Term Paper – LASSO-Regularization in a Location-Scale Regression Model

Emma Fössing
Emma.Foessing01@stud.uni-goettingen.de

Lucas Mohrhagen
Lucas.Mohrhagen@stud.uni-goettingen.de

Bent Rick
Bent.Rick@stud.uni-goettingen.de

October 16th, 2022
Advanced Statistical Programming with R

Inhaltsverzeichnis

1. Introduction

Regularization is used in regression when the coefficient estimator is numerically unstable. This is oftentimes the case when the variables outnumber the observations in the data. Instead of using variable selection methods that require high computational power, the regularization can be used simultaneously to the model estimation and shrink the effects of coefficients. In the case of the Least Absolute Shrinkage and Selection Operator (LASSO), it can also fully eliminate variables without any significant effect by shrinking their effect to zero (Fahrmeir et al. 2013, Tibshirani 1996).

This paper addresses the implementation of a LASSO Regression with location and scale coefficients in the R-package *asp22lasso*. Our package is an extension to the parent package *lmls*[1], which performs location-scale regression without penalization. The location-scale linear model uses location and scale parameters in the linear regression to reduce bias created by homoscedasticity and non-linearity in the data.

After a short description of the theoretical background (section 2) we will give an overview on the implementation of the LASSO regression in the package and its intended usage (section 3). We will proceed to describe the results of our simulation study (section 4) and end with a conclusion (section 5).

2. Theoretical Background

Considering a linear regression model without the location-scale differentiation Tibshirani (1996) introduced a least squares estimate with a penalization term, called the least absolute shrinkage and selection operator (LASSO)

$$\hat{\boldsymbol{\beta}}_{LASSO} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}}(\boldsymbol{y} - \boldsymbol{X\beta})'(\boldsymbol{y} - \boldsymbol{X\beta}) + \lambda \sum_{j=1}^{k} |\beta_j|,$$

where j denotes the jth column of $\boldsymbol{X}$ and λ is a smoothing parameter, whose size is crucial for the balance between a fit to the data with the least squares parameter or a regularized solution with the penalty $\sum_{j=1}^{k} |\beta_j|$. The intercept is not penalized.

Whereas the classical linear model assumes the response variable to be normally distributed

$$\boldsymbol{y} \overset{\text{ind.}}{\sim} \mathcal{N}(\boldsymbol{X}'\boldsymbol{\beta}, \boldsymbol{\sigma}^2),$$

the location-scale regression model extends this to

[1] View *lmls* package on CRAN: https://cran.r-project.org/web/packages/lmls/index.html.

$$\boldsymbol{y} \overset{\text{ind.}}{\sim} \mathcal{N}(\boldsymbol{X'\beta}, exp(\boldsymbol{Z'\gamma})^{\mathbf{2}}),$$

where $\boldsymbol{X}$ is the covariate matrix and $\boldsymbol{\beta}$ are the regression coefficients for the mean and $\boldsymbol{Z}$ is the covariate matrix and $\boldsymbol{\gamma}$ are the regression coefficients for the standard deviation.

Since the penalization term contains an absolute value, it is not differentiable, and the LASSO estimator does not have a closed form solution (Fahrmeir et al. 2013). The penalization term will therefore be approximated with $\lambda_\beta\sqrt{\boldsymbol{\beta'K\beta}+c}$ for the mean and $\lambda_\gamma\sqrt{\boldsymbol{\gamma'K\gamma}+c}$ for the standard deviation, where c is a small constant and where $\boldsymbol{\beta'K\beta}$ and $\boldsymbol{\gamma'K\gamma}$ equal the quadratic form of the covariate matrices. $\boldsymbol{K}$ is a identity matrix extended by one column of zeros on the lefthand side in order not to penalize the intercept.

As the *lmls*-package uses a maximum-likelihood (ML) algorithm, we adjust the likelihood formula with the estimators and end up with the following equation for the *lmls*-LASSOs:

$$\widehat{\boldsymbol{\beta}}, \widehat{\boldsymbol{\gamma}}_{LASSO} = \underset{\boldsymbol{\beta},\boldsymbol{\gamma}}{\operatorname{argmax}}\, loglik(\boldsymbol{\beta}, \boldsymbol{\gamma}) - \lambda_\beta\sqrt{\boldsymbol{\beta'K\beta}+c} - \lambda_\gamma\sqrt{\boldsymbol{\gamma'K\gamma}+c}$$

The optimal LASSO estimators are approximated in the already existing – for this method slightly adapted – ML algorithm with weighted least squares for $\widehat{\boldsymbol{\beta}}$ and Fisher scoring updates for $\widehat{\boldsymbol{\gamma}}$ (Riebl 2022):

The first (score function) and negative second (Fisher information) derivatives of the likelihood are:

$$score\_beta = s(\boldsymbol{\beta}) - \frac{\lambda_\beta}{2}(\boldsymbol{\beta'K\beta}+c)^{-\frac{1}{2}}\boldsymbol{\beta}(\boldsymbol{K}+\boldsymbol{K'})$$

$$score\_gamma = s(\boldsymbol{\gamma}) - \frac{\lambda_\gamma}{2}(\boldsymbol{\gamma'K\gamma}+c)^{-\frac{1}{2}}\boldsymbol{\gamma}(\boldsymbol{K}+\boldsymbol{K'})$$

$$info\_beta = F^*(\boldsymbol{\beta}) + \lambda_\beta\left((\boldsymbol{\beta'K\beta}+c)^{-\frac{1}{2}}\boldsymbol{K} - (\boldsymbol{\beta'K\beta}+c)^{-\frac{3}{2}}\boldsymbol{K\beta}(\boldsymbol{K\beta})'\right)$$

$$info\_gamma = F^*(\boldsymbol{\gamma}) + \lambda_\gamma\left((\boldsymbol{\gamma'K\gamma}+c)^{-\frac{1}{2}}\boldsymbol{K} - (\boldsymbol{\gamma'K\gamma}+c)^{-\frac{3}{2}}\boldsymbol{K\gamma}(\boldsymbol{K\gamma})'\right)$$

The initialization and update formulas are adapted as follows:

