Department of Statistics
Ludwig-Maximilians-University Munich

Stability selection for component-wise gradient boosting in multiple dimensions

Master's thesis

Janek Thomas

Supervisors: Prof. Dr. Bernd Bischl
Ludwig-Maximilian-University Munich

Dr. Andreas Mayr
Dr. Benjamin Hofner
Friedrich-Alexander-University Erlangen-Nürnberg

February 19, 2016

Statutory Declaration

I declare that I have developed and written the enclosed Master's Thesis completely by myself, and have not used sources or means without declaration in the text. Any thoughts from others or literal quotations are clearly marked. The Master's Thesis was not used in the same or in a similar version to achieve an academic grading or is being published elsewhere.

Munich, February 19, 2016

. . . . . .

Janek Thomas

Abstract

In a large scale analysis to investigate the abundance of the common eider in Massachusetts, a hurdle model was fitted to handle excess zeros, overdispersion, nonlinearity and spatiotemporal structures. The number of birds was estimated via boosting generalized additive models for location, scale and shape (GAMLSS), allowing both mean and overdispersion to be regressed on covariates and incorporating variable selection.

An increasingly popular way to obtain stable sets of covariates while controlling the false discovery rate (FDR) is stability selection. The model is fitted repeatedly to subsampled data and variables with high selection frequencies are extracted.

Currently, this leads to a fundamental problem with boosted GAMLSS, where in every boosting iteration, the algorithm sequentially selects the best fitting effect for each distribution parameter. Thus, it is currently not possible to stop fitting individual parameters as soon as they are sufficiently modeled.

In order to solve this problem, we developed a new approach to fit boosted GAMLSS. Instead of updating all distribution parameters in each iteration, only the update of the parameter which leads to the biggest reduction in loss is performed. With this modification, the stability selection framework can be applied. Furthermore, optimizing the tuning parameters of boosting is reduced from a multidimensional to a one-dimensional problem.

The performance of the algorithm is evaluated in a large-scale simulation study and the application is demonstrated for the seabirds data, selecting stable predictors while controlling the FDR.

Contents

1 Introduction to boosting

"Garnering wisdom from a council of fools"

This quote by Schapire and Freund [2012] grasps the idea behind *statistical boosting* in one sentence. The fools in this case are so called *(weak) base-learners*, which are combined to estimate complex statistical dependencies. They are *weak* in the sense, that they only possess moderate prediction accuracy, but on the other hand are easy and fast to fit.
One base-learner by itself will most likely not be enough to answer a statistical question, but if a large number is combined in a smart way, they can compete with advanced and complex model classes.
Some examples for base-learners are *linear regression models*, *stumps* of regression or classifcation trees or nonlinear functions like P or B*-splines* [Eilers and Marx, 1996].

1.1 Boosting - Why?

In a standard regression model, the number of observations n has to be larger than the number of possible covariates p. If this is not the case, $\boldsymbol{X}'\boldsymbol{X}$, with $\boldsymbol{X}$ as the *designmatrix* of the data $\boldsymbol{x}$, can't be inverted and standard fitting algorithms cannot be applied. These are so called $p > n$ problems.
An area of research where this kind of data is very widespread, is the analysis of biometric data. For example the *ARCANE* dataset used in Guyon et al. [2004], consists of mass-spectrometric data of 900 patients with 10000 covariates. A general introduction to $p > n$ problems, especially in biometric research, can be found in Van De Geer et al. [2004].
Boosting can handle these $p > n$ situations, because each base-learner only takes a small number of covariates (in most cases just one) and thus can be easily fitted to the outcome. Combined, the base-learners have the ability to model complex nonlinear data even when the number of observations is small compared to the number of covariates.
In these situations with a large number of covariates, it is often in the interest of the scientist to identify a subset of covariates to model the response variable. This step is especially important the case of $p > n$ situations, but should not be ignored situations where a large enough number of observations is available as well.
The boosting methodology is a method for intrinsic variable selection. In each step, only the best fitting base-learner is chosen to be updated. In most cases one base-learner directly corresponds to a covariate, so it is possible to directly measure the influence a covariate has on the model accuracy. Variables with no influence will (ideally) never be included in the update and thus are dropped from

the model.
Using only a subset of the *best* variables has some major advantages: (a) the model is easier and faster to fit, (b) the prediction quality of the model can be improved, (c) a better interpretation of the model is possible, and (d) it will most likely be easier and cheaper to obtain additional data, which can be required if the model fit is not sufficient.

The concept of boosting is wildly used and can be expanded in different ways, compare for example Mayr et al. [2014]. One extension is to apply it to multidimensional prediction functions as we will see in Chapter 2. The concept of variable selection can be expanded with *stability selection*, which is shown in Chapter 3. A combination of both approaches and their complexity is discussed in Chapters 4 to 6.

1.2 Boosting - How?

Generally we want to model a response variable $Y = y_1, \ldots, y_n$, by a set of covariates $\boldsymbol{x}$. Each covariate $\boldsymbol{x}_j,\ j = 1, \ldots, p$ consists of n observations $\boldsymbol{x}_j = x_{j1}, \ldots, x_{jn}$. The designmatrix $\boldsymbol{X}$ consints of all p covariates as well as an constant *intercept* term: $\boldsymbol{X} = (\mathbf{1}, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_p)$.
The relation of the covariates to the response can be represented by an *additive predictor* η:

$$\eta(\boldsymbol{X}) = \beta_0 + \sum_{j=1}^{p} f_j(\boldsymbol{x}_j), \tag{1}$$

where β_0 is the *intercept* coefficient and $f_j(\cdot)$ are functions of the different covariates.
The relation between the expected value of Y and η is specified via the *link function* $g(\cdot)$:

$$\eta(\boldsymbol{X}) = g(\mathbb{E}(Y)). \tag{2}$$

The simplest form for the $f(x_j)$'s are linear effects $f(x_j) = \beta_j x_j$. With an adequate response distribution the model represents a *Generalized Linear Model* (GLM) [McCullagh and Nelder, 1989]. For other forms of base-learners, e.g. P-splines [Eilers and Marx, 1996], the model class is called *Generalized Additive Model* (GAM) [Hastie and Tibshirani, 1990].
To fit such a model, we want to minimize the *empirical risk*:

$$R = \sum_{i=1}^{n} \rho(y_i, \eta(\boldsymbol{x}_{.i})), \tag{3}$$

where $\boldsymbol{x}_{.i}$ is a vector of the i'ts elements of every covariate vector and $\rho(\cdot, \cdot)$ is the *loss function*, which measures the discrepancy between the true value of $\boldsymbol{Y}$ and the estimation based on $\boldsymbol{X}$.
This can be the absolute or quadratic difference, as well as the negative log-likelihood of an arbitrary distribution family of Y, depending on the desired model [Friedman et al., 2000].
Here, a boosting approach can be utilized: Friedman [2001] showed, that to minimize the empirical risk (3), it is sufficient to sequentially add base-learners $h(x_j)$ to the model in a greedy way. This means, that in each iteration a base-learner is added to the model, without adjusting the parameters of base-learners that already have been added.
This *boosting approach* can be used to estimate Equation 1.
The estimated additive predictor $\hat{\eta}$ at iteration m can be written as:

$$\hat{\eta}^{[m]} = \hat{\eta}^{[m-1]} + \nu \cdot \hat{h}_j^{[m]}(x_j), \tag{4}$$

with $0 < \nu \ll 1$ as the *learning rate* or *shrinkage parameter*. That is, in one step of the iterative fitting, only a small margin of the actual base-learners is added to the model. This is useful to prevent the model from overfitting the data and develop bad prediction accuracy on new observations. The choice of ν is not of critical importance as long as it is chosen small enough [Schmid and Hothorn, 2008].
To fit the model, a single base-learner has to be selected in every step of the fitting process. There are several different ways to fit and select base-learners. The idea of *gradient* boosting is to calculate the negative gradient of the loss function,

$$\boldsymbol{u} = -\frac{\partial}{\partial \eta} \rho(y, \eta) \tag{5}$$

and subsequently fit each base-learner separately to $\boldsymbol{u}$ instead of the data directly. For this linear base-learner in a GLM setting, the estimation can be done via *least squares*. For smooth base-learner in a GAM setting *penalized* least squares has to be used, compare for example Eilers and Marx [1996].
The negative gradient has a similar form to the *residuals* in a classic regression setting (in case of a square loss, they are actually identical) and are consequently named *pseudo residuals*. The best base-learner is then selected based on the minimization of the *quadratic loss*:

$$\sum_{i=1}^{n} (u_i - h_j(x_{ij}))^2. \tag{6}$$

Because of the quadratic loss this process is identical to the minimization of the *residual sum of squares* (RSS) criterion. Consequently this means, that Equation 6 can be fitted with the *least squares* criterion, regardless of the actually used loss function. In other words, the loss function ρ does no need to be identical to the quadratic-loss of the RSS.
The algorithm of *component-wise gradient boosting* fitting is shown as Algorithm 1. After initialization, the algorithm follows the steps described above. The negative gradient is computed, the base-learners are fitted to the negative gradient, the best base-learner is selected via Equation 6 and then added to the set of already calculated base-learners.
This process represents an optimization of the empirical risk (Equation 3) by steepest gradient descent in function space [Friedman, 2001], [Bühlmann and Hothorn, 2007].

1.2.1 The importance of early stopping

Considering Algorithm 1, it can be seen, that additional to the loss function $\rho(\cdot, \cdot)$ and the base-learner $h_j(x)$, one more hyperparameter has to be specified: The number of iterations m_{stop}. As mentioned the choice of the learning rate is not really important as long as it is chosen small enough. Depending on the base-learner, additional hyperparameters can be required as well, for example the degrees-of-freedom in B or P-splines.
The number of iterations has a crucial influence on the performance of the model. If the number of iterations is too small, the model cannot fully grasp the influence of the covariates on the response and will consequently have a bad performance. On the other hand, too many iterations will result in *overfitting* the model. This means that the model tries to explain the *random* error in the data, instead of only the influence of covariates on the response. Generally, the model will predict observations used to fit the model very well (the risk, Equation 3, will be small), but *new* observations will be predicted very poorly.
To avoid overfitting, *cross-validation* can be used to evaluate the model. The model is estimated only on a subset of the existing data (the *training set*) and evaluated on the remaining observations (the *test set*). This process is repeated multiple times with different training and test sets to evaluate the model for different m_{stop}. Thus, the optimal choice of m_{stop} is based on the performance on data points that are not used to fit the model, so that the model is less likely to overfit.
Different *resampling* approaches can be applied via the *out-of-bag* (OOB) risk: *Bootstrapping* means to use B samples of size n, drawn *with replacement* from the data to estimate the model and evaluate it on the data not sampled. *K-fold cross-*

Algorithm 1 Component-wise gradient boosting, Mayr et al. [2014]

- Initialization:
  - Set $\hat{\eta}_i^{[0]}$ to an offset value
  - Specify a set of base-learners $h_1(x_1), \ldots, h_J(x_J)$ where J is the number of different base-learner (note that for each covariate more than one base-learner can be used).
- For $m = 1$ to m_{stop}:
  1. Compute the negative gradient u_i of the loss function $\rho(y_i, \eta_i)$, evaluated at the previous iteration:
  $$u_i^{[m]} = -\frac{\partial}{\partial \eta_i} \rho(y_i, \eta_i)\Big|_{\eta_i = \hat{\eta}_i^{[m-1]}}$$
  2. Fit each base-learner to the negative gradient vector $\boldsymbol{u}^m$ with (penalized) least squares estimation:
  $$\boldsymbol{u}^{[m]} \overset{\text{base-learner}}{\longrightarrow} \hat{h}_j^{[m]}(x_j) \quad \text{for } \; j = 1, \ldots, J$$
  3. Select the base-learner j^* that minimizes the RSS:
  $$j^* = \underset{1 \leq j \leq J}{\operatorname{argmin}} \sum_{i=1}^{n} (u_i^{[m]} - \hat{h}_j^{[m]}(x_j))^2$$
  4. Update the additive predictor $\hat{\eta}$ with the selected base-learner:
  $$\hat{\eta}^{[m]} = \hat{\eta}^{[m-1]} + \nu \cdot \hat{h}_{j^*}^{[m]}(x_{j^*})$$

validation is the process of splitting the data in K fractions and fit the model on all but one fraction, which is then used to measure the model performance. This is repeated for every of the K fractions. *Subsampling* is similar to the *bootstrap* approach, but samples are drawn *without replacement* and the sample-size is smaller than n.
Choosing the optimal number of iterations via resampling procedures will most likely result in *early stopping* of the fitting process and *shrinkage* of the coefficients. This means, that the resulting coefficients are smaller than the true coefficients or even zero. This early stopping rule generally results in better prediction accuracy than if the algorithm is run until convergence [Mayr et al., 2012b]. Here the intrinsic variable selection mechanism of boosting can be observed, as non-informative variables will most likely be not included in the model.

1.3 Example: Seabird populations

In a large scale analysis by Smith et al. [2016], the abundance of wintering sea ducks in Nantucket Sound, USA, was investigated and their implications for marine spatial planning, especially for offshore wind energy developments (OWED) was assessed.
The study was conducted between 2003 and 2005. The research area was split in $2.25m^2$ segments, see Figure 1. The spatial and temporal variation of three different species was investigated: Common eider, long-tailed duck and scoter. For the following analysis only the common eider is considered.
The data was collected by counting seaducks on multiple aerial strip transects with a small plain. The researchers were interested in variables that explain the distribution of the common eider in the examined area.
A two-step hurdle model [Mullahy, 1986] was used. In a first step it was estimated if a segment is populated at all (occupancy model). For all populated areas the abundance was estimated (conditional abundance model). In this introductory example only the occupancy model will be considered. In Chapter 7 the conditional abundance model is estimated.
To answer the question whether a given segment will be populated at all, the response variable has to be discretized. All segments with at least one observed common eider are marked populated (1) and segments with a count of zero are assumed unpopulated (0). This results in a binary classification problem and thus can be fitted with a *binomial loss*:

$$\rho(y, \eta(\boldsymbol{x})) = -y \log \left(\frac{e^{\eta(\boldsymbol{x})}}{e^{\eta(\boldsymbol{x})} + e^{-\eta(\boldsymbol{x})}} \right) - (1 - y) \log \left(1 - \frac{e^{\eta(\boldsymbol{x})}}{e^{\eta(\boldsymbol{x})} + e^{-\eta(\boldsymbol{x})}} \right). \quad (7)$$

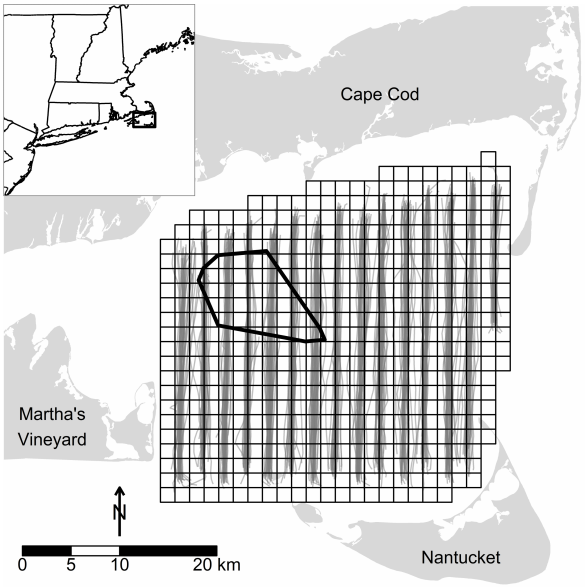

Figure 1: Area of research of the seabird study

A large number of biophysical variables had to be considered as potential predictors (compare Table 4 in Chapter 7). These can be classified into three groups: 5 temporal predictors (T), like the date or the North Atlantic Oscillation index, 11 spatial predictors (S), like the sediment grain size or the sea floor surface area, and 5 spatio-temporal (ST) predictors such as different water temperature measurements and the effect of the actual surveyed area.
All of these effects can either be linear or nonlinear in their influence. These effect types can be intrinsicly selected with an approach by Kneib et al. [2009]. The main idea is to deconstruct a smooth base-learner in a linear part and smooth part which captures the deviation from linearity [Fahrmeir et al., 2004]. We assign the same degrees of freedom for both base-learners to reduce selection bias, [Hofner et al., 2011]. In case of a linear and a smooth base-learner, selection bias means, that smooth base-learners will be preferred due to their higher flexibility. With this approach 48 different base-learners can be selected by the boosting algorithm. For the smooth base-learners centered P-splines with one degree of freedom are used. The linear base-learner are simple linear regressions of the form $\beta_i x_i$, that are fitted *without* an intercept term.
The model is fitted with a learning rate ν of 0.3 and the model quality is evaluated with the OOB risk determined by 25-fold subsampling (Figure 2). Even though the learning rate is chosen relatively high, no overfitting can be observed. In

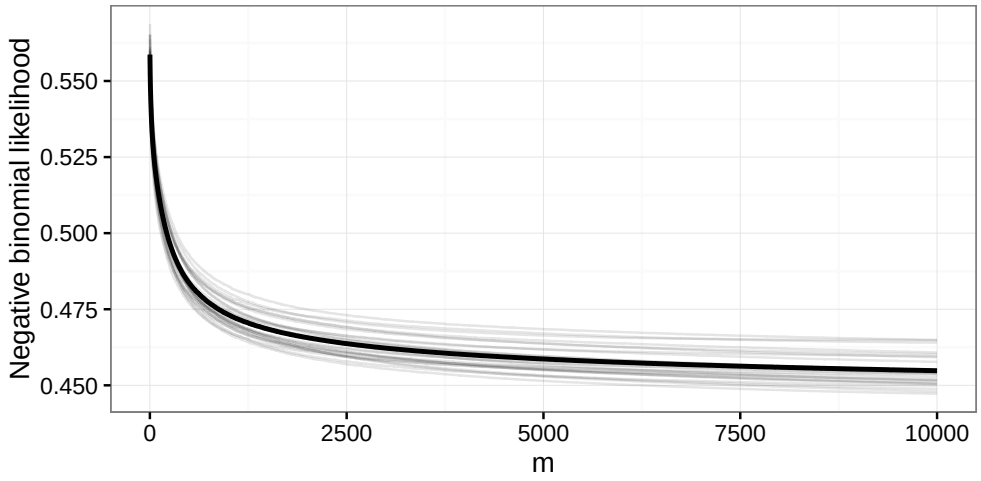

Figure 2: OOB risk of the the population model for up to 10000 iterations. No overfitting is present

principle even more iterations could be performed, but the improvement in the last iterations is only around 0.017. Out of the 48 possible predictors 42 are included in the boosting model. The selected variables and their *importance* are shown in Figure 3. The importance of a base-learner is defined as the relative overall loss reduction gained by a respective base-learner [Friedman, 2001]. More than half of the reduction in loss is based on the smooth spatial term of the measured segment. All other base-learners have a way smaller influence, with a smooth term of the sediment grain size (meanphi) as second most important base-learner with an relative importance of 6.5%. The full variable names and descriptions can be found in Table 4.
As can be seen, this is a rather large model with hardly any variable selection. This hampers the interpretability a lot. One possible way to obtain a sparser model is *stability selection*, which is introduced in Chapter 3 and applied to the population model in Chapter 3.1.

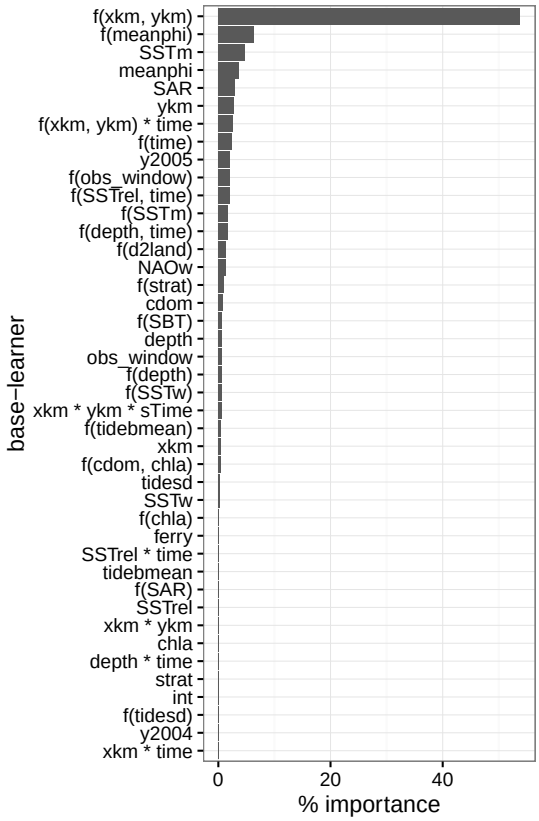

Figure 3: Relative importance of base-learners in the boosting model for common eider population.

2 Component-wise gradient boosting in multiple dimensions

The framework of component-wise gradient boosting, which was introduced in Chapter 1, can be expanded to multidimensional prediction functions, as shown by Schmid et al. [2010]. Until now, we assumed that the conditional distribution of the response variable $\boldsymbol{Y}$ depends on only one parameter, usually the mean, or if the distribution has multiple parameter, all but one are *constant*.
For example in a regression setting with an assumed *normal distribution*, the *location parameter* μ_i depends on the observation i and possibly a set of covariates, while the *scale parameter* σ_i is assumed to be constant $\sigma_i \equiv \sigma$.
Such an assumption cannot always be made and should be critically examined. One case in which the assumption of constant parameter(s) is not adequate, is if the data features *heteroscedasticity*. If the *homoscedasticity* assumption holds, the range in which the response varies should be similar for all possible values of y. As seen in Figure 4, the variance of y shows a strong trend. In the left panel no influence of X_1 on the range of the data can be seen. On the right panel, with larger values of X_1 the range of y increases. Such a behavior may be explained with covariates, just like the distinction in the location parameter for different observations.

When there is a dependency between the scale parameter and possible predictors, it can be modeled in a similar way to the conditional mean (i.e. location parameter). The prediction function η_σ is analogous to Equation 1:

$$g(\sigma_i) = \eta_{\sigma,i} = \beta_{0,\sigma} + \sum_{j=1}^{p_\sigma} f_{j\sigma}(x_j). \tag{8}$$

To distinguish between the different prediction functions they are indexed with their corresponding parameter. The same idea applies, if different distribution parameters or higher moments, like the *shape* ν or *kurtosis* τ, are modeled.
One case where the skewness of the distribution changes is the age and body mass index (BMI) measurement of 7482 Dutch boys by Fredriks et al. [2000]. The measurements are visualized in Figure 4. It can be seen, that the skewness of the distribution increases with age of the boys. Older boys have a tendency for more extreme values in one direction (higher BMI) and less in the other (lower BMI).
One last, but important example is handling count data with *overdispersion* and/or large amount of zeros. We will see this in the analysis of the abundance of the common eider in Chapter 7.

