

# Package: ACSSpack (via r-universe)

May 27, 2026

**Type** Package

**Title** ACSS, Corresponding INSS, and GLP Algorithms

**Version** 1.0.0.2

**Date** 2025-10-10

**Description** Allow user to run the Adaptive Correlated Spike and Slab (ACSS) algorithm, corresponding INdependent Spike and Slab (INSS) algorithm, and Giannone, Lenza and Primiceri (GLP) algorithm with adaptive burn-in. All of the three algorithms are used to fit high dimensional data set with either sparse structure, or dense structure with smaller contributions from all predictors. The state-of-the-art GLP algorithm is in Giannone, D., Lenza, M., & Primiceri, G. E. (2021, ISBN:978-92-899-4542-4) ``Economic predictions with big data: The illusion of sparsity". The two new algorithms, ACSS algorithm and INSS algorithm, and the discussion on their performance can be seen in Yang, Z., Khare, K., & Michailidis, G. (2024, submitted to Journal of Business & Economic Statistics) ``Bayesian methodology for adaptive sparsity and shrinkage in regression".

**License** GPL-3

**Encoding** UTF-8

**Imports** stats, HDCI (>= 1.0-2), MASS (>= 7.3-60), extraDistr (>= 1.4-4)

**LinkingTo** Rcpp (>= 1.0.11), RcppArmadillo (>= 0.12.6.3.0)

**RoxygenNote** 7.3.3

**Depends** R (>= 3.0.2)

**LazyData** true

**NeedsCompilation** yes

**Author** Ziqian Yang [cre, aut], Kshitij Khare [aut], George Michailidis [aut]

**Maintainer** Ziqian Yang <zi.yang@ufl.edu>

**Repository** <https://cranhaven.r-universe.dev>

**Date/Publication** 2026-05-27 05:02:00 UTC

**RemoteUrl** <https://github.com/cranhaven/cranhaven.r-universe.dev>

**RemoteRef** package/ACSSpack

**RemoteSha** d5313a5579255679a92762fb4c8b073395a21046

**RemoteSubdir** ACSSpack

## Contents

ACSS_gs . . . . .	2
Econ_data . . . . .	4
GLP_gs . . . . .	5
INSS_gs . . . . .	6

<b>Index</b>	<b>9</b>
--------------	----------

---

ACSS_gs	<i>ACSS algorithm</i>
---------	-----------------------

---

## Description

Adaptive Correlated Spike and Slab (ACSS) algorithm with/without adaptive burn-in Gibbs sampler. See paper of Yang, Z., Khare, K., & Michailidis, G. (2024) for details.

## Usage

```
ACSS_gs(
  Y,
  X,
  para = c("1sqrtp", "11", "4sqrtp", "Unif", "customize"),
  a = 1,
  b = 1,
  c = 1,
  s = NA,
  Max_burnin = 10,
  nmc = 5000,
  adaptive_burn_in = TRUE
)
```

## Arguments

Y	A (centered) vector.
X	A (centered in column) matrix.
para	Parameter pre-set. Options include "1sqrtp" (default), "11", "4sqrtp", "Unif", and "customize". See details below. <ul style="list-style-type: none"> <li>"1sqrtp": a=1, b=1, c=1, s=sqrt(p) (default)</li> </ul>

- "11": a=1, b=1, c=1, s=1
- "4sqrtp": a=1, b=1, c=4, s=sqrt(p)
- "Unif": a=1, b=1, c=0.8, s=p/1.2
- "customize": user-defined a,b,c and s. If choose this option, please provide values for a,b,c and s.

