Shrinkage Methods

Evalyne Muiruri

R-Users

Nov 2018

Why bother?

For a standard linear model:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots \beta_p X_p$$

- If $N \gg p$ where,

  N = No. of observations

  p = No. of predictor variables

  ✓ estimates for β coefficients are unbiased.

- As $N \to p$, overfitting occurs and models have less and less predictive power.
- If $N < p$, coefficient estimates have very high variance and cannot be trusted

Do we really need all p predictors?

Alternatives?

Subset Selection

Fits models to a subset of p predictors.

e.g. Forward/Backward selection

Unsupervised, computationally intensive

Biased towards small coefficient estimates and small p-values

Perform poorly out of sample

Ridge regression

Fit a full model and shrink (regularize) estimates close to zero.

Prevents overfitting

Robust to correlated variables

Lasso Regression

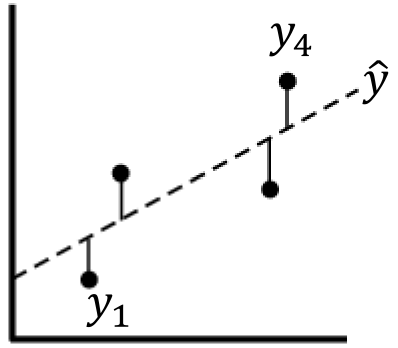
Fit a full model and regularize coefficients but shrink some to exactly zero

Achieves both regularization and variable selection

Ridge Regression

(Also known as L2 Regularization)

Recall that residual sum of squares is



$$RSS = \sum_{i=1}^{N} (y_i - \hat{y})^2$$

Where $\hat{y}$ is the linear model/ line through the data.

Simple linear models try to minimise RSS.

In Ridge regression, we try to minimise a slightly different quantity:

$$\sum_{i=1}^{N} (y_i - \hat{y})^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

Tuning parameter ≥0

Shrinkage penalty

Note: we are not shrinking the intercept, β_0

Ridge Regression

(Also known as L2 Regularization)

This might also be written as:

$$\hat{\beta}_{ridge} = \min_{\beta}\left\{\sum_{i=1}^{N}(y_i - \hat{y})^2 + \lambda \sum_{j=1}^{p} \beta_j^2\right\}$$

Tuning parameter

Shrinkage penalty

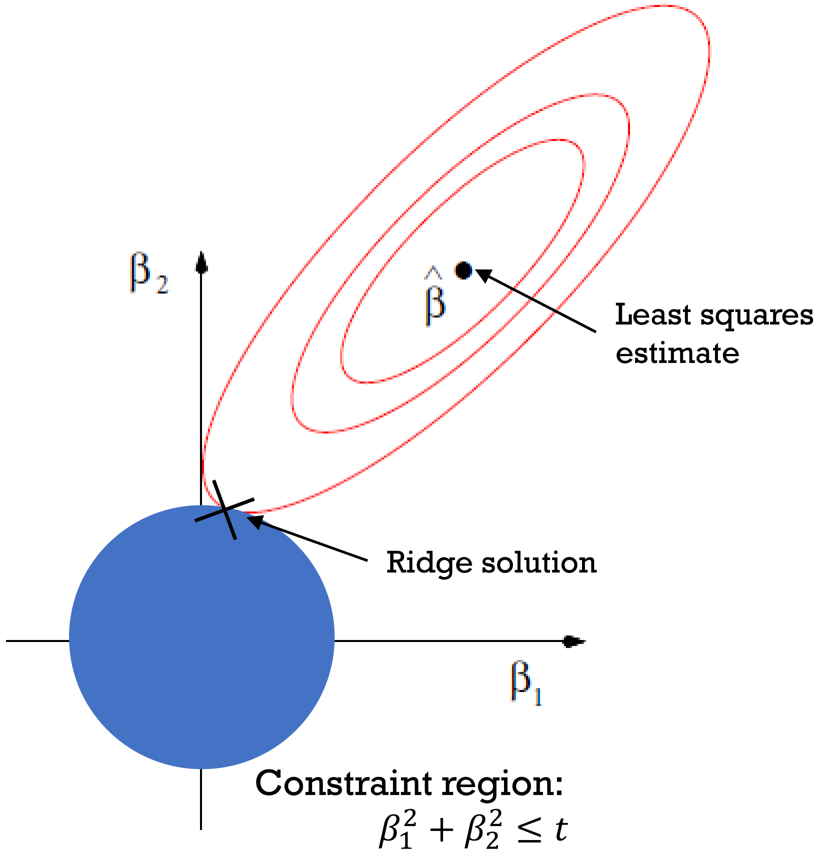
Or alternatively

$$\hat{\beta}_{ridge} = \min_{\beta}\left\{\sum_{i=1}^{N}(y_i - \hat{y})^2\right\}$$

$$subject\ to\ \sum_{j=1}^{p} \beta_j^2 \leq t$$

for every value of λ, there is some t such that will give the same lasso coefficient estimates (and vice-versa)

Squaring in shrinkage penalty amplifies extreme values → Need to scale and standardize all predictors beforehand



This might also be written as:

$$\hat{\beta}_{ridge} = \min_{\beta}\left\{\sum_{i=1}^{N}(y_i - \hat{y})^2 + \lambda \sum_{j=1}^{p}\beta_j^2\right\}$$

