

Package: RmlxStats (via r-universe)

July 31, 2026

Title MLX-Accelerated Statistical Models

Version 0.3.0

Description Fast statistical routines on Apple Silicon using the Rmlx package.

License GPL (>= 2)

URL <https://hughjonesd.github.io/RmlxStats>,
<https://github.com/hughjonesd/RmlxStats>

BugReports <https://github.com/hughjonesd/RmlxStats/issues>

Imports Rmlx (>= 0.4.0), generics

Remotes hughjonesd/Rmlx

Suggests glmnet, knitr, lmtest, sandwich, testthat (>= 3.0.0), withr

Config/testthat/edition 3

Config/Needs/website knitr, rmarkdown, ggplot2, nycflights13, bench, boot, bigstatsr, car, fixest, irlba, RcppEigen, rsvd, speedglm, fastglm, lmboot, huxtable

Encoding UTF-8

Roxygen list(markdown = TRUE)

Config/roxygen2/version 8.0.0

Config/pak/sysreqs cmake

Repository <https://r-multiverse.r-universe.dev>

Date/Publication 2026-05-26 15:03:33 UTC

RemoteUrl <https://github.com/hughjonesd/RmlxStats>

RemoteRef v0.3.0

RemoteSha 36963b9c305b2487d6f93264c34d1c37bd0d5ea5

Contents

generics-reexports	2
mlxs-glm-methods	3
mlxs-lm-methods	5
mlxs-model-methods	8
mlxs-prcomp-methods	8
mlxs_binomial	9
mlxs_boot	10
mlxs_cv_glmnet	11
mlxs_gaussian	12
mlxs_glm	13
mlxs_glm_control	14
mlxs_glmnet	15
mlxs_lm	16
mlxs_lm_fit	17
mlxs_poisson	18
mlxs_prcomp	19
mlxs_quasibinomial	20
mlxs_quasipoisson	21
Index	22

generics-reexports	<i>Re-export generics</i>
--------------------	---------------------------

Description

These generics are re-exported from the generics package for convenience.

Usage

```
tidy(x, ...)  
  
glance(x, ...)  
  
augment(x, ...)
```

Arguments

x, ... Passed to the generic.

mlxs-glm-methods	<i>mlxs_glm method utilities</i>
------------------	----------------------------------

Description

Support functions that provide a familiar S3 surface for `mlxs_glm` fits by delegating to equivalent base `glm` behaviour where helpful.

Usage

```
## S3 method for class 'mlxs_glm'
weights(object, type = c("prior", "working"), ...)

## S3 method for class 'mlxs_glm'
predict(
  object,
  newdata = NULL,
  type = c("link", "response"),
  se.fit = FALSE,
  ...
)

## S3 method for class 'mlxs_glm'
fitted(object, ...)

## S3 method for class 'mlxs_glm'
residuals(object, type = c("deviance", "pearson", "working", "response"), ...)

## S3 method for class 'mlxs_glm'
vcov(object, ...)

## S3 method for class 'mlxs_glm'
confint(
  object,
  parm,
  level = 0.95,
  ...,
  bootstrap = FALSE,
  bootstrap_args = list(B = 200L, seed = NULL, progress = FALSE, bootstrap_type = "case")
)

## S3 method for class 'mlxs_glm'
print(x, digits = max(3, getOption("digits") - 3), ...)

## S3 method for class 'mlxs_glm'
summary(
  object,
```

```

    bootstrap = FALSE,
    bootstrap_args = list(B = 200L, seed = NULL, progress = FALSE, bootstrap_type = "case"),
    confint = FALSE,
    level = 0.95,
    ...
)

## S3 method for class 'summary.mlxs_glm'
print(x, digits = max(3, getOption("digits") - 3), ...)

## S3 method for class 'mlxs_glm'
anova(object, ...)

## S3 method for class 'mlxs_glm'
model.frame(formula, ...)

## S3 method for class 'mlxs_glm'
model.matrix(object, ...)

## S3 method for class 'mlxs_glm'
terms(x, ...)

## S3 method for class 'mlxs_glm'
nobs(object, ...)

## S3 method for class 'mlxs_glm'
tidy(x, ...)

## S3 method for class 'mlxs_glm'
glance(x, ...)

## S3 method for class 'mlxs_glm'
augment(
  x,
  data = x$model,
  newdata = NULL,
  type.predict = c("response", "link"),
  type.residuals = c("response", "deviance"),
  se_fit = FALSE,
  output = c("data.frame", "mlx"),
  ...
)

## S3 method for class 'mlxs_glm'
estfun(x, ..., output = c("matrix", "mlx"))

## S3 method for class 'mlxs_glm'
hatvalues(model, ..., output = c("matrix", "mlx"))