$\boldsymbol{\beta}$ is initialized with the penalized OLS

$$init\_beta = \left(\boldsymbol{X'X} + \lambda_\beta\boldsymbol{K}\right)^{-1}\boldsymbol{X'y}$$

$\boldsymbol{\gamma}$ is initialized with

$$init\_gamma = \left(\boldsymbol{Z'Z} + \lambda_\beta\boldsymbol{K}\right)^{-1}\boldsymbol{Z'\hat{s}},$$

$$\hat{\boldsymbol{s}} = log\left|\boldsymbol{y} - \boldsymbol{X}\widehat{\boldsymbol{\beta}}^{(\boldsymbol{OLS})}\right| + 0{,}6351814$$

While the *update_gamma* function can remain the same, the update for $\boldsymbol{\beta}$ is changed to

$$update\_beta = \left(\boldsymbol{X}'\boldsymbol{W}\boldsymbol{X} + \lambda_\beta \boldsymbol{K}\right)^{-1}\boldsymbol{X}'\boldsymbol{W}\boldsymbol{y}$$

$$\boldsymbol{W} = \frac{1}{\exp(\boldsymbol{Z}'\boldsymbol{\gamma})}$$

To find the optimal smoothing parameters λ_β and λ_γ a cross-validation function (*lasso_lmls*) is used:

First, a range of lambda values is set. The underlying data is split randomly into k-folds. Each of the k-folds is once serving as a validation set and the rest of the data is used as the training dataset.

For each fold and each combination of λ_β and λ_γ the optimal LASSO estimators are calculated for the training dataset and are used to predict the validation set response variable. Using the negative loglikelihood, the loss is evaluated. We create a loss-matrix containing the loss for each lambda combination and summed over all k-folds.

The optimal lambda value combination λ_β^* and λ_γ^* is defined by the smallest value in the matrix. By evaluating the function for a high number of lambda parameters within the range, the accuracy of the optimal smoothing parameters will increase. Using a large dataset and a sufficient number of k-folds will create more stable loss values and therefore increase the probability of choosing the correct lambda combinations within the given set.

Finally, the optimal smoothing parameters λ_β^* and λ_γ^* are used to refit the *lmls* model with the full dataset which will be returned by the function.

3. Implementation and usage of the package

In this section, we will explain how we implemented a location-scale regression model with the LASSO estimators in our R-package *asp22lasso*. We start off in the first subsection by giving a description on how to use our main function *lasso_lmls* and continue in the second subsection by explaining our implementation of the package which consists of the general approach of implementing the package and an error analysis.

3.1 Usage

Here, we will provide an overview on how users can perform a location-scale regression with the LASSO penalization by using our function *lasso_lmls*. A visual overview of the function as a flowchart can be found in the appendix. To estimate the locations-scale model with the LASSO, one must submit formulas of model specifications and the data. An example of the use of the function *lasso_lmls* is given in Code Excerpt 1, where the built-in *abdom* dataset from the package *lmls* is used.

Code Excerpt 1

```
data(abdom)
m_opt <- lasso_lmls(location = y ~ x,
                    scale = ~ x,
                    data = abdom,
                    k_fold = 5,
                    steps = 10)
```

Firstly, the user must specify the *location* variables and the *data* in which they are specified. The input *data* should be convertible into a data frame – if it is not a data frame already –, which we consider the standard input type. The *location* input requires a two-sided formula with the response variable on the lefthand side and the predictor for the mean on the righthand side.
All other inputs have default values but can also be varied by the user.
The *scale* input sets the predictor for the standard deviation on the righthand side of a one-sided formula and per default only uses an intercept of one.
Furthermore, the function includes the option of changing the number of folds that should be performed for the k-fold cross-validation (*folds*). A higher number of folds will train the model on a larger training set, which should lead to a lower prediction error (loss), but also requires more computational power. The default for this parameter is set to 10 folds.
The user can also customize the number of lambda values that should be evaluated to find the optimal value for both lambdas (*steps*). However, this does not change the range of the lambda values, as they are set to lay between 1e+1.5 and 1e+2. This simply changes how many lambda values should be evaluated between those two values. A higher number of *steps* will once again lead to higher computational cost, but also to more accurate estimations of the optimal lambda values. The default is set to 20. Furthermore, we include the option to change the size of the constant used in the quadratic approximation of the LASSO penalty, that has a default value of 1e-6.
The function *lasso_lmls* returns an instance of the S3-class “m_opt”, that provides the coefficients of the location-scale LASSO regression with optimal lambdas for location and scale and other important information. Code Excerpt 2 shows an example of *summary(m_opt)* with

similar data as later used in the simulation study in section 4 (the only difference is the seed set in the data generation).

Code Excerpt 2

```
> summary(m_opt)

Call:
lmls(location = location, scale = scale, data = data, lasso = TRUE,
    lambda = c(opt_loc, opt_scale), const = const, light = FALSE)

Deviance residuals:
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-3.043000 -0.676100 -0.012910  0.005848  0.654300  2.890000

Chosen Lambda values (location/scale):
 12.58925 12.58925

Location coefficients (identity link):
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.03043    0.05450  -0.558   0.5767
X1          -0.03990    0.05584  -0.715   0.4749
X2           0.17842    0.06215   2.871   0.0041 **
X3           1.09394    0.09853  11.102   <2e-16 ***
X4           1.74858    0.05486  31.875   <2e-16 ***
X5           2.29847    0.06125  37.525   <2e-16 ***
X6           2.64541    0.12068  21.921   <2e-16 ***
X7           3.00397    0.05983  50.208   <2e-16 ***
X8           3.70054    0.09646  38.363   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Scale coefficients (log link):
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.02386    0.03766  -0.634  0.52637
X1           0.07268    0.03777   1.924  0.05430 .
X2           0.02171    0.04124   0.526  0.59859
X3           0.08998    0.06407   1.404  0.16022
X4           0.09351    0.03670   2.548  0.01084 *
X5           0.13867    0.04101   3.382  0.00072 ***
X6           0.19337    0.07770   2.489  0.01282 *
X7           0.21141    0.04003   5.282 1.28e-07 ***
X8           0.28927    0.06454   4.482 7.39e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual degrees of freedom: 982
Log-likelihood: -1581.81
AIC: 3199.63
BIC: 3287.97
```

The returned object has various entries which include all “lmls” object entries from the parent package and some additional useful entries for a LASSO model. This includes a list of optimal lambdas λ_β^* and λ_γ^* (*m_opt$lambda*), the minimal loss of this lambda combination (*m_opt$loss*) and the full loss matrix for all possible lambda combinations (*m_opt$lossmatrix*).