Tuning parameter

Shrinkage penalty

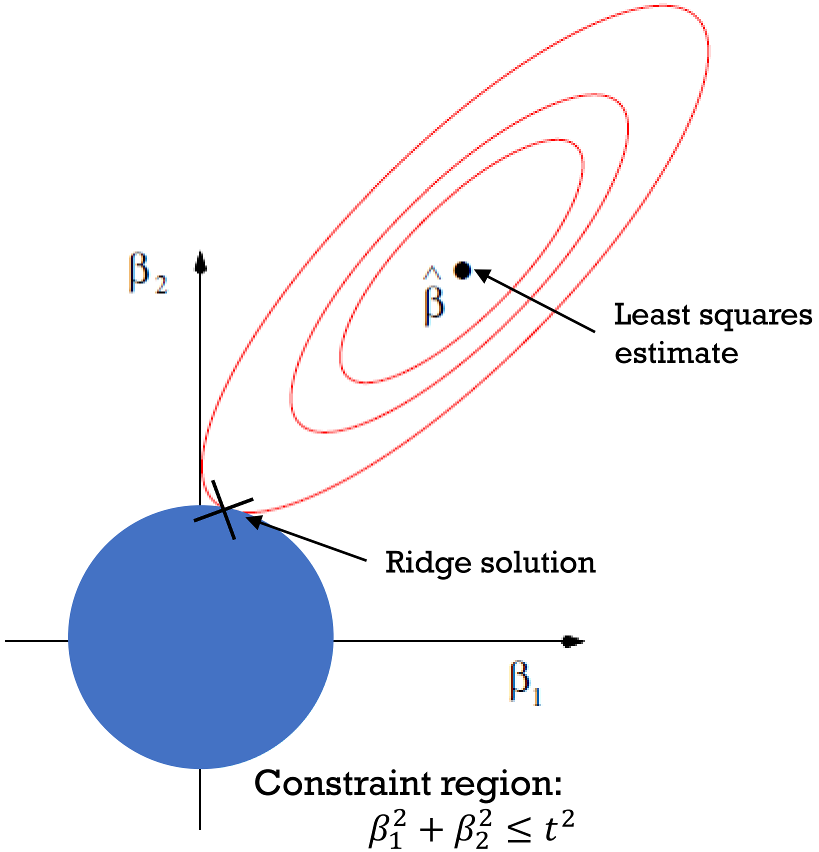
Or alternatively

$$\hat{\beta}_{ridge} = \min_{\beta}\left\{\sum_{i=1}^{N}(y_i - \hat{y})^2\right\}$$

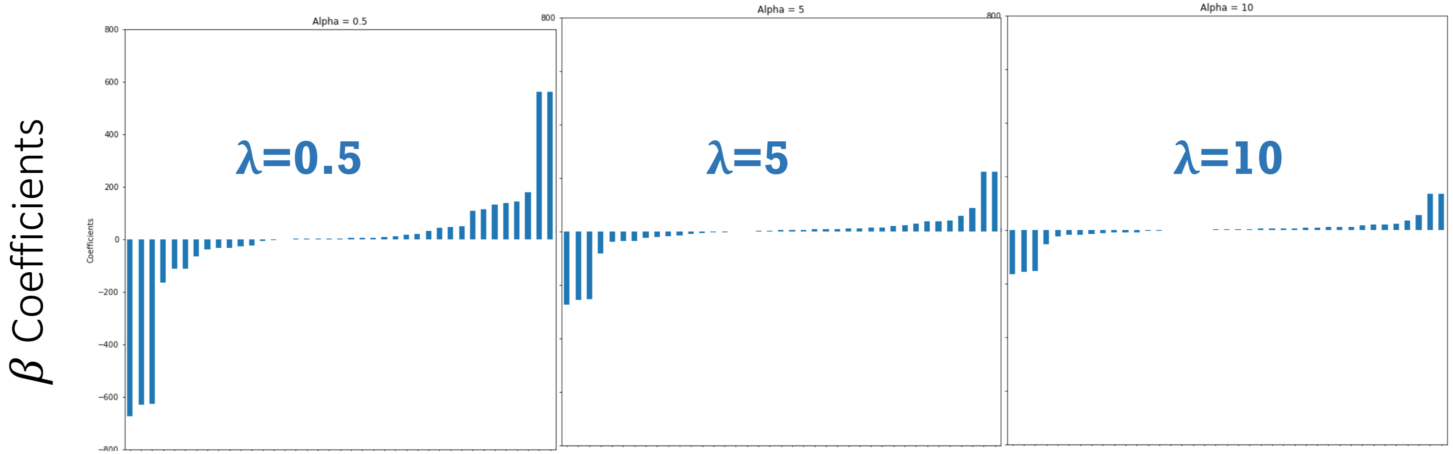
$$subject\ to\ \sum_{j=1}^{p}\beta_j^2 \leq t$$

for every value of λ, there is some t such that will give the same lasso coefficient estimates (and vice-versa)

Squaring in shrinkage penalty amplifies extreme values → Need to scale and standardize all predictors beforehand

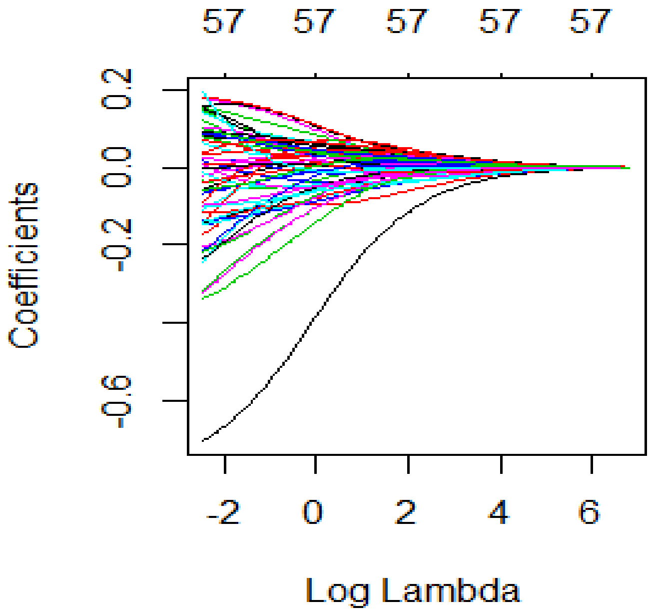


- The ridge shrinkage penalty $\sum_{j=1}^{p} \beta_j^2$, pulls coefficient estimates closer to zero
- Shrinks together the coefficients of correlated predictors
- For smaller values of λ, the constraint region relaxes and the ridge solution tends towards the least squares solution. $\hat{\beta}_{ridge} \approx \hat{\beta}_{least\ squares}$