```

```
## S3 method for class 'mlxs_glm'
bread(x, ...)
```

Arguments

<code>object, model</code>	An <code>mlxs_glm</code> model fit.
<code>type</code>	Character string indicating the scale of the prediction or residuals to return.
<code>...</code>	Additional arguments passed to underlying methods.
<code>newdata</code>	Optional data frame used for prediction.
<code>se.fit</code>	Logical. Should standard errors of the fit be returned when supported?
<code>parm</code>	Parameter specification for confidence intervals.
<code>level</code>	Confidence level for intervals.
<code>bootstrap</code>	Logical; should bootstrap standard errors or confidence intervals be computed?
<code>bootstrap_args</code>	List of bootstrap configuration options. See mlxs_boot() .
<code>x</code>	An <code>mlxs_glm</code> model fit (for methods with a leading <code>x</code> argument).
<code>digits</code>	Number of significant digits to print for summaries.
<code>confint</code>	Logical; should confidence intervals be included in the summary object?
<code>formula, data</code>	Optional formula and data overrides used by <code>augment.mlxs_glm()</code> .
<code>type.predict, type.residuals</code>	Character strings controlling the scale of fitted values and residuals returned by <code>augment.mlxs_glm()</code> .
<code>se_fit</code>	Logical; standard-error analogue for <code>augment</code> .
<code>output</code>	Character string; return format ("data.frame" or "mlx").

<code>mlxs-lm-methods</code>	<i>mlxs_lm method utilities</i>
------------------------------	---------------------------------

Description

These helpers provide the familiar S3 surface for `mlxs_lm` fits.

Usage

```
## S3 method for class 'mlxs_lm'
predict(object, newdata = NULL, ...)

## S3 method for class 'mlxs_lm'
fitted(object, ...)

## S3 method for class 'mlxs_lm'
residuals(object, ...)
```

```
## S3 method for class 'mlxs_lm'
vcov(object, ...)

## S3 method for class 'mlxs_lm'
confint(
  object,
  parm,
  level = 0.95,
  ...,
  bootstrap = FALSE,
  bootstrap_args = list(B = 200L, seed = NULL, progress = FALSE, bootstrap_type = "case")
)

## S3 method for class 'mlxs_lm'
anova(object, ...)

## S3 method for class 'mlxs_anova'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)

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

## S3 method for class 'mlxs_anova'
tidy(x, ...)

## S3 method for class 'mlxs_lm'
summary(
  object,
  bootstrap = FALSE,
  bootstrap_args = list(B = 200L, seed = NULL, progress = FALSE, bootstrap_type = "case"),
  confint = FALSE,
  level = 0.95,
  ...
)

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

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

## S3 method for class 'mlxs_lm'
model.frame(formula, ...)

## S3 method for class 'mlxs_lm'
model.matrix(object, ...)

## S3 method for class 'mlxs_lm'
```

```

terms(x, ...)

## S3 method for class 'mlxs_lm'
nobs(object, ...)

## S3 method for class 'mlxs_lm'
tidy(x, ...)

## S3 method for class 'mlxs_lm'
glance(x, ...)

## S3 method for class 'mlxs_lm'
augment(
  x,
  data = model.frame(x),
  newdata = NULL,
  se_fit = FALSE,
  output = c("data.frame", "mlx"),
  ...
)

## S3 method for class 'mlxs_lm'
estfun(x, ..., output = c("matrix", "mlx"))

## S3 method for class 'mlxs_lm'
hatvalues(model, ..., output = c("matrix", "mlx"))

## S3 method for class 'mlxs_lm'
bread(x, ...)

```

Arguments

<code>object, model</code>	An <code>mlxs_lm</code> model fit.
<code>newdata</code>	Optional data frame for prediction.
<code>...</code>	Additional arguments passed to underlying methods.
<code>parm</code>	Parameter specification for confidence intervals.
<code>level</code>	Confidence level for intervals.
<code>bootstrap</code>	Logical; should bootstrap standard errors or confidence intervals be computed?
<code>bootstrap_args</code>	List of bootstrap configuration options. See mlxs_boot() .
<code>x</code>	An <code>mlxs_lm</code> model fit (for methods with a leading <code>x</code> argument).
<code>row.names</code>	Optional row names for data frame conversion.
<code>optional</code>	Logical; passed to <code>as.data.frame</code> .
<code>confint</code>	Logical; should confidence intervals be included in the summary object?
<code>formula</code>	An <code>mlxs_lm</code> object used in place of formula for <code>model.frame</code> .
<code>data</code>	Optional data frame for <code>augment</code> .