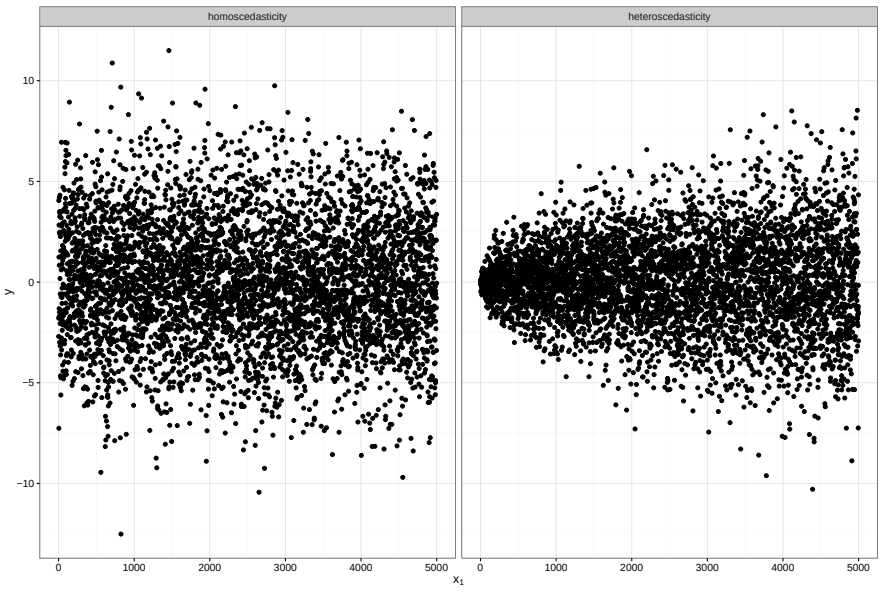

Figure 4: Data with heteroskedasticity (right) and without (left)

2.1 Generalized additive models for location scale and shape (GAMLSS)

The framework to fit different prediction functions to multiple distribution parameters was developed by Rigby and Stasinopoulos [2005]. Given a conditional density $f(y_i|\boldsymbol{\theta}_i)$, the idea is to fit multiple equations, similar to Equation 1:

$$\eta_{\theta_k} = \beta_{0\theta_k} + \sum_{j=1}^{p_k} f_{j\theta_k}(x_{kj}), \qquad k = 1, \ldots, d \tag{9}$$

for each distribution parameter θ_k. In principle the vector of distribution parameter $\boldsymbol{\theta} = (\theta_1, \ldots \theta_d)$ can be an arbitrary large vector of distribution parameters. Yet, in practical applications more than four are not recommended, [Rigby and Stasinopoulos, 2005]. The four distribution parameters $\boldsymbol{\theta} = (\theta_1, \theta_2, \theta_3, \theta_4)$ will be called *location* (μ), *scale* (σ), *shape* (ν) and *kurtosis* (τ), in this exact order for the rest of this thesis. Even though the actual parameters may include for example the *degrees of freedom* of a t-distribution or an *zero-inflation* parameter for zero-inflated count models. Subsequently the distribution parameter can be written as $\boldsymbol{\theta}_i = (\mu_i, \sigma_i, \nu_i, \tau_i)$. This model class is called *generalized additive model for location, scale and shape* (GAMLSS), because of the additive structure of the predictor (Equation 9) and the fitting of different distribution parameters.

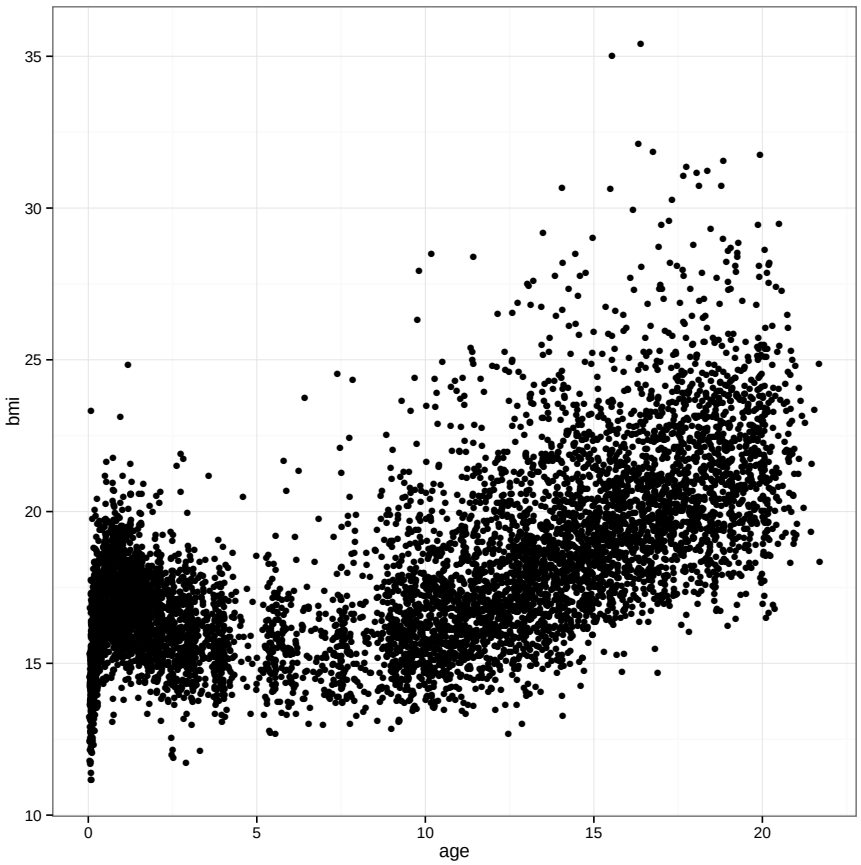

Figure 5: BMI and age measurments of 7482 dutch boys

To link the linear predictors to the estimated distribution parameters, multiple link functions $g_k()$ similar to Equation 2, have to be specified. The inverse link function of the corresponding additive predictor results in the estimated distribution parameter:

$$\begin{aligned}\hat{\mu}_i &= g_\mu^{-1}(\hat{\eta}_\mu)\\ \hat{\sigma}_i &= g_\sigma^{-1}(\hat{\eta}_\sigma)\\ \hat{\nu}_i &= g_\nu^{-1}(\hat{\eta}_\nu)\\ \hat{\tau}_i &= g_\tau^{-1}(\hat{\eta}_\tau)\end{aligned}$$

For example, the link functions in a GAMLSS model with normal distribution are $g_\mu(x) = x$ and $g_\sigma(x) = \exp(x)$.

Typically these models are estimated via *penalized likelihood*. For details on the fitting algorithm see Rigby et al. [2008]. Even though these models can be applied to a large number of different situations, such as *risk management* [Scandroglio et al., 2013], *public health* [Rees et al., 2010] or *biometric* applications [Fenske et al., 2008], they inherit some shortcomings based on the estimation method:

(1) It is not possible to estimate models with a larger number of covariates than observations. As mentioned in the introduction these problems often occur in biometric research.

(2) One important step in (almost) every statistical model is a correct selection of relevant covariates. Unfortunately, maximum likelihood estimation does not have an incorporated way for variable selection. There are different ways to achieve variable selection *after* fitting the model, like *Akaikes information criterion* (AIC) [Akaike, 1974], or the *Bayes information criterion* (BIC) [Schwarz et al., 1978]. For GAMLSS models the standard AIC has been expanded to the *generalized* AIC (GAIC) by Rigby and Stasinopoulos [2005] to be applied to multidimensional prediction functions, see Equation 9.

(3) In a GAM setting the question, which predictors to model linear or nonlinear, is not trivial. Unneeded smooth terms increase the complexity of the model as well as the computation time. Similar to the problem of variable selection, the GAIC criterion has to be used to choose between linear and nonlinear terms.

All of these problems can be addressed with a different fitting approach, similar to the component-wise gradient boosting approach shown in the introduction.

2.2 Boosting GAMLSS

To fit multiple prediction functions $\eta_\mu, \eta_\sigma, \eta_\nu, \eta_\tau$ at once, the algorithm for gradient boosting has to be expanded.
The first idea that may come to mind is to optimize every parameter independently of each other with the standard Algorithm 1. While this may give good results for particular cases, it is a bad idea overall. For a lot of cases the gradients to which we fit the base-learners cannot be viewed without consideration of the other distribution parameters. Even if we look at a simple example with the normal

distribution, the gradients of $f(x|\mu,\sigma)$ are

$$\frac{\partial f(x|\mu,\sigma)}{\partial \mu} = -\frac{(x-\mu)}{\sigma^2} f(x|\mu,\sigma), \tag{10}$$

$$\frac{\partial f(x|\mu,\sigma)}{\partial \sigma} = \frac{(x-\mu)^2-\sigma^2}{\sigma^3} f(x|\mu,\sigma). \tag{11}$$

The calculation for the both gradients is found in the Appendix Chapter 10.1, Equation 18 and 19. Each gradient contains μ *and* σ, so they cannot be fitted independently of each other.
Based on an approach of Schmid et al. [2010] to fit zero-inflated count models, Mayr et al. [2012a] developed a general purpose algorithm to fit multidimensional prediction functions with component-wise gradient boosting. The idea is to cycle through the distribution parameters in the fitting process, instead of fitting each parameter independently of the others. In one iteration of the algorithm, a base-learner is fitted to *each* distribution and the updated parameters are directly plugged in the gradients of the others. While updating one parameter, all others are viewed as *offset* or *nuisance* parameters in the gradient.
For a four dimensional prediction function $\boldsymbol{\theta} = (\mu, \sigma, \nu, \tau)$ the fitting process looks like this [Mayr et al., 2012a]:

$$\begin{aligned}
\frac{\partial \rho}{\partial \eta_\mu}(y, \hat{\mu}^{[m]}, \hat{\sigma}^{[m]}, \hat{\nu}^{[m]}, \hat{\tau}^{[m]}) &\xrightarrow{\text{update}} \eta_\mu^{[m+1]} \Longrightarrow \hat{\mu}^{[m+1]} \\
\frac{\partial \rho}{\partial \eta_\sigma}(y, \hat{\mu}^{[m+1]}, \hat{\sigma}^{[m]}, \hat{\nu}^{[m]}, \hat{\tau}^{[m]}) &\xrightarrow{\text{update}} \eta_\sigma^{[m+1]} \Longrightarrow \hat{\sigma}^{[m+1]} \\
\frac{\partial \rho}{\partial \eta_\nu}(y, \hat{\mu}^{[m+1]}, \hat{\sigma}^{[m+1]}, \hat{\nu}^{[m]}, \hat{\tau}^{[m]}) &\xrightarrow{\text{update}} \eta_\nu^{[m+1]} \Longrightarrow \hat{\nu}^{[m+1]} \\
\frac{\partial \rho}{\partial \eta_\tau}(y, \hat{\mu}^{[m+1]}, \hat{\sigma}^{[m+1]}, \hat{\nu}^{[m+1]}, \hat{\tau}^{[m]}) &\xrightarrow{\text{update}} \eta_\tau^{[m+1]} \Longrightarrow \hat{\tau}^{[m+1]} \\
&\dots \quad ,
\end{aligned}$$

where m is the previous iteration and ρ the specified loss function.
At the beginning of the fitting process, each predictor has to be set to an initial value. This can be the *empirical mean* for $\hat{\mu}^{[0]}$, the *sample variance* for $\hat{\sigma}^{[0]}$ or generally a constant value c with property:

$$\hat{\eta}_{\theta_k}^{[0]} := \underset{c}{\operatorname{argmin}} \sum_{i=1}^{n} \rho(y_i, \eta_{\theta_k} = c) \tag{12}$$

The full algorithm is can be found as Algorithm 2 and it is visualized in Figure 6. After initialization, the algorithm cycles through the distribution parameters and adds a base-learner to each prediction function in a similar fashion than Algorithm

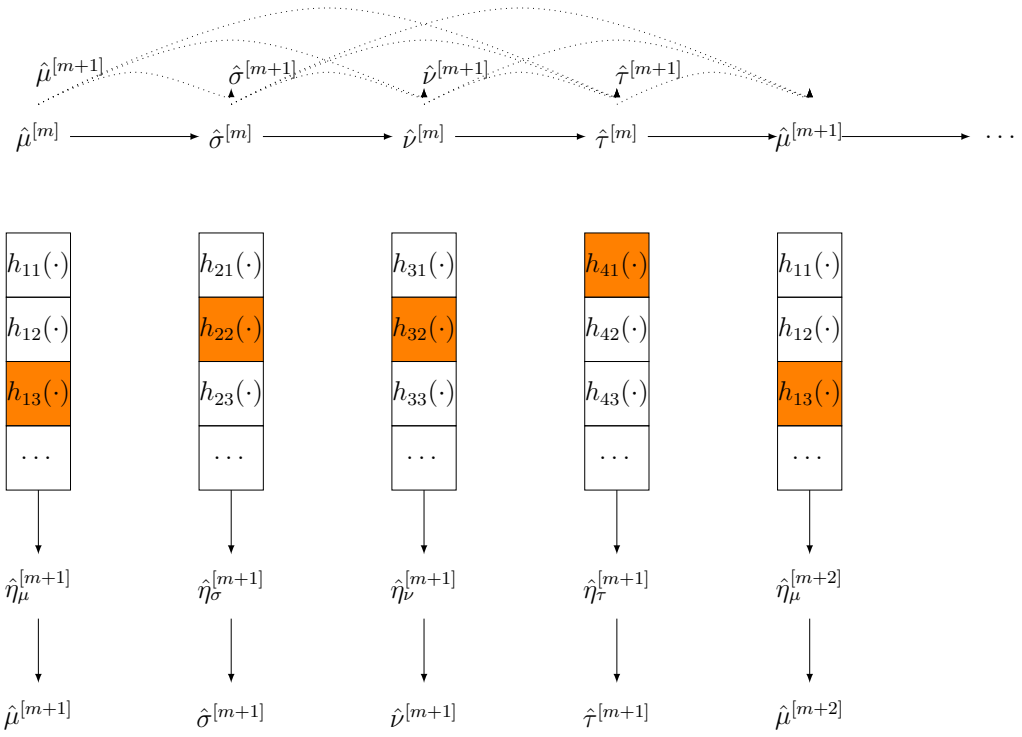

Figure 6: *Cyclical* fitting algorithm as defined in Mayr et al. [2012a]. The highlighted boxes represent the selected base-learners which yield the updated prediction function and consequently the estimated distribution parameter. The dashed lines shows the usage of the updated parameters as nuisance parameters in the succeeding parameters.

1. Separate stopping values for each distribution parameter have to be specified as $m_{\text{stop}} = (m_{\text{stop},1}, m_{\text{stop},2}, m_{\text{stop},3}, m_{\text{stop},4})$. This is required because the prediction functions will most likely require different levels of complexity and therefore varying numbers of iterations to be optimal.
The main characteristic feature of the presented algorithm is its property of cycling through all distribution parameters in one iteration. In view of other algorithms that will be introduced later, this one will be called *cyclical* fitting throughout the thesis. A nice property of GAMLSS is, that a seperate set of covariates can be specified for each additive predictor and consequently each distribution. In combination with the internal variable selection in the boosting algorithm, the most relevant covariates for each distribution parameter can be found. This can give

Algorithm 2 Component-wise gradient boosting in multiple dimensions by Mayr et al. [2012a]

Initialize

(1) Initialize the additive predictors $\hat{\eta}_{\mu_i}^{[0]}$, $\hat{\eta}_{\sigma_i}^{[0]}$, $\hat{\eta}_{\nu_i}^{[0]}$, $\hat{\eta}_{\tau_i}^{[0]}$ with offset values. Set the iteration counter $m = 0$

(2) For each distribution parameter $\theta_k, k = 1, \ldots, 4$ specify a set of base-learners: i.e. for parameter θ_k: $h_{k1}(\cdot), \ldots, h_{kp_k}(\cdot), k = 1, \ldots, 4$, where p_k is the cardinality of the set of base-learners specified for θ_k.

Boosting in multiple dimensions

(3) **Start** a new boosting iteration: increase m by 1 and set $k := 0$.

(4) (a) Increase k by 1.

If $m > m_{\text{stop},k}$ set $\hat{\eta}_{\theta_k}^{[m]} := \hat{\eta}_{\theta_k}^{[m-1]}$ and skip this iteration.

Else compute negative partial derivative $-\frac{\partial}{\partial \eta_{\theta_k}} \rho(y_i, \boldsymbol{\eta}_i)$ an plug in the current estimates $\boldsymbol{\eta}_i = (\hat{\eta}_{\mu_i}^{[m-1]}, \hat{\eta}_{\sigma_i}^{[m-1]}, \hat{\eta}_{\nu_i}^{[m-1]}, \hat{\eta}_{\tau_i}^{[m-1]})$:

$$\boldsymbol{u}_k^{[m]} = \left(\frac{\partial}{\partial \eta_{\theta_k}} \rho(y_i, \boldsymbol{\eta}_i) \right)_{i=1,\ldots,n} \Big|_{\eta = \hat{\eta}_i^{[m-1]}}$$

(b) **Fit** the negative gradient vector $\boldsymbol{u}_k^{[m]}$ to each of the base-learners contained in the set of base-learners specified for the predictor η_{θ_k} in step (2).

(c) **Select** the component j^* that best fits the negative partial-derivative vector according to the leas-squares criterion, i.e. select the base-learner h_{kj^*} defined by

$$j^* = \underset{1 \leq j \leq p_k}{\operatorname{argmin}} \sum_{i=1}^{n} (u_{ik}^{[m]} - h_{kj}(\cdot))^2.$$

(d) **Update** the additive predictor η_{θ_k} as follows:

$$\hat{\eta}_{\theta_k}^{[m]} = \hat{\eta}_{\theta_k}^{[m-1]} + \nu \cdot \hat{h}_{j^*}^{[m]}(\cdot).$$

(e) **Iterate** steps 4(a) to 4(d) for $k = 2, \ldots, 4$.

Iterate

(5) Iterate steps 3 to 4 until $m > m_{\text{stop},k}$ for all $k = 1, \ldots, 4$

interesting insights in hidden dependencies for the researcher. A covariate can for example have a relevant influence on the variance of the data but not on the mean. We will see this behavior in the *seabird* dataset in Chapter 7.
The optimization of one stopping value was shortly discussed in Chapter 1.2.1. In case of multi-dimensional boosting the same techniques are applied but the optimization gets considerably more difficult. The different m_{stop} values are not independent of each other and the number of configurations to check in a *grid search* is growing exponentially with each additional distribution parameter. For example a grid with 5 possible settings for 2 parameters requires to check 25 combinations, with 3 parameters 125 combinations have to be examined and for 4 parameters over 600.
With boosting, the problem requires a bit less computational power, because a whole path of iterations from 1 to m_{stop} is returned, but the computational effort is still high.
One solution would certainly be to use more efficient methods like *model based optimization* [Jones et al., 1998], but a different approach was utilized in this thesis. In Chapter 5 a way of fitting boosting GAMLSS models is introduced, with this algorithm the optimization of the stopping iterations is reduced to a one-dimensional problem, independent of the number of distribution parameters to fit.

Function	Description
`bols(x)`	Linear effect $\boldsymbol{x}^T\beta$ with intercept (dummy coding for factors)
`bols(x, by = y)`	Linear effect interaction
`bbs(x)`	Smooth P-spline
`bbs(x, by = z)`	Varying coefficient
`bbs(x)`	Smooth P-spline
`bspatial(x)`	Spatial effect: tensor product P-spline
`bmrf(x)`	Discrete spatial effect: Markov random field
`brandom(x)`	Random intercept
`brandom(x, by = x)`	Random slope

Table 1: Overview of the most important base-learners in `mboost`

2.3 Implementation

The algorithm developed by Mayr et al. [2012a] is implemented in `gamboostLSS` [Hofner et al., 2015a] as a package for the *R language for statistical computing* [R Core Team, 2015]. The implementation is based on the popular package `mboost` [Hothorn et al., 2010, 2015] for component-wise gradient boosting. Essentially, a `gamboostLSS` model is a combination of multiple `mboost` objects, one for each distribution parameter. While fitting the models, the fit of one `mboost` model is iteratively written in the nuisance value(s) of the remaining model(s).
This implementation allows the usage of the full `mboost` framework. Among others, linear and nonlinear base-learners, spatial or random effects can be used out of the box. An overview of the most important base-learners can be seen in Table 1.
Additionally, a lot of different distributions and regression models are built in `gamboostLSS`; see Table 2. Distributions used in the `gamlss` package can be converted with `as.families()` to distributions that `gamboostLSS` understands. The number of supported distributions is further increased by the package `gamlss.dist` [Stasinopoulos et al., 2015], which contains even more distributions.

