# Stability selection in high-dimensional regression

Enrico Moriggi, Matteo Suardi

2025-11-16

## Contents

**1 Introduction** **2**

**2 Methodological background** **3**

2.1 Lasso and regularisation path . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3

2.2 Stability selection and error control . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3

2.3 Evaluation and error bound (PFER) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3

**3 Simulation study: from instability to error control** **5**

3.1 Design A: stability selection with correlated predictors . . . . . . . . . . . . . . . . . . . . . . 5

3.2 Design B: complementary pairs stability selection and error control . . . . . . . . . . . . . . . 14

3.3 Conclusions: simulation study . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 22

**4 Real-data application: stability selection on kidney transplant gene expression** **23**

4.1 Data description and pre-processing . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 23

**5 General conclusions** **38**

**6 Further improvements and perspectives** **39**

```r
library(glmnet)
```

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

```
## Loaded glmnet 4.1-10
```

```r
library(stabs)
```

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

```r
set.seed(123)
```

## 1 Introduction

In high-dimensional regression, particularly when the number of predictors largely exceeds the sample size ($p \gg n$), standard penalised methods like the Lasso provide sparse solutions but often suffer from severe instability. As noted by Meinshausen and Bühlmann (2010), small perturbations in the data (such as removing a few observations) can lead to arbitrarily different active sets, casting doubt on the scientific reliability of the selected variables.

Stability selection addresses this structural instability by combining resampling techniques with high-dimensional selection algorithms. Unlike standard cross-validation, which aims to minimise prediction error (often retaining too many noise variables), stability selection focuses on **variable recovery**, seeking predictors that persist across repeated random splits of the data. Crucially, this framework provides a theoretical upper bound on the expected number of False Positives (Per-Family Error Rate, PFER).

In this report, we analyse the performance of stability selection through a structured approach combining simulations and real data:

- **Design A (correlated predictors):** we simulate a realistic scenario with block-correlated regressors. Here, we manually implement Complementary Pairs Stability Selection (CPSS) to illustrate how the method handles multicollinearity and "signal dilution" compared to a standard single-run Lasso.
- **Design B (theoretical validation):** we employ an orthogonal design to strictly verify the error control guarantees, explicitly connecting the empirical results to the PFER upper bound derived by Meinshausen and Bühlmann (2010).
- **Real-world application:** we extend the methodology to the `Einecke2010Kidney` high-dimensional gene expression dataset, contrasting the volatile selection of a logistic Lasso with the robust signature identified by stability selection.

# 2 Methodological background

## 2.1 Lasso and regularisation path

We consider a linear regression model with response $Y \in \mathbb{R}^n$ and predictor matrix $X \in \mathbb{R}^{n \times p}$. The Lasso estimator is defined as the solution to the $\ell_1$-penalised optimisation problem:

$$\hat{\beta}(\lambda) = \arg\min_{\beta \in \mathbb{R}^p} \left\{ \frac{1}{2n} \|Y - X\beta\|_2^2 + \lambda \|\beta\|_1 \right\}$$

where $\lambda \geq 0$ is the tuning parameter governing the trade-off between the goodness of fit and the sparsity of the solution. The collection of estimates $\{\hat{\beta}(\lambda) : \lambda \in \Lambda\}$ for a grid of penalty parameters $\Lambda$ constitutes the regularisation path. While effective for variable selection, the Lasso path can be unstable in high-dimensional settings ($p \gg n$), where small perturbations in the data may result in different active sets $\hat{S}(\lambda) = \{j : \hat{\beta}_j(\lambda) \neq 0\}$.

## 2.2 Stability selection and error control

To address the instability of single-run selection methods, stability selection aggregates selection results across repeated subsampling of the data. Modern implementations, such as the `stabs` package used in this analysis, typically employ **Complementary Pairs Stability Selection (CPSS)**.

Instead of drawing simple random subsamples, CPSS draws $B$ pairs of subsamples $\{(I_{2b-1}, I_{2b})\}_{b=1}^{B}$, where each pair forms a disjoint partition of the dataset $\{1, \ldots, n\}$ of size $\lfloor n/2 \rfloor$. By using both halves of the data in each iteration, this approach ensures that every observation is used equally, significantly reducing the variance of the stability estimators compared to standard subsampling.

For a given $\lambda \in \Lambda$, the empirical *selection probability* of variable $j$ is defined as:

$$\hat{\pi}_j(\lambda) = \frac{1}{2B} \sum_{i=1}^{2B} \mathbf{1}\{j \in \hat{S}_\lambda^{(i)}\}$$

where $\hat{S}_\lambda^{(i)}$ denotes the set of variables selected by the Lasso on the $i$-th subsample. The set of curves $\{\hat{\pi}_j(\lambda)\}_{\lambda \in \Lambda}$ forms the *stability path*. It is important to emphasize that the empirical frequency $\hat{\pi}_j(\lambda)$ estimates the selection probability conditional on the subsample size $\lfloor n/2 \rfloor$, denoted as $P(j \in \hat{S}_{\lfloor n/2 \rfloor})$. Since variable selection is generally harder with fewer observations (lower statistical power), these probabilities are typically lower than those associated with the full sample size $n$. Consequently, stability selection is an inherently conservative procedure: a variable must be robust enough to be consistently detected even with only half of the available information.

## 2.3 Evaluation and error bound (PFER)

Variables are considered "stable" if their selection probability exceeds a pre-defined threshold $\pi$ (typically $\pi \in [0.6, 0.9]$). The set of stable predictors is determined by the maximum selection probability over the path:

$$\hat{S}_{stab} = \{j : \max_{\lambda \in \Lambda} \hat{\pi}_j(\lambda) \geq \pi_{thr}\}$$

A fundamental advantage of stability selection is the ability to control the **Per-Family Error Rate (PFER)**, defined as the expected number of false positives $E(V)$. Meinshausen and Bühlmann (2010) derived a theoretical upper bound which relates the error rate to the stability threshold $\pi$ and the average number of selected variables $q$:

$$E(V) \leq \frac{q^2}{(2\pi - 1)p}$$

This bound assumes that the selection procedure is better than random guessing and holds under mild conditions (exchangeability). It allows us to explicitly control the amount of noise entering the model: by fixing an acceptable upper bound for $E(V)$ and choosing $q$ (e.g., by restricting the regularization path to the first $q$ variables), we can derive the necessary threshold $\pi$ to guarantee high-confidence selection.

## 3 Simulation study: from instability to error control

In this section, we employ two simulated designs to illustrate distinct properties of the stability selection framework. Both designs rely on the linear model formulation introduced in Section 2, yet they address complementary objectives:

- **Design A (correlated predictors - Section 3.1):** this scenario mimics a realistic high-dimensional setting where predictors exhibit a Toeplitz correlation structure (organized in blocks). The goal is to demonstrate how standard Lasso selection suffers from instability in the presence of collinearity, often selecting correlated surrogates instead of the true active set, and to evaluate how stability selection (implemented via complementary pairs) mitigates this signal dilution.
- **Design B (independence - Section 3.2):** this scenario adopts an independent design that closely aligns with the theoretical assumptions of Meinshausen and Bühlmann (2010). In this controlled environment, we strictly validate the **Complementary Pairs Stability Selection (CPSS)** procedure, explicitly verifying that the empirical False Discovery Rate respects the theoretical upper bound on the **Per-Family Error Rate (PFER)**.

By contrasting these two designs, we disentangle the impact of feature correlation from the baseline performance of the algorithm, establishing a robust ground before applying the methodology to real-world data.

### 3.1 Design A: stability selection with correlated predictors

In this first example, we study a linear regression setting where predictors are organised into correlated blocks. This design mimics realistic high-dimensional applications (e.g., omics or imaging), in which groups of variables tend to be co-expressed. We demonstrate that, while a standard Lasso fit is sensitive to multicollinearity, stability selection can better distinguish robust predictors, provided we look at their behaviour across the entire regularisation path.

#### 3.1.1 Data generation

We simulate a linear regression model with $n = 150$ observations and $p = 300$ predictors. Predictors are organised into three blocks: a strongly correlated block (1–100, $\rho = 0.7$), a moderately correlated block (101–200, $\rho = 0.3$), and an independent block (201-300, $\rho = 0$). The 6 active variables are all located in the first (high correlation) block. This represents a “worst-case scenario” for variable selection.

Sample size and number of predictors:

```r
n <- 150
p <- 300
```

Indexes of truly active variables (all in the first block):

```r
S_true <- c(1, 2, 3, 10, 20, 30)
s <- length(S_true)
```

Regression coefficients (non-zero only on `S_true`):

```r
beta <- rep(0, p)
beta[S_true] <- c(1.5, 1.2, 1.0, 0.8, 0.8, 0.6) # different effect sizes
```

Build a block-correlated covariance matrix for X:

```
Sigma <- diag(p)
```

Block 1: strong correlation (variables 1–100):

```
rho1 <- 0.7
Sigma[1:100, 1:100] <- rho1
diag(Sigma[1:100, 1:100]) <- 1
```

Block 2: moderate correlation (variables 101–200):

```
rho2 <- 0.3
Sigma[101:200, 101:200] <- rho2
diag(Sigma[101:200, 101:200]) <- 1
```

Block 3 (201–300) remains independent (identity).

Small ridge term to improve numerical stability:

```
Sigma <- Sigma + 1e-6 * diag(p)
```

Simulate $X \sim N(0, \Sigma)$:

```
library(MASS)
X <- MASS::mvrnorm(n = n, mu = rep(0, p), Sigma = Sigma)
```

Choose noise variance to obtain SNR = 2:

```
signal_var <- as.numeric(t(beta) %*% Sigma %*% beta)
SNR <- 2
sigma2 <- signal_var / SNR
```

Generate response: $Y = X\beta + \varepsilon$

```
eps <- rnorm(n, mean = 0, sd = sqrt(sigma2))
y <- as.numeric(X %*% beta + eps)
```

Colours for plots (true signals red, others grey):

```
col_vec <- rep("lightgrey", p)
col_vec[S_true] <- "red"
```

### 3.1.2 Lasso fit and regularisation path

Before introducing stability selection, we first analyse the behaviour of the lasso when applied once to the full dataset. This preliminary step provides a baseline for comparison: by examining the regularisation path and the set of predictors selected at the cross-validated value $\lambda_{1se}$, we can assess how the lasso reacts under strong multicollinearity. In the block–correlated design considered here, this analysis reveals that the lasso recovers part of the true signal but also selects several correlated substitutes, motivating the need for a more robust and stable selection approach.

Lasso fit on the full dataset:

```r
fit <- glmnet(X, y)
```

Lambda grid used by glmnet:

```r
Lambda <- fit$lambda

Lambda
```

```
##   [1] 4.66564969 4.45358872 4.25116623 4.05794417 3.87350434 3.69744759
##   [7] 3.52939289 3.36897654 3.21585136 3.06968595 2.93016398 2.79698351
##  [13] 2.66985629 2.54850720 2.43267362 2.32210485 2.21656160 2.11581545
##  [19] 2.01964837 1.92785224 1.84022838 1.75658716 1.67674756 1.60053680
##  [25] 1.52778993 1.45834952 1.39206528 1.32879377 1.26839804 1.21074739
##  [31] 1.15571705 1.10318793 1.05304634 1.00518376 0.95949660 0.91588600
##  [37] 0.87425757 0.83452121 0.79659094 0.76038465 0.72582399 0.69283417
##  [43] 0.66134379 0.63128469 0.60259183 0.57520310 0.54905923 0.52410364
##  [49] 0.50028232 0.47754372 0.45583862 0.43512005 0.41534317 0.39646519
##  [55] 0.37844523 0.36124431 0.34482520 0.32915236 0.31419188 0.29991137
##  [61] 0.28627994 0.27326807 0.26084762 0.24899169 0.23767463 0.22687196
##  [67] 0.21656028 0.20671728 0.19732166 0.18835308 0.17979215 0.17162031
##  [73] 0.16381991 0.15637404 0.14926660 0.14248220 0.13600616 0.12982447
##  [79] 0.12392375 0.11829123 0.11291471 0.10778256 0.10288368 0.09820745
##  [85] 0.09374377 0.08948297 0.08541583 0.08153355 0.07782773 0.07429034
##  [91] 0.07091373 0.06769059 0.06461395 0.06167714 0.05887382 0.05619792
##  [97] 0.05364363 0.05120545 0.04887808 0.04665650
```

Cross-validated choice of lambda:

```r
cvfit <- cv.glmnet(X, y)
lambda_1se <- cvfit$lambda.1se

lambda_1se
```

```
## [1] 1.268398
```

Variables selected by the lasso at $\lambda_{1se}$:

```r
S_hat_lasso <- which(coef(fit, s = lambda_1se)[-1] != 0)

S_hat_lasso
```

```
##  [1]  1  2  9 10 20 27 30 34 48 60 74 86 93 97
```

