Stability Selection

Stability path

Stability selection is not a new variable selection technique. Its aim is rather to enhance and improve existing methods.

Stability paths are derived from the concept of regularisation paths. A *regularisation path* is given by the coefficient value of each variable over all regularisation parameters

$$\{\hat{\beta}_j^\lambda; \lambda \in \Lambda, j = 1, \dots, p\}$$

Stability paths are, in contrast, the probability for each variable to be selected when randomly resampling from the data. Suppose for simplicity that n is an even number. Then

1. Randomly select $n/2$ indices from $\{1, \dots, n\}$ without replacement;
2. Run the lasso algorithm by using the $n/2$ observations obtained in step 1 to select a subset of variables $\hat{S}^\lambda = \{j : \hat{\beta}_j^\lambda \neq 0\} \subseteq \{1, \dots, p\}$ for each $\lambda \in \Lambda$;
3. Do steps 1-2 many times;
4. Compute $\hat{\Pi}_j^\lambda$, the fraction of times the jth variable is selected for a given value of λ;

The set of *stable predictors* is defined as

$$\hat{S}_{\text{stable}} = \{j : \max_{\lambda \in \Lambda} \hat{\Pi}_j^\lambda > \pi_{\text{thr}}\}$$

where π_{thr} denote a specified cutoff.

```r
# data generation
rm(list=ls())
set.seed(123)
n = 100
p = 200
# design matrix
X = matrix(runif(n*p), ncol=p)
colnames(X) = paste0("X",1:p)
# betas
beta = c(rep(2,5),rep(0,p-5))
# response
y = 2 + X %*% beta + rnorm(n)
# data
yX = data.frame(y,X)

# stability path
require(glmnet)
```

```
## Loading required package: glmnet
```

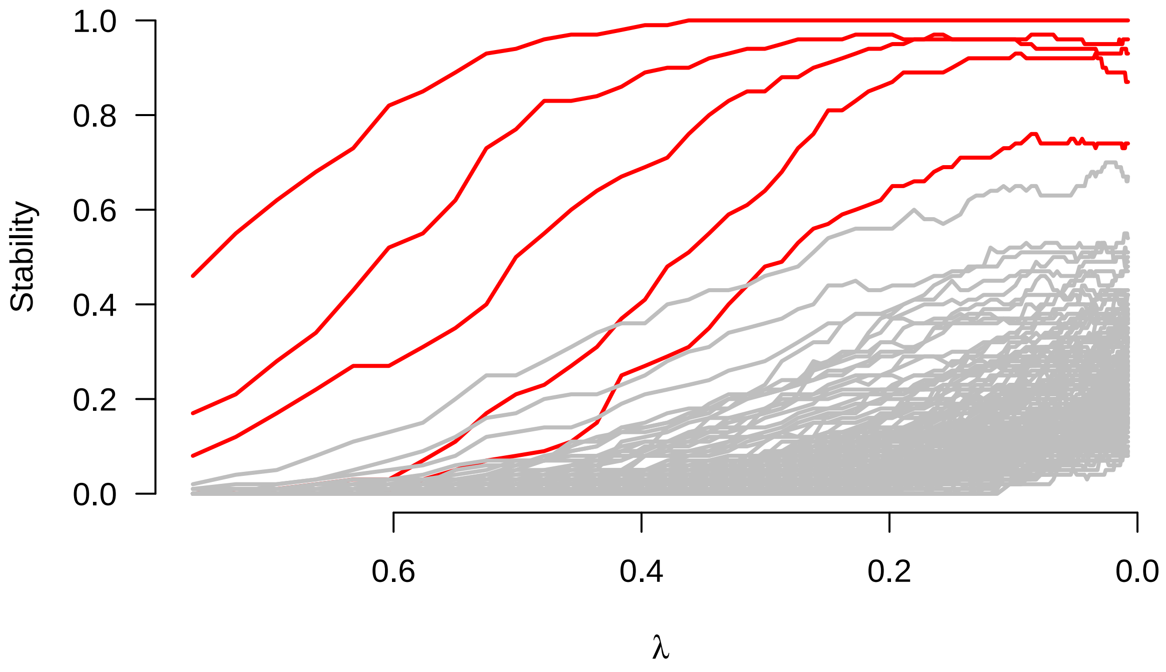
```
## Loading required package: Matrix
```

```
## Loading required package: foreach
```

```
## Loaded glmnet 2.0-13
```

```
fit <- glmnet(X, y)
SS <- array(NA, dim=c(100, p, length(fit$lambda)), dimnames=list(1:100, colnames(X), fit$lambda))
Q <- matrix(NA, 100, length(fit$lambda))
for (i in 1:100) {
  ind <- as.logical(sample(rep(0:1, each=n/2)))
  fit.i <- glmnet(X[ind,], y[ind], lambda=fit$lambda)
  SS[i,,] <- as.matrix(coef(fit.i)[-1,]!=0)
  Q[i,] <- sapply(predict(fit.i, type="nonzero"), length)
}
S <- apply(SS, 2:3, mean)
q <- apply(Q, 2, mean)

l <- fit$lambda
col <- rep("gray", p)
col[1:5]<-"red"
matplot(l, t(S), type="l", lty=1, xlim=rev(range(l)), col=col, lwd=2, las=1, bty="n", xlab=expression(la
```