Ridge Regression

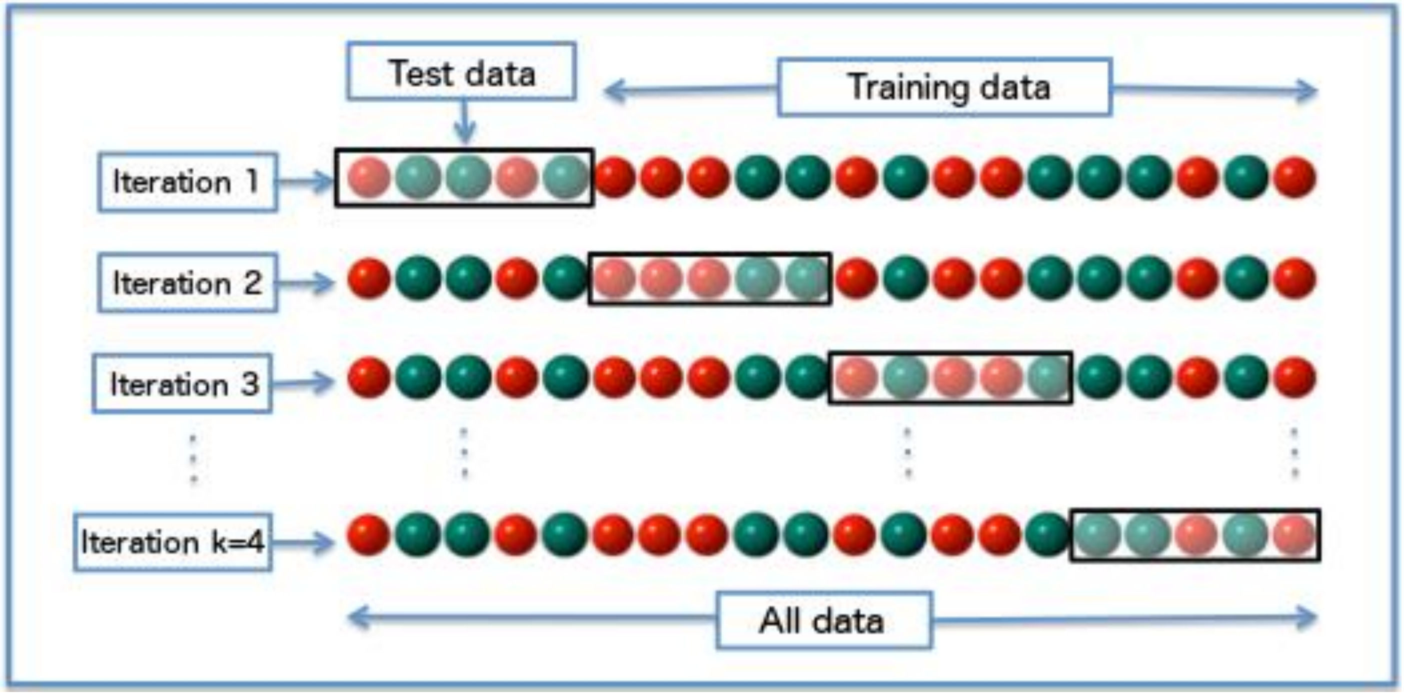
```
install.packages("glmnet", "glmnetUtils")
library(glmnet); library(glmnetUtils)
ridgefit1 <- glmnet(log(DensAbund_N_Sqkm) ~ .,
    data=numd.sc, alpha=0, family="gaussian")
plot.glmnet(ridgefit1, "lambda")
```



It is critical to pick a good value of λ for the right coefficients

Ridge Regression

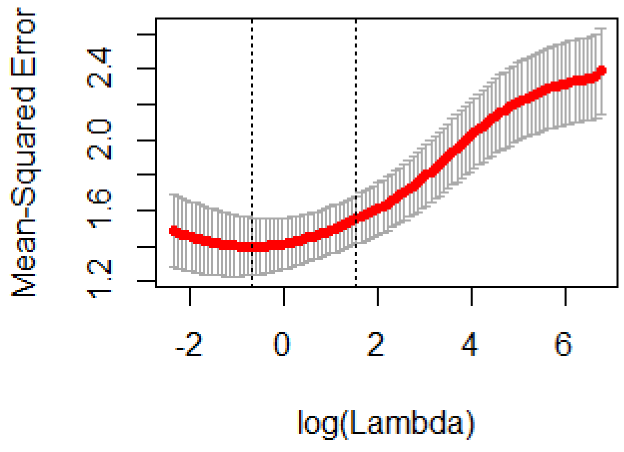
- Implement Cross Validation to pick the right tuning parameter, λ.
- Glmnet comes with an inbuilt function but we still need to iterate it lots of times as the results are random.



