

The results below are generated from an R script.

```
source("utilAD.R")

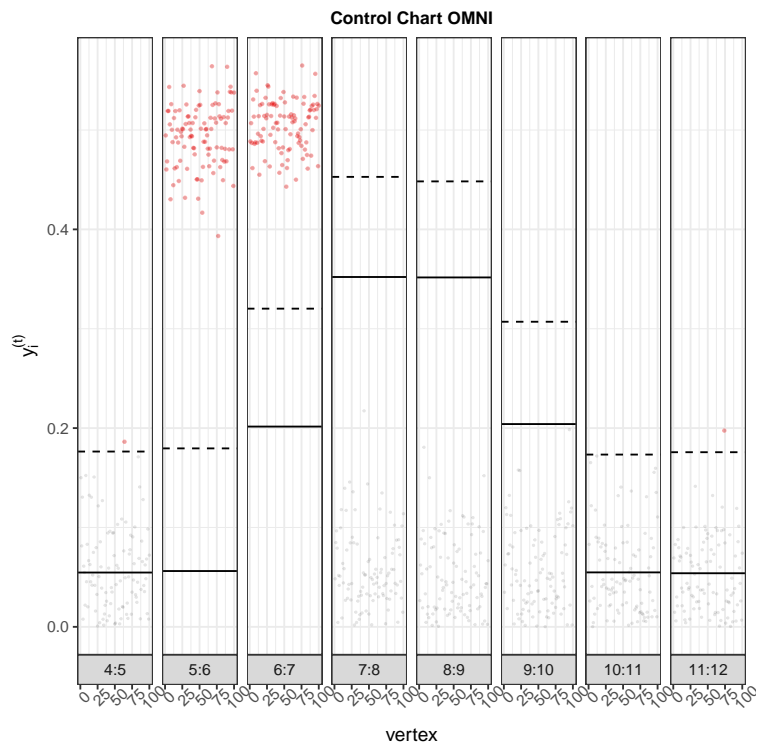
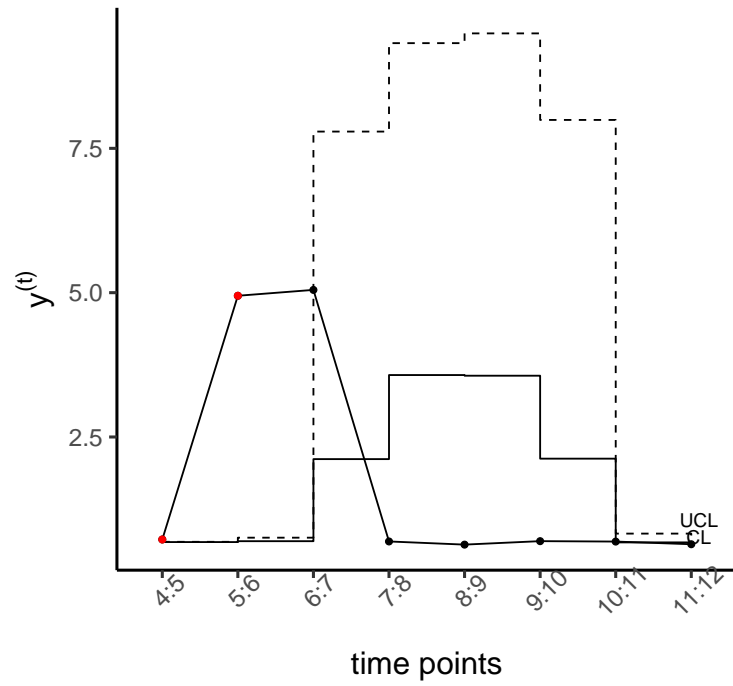
##
## Attaching package: 'igraph'
## The following objects are masked from 'package:stats':
##
##   decompose, spectrum
## The following object is masked from 'package:base':
##
##   union
##
## Attaching package: 'dplyr'
## The following objects are masked from 'package:igraph':
##
##   as_data_frame, groups, union
## The following objects are masked from 'package:stats':
##
##   filter, lag
## The following objects are masked from 'package:base':
##
##   intersect, setdiff, setequal, union
##
##   _ _ _ _ _
## / _ | / _ _ / _ _ | Quality Control Charts and
## | ( _ | | ( _ | ( _ _ Statistical Process Control
## \ _ _ | \ _ _ _ \ _ _ _ |
##   | _ |           version 2.7
## Type 'citation("qcc")' for citing this R package in publications.
##
## Attaching package: 'gtools'
## The following object is masked from 'package:igraph':
##
##   permute

source("simulation.R")
#Example 1
#Generate a time series of ER graphs of length 12,
#create graph anomaly at time point 6
n <- 100
glist <- list()
for (i in 1:5) {
  glist[[i]] <- sample_gnp(n,.1)
}
glist[[6]] <- sample_gnp(n,.9)
glist[[7]] <- sample_gnp(n,.1)
for (i in 8:12) {
  glist[[i]] <- sample_gnp(n,.1)
}
# Do anomaly detection with OMNI, provide the quantitative control chart for GraphAD and VertexAD
result.OMNI<- qccAD(glist, l=4,d=1,dsvd=NULL,method="OMNI",
  diag.augment = TRUE, approx=FALSE, par=FALSE, numpar=2)

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

