteria. It allows for missing data in the explanatory variables. Bootstrap confidence intervals constructions are also available.

**License** GPL-3

**Encoding** UTF-8

**Classification/MSC** 62J12, 62J99

**LazyData** true

**VignetteBuilder** knitr

**RoxygenNote** 7.2.1

**URL** https://fbertran.github.io/plsRbeta/, https://github.com/fbertran/plsRbeta/

**BugReports** https://github.com/fbertran/plsRbeta/issues/

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2023-03-15 13:40:03 UTC

# Contents

plsRbeta-package *plsRbeta-package*

## Description

Provides Partial least squares Regression for (weighted) beta regression models (Bertrand 2013, <http://journal-sfds.fr/article/view/215>) and k-fold cross-validation of such models using various criteria. It allows for missing data in the explanatory variables. Bootstrap confidence intervals constructions are also available.

## References

Partial least squares Regression for (weighted) beta regression models (Bertrand 2013, <http://journal-sfds.fr/article/view/215>), https://github.com/fbertran/plsRbeta/ et https://fbertran.github.io/plsRbeta/

## Examples

```
data("GasolineYield",package="betareg")
modpls <- plsRbeta(yield~.,data=GasolineYield,nt=3,modele="pls-beta")
modpls$pp
modpls$Coeffs
modpls$Std.Coeffs
modpls$InfCrit
modpls$PredictY[1,]
rm("modpls")

data("GasolineYield",package="betareg")
yGasolineYield <- GasolineYield$yield
XGasolineYield <- GasolineYield[,2:5]
modpls <- plsRbeta(yGasolineYield,XGasolineYield,nt=3,modele="pls-beta")
modpls$pp
modpls$Coeffs
modpls$Std.Coeffs
modpls$InfCrit
modpls$PredictY[1,]
rm("modpls")
```

bootplsbeta *Non-parametric Bootstrap for PLS beta regression models*

## Description

Provides a wrapper for the bootstrap function `boot` from the `boot` R package.
Implements non-parametric bootstrap for PLS beta regression models by case resampling.

## Usage

```
bootplsbeta(
  object,
  typeboot = "plsmodel",
  R = 250,
  statistic = NULL,
  sim = "ordinary",
  stype = "i",
  stabvalue = 1e+06,
  ...
)
```

## Arguments

| | |
|---|---|
| object | An object of class plsRbetamodel to bootstrap |
| typeboot | The type of bootstrap. Either (Y,X) boostrap (typeboot="plsmodel") or (Y,T) bootstrap (typeboot="fmodel_np"). Defaults to (Y,T) resampling. |
| R | The number of bootstrap replicates. Usually this will be a single positive integer. For importance resampling, some resamples may use one set of weights and others use a different set of weights. In this case R would be a vector of integers where each component gives the number of resamples from each of the rows of weights. |
| statistic | A function which when applied to data returns a vector containing the statistic(s) of interest. statistic must take at least two arguments. The first argument passed will always be the original data. The second will be a vector of indices, frequencies or weights which define the bootstrap sample. Further, if predictions are required, then a third argument is required which would be a vector of the random indices used to generate the bootstrap predictions. Any further arguments can be passed to statistic through the ... argument. |
| sim | A character string indicating the type of simulation required. Possible values are "ordinary" (the default), "balanced", "permutation", or "antithetic". |
| stype | A character string indicating what the second argument of statistic represents. Possible values of stype are "i" (indices - the default), "f" (frequencies), or "w" (weights). |
| stabvalue | A value to hard threshold bootstrap estimates computed from atypical resamplings. |
| ... | Other named arguments for statistic which are passed unchanged each time it is called. Any such arguments to statistic should follow the arguments which statistic is required to have for the simulation. Beware of partial matching to arguments of boot listed above. |

## Details

More details on bootstrap techniques are available in the help of the boot function.

## Value

An object of class "boot". See the Value part of the help of the function boot.

## Author(s)

Frédéric Bertrand
<frederic.bertrand@utt.fr>
https://fbertran.github.io/homepage/

## References

Frédéric Bertrand, Nicolas Meyer, Michèle Beau-Faller, Karim El Bayed, Izzie-Jacques Namer, Myriam Maumy-Bertrand (2013). Régression Bêta PLS. *Journal de la Société Française de Statistique*, **154**(3):143-159. http://publications-sfds.math.cnrs.fr/index.php/J-SFdS/article/view/215

## See Also

boot

## Examples

```
data("GasolineYield",package="betareg")

# Std coefficients
modplsbeta <- plsRbeta(yield~., data=GasolineYield, nt=3, modele="pls-beta")
GazYield.boot <- bootplsbeta(modplsbeta, sim="ordinary", stype="i", R=250)
boot::boot.ci(GazYield.boot, conf = c(0.90,0.95),
type = c("norm","basic","perc","bca"), index=1)
boot::boot.ci(GazYield.boot, conf = c(0.90,0.95),
type = c("norm","basic","perc","bca"), index=2)
boot::boot.ci(GazYield.boot, conf = c(0.90,0.95),
type = c("norm","basic","perc","bca"), index=3)
boot::boot.ci(GazYield.boot, conf = c(0.90,0.95),
type = c("norm","basic","perc","bca"), index=4)
boot::boot.ci(GazYield.boot, conf = c(0.90,0.95),
type = c("norm","basic","perc","bca"), index=5)
boot::boot.ci(GazYield.boot, conf = c(0.90,0.95),
type = c("norm","basic","perc","bca"), index=6)

plsRglm::boxplots.bootpls(GazYield.boot)
plsRglm::confints.bootpls(GazYield.boot)
plsRglm::plots.confints.bootpls(plsRglm::confints.bootpls(GazYield.boot))

#Raw coefficients
modplsbeta <- plsRbeta(yield~.,data=GasolineYield,nt=3, modele="pls-beta")
GazYield.boot.raw <- bootplsbeta(modplsbeta, sim="ordinary", stype="i",
R=250, statistic=coefs.plsRbeta.raw)
boot::boot.ci(GazYield.boot.raw, conf = c(0.90,0.95),
type = c("norm","basic","perc","bca"), index=1)
boot::boot.ci(GazYield.boot.raw, conf = c(0.90,0.95),
```

```
type = c("norm","basic","perc","bca"), index=2)
boot::boot.ci(GazYield.boot.raw, conf = c(0.90,0.95),
type = c("norm","basic","perc","bca"), index=3)
boot::boot.ci(GazYield.boot.raw, conf = c(0.90,0.95),
type = c("norm","basic","perc","bca"), index=4)
boot::boot.ci(GazYield.boot.raw, conf = c(0.90,0.95),
type = c("norm","basic","perc","bca"), index=5)
boot::boot.ci(GazYield.boot.raw, conf = c(0.90,0.95),
type = c("norm","basic","perc","bca"), index=6)

plsRglm::boxplots.bootpls(GazYield.boot.raw)
plsRglm::confints.bootpls(GazYield.boot.raw)
plsRglm::plots.confints.bootpls(plsRglm::confints.bootpls(GazYield.boot.raw))


plot(GazYield.boot.raw,index=2)
boot::jack.after.boot(GazYield.boot.raw, index=2, useJ=TRUE, nt=3)
plot(GazYield.boot.raw, index=2,jack=TRUE)

# PLS bootstrap balanced

GazYield.boot.bal <- bootplsbeta(plsRbeta(yield~.,data=GasolineYield,nt=3,
modele="pls-beta"), sim="balanced", stype="i", R=250)
boot::boot.ci(GazYield.boot.bal, conf = c(0.90,0.95),
type = c("norm","basic","perc","bca"), index=1)
boot::boot.ci(GazYield.boot.bal, conf = c(0.90,0.95),
type = c("norm","basic","perc","bca"), index=2)
boot::boot.ci(GazYield.boot.bal, conf = c(0.90,0.95),
type = c("norm","basic","perc","bca"), index=3)
boot::boot.ci(GazYield.boot.bal, conf = c(0.90,0.95),
type = c("norm","basic","perc","bca"), index=4)
boot::boot.ci(GazYield.boot.bal, conf = c(0.90,0.95),
type = c("norm","basic","perc","bca"), index=5)
boot::boot.ci(GazYield.boot.bal, conf = c(0.90,0.95),
type = c("norm","basic","perc","bca"), index=6)


plsRglm::boxplots.bootpls(GazYield.boot.bal)
plsRglm::confints.bootpls(GazYield.boot.bal)
plsRglm::plots.confints.bootpls(plsRglm::confints.bootpls(GazYield.boot.bal))



plot(GazYield.boot.bal)
boot::jack.after.boot(GazYield.boot.bal, index=1, useJ=TRUE, nt=3)
plot(GazYield.boot.bal,jack=TRUE)


# PLS permutation bootstrap

GazYield.boot.perm <- bootplsbeta(plsRbeta(yield~.,data=GasolineYield,nt=3,
modele="pls-beta"), sim="permutation", stype="i", R=250)
boot::boot.ci(GazYield.boot.perm, conf = c(0.90,0.95),
```

```
type = c("norm","basic","perc"), index=1)
boot::boot.ci(GazYield.boot.perm, conf = c(0.90,0.95),
type = c("norm","basic","perc"), index=2)
boot::boot.ci(GazYield.boot.perm, conf = c(0.90,0.95),
type = c("norm","basic","perc"), index=3)
boot::boot.ci(GazYield.boot.perm, conf = c(0.90,0.95),
type = c("norm","basic","perc"), index=4)
boot::boot.ci(GazYield.boot.perm, conf = c(0.90,0.95),
type = c("norm","basic","perc"), index=5)
boot::boot.ci(GazYield.boot.perm, conf = c(0.90,0.95),
type = c("norm","basic","perc"), index=6)
plsRglm::boxplots.bootpls(GazYield.boot.perm)
plot(GazYield.boot.perm)
```

---

`coefs.plsRbeta` *Coefficients function for bootstrap techniques*

---

## Description

Returns the coefficients of a "plsRbeta" model.

## Usage

```
coefs.plsRbeta(
  dataset,
  ind,
  nt,
  modele,
  family = NULL,
  method = "logistic",
  link = NULL,
  link.phi = NULL,
  type = "ML",
  maxcoefvalues,
  ifbootfail,
  verbose = TRUE
)
```

## Arguments

| | |
|---|---|
| dataset | dataset to resample |
| ind | indices for resampling |
| nt | number of components to use |
| modele | name of the PLS glm or PLS beta model to be fitted ("pls", "pls-glm-Gamma", "pls-glm-gaussian", "pls-glm-inverse.gaussian", "pls-glm-logistic", "pls-glm-poisson", "pls-glm-polr", "pls-beta"). Use "modele=pls-glm-family" to enable the family option. |

| | |
|---|---|
| family | family to use if GLM model, see plsRbeta |
| method | method for beta regression |
| link | link for beta regression |
| link.phi | link.phi for beta regression |
| type | type of estimates |
| maxcoefvalues | maximum values allowed for the estimates of the coefficients to discard those coming from singular bootstrap samples |
| ifbootfail | value to return if the estimation fails on a bootstrap sample |
| verbose | should info messages be displayed ? |

## Value

Coefficients' Estimates on a sample.

## Author(s)

Frédéric Bertrand
<frederic.bertrand@utt.fr>
https://fbertran.github.io/homepage/

## References

Frédéric Bertrand, Nicolas Meyer, Michèle Beau-Faller, Karim El Bayed, Izzie-Jacques Namer, Myriam Maumy-Bertrand (2013). Régression Bêta PLS. *Journal de la Société Française de Statistique*, **154**(3):143-159. http://publications-sfds.math.cnrs.fr/index.php/J-SFdS/article/view/215

## See Also

See also bootplsbeta.

## Examples

```
data("GasolineYield",package="betareg")
bootplsbeta(plsRbeta(yield~.,data=GasolineYield,nt=3, modele="pls-beta"), typeboot="plsmodel",
R=250, statistic=coefs.plsRbeta)
```

`coefs.plsRbeta.raw` *Raw coefficients function for bootstrap techniques*

## Description

Returns the coefficients of a "plsRbeta" model.

## Usage

```
coefs.plsRbeta.raw(
  dataset,
  ind,
  nt,
  modele,
  family = NULL,
  method = "logistic",
  link = NULL,
  link.phi = NULL,
  type = "ML",
  maxcoefvalues,
  ifbootfail,
  verbose = TRUE
)
```

## Arguments

| | |
|---|---|
| dataset | dataset to resample |
| ind | indices for resampling |
| nt | number of components to use |
| modele | name of the PLS glm or PLS beta model to be fitted ("pls", "pls-glm-Gamma", "pls-glm-gaussian", "pls-glm-inverse.gaussian", "pls-glm-logistic", "pls-glm-poisson", "pls-glm-polr", "pls-beta"). Use "modele=pls-glm-family" to enable the family option. |
| family | family to use if GLM model, see plsRbeta |
| method | method for beta regression |
| link | link for beta regression |
| link.phi | link.phi for beta regression |
| type | type of estimates |
| maxcoefvalues | maximum values allowed for the estimates of the coefficients to discard those coming from singular bootstrap samples |
| ifbootfail | value to return if the estimation fails on a bootstrap sample |
| verbose | should info messages be displayed ? |

## Value

Coefficients' Estimates on a sample.

## Author(s)

Frédéric Bertrand
<frederic.bertrand@utt.fr>
https://fbertran.github.io/homepage/

## References

Frédéric Bertrand, Nicolas Meyer, Michèle Beau-Faller, Karim El Bayed, Izzie-Jacques Namer, Myriam Maumy-Bertrand (2013). Régression Bêta PLS. *Journal de la Société Française de Statistique*, **154**(3):143-159. http://publications-sfds.math.cnrs.fr/index.php/J-SFdS/article/view/215

## See Also

See also bootplsbeta.

## Examples

```
data("GasolineYield",package="betareg")
bootplsbeta(plsRbeta(yield~.,data=GasolineYield,nt=3, modele="pls-beta"), typeboot="fmodel_par",
R=250, statistic=coefs.plsRbeta.raw)
```

---

# coefs.plsRbetanp *Coefficients for bootstrap computations of PLSBeta models*

---

## Description

A function passed to boot to perform bootstrap.

## Usage

```
coefs.plsRbetanp(
  dataRepYtt,
  ind,
  nt,
  modele,
  family = NULL,
  method = "logistic",
  link = NULL,
  link.phi = NULL,
  type = "ML",
  verbose = TRUE,
```

```
  maxcoefvalues,
  wwetoile,
  ifbootfail
)
```

## Arguments

| | |
|---|---|
| dataRepYtt | components' coordinates to bootstrap |
| ind | indices for resampling |
| nt | number of components to use |
| modele | type of modele to use, see plsRbeta |
| family | glm family to use, see plsRbeta |
| method | method for beta regression |
| link | link for beta regression |
| link.phi | link.phi for beta regression |
| type | type of estimates |
| verbose | should info messages be displayed ? |
| maxcoefvalues | maximum values allowed for the estimates of the coefficients to discard those coming from singular bootstrap samples |
| wwetoile | values of the Wstar matrix in the original fit |
| ifbootfail | value to return if the estimation fails on a bootstrap sample |

## Value

estimates on a bootstrap sample or ifbootfail value if the bootstrap computation fails.