If no parameter pre-set is provided, the default "1sqrtp" will be used.

a	shape parameter for marginal of q; default=1.
b	shape parameter for marginal of q; default=1.
c	shape parameter for marginal of $\lambda^2$ ; larger c introduce more shrinkage and stronger correlation. default=1.
s	scale (inversed) parameter for marginal of $\lambda^2$ ; larger s introduce more shrinkage; default=sqrt(p).
Max_burnin	Maximum burn-in (in 100 steps) for adaptive burn-in Gibbs sampler. Minimum value is 10, corresponding to 1000 hard burn-insteps. Default=10.
nmc	Number of MCMC samples. Default=5000.
adaptive_burn_in	Logical. If TRUE, use adaptive burn-in Gibbs sampler; If false, use fixed burn-in with burn-in = Max_burnin. Default=TRUE.

## Value

A list with `betahat`: predicted beta hat from majority voting, and `Gibbs_res`: 5000 samples of beta, q and  $\lambda^2$  from Gibbs sampler.

## Examples

```
## A toy example is given below to save time. The full example can be run to get better results
## by letting nmc=5000 (default).
```

```
n = 30;
p = n;

beta1 = rep(0.1, p);
beta2 = c(rep(0.2, p / 2), rep(0, p / 2));
beta3 = c(rep(0.15, 3 * p / 4), rep(0, ceiling(p / 4)));
beta4 = c(rep(1, p / 4), rep(0, ceiling(3 * p / 4)));
beta5 = c(rep(3, ceiling(p / 20)), rep(0, 19 * p / 20));
betas = list(beta1, beta3, beta2, beta4, beta5);

set.seed(123);
X = matrix(rnorm(n * p), n, p);
Y = c(X %%% betas[[1]] + rnorm(n));

## A toy example with p=30, total Gibbs steps=1100, takes ~0.6s
system.time({mod = ACSS_gs(Y, X, para = "1sqrtp", nmc = 100)};})

mod$beta; ## estimated beta after the Majority voting
hist(mod$Gibbs_res$betamat[1,]); ## histogram of the beta_1
```

```

hist(mod$Gibbs_res$q); ## histogram of the q
hist(log(mod$Gibbs_res$lambda^2)); ## histogram of the log(lambda^2)
hist(mod$Gibbs_res$R_2); ## histogram of the R^2 for each iteration in Gibbs sampler
plot(mod$Gibbs_res$q, type = "l"); ## trace plot of the q
## joint posterior of model density and shrinkage
plot(log(mod$Gibbs_res$q / (1 - mod$Gibbs_res$q)), -log(mod$Gibbs_res$lambda^2),
      xlab = "logit(q)", ylab = "-log(lambda^2)",
      main = "Joint Posterior of Model Density and Shrinkage");

```

---

Econ\_data

*Economic data from the GLP paper*


---

### Description

A list contains the five data set used in the paper of Giannone, Lenza, and Primiceri (2021). Contains the following data sets: Macro1, Macro2, Micro1, Micro2, and Finance1

### Usage

```
Econ_data
```

### Format

```
## 'Econ_data' A list contains the five lists, as the 5 data sets
```

**Macro1** A list with a vector and a data frame, 'Y' and 'X'. 'Y' is the response vector, with 659 observations in it. 'X' is the data frame contains all predictors, with n=659, p=130. It have the structure of time series data.

**Macro2** A list with a vector and a data frame, 'Y' and 'X'. 'Y' is the response vector, with 90 observations in it. 'X' is the data frame contains all predictors, with n=90, p=69. It have the structure of sectional data.

**Micro1** A list with a vector and a data frame, 'Y' and 'X'. 'Y' is the response vector, with 576 observations in it. 'X' is the data frame contains all predictors, with n=576, p=285. It have the structure of panel data with 48 units on 12 time points.

**Micro2** A list with a vector and a data frame, 'Y' and 'X'. 'Y' is the response vector, with 312 observations in it. 'X' is the data frame contains all predictors, with n=312, p=138. It have the structure of panel data with 12 units on 26 time points.

**Finance1** A list with a vector and a data frame, 'Y' and 'X'. 'Y' is the response vector, with 68 observations in it. 'X' is the data frame contains all predictors, with n=68, p=16. It have the structure of time series data.