True positives and missed signals:

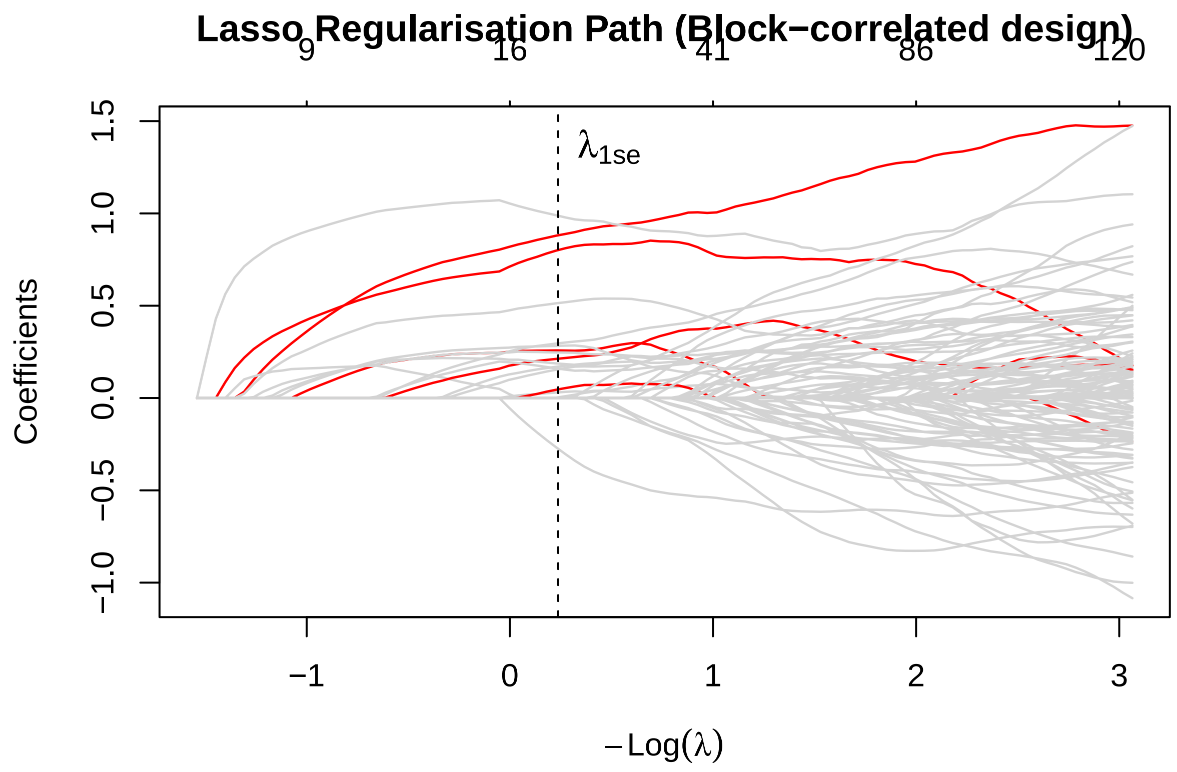
```r
intersect(S_hat_lasso, S_true) # correctly recovered signals
```

```
## [1]  1  2 10 20 30
```

```
setdiff(S_true, S_hat_lasso) # true signals missed by the lasso
```

```
## [1] 3
```

```
plot(fit, xvar = "lambda", col = col_vec, lwd = 1.2,
     main = "Lasso Regularisation Path (Block-correlated design)")

abline(v = log(lambda_1se), lty = 2)

text(x = log(lambda_1se), y = max(coef(fit)[-1,]) * 0.9,
     labels = expression(lambda["1se"]), pos = 4, cex = 1.2, font = 2)
```



The plot displays the lasso regularisation path for the block–correlated design. The truly active variables (highlighted in red) belong to the first block with strong correlation ($\rho = 0.7$). As $\lambda$ decreases, several grey curves from the same block enter the model with similar shapes, indicating that correlated predictors act as substitutes.

At $\lambda_{1se}$, the lasso selects 14 predictors: it correctly recovers 5 out of the 6 true signals, but it also includes 9 spurious variables, most of them coming from the highly correlated first block. This behaviour illustrates how, in the presence of multicollinearity, a single lasso fit may mix true signals with correlated noise variables, leading to an unstable and harder-to-interpret model.

### 3.1.3 Stability paths via Complementary Pairs (CPSS)

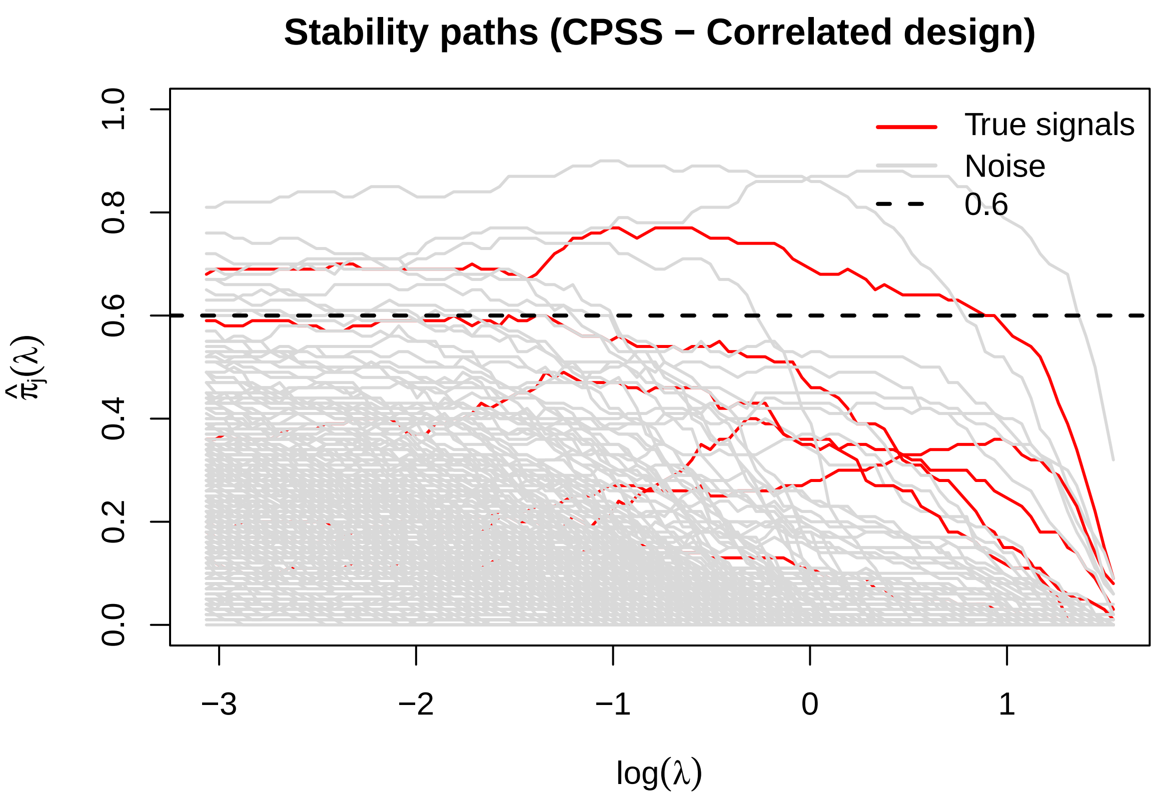
To address the instability observed in the single Lasso fit and to strictly control variance, we implement Complementary Pairs Stability Selection (CPSS) (Shah & Samworth, 2013) manually for this design. Instead of simple random subsampling, we repeatedly split the data into two disjoint halves $(I_{2b-1}, I_{2b})$ of size $n/2$. The Lasso is fitted on both halves using the same $\lambda$ grid.

```
B <- 50 # 50 iterations = 100 fits (2 per iteration)
S_hat_cpss <- array(NA, dim = c(2 * B, p, length(Lambda)))

set.seed(123)

for (b in 1:B) {
  perm <- sample(1:n)
  ind1 <- perm[1:(n/2)] # first half
  ind2 <- perm[(n/2 + 1):n] # second half

  fit_1 <- glmnet(X[ind1, ], y[ind1], lambda = Lambda)
  fit_2 <- glmnet(X[ind2, ], y[ind2], lambda = Lambda)

  S_hat_cpss[2*b - 1, , ] <- as.matrix(coef(fit_1)[-1, ] != 0)
  S_hat_cpss[2*b,     , ] <- as.matrix(coef(fit_2)[-1, ] != 0)
}

pi_hat <- apply(S_hat_cpss, 2:3, mean)
```

In this manual implementation, we deliberately fixed the regularization sequence $\Lambda$ to be identical across the full dataset and all subsamples. While standard practice in exploratory stability selection (Meinshausen and Bühlmann, 2010), it is worth nothing that the effective penalization strength relative to the sample size changes when $n$ drops to $\lfloor n/2 \rfloor$. This fixed-grid approach allows for a direct visual comparison of coefficient paths, but it contributes to the conservative nature of the selection: at a fixed numerical $\lambda$, the subsampled Lasso (with higher variance and lower power) tends to select fewer variables than the full-model Lasso. In the subsequent sections (Design B and Real data), we transition to the `stabs` package, which abstracts this issue by focusing on the $q$-parameter (number of selected variables) rather than the raw $\lambda$ values.

### 3.1.4 Visualization of the stability path

```
tau <- 0.6
cols_all <- rep("grey85", p)
cols_all[S_true] <- "red"

par(mar = c(5, 5, 3, 2))
matplot(
  log(Lambda), t(pi_hat),
  type = "l", lty = 1, lwd = 1.5, col = cols_all,
  xlab = expression(log(lambda)),
  ylab = expression(hat(pi)[j](lambda)),
  main = "Stability paths (CPSS - Correlated design)",
  ylim = c(0, 1)
)
abline(h = tau, lty = 2, lwd = 2)
legend("topright", legend = c("True signals", "Noise", expression(tau=0.6)),
       col = c("red", "grey85", "black"), lty = c(1, 1, 2), lwd = 2, bty="n")
```



The figure displays the CPSS stability paths $\hat{\pi}_j(\lambda)$ for the correlated design.
The true signals (red) stand out from the noise variables (grey), but most of them attain only moderate selection probabilities.
Only a single active predictor exceeds the stability threshold $\tau = 0.6$, reflecting the conservative nature of CPSS when subsampling reduces the effective sample size. Despite this conservativeness, most of the true signals lie above the noise cloud, showing that CPSS successfully suppresses false positives while still highlighting the most robust components of the underlying signal, even in the presence of strong correlation among predictors.

### 3.1.5 Stability-selected set (maximum probability criterion)

The stability paths reveal distinct behaviors. However, due to the strong block correlation ($\rho = 0.7$), we anticipate a “signal dilution” effect: the Lasso splits its selection frequency among correlated surrogates, preventing some true signals from reaching high stability scores.