3.2 Implementation

3.2.1 General approach

We implement the *asp22lasso* package by modification of the existing code in the parent package *lmls*. It is mostly modified by adding code, but estimating location-scale models without the LASSO penalty is still possible. We only write one completely new function, which is the *lasso_lmls* function. It mimics the *lmls* function in the parent package as it returns a similar object and requires a similar input. The package provides a description of all inputs and returns of the *lasso_lmls* function.

Functions used in the ML algorithm that require an adaption for the LASSO model are changed by adding conditional statements asking whether the model is the LASSO model or the *lmls* model. If the model is a LASSO model, the function is extended with LASSO model return. Most function adaptions for the LASSO model are specified in Section 2. Small function adaptions like the addition of arguments which are unused in the *lmls* function are not further discussed as they are rather trivial.

3.2.2 Error Analysis

For testing we use the package *testthat* by Wickham (2011). With tests one can check the code for functionality and use test errors to see where problems occur in the designed package. If a function does not return the expected results which are defined prior, the test will fail. An advantage is that these tests are automated, so one does not have to check for plausibility manually but can run these pre-defined tests to see if the functions run correctly. We used the tests to adapt code in our package and provide *testthat* functions that should not return errors. The package is tested for several individual functions. Some functions are intentionally left out in the tests for *asp22lasso*, due to the difficulty of creating a correct reference output. This is the case for the log likelihood and the fisher information. To our knowledge there is no available reference package which implemented a LASSO regression in a location and scale framework as we did. We do not run automated test on the coefficients but rather discuss their behaviour in the following simulation study (section 4).

When we run the R check *rmd* on our package it runs through with no errors, warnings, or notes.

Choice of lambda range The choice of the range of lambdas for which the optimal combination is found by the model was not an easy one. From our experience with the model, the code always selects the smallest lambdas that we have in the range, no matter what range values are set.

We see two possible origins for this problem. The most likely origin is that the model loss is lowest with no penalization ($\lambda_\beta^* = \lambda_\gamma^* = 0$) and therefore the optimal model is the one that penalizes the least that is possible. The LASSO estimator is supposed to be used in a model with a high proportion of variables compared to the number of observations, which does not work in our case, as we have negative eigenvalues in the Cholesky decompositions for this data in our Fisher information. This error causes the code to break. We could not detect an underlying problem leading to this error and are forced to work with data with enough observations and a sufficiently small number of variables to run our *lasso_lmls* code. This structure of the data is unlikely to produce a location-scale model that overfits and a penalization is not needed.

Another, in our opinion less likely origin of the problem, is a false calculation of the models' losses for each combination of λ_β and λ_γ. If the specifications for the loss calculation are false in the sense that the first row and first column of the matrix always have the lowest loss, this would then also lead to a selection of the lowest smoothing parameters.

As we were unfortunately not able to fix the underlying problem of this occurrence, we start the range of lambdas at a very high number ($\lambda_\beta = \lambda_\gamma \approx 31$). The model will choose the lowest, but manually set high value and we are able to see a difference between the *lmls* and the *lasso_lmls* model in the simulation study. This would not be the case if we have a very small minimal lambda value or even zero included in the range, as our model would choose this parameter and the results of the *lasso_lmls* would equal those of the *lmls* model.

Simulation of correlated variables The problem of the negative eigenvalues in the Cholesky decomposition arises again when we simulate correlated variables. As we expect the LASSO model to reduce the effect of correlated variables, we introduced correlation (see section 4). This expectation cannot be evaluated as the function *lasso_lmls* fails with the described error.

4. Simulation Study

To test if our statistical methods are working as intended, we perform a simulation study. We will describe our study design, the data generation process and present our results.

4.1 Design

While designing our simulation study, we used Morris et al. (2019) as a guideline. They suggest using "ADEMP", a systematic approach to planning simulation studies, which includes "Aims", "Data-generating mechanism", "Estimands", "Methods" and "Performance measures". In the following, we will go through all these categories, and we will explain how we implemented them in the design of our simulation study.

4.1.1 Aims

We want to show that our code works as intended and that our model is correctly executing a LASSO regression model with location and scale coefficients. To test this, we want to analyze the following:

1. Variable Selection: Compared to the *lmls* model, does our model successfully exclude non-informative regressors from the LASSO regression?
2. Bias: Does our model introduce a (small) bias on the effect size of the parameters? The estimated coefficients of our model should be generally smaller compared to the *lmls* model, as we introduce a general shrinkage effect (Bishop 2006).
3. Variance: Do our coefficients have less variance than the in *lmls* model? We want to show the bias-variance-tradeoff (Bishop 2006).

4.1.2 Data-generating mechanism

To test our model, we generate data by pseudo random sampling. This will allow us to understand the behavior of our model, as we know the actual parameter of interest. In our case, we generate our data based on set values for beta and gamma. This way, we know the true size of the coefficients in our model.

For our evaluation, we always use the same design matrix for location and scale (X=Z). We vary the number of observations (n) between 1.000 and 5.000. For every iteration of our simulation study, we generate data based on set β and γ values. We define eight covariates with:

$$\tilde{\beta} = [0, 0.15, 0.3, 0.5, 0.7, 1, 1.2, 1.5]$$
$$\tilde{\gamma} = [0, 0.03, 0.05, 0.1, 0.15, 0.19, 0.28, 0.31]$$

For β_1 and γ_1 we choose a size of zero. Our model should not show any effect of these, as they cannot be shrunken any further. Our γ values are chosen rather small, as we try to avoid an overly large variance in the prediction because we transform our scale model exponentially. We only use positive values for our coefficients. This will allow us to interpret our simulation study results more easily, because all coefficients are supposed to shrink towards zero and we avoid the cancelation of positive and negative effects.

