MScProject

Including library for dynamic factor model - MARSS, and for PCA and low-rank reduction - h2o.

```
library(MARSS)
```

```
## Warning: package 'MARSS' was built under R version 3.6.3
```

```
library(h2o)  # for fitting GLRMs
```

```
## Warning: package 'h2o' was built under R version 3.6.3
```

```
##
## ----------------------------------------------------------------------
##
## Your next step is to start H2O:
##     > h2o.init()
##
## For H2O package documentation, ask for help:
##     > ??h2o
##
## After starting H2O, you can use the Web UI at http://localhost:54321
## For more information visit https://docs.h2o.ai
##
## ----------------------------------------------------------------------
```

```
##
## Attaching package: 'h2o'
```

```
## The following objects are masked from 'package:stats':
##
##     cor, sd, var
```

```
## The following objects are masked from 'package:base':
##
##     %*%, %in%, &&, ||, apply, as.factor, as.numeric, colnames,
##     colnames<-, ifelse, is.character, is.factor, is.numeric, log,
##     log10, log1p, log2, round, signif, trunc
```

Other data manupulation libraries.

```
library("dplyr")
```

```
## Warning: package 'dplyr' was built under R version 3.6.3
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library (plyr)
```

```
## ------------------------------------------------------------------------------
```

```
## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dply
r:
## library(plyr); library(dplyr)
```

```
## ------------------------------------------------------------------------------
```

```
##
## Attaching package: 'plyr'
```

```
## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
```

I have worked on multivariate time series data. Original data had daily observations for the concentration of various pollutants. I have considered average weekly data to bring down the noise.

```
df = read.csv("C:\\Users\\vishw\\Contents\\Semester 10\\M Sc Project\\city_weekly.cs
v", header=T)
print(head(df))
```

```
##         Date    PM2.5       NO      NO2      NOx       NH3        CO  Benzene
## 1 2015-01-04 184.6050 45.49750 34.12000 73.82500  66.39750 11.722500 8.270000
## 2 2015-01-11 196.0057 21.99714 41.18714 49.00000 133.79429 10.171429 4.744286
## 3 2015-01-18 187.6500 36.82286 49.24000 67.43429 138.23857 10.181429 7.085714
## 4 2015-01-25 156.3571 15.00000 24.63429 34.92714  74.57857  9.458571 3.674286
## 5 2015-02-01 162.7843 25.28571 37.27857 49.13714  63.97000 10.837143 4.037143
## 6 2015-02-08 149.7229 25.58286 44.22571 53.58857  63.38714  8.181429 4.285714
##     Toluene      AQI
## 1 16.222500 347.0000
## 2  9.361429 358.4286
## 3 15.117143 360.0000
## 4  8.230000 326.1429
## 5  8.560000 324.1429
## 6  9.508571 316.1429
```