<code>se_fit</code>	Logical; should standard errors of fit be included?
<code>output</code>	Character string; return format ("data.frame", "matrix", "vector" or "mlx"). To make methods from other packages work, the usual default is to return a base R object.

mlxs-model-methods	<i>Shared mlxs model methods</i>
--------------------	----------------------------------

Description

Methods for behavior shared by `mlxs_lm` and `mlxs_glm` through their `mlxs_model` superclass.

Usage

```
## S3 method for class 'mlxs_model'
update(object, ..., evaluate = TRUE)

## S3 method for class 'mlxs_model'
coef(object, ..., output = c("vector", "mlx"))
```

Arguments

<code>object</code>	An <code>mlxs_model</code> fit.
<code>...</code>	Additional arguments passed to underlying methods.
<code>evaluate</code>	Logical; evaluate the updated call?

mlxs-prcomp-methods	<i>PCA methods for mlxs_prcomp</i>
---------------------	------------------------------------

Description

`predict.mlxs_prcomp()` returns MLX scores. `summary()` and `plot()` only materialize standard deviations for base-style output; `print()` and `biplot()` materialize rotations and scores as needed for display.

Usage

```
## S3 method for class 'mlxs_prcomp'
predict(object, newdata, ...)

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

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

```
## S3 method for class 'mlxs_prcomp'
plot(x, main = deparse1(substitute(x)), ...)

## S3 method for class 'mlxs_prcomp'
biplot(x, ...)

## S3 method for class 'mlxs_prcomp'
nobs(object, ...)

## S3 method for class 'mlxs_prcomp'
tidy(x, ...)

## S3 method for class 'mlxs_prcomp'
augment(x, data = NULL, newdata = NULL, output = c("data.frame", "mlx"), ...)
```

Arguments

<code>object, x</code>	A fitted <code>mlxs_prcomp</code> object.
<code>newdata</code>	Optional new observations to project.
<code>...</code>	Passed through to the corresponding base method.
<code>data</code>	Optional original data to append PCA scores to in <code>augment.mlxs_prcomp()</code> .
<code>output</code>	Output format for <code>augment.mlxs_prcomp()</code> : either a data frame with appended score columns or the MLX score matrix directly.

Value

Method-specific output. `predict.mlxs_prcomp()` returns an MLX matrix. `augment.mlxs_prcomp()` returns either a data frame or MLX matrix.

<code>mlxs_binomial</code>	<i>MLX-friendly binomial family</i>
----------------------------	-------------------------------------

Description

Construct a binomial GLM family whose core link and deviance helpers are implemented in R so they work with MLX arrays as well as base R vectors. This avoids calling into compiled C routines that only handle base types.

Usage

```
mlxs_binomial(link = "logit")
```

Arguments

link a specification for the model link function. This can be a name/expression, a literal character string, a length-one character vector, or an object of class `"link-glm"` (such as generated by `make.link`) provided it is not specified *via* one of the standard names given next.

The gaussian family accepts the links (as names) `identity`, `log` and `inverse`; the binomial family the links `logit`, `probit`, `cauchit`, (corresponding to logistic, normal and Cauchy CDFs respectively) `log`, `identity` and `cloglog` (complementary log-log); the Gamma family the links `inverse`, `identity` and `log`; the poisson family the links `log`, `identity`, and `sqrt`; and the `inverse.gaussian` family 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`, and the function `power` can be used to create a power link function.

Details

Currently the `logit`, `log`, `cloglog`, and `cauchit` links are supported. For other link specifications, fall back to `stats::binomial()`.

Value

A family object compatible with `stats::glm()` and `mlxs_glm()`.

mlxs_boot

Bootstrap MLX arrays along the first dimension

Description

`mlxs_boot()` resamples observations from one or more MLX arrays, calls a user-supplied function on each resampled batch, and returns the collected results. Every argument supplied via `...` must share the same size in its first dimension (number of observations). Arguments that do not need resampling should be captured in the environment of `fun` instead of being passed through `...`.

Usage