Stability selection

A problem of many variable selection procedures is that noise variables might be erroneously selected. To improve the selection process and to obtain an error control for the number of falsely selected noise variables Meinshausen and Bühlmann (2010) proposed *stability selection*, which was later enhanced by Shah and Samworth (2013).

Stability selection is a versatile approach, which can be combined with all highdimensional variable selection approaches. It is based on sub-sampling and controls the *per-family error rate* $\mathbb{E}(V)$, where V is the number of false positive variables.

Consider a data set with p variables $x_1, \ldots, x_p$ and a response y. Let $S \subseteq \{1, \ldots, p\}$ be the set of *signal* variables, and let $N = \{1, \ldots, p\} \setminus S$ be the set of *noise* variables.

The set of variables that are *selected* by the statistical learning procedure is denoted by $\hat{S}_n \subseteq \{1, \ldots, p\}$. This set $\hat{S}_n$ can be considered to be an estimator of S, based on a data set with n observations.

In short, for stability selection one proceeds as follows:

1. Select a random subset of size $\lfloor n/2 \rfloor$ of the data, where $\lfloor x \rfloor$ denotes the largest integer $\leq x$.
2. Run a *selection procedure* by using the $\lfloor n/2 \rfloor$ observations obtained in step 1 until $q \leq p$ variables are selected. Let $\hat{S}^b_{\lfloor n/2 \rfloor}$ denotes the set of selected variables.

3.Repeat the steps 1. and 2. for $b = 1, \ldots, B$

4. Compute the relative selection frequencies $\hat{\pi}_j = \frac{1}{B} \sum_{b=1}^{B} I\{j \in \hat{S}^b_{\lfloor n/2 \rfloor}\}$ per variable $j = 1, \ldots, B$, where $I\{\cdot\}$ is the indicator function.
5. Select all base-learners that were selected with a frequency of at least π_{thr}, , where π_{thr} is a pre-specified threshold value. Thus, we obtain a set of stable variables $\hat{S}_{\text{stable}} = \{j : \hat{\pi}_j \geq \pi_{\text{thr}}\}$.

Meinshausen and Bühlmann (2010) show that this selection procedure controls the *per-family error rate* PFER= $\mathbb{E}(V)$. In general it holds that FWER $\leq$ PFER, thus, for a fixed significance level α it holds that PFER-control is more conservative than FWER-control.

An upper bound is given by

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

where q is the number of selected variables per run, p is the number of (possible) variables and π_{thr} is the threshold for selection probability. The theory requires two assumptions to ensure that the error bound holds:

(i) The distribution $\{I_{j \in \hat{S}_{\text{stable}}}, j \in N\}$ needs to be *exchangeable* for all noise variables N

(ii) The original selection procedure must not be worse than random guessing

In practice, assumption (i) essentially means that each noise variable has the same selection probability. Thus, all noise variables should, for example, have the same correlation with the signal variables (and the outcome). For examples of situations where exchangeability is given see Meinshausen and Bühlmann.