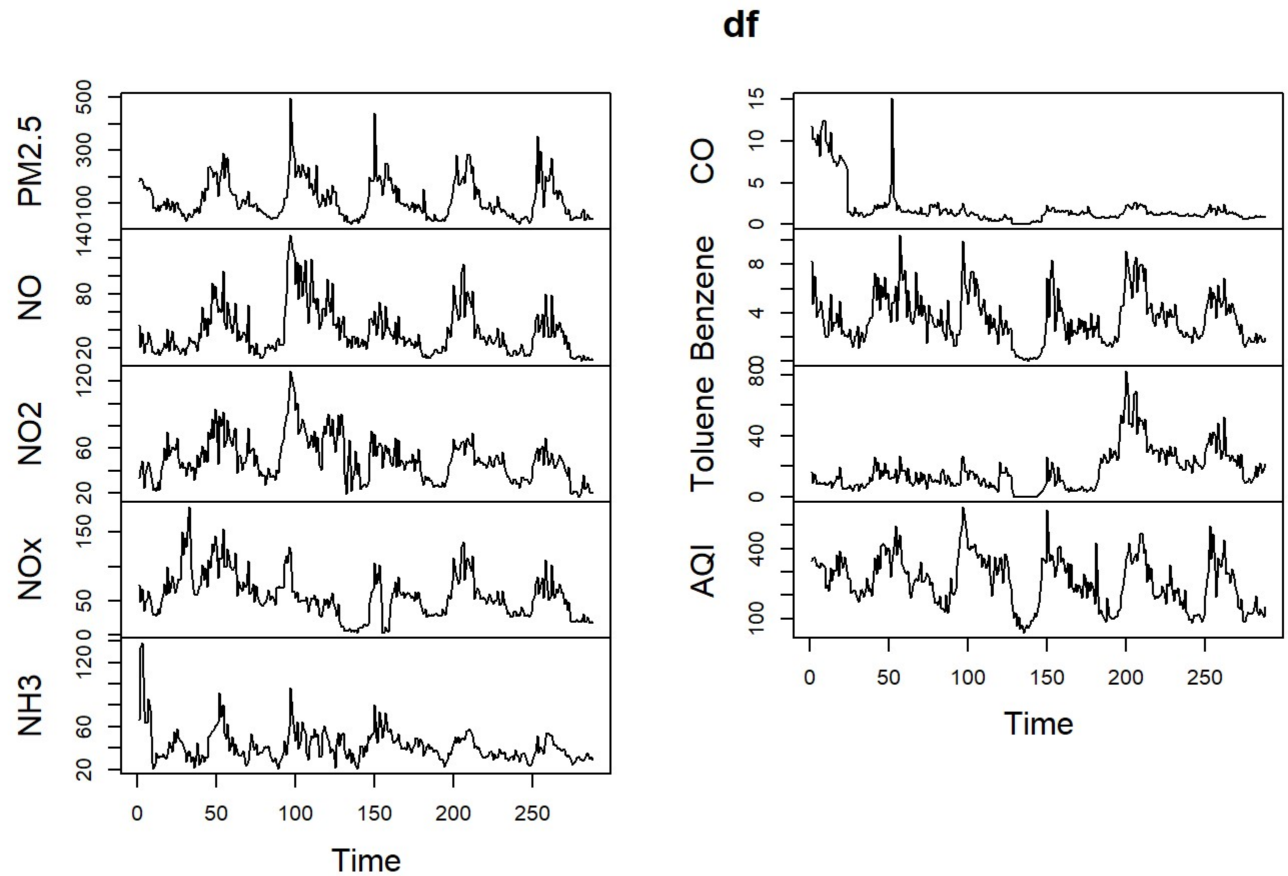
```
df$Date <- as.Date(df$Date, format = "%Y-%m-%d")
dim(df)
```

```
## [1] 288  10
```

```
rownames(df) <- df$Date
df = subset(df, select = -c(Date) )
```

Initial plot of data.

```
plot.ts(df)
```



Decomposing all individual time series of all pollutants except AQI series.

```
dfDecom.PM2.5 <- decompose(ts(df[, 1], frequency = 52))
dfDecom.NO <- decompose(ts(df[, 2], frequency = 52))
dfDecom.NO2 <- decompose(ts(df[, 3], frequency = 52))
dfDecom.NOx <- decompose(ts(df[, 4], frequency = 52))
dfDecom.NH3 <- decompose(ts(df[, 5], frequency = 52))
dfDecom.CO <- decompose(ts(df[, 6], frequency = 52))
dfDecom.Benzene <- decompose(ts(df[, 7], frequency = 52))
dfDecom.Toluene <- decompose(ts(df[, 8], frequency = 52))
```

```
print(dfDecom.PM2.5)
```

```
## $x
## Time Series:
## Start = c(1, 1)
## End = c(6, 28)
## Frequency = 52
##   [1] 184.60500 196.00571 187.65000 156.35714 162.78429 149.72286 161.09429
##   [8] 155.09429 129.80429  82.48571  85.57571  69.44857  93.74714  71.27286
##  [15]  92.57143  93.61286 110.25000  77.63857 120.33286  89.70571  73.66857
##  [22] 107.67714  66.46429  97.80571 106.39714  82.33571  73.90286  50.77000
##  [29]  62.66286  40.78429  32.75571  60.74571  63.74286  48.58857  58.94000
##  [36]  79.71571  68.17429  86.36857  62.79571  90.26571 144.83429 109.29714
##  [43] 157.71429 150.76000 236.74143 236.73286 222.77714 199.59000 216.25571
##  [50] 216.27857 129.05143 181.33714 226.91429 290.92714 194.99143 272.09143
##  [57] 271.96143 160.92143 146.91000 136.73714 131.82714 134.36714  89.56714
##  [64]  77.62143  87.71857 112.15857 116.63857  98.47571 110.90857 145.14000
##  [71]  90.76429  93.10143  87.84000  99.15714  95.83857  70.86714  81.25286
##  [78]  76.76286  67.28857  60.72286  60.72571  48.36714  52.09571  44.70286
##  [85]  42.09714  48.84286  46.37143  45.36286  56.83143  67.04286  65.76286
##  [92]  90.33286 121.87571 119.09857 153.06857 258.64857 495.15286 318.20714
##  [99] 274.34286 198.74857 222.44143 243.45143 200.69143 246.53857 230.71429
