# Introduction to the `SeaGuLl` package

Jan Klosa, Noah Simon, Dörte Wittenburg

2020-01-13

## Contents



### Introduction

SeaGuLl is a package that fits a mixed model via sparse-group lasso (SGL) regularization. Limiting cases of this penalty are the lasso and the group lasso, which are also available in this package. Solutions are obtained via *proximal gradient descent* on a grid of values for the regularization parameter $\lambda$. The grid search for this value is implemented using *warm starts*. And since proximal gradient descent is an iterative procedure, the step size between consecutive iterations is a crucial parameter for convergence. To determine this step size, *backtracking line search* is implemented.

We assume the following underlying mixed model:

$$y = Xb + Zu + e,$$

where $b$ is a vector of *fixed effects* and $u$ a vector of *random effects*. The inclusion of fixed effects is not mandatory in order for the algorithms to work.

Let $n$, $p$, and $L$ be the number of observations, random effects, and groups, respectively. `seagull` solves the following SGL problem:

$$\min_{(b,u)} \frac{1}{2n} \|y - Xb - Zu\|_2^2 + \alpha\lambda \left\|\text{diag}\left\{\omega_1^F, \ldots, \omega_p^F\right\} u\right\|_1 + (1-\alpha)\lambda \sum_{l=1}^{L} \omega_l^G \left\|u^{(l)}\right\|_2 .$$

Each $\omega$ represents a weight, either for a particular feature ($F$) or group ($G$). The *mixing parameter* $\alpha$ may range between 0 and 1. And as can be seen in the above formula, $\alpha = 1$ leads to the lasso, whereas $\alpha = 0$ leads to the group lasso.

### Example

Once installed, the package shall be loaded in the common way. A simulated example data set is available and loaded just as easily:

```r
# install.packages("devtools")
devtools::install_github("jklosa/seagull")
library("seagull")
data("seagull_data")
```

The data consists of a response variable $y$ named `phenotypes`, a design matrix $Z$ named `genotypes`, and a variable called `group`. $Z$ consists of 1000 observations and 466 explanatory variables. These variables consist of genotype data from single nucleotide polymorphism marker. The variable $y$ is another matrix, which harbors information of 3 traits. (One trait per column.) The variable `group` stores information about group assignments of each explanatory variable.

To fit the SGL, we can just call:

```r
fit_sgl1 <- seagull(y = phenotypes[, 1], Z = genotypes, groups = groups)
```

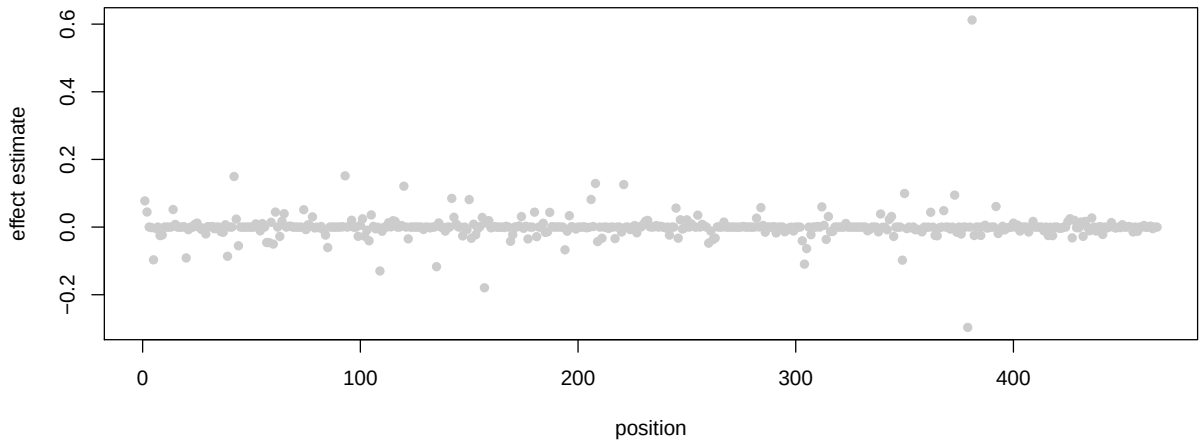
The created output is a list with the following attributes:

```r
attributes(fit_sgl1)
```

```
## $names
## [1] "random_effects" "lambda"         "iterations"     "alpha"
## [5] "rel_acc"        "max_iter"       "gamma_bls"      "xi"
## [9] "loops_lambda"
```