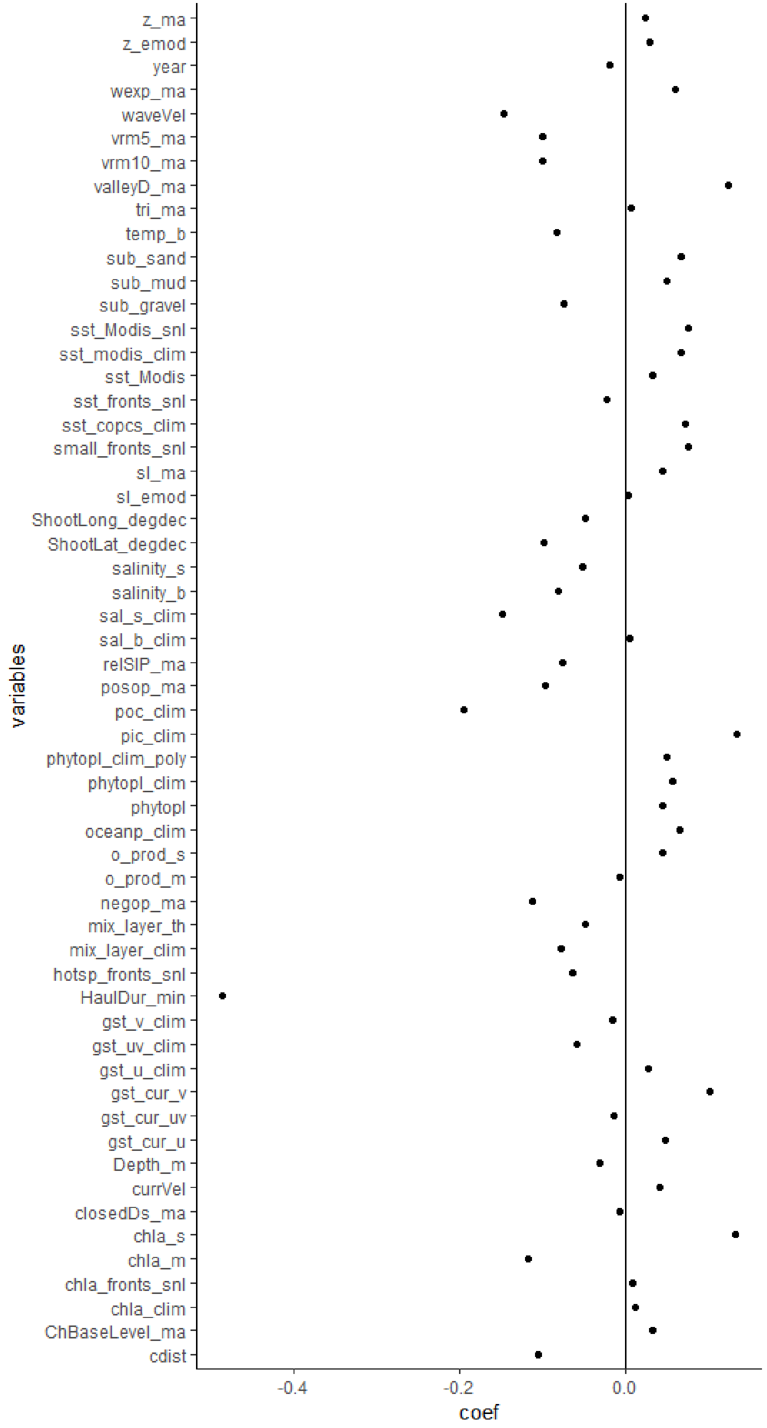
Ridge Regression

- Implement Cross Validation to pick the right tuning parameter, λ.
- Glmnet comes with an inbuilt function but we still need to iterate it lots of times as the results are random.

```
ridgefit1.cv <- cv.glmnet(log(DensAbund_N_Sqkm) ~ .,
                data=numd.sc, alpha=0, family="gaussian")
plot.cv.glmnet(ridgefit1.cv)
log(ridgefit1.cv$lambda.min) # for 1 iteration
[1] -0.3917911
```

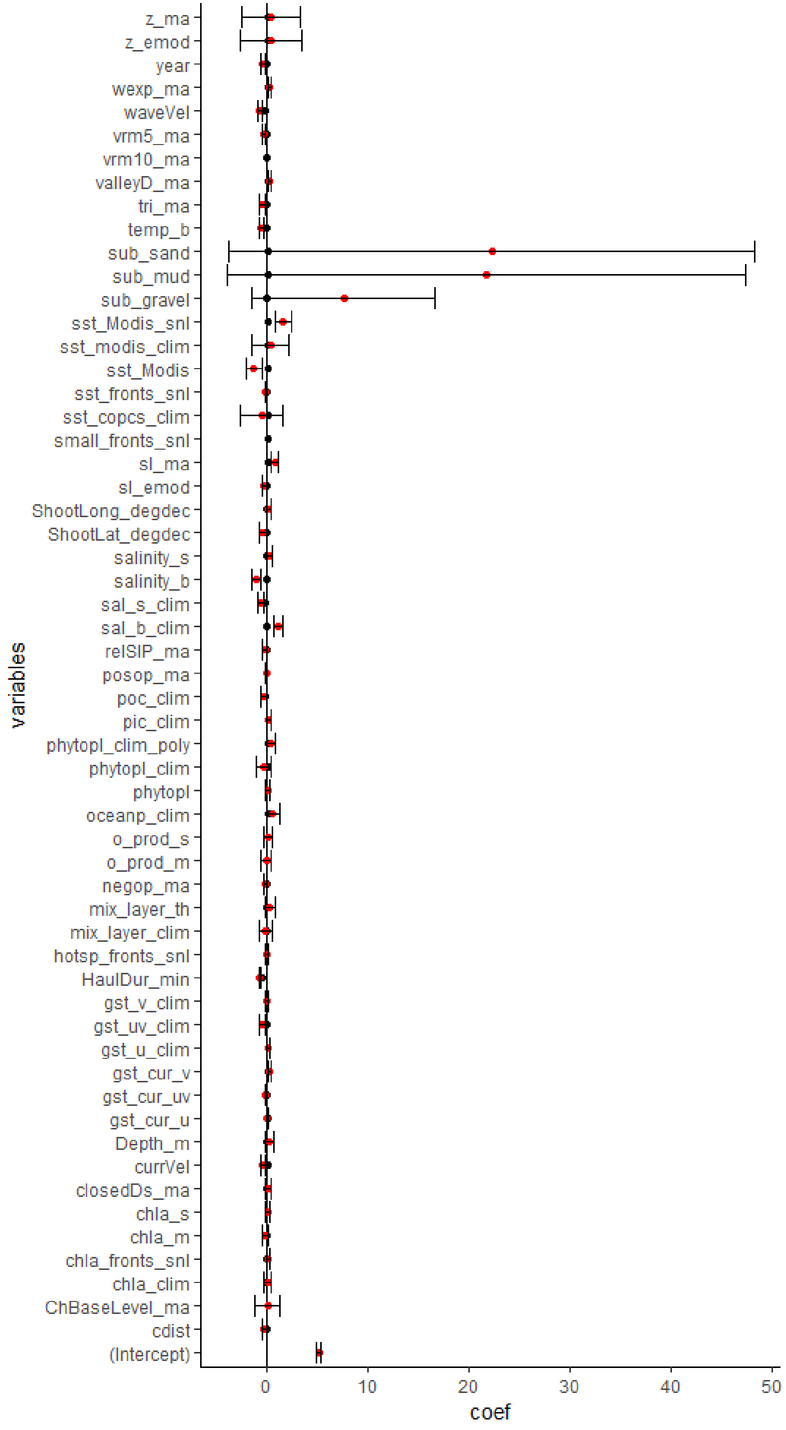


Log(λ)=-0.39
In one iteration



Ridge

Ridge and least squares



substantial reduction in coefficients, particularly those with large standard error

Lasso Regression

(least absolute shrinkage and selection operator)

Ridge regression leaves us with large, complex models (no matter the value of λ) with potentially many noisy variables.