## [106] 207.45286 196.74571 238.02714 159.20571 177.61857 133.16857 139.42143
## [113] 244.81143 115.93714  87.95000 106.63429 136.82286 110.33714  90.86000
## [120] 154.83714 133.07000  96.78857 167.48286 142.72714 144.90429  94.82286
## [127]  90.08143  58.47714  58.12714  57.14286  32.95000  58.00000  31.55571
## [134]  31.48286  21.16000  35.28143  36.93000  33.62429  43.39857  26.42857
## [141]  47.17714  59.26286  44.02714  72.08571  76.91286 122.44714 198.32714
## [148] 183.89429 187.99571 438.44714 202.21143 185.72000 205.89714 173.34571
## [155] 130.51143 191.21714 251.98143 247.29857 188.26000 197.46000 148.48857
## [162] 137.04714 169.39000 103.70857 158.08286 104.86000  96.94857 102.21429
## [169]  90.97000  97.72857  94.45286  75.25000  92.28000 103.69000  68.93714
## [176]  99.39286  83.68286  98.57714  80.00143  64.88286 152.40429  68.25857
## [183]  38.87571  51.74429  43.47000  36.80857  34.58286  59.50429  39.05429
## [190]  38.74143  31.80714  37.85429  41.95000  33.33000  49.20857  57.55857
## [197]  83.38714  96.91143 140.48286 171.15429 202.76143 283.14857 173.08857
## [204] 168.86714 186.70857 201.87286 175.70143 261.53571 287.39286 282.41143
## [211] 215.72000 239.19571 116.32714 166.56714 134.56429 152.75857  90.08714
## [218]  80.32000  86.03000  82.42143  79.07571  87.09143 104.12286  72.85857
## [225]  61.77000  88.33000  85.77429 129.04857  86.87000  67.29143  78.26000
## [232]  70.20857  69.01286  53.53857  56.73286  46.77429  62.93857  42.65000
## [239]  42.49000  34.18857  28.48714  20.78143  45.11857  39.64143  46.80571
## [246]  50.56857  39.85143  25.55000  43.25571  87.52286 111.85143 140.25571
## [253] 353.36857 176.22286 297.92714 152.52143  86.82571 204.55429 184.09143
## [260] 185.57857 234.32857 271.12143 139.47714 134.74571 150.86714 115.56571
## [267] 149.54571 129.76429 106.28000  90.53571  62.04429  60.51714  77.45286
## [274]  38.48286  36.53143  44.01857  48.34857  48.89000  43.19143  46.70000
## [281]  62.76571  79.43286  38.33571  46.19571  58.95714  41.85857  41.01429
## [288]  46.22333
##
## $seasonal
## Time Series:
## Start = c(1, 1)
```

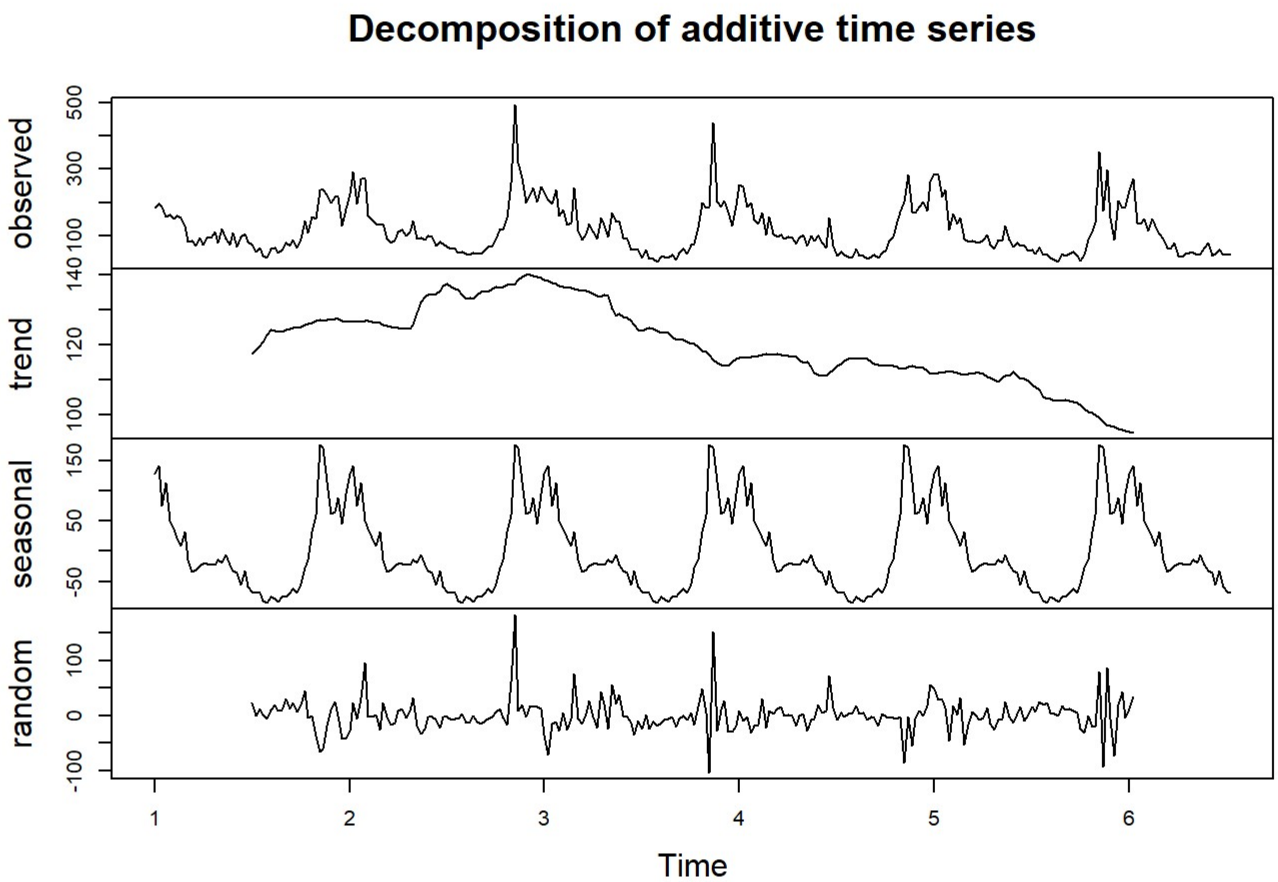
```