Assumption (ii) means that signal variables should be selected with higher probability than noise variables. This assumption is usually not very restrictive as we wouuld expect it to hold for any sensible selection procedure.

Choice of parameters

The stability selection procedure mainly depends on two parameters: the number of selected variables per run q and the threshold value for stable variables π_{thr}.

Meinshausen and Bühlmann (2010) propose to chose $\pi_{\text{thr}} \in (0.6, 0.9)$ and claim that the threshold has little influence on the selection procedure. In general, any value $\in (0.5, 1)$ is potentially acceptable, i.e. a variable should be selected in more than half of the fitted models in order to be considered stable.

The number of selected variables q should be chosen so high that in theory all signal variables S can be chosen. If q was too small, one would inevitably select only a small subset of the signal variables S in the set $\hat{S}_{\text{stable}}$ as $\#\hat{S}_{\text{stable}} \leq \#\hat{S}^b_{\lfloor n/2 \rfloor} = q$ (if $\pi_{\text{thr}} > 0.5$).

The choice of the number of subsamples B is of minor importance as long as it is large enough. Meinshausen and Bühlmann (2010) propose to use $B = 100$ replicates, which seems to be sufficient for an accurate estimation of $\hat{\pi}_j$ in most situations.

In general, we would recommend to choose an upper bound PFER_{max} for PFER and specify either q or π_{thr}, preferably q. The missing parameter can then be computed from

$$\text{PFER}_{\text{max}} = \frac{q^2}{(2\pi_{\text{thr}} - 1)p}$$

For a fixed value q, we can easily vary the desired error bound PFER_{max} by varying the threshold π_{thr} accordingly. As we do not need to re-run the subsampling procedure, this is very easy and fast. In a second step, one should check that the computed value is sensible, i.e. that $\pi_{\text{thr}} \in (0.5, 1)$, or that q is not too small, or that PFER_{max} is not too small or too large. Note that the PFER can be greater than one as it resembles the tolerable expected number of falsely selected noise variables.

The size of the subsamples is no tuning parameter but should always be chosen to be $\lfloor n/2 \rfloor$. This an essential requirement for the derivation of the error bound. Other (larger) subsample sizes would theoretically be possible but would require the derivation of a different error bound for that situation.

One should keep in mind that stability selection controls the per-family error rate, which is very conservative. Specifying the error rate such that $\alpha \leq \text{PFER}_{\text{max}} \leq m\alpha$, with significance level α and m hypothesis tests, might provide a good idea for a sensible error control in high-dimensional settings with FWER control ($\text{PFER}_{\text{max}} = \alpha$) and no multiplicity adjustment ($\text{PFER}_{\text{max}} = m\alpha$) as the extreme cases.

Furthermore, prediction models might not always benefit from stability selection. If the error control is tight, i.e. PFER_{max} is small, the true positive rate is usually smaller than in a cross-validated prediction model without stability selection and the prediction accuracy suffers. Prediction and variable selection are two different goals.