Based on these true $\tilde{\beta}$ and $\tilde{\gamma}$ values, we draw n observations from a multivariate normal distribution

$$\tilde{\boldsymbol{X}} \sim \mathcal{N}(\mu_K, \Sigma_X),$$

where k = 1, …, 8, μ is set to a random value between -0.3 and 0.3, and $\sum_X$ is a covariance matrix where the variance is set randomly between 0.01 and 0.4, and the covariances are set to zero. As already discussed earlier, we are unable to evaluate our model with different covariances because our code cannot handle large covariances and breaks.

4.1.3 Estimands

This simulation study aims to evaluate our method of estimating the following estimands:

1. $\hat{\beta}_{Lasso}$: The optimal regression coefficients for the mean, estimated by penalized LASSO regularization.
2. $\hat{\gamma}_{Lasso}$: The optimal regression coefficients for the standard deviation, estimated by penalized LASSO regularization.

4.1.4 Methods

Our method for the analysis of the defined aims is a comparison between our LASSO model and the *lmls* model. We specify the full model as

$$\boldsymbol{y} \overset{\text{ind.}}{\sim} \mathcal{N}(\boldsymbol{X}'\boldsymbol{\beta}, exp(\mathbf{X}'\boldsymbol{\gamma})^{\mathbf{2}}).$$

We fit the unpenalized model using the *lmls* function of the parent-package *lmls* and the LASSO model using our own *lasso_lmls* function. We generate data based on the data-generating mechanism described in the previous section. Both models are then fit on our generated data and are evaluated according to our aims. The specific performance measures are described in the next section.

We repeat this process with different data 100 times. For each iteration of this study, we generate new data and compare the penalized model with the unpenalized model. The results are averaged over all iterations. Due to the high computational cost of the *lasso_lmls* function, we do not evaluate our model over more iterations. This process is performed with 1000 and 5000 observations.

4.1.5 Performance Measures

To assess if our model sufficiently achieves the goals that we formulated in section 4.1.1, we set the following performance measures:

1. Variable Selection: Our LASSO model should exclude regressors more frequently than the *lmls* model. We measure the frequency in which both models set the effect of certain regressors to zero. Variables count as excluded, if zero lays within one standard deviation of their coefficient effect.
2. Bias: To assess whether the general shrinkage of our penalized model works and if we successfully introduce a small bias, we compare the bias of the means for our LASSO model and the *lmls* model. We can measure the bias by subtracting our true values (β

and γ each) from our estimated coefficients. Since we only use positive true values, the coefficients of the LASSO model should show a negative bias. Meanwhile, there should be no significant deviation from the true values for the estimated *lmls*-coefficients.

3. Variance: Based on the concept of the bias-variance-tradeoff, the variance in our LASSO-model should be lower than in the *lmls* model. To evaluate this, we look at the mean of all the standard deviations of all coefficients in both models.

4.2 Results

In this part, we summarize the results of our simulation study for n=1000 and n=5000 after 100 runs.

4.2.1 Variable selection

Table 1 Exclusion frequency comparison for the lasso_lmls and lmls function

Variable		True Value	Exclusion rate by *lasso_lmls*		Exclusion rate by *lmls*	
		($\tilde{\beta}$ / $\tilde{\gamma}$)	n = 1000	n = 5000	n = 1000	n = 5000
location	V_1	0	1	1	1	1
	V_2	0.15	1	1	1	1
	V_3	0.3	0.89	0.92	0.92	0.92
	V_4	0.5	0.03	0	0	0
	V_5	0.7	0	0	0	0
	V_6	1	0.01	0	0	0
	V_7	1.2	0.01	0	0	0
	V_8	1.5	0.03	0	0	0
scale	V_1	0	0.99	1	0.94	1
	V_2	0.027	0.92	0.99	0.9	0.99
	V_3	0.045	0.78	0.91	0.83	0.91
	V_4	0.090	0.40	0.03	0.44	0.04
	V_5	0.135	0.13	0.01	0.13	0.01
	V_6	0.171	0.12	0.01	0.07	0.01
	V_7	0.252	0.19	0.01	0.07	0
	V_8	0.279	0.15	0.02	0.06	0

Remarks: V_i, with $i = 1, \ldots, 8$ are the Variables of the simulated models, n is the number of observations simulated, the true values $\tilde{\gamma}$ are rounded three digits.

Table 1 shows the exclusion rate in which each variable is counted as excluded in our LASSO model and in the *lmls* model. As discussed earlier, we expect the LASSO model to exclude coefficients more often than the unpenalized model. In the case of the location-parameters, the first coefficient with a value of 0 is excluded in 100% of all iterations for 1000 and 5000 observations and in both models. The same applies for the second coefficient with a value of 0.15.

For most of the other variables, we observe a slight difference in the exclusion rate between the two models for 1000 observations. The majority of the coefficients show a higher exclusion rate in our LASSO model. While V_4, V_6, V_7 and V_8 are never excluded in the *lmls* model, the LASSO model excludes them in one to three percent of cases. V_5 is never excluded in both models and V_3 even has a slightly higher exclusion rate in the *lmls* model. This might be due to the fact that the effect size of this coefficient will increase when other coefficients effects are shrunken. This might outweigh the LASSO shrinkage effect. For 5000 observations however, the exclusion rate in the LASSO model equals the *lmls* model exclusion rate.

For the scale-parameters, the higher exclusion rate in the LASSO model is more visible than in the location-case. For 1000 observations, the first variables get excluded in 94% of cases in the *lmls* model and 99% in the LASSO model. For most of the other variables the exclusion rate for the LASSO model is slightly higher than in the unpenalized model. Only V_4 and V_5 as location coefficients are excluded more frequently in the unpenalized model, which may also be due to the higher variance in the *lmls* model. For 5000 observations, the exclusion rates in the scale coefficients are roughly the same as for the location coefficients. However, the last two variables are only excluded in the LASSO model.

In conclusion, our model seems to exclude variables more frequently than the *lmls* model. This indicates that our code works as intended regarding variable selection. However, especially in the location-case, this effect is barely visible with our chosen exclusion criterion.