```
## Loading required package: rARPACK
## Warning: 'switch' is deprecated.
## Use 'strip.position' instead.
## See help("Deprecated")
```

Control Chart OMNI



```

# print the number of deviation for GraphAD, only positive ones are meaningful
print(result.OMNI$GraphAD)

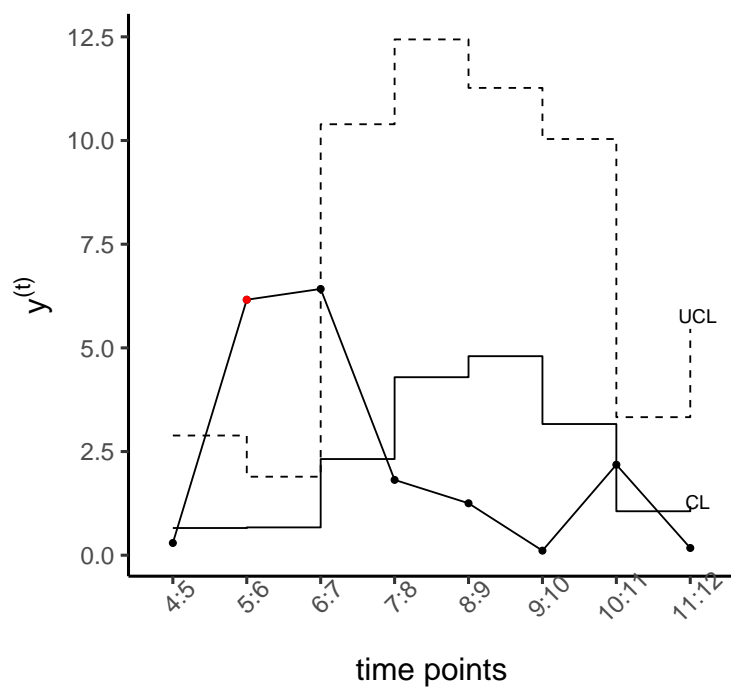
##      Samples
## Group      [,1]
##    1 32.4517461
##    2 210.3132067
##    3  1.5498254
##    4 -1.5041821
##    5 -1.4802684
##    6 -0.7307014
##    7  0.2857340
##    8 -1.0430350

