Package 'NVCSSL'

January 20, 2025

Type Package

Title Nonparametric Varying Coefficient Spike-and-Slab Lasso

Version 2.0

Date 2023-09-17

Author Ray Bai

Maintainer Ray Bai <raybaistat@gmail.com>

Description Fits Bayesian regularized varying coefficient models with the Nonparametric Varying Coefficient Spike-and-Slab Lasso (NVC-SSL) introduced by Bai et al. (2023) <arXiv:1907.06477>. Functions to fit frequentist penalized varying coefficients are also provided, with the option of employing the group lasso penalty of Yuan and Lin (2006) <doi:10.1111/j.1467-9868.2005.00532.x>, the group minimax concave penalty (MCP) of Breheny and Huang <doi:10.1007/s11222-013-9424-2>, or the group smoothly clipped absolute deviation (SCAD) penalty of Breheny and Huang (2015) <doi:10.1007/s11222-013-9424-2>.

License GPL-3

LazyData true

Depends R (>= 3.6.0)

Imports stats, splines, dae, plyr, Matrix, GIGrvg, MASS, MCMCpack, grpreg, mvtnorm

NeedsCompilation yes

Repository CRAN

Date/Publication 2023-09-17 22:50:19 UTC

Contents

NVC_frequentist *Fits frequentist penalized nonparametric varying coefficient (NVC) models*

Description

This function implements frequentist penalized nonparametric varying coefficient (NVC) models. It supports the following penalty functions: the group lasso penalty of Yuan and Lin (2006), the group minimax concave penalty (MCP) of Breheny and Huang (2015), and the group smoothly clipped absolute deviation (SCAD) penalty of Breheny and Huang (2015). This function solves a penalized regression problem of the form,

$$argmax_\gamma \frac{1}{N}\ell(\gamma) + pen_\lambda(\gamma),$$

where N is the total number of observations, $\ell(\gamma)$ is the loss function, and $pen_\lambda(\cdot)$ is a penalty function with regularization parameter $\lambda > 0$. Since the objective function is rescaled by $1/N$, the penalty λ is typically smaller than the spike hyperparameter λ_0 used by the NVC_SSL function. The BIC criterion is used to select the optimal tuning parameter λ.

Usage

```
NVC_frequentist(y, t, X, n_basis=8, penalty=c("gLASSO","gSCAD","gMCP"),
                lambda=NULL, include_intercept=TRUE)
```

Arguments

y	$N \times 1$ vector of response observations $y_{11}, ..., y_{1m_1}, ..., y_{n1}, ..., y_{nm_n}$
t	$N \times 1$ vector of observation times $t_{11}, ..., t_{1m_1}, ..., t_{n1}, ..., t_{nm_n}$
X	$N \times p$ design matrix with columns $[X_1, ..., X_p]$, where the kth column contains the entries $x_{ik}(t_{ij})$'s
n_basis	number of basis functions to use. Default is n_basis=8.
penalty	string specifying which penalty function to use. Specify "gLASSO" for group lasso, "gSCAD" for group SCAD, or "gMCP" for group MCP.
lambda	grid of tuning parameters. If lambda is not specified (i.e. lambda=NULL), then the program automatically chooses a grid for lambda. Note that since the objective function is scaled by $1/N$, the automatically chosen grid for lambda typically consists of smaller values than the default grid for lambda0 used by the function NVC_SSL.
include_intercept	Boolean variable for whether or not to include an intercept function $\beta_0(t)$ in the estimation. Default is include_intercept=TRUE.

Value

The function returns a list containing the following components:

`t_ordered` all N time points ordered from smallest to largest. Needed for plotting.

`classifications`
$p \times 1$ vector of indicator variables, where `"1"` indicates that the covariate is selected and `"0"` indicates that it is not selected. These classifications are determined by the optimal `lambda` chosen from BIC. Note that this vector does not include an intercept function.

`beta_hat` $N \times p$ matrix of the estimates for varying coefficient functions $\beta_k(t), k = 1, ..., p$, using the optimal `lambda` chosen from BIC. The kth column in the matrix is the kth estimated function at the observation times in `t_ordered`.

`beta0_hat` estmate of the intercept function $\beta_0(t)$ at the observation times in `t_ordered` for the optimal `lambda` chosen from BIC. This is not returned if `include_intercept = FALSE`.