The last attribute of this list (`loops_lambda`) determines the number of different values for the penalty parameter $\lambda$ that were used throughout the grid search. Since we didn't specialize this value when calling the function `seagull`, it was set to its default, i.e. `loops_lambda = 50`. This value determines the number of rows of the variables `random_effects` and `iterations` (and also `fixed_effects` if present). So, if we want to visualize the results from a certain $\lambda$ along the grid, say the very last solution, where $\lambda$ is at its smallest, we could do this as follows:

```r
last_solution <- fit_sgl1$loops_lambda
plot(x = seq(1, dim(genotypes)[2], 1),
     y = fit_sgl1$random_effects[last_solution,],
     xlab = "position", ylab = "effect estimate",
     col = "gray80", pch = 16)
```

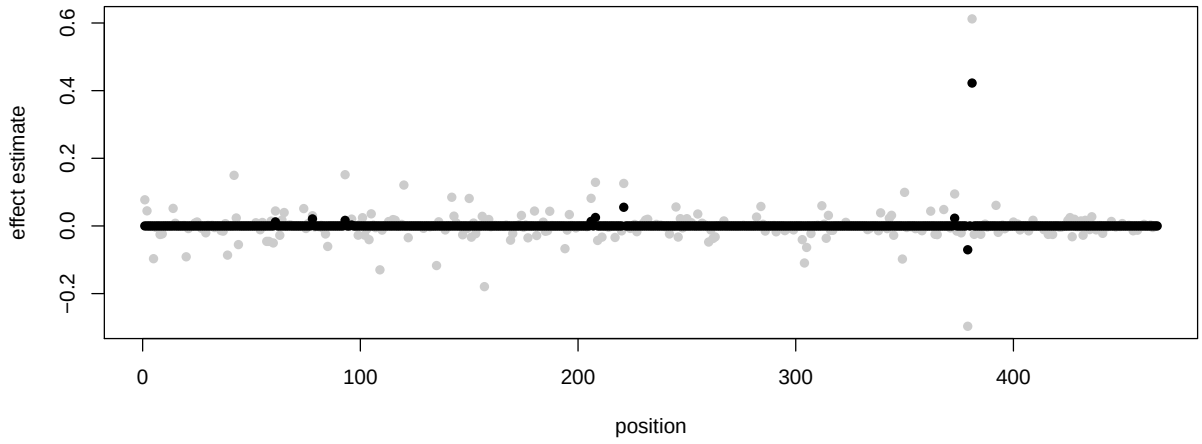


In total, there were 466 features to be estimated. We can check the number of features of this last solution which remained to be exactly equal to zero:

```
## [1] "The number of ZEROS in the last solution is: "
## [2] "28"
```

As this is the result based on the smallest $\lambda$, the solution is noisy. (By that we mean: The number of zeros among the estimates is low compared to the total number of estimates.) We will compare this to a solution that was obtained with a larger value:

```
points(x = seq(1, dim(genotypes)[2], 1),
       y = fit_sgl1$random_effects[20,],
       pch = 16)
```




```
## [1] "The number of ZEROS in the solution in line 20 is: "
## [2] "456"
```

This is a significantly larger number of zeros among the estimates than what we saw previously. In fact, for this value of $\lambda$ the solution is sparse. Which we expect from a selection operator such as these lasso variants.

## Input parameters

In this section we present a comprehensive list of input parameters: We will describe these variables and give additional comments about their use.

- `y` is a numeric vector of $n$ observations.
- `X` is an optional design matrix of dimension $n \times q$ which relates $y$ to fixed effects $b$.
- `Z` is a design matrix of dimension $n \times p$ which relates $y$ to random effects $u$.

Note: The above mentioned variables will neither be centered nor standardized throughout any of the calculations of this package.

- `weights_u` is an optional vector of weights for the feature vector $u$. Its length is $p$. These weights correspond to $\omega^F$ in the above SGL problem. The default value is `1` for each feature. The group weights $\omega^G$ are calculated as follows:

$$\omega_l^G = \sqrt{\sum_{j \in \text{ group } l} \omega_j^F}.$$

  So, in the case of all $\omega^F$ being equal to one in a certain group, the above expression collapses to the square root of the group size.

- `groups` is an integer vector of length $p$ or $q + p$. The entry is supposed to be a code for a group, e.g., say the first entry is `4`, then the first feature belongs to group `4`. This vector will remain unused for the lasso, so it doesn't need to be provided. But it is mandatory for group lasso and SGL. If fixed effects are incorporated, they may be assigned to a group. In this case, the vector shall be of length $q + p$. If no group is assigned for fixed effects, they will all automatically be assigned to the same group. In this case, or if no fixed effects are present, the length of this vector shall be $p$. The entries of this vector don't need to be in any kind of order.