In Lasso, rather than minimising:

$$\hat{\beta}_{ridge} = \min_{\beta}\left\{\sum_{i=1}^{N}(y_i - \hat{y})^2 + \lambda\sum_{j=1}^{p}\beta_j^2\right\}$$

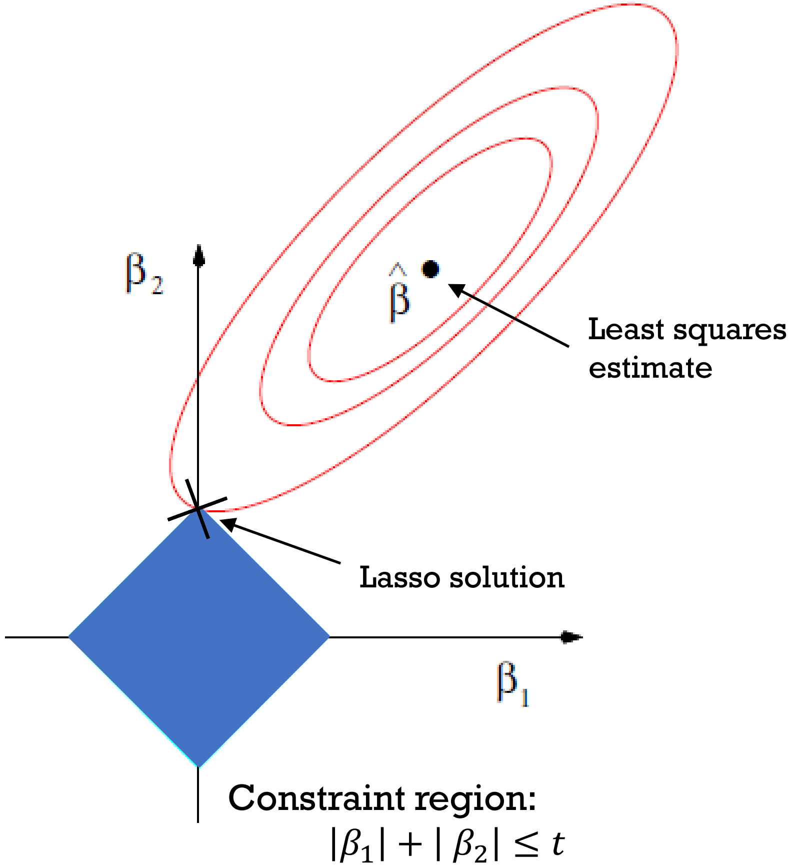
Lasso minimises:

$$\hat{\beta}_{lasso} = \min_{\beta}\left\{\sum_{i=1}^{N}(y_i - \hat{y})^2 + \lambda\sum_{j=1}^{p}|\beta_j|\right\}$$

Ridge Shrinkage penalty, $l_2 penalty$

Tuning parameter ≥0

Lasso Shrinkage penalty, $l_1 penalty$



$$\hat{\beta}_{lasso} = \min_{\beta} \left\{ \sum_{i=1}^{N} (y_i - \hat{y})^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\}$$

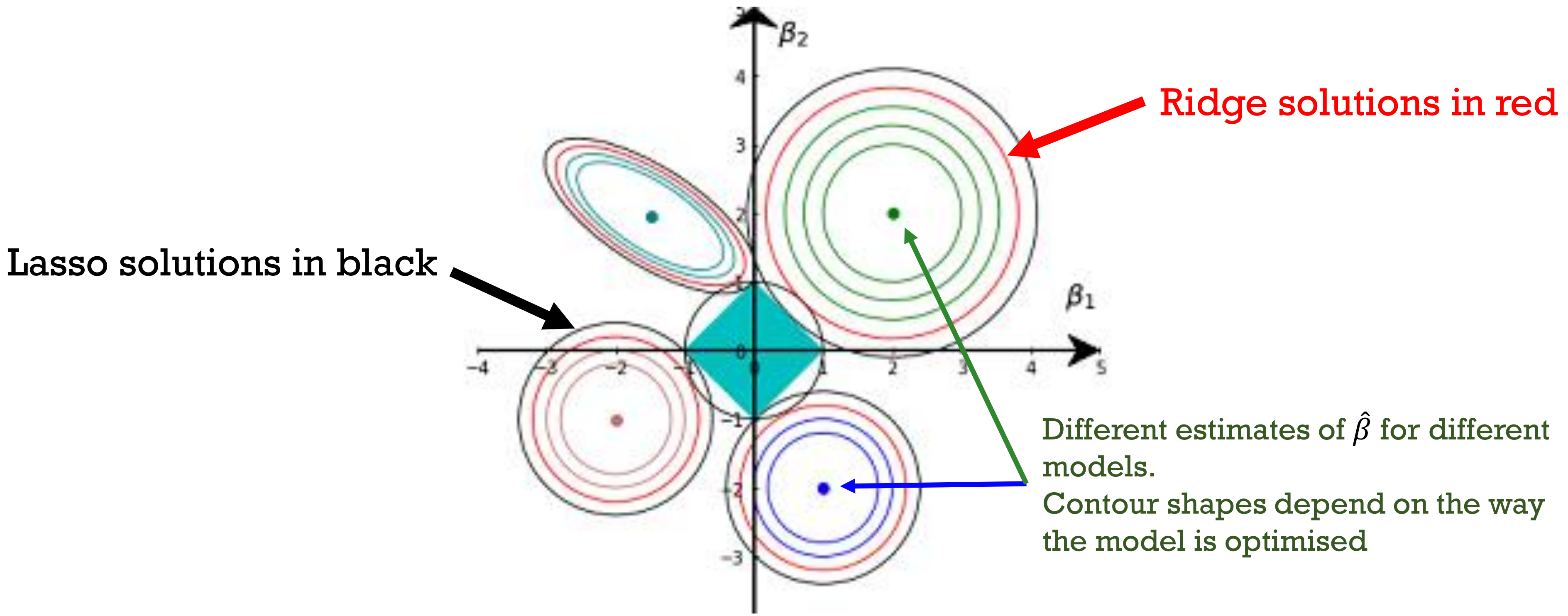
Tuning parameter

Shrinkage penalty

Or alternatively

$$\hat{\beta}_{lasso} = \min_{\beta} \left\{ \sum_{i=1}^{N} (y_i - \hat{y})^2 \right\}$$