To define the stable set, we use the restricted maximization criterion ($q_\lambda \leq 20$) to avoid the noise tail.

```
q_hat <- apply(pi_hat, 2, sum)
valid_indices <- which(q_hat <= 20)

pi_max <- apply(pi_hat[, valid_indices], 1, max)

S_stab <- which(pi_max >= tau)

cat("Selected variables (CPSS):", S_stab, "\n")
```

```
## Selected variables (CPSS): 2 9 93 236
```

```r
cat("True signals recovered:", intersect(S_stab, S_true), "\n")
```

```
## True signals recovered: 2
```

```r
cat("False Positives:", setdiff(S_stab, S_true), "\n")
```

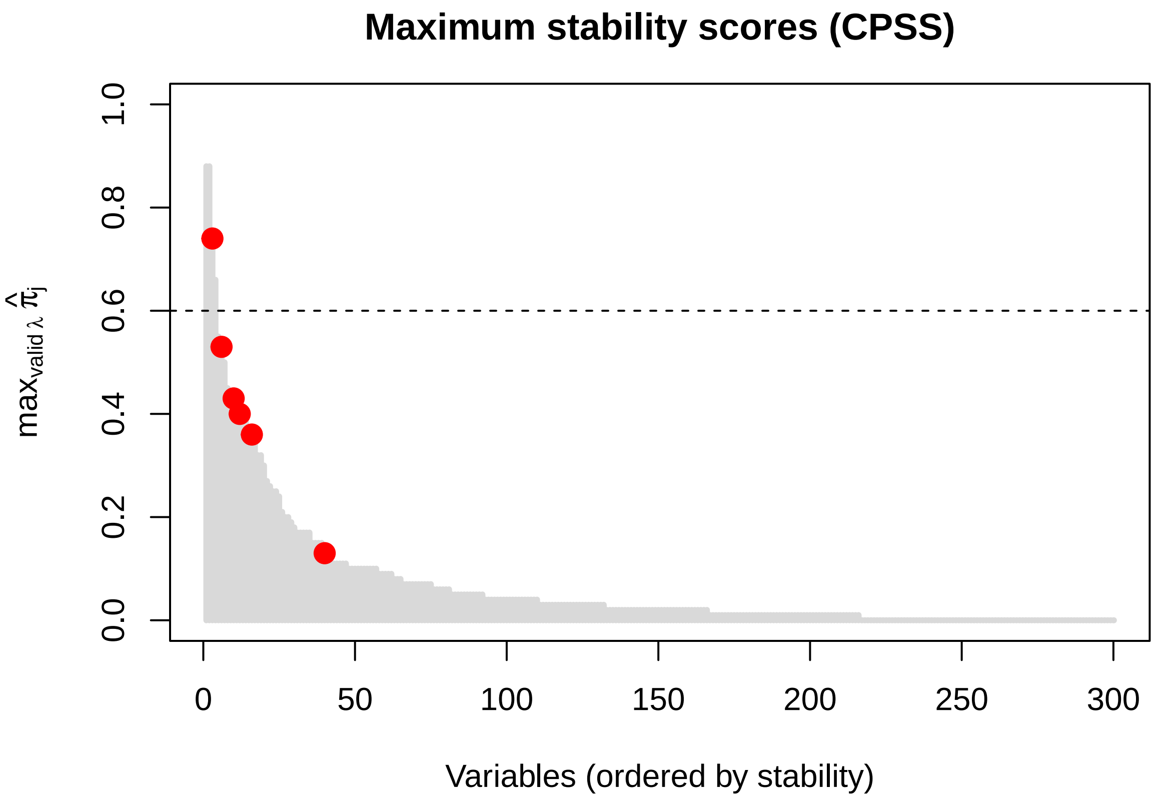
```
## False Positives: 9 93 236
```

```r
q_limit <- 20
PFER_A <- (q_limit^2) / ((2 * tau - 1) * p)

cat("Theoretical PFER Upper Bound (worst-case):", round(PFER_A, 2), "\n")
```

```
## Theoretical PFER Upper Bound (worst-case): 6.67
```

We successfully recovered only **1 out of 6** true signals (Variable 2). This low sensitivity is a direct consequence of the block-correlation structure. As the Lasso arbitrarily alternates between correlated surrogates across subsamples, the selection probability is “diluted” among the group, preventing other active variables from reaching the threshold $\tau = 0.6$.

```r
ord_A <- order(pi_max, decreasing = TRUE)
pi_ordA <- pi_max[ord_A]

par(mar = c(5, 5, 3, 2))
plot(x = seq_along(pi_ordA), y = pi_ordA, type = "h", lwd = 3, col = "grey85",
     xlab = "Variables (ordered by stability)",
     ylab = expression(max[valid~lambda] ~ hat(pi)[j]),
     main = "Maximum stability scores (CPSS)", ylim = c(0, 1))

abline(h = tau, lty = 2)

is_signal_ord <- ord_A %in% S_true
points(x = which(is_signal_ord),
       y = pi_ordA[is_signal_ord],
       pch = 19,
       cex = 1.4,
       col = "red")
```



The bar plot vividly illustrates the challenge of strong multicollinearity ($\rho = 0.7$). Due to signal dilution (vote-splitting), the Lasso arbitrarily alternates between correlated surrogates across subsamples. Consequently, the selection probability is split among the true predictors (red dots), preventing them from reaching the strict stability threshold $\tau = 0.6$. While this results in low sensitivity, the method's value lies in its comparison with the standard Lasso. The single Lasso fit at $\lambda_{1se}$ selected 14 variables (9 false positives). Stability selection, even in this "hostile" scenario, filtered out the vast majority of that noise, highlighting the inherent ambiguity of the data rather than providing a misleadingly confident model.

#### 3.1.6 Discussion of Design A results

The results of Design A illustrate the inherent trade-off when applying standard stability selection to highly correlated data (a violation of the *Exchangeability condition*, Meinshausen 2010).

1. **Violation of exchangeability**: we recovered only a subset of true signals. This is theoretically expected: the strong block correlation ($\rho = 0.7$) violates the exchangeability assumption required for optimal selection (Meinshausen and Bühlmann, 2010). When signal and noise variables are highly correlated, they become "exchangeable" to the Lasso, which arbitrarily splits the selection probability among them.

2. **Signal dilution (vote splitting)**: since the true variables are highly correlated ($\rho = 0.7$), the standard Lasso arbitrarily alternates between them across subsamples. This "vote splitting" prevents individual variables from reaching the strict threshold $\tau$. While the *Randomized Lasso* (Meinshausen and Bühlmann, 2010) is known to mitigate this issue by introducing random penalties to de-correlate selections, we deliberately employed the standard Lasso in this experiment. Our objective was to isolate and quantify the conservative behavior of the core stability selection algorithm under stress, establishing a baseline before considering advanced extensions.

3. **Noise reduction**: despite the dilution of the signal, the method successfully filtered out the 9 FPs found by the single Lasso fit at $\lambda_{1se}$.

In high-correlation regimes without random re-weighting, stability selection prioritizes **safety over power**. It acts as a conservative filter: it may miss some redundant true signals (FNs), but it strictly controls the inclusion of noise (FPs).

### 3.2 Design B: complementary pairs stability selection and error control

The second simulated design focuses on Complementary Pairs Stability Selection (CPSS) and on the theoretical error bound derived by Meinshausen and Bühlmann (2010). To match as closely as possible the assumptions behind the theory, we consider a high-dimensional linear model with independent predictors: only the first $s = 20$ variables carry signal, while the remaining $p - s$ predictors are pure noise. In this setting we are less interested in the instability induced by correlation, and more in quantifying how conservative the theoretical Per-Family Error Rate (PFER) bound is when applied to CPSS in practice.

#### 3.2.1 Data generation with independent predictors

We generate a high-dimensional linear regression model with $n_2 = 150$ observations and $p_2 = 800$ predictors. Only the first $s_2 = 20$ variables are truly active with coefficient equal to 1, while the remaining predictors are pure noise.
The predictors are independent, and the noise variance is chosen so that the signal-to-noise ratio (SNR) equals 1, which makes the selection problem non-trivial but still interpretable.

```r
set.seed(777)

n2 <- 150
p2 <- 800
s2 <- 20 # number of true signals

beta2 <- c(rep(1, s2), rep(0, p2 - s2))
S_true2 <- which(beta2 != 0) # indices of true signals
```

Independent predictors ($\Sigma = I$):

```r
Sigma2 <- diag(p2)
```

Noise variance for SNR = 1:

```r
signal_var2 <- as.numeric(t(beta2) %*% Sigma2 %*% beta2)
SNR2 <- 1
sigma2_2 <- signal_var2 / SNR2
```

Simulate $X_2$ and $y_2$:

```r
X2 <- matrix(rnorm(n2 * p2), nrow = n2, ncol = p2)
eps2 <- rnorm(n2, mean = 0, sd = sqrt(sigma2_2))
y2 <- as.numeric(X2 %*% beta2 + eps2)

col_vec2 <- rep("lightgrey", p2)
col_vec2[S_true2] <- "red"
```

#### 3.2.2 CPSS algorithm with `stabsel`

We apply complementary pairs stability selection using the `stabsel` function from the **stabs** package, taking the lasso as base selection procedure $\hat{S}_n$.
The tuning parameters are set to $q = 40$ (expected number of selected variables on a half-sample) and cutoff $\tau = 0.6$.

```r
q2 <- 40 # expected number of selected variables per half-sample
tau2 <- 0.6

fit_stab2 <- stabsel(
  x = X2,
  y = y2,
  fitfun = glmnet.lasso,
  q = q2,
  cutoff = tau2,
  assumption = "none"
  )

fit_stab2
```

```
##  Stability Selection without further assumptions
##
## Selected variables:
##  1  4  7 11
##  1  4  7 11
##
## Selection probabilities:
##   21   22   28   34   39   43   51   57   59   65   77   85   86   94   97  102
## 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
##  105  110  113  120  128  129  131  132  133  134  136  138  145  146  147  150
## 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
##  159  164  171  176  177  185  195  196  197  198  201  204  205  206  208  210
## 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
##  215  219  222  227  236  244  245  248  251  253  261  263  285  290  294  296
## 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
##  298  299  302  304  308  309  313  315  317  318  321  322  326  333  338  341
## 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
##  344  359  360  372  373  376  378  384  386  390  397  400  403  405  413  420
## 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
##  426  427  428  431  432  439  445  451  453  457  467  468  469  478  481  483
## 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
##  488  489  492  495  496  497  504  506  510  512  513  514  519  521  523  527
## 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
##  531  537  539  540  541  542  544  550  552  553  554  562  565  570  573  575
## 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
##  580  584  587  589  596  599  606  608  612  614  620  621  625  628  634  635
## 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
##  640  646  650  660  665  667  670  680  681  691  692  694  698  699  706  713
## 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
##  715  718  721  723  725  726  729  731  733  737  739  741  744  747  754  755
## 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
##  761  764  766  769  775  779  780  786  792    2   19   24   25   26   31   38
## 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 0.01
##   46   49   50   52   58   67   70   71   74   78   79   80   83   84   89   91
## 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01
##   92   99  101  106  114  115  117  127  130  135  139  140  141  144  148  158
## 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01
##  166  168  175  181  183  200  203  223  229  238  249  256  258  260  267  272
## 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01
##  276  278  282  283  284  306  311  312  316  324  331  346  357  361  371  375
## 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01
##  382  383  392  399  401  402  406  407  408  409  410  411  415  416  424  434
## 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01
##  447  458  459  464  465  470  474  494  501  509  511  516  518  524  529  530
## 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01
##  534  535  538  543  557  559  563  569  576  577  579  583  588  594  597  603
## 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01
##  604  609  617  618  623  629  632  641  651  653  654  656  661  663  668  671
## 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01
##  672  675  677  684  696  705  711  716  719  730  736  748  749  752  768  772
## 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01 0.01
##  782  794  797   13   23   33   36   37   40   45   54   64   66   75   90   93
## 0.01 0.01 0.01 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02
##   95   98  103  109  122  125  137  149  162  167  179  180  182  188  199  202
## 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02
##  207  211  216  235  237  246  266  270  279  289  292  295  297  301  320  330
## 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02
##  335  337  345  347  369  379  387  395  423  430  438  444  448  454  472  480
## 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02
##  485  490  500  503  505  508  532  546  556  560  566  567  592  598  610  613
## 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02
##  631  633  637  643  664  669  682  683  701  702  708  710  724  732  745  751
## 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02 0.02
##  756  758  796   29   48   62   82  112  153  165  169  174  178  187  193  209
## 0.02 0.02 0.02 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
##  214  231  240  252  262  265  268  288  314  336  342  349  350  353  355  356
## 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 0.03
##  370  422  425  456  460  462  463  491  507  517  525  533  549  578  585  591
## 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 0.03
##  595  600  601  605  607  611  619  648  649  662  666  678  688  697  700  712
## 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 0.03
##  714  742  757  759  781  783  787  793   35   47   60   76   88  107  118  119
## 0.03 0.03 0.03 0.03 0.03 0.03 0.03 0.03 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04
##  161  172  192  233  259  271  275  281  307  332  334  339  340  351  354  362
## 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 0.04
##  388  393  417  419  421  449  461  476  479  482  520  547  548  555  561  582
## 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 0.04
##  622  626  630  636  642  645  652  655  693  695  717  738  740  746  753  785
## 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 0.04
##  788  790  798  800   27   30   41   42   44   56   96  100  108  111  123  126
## 0.04 0.04 0.04 0.04 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05
##  143  152  156  213  224  226  280  293  319  325  348  363  364  365  396  440
## 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05
##  455  471  477  486  493  564  571  574  639  674  687  707  709  735  767  784
## 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05
##   32   55   69   81  121  163  186  218  232  300  380  418  429  433  446  498
## 0.06 0.06 0.06 0.06 0.06 0.06 0.06 0.06 0.06 0.06 0.06 0.06 0.06 0.06 0.06 0.06
##  568  638  690  722  762  776   53  184  189  230  269  274  277  287  329  366
## 0.06 0.06 0.06 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
##  367  377  536  624  673  686  720  734  770  789   12   63  228  343  381  435
## 0.07 0.07 0.07 0.07 0.07 0.07 0.07 0.07 0.07 0.07 0.08 0.08 0.08 0.08 0.08 0.08
##  443  450  499  551  616  774   87  104  225  257  264  291  305  323  368  441
## 0.08 0.08 0.08 0.08 0.08 0.08 0.09 0.09 0.09 0.09 0.09 0.09 0.09 0.09 0.09 0.09
##  502  627  647  791    9   72   73  151  155  241  254  310  487  545  586  676
## 0.09 0.09 0.09 0.09 0.10 0.10 0.10 0.10 0.10 0.10 0.10 0.10 0.10 0.10 0.10 0.10
##  685  773  795    6   61  242  243  247  385  689  743  778  212  286  327  412
## 0.10 0.10 0.10 0.11 0.11 0.11 0.11 0.11 0.11 0.11 0.11 0.11 0.12 0.12 0.12 0.12
##  452  466  473  615  170  273  352  398  404  528  658  765   14  303  436  581
## 0.12 0.12 0.12 0.12 0.13 0.13 0.13 0.14 0.14 0.14 0.14 0.14 0.15 0.15 0.15 0.15
##  657   17  160  358  389  437  558  704   18  414  602  659  679  217  234  442
## 0.15 0.16 0.16 0.16 0.16 0.16 0.16 0.16 0.17 0.17 0.17 0.17 0.17 0.18 0.18 0.18
##  475  777  727  116  190  522  526  763    5   68  374  142  239  484  572  593
## 0.18 0.18 0.19 0.20 0.20 0.20 0.20 0.20 0.21 0.21 0.21 0.22 0.22 0.22 0.22 0.22
##  799  220  750  250  515  644  728   20  760  771  157  154  194  124  191  255
## 0.22 0.23 0.23 0.24 0.24 0.24 0.24 0.25 0.25 0.25 0.27 0.29 0.29 0.31 0.31 0.32
##  173   10  394   16  328  391  221  703  590    3    8   15    1    4    7   11
## 0.34 0.35 0.35 0.39 0.39 0.39 0.41 0.42 0.45 0.47 0.49 0.51 0.60 0.66 0.79 0.88
##
## ---
## Cutoff: 0.6; q: 40; PFER (*):  10
##    (*) or expected number of low selection probability variables
## PFER (specified upper bound):  10
## PFER corresponds to signif. level 0.0125 (without multiplicity adjustment)
```