Function	Distribution	Parameters
`GaussianLSS()`	Normal distribution	2
`GammaLSS()`	Gamma distribution	2
`StudentTLSS()`	Student's distribution	3
`NbinomialLSS()`	Negative binomial distribution	2
`WeibullLSS()`	Weibull distribution	2
`LogLogLSS()`	Log-logistic distribution	2
`LogNormalLSS()`	Log-normal distribution	2
Additional distributions via `as.families()` in `gamboostLSS`		
`GT()`	Generalized Student's distribution	4
`BCT()`	Box-Cox Student's distribution	4
`GU()`	Gumbel distribution	2
`RG()`	Reverse Gumbel distribution	2
`IGAMMA()`	Inverse Gamma distribution	2
`ZAGA()`	Zero adjusted Gamma distribution	3
`IG()`	Inverse Normal distribution	2
`BCCG()`	Box-Cox Cole and Green distribution	3
`PARETO2()`	Pareto type 2 distribution	2
`BCPE()`	Box-Cox power exponential distribution	4
`BE()`	Beta distribution	2
`BEINF()`	Beta inflated distribution	4
`BB()`	Beta Binomial distribution	2
`ZINBI()`	Zero inflated negative binomial distribution	3
`SI()`	Sichel distribution	3
`DEL()`	Delaporte distribution	3

Table 2: Overview of supported distributions in `gammboostLSS`

3 Stability selection

As mentioned in Chapter 1.1 of this thesis, selecting an optimal subset of explanatory variables is a crucial step in almost every data analysis problem. Especially in situations with a large number of covariates it is often almost impossible to get meaningful results without *automatic*, or at least *semi-automatic*, selection of the most relevant predictors.
Selection of covariate subsets based on modified R^2 criteria (e.g. the AIC) can be unstable, see for example Flack and Chang [1987]. Methods like *wrapper* (see Kohavi and John [1997] for an overview) or *genetic algorithms* [Vafaie and De Jong, 1992] to search the space of covariates, $\{1, 0\}^p$, can be infeasible for high dimensional data, [Phuong et al., 2006].
One solution to select predictors in high dimensions and/or $p > n$ problems are boosting algorithms. As discussed in Chapter 1.2, boosting with cross-validation for *early stopping* offers an internal way to do variable selection while fitting the model.
A problem with variable selection via boosting is that the resulting model can still contain a rather large number of noise variables (compare for example Bühlmann and Yu [2006] or Mayr et al. [2012b]) . These are variables that have (almost) no influence on the response. Similar behavior can be observed with other *regularization* based selection methods like *lasso* [Tibshirani, 1996] or *elastic net* [Zou and Hastie, 2005] regression [Meinshausen and Bühlmann, 2010].
To circumvent this problem, Meinshausen and Bühlmann [2010] developed the *stability selection* approach. This general purpose algorithm can be applied to boosting and all other variable selection methods.
The main idea of *stability selection* is to run the selection algorithms on multiple subsamples of the original data. The idea is, that highly relevant variables should be selected in (almost) all subsamples. The process of *stability selection* for boosting is shown in Algorithm 3. B random subsets are drawn and a boosting model is fitted to each one (see: *subsampling* in Chapter 1.2.1). These can have at most m_{stop} iterations but are interrupted as soon as q different base-learners have entered the model. For each base-learner the selection frequency can be calculated with Equation 13. The selection frequency π_j is the fraction of subsets and consequently models, in which the base-learner j was selected. All selection frequencies are then compared with a user specified threshold π_{thr} and included in the model if $\pi_j \geq \pi_{\text{thr}}$, see Equation 14.
It can be shown that this approach leads to error bounds for the *per-family error-rate* (PFER) of including non-informative variables. The PFER is defined as the expected number of noise variables wrongly included in the model $\mathbb{E}(V)$, with V as the number of non-informative variables. The upper bound for the PFER, based

Algorithm 3 Stability selection for model-based boosting [Hofner et al., 2015b]

1. For $b = 1, \ldots, B$:

   a) Draw a subset of size $\lfloor n/2 \rfloor$ from the data

   b) Fit a boosting model until the number of selected covariates is equal to q or the number of iterations reaches m_{stop}.

2. Compute the relative selection frequencies per base-learner:

$$\hat{\pi}_j := \frac{1}{B} \sum_{b=1}^{B} \mathbb{I}_{\{j \in \hat{S}_{\lfloor n/2 \rfloor, b}\}}, \tag{13}$$

   where $\hat{S}_{\lfloor n/2 \rfloor, b}$ denote the set of selected variables in iteration b.

3. Select base-learners with a selection frequency of at least π_{thr}, which yields a set of stable covariates

$$\hat{S}_{\text{stable}} := \{j : \hat{\pi}_j \geq \pi_{\text{thr}}\}. \tag{14}$$

on Meinshausen and Bühlmann [2010], is given by

$$\mathbb{E}(V) \leq \frac{q^2}{(2\pi_{\text{thr}} - 1)p}, \tag{15}$$

where q is the number of selected variables in each subset and π_{thr} is the user-defined fraction of subsets the variable has to be included, to be considered relevant. In order for Equation 15 to hold, two assumptions have to be made [Meinshausen and Bühlmann, 2010]:

(i) The distribution $\{\mathbb{I}_{j \in \hat{S}_{\text{stable}}}, j \in N\}$ has to be exchangeable for all noise variables N.

(ii) The selection process must not be worse than random guessing.

Assumption (i) means, that each noise variable has the same selection probability. In other words, the correlation between a noise variable and the response should be identical for all noise variables. Assumption (ii) is rather self-explanatory. If the selection process does not work better than randomly selecting covariates, stability selection cannot find meaningful predictors.
An advancement of the original algorithm was developed by Shah and Samworth [2013], in order to receive stricter error bounds than Equation 15. They incorporate error control for the *expected number of selected variables with low selection probability* and multiple assumptions (E1), (E1) and (E3) which are described in detail in Shah and Samworth [2013] and Hofner et al. [2015b]. They also use *complementary pairs*, which essentially means to use subsamples which contain 50% of the observations and use the selected and not selected observations as two different subsamples.
One of the main difficulties of stability selection in practice is the choice of the parameters q, π_{thr} and PFER. Even though only two need to be specified (the last one can then be derived using Equation 15) their choice isn't trivial and not always intuitive for the user. Meinshausen and Bühlmann [2010] state that the threshold should be $\pi_{\text{thr}} \in (0.6, 0.9)$ and has little influence on the result.
The number of variables q has to be sufficiently large, that means that q should be at least as big as the number of informative variables in the data. This is obviously a problem for real world applications, because the number of informative variables is unknown. Simulation studies like Hofner et al. [2015b] have shown, that the PFER is quite conservative and the true number of false positives will be most likely smaller than the specified value. One nice property is, that if q is fixed, π_{thr} and the PFER can be varied without the need to refit the model.
In practical application two different approaches to select the parameters are typically used. Both assume that the number of covariates to include, q, is chosen

intuitively by the user, which can be a problem by itself. After that, the remaining two parameters can easily be changed without the need to refit the model. The first idea is to look at the calculated $\hat{\pi}_j$ and look for a *breakpoint* in the decreasing order of the values. The threshold can be then chosen so that all covariates larger than the breakpoint are included and the result PFER is only used as an additional information. The second possibility is to fix a PFER as a *conservative* upper bound for including non-informative variables.

3.1 Stability selection for the seabird population model

A boosted model for the population of common eider in segments of Nantucket Sound, USA, was fitted in Chapter 1.3. The intrinsic variable selection mechanism of boosting already selected a subset of predictors. Yet, this resulted in a quite large model, where 42 out of 48 possible predictors were included. To reduce the model complexity, stability selection will be applied as a second step to achieve a sparser model with error control for the inclusion of non-informative variables. Different parameter settings where tested by Smith et al. [2016] and came to the conclusion that $q = 35$ and a PFER of 3 yield plausible results. Considering $p = 48$ base-learners, the threshold π_{thr} can be calculated as 0.99 with an additional unimodality assumption (compare Shah and Samworth [2013]). In Figure 7 the relative selection frequencies $\hat{\pi}_j$ are plotted in decreasing order. The threshold of 0.99 is drawn as a dashed line, so that the selected variables are easy visually identified. All covariates *right* of the line are selected by stability selection. Compared with the 42 predictors that were selected with the boosting model from Chapter 1.3, the stability selection approach yields a considerably sparser model. A different threshold could be selected if we look for breakpoints in the selection frequency. One rather small breakpoint is at $\pi_{\text{thr}} = 0.99$ and a larger breakpoint at around $\pi_{\text{thr}} = 0.95$ which would result in a larger model. With these settings 12 out of 48 covariates are selected.

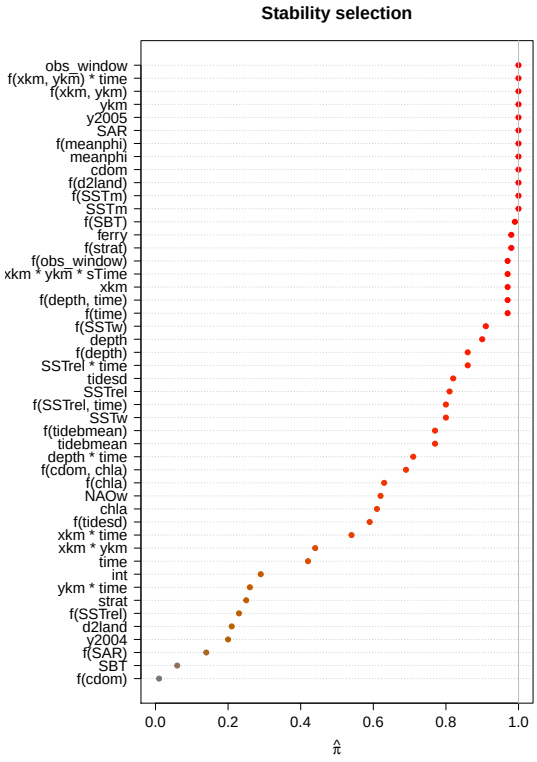

Figure 7: Stability selection for the common eider population model with $q = 35$ included variables in each sample, a PFER of 3 and a threshold of 0.99. Covariates right of the dashed line are selected.

4 Stability selection and boosted GAMLSS

In the last chapter *stability selection* was introduced as a method to achieve sparser models for boosting methods. The scope of this thesis was to find a efficient way to apply the stability selection methodology to boosting of multidimensional prediction functions.
The question of variable selection in GAMLSS and boosted GAMLSS is even more critical than in regular models with one dimensional prediction functions. The question of including a variable implies not only if the covariate should be used in the model at all, but for which distribution parameter(s) it should be used. Essentially, the number of possible covariates doubles in a distribution with two parameters, triples in one with three parameters and so on. This is especially difficult in situations with a large amount of covariates in the beginning and in $p > n$ situations.

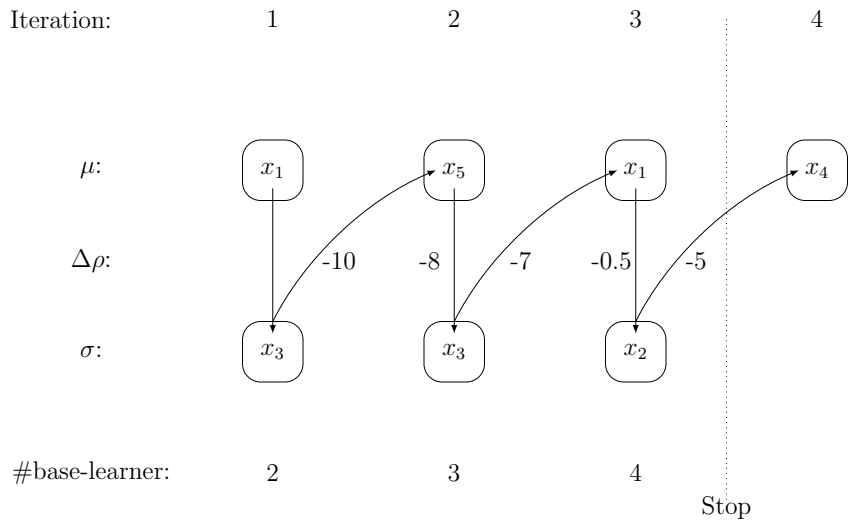

Figure 8: Fitting a boosted GAMLSS with two parameters, until $q = 4$ variables are selected. The nodes represent the selected variable in iteration m and the numbers on the edges, $\Delta\rho$, the loss reduction.

The method of fitting boosted GAMLSS models in a cyclical way leads to a severe problem when used in combination with stability selection. Base-learners are counted separately for each distribution parameter, so a variable that is selected for the location *and* scale parameter counts as two different variables regarding the maximum number of variables to consider in the model. An alternative, in

which a covariate is only counted once, is discussed in the outlook of Chapter 8. In one iteration of the algorithm *all* distribution parameters will receive an additional base-learner as long as their m_{stop} limit is not exceeded. This means, that base-learners (and consequently new variables) are added to the additive predictor of parameters in the model, that have rather small importance compared to variables that may have larger influence based on *other* distribution parameters. This gets especially relevant if the number of informative variables varies drastically between distribution parameters. In Figure 8, a small toy example of the problem is visualized. In this example, we set the maximum number of selected variables to $q = 4$, so the fitting stops as soon as four variables are included in the model. This happens at iteration 3 (obviously in real world applications the number will be much higher) which is after variable x_2 enters the model to fit σ. This means that variable x_4 won't be added to the model any more, even though the overall gain for the model fit $\Delta\rho$ is way higher than the gain of x_2 for σ, that was forced in the model based on the cyclical fitting structure.
It can be argued to choose a higher value of q, but the problem is still present, especially when the number of informative variables differs a lot, as we will see in the simulation study in Chapter 6. One aspect of the problem is, that the possible model improvement between different *distribution parameters* is not considered.
A careful selection of the number of iterations may resolve the problem, but the process will still be unstable because the margin of base-learner selection in later stages of the algorithm is quite small.

5 Noncyclical fitting algorithm

In the previous chapters we presented two major disadvantages of the *cyclical* fitting algorithm:

(i) Choosing optimal stopping values gets harder, the more distribution parameters have to be fitted. This problem was shortly discussed in Chapter 2.2. As the optimization can get very time consuming, it would be good to have the possibility to optimize these hyperparameters independent of the number of distribution parameters.

(ii) The concept of stability selection does not work well in the context of a cyclical fitting method. This can be traced back to the missing comparison of model improvement between the different distribution parameters.

We solved both of these problems with a new fitting method.

The core idea is, instead of cycling through all distribution parameters in one step of the algorithm, only the *best* distribution parameter is selected in each iteration. The notion of *best* means the distribution parameter that leads to the highest reduction of the loss.
In *gradient* boosting, the selection of the base-learners is done via comparing the RSS of the base-learners against the gradient of the parameter. Unfortunately, it is not possible to compare gradients between different distribution parameters, because they may be scaled differently.

5.1 Cyclical fitting via inner loss

Because the comparison of base-learners from different distribution parameters cannot be done via the RSS criterion (Equation 6), a different selection criterion has to be found. An obvious way is to check which distribution parameter leads to the biggest reduction in the empirical risk (Equation 3).
In the first step, the best base-learner for each distribution parameter is selected via minimizing the RSS of the respective gradients. In a second step, the potential improvement in the empirical loss $\Delta\rho$ is measured for the selected base-learners. Only the parameter and consequently base-learner with the biggest improvement is added to the model. A schematic overview of the fitting process can be seen in Figure 9. Because the base-learner selection is still done with the *inner* loss (RSS), this algorithm will be called analogously.

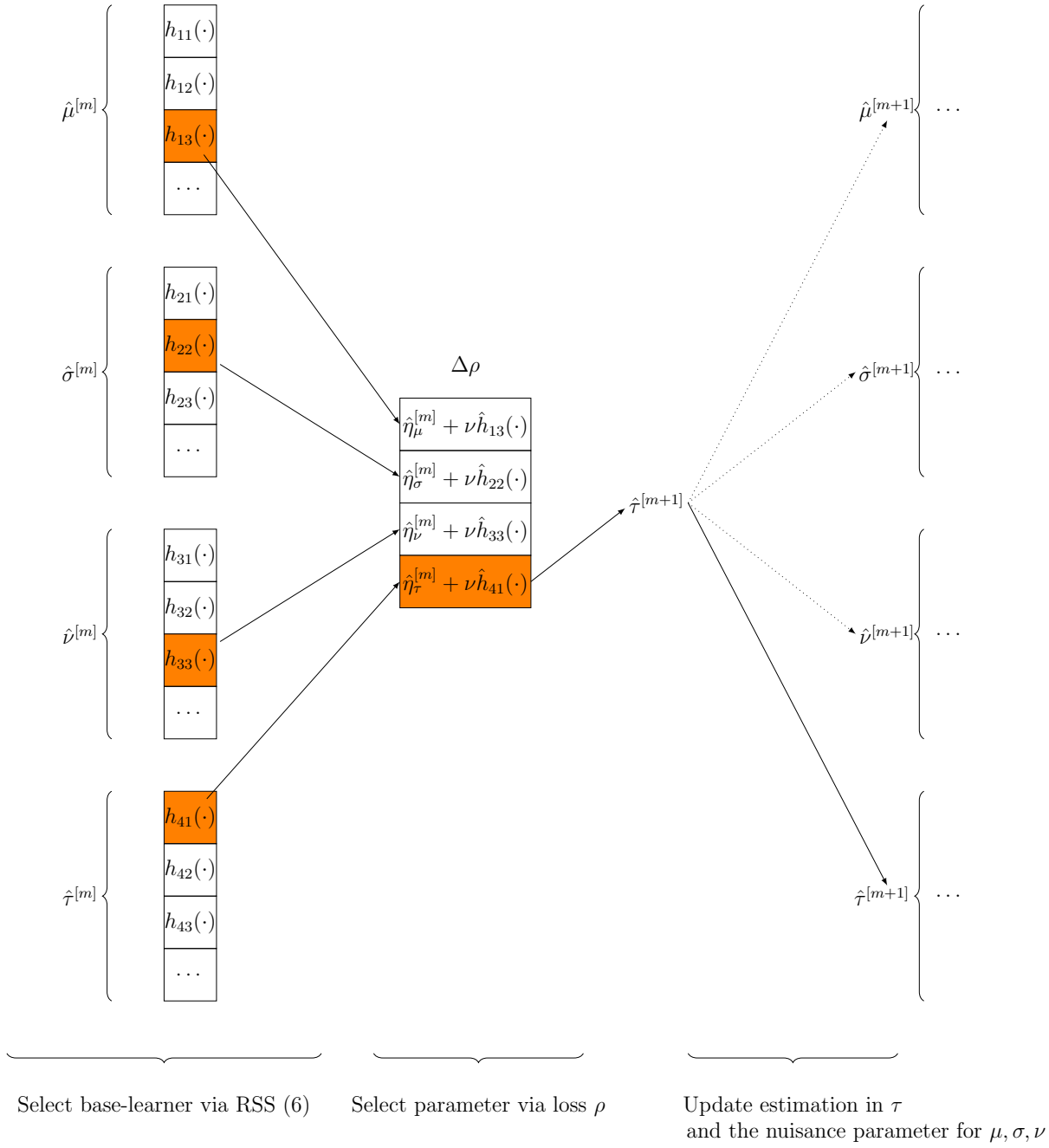

Figure 9: Noncyclical fitting algorithm - base-learner selection via *inner* loss

This alternative fitting method is defined in Algorithm 4. Each estimated predictor has its own number of iterations m_1, m_2, m_3, m_4. The overall number of iterations is the sum of all:

$$m = m_1 + m_2 + m_3 + m_4. \tag{16}$$

Here, the advantage compared to the cyclical fitting algorithm can be seen. It solves problem (i) because m_{stop} is always scalar and is adaptively composed from the iterations for each distribution parameter. The optimal partitioning (and sequence) of base-learners between different parameters is done internally while fitting the model. The only hyperparameter that has to be optimized is the (one dimensional) number of iterations, which can be done very efficiently.

Algorithm 4 Noncyclical component-wise gradient boosting in multiple dimensions – via inner loss

Initialize

- Initialize the additive predictors $\hat{\eta}_{\mu_i}^{[0]}$, $\hat{\eta}_{\sigma_i}^{[0]}$, $\hat{\eta}_{\nu_i}^{[0]}$, $\hat{\eta}_{\tau_i}^{[0]}$ with offset values.
- for each distribution parameter $\theta_k, k = 1, \ldots, 4$ specify a set of base-learners: i.e. for parameter θ_k by $h_{k1}(\cdot), \ldots, h_{kp_k}(\cdot), k = 1, \ldots, 4$, where p_k is the cardinality of the set of base-learners specified for θ_k

Boosting in multiple dimensions

For $m = 1$ to m_{stop}:

(1) For $k = 1$ to 4:

(a) Compute negative partial derivatives $-\frac{\partial}{\partial \eta_{\theta_k}} \rho(y_i, \boldsymbol{\eta})$ and plugging in the current estimates $\hat{\eta}_i = (\hat{\eta}_{\mu_i}^{[m-1]}, \hat{\eta}_{\sigma_i}^{[m-1]}, \hat{\eta}_{\nu_i}^{[m-1]}, \hat{\eta}_{\tau_i}^{[m-1]})$:

$$\boldsymbol{u}_k^{[m]} = \left(\frac{\partial}{\partial \eta_{\theta_k}} \rho(y_i, \boldsymbol{\eta}) \right)_{i=1,\ldots,n} \Big|_{\eta = \hat{\eta}_i^{[m-1]}}$$

(b) Fit the negative gradient vector $\boldsymbol{u}_k^{[m]}$ to each of the base-learners contained in the set of base-learners specified for the predictor η_{θ_k} in the initialization.

(c) Select the component j^* that best fits the negative partial-derivative vector according to the least squares criterion, i.e. select the base-learner h_{kj^*} defined by:

$$j^* = \underset{1 \leq j \leq p_k}{\operatorname{argmin}} \sum_{i=1}^{n} (u_{ik}^{[m_k]} - h_{kj}(\cdot))^2$$

(d) Calculate the possible reduction in the *empirical loss* as follows:

$$\Delta \rho_k = \sum_{i=1}^{n} \rho(y, \hat{\eta}_{\theta_k}^{[m-1]} + \nu \cdot \hat{h}_{kj^*}^{[m]}(\cdot))$$

(2) Depending on the value of the loss reduction

$k^* = \operatorname{argmin}_k(\Delta \rho_k), \quad k = 1, \ldots, 4$ execute only the best step:

$$\hat{\eta}_{\theta_{k^*}}^{[m]} = \hat{\eta}_{\theta_{k^*}}^{[m-1]} + \nu \cdot \hat{h}_{k^* j^*}^{[m]}(\cdot)$$

(3) Set $\hat{\eta}_{\theta_k}^{[m]} := \hat{\eta}_{\theta_k}^{[m-1]}$ for all $k \neq k^*$.

5.2 Noncyclical fitting via outer loss

Choosing base-learners and parameters on two different optimization criteria is not ideal. A better solution would be to find one criterion that can measure the improvement over all base-learners and distribution parameters together. Such a behavior can be achieved if the complete selection is based on the *empirical loss*.
An schematic overview of this process can be seen in Figure 10. The gradient is now only used to compute the base-learner $\hat{h}_{11}(\cdot), \ldots, \hat{h}_{1p_1}(\cdot), \hat{h}_{21}(\cdot), \ldots\ldots, \hat{h}_{4p_4}(\cdot)$.
In the second step the possible improvement in the risk is calculated for each base-learner of every distribution parameter and only the overall best base-learner is actually updated. The last step is identical to the fitting method via the *inner* loss, the chosen predictor gets updated and written in the gradients.
Instead of the using the *inner* loss (RSS) the whole selection is done based on the *outer* loss (empiric loss) and the method is named accordingly. Algorithm 5 shows the fitting method via *outer* loss.
This approach has a lot of similarities to coordinate descent (compare for example Wright [2015]). Coordinate descent is a general purpose optimization algorithm, in which, in every step, exactly one component is updated. Similar to our *outer* loss boosting approach, the stepsize as well as the direction can be determined by the gradient.
Coordinate descent algorithms are popular due to their convenience and easy expandability, e.g. to include bounds for coefficients.
Doing the complete selection process with the outer loss takes more computing time, because, additionally to the gradients, which have to be calculated anyway, the loss reduction for *every* base-learner has to be calculated as well.
We introduced two different ways of noncyclical fitting for component-wise gradient boosting in multiple dimensions, both having the major advantage of easier hyperparameter optimization in comparison to the cyclical algorithm. The performance of noncyclical fitting is evaluated in Chapter 5.3 and the performance in regards of stability selection is evaluated in Chapter 6.