4.2.2 Bias

Figure 1 Bias in the location and scale model for n=1000

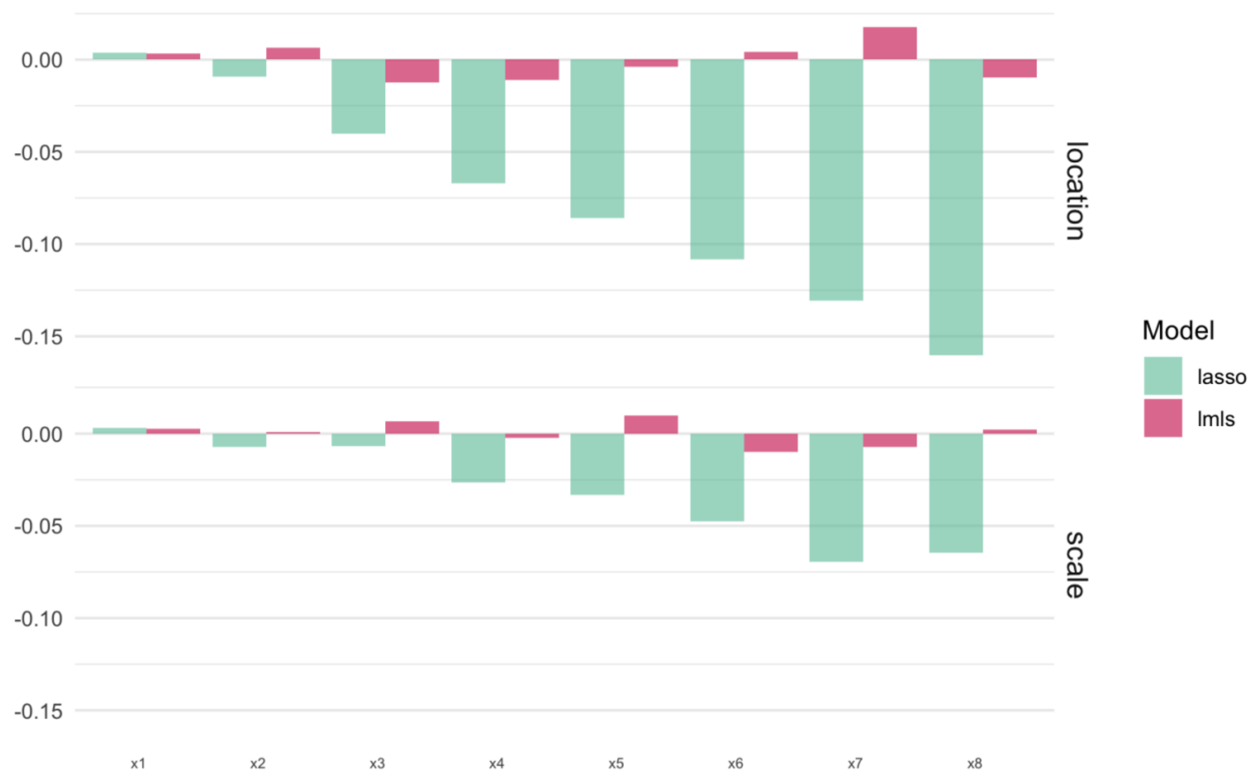
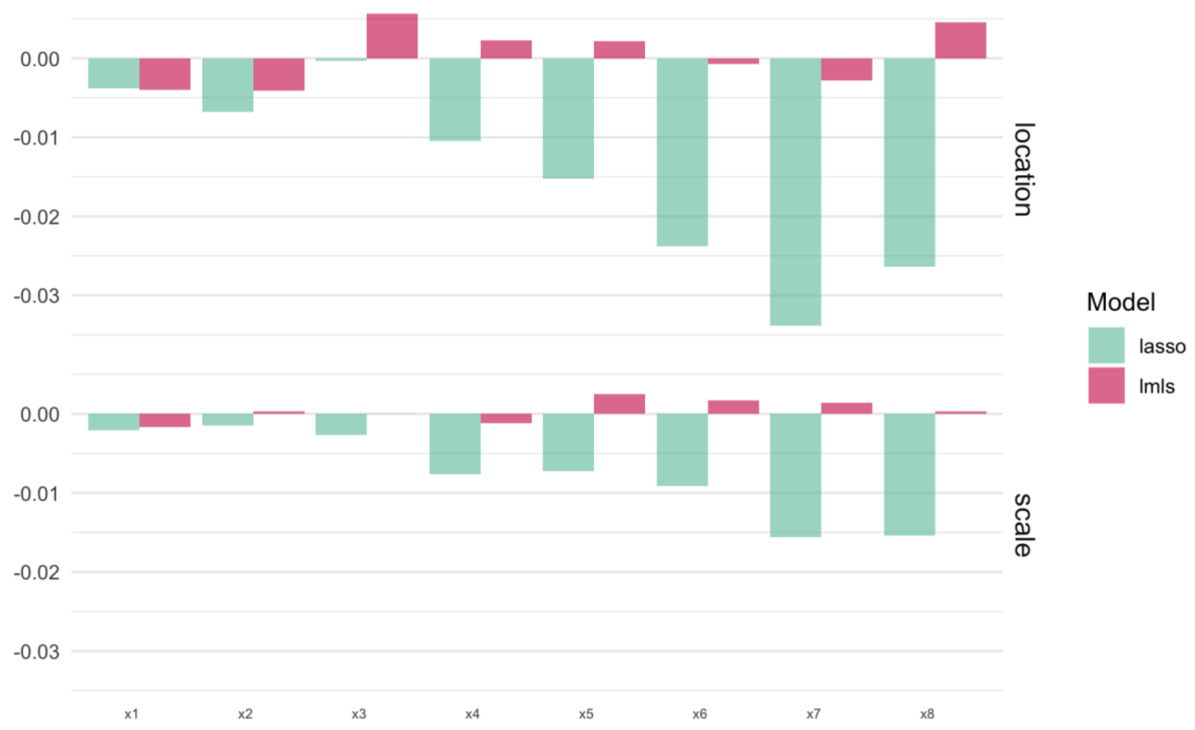


Figure 2 Bias in the location and scale model for n=5000



In figure 1 one can see the bias for both the *lasso_lmls* and *lmls* model for a number of 1000 observations. The effects for the first two location coefficients have almost no bias in neither *lasso_lmls* nor *lmls*. The third coefficient yields a visible increase in bias, the coefficients four to seven even more so for the *lasso_lmls* model. The eighth coefficient has the highest bias of all *lasso_lmls* coefficients for this simulated dataset. The *lmls* bias is rather small and even positive in covariate six and seven. For the scale coefficients a similar effect is observed on a slightly smaller scale. It can be concluded that the *lasso_lmls* model shrinks the coefficients as intended, while *lmls* does not introduce such shrinkage. This observation also holds for the models with 5000 observations in figure 2: The LASSO model introduces a significant shrinkage effect compared to the *lmls* model. As expected in a model with more observations, the coefficients' effects are more stable, and the shrinkage effect is lower than in the model with 1000 observations.