The `stabsel` output reports four CPSS-selected variables:

$$\hat{S}_{\text{stab}} = \{1, 4, 7, 11\},$$

with the following estimated selection probabilities:

$$\hat{\pi}_1 = 0.60, \quad \hat{\pi}_4 = 0.66, \quad \hat{\pi}_7 = 0.79, \quad \hat{\pi}_{11} = 0.88.$$

Since in Design B the true active set is

$$S_{\text{true}} = \{1, 2, \ldots, 20\},$$

all four selected variables are genuine signals. The empirical performance is therefore:

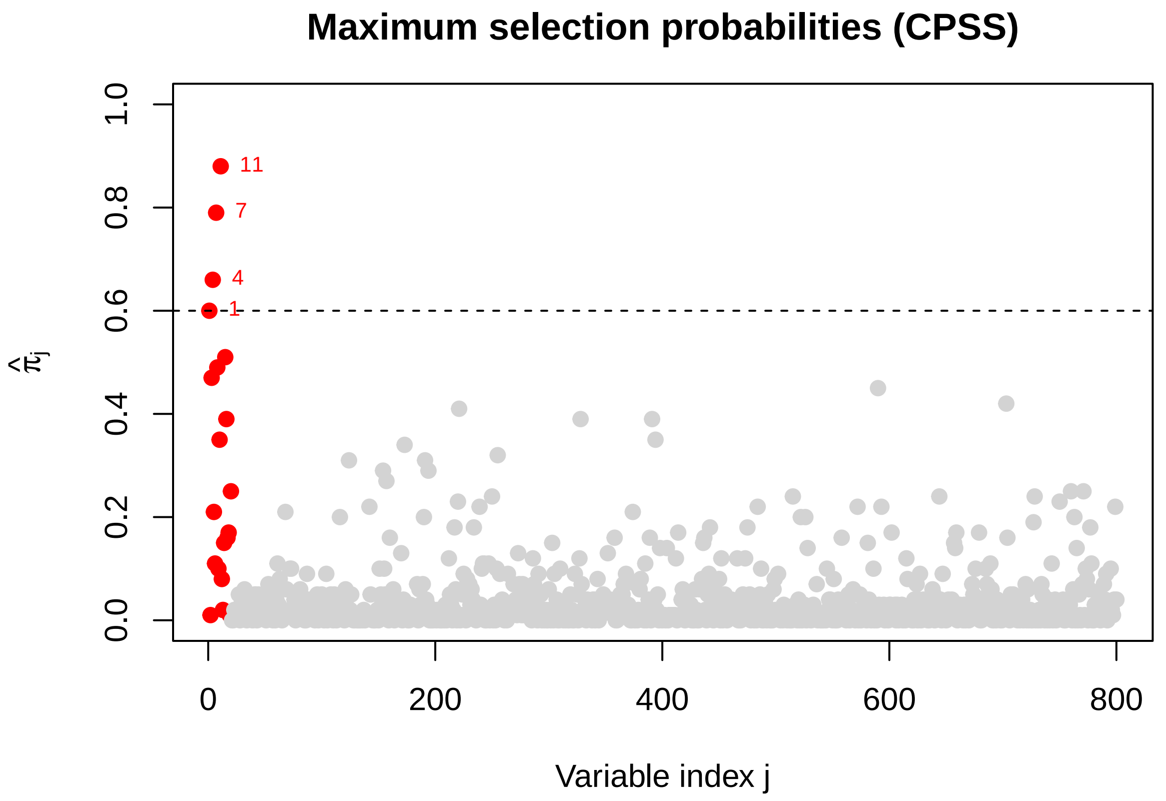
$$TP_{\text{emp}} = 4, \qquad FP_{\text{emp}} = 0.$$

### 3.2.3 Maximum estimated selection probability for each variable

In this section we examine the behaviour of CPSS by analysing the *maximum* estimated selection probability $\hat{\pi}_j$ for each predictor.
Instead of tracking full stability paths across $\lambda$, CPSS aggregates information from complementary half–samples and assigns to each variable a single stability score summarising how often it is selected across all resampled fits.
Plotting these maximum selection probabilities provides an immediate visual separation between truly active predictors and noise variables, and allows us to assess how conservative CPSS is under the chosen tuning parameters $(q, \tau)$. This diagnostic step is crucial for understanding the global stability pattern of the model and for verifying whether CPSS successfully isolates the informative variables while suppressing spurious selections.

```
pi_hat_max <- fit_stab2$max

col_vec2 <- rep("lightgrey", p2) # noise
```

```r
col_vec2[S_true2] <- "red" # true signals

par(mar = c(5, 5, 3, 2))

plot(
  x = 1:p2,
  y = pi_hat_max,
  col = col_vec2,
  pch = 19,
  xlab = "Variable index j",
  ylab = expression(hat(pi)[j]),
  main = "Maximum selection probabilities (CPSS)",
  ylim = c(0, 1)
  )

abline(h = tau2, lty = 2)

true_above <- S_true2[ pi_hat_max[S_true2] >= tau2 ]

text(
  x = true_above,
  y = pi_hat_max[true_above],
  labels = true_above,
  pos = 4,
  cex = 0.7,
  col = "red"
  )
```



The figure above displays the *maximum selection probabilities* $\hat{\pi}_j$. Each point corresponds to a variable: truly active predictors (indices 1–20) are highlighted in red, whereas noise variables are shown in grey. The horizontal dashed line marks the stability cutoff $\tau = 0.6$.

The plot shows a clear separation between signal and noise. Four active predictors (variables 1, 4, 7 and 11) lie well above the threshold, with selection probabilities between 0.6 and 0.88, and are therefore retained by CPSS as stable variables. The remaining truly active predictors have intermediate selection probabilities, typically in the range 0.15–0.55, and some of them fall just below the cutoff: they are consistently more stable than most noise variables, but not sufficiently so to be included in the final CPSS set under $\tau = 0.6$.

In contrast, the vast majority of noise predictors have very low selection probabilities, concentrated in the range of 0–0.2, with only a small fraction reaching values around 0.2–0.4 and none approaching the stability cutoff. As a result, no pure-noise variable crosses the threshold. Overall, the figure illustrates how CPSS strongly suppresses false positives and prioritises the most stable true signals, at the cost of discarding moderately stable active predictors when a relatively stringent cutoff is used.

### 3.2.4 Empirical false positives vs Meinshausen & Bühlman bound

In this final part of the analysis we compare the empirical behaviour of CPSS with the theoretical guarantees provided by Meinshausen and Bühlmann (2010).
Specifically, we quantify the number of true and false positives produced by the CPSS-selected set $\hat{S}_{\text{stab}}$ and assess whether this empirical error aligns with the upper bound on the expected number of false selections, namely

$$\mathbb{E}\Big(|\hat{S}_{\text{stab}} \cap N|\Big) \;\le\; \frac{1}{2\tau - 1}\,\frac{q^2}{p}.$$

This bound (the empirical Per-Family Error Rate, PFER) provides an explicit, a priori control on false discoveries based solely on the tuning parameters $(q, \tau)$ and the dimensionality $p$.

By contrasting the empirical false positives with the theoretical PFER, we can evaluate whether CPSS behaves conservatively in practice and whether the theoretical control mechanism is informative for guiding parameter choices in high-dimensional applications.

Stability-selected set from CPSS:

```
S_stab2 <- fit_stab2$selected
S_stab2
```

```
##  1  4  7 11
##  1  4  7 11
```

True and noise indices:

```
N2 <- setdiff(1:p2, S_true2)
```

Empirical number of false positives and true positives:

```
FP_emp <- length(intersect(S_stab2, N2))
TP_emp <- length(intersect(S_stab2, S_true2))

FP_emp
```

```
## [1] 0
```

```
TP_emp
```

```
## [1] 4
```

Theoretical upper bound on the expected number of false positives:

```
bound_FP <- (1 / (2 * tau2 - 1)) * (q2^2 / p2)
bound_FP
```

```
## [1] 10
```

In our CPSS experiment we consider $p_2 = 800$ predictors, with tuning parameters $q = 40$ and $\tau = 0.6$. The theoretical upper bound on the expected number of false positives is therefore

$$\frac{1}{2 \cdot 0.6 - 1} \cdot \frac{40^2}{800} = \frac{1}{0.2} \cdot \frac{1600}{800} = 5 \cdot 2 = 10.$$

The stability-selected set returned by `stabsel` is

$$\hat{S}_{\text{stab}} = \{1, 4, 7, 11\},$$

and all four selected variables belong to the true active set $S_{\text{true}}$.

Consequently, the empirical numbers of true and false positives are

$$TP_{\text{emp}} = 4, \qquad FP_{\text{emp}} = 0.$$

Thus, in this realisation CPSS selects four true signals and no noise variables.
The empirical number of false positives equals zero, which is far below the theoretical upper bound of 10. This confirms the conservative nature of the Meinshausen–Bühlmann error bound and shows that, under these tuning parameters, CPSS achieves perfect false-positive control while still recovering the informative predictors.

### 3.2.5 Discussion of Design B results

In this independent setting, stability selection works flawlessly, serving as a strict empirical validation of the theory:

1. **Perfect separation**: as shown in the plot, the true signals (red) clearly separate from the noise (grey). Unlike Design A, there is no “signal dilution” because the predictors are orthogonal ($\rho = 0$). This confirms that when the data structure aligns with the model assumptions, CPSS is highly sensitive.
2. **Strict error control**: the algorithm selected 0 False Positives. This is fully consistent with the theoretical PFER bound, which guaranteed $E(V) \leq 10$. The fact that the empirical error (0) is strictly below the bound demonstrates the conservative “safety margin” provided by the Meinshausen-Bühlmann framework.

In conclusion, this experiment acts as a “control group”. It proves that stability selection is not just a heuristic for messy data, but a rigorous statistical procedure. When the exchangeability assumption holds, it recovers the exact structure of the model without admitting false discoveries.

## 3.3 Conclusions: simulation study

The simulation study confirms the dual nature of stability selection, revealing its adaptability to different data structures. In the “ideal” independent setting of Design B, the method achieved perfect structure recovery, strictly adhering to the theoretical PFER bound and demonstrating that, when the exchangeability assumption holds, stability selection is both safe and sensitive. Conversely, in the high-correlation “stress test” of Design A, the method shifted towards a conservative stance: while signal dilution prevented the full recovery of all true predictors, the procedure effectively suppressed the noise that plagued the standard Lasso fit. This contrast highlights that stability selection is not merely a variable selection technique, but a robust statistical framework that prioritizes error control over raw discovery power, providing a crucial safeguard against false positives even in challenging, multicollinear environments.