### Source

<[https://www.econometricsociety.org/publications/econometrica/2021/09/01/economic-predictions-big-data-illusion-sparsity/supp/17842\\_Data\\_and\\_Programs.zip](https://www.econometricsociety.org/publications/econometrica/2021/09/01/economic-predictions-big-data-illusion-sparsity/supp/17842_Data_and_Programs.zip)>

<<https://research.stlouisfed.org/econ/mccracken/fred-databases/>>

GLP\_gs

*GLP algorithm***Description**

Giannone, Lenza and Primiceri (GLP) algorithm with/without adaptive burn-in Gibbs sampler. See paper Giannone, D., Lenza, M., & Primiceri, G. E. (2021) and Yang, Z., Khare, K., & Michailidis, G. (2024) for details.

Most of the codes are from <https://github.com/bfava/IllusionOfIllusion> with our modification to make it have adaptive burn-in Gibbs sampler, and some debugs.

**Usage**

```
GLP_gs(
  Y,
  X,
  a = 1,
  b = 1,
  A = 1,
  B = 1,
  Max_burnin = 10,
  nmc = 5000,
  adaptive_burn_in = TRUE
)
```

**Arguments**

Y	A (centered) vector.
X	A (centered in column) matrix.
a	shape parameter for marginal of q; default=1.
b	shape parameter for marginal of q; default=1.
A	shape parameter for marginal of $R^2$ ; default=1.
B	shape parameter for marginal of $R^2$ ; default=1.
Max_burnin	Maximum burn-in (in 100 steps) for adaptive burn-in Gibbs sampler. Minimum value is 10, corresponding to 1000 hard burn-in steps. Default=10.
nmc	Number of MCMC samples. Default=5000.
adaptive_burn_in	Logical. If TRUE, use adaptive burn-in Gibbs sampler; If false, use fixed burn-in with burn-in = Max_burnin. Default=TRUE.

**Value**

A list with `betahat`: predicted beta hat from majority voting, and `Gibbs_res`: 5000 samples of beta, q and  $\lambda^2$  from Gibbs sampler.

## Examples

```
## A toy example is given below to save your time, which will still take ~10s.
## The full example can be run to get BETTER results, which will take more than 80s,
## by letting nmc=5000 (default).
```

```
n = 30;
p = n;

beta1 = rep(0.1, p);
beta2 = c(rep(0.2, p / 2), rep(0, p / 2));
beta3 = c(rep(0.15, 3 * p / 4), rep(0, ceiling(p / 4)));
beta4 = c(rep(1, p / 4), rep(0, ceiling(3 * p / 4)));
beta5 = c(rep(3, ceiling(p / 20)), rep(0, 19 * p / 20));
betas = list(beta1, beta3, beta2, beta4, beta5);

set.seed(123);
X = matrix(rnorm(n * p), n, p);
Y = c(X %*% betas[[1]] + rnorm(n));

## A toy example with p=30, total Gibbs steps=1100
system.time({mod = GLP_gs(Y, X, nmc = 100);})

mod$beta; ## estimated beta after the Majority voting
hist(mod$Gibbs_res$betamat[1,]); ## histogram of the beta_1
hist(mod$Gibbs_res$q); ## histogram of the q
hist(log(mod$Gibbs_res$lambda^2)); ## histogram of the log(lambda^2)
hist(mod$Gibbs_res$R_2); ## histogram of the R^2 for each iteration in Gibbs sampler
plot(mod$Gibbs_res$q, type = "l"); ## trace plot of the q
## joint posterior of model density and shrinkage
plot(log(mod$Gibbs_res$q / (1 - mod$Gibbs_res$q)), -log(mod$Gibbs_res$lambda^2),
      xlab = "logit(q)", ylab = "-log(lambda^2)",
      main = "Joint Posterior of Model Density and Shrinkage");
```

---

INSS\_gs

*INSS algorithm*

---

## Description

INdependent Spike and Slab (INSS) algorithm with/without adaptive burn-in Gibbs sampler. See paper of Yang, Z., Khare, K., & Michailidis, G. (2024) for details.