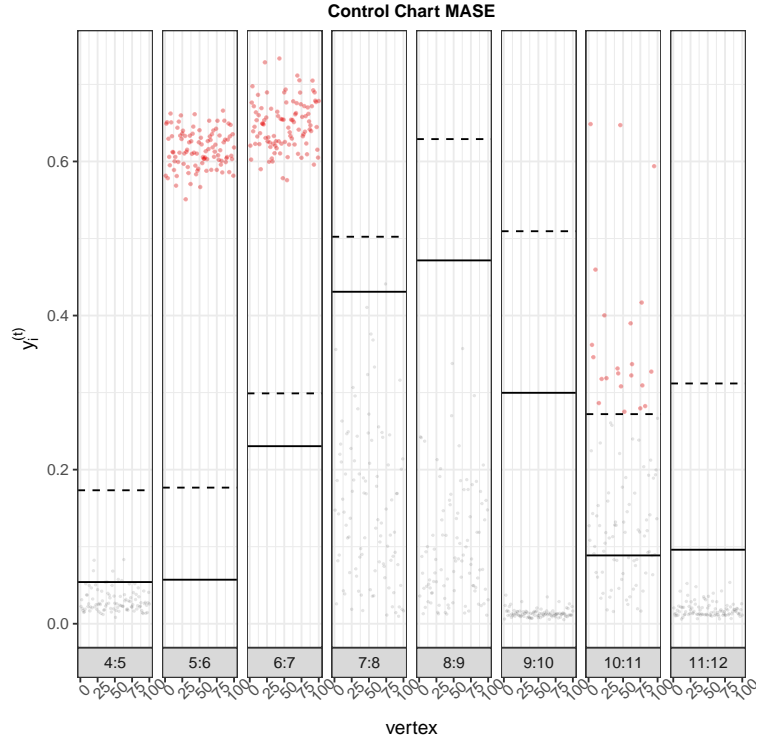
# Do anomaly detection with MASE
result.MASE<- qccAD(glist, l=4,d=2,dsvd=2,method="MASE",
                    diag.augment = TRUE, approx=FALSE, par=FALSE, numpar=2)

## Warning: 'switch' is deprecated.
## Use 'strip.position' instead.
## See help("Deprecated")

```

Control Chart MASE





```
#print the number of deviation for GraphAD, only positive ones are meaningful
print(result.MASE$GraphAD)
```

```
##      Samples
## Group      [,1]
##    1 -0.4848663
##    2 13.4713076
##    3  1.5242595
##    4 -0.9118849
##    5 -1.6446440
##    6 -1.3327488
##    7  1.4852639
##    8 -0.7069135
```

```
#Example 2
```

```
# Sample a time series of RDPG graph (length tmax > 17) with same 1-1 matched vertices unweighted
# hollow symmetric undirected graphs, the latent positions i.i.d uniform.
# Some vertices in 16-th and 17-th graphs are given perturbations so there exists anomalies at 16:17.
```

```
n <- 100 #number of vertices
```

```
nperturb <- 20 #number of perturbed vertices
```

```
cperturb <- .12 #number of perturbation, larger cperturb means more obvious anomalies.
```

