Best subset selection for transformation models

Lucas Kook

August 20, 2022

Abstract

The **tramvs** package implements best subset selection for various kinds of transformation models via the abess algorithm. The optimal subset is elicited based on greedily updating the active set of covariates via changes in log-likelihood when in- or excluding a variable. This vignette illustrates the package's functionalities, S3 classes and -methods using simulated and real datasets.

1 Introduction

After introducing notation, the abess algorithm is described for linear transformation models. Extensions to more general transformation models are presented thereafter.

1.1 Notation

Let Y denote a univariate response, $\boldsymbol{x} \in \mathbb{R}^p$ the observed covariates, F_Z an inverse link function, and h the transformation function. We model the conditional distribution of $Y|\boldsymbol{X} = \boldsymbol{x}$ via (Hothorn et al., 2014, 2018)

$$F_Y(y|\boldsymbol{x}) = F_Z(h(y) \pm \boldsymbol{x}^\top \boldsymbol{\beta}). \tag{1}$$

The parameters in the baseline transformation h remain unpenalized. We then take a shorthand in writing $\ell(\boldsymbol{\beta}) := -\sum_{i=1}^n \ell_i(\boldsymbol{\vartheta}, \boldsymbol{\beta}; y_i, \boldsymbol{x}_i)$ for the negative log-likelihood of a transformation model. Note that the likelihood can handle any outcome with at least ordered sample space and any type of uninformative censoring (Hothorn et al., 2018).

Because the parameters in h should remain unpenalized, we consider only $\boldsymbol{\beta} = (\beta_1, \dots, \beta_p)^\top$ for selection. Let $\mathcal{S} = [p]$, *i.e.* the set of all integers $1, \dots, p$. For any $\mathcal{A} \subset \mathcal{S}$, $\mathcal{A}^c = \mathcal{S} \backslash \mathcal{A}$ denotes the complement of $\mathcal{A}$ and $|\mathcal{A}|$ its cardinality. The support (or active set) of $\boldsymbol{\beta}$ is denoted by $\operatorname{supp} \boldsymbol{\beta} = \{j : \beta_j \neq 0\}$. By $\boldsymbol{\beta}^{\mathcal{A}}$ we denote the restriction of $\boldsymbol{\beta}$ to the support set $\mathcal{A}$, *i.e.* $\beta_j^{\mathcal{A}} = 0$ if $j \notin \mathcal{A}$. The ℓ_0 norm can be written as $\|\boldsymbol{\beta}\|_0 = |\operatorname{supp} \boldsymbol{\beta}|$, *i.e.* the number of non-zero entries in $\boldsymbol{\beta}$.

1.2 The abess algorithm

The abess algorithm (Zhu et al., 2020) performs best subset selection for a fixed support size $s \in [p]$,

$$\min_{\boldsymbol{\beta}} \ell(\boldsymbol{\beta}), \quad \text{s.t. } \|\boldsymbol{\beta}\|_0 \leq s, \tag{2}$$

for a general class of models. It requires the computation of two "sacrifices", namely a backward sacrifice ξ_j and a forward sacrifice ζ_j. The backward sacrifice measures the drop in goodness of fit (as measured by the negative log-likelihood, *i.e.* smaller is better) when discarding variable j via

$$\xi_j = \ell(\hat{\boldsymbol{\beta}}^{\mathcal{A}}) - \ell(\hat{\boldsymbol{\beta}}^{\mathcal{A} \backslash \{j\}}). \tag{3}$$

The forward sacrifice measures the benefit of adding variable j via

$$\zeta_j = \ell(\hat{\boldsymbol{\beta}}^{\{j\}})\big|_{\hat{\boldsymbol{\beta}}^{\mathcal{A}}} - \ell(\hat{\boldsymbol{\beta}}^{\mathcal{A}}), \tag{4}$$

where $\ell(\hat{\boldsymbol{\beta}}^{\{j\}})\big|_{\hat{\boldsymbol{\beta}}^{\mathcal{A}}}$ denotes the maximum likelihood when estimating $\boldsymbol{\beta}^{\{j\}}$ while keeping $\hat{\boldsymbol{\beta}}^{\mathcal{A}}$ fixed.