## Note

~~some notes~~

## Author(s)

Frédéric Bertrand
<frederic.bertrand@utt.fr>
https://fbertran.github.io/homepage/

## See Also

See also bootplsbeta

## Examples

```
data("GasolineYield",package="betareg")
bootplsbeta(plsRbeta(yield~.,data=GasolineYield,nt=3, modele="pls-beta"), typeboot="fmodel_np",
R=250, statistic=coefs.plsRbetanp)
```

`colon` *Tumor rate and spectral data*

## Description

This dataset feature tumor rate data with spectral data descriptors. It is used as an example in the second vignette of the package.

## Usage

```
colon
```

## Format

A data frame with 80 observations on the following 180 variables.

**X..Tumor.Cells** a numeric vector

**X4.69499969** a numeric vector

**X4.68499947** a numeric vector

**X4.67499971** a numeric vector

**X4.66499949** a numeric vector

**X4.65499973** a numeric vector

**X4.6449995** a numeric vector

**X4.63499975** a numeric vector

**X4.62499952** a numeric vector

**X4.61499977** a numeric vector

**X4.60499954** a numeric vector

**X4.59499979** a numeric vector

**X4.58499956** a numeric vector

**X4.57499981** a numeric vector

**X4.56499958** a numeric vector

**X4.55499983** a numeric vector

**X4.5449996** a numeric vector

**X4.53499985** a numeric vector

**X4.52499962** a numeric vector

**X4.51499987** a numeric vector

**X4.50499964** a numeric vector

**X4.49499989** a numeric vector

**X4.48499966** a numeric vector

**X4.4749999** a numeric vector

**X4.46499968** a numeric vector
**X4.45499992** a numeric vector
**X4.44499969** a numeric vector
**X4.43499947** a numeric vector
**X4.42499971** a numeric vector
**X4.41499949** a numeric vector
**X4.40499973** a numeric vector
**X4.3949995** a numeric vector
**X4.38499975** a numeric vector
**X4.37499952** a numeric vector
**X4.36499977** a numeric vector
**X4.35499954** a numeric vector
**X4.34499979** a numeric vector
**X4.33499956** a numeric vector
**X4.32499981** a numeric vector
**X4.31499958** a numeric vector
**X4.30499983** a numeric vector
**X4.2949996** a numeric vector
**X4.28499985** a numeric vector
**X4.27499962** a numeric vector
**X4.26499987** a numeric vector
**X4.25499964** a numeric vector
**X4.24499989** a numeric vector
**X4.23499966** a numeric vector
**X4.2249999** a numeric vector
**X4.21499968** a numeric vector
**X4.20499992** a numeric vector
**X4.19499969** a numeric vector
**X4.18499994** a numeric vector
**X4.17499971** a numeric vector
**X4.16499949** a numeric vector
**X4.15499973** a numeric vector
**X4.1449995** a numeric vector
**X4.13499975** a numeric vector
**X4.12499952** a numeric vector
**X4.11499977** a numeric vector
**X4.10499954** a numeric vector

**X4.09499979** a numeric vector

**X4.08499956** a numeric vector

**X4.07499981** a numeric vector

**X4.06499958** a numeric vector

**X4.05499983** a numeric vector

**X4.0449996** a numeric vector

**X4.03499985** a numeric vector

**X4.02499962** a numeric vector

**X4.01499987** a numeric vector

**X4.00499964** a numeric vector

**X3.99499965** a numeric vector

**X3.98499966** a numeric vector

**X3.97499967** a numeric vector

**X3.96499968** a numeric vector

**X3.95499969** a numeric vector

**X3.94499969** a numeric vector

**X3.9349997** a numeric vector

**X3.92499971** a numeric vector

**X3.91499972** a numeric vector

**X3.90499973** a numeric vector

**X3.89499974** a numeric vector

**X3.88499975** a numeric vector

**X3.87499976** a numeric vector

**X3.86499977** a numeric vector

**X3.85499978** a numeric vector

**X3.84499979** a numeric vector

**X3.8349998** a numeric vector

**X3.82499981** a numeric vector

**X3.81499982** a numeric vector

**X3.80499983** a numeric vector

**X3.7949996** a numeric vector

**X3.78499961** a numeric vector

**X3.77499962** a numeric vector

**X3.76499963** a numeric vector

**X3.75499964** a numeric vector

**X3.74499965** a numeric vector

**X3.73499966** a numeric vector

**X3.72499967** a numeric vector
**X3.71499968** a numeric vector
**X3.70499969** a numeric vector
**X3.69499969** a numeric vector
**X3.6849997** a numeric vector
**X3.67499971** a numeric vector
**X3.66499972** a numeric vector
**X3.65499973** a numeric vector
**X3.64499974** a numeric vector
**X3.63499975** a numeric vector
**X3.62499976** a numeric vector
**X3.61499977** a numeric vector
**X3.60499978** a numeric vector
**X3.59499979** a numeric vector
**X3.5849998** a numeric vector
**X3.57499981** a numeric vector
**X3.56499982** a numeric vector
**X3.55499983** a numeric vector
**X3.54499984** a numeric vector
**X3.53499961** a numeric vector
**X3.52499962** a numeric vector
**X3.51499963** a numeric vector
**X3.50499964** a numeric vector
**X3.49499965** a numeric vector
**X3.48499966** a numeric vector
**X3.47499967** a numeric vector
**X3.46499968** a numeric vector
**X3.45499969** a numeric vector
**X3.44499969** a numeric vector
**X3.4349997** a numeric vector
**X3.42499971** a numeric vector
**X3.41499972** a numeric vector
**X3.40499973** a numeric vector
**X3.39499974** a numeric vector
**X3.38499975** a numeric vector
**X3.37499976** a numeric vector
**X3.36499977** a numeric vector

**X3.35499978** a numeric vector

**X3.34499979** a numeric vector

**X3.3349998** a numeric vector

**X3.32499981** a numeric vector

**X3.31499982** a numeric vector

**X3.30499983** a numeric vector

**X3.29499984** a numeric vector

**X3.28499961** a numeric vector

**X3.27499962** a numeric vector

**X3.26499963** a numeric vector

**X3.25499964** a numeric vector

**X3.24499965** a numeric vector

**X3.23499966** a numeric vector

**X3.22499967** a numeric vector

**X3.21499968** a numeric vector

**X3.20499969** a numeric vector

**X3.19499969** a numeric vector

**X3.1849997** a numeric vector

**X3.17499971** a numeric vector

**X3.16499972** a numeric vector

**X3.15499973** a numeric vector

**X3.14499974** a numeric vector

**X3.13499975** a numeric vector

**X3.12499976** a numeric vector

**X3.11499977** a numeric vector

**X3.10499978** a numeric vector

**X3.09499979** a numeric vector

**X3.0849998** a numeric vector

**X3.07499981** a numeric vector

**X3.06499982** a numeric vector

**X3.05499983** a numeric vector

**X3.04499984** a numeric vector

**X3.03499985** a numeric vector

**X3.02499962** a numeric vector

**X3.01499963** a numeric vector

**X3.00499964** a numeric vector

**X2.99499965** a numeric vector

**X2.98499966** a numeric vector

**X2.97499967** a numeric vector

**X2.96499968** a numeric vector

**X2.95499969** a numeric vector

**X2.94499969** a numeric vector

**X2.9349997** a numeric vector

**X2.92499971** a numeric vector

**X2.91499972** a numeric vector

## References

Régression Bêta PLS. (French) [PLS Beta regression.], F. Bertrand, N. Meyer, M. Beau-Faller, K. El Bayed, N. Izzie-J., M. Maumy-Bertrand, (2013), J. SFdS, 154(3):143-159

Partial Least Squares Regression for Beta Regression Models. F. Bertrand, M. Maumy (2021). useR! 2021, Zurich.

## Examples

```
data(colon)
str(colon)
```

---

`ind_BCa_nt1BC` *ind_BCa_nt1BC*

---

## Description

Variable selection results for the vignette.

## Usage

```
ind_BCa_nt1BC
```

## Format

Logical vector of length 60.

## References

Régression Bêta PLS. (French) [PLS Beta regression.], F. Bertrand, N. Meyer, M. Beau-Faller, K. El Bayed, N. Izzie-J., M. Maumy-Bertrand, (2013), J. SFdS, 154(3):143-159

Partial Least Squares Regression for Beta Regression Models. F. Bertrand, M. Maumy (2021). useR! 2021, Zurich.

## Examples

```
data(ind_BCa_nt1BC)
## maybe str(ind_BCa_nt1BC) ; plot(ind_BCa_nt1BC) ...
```

---

`ind_BCa_nt1BR` *ind_BCa_nt1BR*

---

## Description

Variable selection results for the vignette.

## Usage

```
ind_BCa_nt1BR
```

## Format

Logical vector of length 60.

## References

Régression Bêta PLS. (French) [PLS Beta regression.], F. Bertrand, N. Meyer, M. Beau-Faller, K. El Bayed, N. Izzie-J., M. Maumy-Bertrand, (2013), J. SFdS, 154(3):143-159

Partial Least Squares Regression for Beta Regression Models. F. Bertrand, M. Maumy (2021). useR! 2021, Zurich.

## Examples

```
data(ind_BCa_nt1BR)
## maybe str(ind_BCa_nt1BR) ; plot(ind_BCa_nt1BR) ...
```

---

`ind_BCa_nt2BC` *ind_BCa_nt2BC*

---

## Description

Variable selection results for the vignette.

## Usage

```
ind_BCa_nt2BC
```

## Format

Logical vector of length 60.

## References

Régression Bêta PLS. (French) [PLS Beta regression.], F. Bertrand, N. Meyer, M. Beau-Faller, K. El Bayed, N. Izzie-J., M. Maumy-Bertrand, (2013), J. SFdS, 154(3):143-159

Partial Least Squares Regression for Beta Regression Models. F. Bertrand, M. Maumy (2021). useR! 2021, Zurich.

## Examples

```
data(ind_BCa_nt2BC)
## maybe str(ind_BCa_nt2BC) ; plot(ind_BCa_nt2BC) ...
```

---

`ind_BCa_nt2BR` *ind_BCa_nt2BR*

---

## Description

Variable selection results for the vignette.

## Usage

```
ind_BCa_nt2BR
```

## Format

Logical vector of length 60.

## References

Régression Bêta PLS. (French) [PLS Beta regression.], F. Bertrand, N. Meyer, M. Beau-Faller, K. El Bayed, N. Izzie-J., M. Maumy-Bertrand, (2013), J. SFdS, 154(3):143-159

Partial Least Squares Regression for Beta Regression Models. F. Bertrand, M. Maumy (2021). useR! 2021, Zurich.

## Examples

```
data(ind_BCa_nt2BR)
## maybe str(ind_BCa_nt2BR) ; plot(ind_BCa_nt2BR) ...
```

## ind_BCa_nt3 *ind_BCa_nt3*

### Description

Variable selection results for the vignette.

### Usage

```
ind_BCa_nt3
```

### Format

Logical vector of length 60.

### References

Régression Bêta PLS. (French) [PLS Beta regression.], F. Bertrand, N. Meyer, M. Beau-Faller, K. El Bayed, N. Izzie-J., M. Maumy-Bertrand, (2013), J. SFdS, 154(3):143-159

Partial Least Squares Regression for Beta Regression Models. F. Bertrand, M. Maumy (2021). useR! 2021, Zurich.

### Examples

```
data(ind_BCa_nt3)
## maybe str(ind_BCa_nt3) ; plot(ind_BCa_nt3) ...
```

## ind_BCa_nt3BC *ind_BCa_nt3BC*

### Description

Variable selection results for the vignette.

### Usage

```
ind_BCa_nt3BC
```

### Format

Logical vector of length 60.

## References

Régression Bêta PLS. (French) [PLS Beta regression.], F. Bertrand, N. Meyer, M. Beau-Faller, K. El Bayed, N. Izzie-J., M. Maumy-Bertrand, (2013), J. SFdS, 154(3):143-159

Partial Least Squares Regression for Beta Regression Models. F. Bertrand, M. Maumy (2021). useR! 2021, Zurich.

## Examples

```
data(ind_BCa_nt3BC)
## maybe str(ind_BCa_nt3BC) ; plot(ind_BCa_nt3BC) ...
```

---

`ind_BCa_nt3BR` *ind_BCa_nt3BR*

---

## Description

Variable selection results for the vignette.

## Usage

```
ind_BCa_nt3BR
```

## Format

Logical vector of length 60.

## References

Régression Bêta PLS. (French) [PLS Beta regression.], F. Bertrand, N. Meyer, M. Beau-Faller, K. El Bayed, N. Izzie-J., M. Maumy-Bertrand, (2013), J. SFdS, 154(3):143-159

Partial Least Squares Regression for Beta Regression Models. F. Bertrand, M. Maumy (2021). useR! 2021, Zurich.

## Examples

```
data(ind_BCa_nt3BR)
## maybe str(ind_BCa_nt3BR) ; plot(ind_BCa_nt3BR) ...
```

## ind_BCa_nt4BC *ind_BCa_nt4BC*

### Description

Variable selection results for the vignette.

### Usage

```
ind_BCa_nt4BC
```

### Format

Logical vector of length 60.

### References

Régression Bêta PLS. (French) [PLS Beta regression.], F. Bertrand, N. Meyer, M. Beau-Faller, K. El Bayed, N. Izzie-J., M. Maumy-Bertrand, (2013), J. SFdS, 154(3):143-159

Partial Least Squares Regression for Beta Regression Models. F. Bertrand, M. Maumy (2021). useR! 2021, Zurich.

### Examples

```
data(ind_BCa_nt4BC)
## maybe str(ind_BCa_nt4BC) ; plot(ind_BCa_nt4BC) ...
```

## ind_BCa_nt4BR *ind_BCa_nt4BR*

### Description

Variable selection results for the vignette.

### Usage

```
ind_BCa_nt4BR
```

### Format

Logical vector of length 60.

## References

Régression Bêta PLS. (French) [PLS Beta regression.], F. Bertrand, N. Meyer, M. Beau-Faller, K. El Bayed, N. Izzie-J., M. Maumy-Bertrand, (2013), J. SFdS, 154(3):143-159

Partial Least Squares Regression for Beta Regression Models. F. Bertrand, M. Maumy (2021). useR! 2021, Zurich.

## Examples

```
data(ind_BCa_nt4BR)
## maybe str(ind_BCa_nt4BR) ; plot(ind_BCa_nt4BR) ...
```

---

`ind_BCa_nt5BC` *ind_BCa_nt5BC*

---

## Description

Variable selection results for the vignette.

## Usage

```
ind_BCa_nt5BC
```

## Format

Logical vector of length 60.