```
rmin <- .2 # parameter for uniform[rmin, rmax].
```

```
rmax <- .8 # parameter for uniform[rmin, rmax].
```

```
tmax <- 22 # number of graphs must be greater than 17.
```

```
#Generate data or load the data you want
```

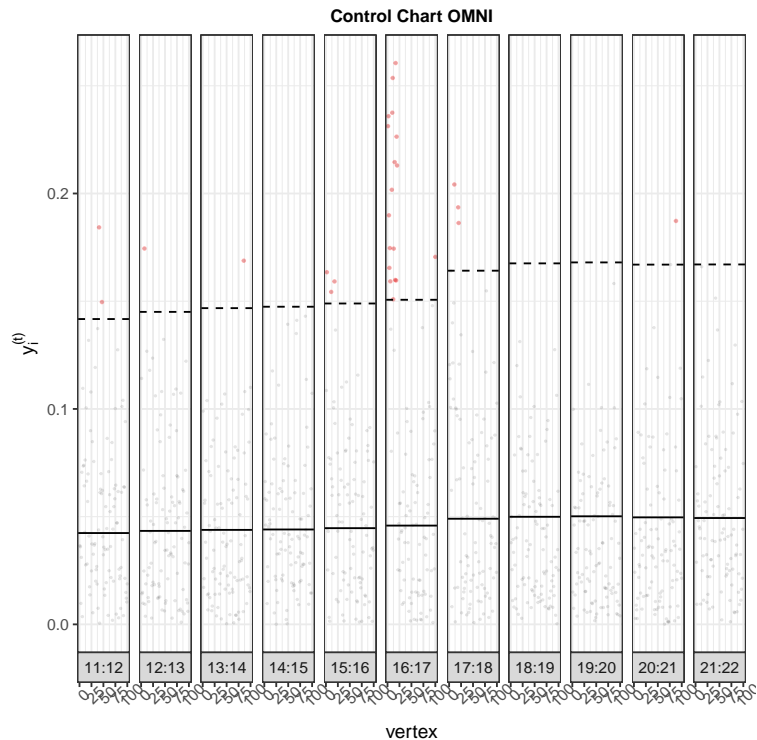
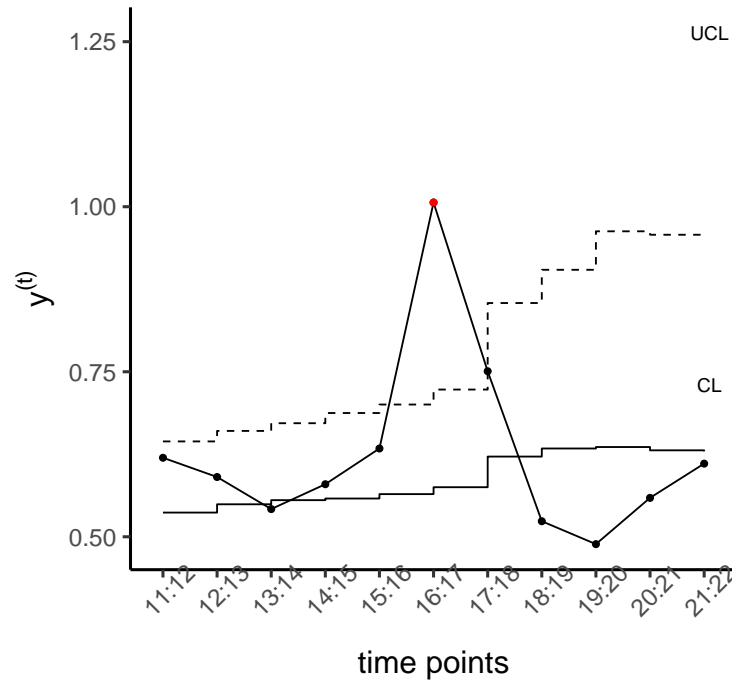
```
glist <- generate.tsg(n, nperturb, cperturb=NULL, rmin, rmax, tmax)$glist
```

```
#Do anomaly detection with OMNI in parallel
```

```
result.OMNI <- qccAD(glist, l=11,d=1,dsvd=NULL,method="OMNI",
  diag.augment = TRUE, approx=FALSE, par=TRUE, numpar=2)
```