Based on both sacrifices, the `abess` algorithm looks for improvements of the active (and inactive) set for any splicing size $k \leq s$ via

$$\mathcal{A}_k = \left\{ j \in \mathcal{A} : \sum_{i \in \mathcal{A}} \mathbb{1}(\xi_j \geq \xi_i) \leq k \right\}, \tag{5}$$

and

$$\mathcal{I}_k = \left\{ j \in \mathcal{I} : \sum_{i \in \mathcal{I}} \mathbb{1}(\zeta_j \leq \zeta_i) \leq k \right\}, \tag{6}$$

where $\mathcal{I} = \mathcal{S} \backslash \mathcal{A}$ denotes the inactive set. Then, the active set is updated via

$$\tilde{\mathcal{A}} = (\mathcal{A} \backslash \mathcal{A}_k) \cup \mathcal{I}_k, \tag{7}$$

if there is an improvement in the negative log-likelihood (controlled via a tuning parameter τ_s. We choose $\tau_s = 0.01 s \log(p) \log(\log(n))/n$ per default. Further detail can be found in Zhu et al. (2020).

When the support size s is unknown, Zhu et al. (2020) recommend tuning s via a high-dimensional Bayesian information criterion (SIC), given by

$$\mathrm{SIC}(\mathcal{A}) = \ell(\boldsymbol{\beta}^{\mathcal{A}}) + \left\|\boldsymbol{\beta}^{\mathcal{A}}\right\|_0 \log(p) \log\log n. \tag{8}$$

For varying support sizes s, the model with minimal SIC is selected. We illustrate this tuning in Section 2.4 and plot regularization and tuning paths.

Choosing the initial support. For choosing the initial $\mathcal{A}$, Zhu et al. (2020) recommend choosing those k covariates most correlated with the response Y. For transformation models this is problematic because empirical correlations are not well-defined for censored responses. Instead, the default in the **tramvs** package is to choose those k covariates most correlated with the score residuals of an unconditional transformation model, *i.e.* for a single observation y, we compute

$$s(y; \hat{h}) = \partial_\alpha \ell(\alpha; y, h)\big|_{\alpha=0,\ h=\hat{h}}, \tag{9}$$

where $\ell(\alpha; y, h)$ is the likelihood contribution of the observation for the transformation model $F_Y(y) = F_Z(h(y) - \alpha)$ and $\hat{h}$ the most likely unconditional transformation (see, *e.g.* Kook et al., 2021).

2 Illustrations

First, the basic usage of **tramvs** is explained. Then, we illustrate the package using various simulated and real datasets.

2.1 Basic usage

The function `abess_tram()` implements the core algorithm for best subset selection for a fixed support size s.

```
args(abess_tram)

## function (formula, data, modFUN, supp, mandatory = NULL, k_max = supp,
##     thresh = NULL, init = TRUE, m_max = 10, m0 = NULL, ...)
## NULL
```

`abess_tram()` is called by the main function `tramvs()`, which loops over the possible range of supports supplied to the function. It computes a high-dimensional information criterion (SIC) for model selection.

```
args(tramvs)

## function (formula, data, modFUN, mandatory = NULL, supp_max = NULL,
##     k_max = NULL, thresh = NULL, init = TRUE, m_max = 10, m0 = NULL,
##     ...)
## NULL
```

However, instead of supplying `modFUN` (a transformation model function), one can call the respective aliases, `<tram>VS()`, *e.g.* `CoxphVS()`, to skip this step. Further arguments to `modFUN` can be supplied via the `...` argument.

2.2 Interfacing tram

We generate a toy example with three out of ten non-zero coefficients in a normal linear regression model. We benchmark directly against the OLS alternative in the **abess** package (Zhu et al., 2020).

```
N <- 1e2; P <- 10; nz <- 3
beta <- rep(c(3, 0), c(nz, P - nz))
X <- matrix(rnorm(N * P), nrow = N, ncol = P)
Y <- 1 + X %*% beta + rnorm(N)

dat <- data.frame(y = Y, x = X)
cont_res <- tramvs(y ~ ., data = dat, modFUN = Lm)
res_abess <- abess(y ~ ., data = dat, family = "gaussian")
```