## References

Régression Bêta PLS. (French) [PLS Beta regression.], F. Bertrand, N. Meyer, M. Beau-Faller, K. El Bayed, N. Izzie-J., M. Maumy-Bertrand, (2013), J. SFdS, 154(3):143-159

Partial Least Squares Regression for Beta Regression Models. F. Bertrand, M. Maumy (2021). useR! 2021, Zurich.

## Examples

```
data(ind_BCa_nt5BC)
## maybe str(ind_BCa_nt5BC) ; plot(ind_BCa_nt5BC) ...
```

ind_BCa_nt5BR *ind_BCa_nt5BR*

## Description

Variable selection results for the vignette.

## Usage

```
ind_BCa_nt5BR
```

## Format

Logical vector of length 60.

## References

Régression Bêta PLS. (French) [PLS Beta regression.], F. Bertrand, N. Meyer, M. Beau-Faller, K. El Bayed, N. Izzie-J., M. Maumy-Bertrand, (2013), J. SFdS, 154(3):143-159

Partial Least Squares Regression for Beta Regression Models. F. Bertrand, M. Maumy (2021). useR! 2021, Zurich.

## Examples

```
data(ind_BCa_nt5BR)
## maybe str(ind_BCa_nt5BR) ; plot(ind_BCa_nt5BR) ...
```

ind_BCa_nt6BC *ind_BCa_nt6BC*

## Description

Variable selection results for the vignette.

## Usage

```
ind_BCa_nt6BC
```

## Format

Logical vector of length 60.

## References

Régression Bêta PLS. (French) [PLS Beta regression.], F. Bertrand, N. Meyer, M. Beau-Faller, K. El Bayed, N. Izzie-J., M. Maumy-Bertrand, (2013), J. SFdS, 154(3):143-159

Partial Least Squares Regression for Beta Regression Models. F. Bertrand, M. Maumy (2021). useR! 2021, Zurich.

## Examples

```
data(ind_BCa_nt6BC)
## maybe str(ind_BCa_nt6BC) ; plot(ind_BCa_nt6BC) ...
```

---

`ind_BCa_nt6BR` *ind_BCa_nt6BR*

---

## Description

Variable selection results for the vignette.

## Usage

```
ind_BCa_nt6BR
```

## Format

Logical vector of length 60.

## References

Régression Bêta PLS. (French) [PLS Beta regression.], F. Bertrand, N. Meyer, M. Beau-Faller, K. El Bayed, N. Izzie-J., M. Maumy-Bertrand, (2013), J. SFdS, 154(3):143-159

Partial Least Squares Regression for Beta Regression Models. F. Bertrand, M. Maumy (2021). useR! 2021, Zurich.

## Examples

```
data(ind_BCa_nt6BR)
## maybe str(ind_BCa_nt6BR) ; plot(ind_BCa_nt6BR) ...
```

`kfolds2Chisq` *Computes Predicted Chisquare for kfold cross validated partial least squares beta regression models.*

## Description

This function computes Predicted Chisquare for kfold cross validated partial least squares beta regression models.

## Usage

```
kfolds2Chisq(pls_kfolds)
```

## Arguments

| | |
|---|---|
| `pls_kfolds` | a kfold cross validated partial least squares regression glm model |

## Value

| | |
|---|---|
| `list` | Total Predicted Chisquare vs number of components for the first group partition |
| `list()` | ... |
| `list` | Total Predicted Chisquare vs number of components for the last group partition |

## Note

Use `PLS_beta_kfoldcv` to create kfold cross validated partial least squares regression glm and beta models.

## Author(s)

Frédéric Bertrand
<frederic.bertrand@utt.fr>
https://fbertran.github.io/homepage/

## References

Frédéric Bertrand, Nicolas Meyer, Michèle Beau-Faller, Karim El Bayed, Izzie-Jacques Namer, Myriam Maumy-Bertrand (2013). Régression Bêta PLS. *Journal de la Société Française de Statistique*, **154**(3):143-159. http://publications-sfds.math.cnrs.fr/index.php/J-SFdS/article/view/215

## See Also

`kfolds2coeff`, `kfolds2Press`, `kfolds2Pressind`, `kfolds2Chisqind`, `kfolds2Mclassedind` and `kfolds2Mclassed` to extract and transforms results from kfold cross validation.

## Examples

```
## Not run:
data("GasolineYield",package="betareg")
yGasolineYield <- GasolineYield$yield
XGasolineYield <- GasolineYield[,2:5]
bbb <- PLS_beta_kfoldcv(yGasolineYield,XGasolineYield,nt=3,modele="pls-beta")
kfolds2Chisq(bbb)

## End(Not run)
```

---

`kfolds2Chisqind` *Computes individual Predicted Chisquare for kfold cross validated partial least squares beta regression models.*

---

## Description

This function computes individual Predicted Chisquare for kfold cross validated partial least squares beta regression models.

## Usage

```
kfolds2Chisqind(pls_kfolds)
```

## Arguments

| | |
|---|---|
| `pls_kfolds` | a kfold cross validated partial least squares regression glm model |

## Value

| | |
|---|---|
| `list` | Individual PChisq vs number of components for the first group partition |
| `list()` | ... |
| `list` | Individual PChisq vs number of components for the last group partition |

## Note

Use `PLS_beta_kfoldcv` to create kfold cross validated partial least squares regression glm models.

## Author(s)

Frédéric Bertrand
<frederic.bertrand@utt.fr>
https://fbertran.github.io/homepage/

## References

Frédéric Bertrand, Nicolas Meyer, Michèle Beau-Faller, Karim El Bayed, Izzie-Jacques Namer, Myriam Maumy-Bertrand (2013). Régression Bêta PLS. *Journal de la Société Française de Statistique*, **154**(3):143-159. http://publications-sfds.math.cnrs.fr/index.php/J-SFdS/article/view/215

## See Also

kfolds2coeff, kfolds2Press, kfolds2Pressind, kfolds2Chisq, kfolds2Mclassedind and kfolds2Mclassed to extract and transforms results from kfold cross validation.

## Examples

```
## Not run:
data("GasolineYield",package="betareg")
yGasolineYield <- GasolineYield$yield
XGasolineYield <- GasolineYield[,2:5]
bbb <- PLS_beta_kfoldcv(yGasolineYield,XGasolineYield,nt=3,modele="pls-beta")
kfolds2Chisqind(bbb)

## End(Not run)
```

---

# kfolds2CVinfos_beta *Extracts and computes information criteria and fits statistics for kfold cross validated partial least squares beta regression models*

---

## Description

This function extracts and computes information criteria and fits statistics for kfold cross validated partial least squares beta regression models for both formula or classic specifications of the model.

## Usage

```
kfolds2CVinfos_beta(pls_kfolds, MClassed = FALSE)
```

## Arguments

| | |
|---|---|
| pls_kfolds | an object computed using PLS_beta_kfoldcv |
| MClassed | should number of miss classed be computed |

## Details

The Mclassed option should only set to TRUE if the response is binary.

## Value

| | |
|---|---|
| list | table of fit statistics for first group partition |
| list() | ... |
| list | table of fit statistics for last group partition |

## Author(s)

Frédéric Bertrand
<frederic.bertrand@utt.fr>
https://fbertran.github.io/homepage/

## References

Frédéric Bertrand, Nicolas Meyer, Michèle Beau-Faller, Karim El Bayed, Izzie-Jacques Namer, Myriam Maumy-Bertrand (2013). Régression Bêta PLS. *Journal de la Société Française de Statistique*, **154**(3):143-159. http://publications-sfds.math.cnrs.fr/index.php/J-SFdS/article/view/215

## See Also

kfolds2coeff, kfolds2Pressind, kfolds2Press, kfolds2Mclassedind and kfolds2Mclassed to extract and transforms results from kfold cross validation.

## Examples

```
## Not run:
data("GasolineYield",package="betareg")
bbb <- PLS_beta_kfoldcv_formula(yield~.,data=GasolineYield,nt=3,modele="pls-beta")
kfolds2CVinfos_beta(bbb)

## End(Not run)
```

---

# modpls.boot3 *Bootstrap distribution of a 3 components model*

---

## Description

A precomputed bootstrap distribution of the coefficients of a model used in the vignette.

## Usage

```
modpls.boot3
```

## Format

a class boot object

## References

Régression Bêta PLS. (French) [PLS Beta regression.], F. Bertrand, N. Meyer, M. Beau-Faller, K. El Bayed, N. Izzie-J., M. Maumy-Bertrand, (2013), J. SFdS, 154(3):143-159

Partial Least Squares Regression for Beta Regression Models. F. Bertrand, M. Maumy (2021). useR! 2021, Zurich.

## Examples

```
data(modpls.boot3)
str(modpls.boot3)
plot(modpls.boot3)
```

---

`modpls_sub4` *A plsRbetamodel model on a data subset*

---

## Description

A precomputed four components plsRbetamodel fitted to a subset of an example dataset and used in the vignette.

## Usage

```
modpls_sub4
```

## Format

a class plsRbetamodel object

## References

Régression Bêta PLS. (French) [PLS Beta regression.], F. Bertrand, N. Meyer, M. Beau-Faller, K. El Bayed, N. Izzie-J., M. Maumy-Bertrand, (2013), J. SFdS, 154(3):143-159

Partial Least Squares Regression for Beta Regression Models. F. Bertrand, M. Maumy (2021). useR! 2021, Zurich.

## Examples

```
data(modpls_sub4)
str(modpls_sub4)
```

permcoefs.plsRbeta *Coefficients function for permutation bootstrap techniques*

## Description

A function passed to boot to perform bootstrap.

## Usage

```
permcoefs.plsRbeta(
  dataset,
  ind,
  nt,
  modele,
  family = NULL,
  method = "logistic",
  link = "logit",
  link.phi = NULL,
  type = "ML",
  maxcoefvalues,
  ifbootfail,
  verbose = TRUE
)
```

## Arguments

| | |
|---|---|
| dataset | dataset to resample |
| ind | indices for resampling |
| nt | number of components to use |
| modele | name of the PLS glm or PLS beta model to be fitted ("pls", "pls-glm-Gamma", "pls-glm-gaussian", "pls-glm-inverse.gaussian", "pls-glm-logistic", "pls-glm-poisson", "pls-glm-polr", "pls-beta"). Use "modele=pls-glm-family" to enable the family option. |
| family | family to use if GLM model, see plsRbeta |
| method | method for beta regression |
| link | link for beta regression |
| link.phi | link.phi for beta regression |
| type | type of estimates |
| maxcoefvalues | maximum values allowed for the estimates of the coefficients to discard those coming from singular bootstrap samples |
| ifbootfail | value to return if the estimation fails on a |
| verbose | should info messages be displayed ? |

## Value

Estimates on a bootstrap sample.

## Author(s)

Frédéric Bertrand
<frederic.bertrand@utt.fr>
https://fbertran.github.io/homepage/

## References

Frédéric Bertrand, Nicolas Meyer, Michèle Beau-Faller, Karim El Bayed, Izzie-Jacques Namer, Myriam Maumy-Bertrand (2013). Régression Bêta PLS. *Journal de la Société Française de Statistique*, **154**(3):143-159. http://publications-sfds.math.cnrs.fr/index.php/J-SFdS/article/view/215

## See Also

See also bootplsbeta.

## Examples

```
data("GasolineYield",package="betareg")

GazYield.boot <- bootplsbeta(plsRbeta(yield~.,data=GasolineYield,nt=3,
modele="pls-beta", verbose=FALSE), sim="ordinary", stype="i", R=250, statistic=permcoefs.plsRbeta)
```

---

# permcoefs.plsRbeta.raw

*Raw coefficients function for permutation bootstrap techniques*

---

## Description

A function passed to boot to perform bootstrap.

## Usage

```
permcoefs.plsRbeta.raw(
  dataset,
  ind,
  nt,
  modele,
  family = NULL,
  method = "logistic",
  link = "logit",
  link.phi = NULL,
```

```
  type = "ML",
  maxcoefvalues,
  ifbootfail,
  verbose = TRUE
)
```

## Arguments

| | |
|---|---|
| dataset | dataset to resample |
| ind | indices for resampling |
| nt | number of components to use |
| modele | name of the PLS glm or PLS beta model to be fitted ("pls", "pls-glm-Gamma", "pls-glm-gaussian", "pls-glm-inverse.gaussian", "pls-glm-logistic", "pls-glm-poisson", "pls-glm-polr", "pls-beta"). Use "modele=pls-glm-family" to enable the family option. |
| family | family to use if GLM model, see plsRbeta |
| method | method for beta regression |
| link | link for beta regression |
| link.phi | link.phi for beta regression |
| type | type of estimates |
| maxcoefvalues | maximum values allowed for the estimates of the coefficients to discard those coming from singular bootstrap samples |
| ifbootfail | value to return if the estimation fails on a |
| verbose | should info messages be displayed ? |

## Value

Estimates on a bootstrap sample.

## Author(s)

Frédéric Bertrand
<frederic.bertrand@utt.fr>
https://fbertran.github.io/homepage/

## References

Frédéric Bertrand, Nicolas Meyer, Michèle Beau-Faller, Karim El Bayed, Izzie-Jacques Namer, Myriam Maumy-Bertrand (2013). Régression Bêta PLS. *Journal de la Société Française de Statistique*, **154**(3):143-159. http://publications-sfds.math.cnrs.fr/index.php/J-SFdS/article/view/215

## See Also

See also bootplsbeta.

## Examples

```
data("GasolineYield",package="betareg")
modplsbeta <- plsRbeta(yield~.,data=GasolineYield,nt=3, modele="pls-beta")
GazYield.boot.raw <- bootplsbeta(modplsbeta, sim="permutation", stype="i",
R=250, statistic=coefs.plsRbeta.raw)
```

---

`permcoefs.plsRbetanp` *Coefficients for permutation bootstrap computations of PLSBeta models*

---

## Description

A function passed to `boot` to perform bootstrap.

## Usage

```
permcoefs.plsRbetanp(
  dataRepYtt,
  ind,
  nt,
  modele,
  family = NULL,
  maxcoefvalues,
  wwetoile,
  ifbootfail
)
```

## Arguments

| | |
|---|---|
| `dataRepYtt` | components' coordinates to bootstrap |
| `ind` | indices for resampling |
| `nt` | number of components to use |
| `modele` | type of modele to use, see plsRbeta |
| `family` | glm family to use, see plsRbeta |
| `maxcoefvalues` | maximum values allowed for the estimates of the coefficients to discard those coming from singular bootstrap samples |
| `wwetoile` | values of the Wstar matrix in the original fit |
| `ifbootfail` | value to return if the estimation fails on a bootstrap sample |

## Value

estimates on a bootstrap sample or `ifbootfail` value if the bootstrap computation fails.