## Usage

```
INSS_gs(
  Y,
  X,
  para = c("1sqrtp", "11", "4sqrtp", "Unif", "customize"),
  a = 1,
  b = 1,
```

```

c = 1,
s = NA,
Max_burnin = 10,
nmc = 5000,
adaptive_burn_in = TRUE
)

```

## Arguments

Y	A (centered) vector.
X	A (centered in column) matrix.
para	<p>Parameter pre-set. Options include "1sqrtp" (default), "11", "4sqrtp", "Unif", and "customize". See details below.</p> <ul style="list-style-type: none"> <li>• "1sqrtp": a=1, b=1, c=1, s=sqrt(p) (default)</li> <li>• "11": a=1, b=1, c=1, s=1</li> <li>• "4sqrtp": a=1, b=1, c=4, s=sqrt(p)</li> <li>• "Unif": a=1, b=1, c=0.8, s=p/1.2</li> <li>• "customize": user-defined a,b,c and s. If choose this option, please provide values for a,b,c and s.</li> </ul> <p>If no parameter pre-set is provided, the default "1sqrtp" will be used.</p>
a	shape parameter for marginal of q; default=1.
b	shape parameter for marginal of q; default=1.
c	shape parameter for marginal of $\lambda^2$ ; larger c introduce more shrinkage and stronger correlation. default=1.
s	scale (inversed) parameter for marginal of $\lambda^2$ ; larger s introduce more shrinkage; default=sqrt(p).
Max_burnin	Maximum burn-in (in 100 steps) for adaptive burn-in Gibbs sampler. Minimum value is 10, corresponding to 1000 hard burn-insteps. Default=10.
nmc	Number of MCMC samples. Default=5000.
adaptive_burn_in	Logical. If TRUE, use adaptive burn-in Gibbs sampler; If false, use fixed burn-in with burn-in = Max_burnin. Default=TRUE.

## Value

A list with `betahat`: predicted beta hat from majority voting, and `Gibbs_res`: 5000 samples of beta, q and  $\lambda^2$  from Gibbs sampler.

## Examples

```

## A toy example is given below to save time. The full example can be run to get better results
## by letting nmc=5000 (default).

```

```

n = 30;
p = n;

```

```

beta1 = rep(0.1, p);
beta2 = c(rep(0.2, p / 2), rep(0, p / 2));
beta3 = c(rep(0.15, 3 * p / 4), rep(0, ceiling(p / 4)));
beta4 = c(rep(1, p / 4), rep(0, ceiling(3 * p / 4)));
beta5 = c(rep(3, ceiling(p / 20)), rep(0, 19 * p / 20));
betas = list(beta1, beta3, beta2, beta4, beta5);

set.seed(123);
X = matrix(rnorm(n * p), n, p);
Y = c(X %%% betas[[1]] + rnorm(n));

## A toy example with p=30, total Gibbs steps=1100, takes ~0.6s
system.time({mod = INSS_gs(Y, X, para = "lsqrtp", nmc = 100)});

mod$beta; ## estimated beta after the Majority voting
hist(mod$Gibbs_res$betamat[1,]); ## histogram of the beta_1
hist(mod$Gibbs_res$q); ## histogram of the q
hist(log(mod$Gibbs_res$lambda^2)); ## histogram of the log(lambda^2)
hist(mod$Gibbs_res$R_2); ## histogram of the R^2 for each iteration in Gibbs sampler
plot(mod$Gibbs_res$q, type = "l"); ## trace plot of the q
## joint posterior of model density and shrinkage
plot(log(mod$Gibbs_res$q / (1 - mod$Gibbs_res$q)), -log(mod$Gibbs_res$lambda^2),
      xlab = "logit(q)", ylab = "-log(lambda^2)",
      main = "Joint Posterior of Model Density and Shrinkage");

```

# Index

## \* datasets

Econ\_data, [4](#)

ACSS\_gs, [2](#)

Econ\_data, [4](#)

GLP\_gs, [5](#)

INSS\_gs, [6](#)