## 4 Real-data application: stability selection on kidney transplant gene expression

In the final part of this report we apply lasso regression and stability selection to a genuine high-dimensional biomedical dataset: the `Einecke2010Kidney` gene expression data from kidney transplant recipients. The aim is to investigate whether gene expression profiles can help discriminate between biopsies with and without rejection, and to assess how stability selection behaves in a real, noisy omics scenario where the true set of relevant predictors is unknown.

### 4.1 Data description and pre-processing

The dataset contains 250 patients (rows) and 10,001 variables (columns): a binary outcome `Reject_Status` indicating whether the graft was rejected (1) or not (0), and the expression levels of 10000 genes, selected among the 25% most variable genes in the original study by Einecke et al. (2010).

```r
load("Einecke2010Kidney.rda")

str(Einecke2010Kidney)
```

```
## 'data.frame':    250 obs. of  10001 variables:
##  $ Reject_Status              : num  0 0 0 0 0 0 1 0 1 0 ...
##  $ X211352_s_at               : num  2.369 -1.488 0.556 0.127 -0.38 ...
##  $ X239275_at                 : num  1.055 -0.606 0.838 1.481 0.397 ...
##  $ X1554536_at                : num  1.2491 -2.0427 -0.7287 0.1914 -0.0522 ...
##  $ X1558452_at                : num  0.842 0.947 0.354 -1.156 1.273 ...
##  $ X1554249_a_at              : num  0.991 -0.967 1.544 0.601 1.417 ...
##  $ X234382_x_at               : num  0.462 1.317 -0.472 -0.296 1.293 ...
##  $ X1557126_a_at              : num  -0.637 0.651 -0.343 1.997 0.68 ...
##  $ X1554865_at                : num  1.45 -3.063 -0.837 1.579 -0.361 ...
##  $ X224489_at                 : num  -0.0863 -0.8983 -1.3072 1.3568 2.3829 ...
##  $ X237956_s_at               : num  -0.575 -0.941 -1.74 -0.401 -0.985 ...
##  $ X206300_s_at               : num  -0.783 -0.839 -0.477 -0.969 0.903 ...
##  $ X243856_at                 : num  -0.4618 -1.4183 -0.0251 -0.4214 0.6196 ...
##  $ X244331_at                 : num  1.051 0.181 1.574 0.471 -1.156 ...
##  $ X227213_at                 : num  2.199 0.441 1.007 0.134 1.041 ...
##  $ X233226_at                 : num  0.802 -0.757 0.203 0.554 -1.554 ...
##  $ X220095_at                 : num  1.579 0.823 -1.715 0.795 0.912 ...
##  $ X226474_at                 : num  0.832 -1.121 -0.384 0.576 1.178 ...
##  $ X242787_at                 : num  1.1296 -1.8732 -0.6058 0.8942 0.0281 ...
##  $ X244260_at                 : num  0.7277 -0.9833 -0.0775 1.0537 -0.0203 ...
##  $ X202887_s_at               : num  -0.374 -1.718 -0.96 -1.477 -0.645 ...
##  $ X226245_at                 : num  -0.666 -0.265 0.381 -1.449 -1.008 ...
##  $ X1556942_at                : num  0.3385 -0.9133 -0.0664 0.8342 -1.1205 ...
##  $ X1555316_a_at              : num  0.5642 0.1896 -1.3789 -0.0788 -1.4648 ...
##  $ X1557921_s_at              : num  -0.0411 1.274 0.0383 -1.3712 0.118 ...
##  $ X1569600_at                : num  0.726 -0.409 0.08 -1.189 0.303 ...
##  $ X1556770_a_at              : num  1.5115 0.375 -0.2152 0.0684 0.9383 ...
##  $ X240450_at                 : num  -0.0104 -0.3621 0.8258 0.3852 -1.0338 ...
##  $ X244356_at                 : num  0.2243 -0.9108 -0.0778 -0.2529 0.1771 ...
##  $ X1561486_at                : num  0.0218 -1.1006 -1.0221 -1.5788 1.4811 ...
##  $ X238169_at                 : num  1.1518 -0.4704 -1.8894 0.6977 0.0208 ...
##  $ X243565_at                 : num  0.29 0.603 1.247 1.566 -0.315 ...
##  $ X231626_at               : num  -1.3244 1.3253 -0.3939 1.2571 0.0962 ...
##  $ X1559310_at              : num  -0.864 -1.507 -0.289 -0.33 2.023 ...
##  $ X225590_at               : num  0.438 0.617 -0.378 0.691 -2.579 ...
##  $ X232704_s_at             : num  -0.4676 0.284 -0.0229 -0.6871 -0.083 ...
##  $ X1569706_at              : num  -0.422 0.555 0.763 -1.017 -0.232 ...
##  $ X1559624_at              : num  0.896 -0.905 0.739 -1.072 1.196 ...
##  $ X211538_s_at             : num  0.479 -0.312 -0.592 -0.395 1.188 ...
##  $ X209470_s_at             : num  -0.0421 0.2646 -0.7182 0.6066 1.5222 ...
##  $ X235382_at               : num  -0.744 0.803 -0.677 0.881 -1.092 ...
##  $ X229266_at               : num  -0.0944 -0.5465 -1.3495 -0.5961 3.493 ...
##  $ X213058_at               : num  1.0995 0.8589 1.3333 -0.5422 0.0676 ...
##  $ X1554676_at              : num  0.474 -1.421 -0.5 -0.132 0.53 ...
##  $ X1556451_at              : num  0.3986 0.0596 0.2087 -0.7991 0.668 ...
##  $ X217534_at               : num  -0.855 0.676 1.18 -0.935 -0.675 ...
##  $ X1553216_at              : num  1.917 -0.611 0.352 0.423 1.433 ...
##  $ X210279_at               : num  1.467 -1.211 -0.342 0.387 1.031 ...
##  $ X240821_at               : num  0.2899 0.0544 2.4642 -0.1297 1.9734 ...
##  $ X1569812_at              : num  -0.073 -0.276 0.329 -1.964 0.934 ...
##  $ X208011_at               : num  -0.6799 -1.6216 -0.8767 1.5949 -0.0553 ...
##  $ X234164_at               : num  0.0446 -0.6934 2.4177 -1.9598 0.0551 ...
##  $ X1560225_at              : num  -0.0232 -0.9918 -0.9085 0.586 -1.7583 ...
##  $ X236513_at               : num  0.0745 -0.3045 -1.1733 -1.2722 0.4857 ...
##  $ X1560066_at              : num  -1.192 0.863 0.233 1.972 0.579 ...
##  $ X243013_at               : num  0.639 -0.537 -1.173 1.026 0.847 ...
##  $ X207912_s_at             : num  -0.244 0.581 0.548 0.456 -0.235 ...
##  $ X228218_at               : num  -0.000209 -0.239307 -0.788782 -0.066802 1.643366 ...
##  $ X202464_s_at             : num  0.903 -0.922 -1.363 -0.473 1.543 ...
##  $ X216012_at               : num  0.841 0.684 0.468 1.807 -0.379 ...
##  $ X1554213_at              : num  -0.894 -0.23 -0.391 0.35 0.663 ...
##  $ X213503_x_at             : num  0.708 -1.551 -0.392 -1.072 0.43 ...
##  $ X218723_s_at             : num  -0.278 -0.677 -0.322 -0.102 0.618 ...
##  $ X213268_at               : num  0.425 -1.274 -0.486 0.757 0.37 ...
##  $ X244548_at               : num  0.991 -0.799 -1.05 -0.23 -0.757 ...
##  $ X1556345_s_at            : num  1.622 0.41 0.536 0.723 0.329 ...
##  $ X216557_x_at             : num  1.421 -0.689 -0.12 -1.066 -0.576 ...
##  $ X1557705_a_at            : num  0.713 0.508 0.815 -0.678 -0.838 ...
##  $ X224200_s_at             : num  0.4347 -1.0572 1.2535 0.0252 1.1058 ...
##  $ X217974_at               : num  0.148 1.95 -2.292 0.511 -1.048 ...
##  $ X230150_at               : num  -0.6573 -0.7265 -1.3111 0.8029 0.0564 ...
##  $ X233807_at               : num  0.46069 0.22188 -2.69903 -0.00228 0.3906 ...
##  $ X210972_x_at             : num  1.468 -1.282 -0.579 0.831 0.877 ...
##  $ X237400_at               : num  -0.313 1.518 -0.365 1.691 0.319 ...
##  $ X220401_at               : num  -0.2452 0.6021 0.8991 -0.8048 -0.0226 ...
##  $ X227868_at               : num  -0.2406 0.0298 -0.4659 -0.1746 1.0155 ...
##  $ X225660_at               : num  0.309 1.386 2.339 -0.479 -1.273 ...
##  $ X238018_at               : num  -0.95 -2 -1.466 -0.886 -0.123 ...
##  $ X226517_at               : num  0.208 -0.377 -0.267 -0.36 0.578 ...
##  $ X214877_at               : num  1.14 1.11 -2.65 1 -0.14 ...
##  $ X240266_at               : num  0.579 -0.605 -1.876 -0.408 0.485 ...
##  $ X1555292_at              : num  0.626 -0.739 0.122 2.923 -0.724 ...
##  $ X1552522_at              : num  0.317 -0.62 1.726 -0.594 0.541 ...
##  $ X240431_at               : num  -1.31 -0.226 -0.741 0.514 -1.621 ...
##  $ X1560207_at              : num  0.781 0.894 0.887 -2.332 -0.209 ...
##  $ X226364_at               : num  0.9001 -0.0916 -0.7608 -0.3236 0.8636 ...
##  $ X224772_at                 : num  0.966 -1.499 -0.534 0.516 1.342 ...
##  $ X202291_s_at               : num  1.607 -1.244 -1.173 -0.828 -0.895 ...
##  $ X243563_at                 : num  1.2956 -0.3337 -1.2289 -0.0563 1.9485 ...
##  $ X228504_at                 : num  0.286 -0.232 0.542 1.09 0.207 ...
##  $ X232398_at                 : num  -0.431 1.122 1.255 1.067 0.761 ...
##  $ X235801_at                 : num  1.744 -0.969 0.46 -1.799 0.811 ...
##  $ X200696_s_at               : num  1.264 -0.688 -1.314 -0.722 -1.409 ...
##  $ X244510_at                 : num  -0.156 -1.742 -2.433 -0.136 2.122 ...
##  $ X236095_at                 : num  -1.833 -0.0872 -0.769 1.8958 1.0325 ...
##  $ X38241_at                  : num  0.784 -0.889 -0.268 0.781 1.3 ...
##  $ X240990_at                 : num  3.308 -0.982 -0.199 0.71 0.282 ...
##  $ X216959_x_at               : num  0.781 -1.389 2.164 1.613 0.431 ...
##  $ AFFX.r2.Ec.bioC.3_at       : num  0.276 0.22 0.93 0.355 0.292 ...
##   [list output truncated]
```