## Note

~~some notes~~

## Author(s)

Frédéric Bertrand
<frederic.bertrand@utt.fr>
https://fbertran.github.io/homepage/

## See Also

See also bootplsbeta

## Examples

```
data("GasolineYield",package="betareg")
modplsbeta <- plsRbeta(yield~.,data=GasolineYield,nt=3, modele="pls-beta")
bootplsbeta(modplsbeta, R=250, statistic=permcoefs.plsRbetanp, typeboot="fmodel_np")
```

---

# plsRbeta *Partial least squares Regression beta regression models*

---

## Description

This function implements Partial least squares Regression generalized linear models complete or incomplete datasets.

## Usage

```
plsRbeta(object, ...)
## Default S3 method:
plsRbetamodel(object,dataX,nt=2,limQ2set=.0975,
dataPredictY=dataX,modele="pls",family=NULL,typeVC="none",EstimXNA=FALSE,
scaleX=TRUE,scaleY=NULL,pvals.expli=FALSE,alpha.pvals.expli=.05,
MClassed=FALSE,tol_Xi=10^(-12),weights,method,sparse=FALSE,sparseStop=TRUE,
naive=FALSE,link=NULL,link.phi=NULL,type="ML",verbose=TRUE, ...)
## S3 method for class 'formula'
plsRbetamodel(object,data=NULL,nt=2,limQ2set=.0975,
dataPredictY,modele="pls",family=NULL,typeVC="none",EstimXNA=FALSE,
scaleX=TRUE,scaleY=NULL,pvals.expli=FALSE,alpha.pvals.expli=.05,
MClassed=FALSE,tol_Xi=10^(-12),weights,subset,start=NULL,etastart,
mustart,offset,method="glm.fit",control= list(),contrasts=NULL,
sparse=FALSE,sparseStop=TRUE,naive=FALSE,link=NULL,link.phi=NULL,type="ML",
verbose=TRUE, ...)
```

## Arguments

| | |
|---|---|
| object | a response (training) dataset or an object of class "formula" (or one that can be coerced to that class): a symbolic description of the model to be fitted. The details of model specification are given under 'Details'. |
| dataX | predictor(s) (training) dataset |
| data | an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the environment from which plsRbeta is called. |
| nt | number of components to be extracted |
| limQ2set | limit value for the Q2 |
| dataPredictY | predictor(s) (testing) dataset |
| modele | name of the PLS glm or PLS beta model to be fitted ("pls", "pls-glm-Gamma", "pls-glm-gaussian", "pls-glm-inverse.gaussian", "pls-glm-logistic", "pls-glm-poisson", "pls-glm-polr", "pls-beta"). Use "modele=pls-glm-family" to enable the family option. |
| family | a description of the error distribution and link function to be used in the model. This can be a character string naming a family function, a family function or the result of a call to a family function. (See family for details of family functions.) To use the family option, please set modele="pls-glm-family". User defined families can also be defined. See details. |
| typeVC | type of leave one out cross validation. For back compatibility purpose.<br>none no cross validation<br>standard no cross validation<br>missingdata no cross validation<br>adaptative no cross validation |
| EstimXNA | only for modele="pls". Set whether the missing X values have to be estimated. |
| scaleX | scale the predictor(s) : must be set to TRUE for modele="pls" and should be for glms pls. |
| scaleY | scale the response : Yes/No. Ignored since non always possible for glm responses. |
| pvals.expli | should individual p-values be reported to tune model selection ? |
| alpha.pvals.expli | level of significance for predictors when pvals.expli=TRUE |
| MClassed | number of missclassified cases, should only be used for binary responses |
| tol_Xi | minimal value for Norm2(Xi) and $\det(pp' \times pp)$ if there is any missing value in the dataX. It defaults to $10^{-12}$ |
| weights | an optional vector of 'prior weights' to be used in the fitting process. Should be NULL or a numeric vector. |
| subset | an optional vector specifying a subset of observations to be used in the fitting process. |
| start | starting values for the parameters in the linear predictor. |

| | |
|---|---|
| etastart | starting values for the linear predictor. |
| mustart | starting values for the vector of means. |
| offset | this can be used to specify an *a priori* known component to be included in the linear predictor during fitting. This should be NULL or a numeric vector of length equal to the number of cases. One or more offset terms can be included in the formula instead or as well, and if more than one is specified their sum is used. See model.offset. |
| method | the method to be used in fitting the model. The default method "glm.fit" uses iteratively reweighted least squares (IWLS). User-supplied fitting functions can be supplied either as a function or a character string naming a function, with a function which takes the same arguments as glm.fit. |
| control | a list of parameters for controlling the fitting process. For glm.fit this is passed to glm.control. |
| contrasts | an optional list. See the contrasts.arg of model.matrix.default. |
| sparse | should the coefficients of non-significant predictors (<alpha.pvals.expli) be set to 0 |
| sparseStop | should component extraction stop when no significant predictors (<alpha.pvals.expli) are found |
| naive | Use the naive estimates for the Degrees of Freedom in plsR? Default is FALSE. |
| link | character specification of the link function in the mean model (mu). Currently, "logit", "probit", "cloglog", "cauchit", "log", "loglog" are supported. Alternatively, an object of class "link-glm" can be supplied. |
| link.phi | character specification of the link function in the precision model (phi). Currently, "identity", "log", "sqrt" are supported. The default is "log" unless formula is of type y~x where the default is "identity" (for backward compatibility). Alternatively, an object of class "link-glm" can be supplied. |
| type | character specification of the type of estimator. Currently, maximum likelihood ("ML"), ML with bias correction ("BC"), and ML with bias reduction ("BR") are supported. |
| verbose | should info messages be displayed ? |
| ... | arguments to pass to plsRmodel.default or to plsRmodel.formula |

## Details

There are seven different predefined models with predefined link functions available :

"pls" ordinary pls models

"pls-glm-Gamma" glm gaussian with inverse link pls models

"pls-glm-gaussian" glm gaussian with identity link pls models

"pls-glm-inverse-gamma" glm binomial with square inverse link pls models

"pls-glm-logistic" glm binomial with logit link pls models

"pls-glm-poisson" glm poisson with log link pls models

"pls-glm-polr" glm polr with logit link pls models

Using the "family=" option and setting "modele=pls-glm-family" allows changing the family and link function the same way as for the glm function. As a consequence user-specified families can also be used.

**The** gaussian **family** accepts the links (as names) identity, log and inverse.

**The** binomial **family** accepts the links logit, probit, cauchit, (corresponding to logistic, normal and Cauchy CDFs respectively) log and cloglog (complementary log-log).

**The** Gamma **family** accepts the links inverse, identity and log.

**The** poisson **family** accepts the links log, identity, and sqrt.

**The** inverse.gaussian **family** accepts the links 1/mu^2, inverse, identity and log.

**The** quasi **family** accepts the links logit, probit, cloglog, identity, inverse, log, 1/mu^2 and sqrt.

**The function** power can be used to create a power link function.

A typical predictor has the form response ~ terms where response is the (numeric) response vector and terms is a series of terms which specifies a linear predictor for response. A terms specification of the form first + second indicates all the terms in first together with all the terms in second with any duplicates removed.

A specification of the form first:second indicates the the set of terms obtained by taking the interactions of all terms in first with all terms in second. The specification first*second indicates the cross of first and second. This is the same as first + second + first:second.

The terms in the formula will be re-ordered so that main effects come first, followed by the interactions, all second-order, all third-order and so on: to avoid this pass a terms object as the formula.

Non-NULL weights can be used to indicate that different observations have different dispersions (with the values in weights being inversely proportional to the dispersions); or equivalently, when the elements of weights are positive integers w_i, that each response y_i is the mean of w_i unit-weight observations.

The default estimator for Degrees of Freedom is the Kramer and Sugiyama's one which only works for classical plsR models. For these models, Information criteria are computed accordingly to these estimations. Naive Degrees of Freedom and Information Criteria are also provided for comparison purposes. For more details, see Kraemer, N., Sugiyama M. (2010). "The Degrees of Freedom of Partial Least Squares Regression". preprint, http://arxiv.org/abs/1002.4112.

## Value

Depends on the model that was used to fit the model.

## Note

Use plsRbeta instead.

## Author(s)

Frédéric Bertrand
<frederic.bertrand@utt.fr>
https://fbertran.github.io/homepage/

## References

Frédéric Bertrand, Nicolas Meyer, Michèle Beau-Faller, Karim El Bayed, Izzie-Jacques Namer, Myriam Maumy-Bertrand (2013). Régression Bêta PLS. *Journal de la Société Française de Statistique*, **154**(3):143-159. http://publications-sfds.math.cnrs.fr/index.php/J-SFdS/article/view/215

## See Also

plsR and plsRglm

## Examples

```
data("GasolineYield",package="betareg")
modpls <- plsRbeta(yield~.,data=GasolineYield,nt=3,modele="pls-beta")
modpls$pp
modpls$Coeffs
modpls$Std.Coeffs
modpls$InfCrit
modpls$PredictY[1,]
rm("modpls")

data("GasolineYield",package="betareg")
yGasolineYield <- GasolineYield$yield
XGasolineYield <- GasolineYield[,2:5]
modpls <- plsRbeta(yGasolineYield,XGasolineYield,nt=3,modele="pls-beta")
modpls$pp
modpls$Coeffs
modpls$Std.Coeffs
modpls$InfCrit
modpls$PredictY[1,]
rm("modpls")
```

---

# PLS_beta *Partial least squares beta regression models*

---

## Description

This function implements Partial least squares beta regression models on complete or incomplete datasets.

## Usage

```
PLS_beta(
  dataY,
  dataX,
  nt = 2,
  limQ2set = 0.0975,
```

```
  dataPredictY = dataX,
  modele = "pls",
  family = NULL,
  typeVC = "none",
  EstimXNA = FALSE,
  scaleX = TRUE,
  scaleY = NULL,
  pvals.expli = FALSE,
  alpha.pvals.expli = 0.05,
  MClassed = FALSE,
  tol_Xi = 10^(-12),
  weights,
  method,
  sparse = FALSE,
  sparseStop = TRUE,
  naive = FALSE,
  link = NULL,
  link.phi = NULL,
  type = "ML",
  verbose = TRUE
)
```

## Arguments

| | |
|---|---|
| dataY | response (training) dataset |
| dataX | predictor(s) (training) dataset |
| nt | number of components to be extracted |
| limQ2set | limit value for the Q2 |
| dataPredictY | predictor(s) (testing) dataset |
| modele | name of the PLS glm or PLS beta model to be fitted ("pls", "pls-glm-Gamma", "pls-glm-gaussian", "pls-glm-inverse.gaussian", "pls-glm-logistic", "pls-glm-poisson", "pls-glm-polr", "pls-beta"). Use "modele=pls-glm-family" to enable the family option. |
| family | a description of the error distribution and link function to be used in the model. This can be a character string naming a family function, a family function or the result of a call to a family function. (See family for details of family functions.) To use the family option, please set modele="pls-glm-family". User defined families can also be defined. See details. |
| typeVC | type of leave one out cross validation. For back compatibility purpose.<br>**list("none")** no cross validation<br>**list("standard")** no cross validation<br>**list("missingdata")** no cross validation<br>**list("adaptative")** no cross validation |
| EstimXNA | only for modele="pls". Set whether the missing X values have to be estimated. |
| scaleX | scale the predictor(s) : must be set to TRUE for modele="pls" and should be for glms pls. |

| | |
|---|---|
| scaleY | scale the response : Yes/No. Ignored since not always possible for glm responses. |
| pvals.expli | should individual p-values be reported to tune model selection ? |
| alpha.pvals.expli | level of significance for predictors when pvals.expli=TRUE |
| MClassed | number of missclassified cases, should only be used for binary responses |
| tol_Xi | minimal value for Norm2(Xi) and $\det(pp' \times pp)$ if there is any missing value in the dataX. It defaults to $10^{-12}$ |
| weights | an optional vector of 'prior weights' to be used in the fitting process. Should be NULL or a numeric vector. |
| method | the link function for pls-glm-polr, logistic, probit, complementary log-log or cauchit (corresponding to a Cauchy latent variable). |
| sparse | should the coefficients of non-significant predictors (<alpha.pvals.expli) be set to 0 |
| sparseStop | should component extraction stop when no significant predictors (<alpha.pvals.expli) are found |
| naive | use the naive estimates for the Degrees of Freedom in plsR? Default is FALSE. |
| link | character specification of the link function in the mean model (mu). Currently, "logit", "probit", "cloglog", "cauchit", "log", "loglog" are supported. Alternatively, an object of class "link-glm" can be supplied. |
| link.phi | character specification of the link function in the precision model (phi). Currently, "identity", "log", "sqrt" are supported. The default is "log" unless formula is of type y~x where the default is "identity" (for backward compatibility). Alternatively, an object of class "link-glm" can be supplied. |
| type | character specification of the type of estimator. Currently, maximum likelihood ("ML"), ML with bias correction ("BC"), and ML with bias reduction ("BR") are supported. |
| verbose | should info messages be displayed ? |

## Details

There are seven different predefined models with predefined link functions available :

**list("\"pls\"")** ordinary pls models

**list("\"pls-glm-Gamma\"")** glm gaussian with inverse link pls models

**list("\"pls-glm-gaussian\"")** glm gaussian with identity link pls models

**list("\"pls-glm-inverse-gamma\"")** glm binomial with square inverse link pls models

**list("\"pls-glm-logistic\"")** glm binomial with logit link pls models

**list("\"pls-glm-poisson\"")** glm poisson with log link pls models

**list("\"pls-glm-polr\"")** glm polr with logit link pls models

Using the "family=" option and setting "modele=pls-glm-family" allows changing the family and link function the same way as for the glm function. As a consequence user-specified families can also be used.

**The** accepts the links (as names) `identity`, `log` and `inverse`.

**list("gaussian")** accepts the links (as names) `identity`, `log` and `inverse`.

**family** accepts the links (as names) `identity`, `log` and `inverse`.

**The** accepts the links `logit`, `probit`, `cauchit`, (corresponding to logistic, normal and Cauchy CDFs respectively) `log` and `cloglog` (complementary log-log).

**list("binomial")** accepts the links `logit`, `probit`, `cauchit`, (corresponding to logistic, normal and Cauchy CDFs respectively) `log` and `cloglog` (complementary log-log).

**family** accepts the links `logit`, `probit`, `cauchit`, (corresponding to logistic, normal and Cauchy CDFs respectively) `log` and `cloglog` (complementary log-log).