## End = c(6, 28)
## Frequency = 52
##   [1] 127.664773 141.312764  74.779381 112.538519  49.812777  36.396074
##   [7]  22.014305   9.314113  32.459896 -14.741619 -33.341062 -31.104709
##  [13] -24.539568 -21.146704 -21.057316 -21.924009 -22.609634 -13.494136
##  [19] -18.996739  -6.303779 -21.582185 -32.511831 -36.004342 -55.601457
##  [25] -31.615224 -58.031299 -67.303281 -67.891070 -69.274598 -81.392397
##  [31] -84.729634 -74.373315 -79.107705 -82.983356 -75.877977 -75.174790
##  [37] -68.757875 -61.568590 -68.505507 -53.581686 -26.516287 -13.310859
##  [43]  32.136985  60.965026 175.430089 171.008855 114.630240  61.725523
##  [49]  64.370218  88.798419  45.097756  94.498828 127.664773 141.312764
##  [55]  74.779381 112.538519  49.812777  36.396074  22.014305   9.314113
##  [61]  32.459896 -14.741619 -33.341062 -31.104709 -24.539568 -21.146704
##  [67] -21.057316 -21.924009 -22.609634 -13.494136 -18.996739  -6.303779
##  [73] -21.582185 -32.511831 -36.004342 -55.601457 -31.615224 -58.031299
##  [79] -67.303281 -67.891070 -69.274598 -81.392397 -84.729634 -74.373315
##  [85] -79.107705 -82.983356 -75.877977 -75.174790 -68.757875 -61.568590
##  [91] -68.505507 -53.581686 -26.516287 -13.310859  32.136985  60.965026
##  [97] 175.430089 171.008855 114.630240  61.725523  64.370218  88.798419
## [103]  45.097756  94.498828 127.664773 141.312764  74.779381 112.538519
## [109]  49.812777  36.396074  22.014305   9.314113  32.459896 -14.741619
## [115] -33.341062 -31.104709 -24.539568 -21.146704 -21.057316 -21.924009
## [121] -22.609634 -13.494136 -18.996739  -6.303779 -21.582185 -32.511831
## [127] -36.004342 -55.601457 -31.615224 -58.031299 -67.303281 -67.891070
## [133] -69.274598 -81.392397 -84.729634 -74.373315 -79.107705 -82.983356
## [139] -75.877977 -75.174790 -68.757875 -61.568590 -68.505507 -53.581686
## [145] -26.516287 -13.310859  32.136985  60.965026 175.430089 171.008855
## [151] 114.630240  61.725523  64.370218  88.798419  45.097756  94.498828
## [157] 127.664773 141.312764  74.779381 112.538519  49.812777  36.396074
## [163]  22.014305   9.314113  32.459896 -14.741619 -33.341062 -31.104709
## [169] -24.539568 -21.146704 -21.057316 -21.924009 -22.609634 -13.494136
## [175] -18.996739  -6.303779 -21.582185 -32.511831 -36.004342 -55.601457
## [181] -31.615224 -58.031299 -67.303281 -67.891070 -69.274598 -81.392397
## [187] -84.729634 -74.373315 -79.107705 -82.983356 -75.877977 -75.174790
## [193] -68.757875 -61.568590 -68.505507 -53.581686 -26.516287 -13.310859
## [199]  32.136985  60.965026 175.430089 171.008855 114.630240  61.725523
## [205]  64.370218  88.798419  45.097756  94.498828 127.664773 141.312764