```
## Loading required package: doParallel
## Loading required package: iterators
## Loading required package: parallel
## Warning: 'switch' is deprecated.
## Use 'strip.position' instead.
## See help("Deprecated")
```

Control Chart OMNI



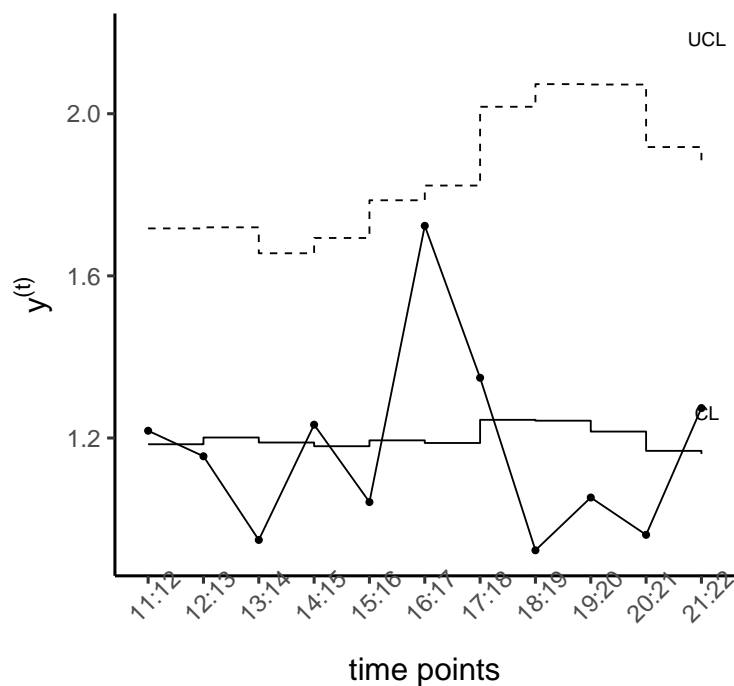
```
#print the number of deviation for GraphAD, only positive ones are meaningful
print(result.OMNI$GraphAD)
```

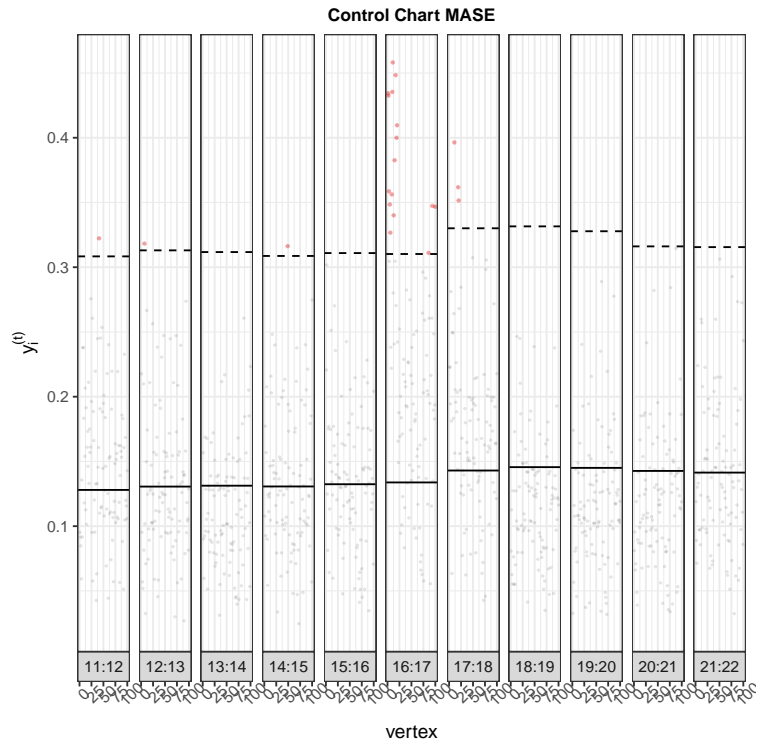
```
##      Samples
## Group      [,1]
##    1  2.3109821
##    2  1.1155971
##    3 -0.3475832
##    4  0.5018647
##    5  1.5277817
##    6  8.7462375
##    7  1.6674425
##    8 -1.2211387
##    9 -1.3504299
##   10 -0.6614613
##   11 -0.1668942
```

```
# Do anomaly detection with MASE in parallel
result.MASE<- qccAD(glist, l=11,d=1,dsvd=2,method="MASE",
                    diag.augment = TRUE, approx=FALSE, par=TRUE, numpar=2)
```

```
## Warning: 'switch' is deprecated.
## Use 'strip.position' instead.
## See help("Deprecated")
```

Control Chart MASE





```
#print the number of deviation for GraphAD, only positive ones are meaningful
print(result.MASE$GraphAD)
```

```
##      Samples
## Group      [,1]
##    1  0.1879697
##    2 -0.2675318
##    3 -1.5444297
##    4  0.3105333
##    5 -0.7710422
##    6  2.5307905
##    7  0.4032188
##    8 -1.1550078
##    9 -0.5702171
##   10 -0.8291563
##   11  0.4696665
```