**The** accepts the links `inverse`, `identity` and `log`.

**list("Gamma")** accepts the links `inverse`, `identity` and `log`.

**family** accepts the links `inverse`, `identity` and `log`.

**The** accepts the links `log`, `identity`, and `sqrt`.

**list("poisson")** accepts the links `log`, `identity`, and `sqrt`.

**family** accepts the links `log`, `identity`, and `sqrt`.

**The** accepts the links `1/mu^2`, `inverse`, `identity` and `log`.

**list("inverse.gaussian")** accepts the links `1/mu^2`, `inverse`, `identity` and `log`.

**family** accepts the links `1/mu^2`, `inverse`, `identity` and `log`.

**The** accepts the links `logit`, `probit`, `cloglog`, `identity`, `inverse`, `log`, `1/mu^2` and `sqrt`.

**list("quasi")** accepts the links `logit`, `probit`, `cloglog`, `identity`, `inverse`, `log`, `1/mu^2` and `sqrt`.

**family** accepts the links `logit`, `probit`, `cloglog`, `identity`, `inverse`, `log`, `1/mu^2` and `sqrt`.

**The function** can be used to create a power link function.

**list("power")** can be used to create a power link function.

The default estimator for Degrees of Freedom is the Kramer and Sugiyama's one which only works for classical plsR models. For these models, Information criteria are computed accordingly to these estimations. Naive Degrees of Freedom and Information Criteria are also provided for comparison purposes. For more details, see Kraemer, N., Sugiyama M. (2010). "The Degrees of Freedom of Partial Least Squares Regression". preprint, http://arxiv.org/abs/1002.4112.

## Value

Depends on the model that was used to fit the model.

## Note

Use `plsRbeta` instead.

## Author(s)

Frédéric Bertrand
<frederic.bertrand@utt.fr>
https://fbertran.github.io/homepage/

## References

Frédéric Bertrand, Nicolas Meyer, Michèle Beau-Faller, Karim El Bayed, Izzie-Jacques Namer, Myriam Maumy-Bertrand (2013). Régression Bêta PLS. *Journal de la Société Française de Statistique*, **154**(3):143-159. http://publications-sfds.math.cnrs.fr/index.php/J-SFdS/article/view/215

## See Also

PLS_beta_wvc and PLS_beta_kfoldcv

## Examples

```
data("GasolineYield",package="betareg")
yGasolineYield <- GasolineYield$yield
XGasolineYield <- GasolineYield[,2:5]
modpls <- PLS_beta(yGasolineYield,XGasolineYield,nt=3,modele="pls-beta")
modpls$pp
modpls$Coeffs
modpls$Std.Coeffs
modpls$InfCrit
modpls$PredictY[1,]
rm("modpls")
```

---

# PLS_beta_formula *Partial least squares beta regression models*

---

## Description

This function implements Partial least squares beta regression models on complete or incomplete datasets (formula specification of the model).

## Usage

```
PLS_beta_formula(
  formula,
  data = NULL,
  nt = 2,
  limQ2set = 0.0975,
  dataPredictY = dataX,
  modele = "pls",
  family = NULL,
  typeVC = "none",
  EstimXNA = FALSE,
  scaleX = TRUE,
  scaleY = NULL,
```

```
  pvals.expli = FALSE,
  alpha.pvals.expli = 0.05,
  MClassed = FALSE,
  tol_Xi = 10^(-12),
  weights,
  subset,
  start = NULL,
  etastart,
  mustart,
  offset,
  method,
  control = list(),
  contrasts = NULL,
  sparse = FALSE,
  sparseStop = TRUE,
  naive = FALSE,
  link = NULL,
  link.phi = NULL,
  type = "ML",
  verbose = TRUE
)
```

## Arguments

| | |
|---|---|
| formula | an object of class "formula" (or one that can be coerced to that class): a symbolic description of the model to be fitted. The details of model specification are given under 'Details'. |
| data | an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the environment from which plsRbeta is called. |
| nt | number of components to be extracted |
| limQ2set | limit value for the Q2 |
| dataPredictY | predictor(s) (testing) dataset |
| modele | name of the PLS glm or PLS beta model to be fitted ("pls", "pls-glm-Gamma", "pls-glm-gaussian", "pls-glm-inverse.gaussian", "pls-glm-logistic", "pls-glm-poisson", "pls-glm-polr", "pls-beta"). Use "modele=pls-glm-family" to enable the family option. |
| family | a description of the error distribution and link function to be used in the model. This can be a character string naming a family function, a family function or the result of a call to a family function. (See family for details of family functions.) To use the family option, please set modele="pls-glm-family". User defined families can also be defined. See details. |
| typeVC | type of leave one out cross validation. For back compatibility purpose.<br>**list("none")** no cross validation<br>**list("standard")** no cross validation |

**list("missingdata")** no cross validation

**list("adaptative")** no cross validation

`EstimXNA` only for `modele="pls"`. Set whether the missing X values have to be estimated.

`scaleX` scale the predictor(s) : must be set to TRUE for `modele="pls"` and should be for glms pls.

`scaleY` scale the response : Yes/No. Ignored since not always possible for glm responses.

`pvals.expli` should individual p-values be reported to tune model selection ?

`alpha.pvals.expli`
level of significance for predictors when pvals.expli=TRUE

`MClassed` number of missclassified cases, should only be used for binary responses

`tol_Xi` minimal value for Norm2(Xi) and $\det(pp' \times pp)$ if there is any missing value in the `dataX`. It defaults to $10^{-12}$

`weights` an optional vector of 'prior weights' to be used in the fitting process. Should be `NULL` or a numeric vector.

`subset` an optional vector specifying a subset of observations to be used in the fitting process.

`start` starting values for the parameters in the linear predictor.

`etastart` starting values for the linear predictor.

`mustart` starting values for the vector of means.

`offset` this can be used to specify an *a priori* known component to be included in the linear predictor during fitting. This should be `NULL` or a numeric vector of length equal to the number of cases. One or more `offset` terms can be included in the formula instead or as well, and if more than one is specified their sum is used. See `model.offset`.

`method` **for fitting glms with glm (** the method to be used in fitting the model. The default method `"glm.fit"` uses iteratively reweighted least squares (IWLS). User-supplied fitting functions can be supplied either as a function or a character string naming a function, with a function which takes the same arguments as `glm.fit`. If `"model.frame"`, the model frame is returned.

**list("\"pls-glm-Gamma\"")** the method to be used in fitting the model. The default method `"glm.fit"` uses iteratively reweighted least squares (IWLS). User-supplied fitting functions can be supplied either as a function or a character string naming a function, with a function which takes the same arguments as `glm.fit`. If `"model.frame"`, the model frame is returned.

**,** the method to be used in fitting the model. The default method `"glm.fit"` uses iteratively reweighted least squares (IWLS). User-supplied fitting functions can be supplied either as a function or a character string naming a function, with a function which takes the same arguments as `glm.fit`. If `"model.frame"`, the model frame is returned.

**list("\"pls-glm-gaussian\"")** the method to be used in fitting the model. The default method `"glm.fit"` uses iteratively reweighted least squares (IWLS). User-supplied fitting functions can be supplied either as a function or a character string naming a function, with a function which takes the same arguments as `glm.fit`. If `"model.frame"`, the model frame is returned.

, the method to be used in fitting the model. The default method "glm.fit" uses iteratively reweighted least squares (IWLS). User-supplied fitting functions can be supplied either as a function or a character string naming a function, with a function which takes the same arguments as glm.fit. If "model.frame", the model frame is returned.

**list("\"pls-glm-inverse.gaussian\"")** the method to be used in fitting the model. The default method "glm.fit" uses iteratively reweighted least squares (IWLS). User-supplied fitting functions can be supplied either as a function or a character string naming a function, with a function which takes the same arguments as glm.fit. If "model.frame", the model frame is returned.

, the method to be used in fitting the model. The default method "glm.fit" uses iteratively reweighted least squares (IWLS). User-supplied fitting functions can be supplied either as a function or a character string naming a function, with a function which takes the same arguments as glm.fit. If "model.frame", the model frame is returned.

**list("\"pls-glm-logistic\"")** the method to be used in fitting the model. The default method "glm.fit" uses iteratively reweighted least squares (IWLS). User-supplied fitting functions can be supplied either as a function or a character string naming a function, with a function which takes the same arguments as glm.fit. If "model.frame", the model frame is returned.

, the method to be used in fitting the model. The default method "glm.fit" uses iteratively reweighted least squares (IWLS). User-supplied fitting functions can be supplied either as a function or a character string naming a function, with a function which takes the same arguments as glm.fit. If "model.frame", the model frame is returned.

**list("\"pls-glm-poisson\"")** the method to be used in fitting the model. The default method "glm.fit" uses iteratively reweighted least squares (IWLS). User-supplied fitting functions can be supplied either as a function or a character string naming a function, with a function which takes the same arguments as glm.fit. If "model.frame", the model frame is returned.

, the method to be used in fitting the model. The default method "glm.fit" uses iteratively reweighted least squares (IWLS). User-supplied fitting functions can be supplied either as a function or a character string naming a function, with a function which takes the same arguments as glm.fit. If "model.frame", the model frame is returned.

**list("\"modele=pls-glm-family\"")** the method to be used in fitting the model. The default method "glm.fit" uses iteratively reweighted least squares (IWLS). User-supplied fitting functions can be supplied either as a function or a character string naming a function, with a function which takes the same arguments as glm.fit. If "model.frame", the model frame is returned.

) the method to be used in fitting the model. The default method "glm.fit" uses iteratively reweighted least squares (IWLS). User-supplied fitting functions can be supplied either as a function or a character string naming a function, with a function which takes the same arguments as glm.fit. If "model.frame", the model frame is returned.

**list("pls-glm-polr")** logistic, probit, complementary log-log or cauchit (corresponding to a Cauchy latent variable).

`control` a list of parameters for controlling the fitting process. For `glm.fit` this is passed to `glm.control`.

`contrasts` an optional list. See the `contrasts.arg` of `model.matrix.default`.

`sparse` should the coefficients of non-significant predictors (<`alpha.pvals.expli`) be set to 0

`sparseStop` should component extraction stop when no significant predictors (<`alpha.pvals.expli`) are found

`naive` Use the naive estimates for the Degrees of Freedom in plsR? Default is `FALSE`.

`link` character specification of the link function in the mean model (mu). Currently, "`logit`", "`probit`", "`cloglog`", "`cauchit`", "`log`", "`loglog`" are supported. Alternatively, an object of class "`link-glm`" can be supplied.

`link.phi` character specification of the link function in the precision model (phi). Currently, "`identity`", "`log`", "`sqrt`" are supported. The default is "`log`" unless `formula` is of type `y~x` where the default is "`identity`" (for backward compatibility). Alternatively, an object of class "`link-glm`" can be supplied.

`type` character specification of the type of estimator. Currently, maximum likelihood ("`ML`"), ML with bias correction ("`BC`"), and ML with bias reduction ("`BR`") are supported.

`verbose` should info messages be displayed ?

## Details

There are seven different predefined models with predefined link functions available :

**list("\"pls\"")** ordinary pls models

**list("\"pls-glm-Gamma\"")** glm gaussian with inverse link pls models

**list("\"pls-glm-gaussian\"")** glm gaussian with identity link pls models

**list("\"pls-glm-inverse-gamma\"")** glm binomial with square inverse link pls models

**list("\"pls-glm-logistic\"")** glm binomial with logit link pls models

**list("\"pls-glm-poisson\"")** glm poisson with log link pls models

**list("\"pls-glm-polr\"")** glm polr with logit link pls models

Using the `"family="` option and setting `"modele=pls-glm-family"` allows changing the family and link function the same way as for the `glm` function. As a consequence user-specified families can also be used.

**The** accepts the links (as names) `identity`, `log` and `inverse`.

**list("gaussian")** accepts the links (as names) `identity`, `log` and `inverse`.

**family** accepts the links (as names) `identity`, `log` and `inverse`.

**The** accepts the links `logit`, `probit`, `cauchit`, (corresponding to logistic, normal and Cauchy CDFs respectively) `log` and `cloglog` (complementary log-log).

**list("binomial")** accepts the links `logit`, `probit`, `cauchit`, (corresponding to logistic, normal and Cauchy CDFs respectively) `log` and `cloglog` (complementary log-log).

**family** accepts the links `logit`, `probit`, `cauchit`, (corresponding to logistic, normal and Cauchy CDFs respectively) `log` and `cloglog` (complementary log-log).

**The** accepts the links `inverse`, `identity` and `log`.

**list("Gamma")** accepts the links `inverse`, `identity` and `log`.

**family** accepts the links `inverse`, `identity` and `log`.

**The** accepts the links `log`, `identity`, and `sqrt`.

**list("poisson")** accepts the links `log`, `identity`, and `sqrt`.

**family** accepts the links `log`, `identity`, and `sqrt`.

**The** accepts the links `1/mu^2`, `inverse`, `identity` and `log`.

**list("inverse.gaussian")** accepts the links `1/mu^2`, `inverse`, `identity` and `log`.

**family** accepts the links `1/mu^2`, `inverse`, `identity` and `log`.

**The** accepts the links `logit`, `probit`, `cloglog`, `identity`, `inverse`, `log`, `1/mu^2` and `sqrt`.

**list("quasi")** accepts the links `logit`, `probit`, `cloglog`, `identity`, `inverse`, `log`, `1/mu^2` and `sqrt`.

**family** accepts the links `logit`, `probit`, `cloglog`, `identity`, `inverse`, `log`, `1/mu^2` and `sqrt`.

**The function** can be used to create a power link function.

**list("power")** can be used to create a power link function.

A typical predictor has the form response ~ terms where response is the (numeric) response vector and terms is a series of terms which specifies a linear predictor for response. A terms specification of the form first + second indicates all the terms in first together with all the terms in second with any duplicates removed.

A specification of the form first:second indicates the the set of terms obtained by taking the interactions of all terms in first with all terms in second. The specification first*second indicates the cross of first and second. This is the same as first + second + first:second.

The terms in the formula will be re-ordered so that main effects come first, followed by the interactions, all second-order, all third-order and so on: to avoid this pass a terms object as the formula.

Non-NULL weights can be used to indicate that different observations have different dispersions (with the values in weights being inversely proportional to the dispersions); or equivalently, when the elements of weights are positive integers w_i, that each response y_i is the mean of w_i unit-weight observations.

The default estimator for Degrees of Freedom is the Kramer and Sugiyama's one which only works for classical plsR models. For these models, Information criteria are computed accordingly to these estimations. Naive Degrees of Freedom and Information Criteria are also provided for comparison purposes. For more details, see Kraemer, N., Sugiyama M. (2010). "The Degrees of Freedom of Partial Least Squares Regression". preprint, http://arxiv.org/abs/1002.4112.

## Value

Depends on the model that was used to fit the model.

## Note

Use plsRbeta instead.