## [211]  74.779381 112.538519  49.812777  36.396074  22.014305   9.314113
## [217]  32.459896 -14.741619 -33.341062 -31.104709 -24.539568 -21.146704
## [223] -21.057316 -21.924009 -22.609634 -13.494136 -18.996739  -6.303779
## [229] -21.582185 -32.511831 -36.004342 -55.601457 -31.615224 -58.031299
## [235] -67.303281 -67.891070 -69.274598 -81.392397 -84.729634 -74.373315
## [241] -79.107705 -82.983356 -75.877977 -75.174790 -68.757875 -61.568590
## [247] -68.505507 -53.581686 -26.516287 -13.310859  32.136985  60.965026
## [253] 175.430089 171.008855 114.630240  61.725523  64.370218  88.798419
## [259]  45.097756  94.498828 127.664773 141.312764  74.779381 112.538519
## [265]  49.812777  36.396074  22.014305   9.314113  32.459896 -14.741619
## [271] -33.341062 -31.104709 -24.539568 -21.146704 -21.057316 -21.924009
## [277] -22.609634 -13.494136 -18.996739  -6.303779 -21.582185 -32.511831
## [283] -36.004342 -55.601457 -31.615224 -58.031299 -67.303281 -67.891070
##
```

```
## $trend
## Time Series:
## Start = c(1, 1)
## End = c(6, 28)
## Frequency = 52
##   [1]        NA        NA        NA        NA        NA        NA        NA
##   [8]        NA        NA        NA        NA        NA        NA        NA
##  [15]        NA        NA        NA        NA        NA        NA        NA
##  [22]        NA        NA        NA        NA        NA 117.43933 118.75886
##  [29] 119.74216 120.92558 123.08819 124.24565 124.21694 123.90404 123.74698
##  [36] 124.26529 124.80253 124.91949 124.94011 125.27527 125.89982 126.17799
##  [43] 126.23109 126.88647 127.25121 126.99955 127.16846 127.22280 127.42332
##  [50] 127.44674 126.94595 126.65059 126.53341 126.56551 126.64258 126.69687
##  [57] 126.95574 126.98745 126.62505 126.41937 126.30096 125.84979 125.41041
##  [64] 125.11552 124.95823 124.98740 124.76729 124.64078 124.69036 125.68308
##  [71] 129.20519 132.47332 133.75255 134.24029 134.29168 134.61243 135.56255
##  [78] 136.87834 137.54181 136.77571 135.98995 135.67927 134.26754 133.34390
##  [85] 133.37232 133.26600 134.37820 135.28738 135.09462 135.35804 136.10916
##  [92] 136.56380 136.29842 136.59249 137.34751 137.09569 137.36845 138.58330
##  [99] 139.60916 140.11618 140.01915 139.84466 139.50316 139.09214 138.57331
## [106] 138.21695 137.91029 137.46746 137.00765 136.61960 136.47933 136.28331
## [113] 136.10839 135.89775 135.62286 135.45522 135.17141 134.78696 134.17918