The two methods agree on the optimal subset, which can be easily extracted using `support()`.

```
support(cont_res)

## [1] "x.1" "x.2" "x.3"

extract(res_abess, support.size = res_abess$best.size)$support.vars

## [1] "x.1" "x.2" "x.3"
```

However, one can leave the Gaussian world behind by using a more flexible transformation function and a simple alias, like `BoxCoxVS()`.

```
BoxCoxVS(y ~ ., data = dat)
```

More low-level arguments to `BoxCox()` such as `order` or `extrapolate` can be supplied via the `...` argument, as shown below.

```
BoxCoxVS(y ~ ., data = dat, order = 3, extrapolate = TRUE)
```

2.3 Handling mandatory covariates

Mandatory covariates are covariates which should remain in the active set at all times. However, their coefficient estimates do not stay constant when other covariates are in- or excluded. In **tramvs**, mandatory covariates can be specified via a formula supplied to the `mandatory` argument.

```
BoxCoxVS(y ~ ., data = dat, mandatory = y ~ x.1)
```

Note that supplying mandatory covariates also alters the initialization of the active set for the `abess` algorithm. Now, instead of the residuals of the unconditional model, the residuals of `modFUN(mandatory, ...)` will be used.

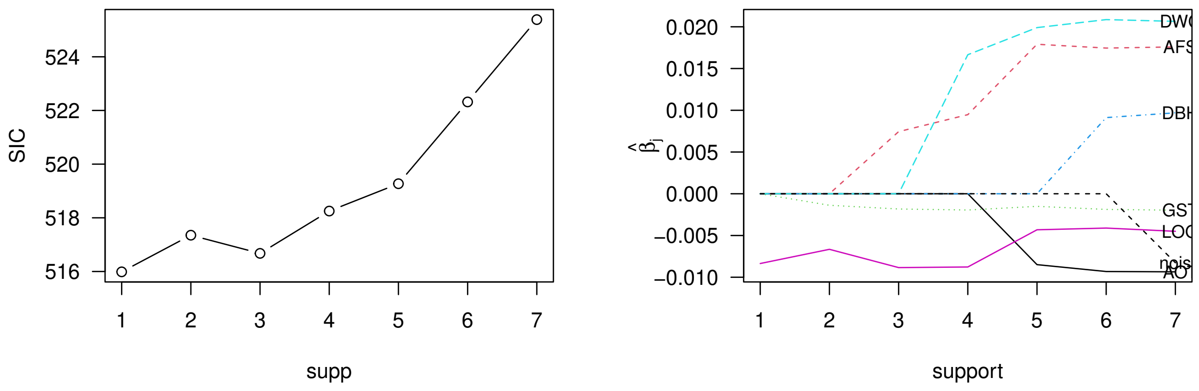

Figure 1: Demonstrating the plotting methods in a **cotram** example.

2.4 Interfacing cotram

Other transformation model add-on packages can be easily included in **tramvs**. The only requirements are a `logLik` method with arguments `newdata` and `parm`, and a `fixed` argument in `modFUN()`. For instance, best subset selection for models in the **cotram** add-on package (Siegfried and Hothorn, 2020) can be done as shown below.

```r
library(cotram)

data("birds", package = "TH.data")
birds$noise <- rnorm(nrow(birds), sd = 10)

# Estimate support sice via HBIC
count_res <- tramvs(SG5 ~ AOT + AFS + GST + DBH + DWC + LOG + noise, data = birds,
                    modFUN = cotram)
```

2.5 Shift-scale transformation models

For shift-scale transformation models,

$$F_Y(y|\boldsymbol{x}) = F_Z\left(\sqrt{\exp(x^\top \boldsymbol{\gamma})}h(y) \pm \boldsymbol{x}^\top \boldsymbol{\beta}\right), \tag{10}$$

the initialization of the active set involves the model matrix for the shift- and scale-terms and the correlation with the shift- and scale-residuals, respectively. Thus, the residuals for the scale term are computed w.r.t. an intercept on the scale of the linear predictor in the scale term which is constrained to zero,

$$r = \partial_\sigma \ell(\sigma; y, \boldsymbol{x}, \hat{h})\big|_{\sigma=1}, \tag{11}$$

for the unconditional transformation model (with $\sigma > 0$)

$$F_Y(y) = F_Z(\sigma h(y)). \tag{12}$$

Shift-scale transformation models can be specified via the formula interface of the usual **tram** functions by using a pipe on the right-hand side of the formula.