`gamma_hat` estimated basis coefficients (needed for prediction) for the optimal `lambda`.

`lambda_min` the individual `lambda` which minimizes the BIC. If only one value was originally passed for `lambda`, then this just returns that `lambda`.

`lambda0_all` grid of all L regularization parameters in `lambda`. Note that since the objective function is scaled by $1/N$ for the penalized frequentist methods in the `NVC_frequentist` function, the `lambda_all` grid that is chosen automatically by `NVC_frequentist` typically consists of smaller values than the default values in the `lambda0_all` grid for `NVC_SSL`.

`BIC_all` $L \times 1$ vector of BIC values corresponding to all L entries in `lambda_all`. The lth entry corresponds to the lth entry in `lambda_all`.

`beta_est_all_lambda`
list of length L of the estimated varying coefficients $\beta_k(t), k = 1, ..., p$, corresponding to all L lambdas in `lambda_all`. The lth entry corresponds to the lth entry in `lambda_all`.

`beta0_est_all_lambda`
$N \times L$ matrix of estimated intercept function $\beta_0(t)$ corresponding to all L entries in `lambda_all`. The lth column corresponds to the lth entry in `lambda_all`. This is not returned if `include_intercept=FALSE`.

`gamma_est_all_lambda`
$dp \times L$ matrix of estimated basis coefficients corresponding to all entries in `lambda_all`. The lth column corresponds to the lth entry in `lambda_all`.

`classifications_all_lambda`
$p \times L$ matrix of classifications corresponding to all the entries in `lambda_all`. The lth column corresponds to the lth entry in `lambda_all`.

`iters_to_converge`
number of iterations it took for the group ascent algorithm to converge for each entry in `lambda_all`. The lth entry corresponds to the lth entry in `lambda_all`.

References

Bai, R., Boland, M. R., and Chen, Y. (2023). "Scalable high-dimensional Bayesian varying coefficient models with unknown within-subject covariance." *arXiv pre-print arXiv:arXiv:1907.06477*.

Breheny, P. and Huang, J. (2015). "Group descent algorithms for nonconvex penalized linear and logistic regression models with grouped predictors." *Statistics and Computing*, **25**:173-187.

Wei, F., Huang, J., and Li, H. (2011). "Variable selection and estimation in high-dimensional varying coefficient models." *Statistica Sinica*, **21**:1515-1540.

Yuan, M. and Lin, Y. (2006). "Model selection and estimation in regression with grouped variables." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **68**:49-67.

Examples

```
## Load data
data(SimulatedData)
attach(SimulatedData)
y = SimulatedData$y
t = SimulatedData$t
id = SimulatedData$id
X = SimulatedData[,4:103]

## Fit frequentist penalized NVC model with the SCAD penalty.
## Can set penalty as "gLASSO", "gSCAD", or "gMCP".
## No need to specify an 'id' argument when using NVC_frequentist() function

NVC_gSCAD_mod = NVC_frequentist(y, t, X, penalty="gSCAD")

## Classifications. First varying coefficients are selected as nonzero
NVC_gSCAD_mod$classifications

## Optimal lambda chosen from BIC
NVC_gSCAD_mod$lambda_min

## Plot first estimated varying coefficient function
t_ordered = NVC_gSCAD_mod$t_ordered
beta_hat= NVC_gSCAD_mod$beta_hat
plot(t_ordered, beta_hat[,1], lwd=3, type='l', col='blue',
     xlab="Time", ylim = c(-12,12), ylab=expression(beta[1]))

## Plot third estimated varying coefficient function
plot(t_ordered, beta_hat[,3], lwd=3, type='l', col='blue',
     xlab="Time", ylim = c(-4,2), ylab=expression(beta[3]))

## Plot fifth estimated varying coefficient function
plot(t_ordered, beta_hat[,5], lwd=3, type='l', col='blue',
     xlab="Time", ylim = c(0,15), ylab=expression(beta[5]))
```

NVC_predict *Prediction for nonparametric varying coefficient (NVC) models*

Description

This is a function to predict the responses $y(t_{new})$ for new subjects at new time points t_{new} with new covariates X_{new}. The function accepts an estimated NVC model that was fit using either the NVC_SSL or NVC_frequentist functions and returns the predicted $y(t)$'s. This function can be used for either out-of-sample predictions or for in-sample predictions if the "new" subjects are the same as the ones used to obtain the fitted NVC model.