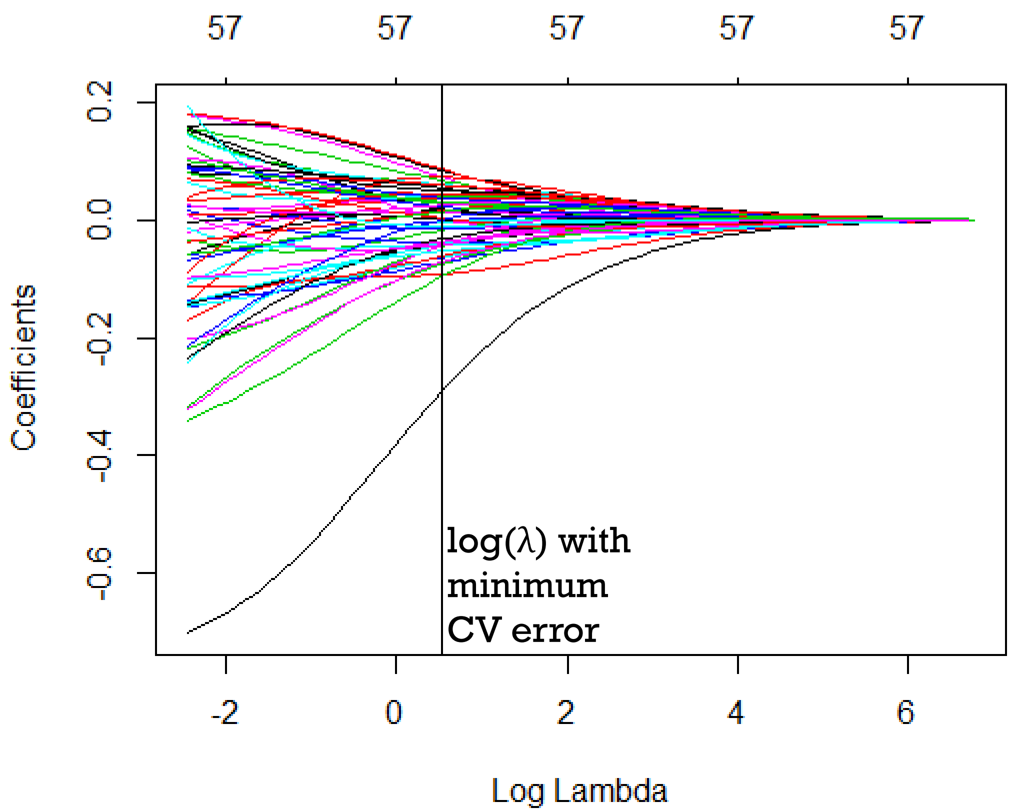
$$subject\ to \sum_{j=1}^{p} |\beta_j| \leq t$$



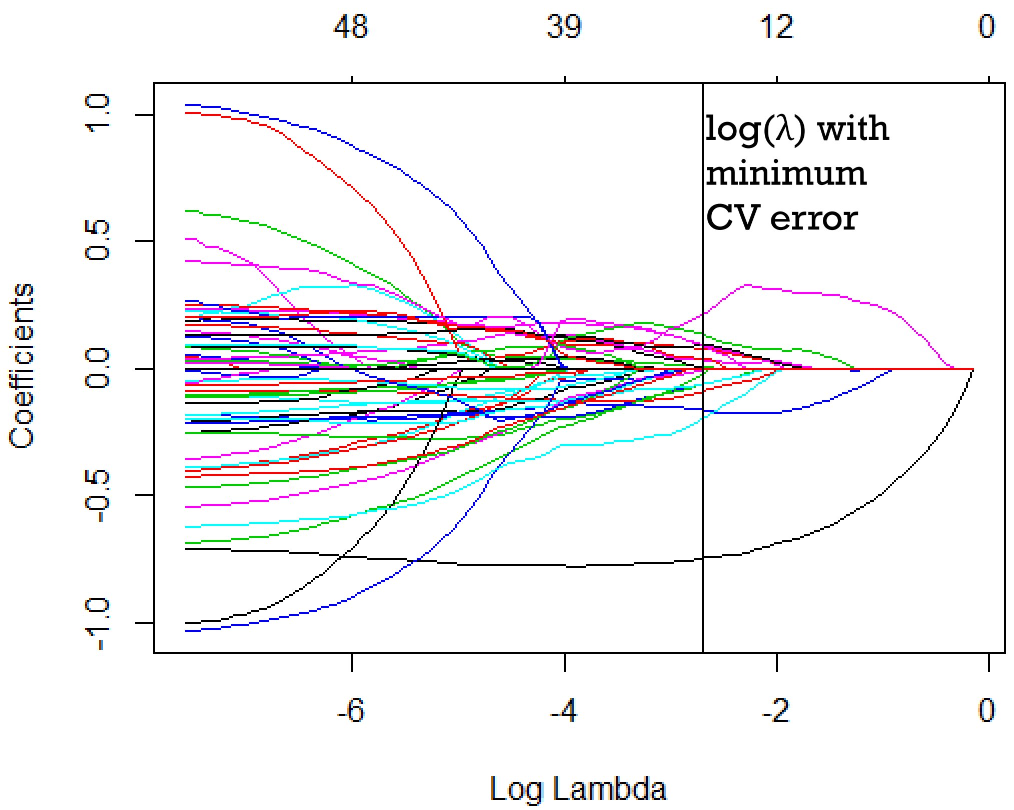
Consistent shrinkage and variable selection by Lasso
Methods are widely applicable across statistics and machine learning