## [120] 133.77904 134.24641 133.96280 130.29058 128.49330 128.95588 128.13703
## [127] 127.85268 127.01951 125.67060 124.46386 124.13641 124.72404 125.02558
## [134] 124.55391 124.06080 123.56764 123.52581 123.53070 122.35338 121.41294
## [141] 121.39295 121.43698 120.95359 120.39146 120.30477 119.57405 118.41658
## [148] 118.09073 117.20953 115.84530 114.83996 114.28739 114.22657 114.19124
## [155] 115.15934 116.17273 116.33659 116.33342 116.38783 116.55360 116.73387
## [162] 117.09585 117.34919 117.41882 117.35657 117.35497 117.41457 117.11496
## [169] 116.91543 116.82556 116.74813 116.56485 115.76312 115.08442 115.10390
## [176] 113.75262 111.97934 111.53727 111.19071 111.28051 111.98933 113.09999
## [183] 114.11662 114.79474 115.39640 116.06174 116.15380 116.12841 116.07739
## [190] 116.21416 116.03199 115.14223 114.80128 114.50598 114.20129 113.98464
## [197] 113.97534 114.04533 113.72897 113.28791 113.30212 113.74916 114.06496
## [204] 113.79478 113.47721 113.51168 112.76104 111.81766 111.84783 111.97174
## [211] 112.11115 112.35452 112.48672 112.31933 111.97430 111.70000 111.65530
## [218] 111.80048 111.86435 112.07680 112.15258 111.75484 111.06118 110.58503
## [225] 110.21945 109.64705 110.79809 111.21810 111.39034 112.43354 111.31596
## [232] 110.38133 110.48779 109.83810 108.59751 107.97872 107.13706 105.39963
## [239] 104.72742 104.56913 104.22279 104.14574 104.08034 104.33427 104.20187
## [246] 103.76062 103.53440 103.05140 101.93409 101.00687 100.60051 100.09223
## [253]  99.30354  98.10228  97.07870  96.96367  96.69652  96.08174  95.75416
## [260]  95.54516  95.28172  95.12528        NA        NA        NA        NA
## [267]        NA        NA        NA        NA        NA        NA        NA
## [274]        NA        NA        NA        NA        NA        NA        NA
## [281]        NA        NA        NA        NA        NA        NA        NA
## [288]        NA
##
## $random
## Time Series:
## Start = c(1, 1)
```

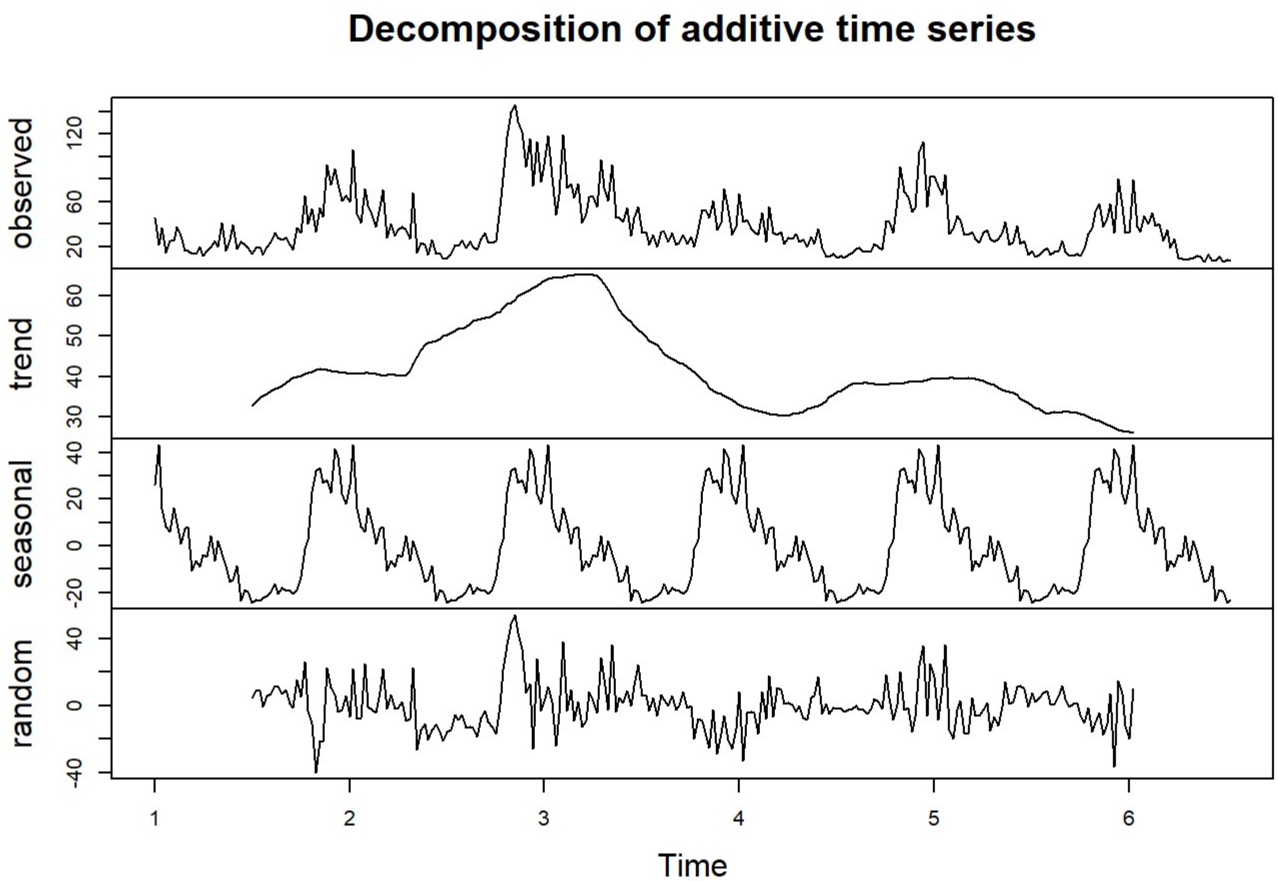
```
## End = c(6, 28)
## Frequency = 52
##   [1]           NA           NA           NA           NA           NA
##   [6]           NA           NA           NA           NA           NA
##  [11]           NA           NA           NA           NA           NA
##  [16]           NA           NA           NA           NA           NA
##  [21]           NA           NA           NA           NA           