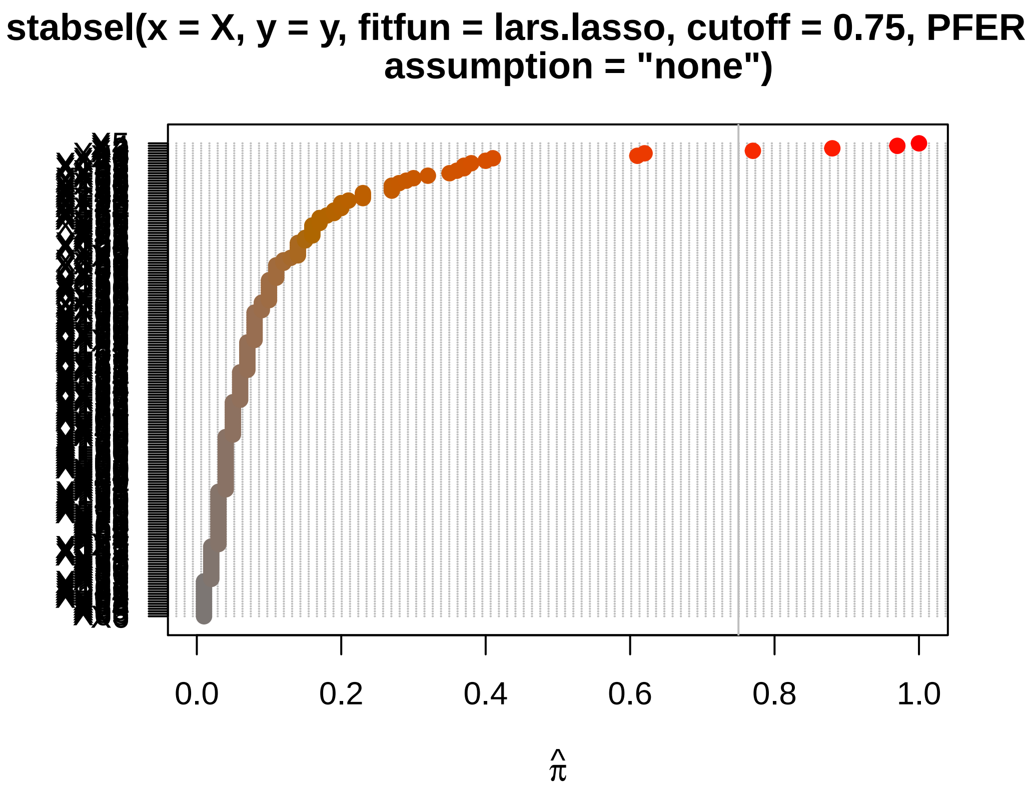
```r
# Stability Selection with stab R package
library(stabs)
```

```
## Loading required package: parallel
```