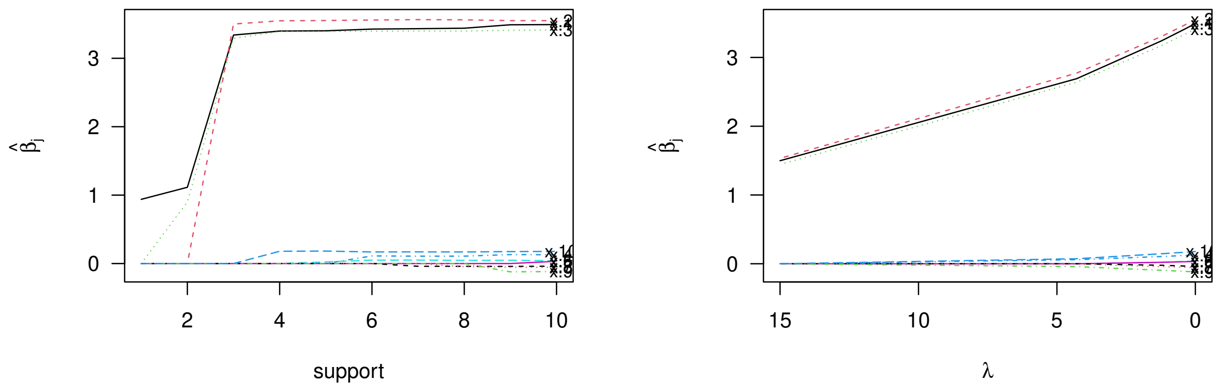

Figure 2: Comparing ℓ_0- and ℓ_1-regularized transformation models from **tramvs** and **tramnet**, respectively.

2.6 Comparison with tramnet

Transformation models with ℓ_1- and ℓ_2-penalties have been described previously and implemented in the **tramnet** package (Kook and Hothorn, 2021). The LASSO and elastic net penalties can be used for variable selection, but also shrink the regression coefficients. Especially the LASSO has trouble dealing with highly correlated covariates and tends to select a single random covariate from a group of highly correlated ones (Zou and Hastie, 2005). Figure 2 juxtaposes ℓ_0- and ℓ_1-regularization paths in the example with continuous response from Section 2.2.

```
m0 <- Lm(y ~ 1, data = dat)
X <- model.matrix(y ~ 0 + ., data = dat)
mt <- tramnet(m0, X, lambda = 0, alpha = 1)
pfl <- prof_lambda(mt, nprof = 5)
```

2.7 S3 methods

Below, all `S3` methods for `"tramvs"` are showcased. Further arguments to the `tram`-specific methods can again be supplied via the ellipses. The plotting methods are illustrated in Fig. 1.

The `summary` method prints the full regularization path alongside a standard `summary.tram` of the best model.

```
# More elaborate summary
summary(cont_res)

##
## L0-penalized tram:
##
##   Normal Linear Regression Model
##
## Call:
## modFUN(formula = formula, data = data, fixed = fix0, theta = theta_init[!names(theta_init) %in%
##     I0])
##
## Coefficients:
##  x.1  x.2  x.3  x.4  x.5  x.6  x.7  x.8  x.9 x.10
## 3.34 3.50 3.29 0.00 0.00 0.00 0.00 0.00 0.00 0.00
##
## Log-Likelihood:
```

```
##  -134 (df = 5)
##
##
## SIC:
##    supp SIC
## 1     1 280
## 2     2 258
## 3     3 145
## 4     4 147
## 5     5 150
## 6     6 153
## 7     7 157
## 8     8 160
## 9     9 163
## 10   10 166
##
##
## Active set: x.1 x.2 x.3
```

The log-likelihood can be computed in and out-of-sample for the best model (`best_only=TRUE`) or all models. Additional arguments to `modFUN` can be supplied, *e.g.* `with_baseline=TRUE`.

```
# logLik of best model
logLik(cont_res)

## 'log Lik.' -134 (df=5)

nparm <- coef(cont_res, with_baseline = TRUE, best_only = TRUE)
logLik(cont_res, newdata = dat[1:5, ], parm = nparm)

## 'log Lik.' -5.05 (df=NULL)
```