4.2.3 Variance

We want to confirm, that the LASSO model has a reduced variance in the coefficients compared to the *lmls* model. The results of the averaged variance are shown in figure 3 and 4. Just as we expected, we observe a lower overall variance in the LASSO model, than in the *unpenalized* model. In combination with the previous section, where we found that our model introduces a small bias, we can confirm that the bias-variance-tradeoff in our model works: A small bias is introduced, to achieve less variance. We also suspect that the overall variance should be lower in the model with 5000 observations than in the one with 1000 observations. This, however, cannot be confirmed in these figures as the difference seems to be rather minor.

Figure 3 Average variance in the location and scale model for n=1000

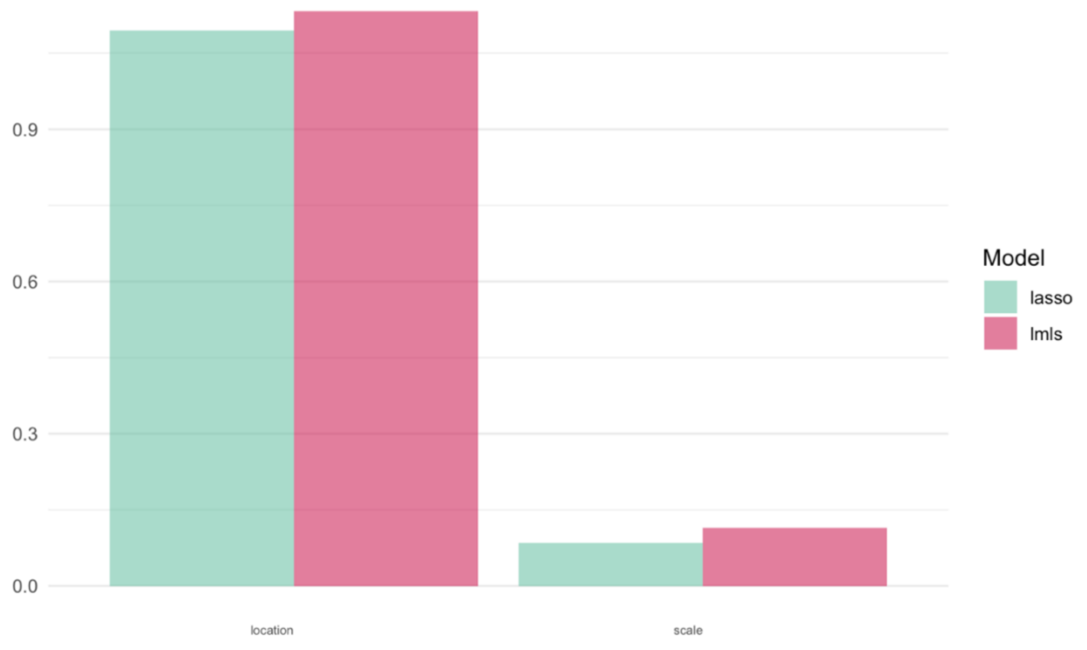
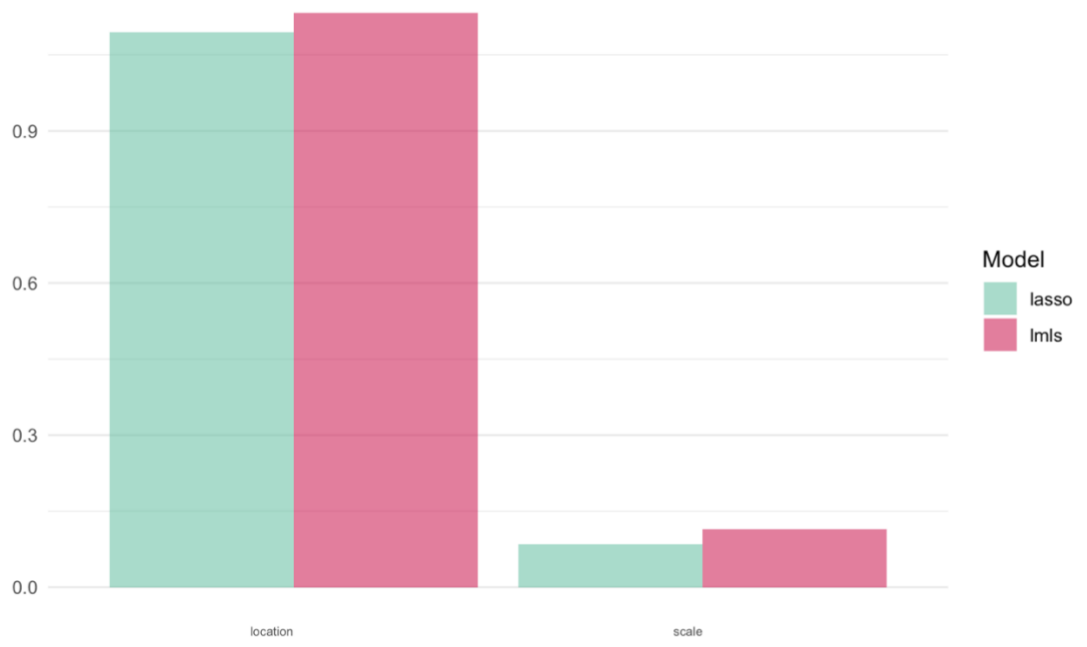


Figure 4 Average variance in the location and scale model for n=5000



Additionally, figure 5 and figure 6 show the individual standard deviations of the different coefficients. We observe the expected lower standard deviation in the LASSO model. The standard deviation is lower for all coefficients except for V_5 as a location coefficient. In these figures, we can also observe a very slight difference between the 1000 and 5000 observation models. For the model with 5000 observations the variance in the LASSO model is closer to the *lmls* model, as the coefficients are less penalized, and the models are more similar.