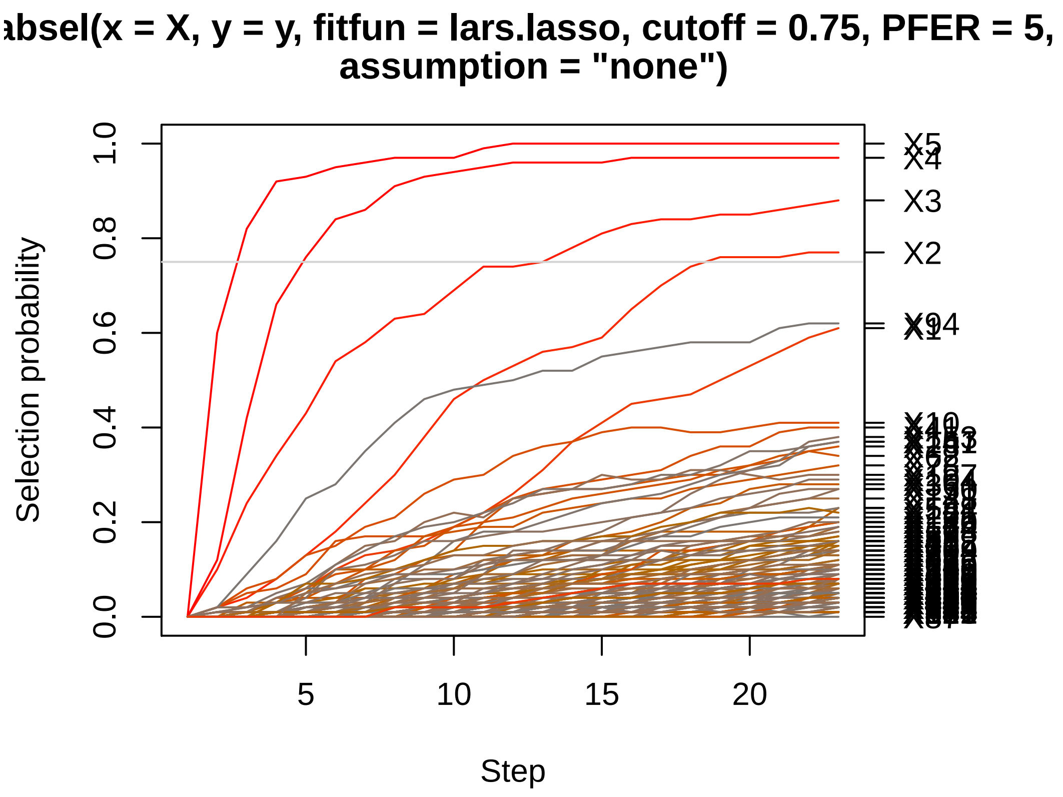
```r
fit <- stabsel(x = X, y = y, fitfun = lars.lasso, cutoff = 0.75,
PFER = 5, assumption ="none")
fit
```

```
##  Stability Selection without further assumptions
##
## Selected variables:
## X2 X3 X4 X5
##  2  3  4  5
##
## Selection probabilities:
##  X46  X74  X87  X89  X93 X131 X133 X147 X159   X6  X65  X66  X69  X84  X88
## 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.01 0.01 0.01 0.01 0.01 0.01
##  X92  X95 X107 X124 X126 X146 X161 X172 X193  X12  X13  X16  X17  X38  X50
## 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.02 0.02 0.02 0.02 0.02 0.02
##  X51  X55  X61  X97 X114 X135 X142 X195   X7  X23  X24  X27  X43  X44  X52
## 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.03 0.03 0.03 0.03 0.03 0.03 0.03
##  X60  X64  X70  X71  X73  X91 X110 X115 X118 X137 X145 X160 X168 X191 X196
## 0.03 0.03 0.03 0.03 0.03 0.03 0.03 0.03 0.03 0.03 0.03 0.03 0.03 0.03 0.03
##  X18  X31  X34  X39  X40  X49  X54  X72  X80 X100 X101 X103 X108 X119 X123
## 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04
## X130 X139 X166 X176 X177 X180 X185  X20  X32  X45  X75  X85  X86 X105 X106
## 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05
## X116 X136 X154 X179 X183 X192  X22  X37  X68  X99 X112 X134 X152 X157 X171
## 0.05 0.05 0.05 0.05 0.05 0.05 0.06 0.06 0.06 0.06 0.06 0.06 0.06 0.06 0.06
## X174 X186 X188  X33  X42  X57  X83 X117 X121 X128 X132 X143 X155 X194 X197
## 0.06 0.06 0.06 0.07 0.07 0.07 0.07 0.07 0.07 0.07 0.07 0.07 0.07 0.07 0.07
##   X9  X63  X96  X98 X113 X120 X169 X175 X187 X190 X199 X200  X15  X48 X102
## 0.08 0.08 0.08 0.08 0.08 0.08 0.08 0.08 0.08 0.08 0.08 0.08 0.09 0.09 0.09
## X138  X11  X30  X35 X111 X122 X150 X156 X158 X165  X19  X26  X29  X67 X149
## 0.09 0.10 0.10 0.10 0.10 0.10 0.10 0.10 0.10 0.10 0.11 0.11 0.11 0.11 0.11
## X189  X56 X163  X53   X8  X79  X90 X109 X129 X164  X21  X82  X47  X76  X77
## 0.11 0.12 0.12 0.13 0.14 0.14 0.14 0.14 0.14 0.14 0.15 0.15 0.16 0.16 0.16
##  X81 X182  X25  X59 X198 X178 X162 X184  X14 X140 X173 X125  X58 X144 X181
## 0.16 0.16 0.17 0.17 0.17 0.18 0.19 0.19 0.20 0.20 0.20 0.21 0.23 0.23 0.23
## X148 X151 X170  X36 X104 X127  X78  X62  X28 X141 X167 X153  X41  X10   X1
## 0.27 0.27 0.27 0.28 0.29 0.30 0.32 0.35 0.36 0.37 0.37 0.38 0.40 0.41 0.61
##  X94   X2   X3   X4   X5
## 0.62 0.77 0.88 0.97 1.00
##
## ---
## Cutoff: 0.75; q: 22; PFER (*):  4.84
##    (*) or expected number of low selection probability variables
## PFER (specified upper bound):  5
## PFER corresponds to signif. level 0.0242 (without multiplicity adjustment)
```

```r
plot(fit)
```



```r
plot(fit, type="path")
```



absel(x = X, y = y, fitfun = lars.lasso, cutoff = 0.75, PFER = 5,
assumption = "none")
Selection probability
0.0
0.2
0.4
0.6
0.8
1.0
5
10
15
20
Step
X5
X4
X3
X2
X94
X1
X10