```
#Example 3
```

```
#five of ER tsg with change point at t=6 and five at t=8.
```

```
n <- 100
```

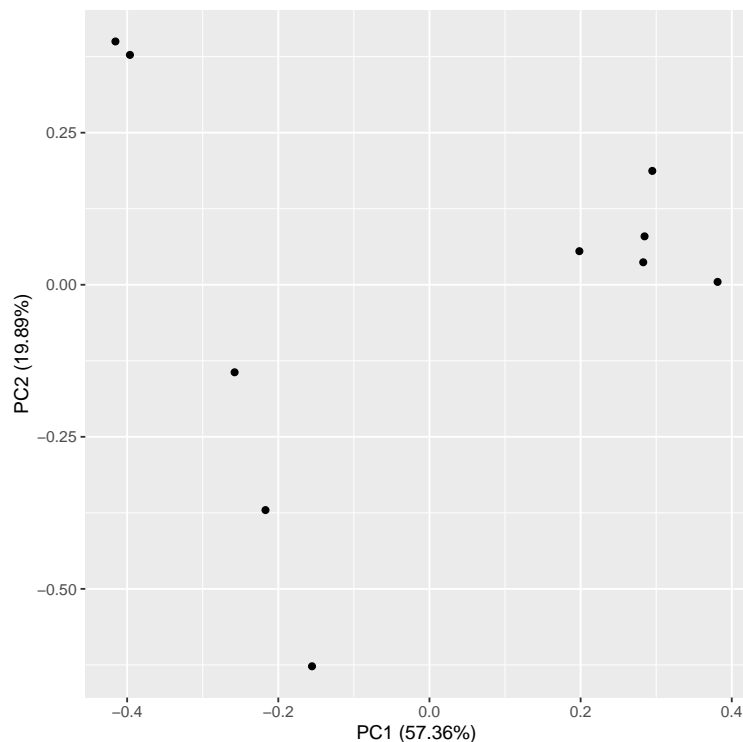
```
dat <- matrix(0, 10, 8)
```

```
for (j in 1:5) {
  glist <- list()
  for (i in 1:5) {
    glist[[i]] <- sample_gnp(n,.1)
  }
  glist[[6]] <- sample_gnp(n,.9)
  for (i in 7:12) {
    glist[[i]] <- sample_gnp(n,.1)
  }
}
```

```

# Do anomaly detection with OMNI, provide the quantitative control chart for GraphAD and VertexAD
result.OMNI<- qccAD(glist, l=4,d=1,dsvd=NULL,method="OMNI",
                    diag.augment = TRUE, approx=FALSE, par=FALSE, numpar=2, plot.figure = FALSE)
dat[j,] <- result.OMNI$GraphAD
}
for (j in 6:10) {
  glist <- list()
  for (i in 1:7) {
    glist[[i]] <- sample_gnp(n,.1)
  }
  glist[[8]] <- sample_gnp(n,.9)
  for (i in 9:12) {
    glist[[i]] <- sample_gnp(n,.1)
  }
  # Do anomaly detection with OMNI, provide the quantitative control chart for GraphAD and VertexAD
  result.OMNI<- qccAD(glist, l=4,d=1,dsvd=NULL,method="OMNI",
                      diag.augment = TRUE, approx=FALSE, par=FALSE, numpar=2, plot.figure = FALSE)
  dat[j,] <- result.OMNI$GraphAD
}
df <- data.frame(dat,class=factor(c(rep("t6",5),rep("t8",5))))
pca_res <- prcomp(dat, scale. = TRUE)
library(ggfortify)
autoplot(pca_res)