Figure 5 Standard deviations of the coefficients for n=1000

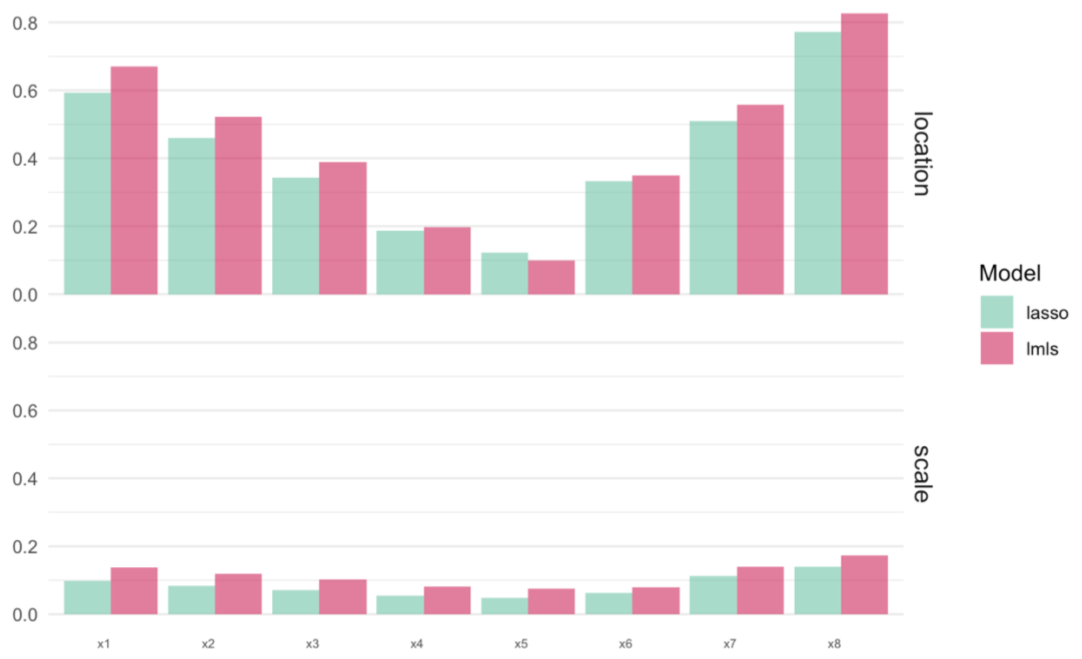
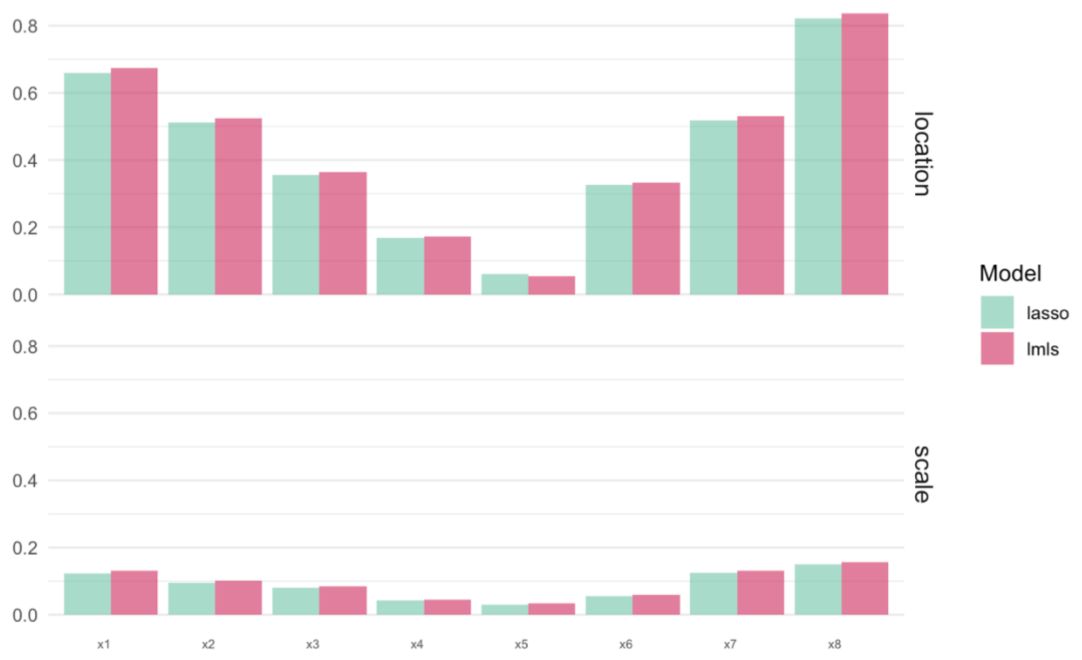


Figure 6 Standard deviations of the coefficients for n=5000



5. Conclusion

This paper discussed the implementation of an R-package that provides location-scale regression with a Least Absolute Shrinkage and Selection Operator via a maximum likelihood algorithm. The basis for the package *asp22lasso* was the parent package *lmls*, which does the same regression without penalization.

The application of LASSO penalties for location and scale parameters results in adapted *lmls* functions and an additional function *lasso_lmls*, which finds the optimal smoothing parameters for the penalty via k-fold cross validation and returns an optimal model.

Unfortunately, the function *lasso_lmls* is not able to fit a model in the data structure that LASSO models are usually used for, due to an error in the code that we could not fix. In the simulation study we manually set high smoothing parameters, which we would expect in the model if it would fit the desired data structure. It shows that the function returns results which we –based on the theory of LASSO penalized models– expect, as it shrinks the variance, as a trade-off introduces a small bias and more frequently than the *lmls* model estimates effects of zero.

The current development version of *asp22lasso* is a functioning package, which could still use some debugging work to fix current restrictions in the data structure and enhance currently still computational costly algorithms.