```
mlxs_boot(fun, ..., B = 200L, seed = NULL, progress = FALSE, compile = FALSE)
```

Arguments

fun Function called on each bootstrap draw. It must accept the same named arguments as supplied through `...`.

... Arrays, matrices, or vectors that should be resampled along the first dimension before being passed to `fun`.

B Number of bootstrap iterations.

seed Optional integer seed for reproducibility.

progress	Logical; if TRUE, show a text progress bar.
compile	Logical; compile fun once via <code>Rmlx::mlx_compile()</code> before entering the re-sampling loop. Defaults to FALSE.

Value

A list with elements `samples` (the raw results from fun), `B`, and `seed`.

mlxs_cv_glmnet	<i>Cross-validated MLX elastic net regression</i>
----------------	---

Description

Cross-validation wrapper around `mlxs_glmnet()` that mirrors the core `glmnet::cv.glmnet()` workflow for the families currently supported by `mlxs_glmnet()`.

Usage

```
mlxs_cv_glmnet(
  x,
  y,
  weights = NULL,
  offset = NULL,
  lambda = NULL,
  type.measure = c("default", "mse", "deviance", "class", "mae", "auc", "C"),
  nfolds = 10,
  foldid = NULL,
  alignment = c("lambda", "fraction"),
  grouped = TRUE,
  keep = FALSE,
  parallel = FALSE,
  gamma = c(0, 0.25, 0.5, 0.75, 1),
  relax = FALSE,
  trace.it = 0,
  family = mlxs_gaussian(),
  ...
)
```

Arguments

x	Numeric matrix of predictors (observations in rows).
y	Numeric response vector.
weights	Optional observation weights. Currently unsupported.
offset	Optional offset. Currently unsupported.
lambda	Optional decreasing lambda sequence. If NULL, the full-data fit chooses the path and the same path is reused inside each fold.

<code>type.measure</code>	Loss used to score the holdout predictions.
<code>nfolds</code>	Number of folds.
<code>foldid</code>	Optional integer vector giving the fold assignment for each observation.
<code>alignment</code>	Alignment mode. Only "lambda" is currently supported.
<code>grouped</code>	Should cross-validation be aggregated fold-by-fold? Only TRUE is currently supported.
<code>keep</code>	Should out-of-fold predictions be stored?
<code>parallel</code>	Logical. Parallel refits are currently unsupported.
<code>gamma, relax</code>	Relaxed fits are currently unsupported.
<code>trace.it</code>	Progress tracing. Currently unsupported.
<code>family</code>	MLX-aware family object, e.g. <code>mlxs_gaussian()</code> or <code>mlxs_binomial()</code> .
<code>...</code>	Additional arguments passed to <code>mlxs_glmnet()</code> , such as <code>alpha</code> , <code>nlambda</code> , <code>lambda_min_ratio</code> , <code>standardize</code> , <code>intercept</code> , <code>maxit</code> , and <code>tol</code> .

Details

The full-data fit defines a master lambda path. Each fold is then refit on the same lambda values and scored on its holdout set.

Current limitations relative to `glmnet::cv.glmnet()`:

- only Gaussian and binomial families are supported
- `weights`, `offset`, `alignment != "lambda"`, `grouped = FALSE`, `parallel = TRUE`, `relax = TRUE`, and non-zero `trace.it` are not implemented
- `type.measure = "auc"` and `type.measure = "C"` are not implemented

Value

An object of class `mlxs_cv_glmnet`.

<code>mlxs_gaussian</code>	<i>MLX-friendly Gaussian family</i>
----------------------------	-------------------------------------

Description

MLX-friendly Gaussian family

Usage

```
mlxs_gaussian(link = "identity")
```

Arguments

link a specification for the model link function. This can be a name/expression, a literal character string, a length-one character vector, or an object of class `"link-glm"` (such as generated by `make.link`) provided it is not specified *via* one of the standard names given next.

The gaussian family accepts the links (as names) identity, log and inverse; the binomial family the links logit, probit, cauchit, (corresponding to logistic, normal and Cauchy CDFs respectively) log, identity and cloglog (complementary log-log); the Gamma family the links inverse, identity and log; the poisson family the links log, identity, and sqrt; and the inverse.gaussian family 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, and the function `power` can be used to create a power link function.

Value

A family object compatible with `mlxs_glm()`.

<code>mlxs_glm</code>	<i>MLX-backed generalized linear model</i>
-----------------------	--

Description

Fit generalized linear models using iterative reweighted least squares (IRLS) with MLX providing the heavy lifting for weighted least squares solves. Final convergence is done at double precision on the cpu.

Usage

