A short summary of the available families is given in the following paragraphs:

`AdaExp()`, `Binomial()` and `AUC()` implement families for binary classification. `AdaExp()` uses the exponential loss, which essentially leads to the AdaBoost algorithm of Freund and Schapire (1996). `Binomial()` implements the negative binomial log-likelihood of a logistic regression model as loss function. Thus, using `Binomial` family closely corresponds to fitting a logistic model. Alternative link functions can be specified.

However, the coefficients resulting from boosting with family `Binomial(link = "logit")` are $1/2$ of the coefficients of a logit model obtained via `glm`. Buehlmann and Hothorn (2007) argue that the family `Binomial` is the preferred choice for binary classification. For binary classification problems the response `y` has to be a `factor`. Internally `y` is re-coded to -1 and $+1$ (Buehlmann and Hothorn 2007).

`Binomial(type = "glm")` is an alternative to `Binomial()` leading to coefficients of the same size as coefficients from a classical logit model via `glm`. Additionally, it works not only with a two-level factor but also with a two-column matrix containing the number of successes and number of failures (again, similar to `glm`).

`AUC()` uses $1-AUC(y, f)$ as the loss function. The area under the ROC curve (AUC) is defined as $AUC = (n_{-1} n_1)^{-1} \sum_{i: y_i = 1} \sum_{j: y_j = -1} I(f_i > f_j)$. Since this is not differentiable in `f`, we approximate the jump function $I((f_i - f_j) > 0)$ by the distribution function of the triangular distribution on $[-1, 1]$ with mean 0, similar to the logistic distribution approximation used in Ma and Huang (2005).

`Gaussian()` is the default family in `mboost`. It implements L_2Boosting for continuous response. Note that families `GaussReg()` and `GaussClass()` (for regression and classification) are deprecated now. `Huber()` implements a robust version for boosting with continuous response, where the Huber-loss is used. `Laplace()` implements another strategy for continuous outcomes and uses the L_1-loss instead of the L_2-loss as used by `Gaussian()`.

`Poisson()` implements a family for fitting count data with boosting methods. The implemented loss function is the negative Poisson log-likelihood. Note that the natural link function $\log(\mu) = \eta$ is assumed. The default step-site `nu = 0.1` is probably too large for this family (leading to infinite residuals) and smaller values are more appropriate.

`GammaReg()` implements a family for fitting nonnegative response variables. The implemented loss function is the negative Gamma log-likelihood with logarithmic link function (instead of the natural link).

`CoxPH()` implements the negative partial log-likelihood for Cox models. Hence, survival models can be boosted using this family.

`QuantReg()` implements boosting for quantile regression, which is introduced in Fenske et al. (2009). `ExpectReg` works in analogy, only for expectiles, which were introduced to regression by Newey and Powell (1987).

Families with an additional scale parameter can be used for fitting models as well: `PropOdds()` leads to proportional odds models for ordinal outcome variables (Schmid et al., 2011). When using this family, an ordered set of threshold parameters is re-estimated in each boosting iteration. An example is given below which also shows how to obtain the thresholds. `NBinomial()` leads to regression models with a negative binomial conditional distribution of the response. `Weibull()`, `Loglog()`, and `Lognormal()` implement the negative log-likelihood functions of accelerated failure time models with Weibull, log-logistic, and lognormal distributed outcomes, respectively. Hence, parametric survival models can be boosted using these families. For details see Schmid and Hothorn (2008) and Schmid et al. (2010).

`Gehan()` implements rank-based estimation of survival data in an accelerated failure time model. The loss function is defined as the sum of the pairwise absolute differences of residuals. The response needs to be defined as `Surv(y, delta)`, where `y` is the observed survial time (subject to censoring) and `delta` is the non-censoring indicator (see `Surv` for details). For details on `Gehan()` see Johnson and Long (2011).

`Cindex()` optimizes the concordance-index for survival data (often denoted as Harrell's C or C-index). The concordance index evaluates the rank-based concordance probability between the model and the outcome. The C-index measures whether large values of the model are associated with short survival times and vice versa. The interpretation is similar to the AUC: A C-index of 1 represents a perfect discrimination while a C-index of 0.5 will be achieved by a completely non-informative marker. The `Cindex()` family is based on an estimator by Uno et al. (2011), which incorporates inverse probability of censoring weighting `ipcw`. To make the estimator differentiable, sigmoid functions are applied; the corresponding smoothness can be controlled via `sigma`. For details on `Cindex()` see Mayr and Schmid (2014).

Hurdle models for zero-inflated count data can be fitted by using a combination of the `Binomial()` and `Hurdle()` families. While the `Binomial()` family allows for fitting the zero-generating process of the Hurdle model, `Hurdle()` fits a negative binomial regression model to the non-zero counts. Note that the specification of the Hurdle model allows for using `Binomial()` and `Hurdle()` independently of each other.

Linear or additive multinomial logit models can be fitted using `Multinomial()`; although is family requires some extra effort for model specification (see example). More specifically, the predictor must be in the form of a linear array model (see `%O%`). Note that this family does not work with tree-based base-learners at the moment. The class corresponding to the last level of the factor coding of the response is used as reference class.

`RCG()` implements the ratio of correlated gammas (RCG) model proposed by Weinhold et al. (2016).