NA
##  [26]           NA  23.76680395  -0.09778946  12.19529846   1.25110615
##  [31]  -5.60283891  10.87338362  18.63362538   7.66788911  11.07099901
##  [36]  30.62521604  12.12963362  23.01766933   6.36111164  18.57212538
##  [41]  45.45075175  -3.56999275  -0.65378396 -37.09149550 -65.93986913
##  [46] -61.27554495 -19.02155869  10.64167483  24.46217208   0.03340835
##  [51] -42.99227572 -39.81227572 -27.28389385  23.04887080  -6.43053465
##  [56]  32.85604159  95.19290972  -2.46209028  -1.72936020   1.00366178
##  [61] -26.93371459  23.25896741  -2.50220704 -16.38938424 -12.70008822
##  [66]   8.31787194  12.92859310  -4.24106006   8.82784791  32.95105876
##  [71] -19.44416789 -33.06811707 -24.33036981  -2.57131418  -2.44876267
##  [76]  -8.14383135 -22.69447352  -2.08418163  -2.94996116  -8.16178671
##  [81]  -5.98963286  -5.91973176   2.55780670 -14.26772902 -12.16747352
##  [86]  -1.43978946 -12.12879495 -14.74972902  -9.50531143  -6.74658891
##  [91]  -1.84079770   7.35073802  12.09358142  -4.18305594 -16.41592682
##  [96]  60.58785890 182.35432043   8.61499076  20.10345505  -3.09313286
## [101]  18.05206219  14.80835340  16.09051274  12.94760065 -35.52379770
## [106] -72.07685722 -15.94395498 -11.97883479 -27.61471391   4.60289598
## [111] -25.32506075  -6.17599481  76.24313980  -5.21898589 -14.33179495
## [116]   2.28377510  26.19101068  -3.30311707 -22.26186020  42.98211302
## [121]  21.43321879 -23.68009509  56.18901892  20.53762469  37.53059172
## [126]  -0.80234440  -1.76690828 -12.94090553 -35.92823726  -9.28970360
## [131] -23.88313424   1.16703197 -24.19526473 -11.67866034 -18.17116308
## [136] -13.91289385  -7.48810539  -6.92305869  -3.07683067 -19.80957792
## [141]  -5.45793506  -0.60553121  -8.42093506   5.27594406 -16.87562187
## [146]  16.18394955  47.77357868   4.83853197 -104.64390759 151.59298527
## [151] -27.25877023   9.70708692  27.30035890 -29.64394056 -29.74566858
## [156] -19.45441858   7.98006219 -10.34761272  -2.90721047 -31.63211775
## [161] -18.05807929 -16.44478259  30.02650519 -23.02436020   8.26639530
## [166]   2.24664598  12.87505944  16.20403609  -1.40585745   2.04971260
## [171]  -1.23795910 -19.39084028  -0.87348451   2.09971260 -27.17001954
## [176]  -8.05598795  -6.71429839  19.55170780   4.81505601   9.20380601
## [181]  72.03018307  13.18988431  -7.93762599   4.84061714  -2.65180319
## [186]   2.13922428   3.15868582  17.74919406   