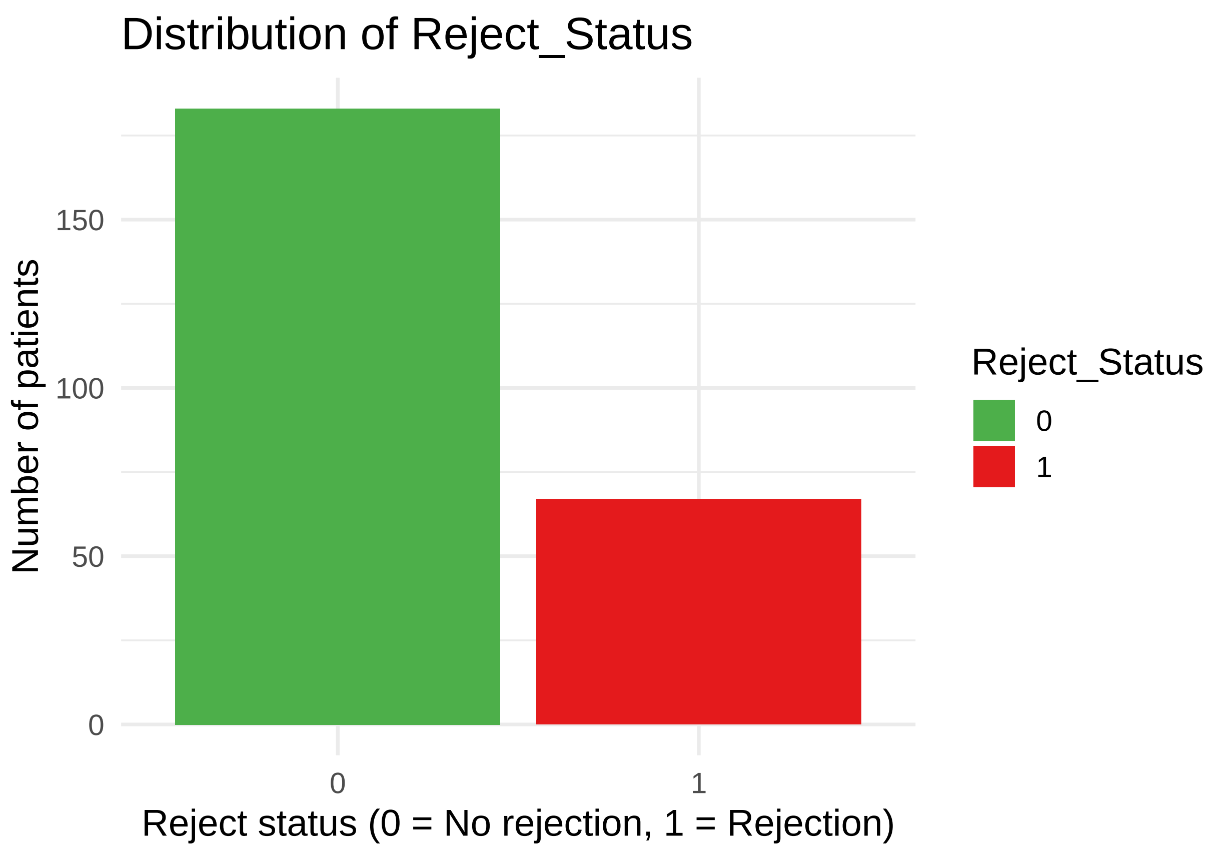
```r
dim(Einecke2010Kidney)
```

```
## [1]   250 10001
```

```r
table(Einecke2010Kidney$Reject_Status)
```

```
##
##   0   1
## 183  67
```

```r
library(ggplot2)
df_counts <- as.data.frame(table(Einecke2010Kidney$Reject_Status))
colnames(df_counts) <- c("Reject_Status", "Count")

ggplot(df_counts, aes(x = Reject_Status, y = Count, fill = Reject_Status)) +
  geom_bar(stat = "identity") +
  scale_fill_manual(values = c("#4DAF4A", "#E41A1C")) +
  labs(
    title = "Distribution of Reject_Status",
    x = "Reject status (0 = No rejection, 1 = Rejection)",
    y = "Number of patients"
  ) +
  theme_minimal(base_size = 14)
```



We treat `Reject_Status` as the binary outcome and all 10000 gene expression variables as predictors. The response variable is imbalanced (183 non-rejection vs 67 rejection), which increases model instability in half-sampling procedures, especially for logistic lasso.

```r
y <- Einecke2010Kidney$Reject_Status # outcome (0/1)

X_raw <- Einecke2010Kidney[, -1] # predictors
dim(X_raw)
```

```
## [1]   250 10000
```

### 4.1.1 Missing values and near-zero variance genes

First, we check for missing values and near-zero variance predictors. We remove genes that are practically constant to avoid numerical issues in Lasso.

```r
sum(is.na(y))
```

```
## [1] 0
```

```r
sum(is.na(X_raw))
```

```
## [1] 0
```

```r
# Remove near-zero variance predictors
library(caret)
```

```
## Loading required package: lattice
```

```r
nzv_idx <- nearZeroVar(X_raw)

if(length(nzv_idx) > 0) {
  X_nzv <- X_raw[, -nzv_idx]
  } else {
    X_nzv <- X_raw
    }

dim(X_nzv)
```

```
## [1]   250 10000
```

No missing values are present in either the outcome variable or in the gene expression matrix. Moreover, the `nearZeroVar` diagnostic did not identify any gene with near-zero variance. This confirms that all 10000 gene expression features exhibit sufficient variability across the 250 patients and can therefore be retained for downstream modelling.

From a biological standpoint, this behaviour is consistent with standard microarray preprocessing pipelines, in which low-intensity or low-variance probes are typically filtered out during the initial quality-control stage. As a consequence, the dataset appears clean, well-behaved, and directly suitable for high-dimensional regression and stability selection, without requiring additional feature removal or imputation at this stage.

### 4.1.2 Standardization and data leakage prevention

Standardization (centering variables to mean zero and scaling to unit variance) is theoretically required for Lasso regression because the $\ell_1$ penalty term depends on the magnitude of the coefficients. If variables are not on the same scale, those with larger variances would be penalized disproportionately. A standard pre-processing pipeline would typically involve the following step:

```r
# THEORETICAL IMPLEMENTATION (not executed to avoid data leakage)

# In a standard setting, we would center and scale the full matrix X:
#library(caret)
#pp <- preProcess(X_raw, method = c("center", "scale"))
#X_scaled_global <- predict(pp, X_raw)

# However, applying this globally BEFORE subsampling constitutes data leakage
```

We deliberately do not apply this global transformation here. Using the global mean and standard deviation to scale observations that will later become "out-of-bag" samples in the stability selection loop constitutes data leakage. It introduces a dependency between the training and validation sets within the resampling procedure, potentially leading to optimistic bias.

Instead, we pass the unscaled data to `glmnet` and rely on its internal argument `standardize = TRUE` (default). This ensures that standardization parameters are calculated strictly within each subsample, preserving the independence of the selection process and ensuring rigorous error control.

Also, in this application we deliberately avoid further unsupervised transformations such as Box-Cox skewness correction, correlation-based filtering, or PCA feature extraction. First, the microarray expression values have already undergone a standard normalisation pipeline (including background correction and log-transformation), so their marginal distributions are reasonably well behaved for penalised regression. Second, our primary objective is gene-level variable selection and biological interpretability, rather than constructing abstract latent components. Introducing PCA or other feature-extraction steps would compress information into linear combinations of genes, making it harder to trace back which specific transcripts drive the prediction of rejection. Finally, redundancy and correlation among genes are handled downstream by the $\ell_1$ penalty and by stability selection, which are explicitly designed to operate in high-dimensional, highly correlated settings without requiring an additional, aggressive pre-filtering of the predictors.

Therefore, for the subsequent analysis, we define `X_mod` simply as the cleaned raw data:

```r
X_mod <- as.matrix(X_nzv)
y_mod <- y
```

### 4.1.3 Lasso logistic regression baseline

As a baseline, we fit a lasso-penalised logistic regression to predict rejection status from all preprocessed genes. We use 10-fold cross-validation to choose the regularisation parameter $\lambda$.

```r
set.seed(123)

cvfit_real <- cv.glmnet(
  X_mod, y_mod,
  family = "binomial",
  alpha = 1, # lasso
  standardize = TRUE,
  nlambda = 300
  )

lambda_min_real <- cvfit_real$lambda.min
lambda_1se_real <- cvfit_real$lambda.1se

lambda_min_real
```

```
## [1] 0.02837507
```

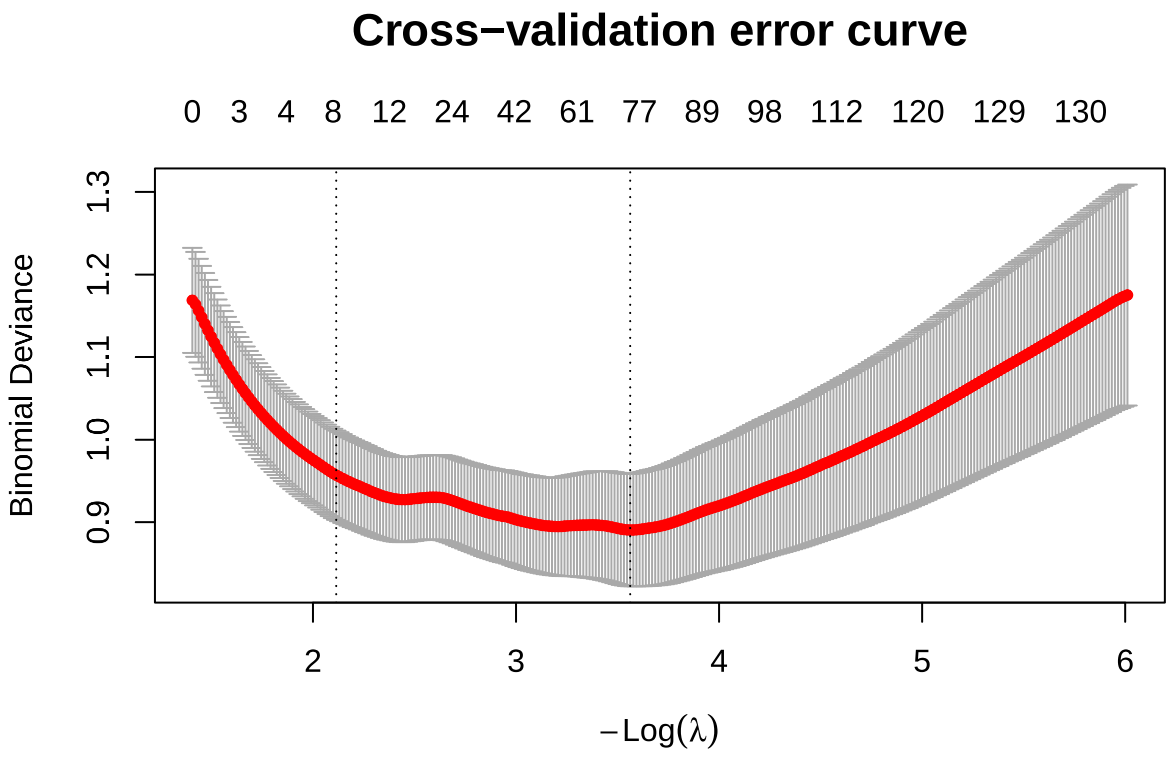
```r
lambda_1se_real
```

```
## [1] 0.1206978
```

For this dataset, cross-validation yields

$$\lambda_{\min} = 0.0284, \qquad \lambda_{1\text{se}} = 0.1207.$$

```r
op <- par(mar = c(5, 4, 6, 2) + 0.1)

plot(cvfit_real, main = "")

mtext(
  "Cross-validation error curve",
  side = 3,
  line = 3,
  cex = 1.4,
  font = 2
)
```



```
par(op)
```

The value $\lambda_{\text{min}}$ corresponds to a less penalised and therefore more complex model, whereas $\lambda_{1\text{se}}$ enforces stronger shrinkage and produces a substantially sparser set of selected genes. In high-dimensional applications such as microarray analysis, the $1se$ rule is often preferred as a baseline because it improves interpretability and reduces the risk of selecting genes that are merely correlated surrogates rather than true biological signals. However, both values of $\lambda$ will serve as useful reference points when comparing the baseline lasso to the subsampling-based stability selection procedure applied in the next section.

```
fit_real <- glmnet(
  X_mod, y_mod,
  family = "binomial",
  alpha = 1,
  lambda = cvfit_real$lambda,
  standardize = TRUE
  )

# Genes selected at lambda_1se
```

```
S_hat_lasso_real <- which(coef(fit_real, s = lambda_1se_real)[-1] != 0)
length(S_hat_lasso_real)
```

```
## [1] 8
```

```
# Gene names of the baseline lasso signature
genes_lasso <- colnames(X_mod)[S_hat_lasso_real]
genes_lasso
```

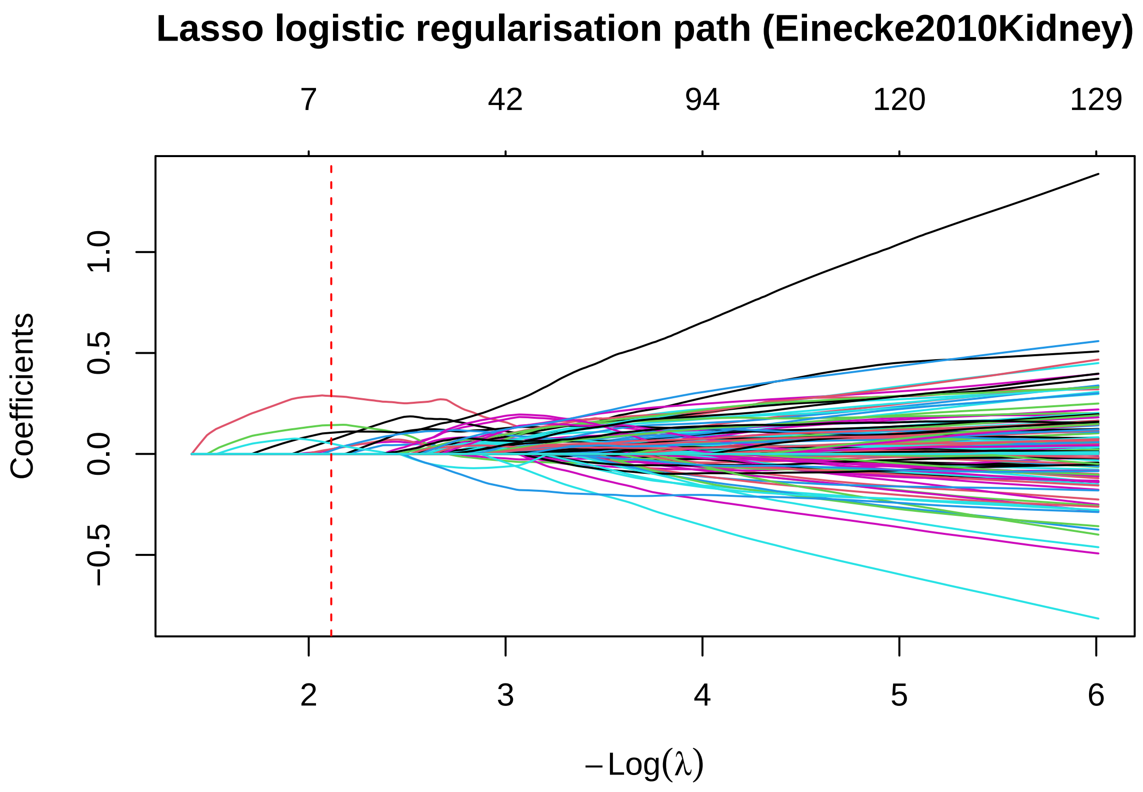
```
## [1] "X226474_at"   "X205758_at"   "X229367_s_at" "X229437_at"   "X244598_at"
## [6] "X238531_x_at" "X220351_at"   "X206914_at"
```