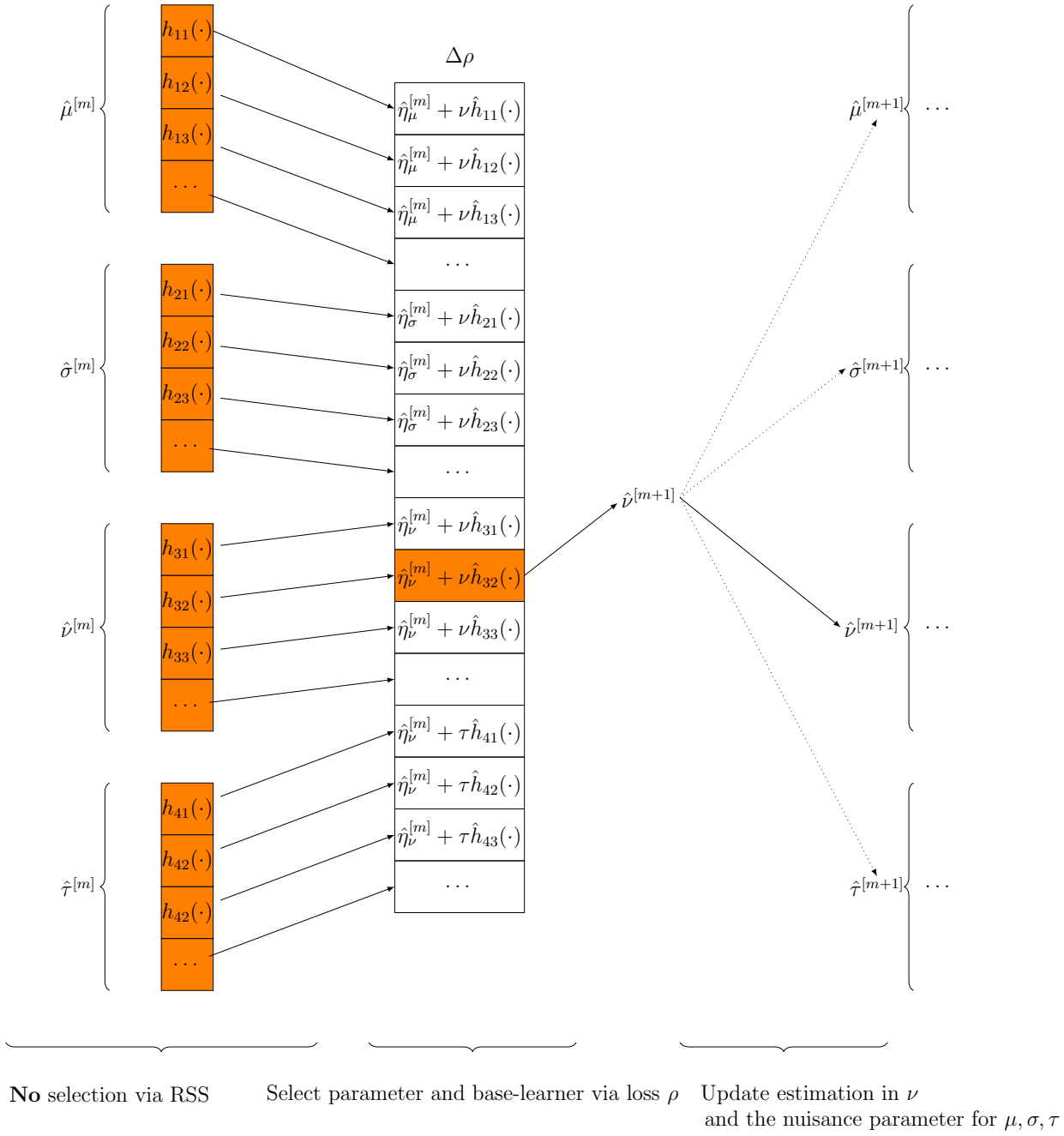

Figure 10: Noncyclical fitting algorithm – base-learner selection via *outer* loss

Algorithm 5 Noncyclical component-wise gradient boosting in multiple dimensions – via outer loss

Initialize

- Initialize the additive predictors $\hat{\eta}_{\mu_i}^{[0]}$, $\hat{\eta}_{\sigma_i}^{[0]}$, $\hat{\eta}_{\nu_i}^{[0]}$, $\hat{\eta}_{\tau_i}^{[0]}$ with offset values.
- for each distribution parameter $\theta_k, k = 1, \ldots, 4$ specify a set of base-learner: i.e. for parameter θ_k by $h_{k1}(\cdot), \ldots, h_{kp_k}(\cdot), k = 1, \ldots, 4$, where p_k is the cardinality of the set of base-learnerss specified for θ_k

Boosting in multiple dimensions

For $m = 1$ to m_{stop}:

(1) For $k = 1$ to 4:

(a) Compute negative partial derivatives $-\frac{\partial}{\partial \eta_{\theta_k}} \rho(y_i, \boldsymbol{\eta})$ and plugging in the current estimates $\hat{\boldsymbol{\eta}}_i^{[m-1]} = (\hat{\eta}_{\mu_i}^{[m-1]}, \hat{\eta}_{\sigma_i}^{[m-1]}, \hat{\eta}_{\nu_i}^{[m-1]}, \hat{\eta}_{\tau_i}^{[m-1]})$:

$$\boldsymbol{u}_k^{[m]} = \left(\frac{\partial}{\partial \eta_{\theta_k}} \rho(y_i, \boldsymbol{\eta}) \right)_{i=1,\ldots,n} \Big|_{\eta = \hat{\eta}_i^{[m-1]}}$$

(b) Fit the negative gradient vector $\boldsymbol{u}_k^{[m]}$ to each of the base-learners contained in the set of base-learners specified for the predictor η_{θ_k} in the initialization.

(c) Calculate the possible reduction in the *empiric loss* for every base-learner as follows:

$$\Delta\rho_{kj} = \sum_{i=1}^{n} \rho(y, \hat{\eta}_{\theta_k}^{[m-1]} + \nu \cdot \hat{h}_{kj}^{[m]}(\cdot)) \qquad j = 1 \ldots, p_k \quad k = 1, \ldots, 4$$

(2) Depending on the value of the loss reduction

$\Delta\rho_{k^*j^*} = \text{argmin}_{kj}(\Delta\rho_{kj}) \quad k = 1, \ldots, 4 \quad j = 1, \ldots, p_k$

execute only the best step:

$$\hat{\eta}_{\theta_{k^*}}^{[m]} = \hat{\eta}_{\theta_{k^*}}^{[m-1]} + \nu \cdot \hat{h}_{k^*j^*}^{[m]}(\cdot)$$

(3) Set $\hat{\eta}_{\theta_k}^{[m]} := \hat{\eta}_{\theta_k}^{[m-1]}$ for all $k \neq k^*$.

5.3 Comparison with cyclical fitting

Regarding these new algorithms, several questions should be evaluated:

(a) Do both new noncyclical fitting methods (via *inner* and *outer* loss) converge to the same solution as GAMLSS [Rigby et al., 2008]? And how much differ the estimated parameters of the four different fitting variants.

(b) How is the convergence speed? Does one method converge faster and consequently needs fewer iterations?

(c) How do the runtimes of the new algorithms compare to the cyclical fitting approach? It is likely that fitting will be slower, as much more gradients and possible model improvements have to be computed. Yet the optimization of the hyperparameter might be much faster.

(d) Does stability selection work better with the new fitting methods? This question is evaluated in Chapter 6.

To answer these question it is useful to conduct a small simulation study. We start with a simulation design that is as simple as possible.
The response y_i is drawn from a normal distribution $N(\mu_i, \sigma_i)$ with μ_i and σ_i composed of 4 covariates each. The covariates are independently drawn from a uniform distribution between -1 and 1: $U(-1, 1)$. Two covariates x_3 and x_4 are shared between both μ and σ and will be informative for both parameters.

To summarize the data generation process for a sample size of $n = 500$:

1. For each $x_1, \ldots, x_6$ draw a sample of size 500 from $U(-1, 1)$.
2. Calculate $\mu_i = x_{1i}\beta_{1\mu} + x_{2i}\beta_{2\mu} + x_{3i}\beta_{3\mu} + x_{4i}\beta_{4\mu}, \quad i = 1, \ldots, 500.$
3. Calculate $\sigma_i = x_{3i}\beta_{1\sigma} + x_{4i}\beta_{2\sigma} + x_{5i}\beta_{3\sigma} + x_{6i}, \beta_{4\sigma} \quad i = 1, \ldots, 500.$
4. Draw y_i from $N(\mu_i, \sigma_i)$ for $i = 1, \ldots, 500$.

5.3.1 Convergence to maximum likelihood solution

It can be shown that the solutions of the cyclical fitting algorithm converge to the same solutions as obtained by maximizing the penalized likelihood [Mayr et al., 2012a].
The question is, if both boosting methods, via *inner* and *outer* loss, also converge to the same solution. The results of a simulation with 100 runs is shown in Figure 11. All four methods seem to converge to the correct solution. Coefficients are

	μ				σ			
	x_1	x_2	x_3	x_4	x_3	x_5	x_5	x_6
cyclic	0.42	0.48	0.44	0.72	0.29	0.31	0.35	0.41
outer	0.40	0.47	0.42	0.71	0.30	0.32	0.36	0.42
inner	0.40	0.47	0.42	0.71	0.30	0.32	0.36	0.42
GAMLSS	0.39	0.48	0.40	0.59	0.31	0.32	0.36	0.41

Table 3: MSE (in 10^{-2}) for the the parameter estimations, based of $B = 100$ simulation runs. All algorithms were fitted until convergence.

underestimated by the boosting algorithms, this is *shrinkage* effect of boosting that was talked about in Chapter 1.2.1.
The *mean squared error* (MSE) of the parameter estimates is shown in Table 3. The estimation of μ seems to be better for boosting with *inner* or *outer* loss, compared to the cycling algorithm. In σ the cyclical algorithm is slightly better than both noncylical boosting methods. Yet, the differences are very small (note the factor of 10^{-2} in the table) and between *inner* and *outer* loss no differences are found at all.

5.3.2 Convergence speed

We have seen that all three boosting methods will converge to the penalized maximum likelihood solution reasonably well. The second question is then, how fast this convergence is. In Figure 12 the risk is plotted against the number of iterations. Here additional non-informative variables were added to the model. Four settings of are considered, $0, 50, 25$ and 500 additional non-informative covariates. Considering the $n = 500$ data points generated, both $p = 250$ and $p = 500$ are $p > n$ situations, because the number of base-learners is doubled as we have two distribution parameters. The shown values are the mean risks of 100 runs on different simulated datasets. *Outer* and *inner* loss boosting have the exact same rate in all four settings. Compared to the *cyclical* algorithm the convergence is faster in the first 500 iterations. After more than 500 iteration the reduction is the same for all three methods. The margin between the *cyclical* and both *noncyclical* algorithms decreases with a larger number of noise variables.

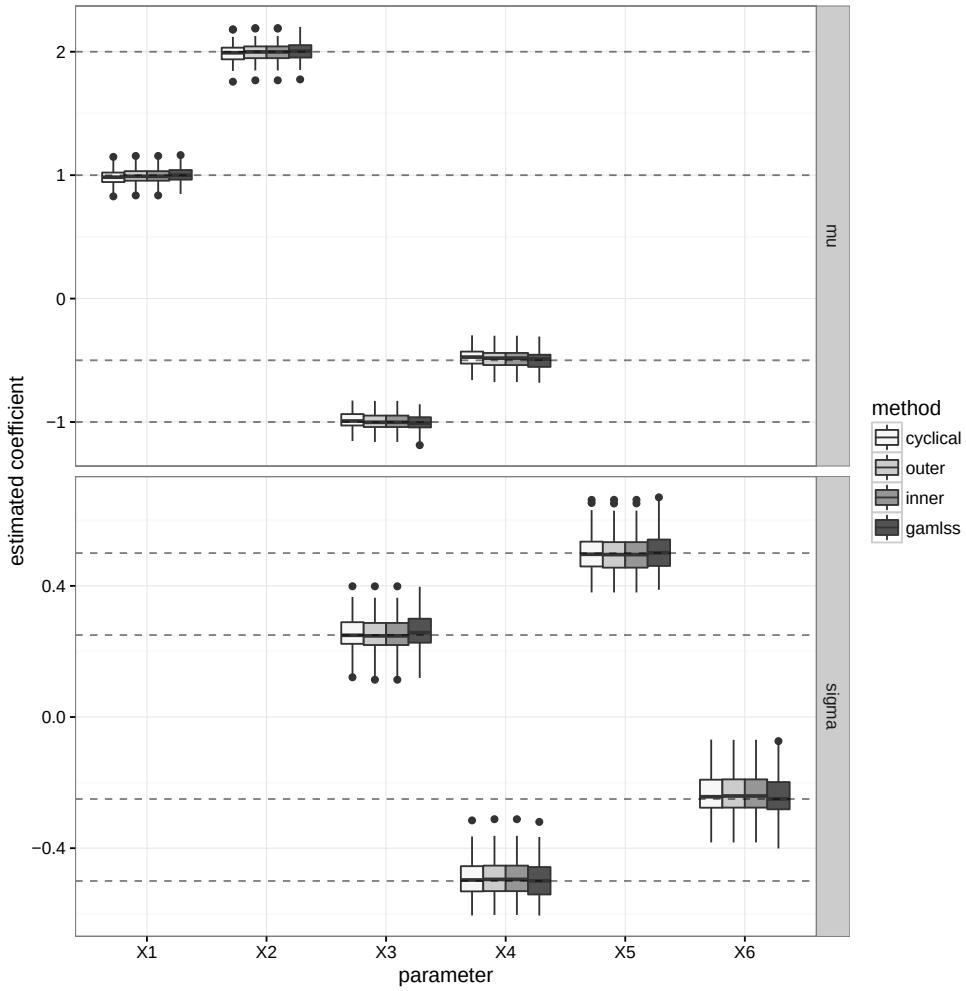

Figure 11: Distribution of coefficient estimates from $B = 100$ simulation runs. The dashed lines shows the true parameters.

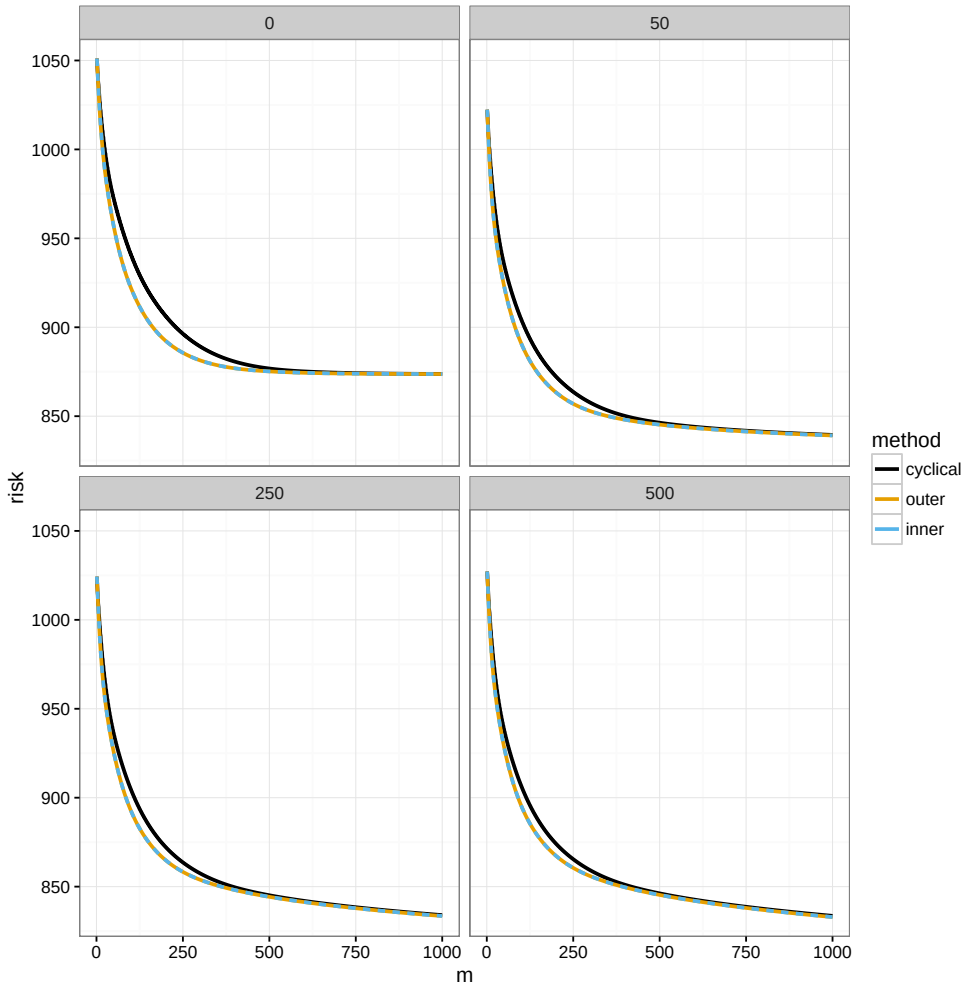

Figure 12: Speed of convergence for different number of non-informative variables: $0, 50, 250, 500$ and different fitting algorithms.

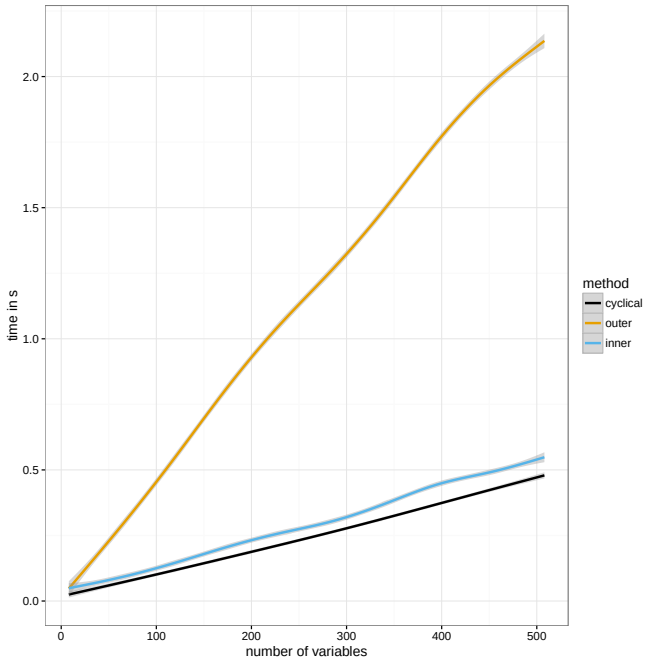

Figure 13: Mean runtime for initialization plus 10 boosting iterations for different numbers of covariates and the three fitting algorithms.

5.3.3 Runtime analysis

The main calculation effort for the algorithms is the base-learner selection, which is different for all three methods. The *cyclical* algorithm has to calculate no potential loss reductions at all. In case of the *inner* loss algorithm, base-learner selection and following possible loss reduction has to be calculated only for the best-fitting base-learner for each distribution parameter in every step of the algorithm. For the *outer* loss version, the possible reduction of the loss has to be calculated for every base-learner in every step of the algorithm, which should takes longer than the other versions.
To estimate the runtime, all algorithms were run 50 times with the settings introduced in Chapter 5.3, but only the first 10 iterations were performed. The mean runtime in seconds was measured for 0 up to 500 additional non-informative base-learners. The result of this small benchmark study can be seen in Figure 13. The mean runtime of the initialization and the first 10 boosting iterations was smoothed and a 0.95 confidence interval was calculated.
The *cyclical* fitting algorithm is the fastest, which is not surprising, because of the fewer number of comparisons that are needed. The *inner* fitting algorithms is a

bit slower but still minimally faster than the *outer* loss method, for a very small number of base-learners. When increasing the number of covariates, the runtime for the *outer* fitting method increases way stronger than for the *inner* and *cyclical* variant.

5.4 Implementation in gamboostLSS

Both new *noncyclical* fitting algorithms are implemented in `gamboostLSS` [Hofner et al., 2015a], so that all advantages of the framework can be used.
We shortly recapitulate the implementation of `gamboostLSS`, which was explained in Chapter 2.3. A `gamboostLSS` object is a list of multiple `mboost` [Hothorn et al., 2015] objects. These are initialized with loss functions and gradients of the loss, based on the used distribution. For example, when fitting a `gamboostLSS` model with a normal distribution, one `mboost` object uses the negative log likelihood of the normal distribution, with the respective gradient of μ, and a second `mboost` object with the same loss function and gradient of σ (compare Equation 10). The main fitting process has to update one `mboost` object, which essentially means to add a new base-learner to the model or update an existing one, then update the new estimated parameter in *every* `mboost` object and update the gradients. This main functionality is captured in a single function, `iBoost`. For an in-depth introduction to the `gamboostLSS` package, see the tutorial by Hofner et al. [2015c].
To achieve a noncyclical fitting algorithm, only the main `iBoost` function has to be replaced, almost all functionality from the `gamboostLSS` package can be used with minimal adaptions. Two new fitting functions `iBoost_inner` and `iBoost_outer` are integrated as an alternative to the *cyclical* `iBoost` function. The R code of both functions is found in the Appendix, Section 10.3.
The working implementation of the new fitting methods can be found on the github page of the `gamboostLSS` package in a separate branch called *noncyclical_fitting*.

5.4.1 Implementation of the inner fitting algorithm

The implementation of the *inner* fitting algorithm, see Chapter 5.1, is pretty straightforward. For each `mboost` object one base-learner is added in a separate environment. The selection of one base-learner for each distribution parameter is performed by the internal `mboost` mechanism, which is the selection via the RSS. Then, the improvement of the loss can be easily calculated for each selected base-learner and only the update with the highest reduction is actually performed. The rest of the fitting process is similar to the *cyclical* algorithm, meaning, that the updated estimate is written as a nuisance parameter in all other `mboost` objects and the gradients are recalculated.