Usage

```
NVC_predict(NVC_mod, t_new, id_new, X_new)
```

Arguments

NVC_mod	an object with a fitted NVC model returned by the NVC_SSL or NVC_frequentist function
t_new	vector of new observation times
id_new	vector of new labels, where a label corresponds to one of the new subjects
X_new	new design matrix with columns $[X_1, \ldots, X_p]$ where the kth column corresponds to the kth covariate. X_new must have the p columns, i.e. the same number of varying coefficients estimated by NVC_mod.

Value

The function returns a list containing the following components:

id	vector of each ith subject's label
time	vector of each jth observation time for each ith subject
y_pred	vector of predicted responses corresponding to each jth observation time for each ith subject

References

Bai, R., Boland, M. R., and Chen, Y. (2023). "Scalable high-dimensional Bayesian varying coefficient models with unknown within-subject covariance." *arXiv pre-print arXiv:arXiv:1907.06477*.

Examples

```
## Load simulated data
data(SimulatedData)
attach(SimulatedData)
y = SimulatedData$y
t = SimulatedData$t
id = SimulatedData$id
```

```
X = SimulatedData[,4:103]

## Fit frequentist penalized NVC model with the group lasso penalty.
## No need to specify an 'id' argument when using NVC_frequentist() function.

NVC_gLASSO_mod = NVC_frequentist(y=y, t=t, X=X, penalty="gLASSO")

## Make in-sample predictions. Here, we DO need to specify 'id' argument

NVC_gLASSO_predictions = NVC_predict(NVC_gLASSO_mod, t_new=t, id_new=id, X_new=X)

## Subjects
NVC_gLASSO_predictions$id

## Observation times
NVC_gLASSO_predictions$time

## Predicted responses
NVC_gLASSO_predictions$y_pred



## Fit NVC-SSL model to the data instead. Here, we do need to specify id

NVC_SSL_mod = NVC_SSL(y=y, t=t, id=id, X=X)
NVC_SSL_predictions = NVC_predict(NVC_SSL_mod, t_new = t, id_new=id, X_new=X)

## Subjects
NVC_SSL_predictions$id

## Observation times
NVC_SSL_predictions$time

## Predicted responses
NVC_SSL_predictions$y_pred
```

NVC_SSL *Nonparametric Varying Coefficient Spike-and-Slab Lasso (NVC-SSL)*

Description

This function implements the Nonparametric Varying Coefficient Spike-and-Slab Lasso (NVC-SSL) model of Bai et al. (2023) for high-dimensional NVC models. The function returns the MAP estimator for the varying coefficients $\beta_k(t), k = 1, ..., p$, obtained from the ECM algorithm described in Bai et al. (2023). The BIC criterion is used to select the optimal spike hyperparameter $lambda_0$.

If the user specifies `return_CI=TRUE`, then this function will also return the 95 percent pointwise posterior credible intervals for the varying coefficients $\beta_k(t), k = 1, ..., p$, obtained from Gibbs

sampling. If the number of covariates p is large, then the user can additionally use the approximate MCMC algorithm introduced in Bai et al. (2023) (`approx_MCMC=TRUE`) which is much faster than the exact Gibbs sampler and gives higher simultaneous coverage.

Finally, this function returns the number of iterations and the runtime for the ECM algorithms and MCMC algorithms which can be used for benchmarking and timing comparisons.

Usage

```
NVC_SSL(y, t, id, X, n_basis=8,
        lambda0=seq(from=300,to=10,by=-10), lambda1=1,
        a=1, b=ncol(X), c0=1, d0=1, nu=n_basis+2, Phi=diag(n_basis),
        include_intercept=TRUE, tol=1e-6, max_iter=100,
        return_CI=FALSE, approx_MCMC=FALSE,
        n_samples=1500, burn=500, print_iter=TRUE)
```