```
mlxs_glm(
  formula,
  family = mlxs_gaussian(),
  data,
  subset,
  weights,
  na.action = stats::na.exclude,
  start = NULL,
  control = list(),
  ...
)
```

Arguments

formula Model formula.

family A mlxs family object (e.g., `mlxs_gaussian()`, `mlxs_binomial()`, `mlxs_poisson()`). You can use "gaussian" etc.

data	Optional data frame, tibble, or environment containing the variables in the model.
subset	Optional expression for subsetting observations.
weights	Optional non-negative observation weights.
na.action	How to handle missing values.
start	Starting values for the parameters in the linear predictor.
control	Optional list of control parameters passed to <code>mlxs_glm_control()</code> . Control parameters can include <code>epsilon</code> , <code>epsilon_f64</code> , <code>maxit</code> , <code>trace</code> , and <code>rank_tol</code> .
...	Additional arguments passed to the family function when family is supplied as a function or string.

Value

An object of class `c("mlxs_glm", "mlxs_model")` containing elements similar to the result of `stats::glm()`. Unlike `stats::glm()`, rank-deficient model matrices are rejected rather than fit with aliased coefficients.

Examples

```
fit <- mlxs_glm(mpg ~ cyl + disp, family = mlxs_gaussian(), data = mtcars)
coef(fit)
```

mlxs_glm_control	<i>Control parameters</i>
------------------	---------------------------

Description

Control parameters

Usage

```
mlxs_glm_control(
  epsilon = 1e-08,
  epsilon_f64 = 1e-06,
  maxit = 25,
  trace = FALSE,
  rank_tol = NULL
)
```

Arguments

epsilon	Convergence tolerance parameter, interpreted as in <code>stats::glm.control()</code> . Iterations converge when $\text{abs}(\text{deviance} - \text{deviance_old}) / (\text{abs}(\text{deviance}) + 0.1) < \text{epsilon}$.
epsilon_f64	Move operations to float64 on the cpu when convergence is this close (using the same expression as above). Doing this allows more precision but slows computation.

maxit	Maximum number of IWLS iterations.
trace	Logical: trace each iteration?
rank_tol	Optional relative tolerance used to detect rank-deficient systems. NULL uses the package default, which varies by dtype and is 1e-6 for float32 matrices. Set to FALSE to skip rank checks entirely. Note that higher numbers indicate <i>lower</i> tolerance.

Value

A list with default values filled in.

mlxs_glmnet	<i>MLX-backed elastic net regression</i>
-------------	--

Description

Fit lasso or elastic-net penalised regression paths using MLX arrays for the heavy linear algebra. Dense Gaussian and binomial paths stay on the MLX backend throughout the iterative updates, with repeated chunk updates traced through `Rmlx::mlx_compile()` to reduce host overhead.

Usage

```
mlxs_glmnet(
  x,
  y,
  family = mlxs_gaussian(),
  alpha = 1,
  lambda = NULL,
  nlambda = 100,
  lambda_min_ratio = 1e-04,
  standardize = TRUE,
  intercept = TRUE,
  use_strong_rules = TRUE,
  maxit = 1000,
  tol = 1e-06
)
```

Arguments

x	Numeric R or mlx matrix of predictors.
y	Numeric R or mlx response vector.
family	MLX-aware family object, e.g. <code>mlxs_gaussian()</code> or <code>mlxs_binomial()</code> .
alpha	Elastic-net mixing parameter (1 = lasso, currently alpha must be strictly positive).
lambda	Optional decreasing sequence of penalty values. If NULL, a sequence of length nlambda is generated from lambda_max down to lambda_max * lambda_min_ratio.

nlambda	Length of automatically generated lambda path.
lambda_min_ratio	Smallest lambda as a fraction of lambda_max.
standardize	Should columns of x be scaled before fitting?
intercept	Should an intercept be fit?
use_strong_rules	Retained for API compatibility. The dense MLX solver keeps all coefficients on device, so this flag currently does not change the computation.
maxit	Maximum proximal-gradient iterations per lambda value.
tol	Convergence tolerance on the coefficient updates.

Value

An object of class `mlxs_glmnet` containing the fitted coefficient path, intercepts, lambda sequence, and scaling information. The `beta` and `a0` components are stored as MLX arrays so downstream methods can keep path computations on the MLX backend. Use `coef()` or `predict()` for ordinary base-R matrix outputs.

Note

`glmnet::glmnet()` is faster on smaller problems. Very roughly as of April 2026, `mlxs_glmnet()` gets competitive at $n \times p = 5,000,000$ or greater.

mlxs_lm	<i>MLX-backed linear regression</i>
---------	-------------------------------------

Description

Fit a linear model via QR decomposition using MLX arrays on Apple Silicon devices. The interface mirrors `stats::lm()` for the common arguments.

Usage