5.4.2 Implementation of the outer fitting algorithm

The implementation of the *outer* loss fitting algorithm, see Chapter 5.2, is more complex than the previous one. In the *outer* loss fitting algorithm, the selection of base-learners over *all* distribution parameters is conducted via the loss function. The main obstacle is, that the base-learner selection within an `mboost` object is always done with the RSS criterion and it is not intended by the `mboost` developers to add or update a different base-learner that the one that minimizes the RSS criterion. In order to still have access to the large number of supported base-learners (compare Table 1), as well as distributions (compare Table 2), that the `mboost` framework offers, the usage of `mboost` objects in the fitting process is crucial.
Instead of just updating one `mboost` object, all possible base-learner objects in one step $\hat{h}_{11}(\cdot), \ldots, \hat{h}_{1p_1}(\cdot), \hat{h}_{21}(\cdot), \ldots\ldots, \hat{h}_{4p_4}(\cdot)$ have to be constructed explicitly and their possible loss reduction has to be calculated. The best base-learner is then added to its corresponding `mboost` object, essentially doing one iteration of the `mboost` fit manually. This process is further complicated by the fact, that this functionality has to be implemented differently depending on the used class of base-learners. For example have linear `cwlin` base-learner a different internal structure than linear `bols` base-learner. This results in a (currently) slower implementation as seen in Figure 13.

6 Stability selection in a simulation study

The main question in this thesis was the usage of *stability selection* with componentwise gradient boosting in multiple dimensions. The theoretical problem arising from the usage of a *cyclical* fitting algorithm in combination with the *stability selection* was examined in Chapter 4. In this chapter, these theoretical considerations were evaluated in a simulation study.
Different simulation settings were constructed to find differences in the behavior of the algorithms. First, three different distributions were considered:

(1) The normal distribution with two parameters, mean, μ_i and standard deviation, σ_i.

(2) The negative binomial distribution with two parameters, first the location μ_i and second σ_i, the overdispersion parameter.

(3) The zero-inflated negative binomial (ZINB) distribution with three parameters: μ_i, σ_i and ν_i, with μ_i, σ_i identical to the negative binomial distribution and ν_i as the zero-inflation probability of the distribution.

Second, different partitioning of informative covariates between distribution parameters are evaluated:

(A) Balanced case: For normal and negative binomial distribution, both μ_i and σ_i consist of four informative covariates, where two are shared. In case of the ZINB distribution, each parameter has three informative covariates each sharing on with the other two parameters.

(B) Unbalanced case: For normal and negative binomial distribution, μ_i consists of five informative covariates, while σ_i has only one and none are shared between the two parameters. For the ZINB distribution, μ_i has five informative variables as well, σ_i has two informative variables, sharing one with μ_i and one with ν_i. Lastly ν_i has only one covariate it shares with σ_i.

To summarize this for a total of six informative variables, $x_1, \ldots, x_6$:

(1A, 2A)
- $\mu_i = \beta_{1\mu}x_{1i} + \beta_{2\mu}x_{2i} + \beta_{3\mu}x_{3i} + \beta_{4\mu}x_{4i}$
- $\sigma_i = \beta_{3\sigma}x_{3i} + \beta_{4\sigma}x_{4i} + \beta_{5\sigma}x_{5i} + \beta_{6\sigma}x_{6i}$

(1B, 2B)
- $\mu_i = \beta_{1\mu}x_{1i} + \beta_{2\mu}x_{2i} + \beta_{3\mu}x_{3i} + \beta_{4\mu}x_{4i} + \beta_{5\mu}x_{5i}$
- $\sigma_i = \beta_{6\sigma}x_{6i}$

(3A)
- $\mu_i = \beta_{1\mu}x_{1i} + \beta_{2\mu}x_{2i} + \beta_{3\mu}x_{3i}$

- $\sigma_i = \beta_{3\sigma} x_{3i} + \beta_{4\sigma} x_{4i} + \beta_{5\sigma} x_{5i}$
- $\nu_i = \beta_{1\nu} x_{1i} + \beta_{5\nu} x_{5i} + \beta_{6\nu} x_{6i}$

(3B)
- $\mu_i = \beta_{1\mu} x_{1i} + \beta_{2\mu} x_{2i} + \beta_{3\mu} x_{3i} + \beta_{4\mu} x_{4i} + \beta_{5\mu} x_{5i}$
- $\sigma_i = \beta_{5\sigma} x_{5i} + \beta_{6\sigma} x_{6i}$
- $\nu_i = \beta_{6\nu} x_{6i}$

The simulation process is identical to Chapter 5.3. To evaluate the quality of stability selection, two criteria have to be considered. First, stability selection should include as many informative variables as possible, ideally all of them. This is the *true positive rate*, or the number of *true positives* (TP). Secondly, as few non-informative variables as possible should be wrongly classified as informative, which is the *false positive rate*, or the number of *false positives* (FP).
In the best case all informative variables are found and none are wrongly identified as informative.
AS the true positive rate should be maximized and the false positive rate should be minimized, there is a trade-off between both criteria. If the model selection is done more *conservatively*, the false positive rate will decrease as well as the true positive rate. On the other hand, if it is easy to include variables in the model, it is more likely to find the informative variables, but also to include non informative variables in the model. Consequently, large models will, in tendency, have a high true *and* false positive rate and small models a low true *and* false positive rate.
Considering stability selection, the most obvious criterion to steer the false and the true positive rate is the threshold π_{thr}. If the threshold is high, less variables are included in the model, so the true positive rate will (most likely) decrease, as well as the false positive rate.
To evaluate the algorithms depending on the settings of stability selection, different values for the number of variables to include in the model q as well as the threshold π_{thr} are considered.
A third factor to consider, is the number of noise variables in the model. The more noise variables in the model, the harder is, to identify the informative variables.
For the simulation four possible values of q are considered: $8, 15, 25, 50$. q should be at least as large as the number of informative variables. The number of covariates is $p = 50, 250$ or 500, minus the number of informative variables. It should be taken into consideration, that the actual number of possible covariates is p times the number of distribution parameters, because each covariate can be included in one or more additive predictors.
The threshold is varying between 0.55 and 0.99 in steps of 0.01, this small increases can be easily calculated, because the stability selection algorithm does not need to be re-run to adjust the threshold.

To visualize the results of the simulation for a given q, the progress of true and false positives are plotted against the threshold for different values of q and p. True and false positives are summed up over all distribution parameters. Visualizations for every single distribution parameter can be found in the Appendix.

6.1 Results for the normal distribution setting

In Figure 14 and 15 the results for the balanced and unbalanced case with normal distribution are displayed (case 1A and 1B).
As mentioned before, it can be observed that with increasing threshold π_{thr}, the number of true positives as well as the number of false positives declines. This is present in every combination of q and p.
In the balanced case (Figure 14) a higher number of true positives of the noncyclical algorithm can be seen, for most simulation settings. Especially for smaller q values ($q = 8, 15$) the true positive rate is always higher than the cyclical variant. For higher q values the margin decreases and for the highest settings both methods have approximately the same progression over π_{thr}, with slightly better results for the cyclical algorithm. Overall, the number of true positives increases with a higher value of q. Hofner et al. [2015b] found similar results for boosting with one dimensional prediction functions, but also showed that the true positive rate decreases again after a certain value of q. This could not be verified for the multidimensional case.
The false positive rate is extremely low for both methods, especially in the high-dimensional settings. The noncyclical fitting method has a constantly smaller or identical false positive rate, the difference between both reduces for higher π_{thr}, as expected. For all settings the false positive rate reaches zero for a threshold higher than 0.9. The setting with the highest false positive rate is $p = 50$ and $q = 25$, so, a low dimensional case with a relatively high threshold. This is also the only settings where on average all 8 informative variables are found (for a threshold of 0.55).

In the unbalanced case, Figure 15 shows similar results. Yet, the number of false positives for the noncyclical variant is higher in almost all settings. The main difference between the balanced and the unbalanced case is that whereas the true positives for the $p = 25, q = 25$ setting was almost identical between both methods, now the noncyclical variant is dominating. On the other hand, the high-dimensional case with a small q ($p = 500, q = 8$) both fitting methods have about the same true positive rate for all possible π_{thr}.

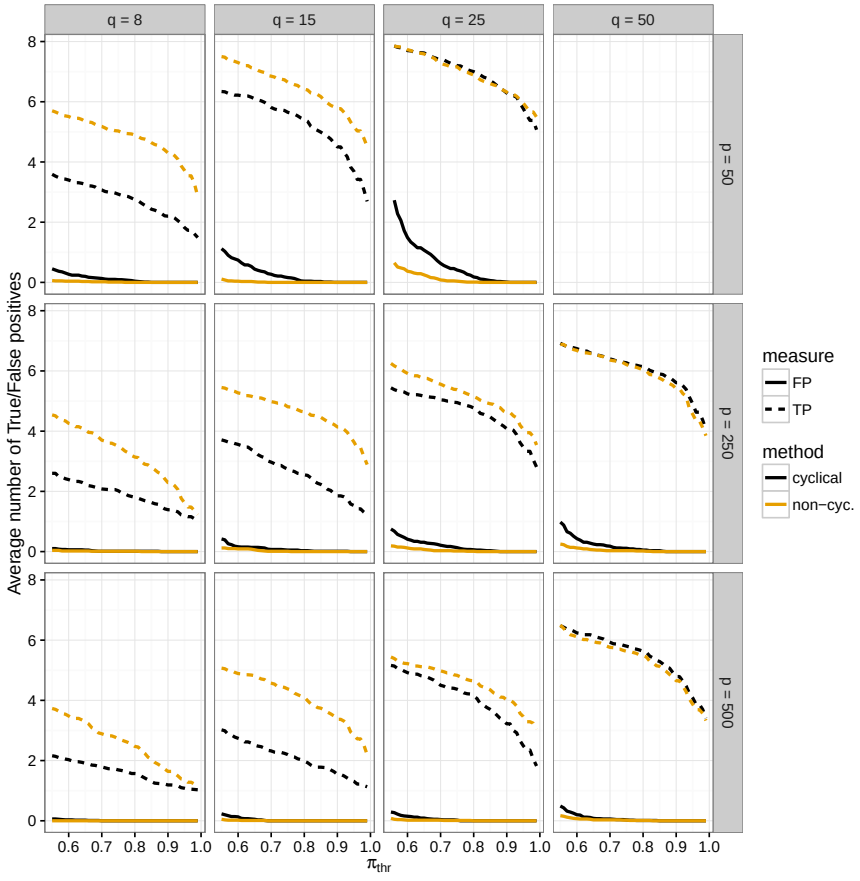

Figure 14: Stability selection: Normal distribution and balanced number of informative variables. The different methods are color-coded. The number of true positives is shown as dashed lines and the false positives as continuous lines.The rows represent the different values of q and the columns the values of p. Case (1A).

In summary, it can be seen, that the new noncyclical way is better in identifying the true positives or at least performs similar to the cyclical method for all shown settings. On the other hand, the (although over all very small) false positive rate is less or identical to the cyclical method. Some simulation showed minor different results for cases when the scale component σ_i was higher compared to the location μ_i, see Table 5 and 6 in the Appendix. There, the cyclical variant achieved slightly better performance (in the true positives) at high p and q values than the noncyclical variant. The results separated for each distribution parameter can be found in the Appendix in Figure 23 for the balanced case and Figure 24 for the unbalanced.

6.2 Results for the negative binomial distribution

In the balanced case of the negative binomial distribution (Figure 16), the number of true positives is almost identical for cyclical and noncyclical fitting in all settings. Generally, the number of true positives is quite high overall. Almost all settings are between 6 and 8, with an optimum of 8 true positives, except for the cases with a very small number of variables to include $q = 8$. This is consistent with the results for stability selection with one dimensional boosting [Hofner et al., 2015b]. The number of false positives in the noncyclical variants is smaller or identical to the cyclical variant in *all* tested settings.
The differences between both methods get smaller for the unbalanced case (Figure 17). For smaller values of $q = 8$ and $q = 15$, the noncyclical variant has still a higher number of true positives for all evaluated thresholds π_{thr}. For larger values of q the true positives are almost identical. The larger number of false positives for the noncyclical variant, as seen in the balanced case, is less pronounced, but still evident for smaller values of q.

Overall the noncyclical variant seems to be better for the here evaluated distributions (normal and negative binomial). The noncyclical variant has a higher number of true positives and a fewer false positives. In most cases these differences are reduced for larger values of the number of variables to include q and a larger threshold π_{thr}.

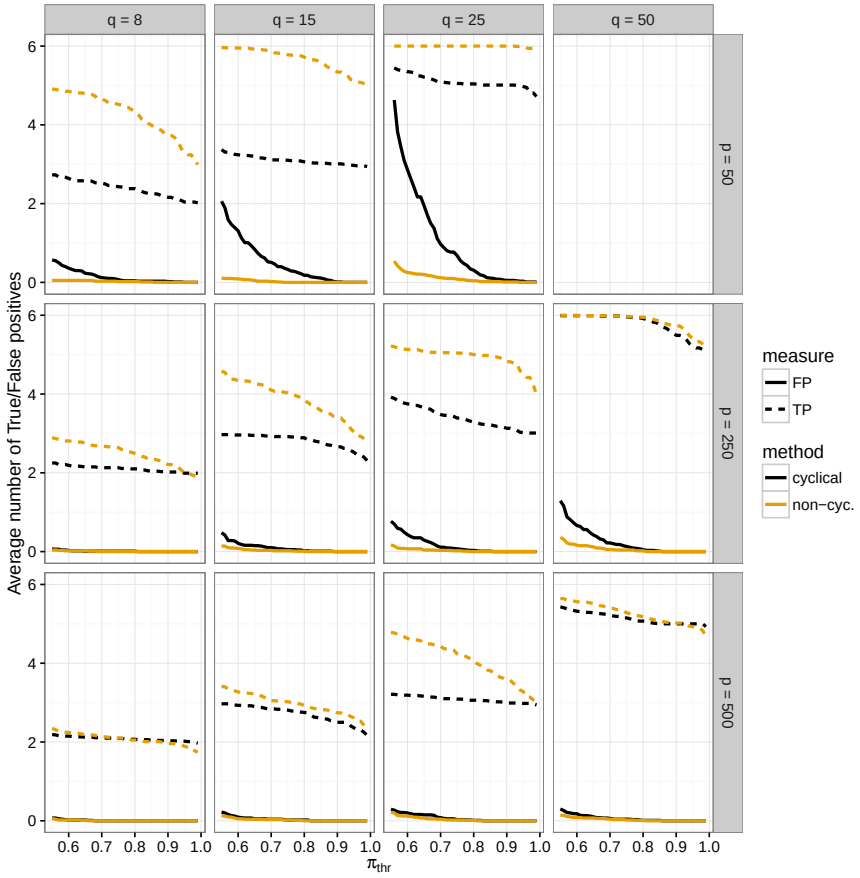

Figure 15: Stability selection: Normal distribution and unbalanced number of informative variables. The different methods are color-coded. The number of true positives is shown as dashed lines and the false positives as continuous lines.The rows represent the different values of q and the columns the values of p. Case (1B).

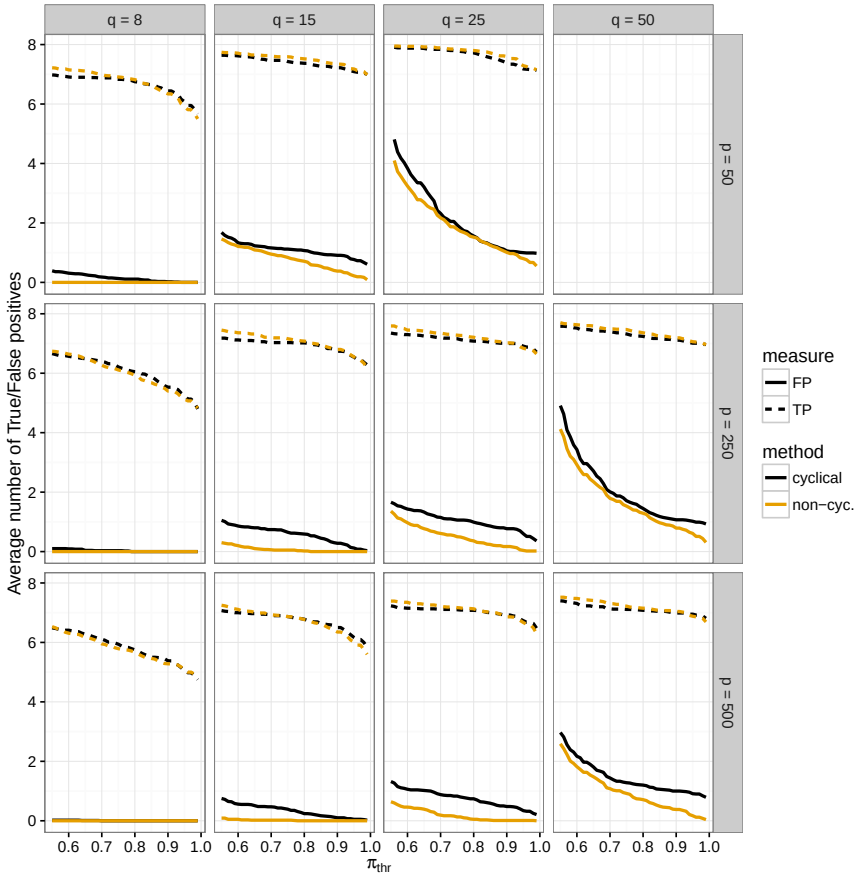

Figure 16: Stability selection: Negative binomial distribution and balanced number of informative variables. The different methods are color-coded. The number of true positives is shown as dashed lines and the false positives as continuous lines.The rows represent the different values of q and the columns the values of p. Case (2A).

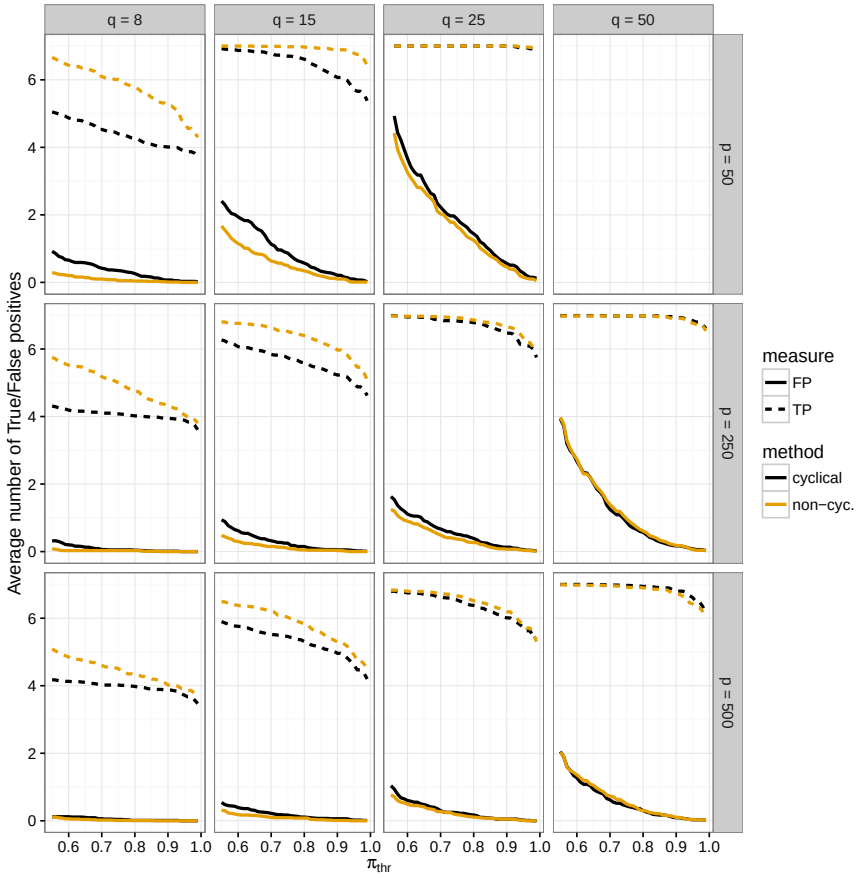

Figure 17: Stability selection: Negative binomial distribution and unbalanced number of informative variables. The different methods are color-coded. The number of true positives is shown as dashed lines and the false positives as continuous lines.The rows represent the different values of q and the columns the values of p. Case (2B).

6.3 Results for the zero-inflated negative binomial distribution

The third considered distribution in our simulation setting is the the zero-inflated negative binomial (ZINB) distribution. Instead of only two distribution parameters, like in the case of normal or negative binomial distribution, the ZINB distribution features three parameters that have to be fitted.
In Figure 18, the results for the balanced case (3A), are visualized. The tendency of a larger number true positives in the noncyclical variant, which could be examined for both two-parametric distributions, is not present. For all settings, except for high dimensional settings with a low number of variables to include, $p = 250, 500$ and $q = 50$, the cyclical variant has a higher number of true positives. Additionally, the number of false positives is constantly higher for the noncyclical variant.
In the unbalanced case (Figure 19) the number of true positives is considerably smaller compared to all other simulated settings. Especially in the high dimensional cases ($p = 250, 500$), not even half of all informative covariates are found. In settings with smaller q the number of true positives can be lower than two. Here, both algorithms have approximately the same number of true positives for all settings, in cases with a very low number of variables to include $q = 8$ or a very high number, $q = 50$, the noncyclical algorithm is slightly better. The number of false positives is, especially compared with the number of true positives, very high. For a lot of settings, more than half of the included variables were non-informative. The number of false positives is higher for the noncyclical case. The difference is especially present in settings with a high number of variables to include and a low threshold, but these are settings, that still have the highest numbers of true positives.
Altogether the trend from the simulated two-parameter distributions is not present in the three-parameter setting. The noncyclical algorithm is worse or identical in both true and false positives for almost all tested settings compared to the cyclical variant. Other distributions with three parameters, like the Student's t-distribution, gave similar results and can be found in the Appendix, Chapter 10.2 Figure 27 and 28.
The bad performance for three-parameter distributions of the noncyclical algorithm, may be due to its additional flexibility. With the more complex distributions, a similar looking density can be achieved with different parameter settings. For example in a zero-inflated negative binomial setting, a small location may be hard to distinguish from a large zero-inflation parameter. This may result in the noncyclical variant trying to fit very different models on each subsample and consequently selecting non-informative variable, or covariates for the wrong distribution parameter and missing informative ones. This behavior is shortly discussed in the outlook Chapter 8.2.2.