Google Developers

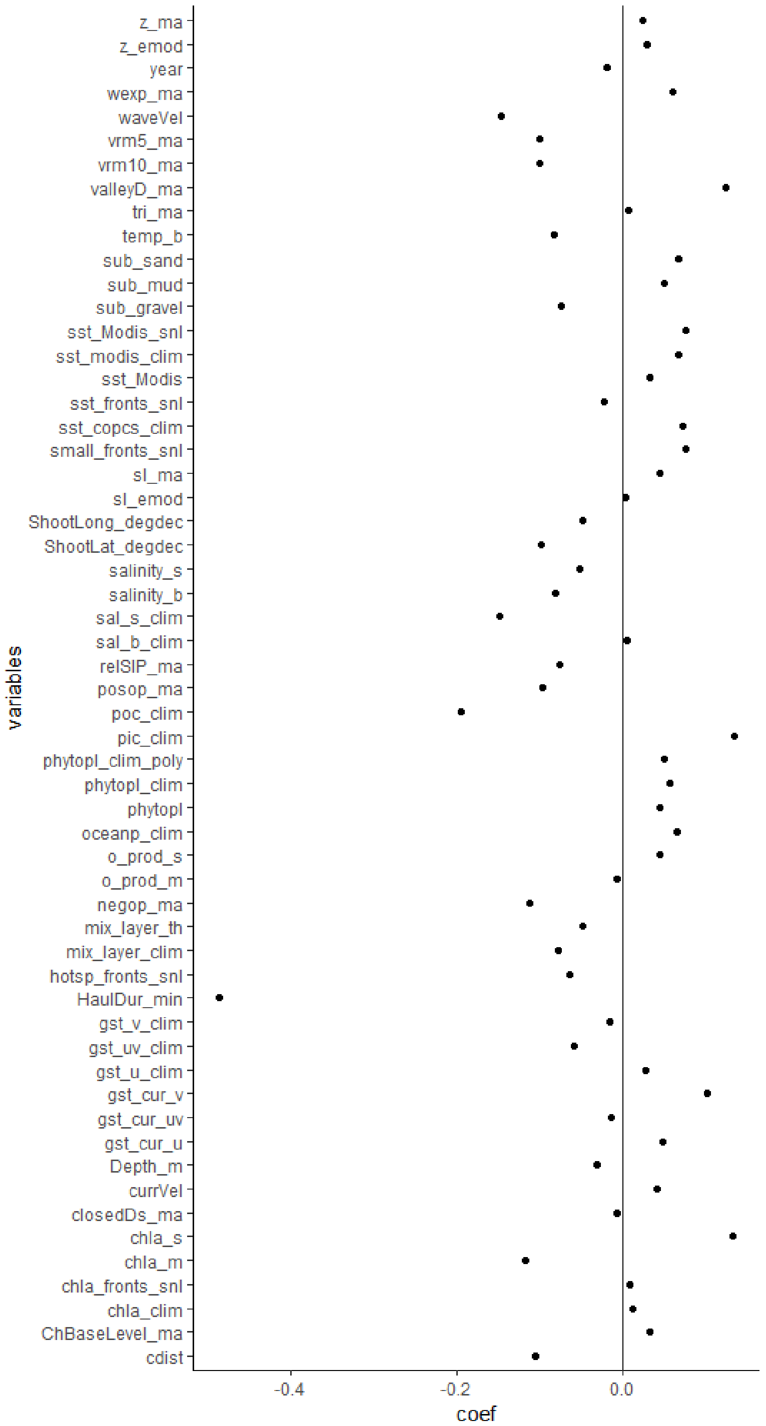
Ridge Shrinkage penalty,
$l_2 penalty$



Lasso Shrinkage penalty,
$l_1 penalty$




```
lassofit1 <- glmnet(log(DensAbund_N_Sqkm) ~.,
                    data=numd.sc, alpha=1, family="gaussian")
plot(lassofit1, "lambda")
```