Literature

Bishop, C. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Heidelberg, Germany: Springer.

Fahrmeir, L., Kneib, T., Lang, S., & Marx, B. (2013). *Regression - Models, methods and applications*. Heidelberg, Germany: Springer.

Morris, T. P., White, I. R., & Crowther, M. J. (2019). *Using simulation studies to evaluate statistical methods.* Statistics in Medicine, *38*(11), 2074–2102. https://doi.org/10.1002/sim.8086. arXiv: 1712.03198.

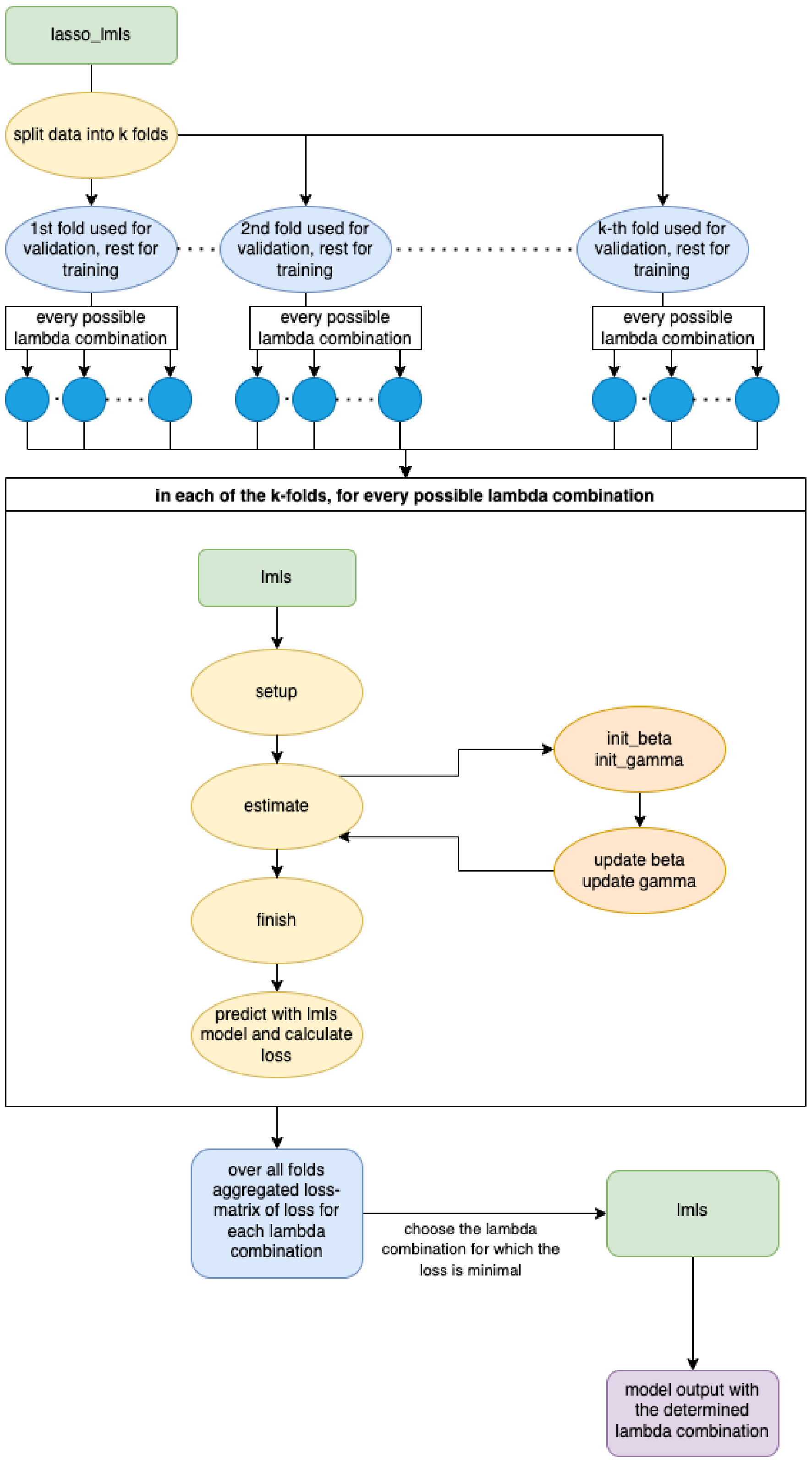
Riebl, H. (2022). *Location-Scale Regression and the lmls Package*. URL: https://cran.r-project.org/package=lmls. Accessed: October 14th, 2022.

Tibshirani, R. (1996). *Regression shrinkage and selection via the lasso*. Journal of the Royal Statistical Society: Series B (Methodological), 58(1), 267–288.

Wickham, H. (2011). *Testthat: Get started with testing*. The R Journal, *3*(1), 5.

Appendix

Flowchart of the structure of the function *lasso_lmls*



lasso_lmls
split data into k folds
1st fold used for validation, rest for training
2nd fold used for validation, rest for training
k-th fold used for validation, rest for training
every possible lambda combination
every possible lambda combination
every possible lambda combination
in each of the k-folds, for every possible lambda combination
lmls
setup
estimate
init_beta init_gamma
update beta update gamma
finish
predict with lmls model and calculate loss
over all folds aggregated loss-matrix of loss for each lambda combination
choose the lambda combination for which the loss is minimal
lmls
model output with the determined lambda combination

Selbstständigkeitserklärung

Wir versichern hiermit, dass wir die vorliegende Arbeit ohne fremde Hilfe selbstständig verfasst und nur die von uns angegebenen Quellen und Hilfsmittel verwendet haben. Wörtlich oder sinngemäß aus anderen Werken entnommene Stellen haben wir unter Angabe der Quellen kenntlich gemacht. Die Richtlinien zur Sicherung der guten wissenschaftlichen Praxis an der Universität Göttingen wurden von uns beachtet. Uns ist bewusst, dass bei Verstoß gegen diese Grundsätze die Prüfung mit nicht bestanden bewertet wird.

Emma Fössing

Göttingen, 16.10.2022

Lucas Mohrhagen

Göttingen, 16.10.2022

Bent Rick

Göttingen, 16.10.2022