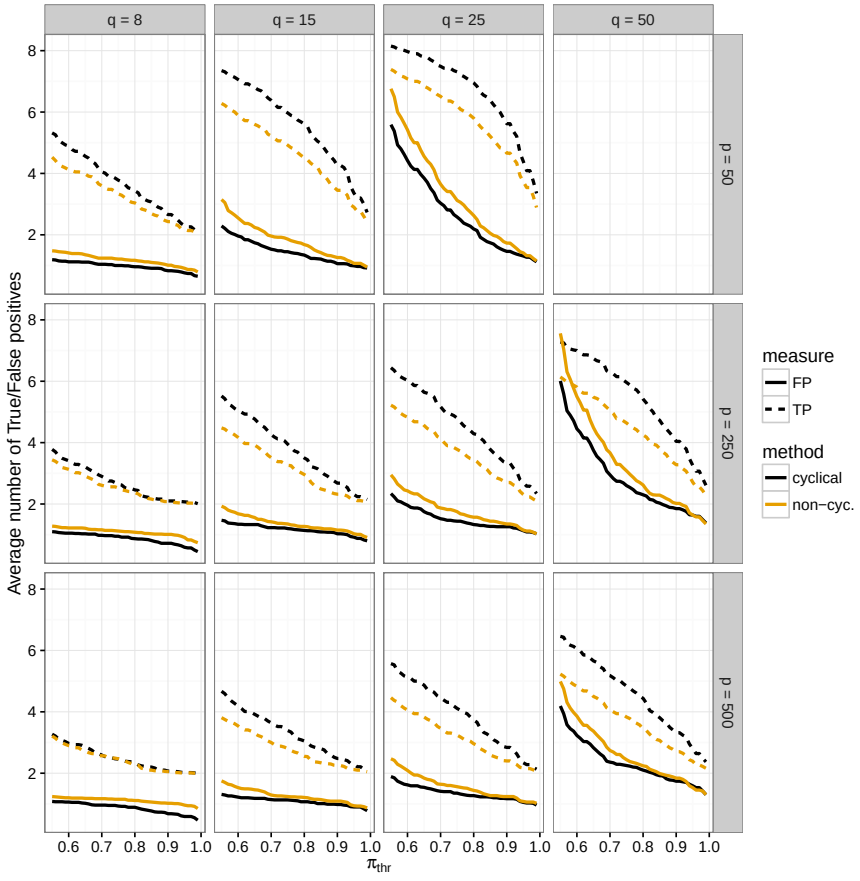

Figure 18: Stability selection: Zero-inflated negative binomial distribution and balanced number of informative variables. The different methods are color-coded. The number of true positives is shown as dashed lines and the false positives as continuous lines.The rows represent the different values of q and the columns the values of p. Case (3A).

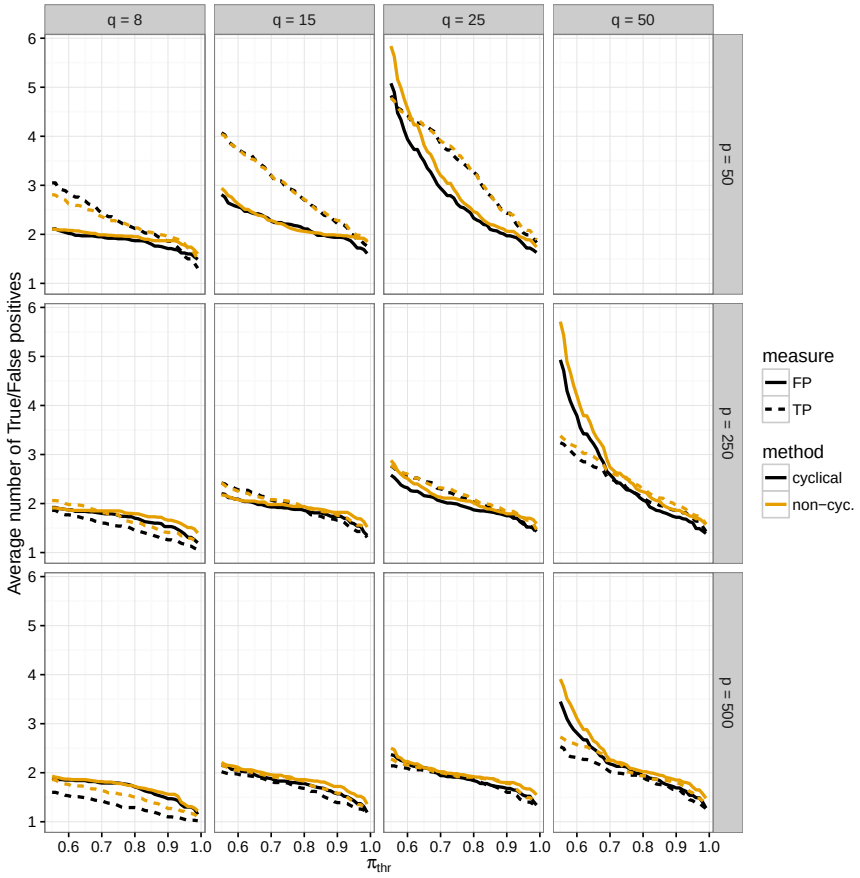

Figure 19: Stability selection: Zero-inflated negative binomial distribution and unbalanced number of informative variables. The different methods are color-coded. The number of true positives is shown as dashed lines and the false positives as continuous lines.The rows represent the different values of q and the columns the values of p. Case (3B).

7 Analysis of the seabird abundance

In this chapter, the analysis of the common eider population from Chapter 1.3 is continued. To recap, the population of common eider in Nantucket Sound, Messachussets are estimated with a hurdle model because of the zero-inflation in the data. The population model was estimated by boosting a GAM with binomial loss (see Chapter 1.3). In the second step, the number of birds in populated segments is estimated with a boosted GAMLSS model. Considering only observations larger than zero for the abundance model, out of originally 7751 observations 1930 remain. In Figure 20 it can be seen that the distribution is quite *long-tailed* and may exhibit *overdispersion*. So, instead of the *poisson* distribution, the *negative binomial* distribution was chosen for this model (compare Mullahy [1986]). Because we used the negative binomial distribution in a hurdle model, the distribution had to be truncated. As compared to zero-inflation models, the count distribution in hurdle models cannot predict zeros. A zero-truncated distribution can be generated from a count distribution $f_{\text{count}}(x, \mu, \sigma)$ with

$$\frac{f_{\text{count}}(x, \mu, \sigma)}{1 - f_{\text{count}}(0, \mu, \sigma)}, \quad \text{for } x > 0 \tag{17}$$

[Zeileis et al., 2007], both distribution parameters, mean *and* overdispersion, were regressed on the biophysical covariates (Table 4). Similar to the population model the covariates were split in linear and nonlinear base-learners. Because of the strong possibility for overdispersion, nonlinearity, large amount of possible effects the use a boosted GAMLSS model is adequate. To generate a zero-truncated negative binomial distribution for GAMLSS, the R package `gamlss.tr` [Stasinopoulos and Rigby, 2015] was used.

To reduce computation time, a rather large learning-rate of 0.3 was used. The optimal number of m_{stop} value(s) is found with cross-validating subsamples of 50% of the original data.

We used the *inner* version (Chapter 5.1) for noncyclical fitting, because the simulation study in Chapter 5.3 has shown only minimal differences between both noncyclical methods and the *inner* version is easier and faster to compute. This yields a m_{stop} value of 2231, split in $m_{\text{stop},\mu} = 1871$ and $m_{\text{stop},\sigma} = 336$. The sequence in which the distribution parameters were fitted is shown in Figure 21. The resulting model selected 46 out of 48 possible covariates in μ and 8 out of 48 in σ. This is (especially in μ) a way too large model, which suggests to use stability selection as an additional way to select variables for a sparser model.

The chosen parameters for stability selection were, numbers of variables to include of $q = 35$ and a per-familiy error-rate of 6. Using Equation (15) this results in a threshold of $\pi_{\text{thr}} = 0.9$. These settings were chosen similar to the originally by Smith et al. [2016] conducted analysis.

Variable	Abbr.	Type	Description
Day of year	time	T	Number of days from 31 December
Bathymetry	depth	S	Bottom depth relative to mean high water
Sediment grain size	meanphi	S	Sediment grain size
Sea floor surface area	SAR	-	Estimate of variability of the sea floor from bathymetry
Epibenthic tidal velocity (mean)	tidebmean	S	Average epibenthic tidal velocity during Finite-Volume Community Ocean Model
Epibenthic tidal velocity (standard deviation)	tidesd	S	standard deviation of epibenthic tidal based on monthly FVCOM structured grids
Water column stratification	strat	S	potential for seasonal thermal stratification calculated as the ratio of depth
Chlorophyll-a	chla	S	Geometric mean of monthly composite chlorophyll-a levels
Chromophoric dissolved	cdom	S	Geometric mean of monthly composite chromophoric material levels
Sea bottom temperature	SBT	ST	Sea bottom (epibenthic) temperature averaged from May to October for each year
Sea surface temperature	SST_m	ST	Monthly sea surface temperature from monthly FVCOM structured grid
Sea surface temperature Nov - Mar	SST_w	ST	Average winter (November through March) sea surface temperature
Sea surface temperature relative	SST_{rel}	ST	Difference between the monthly sea surface temperature of a segment and the average sea surface temperature of the entire study area
North Atlantic Oscillation Dec - Mar	NAO_w	T	Winter (December through March) North Atlantic Oscillation index
Distance to land	d2land	S	Distance to the nearest location of zero depth (from bathymetry)
Ferry route within 1 km	ferry	S	Indicator of whether a ferry route intersects a given segment
Winter 2004	w2004	T	Indicator comparing surveys occurring from November 2004 through April 2005 with those occurring from December 2003 through April 2004
Winter 2005	y2005	T	Indicator comparing surveys occurring from October 2005 through March 2006 with those occurring from December 2003 through April 2004
Easting	xkm	S	Distance between the easting of a segment center from the median easting of all segments.
Northing	ykm	S	Distance between the northing of a segment center from the median northing of all segments
Survey effort	obs_window	ST	Area of strip transect surveyed in a given segment on a given survey date

Table 4: Predictor descriptions, taken from Smith et al. [2016]

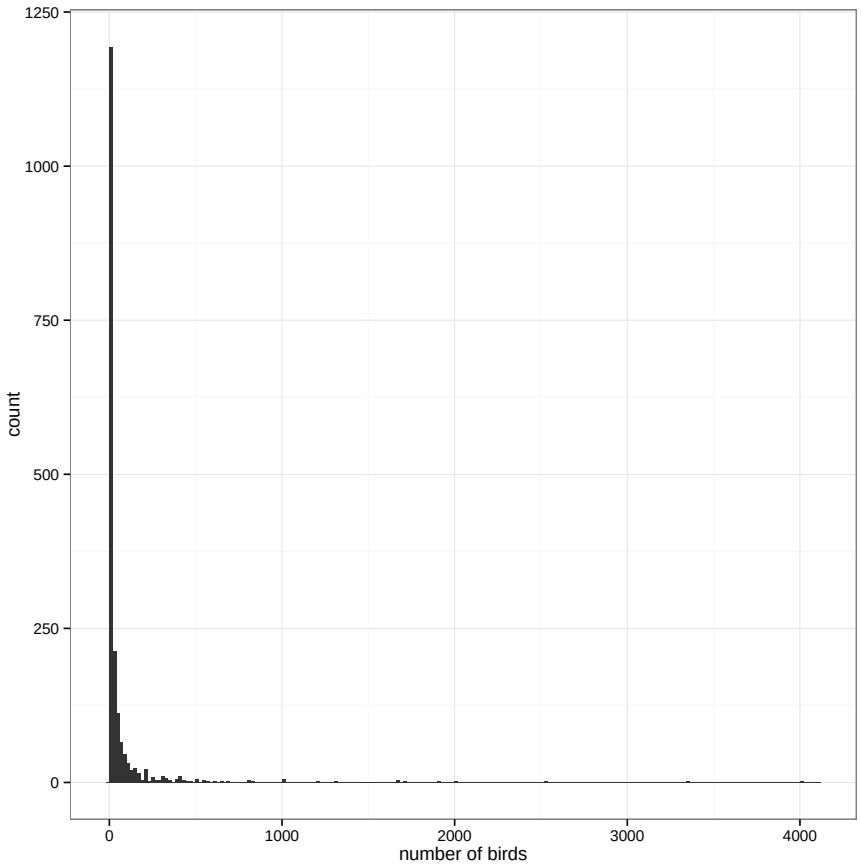

Figure 20: Histogram of bird counts larger zero. 28 observations in the range of 5000 – 30000 are omitted.

The result of stability selection can be seen in Figure 22. 10 effects were selected for the location: The intercept, the relative surface temperature (smooth), the chlorophyil-a material levels (smooth), the chromophoric material levels (smooth), sediment grain size (linear and smooth), sea floor surface area (smooth), the mean epidenthic tidal velocity (smooth) and the smooth spatial interaction as well as the two factors for the year and nearby ferry routes. For the overdispersion, 5 effects were selected: The sea surface temperatur (linear), the bathymetry (linear), the mean (smooth) and standard deviation (linear) of the epibenthic tidal velocity and the linear spatial interaction. Only the sediment grain size was selected linearly as well as nonlinearly in the model. For the location, all metric variables entered the model nonlinearly. For the overdispersion it was the other way round, the mean epibenthic velocity was selected as a smooth effect, all others were selected

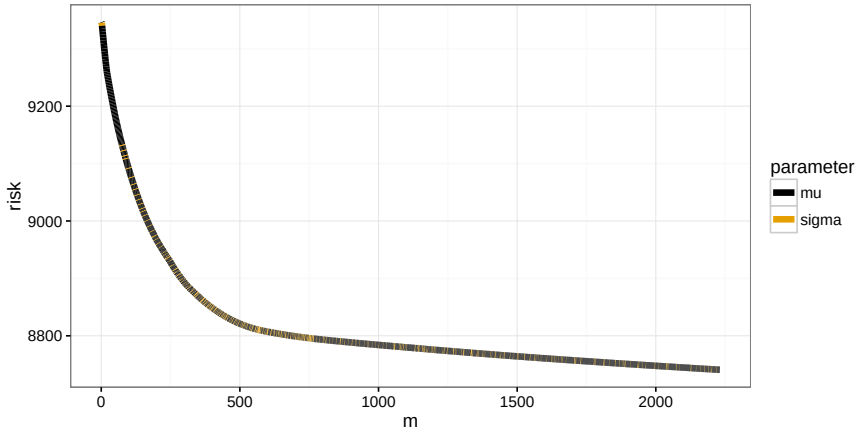

Figure 21: Convergence of the risk for the seabird abundance model. The color notes which parameter was updated in a step.

as linear effects.
If we compare the resulting model with Smith et al. [2016], the noncyclical model is larger in μ (10 effects, compared to 8 effects), but smaller in σ (5 effects, compared to 7 effects). Overall both models contain the same number of effects, if a linear and nonlinear base-learner of the same effect is only counted once. The chlorophyill-a levels, mean epibenthic tidal velocity, smooth spatial variation and the year were not selected for the mean by stability selection, after fitting with the cyclical fitting algorithm. On the other hand bathymetry was selected by the cyclical fitting method, but not by the noncyclical. For the overdispersion parameter the cyclical algorithm selected the year and the northing of a segment (how much further north the segment is than the median) additional to all selected effects by the noncyclical variant.
Overall, the differences in the selected effects are rather small. Most effects were selected by both the cyclical and the noncyclical algorithm. In the simulation study for the negative binomial distribution (Chapter 6.2), the noncyclical variant had a smaller false positive rate and a higher true positive rate. Even though the simulation was simplified compared to this real data application (only linear effects, known true number of informative covariates, uncorrelated effects), the result suggest to believe the noncyclical variant. Yet, the final decision, which model to trust more and how to interpret it, should be evaluated by a specialist with *subject matter expertise*.

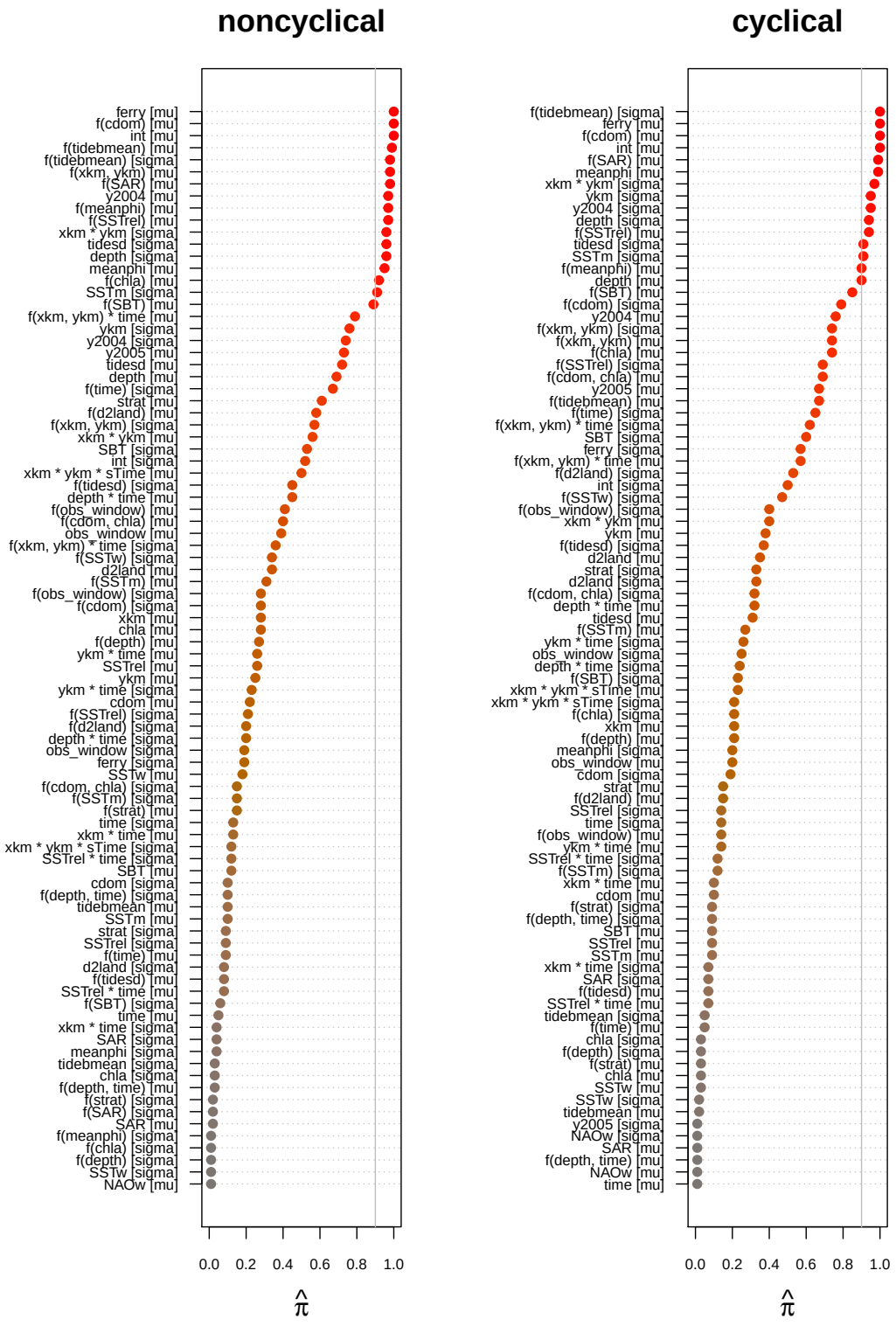

Figure 22: Selection frequencies for biophyiscal covariates of common eider abundance, determined by stability selection. Variables included in each iteration $q = 35$ and a per-family error-rate of 6

8 Conclusion and outlook

In this thesis a new way of fitting boosted generalized additive models for location scale and shape was introduced and thoroughly analyzed. In this chapter the results are summarized and some outlook and future work will be discussed.

8.1 Conclusion

For complex data situations flexible fitting methods are often required. One of these methods is the `gamboostLSS` framework by Hofner et al. [2015a]. The framework suffered from a very costly optimization phase, especially if a large number of iterations is required. Our new noncyclical fitting algorithm simplifies the model optimization tremendously. Even though the initial runtime to fit a single model is higher (especially if the base-learner selection is done via the *outer* loss approach), the time is regained while finding the optimal number of iterations. In case of the seabird abundance model, the runtime to optimize the number of iterations was reduced from more than one week (the runtime was not exactly measured and only extrapolated from a smaller fit on the dataset) to a mere two days. The convergence speed of the new algorithm is faster, as shown in a simulation, and consequently fewer iterations are needed compared to the cyclical way of fitting such models. Both different noncyclical fitting methods, one in which base-learner selection is conducted via RSS within each distribution parameter and with the loss function between the different distribution parameters, and one where all selection is based on the loss function, are almost identical in their performance.
The second argument for a noncyclical fitting approach was the combination of boosted GAMLSS with stability selection, to achieve sparser models and select informative effects. Because of the cyclical nature of the algorithm, it may *force* effects into the model that have small influence on the response, compared to effects that would be added in a later iteration for other distribution parameters. We conducted a simulation study with different distributions to validate this theory. For the tested two-parameter distributions, the noncyclical algorithm had fewer false positives as well as more true positives compared to the cyclical variant for almost all settings. For high dimensional cases, the differences between both methods reduced and, especially in the number of true positives, approximately equal results were achieved. For three-parameter distributions these results could not be verified and the cyclical variant achieved better values in both true and negative positives. In the abundance of the common eider populations, both cyclical and noncyclical fitting methods were tested on how they influence the results of the stability selection. The selected models were similar in most effects, but the noncyclical variant selected more effects for μ, whereas the cyclical variant selected

two additional effects in σ.
Overall, out new fitting method is faster and showed a better performance for stability selection in two-parameter distributions.

8.2 Outlook

In this Chapter some additional remarks are summarized where the presented techniques may be expanded in the future.

8.2.1 Stability selection: Counting base-learners independent of their parameter

Currently, if a effect is selected multiple times for different distribution parameters, it is counted multiple times to check if the maximum number of variables to include q, is reached. An alternative would be, to count the base-learners globally over all distribution parameters. An effect that is selected for μ *and* σ would still count only as one effect to check if the required number of base-learners is reached.
The question if this is a reasonable thing to do, is highly dependent on the situation and the actual problem definition. There may be situations, in which a researcher is interested exactly which effects are meaningful for a specific distribution parameter (e.g. which effects have influence on the variance in a normal distribution). On the other hand, there are situations where only the relevant question is, if a effect is informative at all. One thing to consider is, that if the base-learners are counted independently of their parameters, situations where distribution parameters have a similar effect may be improved. This could be especially useful for distributions with more than two parameters. The simulation study in Chapter 6 showed that both, cyclical and noncyclical algorithm have problems finding the informative variables in these cases.