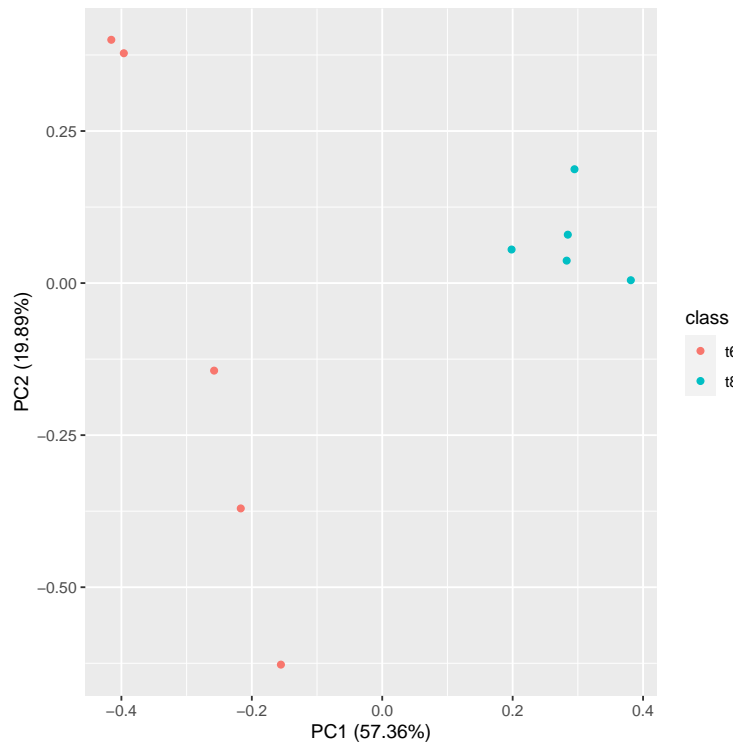
```



```

autoplot(pca_res, data = df, colour="class")

```



The R session information (including the OS info, R version and all packages used):

```
sessionInfo()

## R version 3.6.3 (2020-02-29)
## Platform: x86_64-apple-darwin15.6.0 (64-bit)
## Running under: macOS Catalina 10.15.6
##
## Matrix products: default
## BLAS: /System/Library/Frameworks/Accelerate.framework/Versions/A/Frameworks/vecLib.framework/Versions/A/Libraries/libBLAS.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/3.6/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] parallel stats graphics grDevices utils datasets methods base
##
## other attached packages:
## [1] ggfortify_0.4.8 doParallel_1.0.15 iterators_1.0.12 rARPACK_0.11-0
## [5] Matrix_1.2-18 foreach_1.5.0 gtools_3.8.1 qcc_2.7
## [9] latex2exp_0.4.0 ggplot2_3.3.0 dplyr_1.0.0 igraph_1.2.5
## [13] knitr_1.28 rmarkdown_2.1
##
## loaded via a namespace (and not attached):
## [1] tidyselect_1.1.0 xfun_0.12 purrr_0.3.3 lattice_0.20-38
## [5] colorspace_1.4-1 vctrs_0.3.1 generics_0.0.2 htmltools_0.4.0
## [9] yaml_2.2.1 rlang_0.4.6 pillar_1.4.3 glue_1.3.2
## [13] withr_2.1.2 lifecycle_0.2.0 stringr_1.4.0 munsell_0.5.0
## [17] gtable_0.3.0 codetools_0.2-16 evaluate_0.14 labeling_0.3
## [21] irlba_2.3.3 fansi_0.4.1 highr_0.8 Rcpp_1.0.4
```

```
## [25] scales_1.1.0      farver_2.0.3      RSpectra_0.16-0   gridExtra_2.3
## [29] packrat_0.5.0      digest_0.6.25     stringi_1.4.6     grid_3.6.3
## [33] cli_2.0.2          tools_3.6.3       magrittr_1.5      tibble_3.0.0
## [37] tidyr_1.0.2        crayon_1.3.4.9000 pkgconfig_2.0.3   ellipsis_0.3.0
## [41] MASS_7.3-51.1      assertthat_0.2.1  R6_2.4.1          compiler_3.6.3

Sys.time()

## [1] "2020-08-13 15:01:48 EDT"
```