At the more conservative value $\lambda_{1\mathrm{se}}$, the lasso-penalised logistic regression selects only a very small subset of the 10000 candidate genes. In our fit, exactly eight genes have non-zero coefficients at $\lambda_{1\mathrm{se}}$: `X226474_at`, `X205758_at`, `X229367_s_at`, `X229437_at`, `X244598_at`, `X238531_x_at`, `X220351_at` and `X206914_at`. This corresponds to an extremely sparse model, where less than 0.1% of the available predictors are retained.

From a practical viewpoint, such a sparse solution is appealing: it yields a concise "gene signature" that is easy to interpret and potentially testable in follow-up biological studies. At the same time, the absence of ground truth in a real microarray dataset means that we cannot directly quantify false positives or false negatives. The eight genes selected by the baseline lasso at $\lambda_{1\mathrm{se}}$ should therefore be regarded as a first, data-driven screening of potentially relevant transcripts. In the next section we will use subsampling-based stability selection to assess how robust these choices are with respect to perturbations of the sample, and to identify which genes remain consistently selected across resampled datasets.

A visual summary of the regularisation path is given by:

```
par(mar = c(5, 5, 5, 2))

plot(
  fit_real,
  xvar = "lambda",
  main = ""
  )

title(
  main = "Lasso logistic regularisation path (Einecke2010Kidney)",
  line = 3
  )

abline(v = -log(lambda_1se_real), lty = 2, col = "red")
```



Each coloured curve represents the coefficient trajectory of one gene as the penalty parameter $\lambda$ decreases (i.e. as $-\log\lambda$ increases). For large values of $\lambda$, all coefficients are shrunk to zero, while progressively smaller penalisation allows more genes to enter the model.

Even in this high-dimensional setting, only a limited number of genes display substantial coefficient growth, while the vast majority remain close to zero across the entire path. The vertical dashed red line marks the value $\lambda_{1\mathrm{SE}}$, which corresponds to a sparse yet cross-validated choice of regularisation. At this point in the path, the lasso selects a small panel of genes (eight in our analysis), forming a preliminary molecular signature for graft rejection.

It is important to emphasise that, unlike the simulated experiments in earlier sections, the true set of biologically active genes is unknown in this real microarray dataset. Therefore we cannot assess false positives or false negatives directly. Moreover, a single lasso fit may be sensitive to minor perturbations of the data: correlated genes can enter or leave the model depending on which patients are included in the training sample. These considerations motivate the next stage of the analysis, where subsampling-based stability selection is used to quantify how consistently each gene is selected across many reweighted versions of the dataset, providing a more robust picture of variable importance.

### 4.1.4 Robust analysis: Complementary Pairs Stability Selection (CPSS)

To obtain a strictly controlled gene signature, we apply **Complementary Pairs Stability Selection (CPSS)** via the `stabs` package. Unlike the baseline Lasso, which aims to minimise prediction error, this procedure explicitly controls the False Discovery Rate.

```
set.seed(123)
```

```r
p_real <- ncol(X_mod)
q_real <- 80
tau_real <- 0.6

fit_stab_real <- stabsel(
  x = X_mod, y = y_mod,
  fitfun = glmnet.lasso,
  args.fitfun = list(
      family = "binomial",
      standardize = TRUE
  ),
  q = q_real,
  cutoff = tau_real,
  assumption = "none"
)
```

We configure the parameters to ensure a conservative selection in this high-dimensional setting ($p = 10000$):

- $q = 80$: we set the expected number of selected variables per subsample to 80. This acts as a generous upper bound on the model complexity, ensuring we explore enough of the regularization path without fitting pure noise.
- $\tau = 0.6$: we enforce a strict stability threshold. A gene must be selected in at least 60% of the subsamples to be considered stable.
- `assumption = "none"`: we deliberately avoid restrictive assumptions (like unimodality) and rely on the robust, worst-case **Meinshausen-Bühlmann bound** for the PFER. This prioritises safety over power.

### 4.1.5 Results and PFER control

```r
S_stab <- fit_stab_real$selected
pi_max_real <- fit_stab_real$max

cat("CPSS strictly selected genes (tau=0.6):\n")
```

```
## CPSS strictly selected genes (tau=0.6):
```

```r
print(names(S_stab))
```

```
## [1] "X238531_x_at"
```

```r
PFER_val <- (q_real^2) / ((2 * tau_real - 1) * p_real)
cat("Theoretical PFER upper bound:", PFER_val, "\n")
```

```
## Theoretical PFER upper bound: 3.2
```

With a stability threshold of $\tau = 0.6$, the CPSS procedure identifies a single stable predictor: `X238531_x_at`.

To assess the reliability of this selection, we compute the theoretical upper bound on the PFER, defined as the expected number of false positives $E(V)$. According to Meinshausen and Bühlmann, the bound is given by:

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

Substituting our parameters ($p = 10000$, $q = 80$, $\tau = 0.6$):

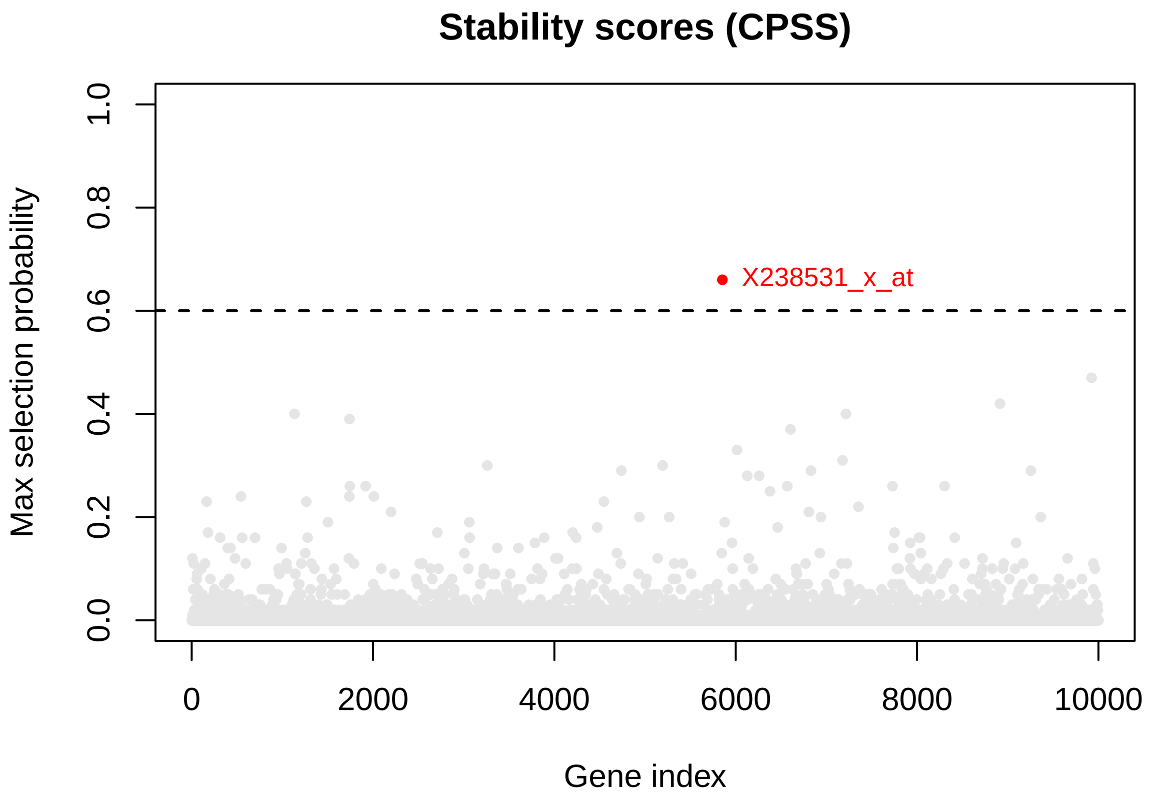
$$\mathbb{E}(V) \leq \frac{80^2}{(2 \cdot 0.6 - 1) \cdot 10000} = \frac{6400}{0.2 \cdot 10000} = \frac{6400}{2000} = 3.2$$

This result implies that, even in a worst-case scenario where the signal is purely random noise, we would expect strictly fewer than 3.2 false positives. Since our procedure selected only 1 variable (which is $< 3.2$), the selection is statistically robust and the probability that this gene is a false discovery is minimized.

#### 4.1.6 Visualising the stability landscape (Manhattan plot)

To provide a global perspective on variable importance, we visualize the maximum stability scores $\hat{\pi}_j^{\max}$ for all $p = 10000$ genes using a Manhattan-style plot.

```r
par(mar = c(5, 5, 3, 2))
plot(1:p_real, pi_max_real,
     pch = 19, col = ifelse(pi_max_real >= tau_real, "red", "grey90"),
     cex = 0.6,
     xlab = "Gene index", ylab = "Max selection probability",
     main = "Stability scores (CPSS)", ylim = c(0, 1))

abline(h = tau_real, lty = 2, col = "black", lwd = 1.5)

sel_idx <- which(pi_max_real >= tau_real)
text(sel_idx, pi_max_real[sel_idx], labels = names(S_stab), pos = 4, cex = 0.8, col = "red")
```



The Manhattan plot offers a striking visual confirmation of the method's selectivity. Unlike the chaotic regularization path of the standard Lasso, which suggested a complex model, the stability landscape reveals that the vast majority of the transcriptome (grey points) remains indistinguishable from noise, with stability scores effectively suppressed near zero. Only a single feature, `X238531_x_at` (highlighted in red), clearly separates from the background distribution, crossing the strict threshold $\tau = 0.6$. This visualises the essence of stability selection: it acts as a high-pass filter for reliability, suppressing the "noise floor" typical of high-dimensional omics data and retaining only the signal that is robust to data perturbation.