8.2.2 Stability selection for three-parameter distributions

Stability selection for boosted GAMLSS models with three-parameter distributions currently does not work very well. Especially for high dimensional cases, it gets very difficult to find informative effects, while controlling the false discovery rate. The new noncyclical way of fitting boosted GAMLSS models results in fewer true positive effects and more false positive ones. We are currently not sure why this is the case, but it may have to do with the additional flexibility of the noncyclical fitting method. Often the decision which variable to include is concise. The noncyclical algorithm has a way larger number of possible base-learners to consider

in each iterations. The subsampling within stability selection, may result in a lot more different selections for the noncyclical method compared to the cyclical fitting method and consequently fewer informative effects are found. This behavior should be further examined, but currently no adequate solution is found and we recommend to use the cyclical algorithm for three-parameter, which showed better performance in most cases.

9 Bibliography

Hirotugu Akaike. A new look at the statistical model identification. *Automatic Control, IEEE Transactions on*, 19(6):716–723, 1974.

Peter Bühlmann and Torsten Hothorn. Boosting algorithms: Regularization, prediction and model fitting. *Statistical Science*, pages 477–505, 2007.

Peter Bühlmann and Bin Yu. Sparse boosting. *The Journal of Machine Learning Research*, 7:1001–1024, 2006.

Paul HC Eilers and Brian D Marx. Flexible smoothing with b-splines and penalties. *Statistical science*, pages 89–102, 1996.

Ludwig Fahrmeir, Thomas Kneib, and Stefan Lang. Penalized structured additive regression for space-time data: a bayesian perspective. *Statistica Sinica*, 14(3): 731–762, 2004.

Nora Fenske, Ludwig Fahrmeir, Peter Rzehak, and Michael Höhle. Detection of risk factors for obesity in early childhood with quantile regression methods for longitudinal data. 2008.

Virginia F Flack and Potter C Chang. Frequency of selecting noise variables in subset regression analysis: a simulation study. *The American Statistician*, 41 (1):84–86, 1987.

A Miranda Fredriks, Stef Van Buuren, Ruud JF Burgmeijer, Joanna F Meulmeester, Roelien J Beuker, Emily Brugman, Machteld J Roede, S Pauline Verloove-Vanhorick, and Jan-Maarten Wit. Continuing positive secular growth change in the netherlands 1955–1997. *Pediatric research*, 47(3):316–323, 2000.

J H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

Jerome Friedman, Trevor Hastie, Robert Tibshirani, et al. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *The annals of statistics*, 28(2):337–407, 2000.

Isabelle Guyon, Steve Gunn, Asa Ben-Hur, and Gideon Dror. Result analysis of the nips 2003 feature selection challenge. In *Advances in Neural Information Processing Systems*, pages 545–552, 2004.

Trevor J Hastie and Robert J Tibshirani. *Generalized additive models*, volume 43. CRC Press, 1990.

B Hofner, A Mayr, N Fenske, and M Schmid. *gamboostLSS: Boosting Methods for GAMLSS Models*, 2015a. URL `http://CRAN.R-project.org/package=gamboostLSS`. R package version 1.2-0.

Benjamin Hofner, Torsten Hothorn, Thomas Kneib, and Matthias Schmid. A framework for unbiased model selection based on boosting. *Journal of Computational and Graphical Statistics*, 2011.

Benjamin Hofner, Luigi Boccuto, and Markus Göker. Controlling false discoveries in high-dimensional situations: boosting with stability selection. *BMC bioinformatics*, 16(1):144, 2015b.

Benjamin Hofner, Andreas Mayr, and Matthias Schmid. gamboostLSS: An R package for model building and variable selection in the GAMLSS framework. *Journal of Statistical Software*, 2015c. Accepted.

Torsten Hothorn, Peter Buehlmann, Thomas Kneib, Matthias Schmid, and Benjamin Hofner. Model-based boosting 2.0. *Journal of Machine Learning Research*, 11:2109–2113, 2010.

Torsten Hothorn, Peter Buehlmann, Thomas Kneib, Matthias Schmid, and Benjamin Hofner. *mboost: Model-Based Boosting*, 2015. URL `http://CRAN.R-project.org/package=mboost`. R package version R package version 2.5-0.

Donald R Jones, Matthias Schonlau, and William J Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13 (4):455–492, 1998.

Thomas Kneib, Torsten Hothorn, and Gerhard Tutz. Variable selection and model choice in geoadditive regression models. *Biometrics*, 65(2):626–634, 2009.

Ron Kohavi and George H John. Wrappers for feature subset selection. *Artificial intelligence*, 97(1):273–324, 1997.

A Mayr, N Fenske, B Hofner, T Kneib, and M Schmid. Generalized additive models for location, scale and shape for high-dimensional data - a flexible approach based on boosting. *Journal of the Royal Statistical Society, Series C - Applied Statistics*, 61(3):403–427, 2012a.

Andreas Mayr, Benjamin Hofner, Matthias Schmid, et al. The importance of knowing when to stop. *Methods Inf Med*, 51(2):178–186, 2012b.

Andreas Mayr, Harald Binder, Olaf Gefeller, Matthias Schmid, et al. The evolution of boosting algorithms. *Methods Inf Med*, 53(6):419–427, 2014.

Peter McCullagh and John A Nelder. *Generalized linear models*, volume 37. CRC press, 1989.

N Meinshausen and P Bühlmann. Stability selection. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(4):417–473, 2010.

John Mullahy. Specification and testing of some modified count data models. *Journal of econometrics*, 33(3):341–365, 1986.

Tu Minh Phuong, Zhen Lin, and Russ B Altman. Choosing snps using feature selection. *Journal of bioinformatics and computational biology*, 4(02):241–257, 2006.

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2015. URL `https://www.R-project.org/`.

Gail Rees, Savita Bakhshi, Alecia Surujlal-Harry, Mikis Stasinopoulos, and Anna Baker. A computerised tailored intervention for increasing intakes of fruit, vegetables, brown bread and wholegrain cereals in adolescent girls. *Public health nutrition*, 13(08):1271–1278, 2010.

R A Rigby and D M Stasinopoulos. Generalized additive models for location, scale and shape. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 54(3):507–554, 2005.

R A Rigby, D M Stasinopoulos, and C Akantziliotou. Instructions on how to use the gamlss package in r. 2008.

Giacomo Scandroglio, Andrea Gori, Emiliano Vaccaro, and Vlasios Voudouris. Estimating var and es of the spot price of oil using futures-varying centiles. *International Journal of Financial Engineering and Risk Management*, 1(1):6–19, 2013.

Robert E Schapire and Yoav Freund. *Boosting: Foundations and algorithms*. MIT press, 2012.

Matthias Schmid and Torsten Hothorn. Boosting additive models using component-wise p-splines. *Computational Statistics & Data Analysis*, 53(2): 298–311, 2008.

Matthias Schmid, Sergej Potapov, Annette Pfahlberg, and Torsten Hothorn. Estimation and regularization techniques for regression models with multidimensional prediction functions. *Statistics and Computing*, 20(2):139–150, 2010.

Gideon Schwarz et al. Estimating the dimension of a model. *The annals of statistics*, 6(2):461–464, 1978.

Rajen D Shah and Richard J Samworth. Variable selection with error control: another look at stability selection. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 75(1):55–80, 2013.

Adam D Smith, Benjamin Hofner, Jason E Osenkowski, Taber Allison, Giancarlo Sadoti, Scott R McWilliams, and Peter W C Paton. Modeling the spatiotemporal abundance of wintering sea ducks in nantucket sound, usa: implications for marine spatial planning. *Ecological Applications*, 2016. under review.

Mikis Stasinopoulos and Bob Rigby. *gamlss.tr: Generating and fitting truncated (gamlss.family) distributions*, 2015. URL `http://CRAN.R-project.org/package=gamlss.tr`. R package version 4.3-1.

Mikis Stasinopoulos, Bob Rigby with contributions from Calliope Akantziliotou, Gillian Heller, Raydonal Ospina, Nicoletta Motpan, Fiona McElduff, Vlasios Voudouris, Majid Djennad, Marco Enea, and Alexios Ghalanos. *gamlss.dist: Distributions to be Used for GAMLSS Modelling*, 2015. URL `http://CRAN.R-project.org/package=gamlss.dist`. R package version 4.3-5.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

Haleh Vafaie and Kenneth De Jong. Genetic algorithms as a tool for feature selection in machine learning. In *Tools with Artificial Intelligence, 1992. TAI'92, Proceedings., Fourth International Conference on*, pages 200–203. IEEE, 1992.

Sara A Van De Geer, Hans C Van Houwelingen, et al. High-dimensional data: $p >> n$ in mathematical statistics and bio-medical applications. *Bernoulli*, 10 (6):939–943, 2004.

Stephen J Wright. Coordinate descent algorithms. *Mathematical Programming*, 151(1):3–34, 2015.

Achim Zeileis, Christian Kleiber, and Simon Jackman. Regression models for count data in r. 2007.

Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.

List of Figures

List of Algorithms

List of Tables

10 Appendix

10.1 Derivates of the normal distribution

Let $f(x|\mu,\sigma)$ be the density of the normal distribution. Then the derivates in regard to μ and σ are:

$$\begin{aligned}\frac{\partial f(x|\mu,\sigma)}{\partial \mu} &= -\frac{1}{\sqrt{2\pi\sigma^2}}\exp(-\frac{(x-\mu)^2}{2\sigma^2})\frac{x-\mu}{\sigma^2} \\ &= -f(x|\mu,\sigma)\frac{(x-\mu)}{\sigma^2}\end{aligned} \tag{18}$$

$$\begin{aligned}\frac{\partial f(x|\mu,\sigma)}{\partial \sigma} &= -\frac{1}{\sqrt{2\pi}\sigma^2}\exp(-\frac{(x-\mu)^2}{2\sigma^2}) + \frac{1}{\sqrt{2\pi\sigma^2}}\exp(-\frac{(x-\mu)^2}{2\sigma^2})\frac{(x-\mu)^2}{\sigma^3} \\ &= -\frac{1}{\sigma}f(x|\mu,\sigma) + f(x|\mu,\sigma)\frac{(x-\mu)^2}{\sigma^3} \\ &= \frac{(x-\mu)^2-\sigma^2}{\sigma^3}f(x|\mu,\sigma)\end{aligned} \tag{19}$$

10.2 Additional tables and graphics for the stability selection simulation

p	q	$\pi_{thr} = 0.6$			$\pi_{thr} = 0.7$			$\pi_{thr} = 0.8$			$\pi_{thr} = 0.9$		
		cyc	outer	PFER	cyc	outer	PFER	cyc	outer	PFER	cyc	outer	PFER
50	8	1|0	1.6|0	1.7	1|0	1.5|0	0.8	1|0	1.2|0	0.5	1|0	1.1|0	0.3
	15	3|0	3.6|0	5.9	3|0	3.4|0	2.9	2.9|0	3.2|0	1.8	2.5|0	3.1|0	1
	25	4|0	4|0	16.4	4|0	4|0	8	3.9|0	3.9|0	5.1	3.8|0	3.8|0	2.7
250	8	1|0	1|0	0.3	1|0	1|0	0.2	1|0	1|0	0.1	1|0	1|0	0.1
	15	1.5|0	1.8|0	1.2	1.3|0	1.6|0	0.6	1.1|0	1.3|0	0.4	1.1|0	1.2|0	0.2
	25	3|0	3|0	3.3	3|0	2.9|0	1.6	2.9|0	2.6|0	1	2.7|0	2.4|0	0.5
	50	4|0	3.9|0	13.2	4|0	3.8|0	6.4	3.9|0	3.6|0	4.1	3.8|0	3.4|0	2.2
500	8	1|0	1|0	0.2	1|0	1|0	0.1	1|0	1|0	0.1	1|0	1|0	0
	15	1.1|0	1.1|0	0.6	1|0	1|0	0.3	1|0	1|0	0.2	1|0	1|0	0.1
	25	2.9|0	2|0	1.6	2.7|0	1.8|0	0.8	2.4|0	1.6|0	0.5	1.9|0	1.3|0	0.3
	50	4|0	3.4|0	6.6	3.9|0	3.2|0	3.2	3.8|0	3|0	2.1	3.7|0	2.8|0	1.1

Table 5: Number *true*|*false positives* for μ in simulation with 100 runs and normal distribution

p	q	$\pi_{thr} = 0.6$			$\pi_{thr} = 0.7$			$\pi_{thr} = 0.8$			$\pi_{thr} = 0.9$		
		cyc	outer	PFER	cyc	outer	PFER	cyc	outer	PFER	cyc	outer	PFER
50	8	3.8|0.1	3.8|0	1.7	3.6|0	3.5|0	0.8	3.4|0	3.3|0	0.5	2.8|0	2.7|0	0.3
	15	4|0.2	4|0	5.9	3.9|0.1	3.9|0	2.9	3.8|0	3.8|0	1.8	3.5|0	3.4|0	1
	25	4|0.6	4|0.2	16.4	4|0.2	4|0	8	4|0	3.9|0	5.1	3.7|0	3.5|0	2.7
250	8	3|0	3|0	0.3	2.7|0	2.7|0	0.2	2.5|0	2.5|0	0.1	2.2|0	2.2|0	0.1
	15	3.4|0.1	3.4|0.1	1.2	3.2|0	3.2|0	0.6	3|0	2.9|0	0.4	2.5|0	2.5|0	0.2
	25	3.7|0.2	3.7|0.2	3.3	3.4|0	3.5|0	1.6	3|0	3.1|0	1	2.6|0	2.5|0	0.5
	50	3.9|0.1	3.9|0.1	13.2	3.7|0	3.7|0	6.4	3.3|0	3.4|0	4.1	2.8|0	2.7|0	2.2
500	8	2.7|0	2.7|0	0.2	2.5|0	2.5|0	0.1	2.3|0	2.2|0	0.1	2.1|0	2.1|0	0
	15	3.2|0	3.2|0	0.6	2.9|0	2.9|0	0.3	2.6|0	2.6|0	0.2	2.3|0	2.3|0	0.1
	25	3.5|0	3.4|0.1	1.6	3.2|0	3.2|0	0.8	2.9|0	2.9|0	0.5	2.4|0	2.4|0	0.3
	50	3.7|0	3.7|0.1	6.6	3.4|0	3.4|0	3.2	3|0	3|0	2.1	2.5|0	2.4|0	1.1

Table 6: Number *true*|*false positives* for σ in simulation with 100 runs and normal distribution

p	q	$\pi_{thr} = 0.6$			$\pi_{thr} = 0.7$			$\pi_{thr} = 0.8$			$\pi_{thr} = 0.9$		
		cyc	outer	PFER	cyc	outer	PFER	cyc	outer	PFER	cyc	outer	PFER
50	8	3|0	3|0	1.7	3|0	2.9|0	0.8	2.9|0	2.8|0	0.5	2.7|0	2.6|0	0.3
	15	3.1|0.2	3.1|0.1	5.9	3.1|0.1	3|0	2.9	3|0	3|0	1.8	2.9|0	2.8|0	1
	25	3.2|1.4	3.2|0.6	16.4	3.1|0.5	3.1|0.2	8	3.1|0.2	3|0.1	5.1	3|0	2.9|0	2.7
250	8	2.7|0	2.8|0	0.3	2.5|0	2.6|0	0.2	2.2|0	2.4|0	0.1	1.8|0	2|0	0.1
	15	3|0	3|0	1.2	3|0	2.9|0	0.6	2.9|0	2.8|0	0.4	2.8|0	2.5|0	0.2
	25	3.1|0.1	3.1|0	3.3	3.1|0.1	3|0	1.6	3|0	2.9|0	1	2.9|0	2.8|0	0.5
	50	3.1|1	3.1|0.4	13.2	3.1|0.3	3.1|0.1	6.4	3|0.1	3|0	4.1	2.9|0	2.9|0	2.2
500	8	2.5|0	2.6|0	0.2	2.2|0	2.4|0	0.1	1.9|0	2.1|0	0.1	1.6|0	1.7|0	0
	15	3|0	2.9|0	0.6	3|0	2.8|0	0.3	2.9|0	2.6|0	0.2	2.6|0	2.3|0	0.1
	25	3.1|0	3|0	1.6	3|0	3|0	0.8	3|0	2.9|0	0.5	2.8|0	2.6|0	0.3
	50	3.1|0.4	3.1|0.2	6.6	3.1|0.1	3|0	3.2	3|0	2.9|0	2.1	2.8|0	2.7|0	1.1

Table 7: Number *true*|*false positives* for μ in simulation with 100 runs and negative binomial distribution

p	q	$\pi_{thr} = 0.6$			$\pi_{thr} = 0.7$			$\pi_{thr} = 0.8$			$\pi_{thr} = 0.9$		
		cyc	outer	PFER	cyc	outer	PFER	cyc	outer	PFER	cyc	outer	PFER
50	8	1.3|1.4	1.3|1.4	1.7	1.1|1.3	1.2|1.3	0.8	1.1|1.1	1.1|1.2	0.5	0.8|1.1	0.8|1	0.3
	15	1.8|1.8	2|2.1	5.9	1.5|1.6	1.8|1.8	2.9	1.3|1.4	1.5|1.6	1.8	1|1.2	1.1|1.3	1
	25	2.1|2.5	2.4|3.5	16.4	1.9|2	2.1|2.5	8	1.6|1.7	1.8|2	5.1	1.3|1.4	1.5|1.6	2.7
250	8	0.8|1.2	0.9|1.2	0.3	0.7|1.1	0.7|1.1	0.2	0.6|0.9	0.6|1	0.1	0.4|0.9	0.4|0.8	0.1
	15	1.3|1.4	1.4|1.5	1.2	1.1|1.3	1.2|1.4	0.6	0.9|1.2	1|1.2	0.4	0.6|1.1	0.7|1.1	0.2
	25	1.4|1.7	1.6|2.1	3.3	1.2|1.4	1.4|1.7	1.6	1.1|1.2	1.2|1.4	1	0.8|1.1	1|1.1	0.5
	50	1.6|2.5	1.9|3.6	13.2	1.5|1.9	1.6|2.6	6.4	1.3|1.5	1.4|1.9	4.1	1.1|1.2	1.2|1.4	2.2
500	8	0.6|1.1	0.7|1.1	0.2	0.5|1	0.5|1	0.1	0.3|0.9	0.3|0.9	0.1	0.2|0.8	0.2|0.8	0
	15	1|1.3	1|1.4	0.6	0.8|1.2	0.9|1.2	0.3	0.6|1.1	0.6|1.1	0.2	0.4|0.9	0.5|1	0.1
	25	1.2|1.4	1.3|1.6	1.6	1.1|1.3	1.2|1.4	0.8	0.9|1.1	1|1.2	0.5	0.7|1	0.7|1.1	0.3
	50	1.4|2	1.7|2.7	6.6	1.3|1.6	1.4|2	3.2	1|1.4	1.2|1.6	2.1	0.8|1.2	1|1.3	1.1

Table 8: Number *true*|*false positives* for σ in simulation with 100 runs and negative binomial distribution

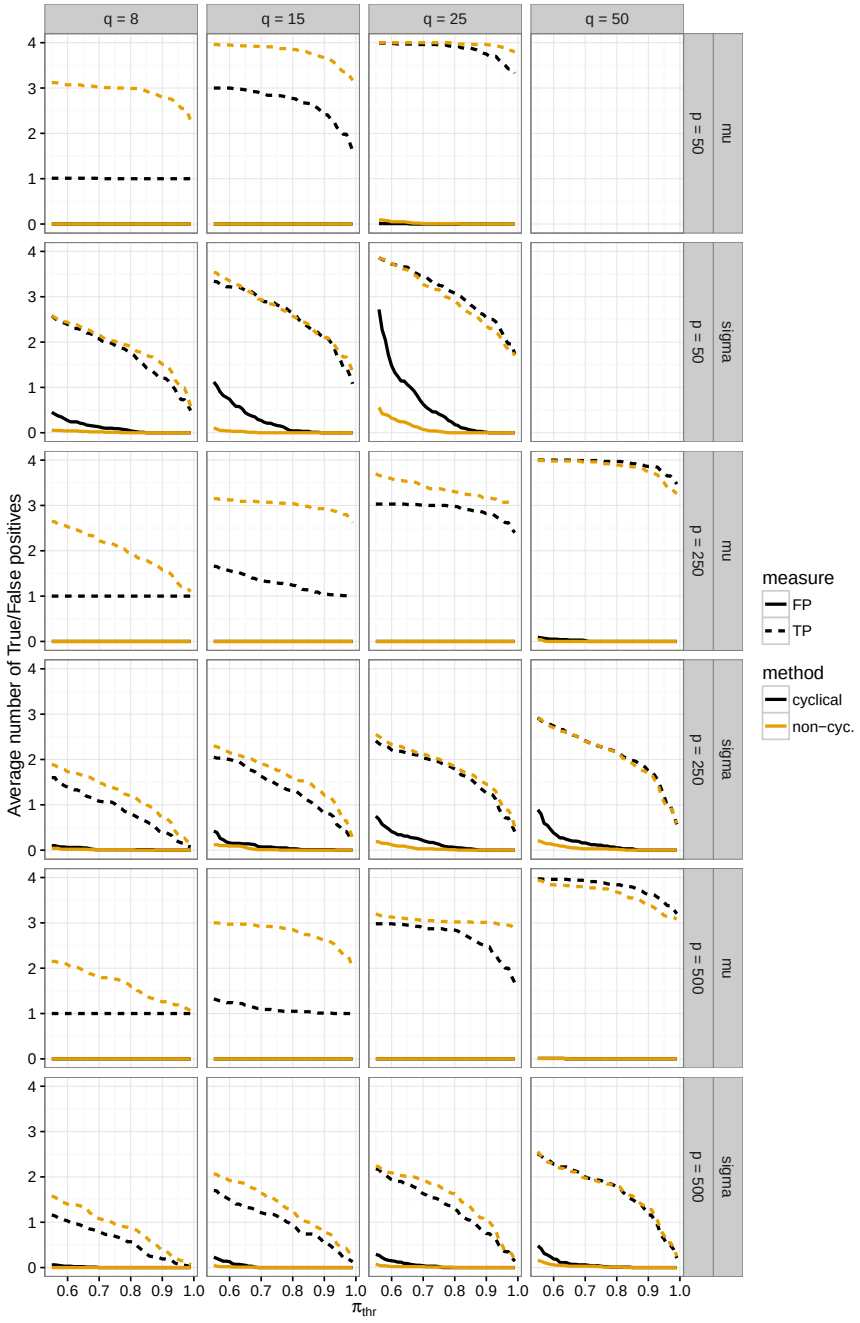

Figure 23: Stability selection: Normal distribution and balanced number of informative variables; Seperate for each parameter