- `alpha` is the optional mixing parameter for the SGL according to the above formula. Its default is `0.9`.

- `rel_acc` (or "$\varepsilon_{rel}$") is an optional parameter which reflects the stopping criterion. Convergence is assumed after iteration $m$, if the following inequality is fulfilled:

$$\left\| \begin{pmatrix} \hat{b} \\ u \end{pmatrix}^{[m]} - \begin{pmatrix} \hat{b} \\ u \end{pmatrix}^{[m-1]} \right\|_\infty \leq \varepsilon_{\text{rel}} \left\| \begin{pmatrix} \hat{b} \\ u \end{pmatrix}^{[m]} \right\|_2 .$$

  Default is $10^{-4}$. The smaller the value, the more iterations are needed to find the solution, which means that also more time is needed for the calculations.

- `max_lambda` is an optional value that sets the start value for the grid search for the penalty parameter $\lambda$. There are algorithms implemented for each lasso variant in order to determine an optimal value. These algorithms are the default option. We don't recommend to set this value, as it will most likely undermine the advantages of the warm starts procedure. For further details, please type `help("lambda_max")`.

- `xi` is an optional parameter for the determination of the minimal penalty parameter "$\lambda_{min}$", i.e., $\lambda_{min} = \xi \lambda_{max}$. We assume $\xi \in (0, 1]$. Default value is `0.01`.

- `loops_Lambda` is an optional, non-negative integer. It sets the number of $\lambda$'s which shall be investigated along the grid search from $\lambda_{max}$ to $\xi \lambda_{max}$. If `xi = 1`, this value will automatically be set to `1` and only a single solution will be computed for $\lambda = \lambda_{max}$. Default is `50`.

- `max_iter` is another optional, non_negative integer. It determines the maximum number of iterations which is allowed in order to try to reach convergence (according to `rel_acc`). Default is `1000`.

- `gamma_bls` is an optional variable. Should satisfy $0 < \gamma_{bls} < 1$. This parameter is related to the backtracking line search. Since proximal gradient descent is an iterative algorithm, a proper step size $(t)$ between iterations needs to be determined. In order to find an admissible update for $\begin{pmatrix} \hat{b} \\ u \end{pmatrix}$ each iteration begins with $t = 1$. If the condition for backtracking line search is satisfied, the update is admissible and will thus be performed. If the condition is not satisfied, $t$ will be decreased to $\gamma_{bls} t$ and the current iteration will restart. The restart is performed until the condition is met. Default is `0.8`.

- `trace progress` is an optional, logical parameter. If `TRUE`, a message will show up to indicate the end of the calculations for each $\lambda$ along the regularization path. This might come in handy for larger data sets. Default is `FALSE`.

## Further notes

We want to point out a correlation to another lasso variant, called *Integrative lasso with Penalty Factors, or short IPF-LASSO*. If we go back to the implemented SGL problem and its limiting case for the lasso, i.e., $\alpha = 1$, we obtain:

$$\min_{(b,u)} \frac{1}{2n} \|y - Xb - Zu\|_2^2 + \lambda \left\| \text{diag}\left\{\omega_1^F, \ldots, \omega_p^F\right\} u \right\|_1 .$$

If we suppose that no fixed effects are present in the model (i.e., $b = 0$) and we multiply the entire expression by the factor $2n$, we get:

$$\min_{(b,u)} \|y - Zu\|_2^2 + 2n\lambda \left\| \text{diag}\left\{\omega_1^F, \ldots, \omega_p^F\right\} u \right\|_1 ,$$

or equivalently:

$$\min_{(b,u)} \|y - Zu\|_2^2 + 2n\lambda \sum_{j=1}^{p} \left| \omega_j^F u_j \right| .$$

The weights for features $\omega^F$ are assumed to be positive. So, we can simplify the last expression by introducing $\lambda_j = 2n\lambda\omega_j^F$:

$$\min_{(b,u)} \|y - Zu\|_2^2 + \sum_{j=1}^{p} \lambda_j \left| u_j \right| .$$

As a last step, we assume that $u$ is obtained from $M$ different sources ("modalities") and we let all $\lambda$'s which belong to the same modality have the same value. Then the last term can be written as a sum over modalities $m$:

$$\sum_{j=1}^{p} \lambda_j \left| u_j \right| = \sum_{m=1}^{M} \lambda_m \left\| u^{(m)} \right\|_1 .$$

And this immediately leads to the IPF-LASSO. So in the `seagull` package, this particular lasso variant is implicitly included for mixed models. The weights for features $\omega^F$ (`weights_u`) just need to be set accordingly, i.e., the same weight for features that belong to the same modality.