Arguments

`y`	$N \times 1$ vector of response observations $y_{11}, ..., y_{1m_1}, ..., y_{n1}, ..., y_{nm_n}$
`t`	$N \times 1$ vector of observation times $t_{11}, ..., t_{1m_1}, ..., t_{n1}, ..., t_{nm_n}$
`id`	$N \times 1$ vector of labels, where each unique label corresponds to one of the subjects
`X`	$N \times p$ design matrix with columns $[X_1, ..., X_p]$, where the kth column contains the entries $x_{ik}(t_{ij})$'s
`n_basis`	number of basis functions to use. Default is `n_basis=8`.
`lambda0`	grid of spike hyperparameters. Default is to tune `lambda0` from the grid of decreasing values $(300, 290, ..., 20, 10)$.
`lambda1`	slab hyperparameter. Default is `lambda1=1`.
`a`	hyperparameter in $B(a, b)$ prior on mixing proportion θ. Default is $a = 1$.
`b`	hyperparameter in $B(a, b)$ prior on mixing proportion θ. Default is $b = p$.
`c0`	hyperparameter in Inverse-Gamma$(c_0/2, d_0/2)$ prior on measurement error variance σ^2. Default is $c_0 = 1$.
`d0`	hyperparameter in Inverse-Gamma$(c_0/2, d_0/2)$ prior on measurement error variance σ^2. Default is $d_0 = 1$.
`nu`	degrees of freedom for Inverse-Wishart prior on Ω. Default is `n_basis+2`.
`Phi`	scale matrix in the Inverse-Wishart prior on Ω. Default is the identity matrix.
`include_intercept`	Boolean variable for whether or not to include an intercept function $\beta_0(t)$ in the estimation. Default is `include_intercept=TRUE`.
`tol`	convergence criteria for the ECM algorithm. Default is `tol=1e-6`.
`max_iter`	maximum number of iterations to run ECM algorithm. Default is `max_iter=100`.
`return_CI`	Boolean variable for whether or not to return the 95 percent pointwise credible bands. Set `return_CI=TRUE` if posterior credible bands are desired.

approx_MCMC Boolean variable for whether or not to run the approximate MCMC algorithm instead of the exact MCMC algorithm. If approx_MCMC=TRUE, then an approximate MCMC algorithm isused. Otherwise, if approx_MCMC=FALSE, the exact MCMC algorithm is used. This argument is ignored if return_CI=FALSE.

n_samples number of MCMC samples to save for posterior inference. The default is to save n_samples=1500. This is ignored if return_CI=FALSE.

burn number of initial MCMC samples to discard during the warm-up period. Default is burn=500. This is ignored if return_CI=FALSE.

print_iter Boolean variable for whether or not to print the progress of the algorithms. Default is print_iter=TRUE.

Value

The function returns a list containing the following components:

t_ordered all N time points ordered from smallest to largest. Needed for plotting

classifications
$p \times 1$ vector of indicator variables, where "1" indicates that the covariate is selected and "0" indicates that it is not selected. These classifications are determined by the optimal lambda0 chosen from BIC. Note that this vector does not include an intercept function.

beta_hat $N \times p$ matrix of the MAP estimates for varying coefficient functions $\beta_k(t), k = 1, ..., p$, using the optimal lambda0 chosen from BIC. The kth column in the matrix is the kth estimated function at the observation times in t_ordered.

beta0_hat MAP estimate of the intercept function $\beta_0(t)$ at the observation times in t_ordered for the optimal lambda0 chosen from BIC. This is not returned if include_intercept = FALSE.

gamma_hat MAP estimates of the basis coefficients (needed for prediction) for the optimal lambda0.

beta_post_mean $N \times p$ matrix of the posterior mean estimates of the varying coefficient functions. The kth column in the matrix is the kth posterior mean estimate for $\beta_k(t)$ at the observation times in t_ordered. This is not returned if return_CI=FALSE.

beta_CI_lower $N \times p$ matrix of the lower endpoints of the 95 percent pointwise posterior credible interval (CI) for the varying coefficient functions. The kth column in the matrix is the lower endpoint for the CI of $\beta_k(t)$ at the observation times in t_ordered. This is not returned if return_CI=FALSE.

beta_CI_upper $N \times p$ matrix of the upper endpoints of the 95 percent pointwise posterior credible interval (CI) for the varying coefficient functions. The kth column in the matrix is the upper endpoint for the CI of $\beta_k(t)$ at the observation times in t_ordered. This is not returned if return_CI=FALSE.

beta0_post_mean
Posterior mean estimate of the intercept function $\beta_0(t)$ at the observation times in t_ordered. This is not returned if return_CI=FALSE.

beta0_CI_lower Lower endpoints of the 95 percent pointwise posterior credible intervals (CIs) for the intercept function $\beta_0(t)$ at the observation times in t_ordered. This is not returned if return_CI=FALSE.