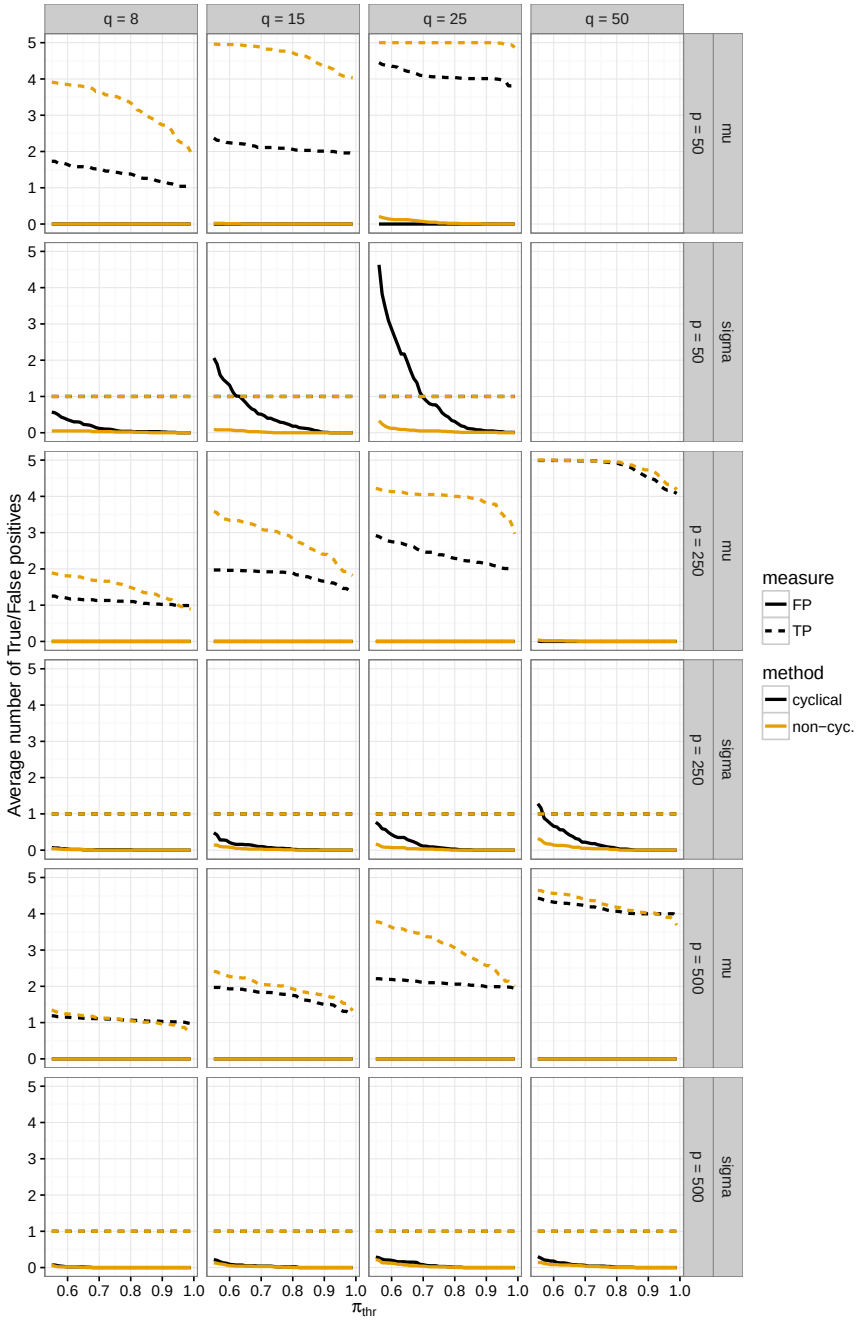

Figure 24: Stability selection: Normal distribution and unbalanced number of informative variables; Seperate for each parameter

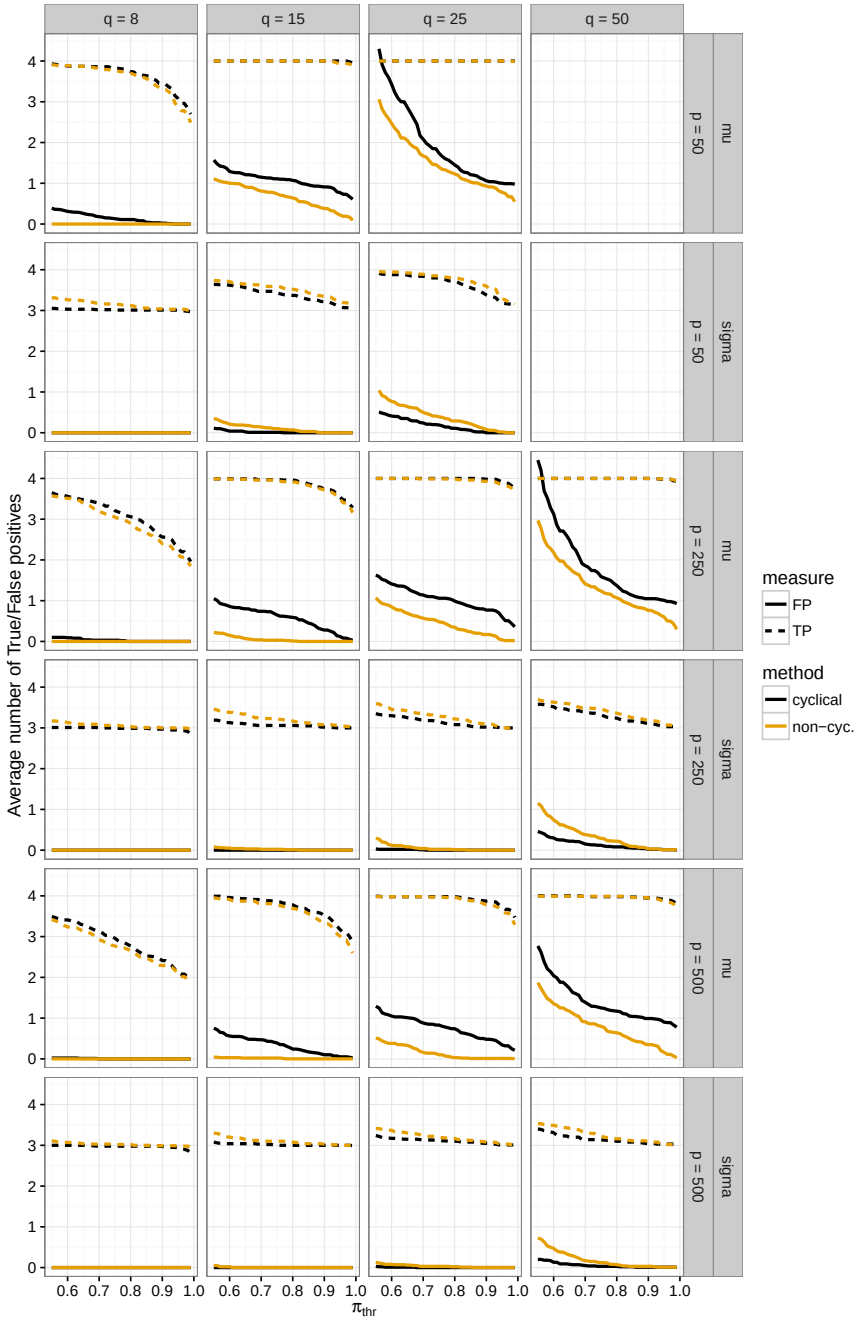

Figure 25: Stability selection: Negative binomial distribution and balanced number of informative variables; Seperate for each parameter

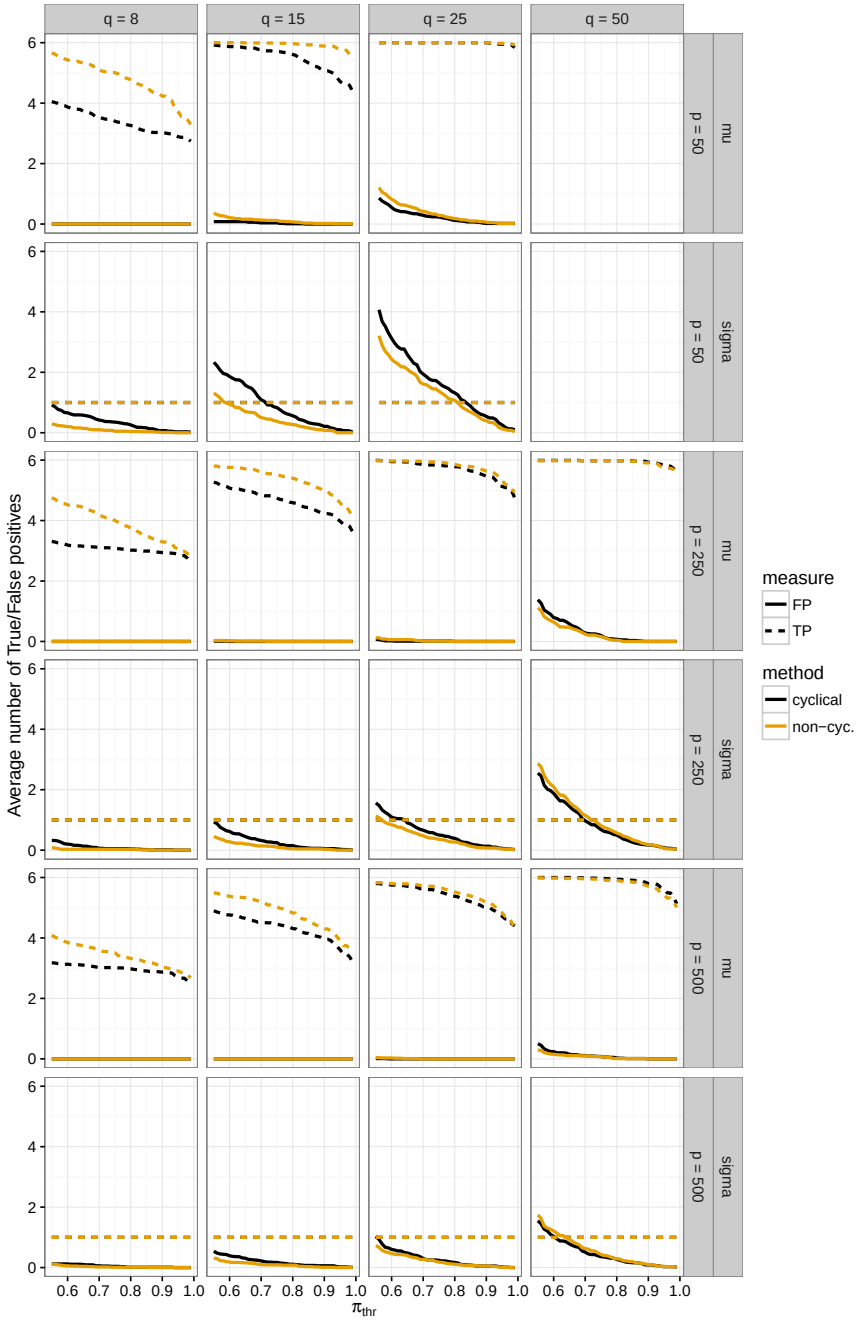

Figure 26: Stability selection: Negative binomial distribution and unbalanced number of informative variables; Seperate for each parameter

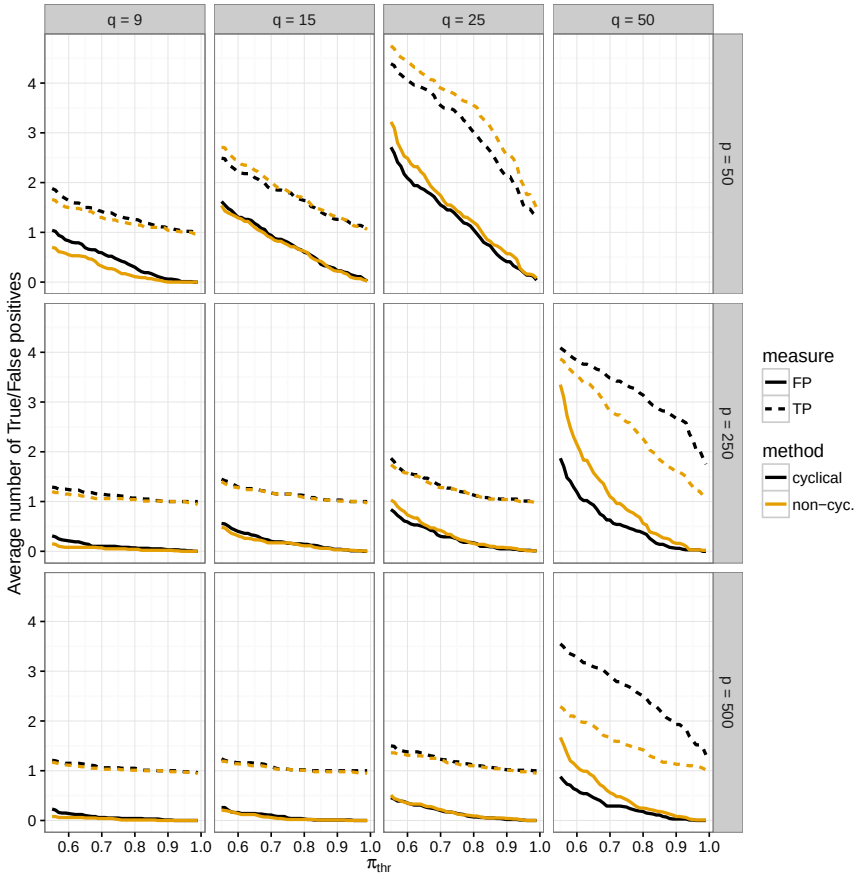

Figure 27: Stability selection: Student's t-distribution and balanced number of informative variables.

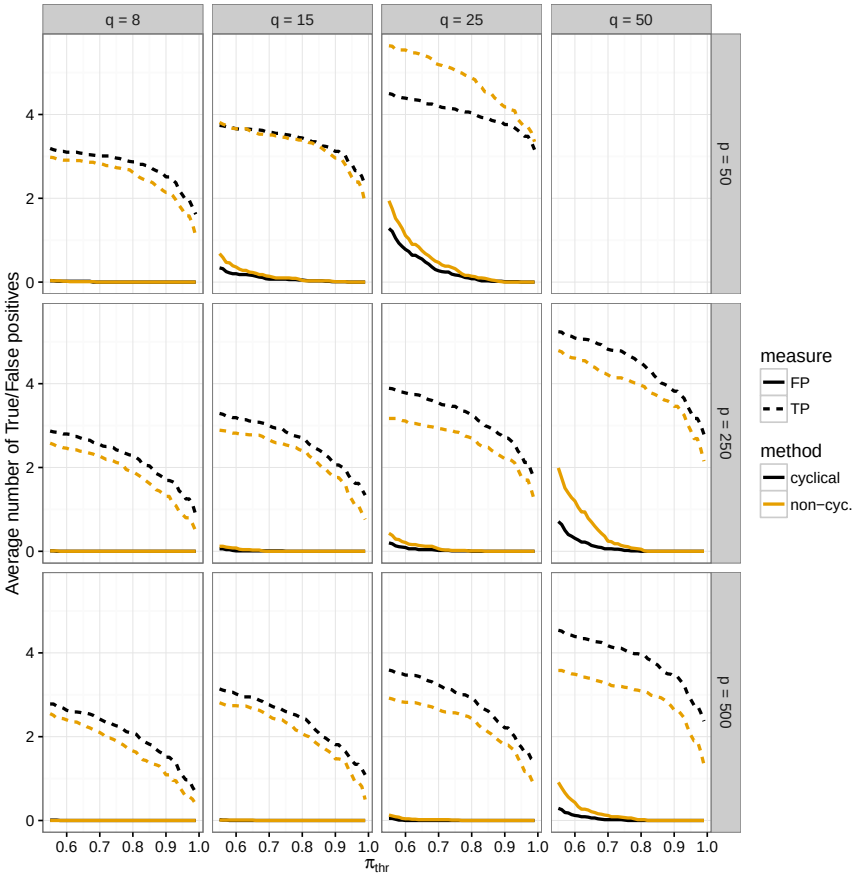

Figure 28: Stability selection: Student's t-distribution and unbalanced number of informative variables.

10.3 Implementation of noncyclical fitting algorithms

```
iBoost_cycling <- function(niter) {

    start <- sapply(fit, mstop)
    mvals <- vector("list", length(niter))
    for (j in 1:length(niter)){
      mvals[[j]] <- rep(start[j] + niter[j], max(niter))
      if (niter[j] > 0)
        mvals[[j]][1:niter[j]] <- (start[j] + 1):(start[j] + niter[j])
    }

    ENV <- lapply(mods, function(j) environment(fit[[j]]$subset))

    for (i in 1:max(niter)){
      for (j in mods){
        ## update value of nuisance parameters
        for (k in mods[-j])
          assign(names(fit)[k], families[[k]]@response(fitted(fit[[k]])),
                 environment(get("ngradient", environment(fit[[j]]$subset))))
        ## update value of u, i.e. compute ngradient with new nuisance parameters

        ENV[[j]][["u"]] <- ENV[[j]][["ngradient"]](ENV[[j]][["y"]],
                                                   ENV[[j]][["fit"]],
                                                   ENV[[j]][["weights"]])
        # same as:
        # evalq(u <- ngradient(y, fit, weights), environment(fit[[j]]£subset))

        ## update j-th component to "m-th" boosting step
        fit[[j]][mvals[[j]][i]]
      }
      if (trace){
        firstRun <- firstRun
        ## which is the current risk? rev() needed to get the last
        ## list element with maximum length
        whichRisk <- names(which.max(rev(lapply(lapply(fit, function(x) x$risk()),
                                                length))))

        do_trace(current = max(sapply(mvals, function(x) x[i])),
                 mstart = ifelse(firstRun, 0, max(start)),
                 mstop = ifelse(firstRun, max(niter) + 1, max(niter)),
                 risk = fit[[whichRisk]]$risk())
      }
    }
    return(TRUE)
  }
```

```
iBoost_outer <- function(niter){

  #this is the case for boosting from the beginning
  if(is.null(attr(fit, "combined_risk")) | niter == 0){
    combined_risk <- vapply(fit, risk, numeric(1))
  }

  best <- which(names(fit) == tail(names(combined_risk), 1))

  ENV <- lapply(mods, function(j) environment(fit[[j]]$subset))

  for (i in seq_len(niter)){

    ## update value of nuisance parameters
    for( k in mods[-best]){
      assign(names(fit)[best], families[[best]]@response(fitted(fit[[best]])),
             environment(get("ngradient", environment(fit[[k]]$subset))))

      evalq(u <- ngradient(y, fit, weights), ENV[[k]])
    }

    #glmboost with cwlin base-learner have to be considered separately
    if (funchar == "glmboost") {
      lik_risks <- list()
      coefs <- list()

      for(i in mods){
        coefs[[i]] <- evalq(environment(fit1)[["est"]](u)/
                              environment(fit1)[["sxtx"]], envir = ENV[[i]])

        all_fitted <- sweep(environment(ENV[[i]][["fit1"]])[["X"]], 2,
                            coefs[[i]] ,`*`)

        lik_risks[[i]] <- sapply(1:ncol(all_fitted), function(j)
          ENV[[i]][["triskfct"]](ENV[[i]][["y"]],
                                 ENV[[i]][["fit"]] + nu[i] * all_fitted[,j]))
      }

      #do a mboost step per hand
      best <- which.min(vapply(lik_risks, min,
                               FUN.VALUE = numeric(1), ...))

      ENV[[best]]$all_fitted <- all_fitted
      ENV[[best]]$lik_risks <- lik_risks
      ENV[[best]]$best <- best
      ENV[[best]]$coefs <- coefs

      #construct base-learner
      evalq({
        xs <- which.min(lik_risks[[best]])
        basses <- list(model = c(coef = coefs[[best]][xs],
                                 xselect = xs,
                                 p = length(coefs[[best]])),
                       fitted = function() {
                         return(coefs[[best]][xs] * environment(fit1)[["X"]][, xs,
```

```
                                                                    drop = FALSE])
                    })
      class(basses) <- c("bm_cwlin", "bm_lin", "bm")
      fit <- fit + nu * basses$fitted()
      u <- ngradient(y, fit, weights)
      mrisk[(mstop + 1)] <- triskfct(y, fit)
      ens[[(mstop + 1)]] <- basses
      xselect[(mstop + 1)] <- xs
      nuisance[[(mstop +1)]] <- family@nuisance()
      mstop <- mstop + 1},
      envir = ENV[[best]])
  }

  #all other base-learners
  else{
    risks <- list()
    for( i in mods){
      risks[[i]] <- evalq({
        ss_new <- lapply(get("blfit", envir = environment(basefit)),
                         function(x) x(u))
        sapply(ss_new, function(x) riskfct(y, fit + nu * x$fitted(), weights))
      }, envir = ENV[[i]])
    }

    best <- which.min(vapply(risks, min,
                             FUN.VALUE = numeric(1)))

    #construct base-learner
    evalq({
      ss <- lapply(get("blfit", envir = environment(basefit)),
                   function(x) x(u))
      xselect[mstop + 1] <- which.min(sapply(ss, function(x)
        riskfct(y, fit + nu * x$fitted(), weights)))
      fit <- fit + nu * ss[[tail(xselect, 1)]]$fitted()
      u <- ngradient(y, fit, weights)
      mrisk[(mstop + 1)] <- triskfct(y, fit)
      ens[[(mstop + 1)]] <- ss[[tail(xselect, 1)]]
      nuisance[[(mstop + 1)]] <- family@nuisance()
      mstop <- mstop + 1},
      envir = ENV[[best]])
  }

  combined_risk[(length(combined_risk) + 1)] <- tail(risk(fit[[best]]), 1)
  names(combined_risk)[length(combined_risk)] <- names(fit)[best]

  if (trace){
    do_trace(current = length(combined_risk[combined_risk != 0]),
             mstart = ifelse(firstRun, length(fit) + 1,
                             length(combined_risk[combined_risk != 0])),
             mstop = ifelse(firstRun, niter - length(fit), niter),
             risk = combined_risk[combined_risk != 0])
  }
}
combined_risk  <<- combined_risk
return(TRUE)
```

10.4 Digital appendix - CD

- /code/

  Contains the complete code that was used for the simulations and the visualizations used in this thesis.

- /gamboostlss

  Contains the current version of the *noncyclical_fitting* branch of the R package `gamboostLSS`.

- /seabirds/

  Contains data and code used for the analysis of the seabird dataset.

- /thesis/

  Contains this thesis in pdf form.

- /graphics/

  Contains all graphics used in this thesis.

- /simulation results/

  Contains the result of all run simulations as well as the code to process them.