```
mlxs_lm(  
  formula,  
  data,  
  subset,  
  weights,  
  na.action = stats::na.exclude,  
  rank_tol = NULL  
)
```

Arguments

formula	Model formula.
data	Optional data frame, tibble, or environment containing the variables in the model.
subset	Optional expression for subsetting observations.
weights	Optional non-negative observation weights.
na.action	How to handle missing values.
rank_tol	Optional relative tolerance used to detect rank-deficient systems. NULL uses the package default, which varies by dtype and is 1e-6 for float32 matrices. Set to FALSE to skip rank checks entirely. Note that higher numbers indicate <i>lower</i> tolerance.

Value

An object of class `c("mlxs_lm", "mlxs_model")` containing components similar to an `"lm"` fit, along with MLX intermediates stored in the `mlx` element. Note that MLX currently operates in single precision, so fitted values and diagnostics may differ from `stats::lm()` at around the 1e-6 level. Unlike `stats::lm()`, rank-deficient model matrices are rejected rather than fit with aliased coefficients.

Examples

```
fit <- mlxs_lm(mpg ~ cyl + disp, data = mtcars)
coef(fit)
```

mlxs_lm_fit

Fit an MLX linear model from design matrices

Description

`mlxs_lm_fit()` powers `mlxs_lm()` by wrapping the QR-based solver that runs entirely on MLX arrays.

Usage

```
mlxs_lm_fit(x, y, weights = NULL, rank_tol = NULL)
```

Arguments

x	MLX design matrix (or object coercible via <code>Rmlx::as_mlx()</code>) whose rows represent observations and columns represent predictors.
y	MLX column vector (or object coercible via <code>Rmlx::as_mlx()</code>) holding the response values.
weights	Optional MLX column vector or numeric vector of non-negative observation weights. When supplied, weighted least squares are fit via the standard square-root weighting.

`rank_tol` Optional relative tolerance used to detect rank-deficient systems. NULL uses the package default, which varies by dtype and is 1e-6 for float32 matrices. Set to FALSE to skip rank checks entirely. Note that higher numbers indicate *lower* tolerance.

Details

Inputs that are not already MLX objects are converted with `Rmlx::as_mlx()` or `Rmlx::mlx_matrix()` so callers can provide base-R matrices or vectors. Weighted fits are performed by applying the standard square-root weight transform before solving the QR system.

Value

A list with components `coefficients`, `fitted.values`, `residuals`, `effects`, and `qr`, mirroring the corresponding pieces of `stats::lm()`. Array-valued components remain MLX matrices to keep downstream GPU pipelines in device memory.

Examples

```
x <- Rmlx::as_mlx(cbind(1, as.matrix(mtcars[c("cyl", "disp")])))
y <- Rmlx::mlx_matrix(mtcars$mpg, ncol = 1)
fit <- mlxs_lm_fit(x, y)
drop(as.matrix(fit$coefficients))
```

mlxs_poisson

MLX-friendly Poisson family

Description

MLX-friendly Poisson family

Usage

```
mlxs_poisson(link = "log")
```

Arguments

`link` a specification for the model link function. This can be a name/expression, a literal character string, a length-one character vector, or an object of class `"link-glm"` (such as generated by `make.link`) provided it is not specified *via* one of the standard names given next.

The gaussian family accepts the links (as names) `identity`, `log` and `inverse`; the binomial family the links `logit`, `probit`, `cauchit`, (corresponding to logistic, normal and Cauchy CDFs respectively) `log`, `identity` and `cloglog` (complementary log-log); the Gamma family the links `inverse`, `identity` and `log`; the poisson family the links `log`, `identity`, and `sqrt`; and the `inverse.gaussian` family 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`, and the function `power` can be used to create a power link function.

Value

A family object compatible with `mlxs_glm()`.

`mlxs_prcomp`

MLX-backed principal components analysis

Description

Perform principal components analysis with MLX arrays, keeping the centred and scaled data on device throughout the decomposition.

Usage