`beta0_CI_upper` Upper endpoints of the 95 percent pointwise posterior credible intervals (CIs) for the intercept function $\beta_0(t)$ at the observation times in `t_ordered`. This is not returned if `return_CI=FALSE`.

`gamma_post_mean`
Posterior mean estimates of all the basis coefficients. This is not returned if `return_CI=FALSE`.

`gamma_CI_lower` Lower endpoints of the 95 percent posterior credible intervals for the basis coefficients. This is not returned if `return_CI=FALSE`.

`gamma_CI_upper` Upper endpoints of the 95 percent posterior credible intervals for the basis coefficients. This is not returned if `return_CI=FALSE`.

`post_incl` $p \times 1$ vector of estimated posterior inclusion probabilities (PIPs) for each of the varying coefficients. The kth entry in `post_incl` is the PIP for β_k. This is not returned if `return_CI=FALSE`.

`lambda0_min` the individual `lambda0` which minimizes the BIC. If only one value was originally passed for `lambda0`, then this just returns that `lambda0`.

`lambda0_all` grid of all L regularization parameters in `lambda0`. Note that since the objective function is scaled by $1/N$ for the penalized frequentist methods in the `NVC_frequentist` function, the `lambda_all` grid that is chosen automatically by `NVC_frequentist` typically consists of smaller values than the default values in the `lambda0_all` grid for `NVC_SSL`.

`BIC_all` $L \times 1$ vector of BIC values corresponding to all L entries in `lambda0_all`. The lth entry corresponds to the lth entry in `lambda0_all`.

`beta_est_all_lambda0`
list of length L of the estimated varying coefficients $\beta_k(t), k = 1, ..., p$, corresponding to all L lambdas in `lambda0_all`. The lth entry corresponds to the lth entry in `lambda0_all`.

`beta0_est_all_lambda0`
$N \times L$ matrix of estimated intercept function $\beta_0(t)$ corresponding to all L entries in `lambda0_all`. The lth column corresponds to the lth entry in `lambda0_all`. This is not returned if `include_intercept=FALSE`.

`gamma_est_all_lambda0`
$dp \times L$ matrix of estimated basis coefficients corresponding to all entries in `lambda0_all`. The lth column corresponds to the lth entry in `lambda0_all`.

`classifications_all_lambda0`
$p \times L$ matrix of classifications corresponding to all the entries in `lambda0_all`. The lth column corresponds to the lth entry in `lambda0_all`.

`ECM_iters_to_converge`
number of iterations it took for the ECM algorithm to converge for each entry in `lambda0_all`. The lth entry corresponds to the lth entry in `lambda0_all`.

`ECM_runtimes` $L \times 1$ vector of the number of seconds it took for the ECM algorithm to converge for each entry in `lambda0_all`. The lth entry corresponds to the lth entry in `lambda0_all`.

`gibbs_runtime` number of minutes it took for the Gibbs sampling algorithm to run for the total number of MCMC iterations given in `gibbs_iters`

`gibbs_iters` total number of MCMC iterations run for posterior inference

References

Bai, R., Boland, M. R., and Chen, Y. (2023). "Scalable high-dimensional Bayesian varying coefficient models with unknown within-subject covariance." *arXiv pre-print arXiv:arXiv:1907.06477*.

Bai, R., Moran, G. E., Antonelli, J. L., Chen, Y., and Boland, M.R. (2022). "Spike-and-slab group lassos for grouped regression and sparse generalized additive models." *Journal of the American Statistical Association*, **117**:184-197.