## Author(s)

Frédéric Bertrand
<frederic.bertrand@utt.fr>
https://fbertran.github.io/homepage/

## References

Frédéric Bertrand, Nicolas Meyer, Michèle Beau-Faller, Karim El Bayed, Izzie-Jacques Namer, Myriam Maumy-Bertrand (2013). Régression Bêta PLS. *Journal de la Société Française de Statistique*, **154**(3):143-159. http://publications-sfds.math.cnrs.fr/index.php/J-SFdS/article/view/215

## See Also

PLS_beta_wvc and PLS_beta_kfoldcv_formula

## Examples

```
data("GasolineYield",package="betareg")
modpls <- PLS_beta_formula(yield~.,data=GasolineYield,nt=3,modele="pls-beta")
modpls$pp
modpls$Coeffs
modpls$Std.Coeffs
modpls$InfCrit
modpls$PredictY[1,]
rm("modpls")
```

---

PLS_beta_kfoldcv *Partial least squares regression beta models with kfold cross validation*

---

## Description

This function implements kfold cross validation on complete or incomplete datasets for partial least squares beta regression models

## Usage

```
PLS_beta_kfoldcv(
  dataY,
  dataX,
  nt = 2,
  limQ2set = 0.0975,
  modele = "pls",
  family = NULL,
  K = nrow(dataX),
  NK = 1,
  grouplist = NULL,
  random = FALSE,
  scaleX = TRUE,
  scaleY = NULL,
  keepcoeffs = FALSE,
  keepfolds = FALSE,
  keepdataY = TRUE,
  keepMclassed = FALSE,
  tol_Xi = 10^(-12),
  weights,
  method,
  link = NULL,
  link.phi = NULL,
  type = "ML",
  verbose = TRUE
)
```

## Arguments

| | |
|---|---|
| dataY | response (training) dataset |
| dataX | predictor(s) (training) dataset |
| nt | number of components to be extracted |
| limQ2set | limit value for the Q2 |
| modele | name of the PLS glm or PLS beta model to be fitted ("pls", "pls-glm-Gamma", "pls-glm-gaussian", "pls-glm-inverse.gaussian", "pls-glm-logistic", "pls-glm-poisson", "pls-glm-polr", "pls-beta"). Use "modele=pls-glm-family" to enable the family option. |
| family | a description of the error distribution and link function to be used in the model. This can be a character string naming a family function, a family function or the result of a call to a family function. (See family for details of family functions.) To use the family option, please set modele="pls-glm-family". User defined families can also be defined. See details. |
| K | number of groups |
| NK | number of times the group division is made |
| grouplist | to specify the members of the K groups |
| random | should the K groups be made randomly |

| | |
|---|---|
| scaleX | scale the predictor(s) : must be set to TRUE for modele="pls" and should be for glms pls. |
| scaleY | scale the response : Yes/No. Ignored since non always possible for glm responses. |
| keepcoeffs | shall the coefficients for each model be returned |
| keepfolds | shall the groups' composition be returned |
| keepdataY | shall the observed value of the response for each one of the predicted value be returned |
| keepMclassed | shall the number of miss classed be returned (unavailable) |
| tol_Xi | minimal value for Norm2(Xi) and $\det(pp' \times pp)$ if there is any missing value in the dataX. It defaults to $10^{-12}$ |
| weights | an optional vector of 'prior weights' to be used in the fitting process. Should be NULL or a numeric vector. |
| method | logistic, probit, complementary log-log or cauchit (corresponding to a Cauchy latent variable). |
| link | character specification of the link function in the mean model (mu). Currently, "logit", "probit", "cloglog", "cauchit", "log", "loglog" are supported. Alternatively, an object of class "link-glm" can be supplied. |
| link.phi | character specification of the link function in the precision model (phi). Currently, "identity", "log", "sqrt" are supported. The default is "log" unless formula is of type y~x where the default is "identity" (for backward compatibility). Alternatively, an object of class "link-glm" can be supplied. |
| type | character specification of the type of estimator. Currently, maximum likelihood ("ML"), ML with bias correction ("BC"), and ML with bias reduction ("BR") are supported. |
| verbose | should info messages be displayed ? |

## Details

Predicts 1 group with the K-1 other groups. Leave one out cross validation is thus obtained for K==nrow(dataX).

There are seven different predefined models with predefined link functions available :

**list("\"pls\"")** ordinary pls models

**list("\"pls-glm-Gamma\"")** glm gaussian with inverse link pls models

**list("\"pls-glm-gaussian\"")** glm gaussian with identity link pls models

**list("\"pls-glm-inverse-gamma\"")** glm binomial with square inverse link pls models

**list("\"pls-glm-logistic\"")** glm binomial with logit link pls models

**list("\"pls-glm-poisson\"")** glm poisson with log link pls models

**list("\"pls-glm-polr\"")** glm polr with logit link pls models

Using the "family=" option and setting "modele=pls-glm-family" allows changing the family and link function the same way as for the glm function. As a consequence user-specified families can also be used.

**The** accepts the links (as names) `identity`, `log` and `inverse`.

**list("gaussian")** accepts the links (as names) `identity`, `log` and `inverse`.

**family** accepts the links (as names) `identity`, `log` and `inverse`.

**The** accepts the links `logit`, `probit`, `cauchit`, (corresponding to logistic, normal and Cauchy CDFs respectively) `log` and `cloglog` (complementary log-log).

**list("binomial")** accepts the links `logit`, `probit`, `cauchit`, (corresponding to logistic, normal and Cauchy CDFs respectively) `log` and `cloglog` (complementary log-log).

**family** accepts the links `logit`, `probit`, `cauchit`, (corresponding to logistic, normal and Cauchy CDFs respectively) `log` and `cloglog` (complementary log-log).

**The** accepts the links `inverse`, `identity` and `log`.

**list("Gamma")** accepts the links `inverse`, `identity` and `log`.

**family** accepts the links `inverse`, `identity` and `log`.

**The** accepts the links `log`, `identity`, and `sqrt`.

**list("poisson")** accepts the links `log`, `identity`, and `sqrt`.

**family** accepts the links `log`, `identity`, and `sqrt`.

**The** accepts the links `1/mu^2`, `inverse`, `identity` and `log`.

**list("inverse.gaussian")** accepts the links `1/mu^2`, `inverse`, `identity` and `log`.

**family** accepts the links `1/mu^2`, `inverse`, `identity` and `log`.

**The** accepts the links `logit`, `probit`, `cloglog`, `identity`, `inverse`, `log`, `1/mu^2` and `sqrt`.

**list("quasi")** accepts the links `logit`, `probit`, `cloglog`, `identity`, `inverse`, `log`, `1/mu^2` and `sqrt`.

**family** accepts the links `logit`, `probit`, `cloglog`, `identity`, `inverse`, `log`, `1/mu^2` and `sqrt`.

**The function** can be used to create a power link function.

**list("power")** can be used to create a power link function.

Non-NULL weights can be used to indicate that different observations have different dispersions (with the values in weights being inversely proportional to the dispersions); or equivalently, when the elements of weights are positive integers w_i, that each response y_i is the mean of w_i unit-weight observations.

## Value

`results_kfolds` list of `NK`. Each element of the list sums up the results for a group division:

- **list** of `K` matrices of size about `nrow(dataX)/K * nt` with the predicted values for a growing number of components
- **list()** ...
- **list** of `K` matrices of size about `nrow(dataX)/K * nt` with the predicted values for a growing number of components

`folds` list of `NK`. Each element of the list sums up the informations for a group division:

- **list** of `K` vectors of length about `nrow(dataX)` with the numbers of the rows of `dataX` that were used as a training set
- **list()** ...

**list** of K vectors of length about nrow(dataX) with the numbers of the rows of dataX that were used as a training set

dataY_kfolds — list of NK. Each element of the list sums up the results for a group division:

**list** of K matrices of size about nrow(dataX)/K * 1 with the observed values of the response

**list()** ...

**list** of K matrices of size about nrow(dataX)/K * 1 with the observed values of the response

call — the call of the function

## Note

Works for complete and incomplete datasets.

## Author(s)

Frédéric Bertrand
<frederic.bertrand@utt.fr>
https://fbertran.github.io/homepage/

## References

Frédéric Bertrand, Nicolas Meyer, Michèle Beau-Faller, Karim El Bayed, Izzie-Jacques Namer, Myriam Maumy-Bertrand (2013). Régression Bêta PLS. *Journal de la Société Française de Statistique*, **154**(3):143-159. http://publications-sfds.math.cnrs.fr/index.php/J-SFdS/article/view/215

## See Also

kfolds2coeff, kfolds2Pressind, kfolds2Press, kfolds2Mclassedind, kfolds2Mclassed and kfolds2CVinfos_beta to extract and transform results from kfold cross validation.

## Examples

```
## Not run:
data("GasolineYield",package="betareg")
yGasolineYield <- GasolineYield$yield
XGasolineYield <- GasolineYield[,2:5]
bbb <- PLS_beta_kfoldcv(yGasolineYield,XGasolineYield,nt=3,modele="pls-beta")
kfolds2CVinfos_beta(bbb)

## End(Not run)
```

PLS_beta_kfoldcv_formula *Partial least squares regression beta models with kfold cross validation*

## Description

This function implements kfold cross validation on complete or incomplete datasets for partial least squares beta regression models (formula specification of the model).

## Usage

```
PLS_beta_kfoldcv_formula(
  formula,
  data = NULL,
  nt = 2,
  limQ2set = 0.0975,
  modele = "pls",
  family = NULL,
  K = nrow(dataX),
  NK = 1,
  grouplist = NULL,
  random = FALSE,
  scaleX = TRUE,
  scaleY = NULL,
  keepcoeffs = FALSE,
  keepfolds = FALSE,
  keepdataY = TRUE,
  keepMclassed = FALSE,
  tol_Xi = 10^(-12),
  weights,
  subset,
  start = NULL,
  etastart,
  mustart,
  offset,
  method,
  control = list(),
  contrasts = NULL,
  sparse = FALSE,
  sparseStop = TRUE,
  naive = FALSE,
  link = NULL,
  link.phi = NULL,
  type = "ML",
  verbose = TRUE
)
```

## Arguments

| | |
|---|---|
| formula | an object of class "formula" (or one that can be coerced to that class): a symbolic description of the model to be fitted. The details of model specification are given under 'Details'. |
| data | an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the environment from which plsRglm is called. |
| nt | number of components to be extracted |
| limQ2set | limit value for the Q2 |
| modele | name of the PLS glm or PLS beta model to be fitted ("pls", "pls-glm-Gamma", "pls-glm-gaussian", "pls-glm-inverse.gaussian", "pls-glm-logistic", "pls-glm-poisson", "pls-glm-polr", "pls-beta"). Use "modele=pls-glm-family" to enable the family option. |
| family | a description of the error distribution and link function to be used in the model. This can be a character string naming a family function, a family function or the result of a call to a family function. (See family for details of family functions.) To use the family option, please set modele="pls-glm-family". User defined families can also be defined. See details. |
| K | number of groups |
| NK | number of times the group division is made |
| grouplist | to specify the members of the K groups |
| random | should the K groups be made randomly |
| scaleX | scale the predictor(s) : must be set to TRUE for modele="pls" and should be for glms pls. |
| scaleY | scale the response : Yes/No. Ignored since non always possible for glm responses. |
| keepcoeffs | shall the coefficients for each model be returned |
| keepfolds | shall the groups' composition be returned |
| keepdataY | shall the observed value of the response for each one of the predicted value be returned |
| keepMclassed | shall the number of miss classed be returned (unavailable) |
| tol_Xi | minimal value for Norm2(Xi) and $\det(pp' \times pp)$ if there is any missing value in the dataX. It defaults to $10^{-12}$ |
| weights | an optional vector of 'prior weights' to be used in the fitting process. Should be NULL or a numeric vector. |
| subset | an optional vector specifying a subset of observations to be used in the fitting process. |
| start | starting values for the parameters in the linear predictor. |
| etastart | starting values for the linear predictor. |
| mustart | starting values for the vector of means. |

offset — this can be used to specify an *a priori* known component to be included in the linear predictor during fitting. This should be NULL or a numeric vector of length equal to the number of cases. One or more offset terms can be included in the formula instead or as well, and if more than one is specified their sum is used. See model.offset.

method — **for fitting glms with glm (** the method to be used in fitting the model. The default method "glm.fit" uses iteratively reweighted least squares (IWLS). User-supplied fitting functions can be supplied either as a function or a character string naming a function, with a function which takes the same arguments as glm.fit. If "model.frame", the model frame is returned.

**list("\"pls-glm-Gamma\"")** the method to be used in fitting the model. The default method "glm.fit" uses iteratively reweighted least squares (IWLS). User-supplied fitting functions can be supplied either as a function or a character string naming a function, with a function which takes the same arguments as glm.fit. If "model.frame", the model frame is returned.

**,** the method to be used in fitting the model. The default method "glm.fit" uses iteratively reweighted least squares (IWLS). User-supplied fitting functions can be supplied either as a function or a character string naming a function, with a function which takes the same arguments as glm.fit. If "model.frame", the model frame is returned.

**list("\"pls-glm-gaussian\"")** the method to be used in fitting the model. The default method "glm.fit" uses iteratively reweighted least squares (IWLS). User-supplied fitting functions can be supplied either as a function or a character string naming a function, with a function which takes the same arguments as glm.fit. If "model.frame", the model frame is returned.

**,** the method to be used in fitting the model. The default method "glm.fit" uses iteratively reweighted least squares (IWLS). User-supplied fitting functions can be supplied either as a function or a character string naming a function, with a function which takes the same arguments as glm.fit. If "model.frame", the model frame is returned.

**list("\"pls-glm-inverse.gaussian\"")** the method to be used in fitting the model. The default method "glm.fit" uses iteratively reweighted least squares (IWLS). User-supplied fitting functions can be supplied either as a function or a character string naming a function, with a function which takes the same arguments as glm.fit. If "model.frame", the model frame is returned.

**,** the method to be used in fitting the model. The default method "glm.fit" uses iteratively reweighted least squares (IWLS). User-supplied fitting functions can be supplied either as a function or a character string naming a function, with a function which takes the same arguments as glm.fit. If "model.frame", the model frame is returned.

**list("\"pls-glm-logistic\"")** the method to be used in fitting the model. The default method "glm.fit" uses iteratively reweighted least squares (IWLS). User-supplied fitting functions can be supplied either as a function or a character string naming a function, with a function which takes the same arguments as glm.fit. If "model.frame", the model frame is returned.

**,** the method to be used in fitting the model. The default method "glm.fit"

uses iteratively reweighted least squares (IWLS). User-supplied fitting functions can be supplied either as a function or a character string naming a function, with a function which takes the same arguments as `glm.fit`. If "model.frame", the model frame is returned.

**list("\"pls-glm-poisson\"")** the method to be used in fitting the model. The default method `"glm.fit"` uses iteratively reweighted least squares (IWLS). User-supplied fitting functions can be supplied either as a function or a character string naming a function, with a function which takes the same arguments as `glm.fit`. If "model.frame", the model frame is returned.

**,** the method to be used in fitting the model. The default method `"glm.fit"` uses iteratively reweighted least squares (IWLS). User-supplied fitting functions can be supplied either as a function or a character string naming a function, with a function which takes the same arguments as `glm.fit`. If "model.frame", the model frame is returned.

**list("\"modele=pls-glm-family\"")** the method to be used in fitting the model. The default method `"glm.fit"` uses iteratively reweighted least squares (IWLS). User-supplied fitting functions can be supplied either as a function or a character string naming a function, with a function which takes the same arguments as `glm.fit`. If "model.frame", the model frame is returned.