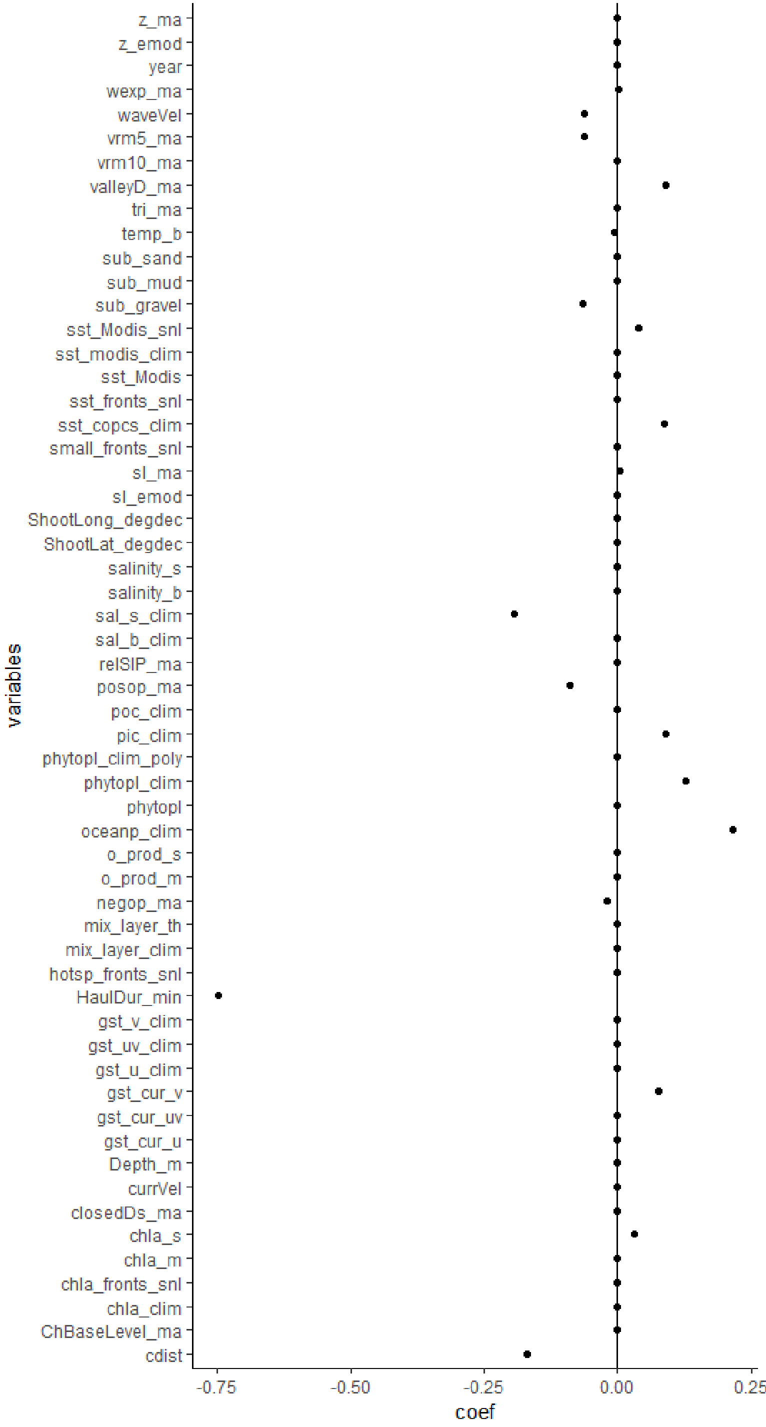


Ridge

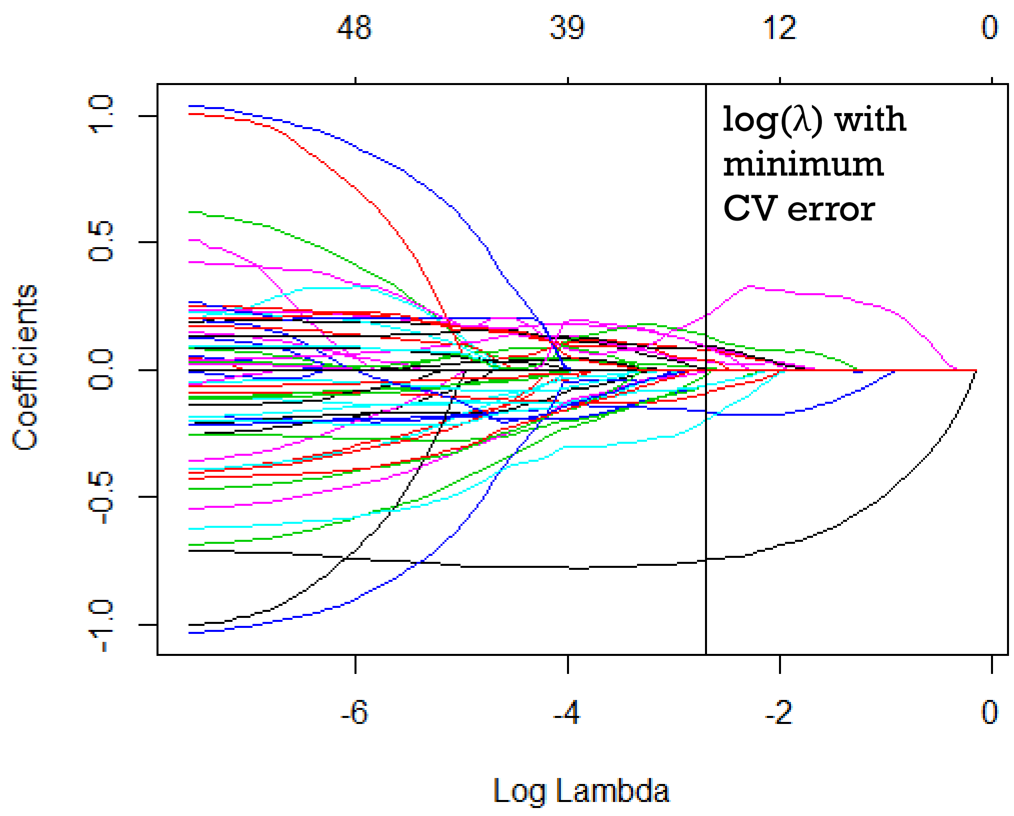
Lasso

Lasso generates sparse models with a subset of p predictors

Ideal if in a setting where a relatively small number of predictors have substantial coefficients, and the remaining predictors have coefficients that are very small or that equal zero.

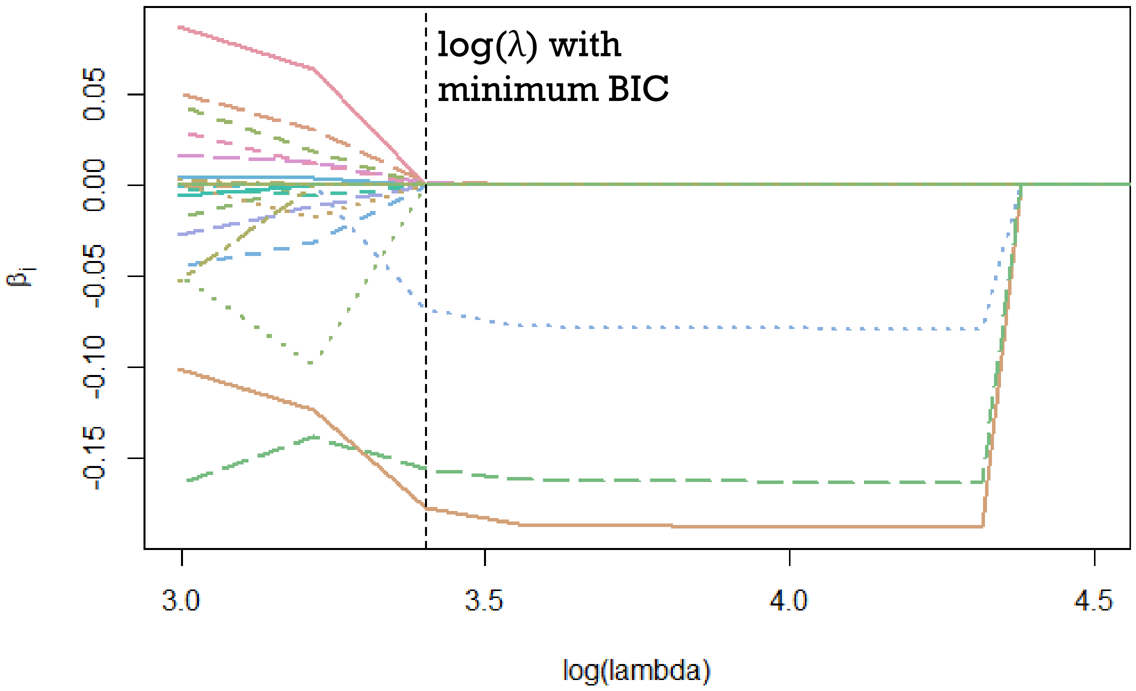


Linear model with L1 regularization (Lasso)




```
lassofit1 <- glmnet(log(DensAbund_N_Sqkm) ~.,
    data=numd.sc, alpha=1, family="gaussian")
plot(lassofit1, "lambda")
```

Linear mixed-effects model with L1 regularization (Lasso)




```
install.packages("glmmLasso")
library(glmmLasso
lassofit3 <- glmmLasso(fix=newformula,
                        rnd =list(Ship=~1),
                        data=numd.sc2,
                    family = gaussian(link="identity"),
                    lambda=165,final.re=TRUE)
```

References

Flom & Cassell (2007) Stopping stepwise: Why stepwise and similar selection methods are bad, and what you should use. *Statistics and Data Analysis,* NESUG https://www.lexjansen.com/pnwsug/2008/DavidCassell-StoppingStepwise.pdf

Hastie, Tibshirani, and Friedman (2009) The elements of statistical learning: Data Mining, Inference, and Prediction. 2nd Edition, Springer Science. Free copy here: https://web.stanford.edu/~hastie/ElemStatLearn//

Dr Google:

https://developers.google.com/machine-learning/crash-course/regularization-for-sparsity/l1-regularization

Python Users:

```
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
```