Examples

```
## Load data
data(SimulatedData)
attach(SimulatedData)
y = SimulatedData$y
t = SimulatedData$t
id = SimulatedData$id
X = SimulatedData[,4:103]

## Fit NVC-SSL model. Default implementation uses a grid of 30 lambdas.
## Below illustration uses just two well-chosen lambdas

NVC_SSL_mod = NVC_SSL(y, t, id, X, lambda0=c(60,50))

## NOTE: Should use default, which will search for lambda0 from a bigger grid
# NVC_SSL_mod = NVC_SSL(y, t, id, X)

## Classifications. First 6 varying coefficients are selected as nonzero
NVC_SSL_mod$classifications

## Optimal lambda chosen from BIC
NVC_SSL_mod$lambda0_min

## Plot first estimated varying coefficient function
t_ordered = NVC_SSL_mod$t_ordered
beta_hat= NVC_SSL_mod$beta_hat
plot(t_ordered, beta_hat[,1], lwd=3, type='l', col='blue',
     xlab="Time", ylim = c(-12,12), ylab=expression(beta[1]))

## Plot third estimated varying coefficient function
plot(t_ordered, beta_hat[,3], lwd=3, type='l', col='blue',
     xlab="Time", ylim = c(-4,2), ylab=expression(beta[3]))

## Plot fifth estimated varying coefficient function
plot(t_ordered, beta_hat[,5], lwd=3, type='l', col='blue',
     xlab="Time", ylim = c(0,15), ylab=expression(beta[5]))


## If you want credible intervals, then set return_CI=TRUE to also run Gibbs sampler.
## Below, we run a total of 1000 MCMC iterations, discarding the first 500 as burnin
## and keeping the final 500 samples for inference.

NVC_SSL_mod_2 = NVC_SSL(y, t, id, X, return_CI=TRUE, approx_MCMC=FALSE,
```

```
                        n_samples=500, burn=500)

## Note that NVC_SSL() always computes a MAP estimator first and then
## initializes the Gibbs sampler with the MAP estimator.

## Plot third varying coefficient function and its credible bands
t_ordered = NVC_SSL_mod_2$t_ordered
beta_MAP = NVC_SSL_mod_2$beta_hat
beta_mean = NVC_SSL_mod_2$beta_post_mean
beta_CI_lower = NVC_SSL_mod_2$beta_CI_lower
beta_CI_upper = NVC_SSL_mod_2$beta_CI_upper

plot(t_ordered, beta_MAP[,3], lwd=3, type='l', col='blue', xlab="Time", ylim=c(-5,3), lty=1,
     ylab=expression(beta[3]), cex.lab=1.5)
lines(t_ordered, beta_mean[,3], lwd=3, type='l', col='red', lty=4)
lines(t_ordered, beta_CI_lower[,3], lwd=4, type='l', col='purple', lty=3)
lines(t_ordered, beta_CI_upper[,3], lwd=4, type='l', col='purple', lty=3)
legend("bottomleft", c("MAP", "Mean", "95 percent CI"), lty=c(1,4,3), lwd=c(2,2,3),
       col=c("blue","red","purple"), inset=c(0,1), xpd=TRUE, horiz=TRUE, bty="n")

## Plot fifth varying coefficient function and its credible bands
plot(t_ordered, beta_MAP[,5], lwd=3, type='l', col='blue', xlab="Time", ylim=c(-1,14), lty=1,
     ylab=expression(beta[5]), cex.lab=1.5)
lines(t_ordered, beta_mean[,5], lwd=3, type='l', col='red', lty=4)
lines(t_ordered, beta_CI_lower[,5], lwd=4, type='l', col='purple', lty=3)
lines(t_ordered, beta_CI_upper[,5], lwd=4, type='l', col='purple', lty=3)
legend("bottomleft", c("MAP", "Mean", "95 percent CI"), lty=c(1,4,3), lwd=c(2,2,3),
       col=c("blue","red","purple"), inset=c(0,1), xpd=TRUE, horiz=TRUE, bty="n")
```

`SimulatedData` *Simulated data for illustration*

Description

This is a simulated dataset for illustration. It contains a total of $N = 436$ observations at irregularly spaced time points for $n = 50$ subjects. There are $p = 100$ covariates.

Usage

```
data(SimulatedData)
```

Details

This simulated dataset contains $N = 436$ observations for $n = 50$ subjects, with $p = 100$ covariates. The first column `y` gives the response variables, the second column `t` gives the observation times, the third column `id` gives the unique IDs for each of the 50 subjects, and columns 4-103 (`x1`, ..., `x100`) give the covariate values.

This synthetic dataset is a slight modification from Experiment 2 in Section 5.1 of Bai et al. (2023). We use $p = 100$ for illustration, instead of $p = 500$ as in the paper.

References

Bai, R., Boland, M. R., and Chen, Y. (2023). "Scalable high-dimensional Bayesian varying coefficient models with unknown within-subject covariance." *arXiv pre-print arXiv:1907.06477*.

Index