**)** the method to be used in fitting the model. The default method `"glm.fit"` uses iteratively reweighted least squares (IWLS). User-supplied fitting functions can be supplied either as a function or a character string naming a function, with a function which takes the same arguments as `glm.fit`. If "model.frame", the model frame is returned.

**list("pls-glm-polr")** logistic, probit, complementary log-log or cauchit (corresponding to a Cauchy latent variable).

`control` a list of parameters for controlling the fitting process. For `glm.fit` this is passed to `glm.control`.

`contrasts` an optional list. See the `contrasts.arg` of `model.matrix.default`.

`sparse` should the coefficients of non-significant predictors (<`alpha.pvals.expli`) be set to 0

`sparseStop` should component extraction stop when no significant predictors (<`alpha.pvals.expli`) are found

`naive` Use the naive estimates for the Degrees of Freedom in plsR? Default is `FALSE`.

`link` character specification of the link function in the mean model (mu). Currently, `"logit"`, `"probit"`, `"cloglog"`, `"cauchit"`, `"log"`, `"loglog"` are supported. Alternatively, an object of class `"link-glm"` can be supplied.

`link.phi` character specification of the link function in the precision model (phi). Currently, `"identity"`, `"log"`, `"sqrt"` are supported. The default is `"log"` unless `formula` is of type `y~x` where the default is `"identity"` (for backward compatibility). Alternatively, an object of class `"link-glm"` can be supplied.

`type` character specification of the type of estimator. Currently, maximum likelihood (`"ML"`), ML with bias correction (`"BC"`), and ML with bias reduction (`"BR"`) are supported.

`verbose` should info messages be displayed ?

## Details

Predicts 1 group with the `K-1` other groups. Leave one out cross validation is thus obtained for `K==nrow(dataX)`.

There are seven different predefined models with predefined link functions available :

**list("\"pls\"")** ordinary pls models

**list("\"pls-glm-Gamma\"")** glm gaussian with inverse link pls models

**list("\"pls-glm-gaussian\"")** glm gaussian with identity link pls models

**list("\"pls-glm-inverse-gamma\"")** glm binomial with square inverse link pls models

**list("\"pls-glm-logistic\"")** glm binomial with logit link pls models

**list("\"pls-glm-poisson\"")** glm poisson with log link pls models

**list("\"pls-glm-polr\"")** glm polr with logit link pls models

Using the `"family="` option and setting `"modele=pls-glm-family"` allows changing the family and link function the same way as for the `glm` function. As a consequence user-specified families can also be used.

**The** accepts the links (as names) `identity`, `log` and `inverse`.

**list("gaussian")** accepts the links (as names) `identity`, `log` and `inverse`.

**family** accepts the links (as names) `identity`, `log` and `inverse`.

**The** accepts the links `logit`, `probit`, `cauchit`, (corresponding to logistic, normal and Cauchy CDFs respectively) `log` and `cloglog` (complementary log-log).

**list("binomial")** accepts the links `logit`, `probit`, `cauchit`, (corresponding to logistic, normal and Cauchy CDFs respectively) `log` and `cloglog` (complementary log-log).

**family** accepts the links `logit`, `probit`, `cauchit`, (corresponding to logistic, normal and Cauchy CDFs respectively) `log` and `cloglog` (complementary log-log).

**The** accepts the links `inverse`, `identity` and `log`.

**list("Gamma")** accepts the links `inverse`, `identity` and `log`.

**family** accepts the links `inverse`, `identity` and `log`.

**The** accepts the links `log`, `identity`, and `sqrt`.

**list("poisson")** accepts the links `log`, `identity`, and `sqrt`.

**family** accepts the links `log`, `identity`, and `sqrt`.

**The** accepts the links `1/mu^2`, `inverse`, `identity` and `log`.

**list("inverse.gaussian")** accepts the links `1/mu^2`, `inverse`, `identity` and `log`.

**family** accepts the links `1/mu^2`, `inverse`, `identity` and `log`.

**The** accepts the links `logit`, `probit`, `cloglog`, `identity`, `inverse`, `log`, `1/mu^2` and `sqrt`.

**list("quasi")** accepts the links `logit`, `probit`, `cloglog`, `identity`, `inverse`, `log`, `1/mu^2` and `sqrt`.

**family** accepts the links `logit`, `probit`, `cloglog`, `identity`, `inverse`, `log`, `1/mu^2` and `sqrt`.

**The function** can be used to create a power link function.

**list("power")** can be used to create a power link function.

A typical predictor has the form response ~ terms where response is the (numeric) response vector and terms is a series of terms which specifies a linear predictor for response. A terms specification of the form first + second indicates all the terms in first together with all the terms in second with any duplicates removed.

A specification of the form first:second indicates the the set of terms obtained by taking the interactions of all terms in first with all terms in second. The specification first*second indicates the cross of first and second. This is the same as first + second + first:second.

The terms in the formula will be re-ordered so that main effects come first, followed by the interactions, all second-order, all third-order and so on: to avoid this pass a terms object as the formula.

Non-NULL weights can be used to indicate that different observations have different dispersions (with the values in weights being inversely proportional to the dispersions); or equivalently, when the elements of weights are positive integers w_i, that each response y_i is the mean of w_i unit-weight observations.

## Value

`results_kfolds` list of `NK`. Each element of the list sums up the results for a group division:

- **list** of `K` matrices of size about `nrow(dataX)/K * nt` with the predicted values for a growing number of components
- **list()** ...
- **list** of `K` matrices of size about `nrow(dataX)/K * nt` with the predicted values for a growing number of components

`folds` list of `NK`. Each element of the list sums up the informations for a group division:

- **list** of `K` vectors of length about `nrow(dataX)` with the numbers of the rows of `dataX` that were used as a training set
- **list()** ...
- **list** of `K` vectors of length about `nrow(dataX)` with the numbers of the rows of `dataX` that were used as a training set

`dataY_kfolds` list of `NK`. Each element of the list sums up the results for a group division:

- **list** of `K` matrices of size about `nrow(dataX)/K * 1` with the observed values of the response
- **list()** ...
- **list** of `K` matrices of size about `nrow(dataX)/K * 1` with the observed values of the response

`call` the call of the function

## Note

Work for complete and incomplete datasets.

## Author(s)

Frédéric Bertrand
<frederic.bertrand@utt.fr>
https://fbertran.github.io/homepage/

## References

Frédéric Bertrand, Nicolas Meyer, Michèle Beau-Faller, Karim El Bayed, Izzie-Jacques Namer, Myriam Maumy-Bertrand (2013). Régression Bêta PLS. *Journal de la Société Française de Statistique*, **154**(3):143-159. http://publications-sfds.math.cnrs.fr/index.php/J-SFdS/article/view/215

## See Also

kfolds2coeff, kfolds2Pressind, kfolds2Press, kfolds2Mclassedind, kfolds2Mclassed and kfolds2CVinfos_beta to extract and transform results from kfold cross validation.

## Examples

```
## Not run:
data("GasolineYield",package="betareg")
bbb <- PLS_beta_kfoldcv_formula(yield~.,data=GasolineYield,nt=3,modele="pls-beta")
kfolds2CVinfos_beta(bbb)

## End(Not run)
```

---

PLS_beta_wvc *Light version of PLS_beta for cross validation purposes*

---

## Description

Light version of PLS_beta for cross validation purposes either on complete or incomplete datasets.

## Usage

```
PLS_beta_wvc(
  dataY,
  dataX,
  nt = 2,
  dataPredictY = dataX,
  modele = "pls",
  family = NULL,
  scaleX = TRUE,
  scaleY = NULL,
  keepcoeffs = FALSE,
  keepstd.coeffs = FALSE,
  tol_Xi = 10^(-12),
  weights,
  method = "logistic",
  link = NULL,
  link.phi = NULL,
  type = "ML",
  verbose = TRUE
)
```

## Arguments

| | |
|---|---|
| dataY | response (training) dataset |
| dataX | predictor(s) (training) dataset |
| nt | number of components to be extracted |
| dataPredictY | predictor(s) (testing) dataset |
| modele | name of the PLS glm or PLS beta model to be fitted ("pls", "pls-glm-Gamma", "pls-glm-gaussian", "pls-glm-inverse.gaussian", "pls-glm-logistic", "pls-glm-poisson", "pls-glm-polr", "pls-beta"). Use "modele=pls-glm-family" to enable the family option. |
| family | a description of the error distribution and link function to be used in the model. This can be a character string naming a family function, a family function or the result of a call to a family function. (See family for details of family functions.) To use the family option, please set modele="pls-glm-family". User defined families can also be defined. See details. |
| scaleX | scale the predictor(s) : must be set to TRUE for modele="pls" and should be for glms pls. |
| scaleY | scale the response : Yes/No. Ignored since non always possible for glm responses. |
| keepcoeffs | whether the coefficients of the linear fit on link scale of unstandardized eXplanatory variables should be returned or not. |
| keepstd.coeffs | whether the coefficients of the linear fit on link scale of standardized eXplanatory variables should be returned or not. |
| tol_Xi | minimal value for Norm2(Xi) and $\det(pp' \times pp)$ if there is any missing value in the dataX. It defaults to $10^{-12}$ |
| weights | an optional vector of 'prior weights' to be used in the fitting process. Should be NULL or a numeric vector. |
| method | logistic, probit, complementary log-log or cauchit (corresponding to a Cauchy latent variable). |
| link | character specification of the link function in the mean model (mu). Currently, "logit", "probit", "cloglog", "cauchit", "log", "loglog" are supported. Alternatively, an object of class "link-glm" can be supplied. |
| link.phi | character specification of the link function in the precision model (phi). Currently, "identity", "log", "sqrt" are supported. The default is "log" unless formula is of type y~x where the default is "identity" (for backward compatibility). Alternatively, an object of class "link-glm" can be supplied. |
| type | character specification of the type of estimator. Currently, maximum likelihood ("ML"), ML with bias correction ("BC"), and ML with bias reduction ("BR") are supported. |
| verbose | should info messages be displayed ? |

## Details

This function is called by PLS_glm_kfoldcv_formula in order to perform cross validation either on complete or incomplete datasets.

There are seven different predefined models with predefined link functions available :

**list("\"pls\"")** ordinary pls models

**list("\"pls-glm-Gamma\"")** glm gaussian with inverse link pls models

**list("\"pls-glm-gaussian\"")** glm gaussian with identity link pls models

**list("\"pls-glm-inverse-gamma\"")** glm binomial with square inverse link pls models

**list("\"pls-glm-logistic\"")** glm binomial with logit link pls models

**list("\"pls-glm-poisson\"")** glm poisson with log link pls models

**list("\"pls-glm-polr\"")** glm polr with logit link pls models

Using the "family=" option and setting "modele=pls-glm-family" allows changing the family and link function the same way as for the glm function. As a consequence user-specified families can also be used.

**The** accepts the links (as names) identity, log and inverse.

**list("gaussian")** accepts the links (as names) identity, log and inverse.

**family** accepts the links (as names) identity, log and inverse.

**The** accepts the links logit, probit, cauchit, (corresponding to logistic, normal and Cauchy CDFs respectively) log and cloglog (complementary log-log).

**list("binomial")** accepts the links logit, probit, cauchit, (corresponding to logistic, normal and Cauchy CDFs respectively) log and cloglog (complementary log-log).

**family** accepts the links logit, probit, cauchit, (corresponding to logistic, normal and Cauchy CDFs respectively) log and cloglog (complementary log-log).

**The** accepts the links inverse, identity and log.

**list("Gamma")** accepts the links inverse, identity and log.

**family** accepts the links inverse, identity and log.

**The** accepts the links log, identity, and sqrt.

**list("poisson")** accepts the links log, identity, and sqrt.

**family** accepts the links log, identity, and sqrt.

**The** accepts the links 1/mu^2, inverse, identity and log.

**list("inverse.gaussian")** accepts the links 1/mu^2, inverse, identity and log.

**family** accepts the links 1/mu^2, inverse, identity and log.

**The** accepts the links logit, probit, cloglog, identity, inverse, log, 1/mu^2 and sqrt.

**list("quasi")** accepts the links logit, probit, cloglog, identity, inverse, log, 1/mu^2 and sqrt.

**family** accepts the links logit, probit, cloglog, identity, inverse, log, 1/mu^2 and sqrt.

**The function** can be used to create a power link function.

**list("power")** can be used to create a power link function.

Non-NULL weights can be used to indicate that different observations have different dispersions (with the values in weights being inversely proportional to the dispersions); or equivalently, when the elements of weights are positive integers w_i, that each response y_i is the mean of w_i unit-weight observations.

## Value

| | |
|---|---|
| `valsPredict` | `nrow(dataPredictY) * nt` matrix of the predicted values |
| `list("coeffs")` | If the coefficients of the eXplanatory variables were requested:<br>i.e. `keepcoeffs=TRUE`.<br>`ncol(dataX) * 1` matrix of the coefficients of the the eXplanatory variables |

## Author(s)

Frédéric Bertrand
<frederic.bertrand@utt.fr>
https://fbertran.github.io/homepage/

## References

Frédéric Bertrand, Nicolas Meyer, Michèle Beau-Faller, Karim El Bayed, Izzie-Jacques Namer, Myriam Maumy-Bertrand (2013). Régression Bêta PLS. *Journal de la Société Française de Statistique*, **154**(3):143-159. http://publications-sfds.math.cnrs.fr/index.php/J-SFdS/article/view/215

## See Also

`PLS_beta` for more detailed results, `PLS_beta_kfoldcv` for cross validating models and `PLS_lm_wvc` for the same function dedicated to plsR models

## Examples

```
data("GasolineYield",package="betareg")
yGasolineYield <- GasolineYield$yield
XGasolineYield <- GasolineYield[,2:5]
modpls <- PLS_beta_wvc(yGasolineYield,XGasolineYield,nt=3,modele="pls-beta")
modpls
rm("modpls")
```

`print.plsRbetamodel` *Print method for plsRbeta models*

## Description

This function provides a print method for the class "plsRbetamodel"

## Usage

```
## S3 method for class 'plsRbetamodel'
print(x, ...)
```

## Arguments

| | |
|---|---|
| x | an object of the class "plsRbetamodel" |
| ... | not used |

## Value

NULL

## Author(s)

Frédéric Bertrand
<frederic.bertrand@utt.fr>
https://fbertran.github.io/homepage/

## References

Frédéric Bertrand, Nicolas Meyer, Michèle Beau-Faller, Karim El Bayed, Izzie-Jacques Namer, Myriam Maumy-Bertrand (2013). Régression Bêta PLS. *Journal de la Société Française de Statistique*, **154**(3):143-159. http://publications-sfds.math.cnrs.fr/index.php/J-SFdS/article/view/215

## See Also

print

## Examples

```
data("GasolineYield",package="betareg")
modpls <- plsRbeta(yield~.,data=GasolineYield,nt=3,modele="pls-beta")
print(modpls)
```

print.summary.plsRbetamodel *Print method for summaries of plsRbeta models*

## Description

This function provides a print method for the class "summary.plsRbetamodel"

## Usage

```
## S3 method for class 'summary.plsRbetamodel'
print(x, ...)
```

## Arguments

| | |
|---|---|
| x | an object of the class "summary.plsRbetamodel" |
| ... | not used |

## Value

| | |
|---|---|
| language | call of the model |

## Author(s)

Frédéric Bertrand
<frederic.bertrand@utt.fr>
https://fbertran.github.io/homepage/

## References

Frédéric Bertrand, Nicolas Meyer, Michèle Beau-Faller, Karim El Bayed, Izzie-Jacques Namer, Myriam Maumy-Bertrand (2013). Régression Bêta PLS. *Journal de la Société Française de Statistique*, **154**(3):143-159. http://publications-sfds.math.cnrs.fr/index.php/J-SFdS/article/view/215

## See Also

print and summary

## Examples

```
data("GasolineYield",package="betareg")
modpls <- plsRbeta(yield~.,data=GasolineYield,nt=3,modele="pls-beta")
print(summary(modpls))
```

`simul_data_UniYX_beta` *Data generating function for univariate beta plsR models*

## Description

This function generates a single univariate rate response value $Y$ and a vector of explanatory variables $(X_1, \ldots, X_{totdim})$ drawn from a model with a given number of latent components.