```
mlxs_prcomp(
  x,
  retx = TRUE,
  center = TRUE,
  scale. = FALSE,
  tol = NULL,
  rank. = NULL,
  oversample = NULL,
  n_iter = 2L,
  seed = 1L,
  ...
)
```

Arguments

<code>x</code>	Numeric matrix-like object or MLX array with observations in rows.
<code>retx</code>	Should the rotated scores be returned?
<code>center</code> , <code>scale.</code>	Passed to <code>base::scale()</code> . User-supplied vectors are supported.
<code>tol</code>	Optional tolerance for omitting components with small standard deviations, relative to the leading component.
<code>rank.</code>	Optional maximal rank. If smaller than $\min(n, p)$, the fit uses the randomized truncated PCA path.
<code>oversample</code>	Oversampling added to the randomized subspace dimension. If <code>NULL</code> , randomized fits use $\min(\text{rank.}, \max(10, \text{ceiling}(\text{rank.} / 2)))$. Ignored for exact fits.
<code>n_iter</code>	Number of randomized power iterations. Ignored for exact fits.
<code>seed</code>	Seed used for the randomized projection basis. Ignored for exact fits.
<code>...</code>	Additional arguments are rejected for compatibility with <code>stats::prcomp()</code> .

Details

The interface follows `stats::prcomp()` closely. Full-rank fits use an exact decomposition. When `rank.` is supplied and smaller than `min(nrow(x), ncol(x))`, a randomized truncated PCA path is used instead.

The randomized path first sketches a slightly larger subspace than the requested rank, then compresses back down to the requested components. The `oversample` parameter controls how much extra space is used in that sketch: larger values make it less likely that the random sketch misses part of the leading principal subspace. The `n_iter` parameter applies additional power iterations, which improve accuracy when the singular values decay slowly but require extra passes over the matrix.

By default, `oversample` is chosen as `min(rank., max(10, ceiling(rank. / 2)))`, which keeps the usual constant-size oversampling for small target ranks while allowing more slack for larger truncated fits. This follows common randomized SVD guidance to start with modest oversampling, often around 5 to 10, and to increase oversampling before increasing the number of power iterations.

Value

An object of class `c("mlxs_prcomp", "prcomp")`.

References

- Halko, N., Martinsson, P.-G., and Tropp, J. A. (2011). Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions. *SIAM Review*, 53(2), 217-288.
- Musco, C. and Musco, C. (2015). Randomized Block Krylov Methods for Stronger and Faster Approximate Singular Value Decomposition. *NeurIPS 2015*.

mlxs_quasibinomial	<i>MLX-friendly quasibinomial family</i>
--------------------	--

Description

MLX-friendly quasibinomial family

Usage

```
mlxs_quasibinomial(link = "logit")
```

Arguments

link	a specification for the model link function. This can be a name/expression, a literal character string, a length-one character vector, or an object of class <code>"link-glm"</code> (such as generated by <code>make.link</code>) provided it is not specified <i>via</i> one of the standard names given next.
------	---

The gaussian family accepts the links (as names) identity, log and inverse; the binomial family the links logit, probit, cauchit, (corresponding to logistic, normal and Cauchy CDFs respectively) log, identity and cloglog (complementary log-log); the Gamma family the links inverse, identity and log; the poisson family the links log, identity, and sqrt; and the inverse.gaussian family 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, and the function [power](#) can be used to create a power link function.

Value

A family object compatible with `mlxs_glm()`.

mlxs_quasipoisson	<i>MLX-friendly quasipoisson family</i>
-------------------	---

Description

MLX-friendly quasipoisson family

Usage