The SIC that is printed in the summary can be obtained via `SIC()`.

```
# High-dimensional information criterion
SIC(cont_res)

##    supp SIC
## 1     1 280
## 2     2 258
## 3     3 145
## 4     4 147
## 5     5 150
## 6     6 153
## 7     7 157
## 8     8 160
## 9     9 163
## 10   10 166

SIC(cont_res, best_only = TRUE)

## [1] 145
```

Coefficients are returned as a sparse matrix for all model. In case of `best_only=TRUE`, a numeric vector is returned. Additional arguments to `coef.tram` can be supplied, as shown below.

```
coef(cont_res)

## 10 x 10 sparse Matrix of class "dgCMatrix"
##
```

```
## x.1  0.938 1.114 3.34 3.398 3.4017 3.4266  3.4326  3.4392  3.4898  3.4925
## x.2  .     .     3.50 3.547 3.5519 3.5592  3.5653  3.5619  3.5505  3.5503
## x.3  .     0.899 3.29 3.391 3.3910 3.3948  3.3982  3.3981  3.4096  3.4134
## x.4  .     .     .    .     .      0.1112  0.1077  0.1080  0.1308  0.1272
## x.5  .     .     .    .     0.0232 0.0496  0.0507  0.0464  0.0476  0.0401
## x.6  .     .     .    .     .      .       .       .       .       0.0327
## x.7  .     .     .    .     .      .      -0.0404 -0.0407 -0.0418 -0.0387
## x.8  .     .     .    .     .      .       .      -0.0467 -0.0529 -0.0559
## x.9  .     .     .    .     .      .       .       .      -0.1214 -0.1176
## x.10 .     .     .    0.179 0.1819 0.1694  0.1706  0.1691  0.1746  0.1776

coef(cont_res, best_only = TRUE)

##  x.1  x.2  x.3  x.4  x.5  x.6  x.7  x.8  x.9 x.10
## 3.34 3.50 3.29 0.00 0.00 0.00 0.00 0.00 0.00 0.00

coef(cont_res, as.lm = TRUE)

## 11 x 10 sparse Matrix of class "dgCMatrix"
##
## (Intercept) 0.26 0.947 0.899 0.901 0.8981 0.892  0.8966  0.8893  0.8936  0.8889
## x.1         3.60 3.322 3.089 3.099 3.1020 3.108  3.1108  3.1136  3.1390  3.1398
## x.2         .    .     3.238 3.236 3.2389 3.228  3.2311  3.2246  3.1935  3.1918
## x.3         .    2.680 3.044 3.093 3.0922 3.079  3.0797  3.0763  3.0668  3.0687
## x.4         .    .     .     .     .      0.101  0.0976  0.0978  0.1176  0.1143
## x.5         .    .     .     .     0.0212 0.045  0.0459  0.0420  0.0428  0.0360
## x.6         .    .     .     .     .      .      .       .       .       0.0294
## x.7         .    .     .     .     .      .     -0.0366 -0.0368 -0.0376 -0.0348
## x.8         .    .     .     .     .      .      .      -0.0423 -0.0476 -0.0502
## x.9         .    .     .     .     .      .      .       .      -0.1092 -0.1057
## x.10        .    .     .     0.164 0.1659 0.154  0.1546  0.1531  0.1571  0.1597
```

Several `tram` methods are applicable for the best model in an object of class `"tramvs"`, such as `predict`, `simulate`, and `residuals`.

```
head(predict(cont_res, which = "distribution", type = "trafo"))
simulate(cont_res)[1:5]
head(residuals(cont_res))
```

References

Torsten Hothorn, Thomas Kneib, and Peter Bühlmann. Conditional transformation models. *Journal of the Royal Statistical Society. Series B: Statistical Methodology*, 76(1):3–27, 2014. doi: 10.1111/rssb.12017.

Torsten Hothorn, Lisa Möst, and Peter Bühlmann. Most Likely Transformations. *Scandinavian Journal of Statistics*, 45(1):110–134, 2018. doi: 10.1111/sjos.12291.