### 4.1.7 Sensitivity analysis: relaxing the error control

Ideally, identifying a panel of multiple biomarkers would be preferable to a single gene. To rule out the possibility that our strict error control ($\tau = 0.6$) was masking weaker but potentially biologically relevant signals, we performed a sensitivity analysis by relaxing the stability threshold. Specifically, we maintained the search depth constant ($q = 80$) but lowered the stability cutoff to $\tau = 0.51$. This setting implicitly accepts a massive increase in the theoretical risk of false discoveries.

```r
stab_relaxed <- stabs::stabsel(
  x = X_mod,
  y = y_mod,
  fitfun = glmnet.lasso,
  args.fitfun = list(family = "binomial", standardize=TRUE),
  q = 80,
  cutoff = 0.51,
  assumption = "none",
  B = 100
)

cat("Selected variables with q=80 and cutoff=0.51:\n")
```

```
## Selected variables with q=80 and cutoff=0.51:
```

```
print(names(stab_relaxed$selected))
```

```
## [1] "X238531_x_at"
```

```
PFER_relaxed <- (80^2) / ((2 * 0.51 - 1) * ncol(X_mod))
cat("Implied PFER (very high risk):", round(PFER_relaxed, 1), "\n")
```

```
## Implied PFER (very high risk): 32
```

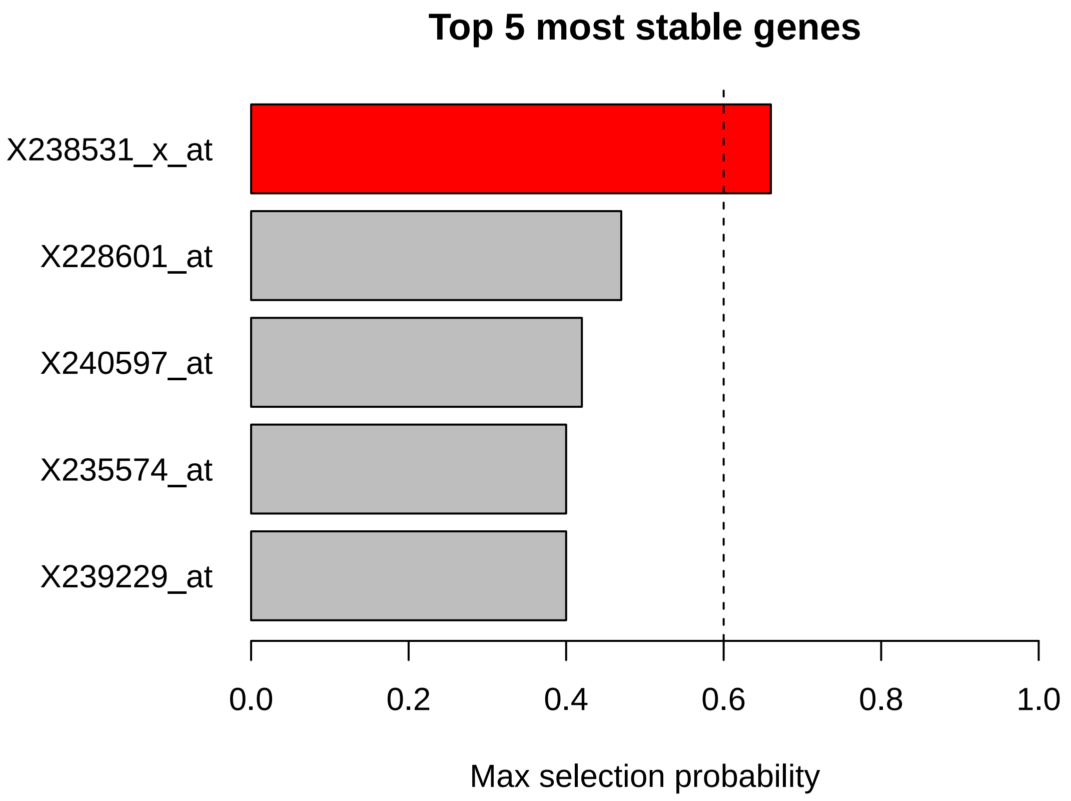
This experiment acts as a methodological "stress test." By lowering the threshold to $\tau = 0.51$, the theoretical Meinshausen-Bühlmann bound skyrockets to PFER $= 32$. In other words, we are theoretically willing to accept up to 32 false positives just to find new candidates. Ideally, such a permissive setting would flood the model with secondary signals. Instead, we observed a striking result: only the single dominant gene (`X238531_x_at`) was recovered. This leads to three crucial conclusions:

1. **Enormous signal gap**: the distance between the top signal and the rest of the transcriptome is vast. There are no "hidden" variables just below the surface waiting to be found; there is one strong driver and a sea of noise.
2. **Invariance**: the selection of `X238531_x_at` is invariant to parameter tuning. Whether we are strict (PFER=3.2) or reckless (PFER=32), it is the only variable that consistently emerges.
3. **Statistical justification**: pursuing additional candidates (like the runner-ups seen in the bar plot in the next subsection) would be statistically unjustified on this dataset alone, as even extreme relaxation fails to distinguish them from random noise.

#### 4.1.8 Exploratory analysis: the top-ranked candidates

While strict error control ($\tau = 0.6$) isolates a single predictor, looking at the ranking of the most stable genes offers valuable biological insights for prioritization. The bar chart below visualizes the "stability gap" between the robust leader and the runners-up.

```
top_idx <- order(pi_max_real, decreasing = TRUE)[1:5]
top_genes <- colnames(X_mod)[top_idx]
top_scores <- pi_max_real[top_idx]

par(mar = c(5, 10, 3, 2))
barplot(rev(top_scores), horiz = TRUE, names.arg = rev(top_genes), las = 1,
        col = ifelse(rev(top_scores) >= tau_real, "red", "grey"),
        xlab = "Max selection probability", main = "Top 5 most stable genes",
        xlim = c(0, 1))
abline(v = tau_real, lty = 2)
```



The chart highlights the stark difference in reliability. The gene `X238531_x_at` (red) is the only one that consistently withstands the subsampling perturbation. The subsequent candidates (grey), such as `X228601_at`, exhibit stability scores in the $0.4 - 0.5$ range. From a decision-making perspective, this plot creates a clear hierarchy for follow-up validation:

1. **High priority**: the red gene is statistically validated with controlled PFER.
2. **Low priority / ambiguous**: the grey genes might be biologically relevant (potential False Negatives due to the conservative nature of the method), but they are statistically indistinguishable from unstable noise selection. Investing resources in them carries a higher risk of failure compared to the stable target.

#### 4.1.9 Discussion of real data results

The application of Stability Selection to the kidney transplant dataset highlights the critical distinction between statistical prediction and structural identification in high-dimensional biomedicine. By contrasting the baseline Lasso with the CPSS framework, we draw three final conclusions:

1. **Overcoming the "Curse of dimensionality":** In a setting with $p = 10,000$ and $n = 250$, standard methods are prone to overfitting and noise accumulation. The standard Lasso selected **8 genes** at $\lambda_{1se}$. While this yields a sparse model, the lack of error control means we cannot ascertain how many of these candidates are false positives driven by the high variance of the sample.

2. **Strict error control as a quality filter:** Stability Selection acted as a rigorous high-pass filter. By enforcing a strict stability threshold ($\tau = 0.6$) and controlling the Per-Family Error Rate ($PFER \leq 3.2$), the method successfully suppressed the noise, reducing the candidate list from 8 down to a **single, high-confidence target**: `X238531_x_at`. This massive reduction is not a loss of information, but a gain in precision. It transforms a "noisy" list of suspects into a single "convicted" biomarker.

3. **Actionable biological insight:** The “Stability Gap” visualized in the bar plot provides a transparent hierarchy for decision-making.

   - **The red gene (`X238531_x_at`):** Is the “Gold Standard” candidate, validated by the rigorous subsampling process.
   - **The grey genes (runner-ups):** Represent the “twilight zone”. They may be biologically relevant (potential false negatives due to the conservative nature of the bound), but they lack the statistical evidence to be claimed with certainty.

In conclusion, Stability Selection successfully prioritized safety over model complexity. It demonstrated that in noisy omics scenarios, it is scientifically preferable to identify one robust, reproducible signal with controlled error guarantees than to pursue a larger panel of predictors contaminated by false discoveries.

## 5 General conclusions

In this report, we investigated **stability selection** as a rigorous statistical framework to address the structural instability of variable selection in high-dimensional regression ($p \gg n$). By shifting the focus from minimising prediction error (as in standard Cross-Validation) to maximizing **selection reliability**, we demonstrated how this method provides a principled solution to the "Curse of Dimensionality."

Our analysis, conducted through two complementary simulation designs and a genuine biomedical application, yields three fundamental insights:

1. **Robustness against multicollinearity (Design A):** in the presence of strong block correlation ($\rho = 0.7$), standard Lasso proved unreliable, selecting a mix of true signals and correlated noise (9 False Positives). By implementing **Complementary Pairs Stability Selection (CPSS)**, we visualized the phenomenon of "Signal Dilution": the algorithm correctly identified the ambiguity of the data, refusing to assign high confidence to surrogate variables. This confirms that in hostile environments, Stability Selection prioritises **safety**, filtering out spurious associations that standard methods would mistakenly validate.
2. **Validation of theoretical guarantees (Design B):** in an idealized independent setting, we strictly verified the **Meinshausen-Bühlmann PFER bound**. The CPSS algorithm achieved **perfect structure recovery** ($TP = 4, FP = 0$), keeping the empirical error strictly below the theoretical limit ($PFER \leq 10$). This serves as a crucial "control experiment," proving that the method is not merely a heuristic, but a procedure with formal statistical control over False Discoveries.
3. **From noise to knowledge (real-world application):** the application to the `Einecke2010Kidney` dataset ($p = 10,000$) provided the most striking evidence of utility. While the baseline Lasso selected a potentially noisy panel of 8 genes, Stability Selection acted as a rigorous high-pass filter. By enforcing a strict error bound ($PFER \leq 3.2$), it distilled the candidate list down to a **single, high-confidence biomarker** (`X238531_x_at`). This result illustrates the practical value of the method: it protects researchers from pursuing expensive follow-up experiments on false positives, effectively separating the "signal" from the background noise of the transcriptome.

**Final recommendation:** stability selection should not be viewed merely as a competitor to the Lasso, but as a necessary **quality assurance layer**. In high-stakes scientific domains (such as genomics or clinical diagnostics) where the cost of a False Positive is high, the explicit error control provided by the PFER bound makes Stability Selection the superior choice for reproducible feature selection.

## 6 Further improvements and perspectives

While our analysis successfully demonstrated the robustness of stability selection, several methodological and biological aspects remain open for further refinement. We outline four key directions for improving sensitivity, theoretical validity, and real-world interpretability, together with a critical assessment of the limitations inherent in our current design.

**1. Stratified stability selection for imbalanced outcomes**

In the kidney dataset (67 rejections vs 183 non-rejections), complementary half-sampling does not guarantee that class proportions are preserved. Some half-samples may contain very few rejection cases, artificially lowering the stability of clinically relevant genes.

*Improvement:* implement **stratified CPSS**, where half-samples are drawn separately within each class to maintain the original class ratio. Since this functionality is not natively supported in the `stabs` package, it would require a custom modification of the subsampling routine. Stratification would reduce variance in the logistic base learner and might enable moderately stable genes (e.g., `X229367_s_at`) to cross the stability threshold $\tau$.

**2. Cluster-based stability selection in high-correlation regimes**

Design A showed clear evidence of "signal dilution": with strong correlations ($\rho = 0.7$), the Lasso alternates among highly correlated surrogates, distributing selection probability across them and violating the exchangeability assumption.

*Improvement:* pre-cluster predictors using gene co-expression or correlation networks, apply stability selection to cluster representatives, and propagate cluster-level stability back to individual genes. This enforces stability at the level of correlated gene modules—more consistent with biological pathways—and mitigates vote-splitting among surrogates.

**3. Refining the base procedure: Randomized Lasso and Elastic Net**

The standard Lasso relies on the *irrepresentable condition*, which is often violated in high-correlation settings. Even with stability selection, this limitation can reduce sensitivity.

*Improvement:*

- **Randomized Lasso** (Meinshausen & Bühlmann, 2010): introduces random penalisation weights, weakening the irrepresentable condition and reducing deterministic preference for specific correlated variables.
- **Elastic Net** (Zou & Hastie, 2005): incorporates an $\ell_2$ penalty that promotes the grouping effect, stabilising the selection of correlated predictors.

Both approaches would likely improve the sensitivity of stability selection in block-correlated designs like Design A.

**4. Towards biological validity: interpretation and external replication**

Selecting a statistically stable gene in a single microarray dataset is insufficient for biological discovery, particularly given the prevalence of platform-specific noise.

*Improvement:*

- **External validation:** the most reliable assessment is to test the stability-selected gene (`X238531_x_at`) on an independent kidney transplant cohort, ideally obtained from a different laboratory or microarray platform to exclude batch-driven artefacts.
- **Quality control:** before interpreting any biological conclusion, rigorous investigation of **batch effects**, platform variability, and preprocessing steps is essential to ensure that the signal is biological rather than technical.

- **Biological interpretation:** perform Gene Ontology enrichment, pathway analysis, and a literature review to establish whether the gene has known mechanistic links to graft rejection.

## 5. Methodological caveats

Two methodological limitations of our current implementation must be explicitly acknowledged:

- **Manual CPSS control in Design A**
  In Design A, we used a fixed $\lambda$-grid for all subsamples and did not explicitly control the parameter $q$ (expected number of selected variables per half-sample). Therefore, the theoretical PFER bound does **not** formally apply in that simulation. Future work will adopt a $q$-constrained selection path or rely directly on the official `stabs` implementation to enforce theoretical guarantees.

- **Threshold selection**
  The stability threshold $\tau = 0.6$ was chosen as a compromise to obtain a low expected PFER ($< 3.2$), but this choice warrants a sensitivity analysis. Lower $\tau$ increases recall but weakens error control; higher $\tau$ increases specificity at the risk of discarding moderately stable true signals. Examining model behaviour across a range of $\tau$ values would provide a more complete understanding of the trade-offs involved.