```
mlxs_quasipoisson(link = "log")
```

Arguments

link	a specification for the model link function. This can be a name/expression, a literal character string, a length-one character vector, or an object of class "link-glm" (such as generated by make.link) provided it is not specified <i>via</i> one of the standard names given next. The gaussian family accepts the links (as names) identity, log and inverse; the binomial family the links logit, probit, cauchit, (corresponding to logistic, normal and Cauchy CDFs respectively) log, identity and cloglog (complementary log-log); the Gamma family the links inverse, identity and log; the poisson family the links log, identity, and sqrt; and the inverse.gaussian family 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, and the function power can be used to create a power link function.
------	--

Value

A family object compatible with `mlxs_glm()`.

Index

`anova.mlx_glm` (`mlx-glm-methods`), 3
`anova.mlx_lm` (`mlx-lm-methods`), 5
`as.data.frame.mlx_anova`
 (`mlx-lm-methods`), 5
`augment` (`generics-reexports`), 2
`augment.mlx_glm` (`mlx-glm-methods`), 3
`augment.mlx_lm` (`mlx-lm-methods`), 5
`augment.mlx_prcomp`
 (`mlx-prcomp-methods`), 8

`base::scale()`, 19
`biplot.mlx_prcomp`
 (`mlx-prcomp-methods`), 8
`bread.mlx_glm` (`mlx-glm-methods`), 3
`bread.mlx_lm` (`mlx-lm-methods`), 5

`coef()`, 16
`coef.mlx_model` (`mlx-model-methods`), 8
`confint.mlx_glm` (`mlx-glm-methods`), 3
`confint.mlx_lm` (`mlx-lm-methods`), 5

`estfun.mlx_glm` (`mlx-glm-methods`), 3
`estfun.mlx_lm` (`mlx-lm-methods`), 5

`fitted.mlx_glm` (`mlx-glm-methods`), 3
`fitted.mlx_lm` (`mlx-lm-methods`), 5

`generics-reexports`, 2
`glance` (`generics-reexports`), 2
`glance.mlx_glm` (`mlx-glm-methods`), 3
`glance.mlx_lm` (`mlx-lm-methods`), 5

`hatvalues.mlx_glm` (`mlx-glm-methods`), 3
`hatvalues.mlx_lm` (`mlx-lm-methods`), 5

`make.link`, 10, 13, 18, 20, 21
`mlx-glm-methods`, 3
`mlx-lm-methods`, 5
`mlx-model-methods`, 8
`mlx-prcomp-methods`, 8
`mlx_binomial`, 9
`mlx_binomial()`, 12, 13, 15
`mlx_boot`, 10
`mlx_boot()`, 5, 7
`mlx_cv_glmnet`, 11
`mlx_gaussian`, 12
`mlx_gaussian()`, 12, 13, 15
`mlx_glm`, 13
`mlx_glm_control`, 14
`mlx_glm_control()`, 14
`mlx_glmnet`, 15
`mlx_glmnet()`, 11, 12
`mlx_lm`, 16
`mlx_lm()`, 17
`mlx_lm_fit`, 17
`mlx_poisson`, 18
`mlx_poisson()`, 13
`mlx_prcomp`, 19
`mlx_quasibinomial`, 20
`mlx_quasipoisson`, 21
`model.frame.mlx_glm`
 (`mlx-glm-methods`), 3
`model.frame.mlx_lm` (`mlx-lm-methods`), 5
`model.matrix.mlx_glm`
 (`mlx-glm-methods`), 3
`model.matrix.mlx_lm` (`mlx-lm-methods`), 5

`nobs.mlx_glm` (`mlx-glm-methods`), 3
`nobs.mlx_lm` (`mlx-lm-methods`), 5
`nobs.mlx_prcomp` (`mlx-prcomp-methods`), 8

`plot.mlx_prcomp` (`mlx-prcomp-methods`), 8
`power`, 10, 13, 19, 21
`predict()`, 16
`predict.mlx_glm` (`mlx-glm-methods`), 3
`predict.mlx_lm` (`mlx-lm-methods`), 5
`predict.mlx_prcomp`
 (`mlx-prcomp-methods`), 8

`print.mlx_anova` (`mlxs-lm-methods`), 5
`print.mlx_glm` (`mlxs-glm-methods`), 3
`print.mlx_lm` (`mlxs-lm-methods`), 5
`print.mlx_prcomp`
 (`mlxs-prcomp-methods`), 8
`print.summary.mlx_glm`
 (`mlxs-glm-methods`), 3
`print.summary.mlx_lm`
 (`mlxs-lm-methods`), 5

`residuals.mlx_glm` (`mlxs-glm-methods`), 3
`residuals.mlx_lm` (`mlxs-lm-methods`), 5
`Rmlx::as_mlx()`, 17, 18
`Rmlx::mlx_compile()`, 11, 15
`Rmlx::mlx_matrix()`, 18

`stats::binomial()`, 10
`stats::glm()`, 10, 14
`stats::glm.control()`, 14
`stats::lm()`, 16–18
`stats::prcomp()`, 19, 20
`summary.mlx_glm` (`mlxs-glm-methods`), 3
`summary.mlx_lm` (`mlxs-lm-methods`), 5
`summary.mlx_prcomp`
 (`mlxs-prcomp-methods`), 8

`terms.mlx_glm` (`mlxs-glm-methods`), 3
`terms.mlx_lm` (`mlxs-lm-methods`), 5
`tidy` (`generics-reexports`), 2
`tidy.mlx_anova` (`mlxs-lm-methods`), 5
`tidy.mlx_glm` (`mlxs-glm-methods`), 3
`tidy.mlx_lm` (`mlxs-lm-methods`), 5
`tidy.mlx_prcomp` (`mlxs-prcomp-methods`),
 8

`update.mlx_model` (`mlxs-model-methods`),
 8

`vcov.mlx_glm` (`mlxs-glm-methods`), 3
`vcov.mlx_lm` (`mlxs-lm-methods`), 5

`weights.mlx_glm` (`mlxs-glm-methods`), 3