Lucas Kook and Torsten Hothorn. Regularized Transformation Models: The tramnet Package. *The R Journal*, 13(1):581–594, 2021. doi: 10.32614/RJ-2021-054.

Lucas Kook, Beate Sick, and Peter Bühlmann. Distributional Anchor Regression. *preprint arXiv:2101.08224*, 2021. URL `http://arxiv.org/abs/2101.08224`.

Sandra Siegfried and Torsten Hothorn. Count transformation models. *Methods in Ecology and Evolution*, 11(7):818–827, 2020. doi: 10.1111/2041-210X.13383.

Junxian Zhu, Canhong Wen, Jin Zhu, Heping Zhang, and Xueqin Wang. A polynomial algorithm for best-subset selection problem. *Proceedings of the National Academy of Sciences*, 117(52):33117–33123, 2020. doi: 10.1073/pnas.2014241117.

Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005. doi: 10.1111/j.1467-9868.2005.00503.x.

3 Session info

```
## R version 4.1.2 (2021-11-01)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 20.04.4 LTS
##
## Matrix products: default
## BLAS:   /usr/lib/x86_64-linux-gnu/blas/libblas.so.3.9.0
## LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.9.0
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=de_CH.UTF-8        LC_COLLATE=C
##  [5] LC_MONETARY=de_CH.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=de_CH.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=de_CH.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] cotram_0.3-1      colorspace_2.0-2  abess_0.3.0       tramnet_0.0-6
##  [5] mlrMBO_1.1.5      smoof_1.6.0.2     checkmate_2.0.0   mlr_2.19.0
##  [9] ParamHelpers_1.14 CVXR_1.0-9        tramvs_0.0-2      tram_0.7-1
## [13] mlt_1.4-1         basefun_1.1-2     variables_1.1-1
##
## loaded via a namespace (and not attached):
##  [1] httr_1.4.2          tidyr_1.1.4         viridisLite_0.4.0
##  [4] jsonlite_1.7.2      bit64_4.0.5         splines_4.1.2
##  [7] ECOSolveR_0.5.4     Formula_1.2-4       assertthat_0.2.1
## [10] highr_0.9           numDeriv_2016.8-1.1 pillar_1.6.4
## [13] backports_1.2.1     lattice_0.20-45     glue_1.4.2
## [16] quadprog_1.5-8      alabama_2015.3-1    digest_0.6.28
## [19] RColorBrewer_1.1-2  sandwich_3.0-1      htmltools_0.5.2
## [22] Matrix_1.4-0        pkgconfig_2.0.3     lhs_1.1.3
## [25] misc3d_0.9-1        purrr_0.3.4         mvtnorm_1.1-3
## [28] scales_1.1.1        parallelMap_1.5.1   mco_1.15.6
## [31] tibble_3.1.5        gmp_0.6-2           generics_0.1.2
## [34] ggplot2_3.3.5       ellipsis_0.3.2      TH.data_1.1-0
## [37] lazyeval_0.2.2      Rmpfr_0.8-6         survival_3.2-13
## [40] RJSONIO_1.3-1.6     magrittr_2.0.1      crayon_1.4.1
## [43] evaluate_0.14       fansi_0.5.0         MASS_7.3-54
## [46] tools_4.1.2         data.table_1.14.2   lifecycle_1.0.1
## [49] BBmisc_1.11         multcomp_1.4-17     stringr_1.4.0
## [52] plotly_4.10.0       munsell_0.5.0       orthopolynom_1.0-5
## [55] compiler_4.1.2      rlang_0.4.12        plot3D_1.4
## [58] grid_4.1.2          coneproj_1.15       htmlwidgets_1.5.4
## [61] tcltk_4.1.2         gtable_0.3.0        codetools_0.2-18
## [64] DBI_1.1.1           BB_2019.10-1        polynom_1.4-0
## [67] R6_2.5.1            zoo_1.8-9           knitr_1.36
## [70] dplyr_1.0.7         fastmap_1.1.0       bit_4.0.4
## [73] utf8_1.2.2          fastmatch_1.1-3     stringi_1.7.5
## [76] parallel_4.1.2      Rcpp_1.0.7          vctrs_0.3.8
## [79] tidyselect_1.1.1    xfun_0.27
```