## Usage

```
simul_data_UniYX_beta(
  totdim,
  ncomp,
  disp = 1,
  link = "logit",
  type = "a",
  phi0 = 20
)
```

## Arguments

| | |
|---|---|
| `totdim` | Number of columns of the X vector (from `ncomp` to hardware limits) |
| `ncomp` | Number of latent components in the model (from 2 to 6) |
| `disp` | Tune the shape of the beta distribution (defaults to 1) |
| `link` | Character specification of the link function in the mean model (mu). Currently, `"logit"`, `"probit"`, `"cloglog"`, `"cauchit"`, `"log"`, `"loglog"` are supported. Alternatively, an object of class "link-glm" can be supplied. |
| `type` | Simulation scheme |
| `phi0` | Simulation scheme "a" parameter |

## Details

This function should be combined with the replicate function to give rise to a larger dataset. The algorithm used is a modification of a port of the one described in the article of Li which is a multivariate generalization of the algorithm of Naes and Martens.

## Value

| | |
|---|---|
| `vector` | $(Y, X_1, \ldots, X_{totdim})$ |

## Author(s)

Frédéric Bertrand
<frederic.bertrand@utt.fr>
https://fbertran.github.io/homepage/

## References

Frédéric Bertrand, Nicolas Meyer, Michèle Beau-Faller, Karim El Bayed, Izzie-Jacques Namer, Myriam Maumy-Bertrand (2013). Régression Bêta PLS. *Journal de la Société Française de Statistique*, **154**(3):143-159. http://publications-sfds.math.cnrs.fr/index.php/J-SFdS/article/view/215

T. Naes, H. Martens (1985). Comparison of prediction methods for multicollinear data. *Commun. Stat., Simul.*, **14**:545-576. <doi:10.1080/03610918508812458>

Baibing Li, Julian Morris, Elaine B. Martin (2002). Model selection for partial least squares regression, *Chemometrics and Intelligent Laboratory Systems*, **64**:79-89. <doi:110.1016/S0169-7439(02)00051-5>

## See Also

simul_data_UniYX

## Examples

```
# logit link
layout(matrix(1:4,nrow=2))
hist(t(replicate(100,simul_data_UniYX_beta(4,4)))[,1])
hist(t(replicate(100,simul_data_UniYX_beta(4,4,disp=3)))[,1])
hist(t(replicate(100,simul_data_UniYX_beta(4,4,disp=5)))[,1])
hist(t(replicate(100,simul_data_UniYX_beta(4,4,disp=15)))[,1])
layout(1)

# probit link
layout(matrix(1:4,nrow=2))
hist(t(replicate(100,simul_data_UniYX_beta(4,4,link="probit")))[,1])
hist(t(replicate(100,simul_data_UniYX_beta(4,4,disp=3,link="probit")))[,1])
hist(t(replicate(100,simul_data_UniYX_beta(4,4,disp=5,link="probit")))[,1])
hist(t(replicate(100,simul_data_UniYX_beta(4,4,disp=15,link="probit")))[,1])
layout(1)

# cloglog link
layout(matrix(1:4,nrow=2))
hist(t(replicate(100,simul_data_UniYX_beta(4,4,link="cloglog")))[,1])
hist(t(replicate(100,simul_data_UniYX_beta(4,4,disp=3,link="cloglog")))[,1])
hist(t(replicate(100,simul_data_UniYX_beta(4,4,disp=5,link="cloglog")))[,1])
hist(t(replicate(100,simul_data_UniYX_beta(4,4,disp=15,link="cloglog")))[,1])
layout(1)

# cauchit link
layout(matrix(1:4,nrow=2))
hist(t(replicate(100,simul_data_UniYX_beta(4,4,link="cauchit")))[,1])
hist(t(replicate(100,simul_data_UniYX_beta(4,4,disp=3,link="cauchit")))[,1])
hist(t(replicate(100,simul_data_UniYX_beta(4,4,disp=5,link="cauchit")))[,1])
hist(t(replicate(100,simul_data_UniYX_beta(4,4,disp=15,link="cauchit")))[,1])
layout(1)

# loglog link
```

```
layout(matrix(1:4,nrow=2))
hist(t(replicate(100,simul_data_UniYX_beta(4,4,link="loglog")))[,1])
hist(t(replicate(100,simul_data_UniYX_beta(4,4,disp=3,link="loglog")))[,1])
hist(t(replicate(100,simul_data_UniYX_beta(4,4,disp=5,link="loglog")))[,1])
hist(t(replicate(100,simul_data_UniYX_beta(4,4,disp=15,link="loglog")))[,1])
layout(1)

# log link
layout(matrix(1:4,nrow=2))
hist(t(replicate(100,simul_data_UniYX_beta(4,4,link="log")))[,1])
hist(t(replicate(100,simul_data_UniYX_beta(4,4,disp=3,link="log")))[,1])
hist(t(replicate(100,simul_data_UniYX_beta(4,4,disp=5,link="log")))[,1])
hist(t(replicate(100,simul_data_UniYX_beta(4,4,disp=15,link="log")))[,1])
layout(1)
```

---

`summary.plsRbetamodel` *Summary method for plsRbeta models*

---

## Description

This function provides a summary method for the class "plsRbetamodel"

## Usage

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

## Arguments

| | |
|---|---|
| object | an object of the class "plsRbetamodel" |
| ... | further arguments to be passed to or from methods. |

## Value

| | |
|---|---|
| call | function call of plsR beta models |

## Author(s)

Frédéric Bertrand
<frederic.bertrand@utt.fr>
https://fbertran.github.io/homepage/

## References

Frédéric Bertrand, Nicolas Meyer, Michèle Beau-Faller, Karim El Bayed, Izzie-Jacques Namer, Myriam Maumy-Bertrand (2013). Régression Bêta PLS. *Journal de la Société Française de Statistique*, **154**(3):143-159. http://publications-sfds.math.cnrs.fr/index.php/J-SFdS/article/view/215

## See Also

summary

## Examples

```
data("GasolineYield",package="betareg")
modpls <- plsRbeta(yield~.,data=GasolineYield,nt=3,modele="pls-beta")
summary(modpls)
```

---

# tilt.bootplsbeta *Non-parametric tilted bootstrap for PLS beta regression models*

---

## Description

Provides a wrapper for the bootstrap function `tilt.boot` from the `boot` R package.
Implements non-parametric tilted bootstrap for PLS beta regression models by case resampling : the `tilt.boot` function will run an initial bootstrap with equal resampling probabilities (if required) and will use the output of the initial run to find resampling probabilities which put the value of the statistic at required values. It then runs an importance resampling bootstrap using the calculated probabilities as the resampling distribution.

## Usage

```
tilt.bootplsbeta(
  object,
  typeboot = "plsmodel",
  statistic = coefs.plsRbeta,
  R = c(499, 250, 250),
  alpha = c(0.025, 0.975),
  sim = "ordinary",
  stype = "i",
  index = 1,
  stabvalue = 1e+06
)
```

## Arguments

| | |
|---|---|
| `object` | An object of class `plsRbetamodel` to bootstrap |
| `typeboot` | The type of bootstrap. Either (Y,X) boostrap (`typeboot="plsmodel"`) or (Y,T) bootstrap (`typeboot="fmodel_np"`). Defaults to (Y,T) resampling. |
| `statistic` | A function which when applied to data returns a vector containing the statistic(s) of interest. `statistic` must take at least two arguments. The first argument passed will always be the original data. The second will be a vector of indices, frequencies or weights which define the bootstrap sample. Further, if predictions are required, then a third argument is required which would be a vector of the random indices used to generate the bootstrap predictions. Any further arguments can be passed to statistic through the ... argument. |

| | |
|---|---|
| R | The number of bootstrap replicates. Usually this will be a single positive integer. For importance resampling, some resamples may use one set of weights and others use a different set of weights. In this case R would be a vector of integers where each component gives the number of resamples from each of the rows of weights. |
| alpha | The alpha level to which tilting is required. This parameter is ignored if R[1] is 0 or if theta is supplied, otherwise it is used to find the values of theta as quantiles of the initial uniform bootstrap. In this case R[1] should be large enough that min(c(alpha, 1-alpha))*R[1] > 5, if this is not the case then a warning is generated to the effect that the theta are extreme values and so the tilted output may be unreliable. |
| sim | A character string indicating the type of simulation required. Possible values are "ordinary" (the default), "balanced", "permutation", or "antithetic". |
| stype | A character string indicating what the second argument of statistic represents. Possible values of stype are "i" (indices - the default), "f" (frequencies), or "w" (weights). |
| index | The index of the statistic of interest in the output from statistic. By default the first element of the output of statistic is used. |
| stabvalue | A value to hard threshold bootstrap estimates computed from atypical resamplings. |

## Value

An object of class "boot".

## Author(s)

Frédéric Bertrand
<frederic.bertrand@utt.fr>
https://fbertran.github.io/homepage/

## References

Frédéric Bertrand, Nicolas Meyer, Michèle Beau-Faller, Karim El Bayed, Izzie-Jacques Namer, Myriam Maumy-Bertrand (2013). Régression Bêta PLS. *Journal de la Société Française de Statistique*, **154**(3):143-159. http://publications-sfds.math.cnrs.fr/index.php/J-SFdS/article/view/215

## See Also

tilt.boot

## Examples

```
data("GasolineYield",package="betareg")
yGasolineYield <- GasolineYield$yield
XGasolineYield <- GasolineYield[,2:5]
modplsRbeta <- plsRbeta(yGasolineYield, XGasolineYield, nt=3,
```

```
modele="pls-beta")
# GazYield.tilt.boot <- tilt.bootplsbeta(modplsRbeta,
# statistic=coefs.plsRbeta, R=c(499, 100, 100),
# alpha=c(0.025, 0.975), sim="balanced", stype="i", index=1)
# boxplots.bootpls(GazYield.tilt.boot,1:2)
```

---

TxTum *Cancer infiltration rates*

---

## Description

This dataset features cancer infiltration rates and microsatellites data.

## Usage

```
TxTum
```

## Format

A data frame with 106 rows and 60 variables.

`CELTUMCO` a numeric vector

`age` a numeric vector

`sexe` a numeric vector

`HISTOADK` a numeric vector

`H2` a numeric vector

`P3` a numeric vector

`P4` a numeric vector

`E1` a numeric vector

`P5` a numeric vector

`R10` a numeric vector

`C3M` a numeric vector

`P6` a numeric vector

`RB` a numeric vector

`FL7A` a numeric vector

`P53` a numeric vector

`W2` a numeric vector

`P2` a numeric vector

`P1` a numeric vector

`W4` a numeric vector

MT1 a numeric vector
MT2 a numeric vector
MT4 a numeric vector
MT3 a numeric vector
HLA a numeric vector
HLD a numeric vector
HLC a numeric vector
HLB a numeric vector
EA1 a numeric vector
EA3 a numeric vector
EA2 a numeric vector
EA4 a numeric vector
EB1 a numeric vector
EB2 a numeric vector
EB3 a numeric vector
EB4 a numeric vector
EGF1 a numeric vector
EGF2 a numeric vector
EGF3 a numeric vector
EGF4 a numeric vector
EGF5 a numeric vector
EGF6 a numeric vector
FL7B a numeric vector
VSFGF7 a numeric vector
F3A a numeric vector
F3B a numeric vector
VSFGFR3 a numeric vector
F4 a numeric vector
Q5 a numeric vector
VSTOP1 a numeric vector
VSTOP2A a numeric vector
VSEGFR a numeric vector
AFRAEGFR a numeric vector
SRXRA a numeric vector
SMT a numeric vector
QMTAMPN a numeric vector
QMTDELN a numeric vector
SHL a numeric vector
SEA a numeric vector
SEB a numeric vector
QPCRFGF7 a numeric vector

## References

Régression Bêta PLS. (French) [PLS Beta regression.], F. Bertrand, N. Meyer, M. Beau-Faller, K. El Bayed, N. Izzie-J., M. Maumy-Bertrand, (2013), J. SFdS, 154(3):143-159

Partial Least Squares Regression for Beta Regression Models. F. Bertrand, M. Maumy (2021). useR! 2021, Zurich.

## Examples

```
data
print(TxTum)
summary(TxTum)
```

---

`TxTum.mod.bootBC1` *Bootstrap distribution TxTum BC1 model*

---

## Description

A precomputed bootstrap distribution of the coefficients of a model used in the vignette.

## Usage

```
TxTum.mod.bootBC1
```

## Format

a class boot object

## References

Régression Bêta PLS. (French) [PLS Beta regression.], F. Bertrand, N. Meyer, M. Beau-Faller, K. El Bayed, N. Izzie-J., M. Maumy-Bertrand, (2013), J. SFdS, 154(3):143-159

Partial Least Squares Regression for Beta Regression Models. F. Bertrand, M. Maumy (2021). useR! 2021, Zurich.

## Examples

```
data(TxTum.mod.bootBC1)
str(TxTum.mod.bootBC1)
plot(TxTum.mod.bootBC1)
```

`TxTum.mod.bootBR6` *Bootstrap distribution TxTum BR6 model*

## Description

A precomputed bootstrap distribution of the coefficients of a model used in the vignette.

## Usage

```
TxTum.mod.bootBR6
```

## Format

a class boot object

## References

Régression Bêta PLS. (French) [PLS Beta regression.], F. Bertrand, N. Meyer, M. Beau-Faller, K. El Bayed, N. Izzie-J., M. Maumy-Bertrand, (2013), J. SFdS, 154(3):143-159

Partial Least Squares Regression for Beta Regression Models. F. Bertrand, M. Maumy (2021). useR! 2021, Zurich.

## Examples

```
data(TxTum.mod.bootBR6)
str(TxTum.mod.bootBR6)
plot(TxTum.mod.bootBR6)
```

# Index