2.08460065   5.51062263
## [191]  -8.34687187  -2.11314935  -4.09340209 -19.60738561   3.51278747
## [196]  -2.84438561  -4.07191308  -3.82304220  -5.38309715  -3.09865209
## [201] -85.97077572  -1.60944605 -55.60662737  -6.65316034   8.86114186
## [206]  -0.43723726  17.84262813  55.21922153  47.88025450  29.12692025
## [211]  28.82946535  14.30267620 -45.97235127  17.85174214   0.57568101
## [216]  31.74445848 -54.02805525 -16.73886226   7.50670780   1.44933829
## [221]  -8.53729976  -3.51670223  14.11899145 -15.80244742 -25.83981693
## [226]  -7.82291102  -6.02706624  24.13424557  -2.93815828 -12.63028396
## [231]   2.94838019  15.42869612  -9.85970704   1.73176618  15.43862401
## [236]   6.68663362  25.07610890  18.64276824  22.49221604   3.99275175
## [241]   3.37205944  -0.38095704  16.91620505  10.48194681  11.36172153
```

```
## [246]    8.37654296    4.82254021  -23.91971528  -32.16209165   -0.17315209
## [251]  -20.88606418  -20.80153671   78.63493857  -92.88827847   86.21820780
## [256]   -6.16776198  -74.24102847   19.67412263   43.23950999   -4.46542132
## [261]   11.38208142   34.68338545            NA            NA            NA
## [266]            NA            NA            NA            NA            NA
## [271]            NA            NA            NA            NA            NA
## [276]            NA            NA            NA            NA            NA
## [281]            NA            NA            NA            NA            NA
## [286]            NA            NA            NA
##
## $figure
##  [1] 127.664773 141.312764  74.779381 112.538519  49.812777  36.396074
##  [7]  22.014305   9.314113  32.459896 -14.741619 -33.341062 -31.104709
## [13] -24.539568 -21.146704 -21.057316 -21.924009 -22.609634 -13.494136
## [19] -18.996739  -6.303779 -21.582185 -32.511831 -36.004342 -55.601457
## [25] -31.615224 -58.031299 -67.303281 -67.891070 -69.274598 -81.392397
## [31] -84.729634 -74.373315 -79.107705 -82.983356 -75.877977 -75.174790
## [37] -68.757875 -61.568590 -68.505507 -53.581686 -26.516287 -13.310859
## [43]  32.136985  60.965026 175.430089 171.008855 114.630240  61.725523
## [49]  64.370218  88.798419  45.097756  94.498828
##
## $type
## [1] "additive"
##
## attr(,"class")
## [1] "decomposed.ts"
```

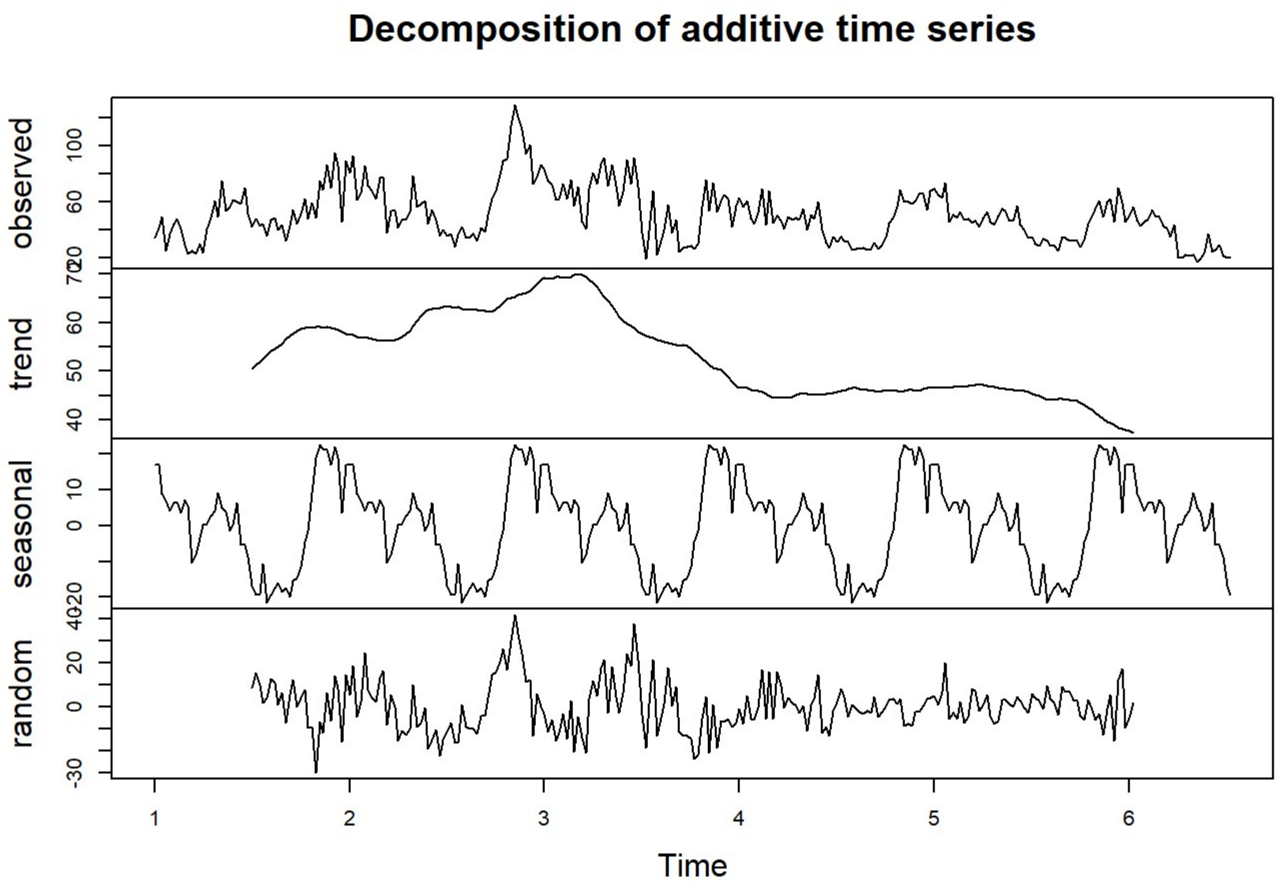
```
plot(dfDecom.PM2.5)
```



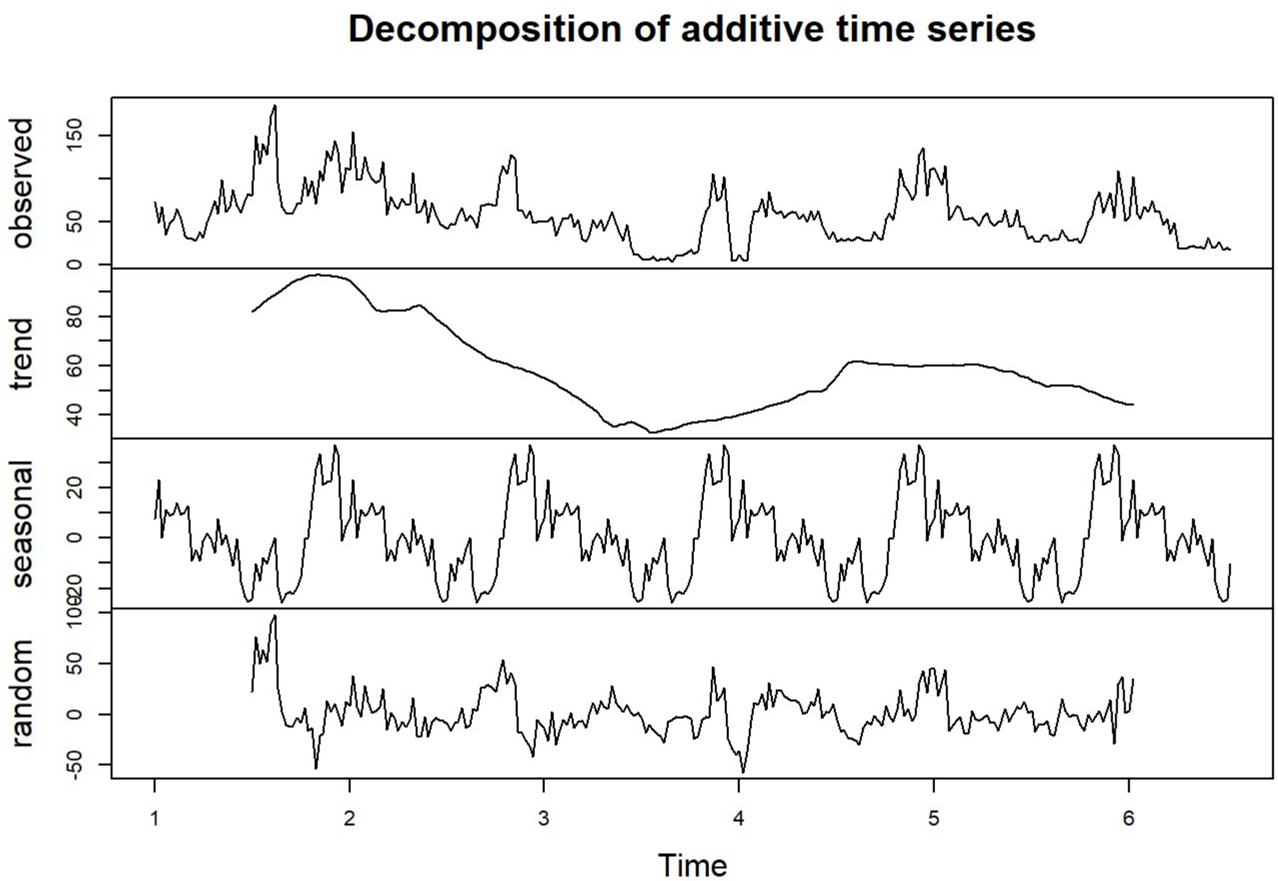

```
plot(dfDecom.NO)
```



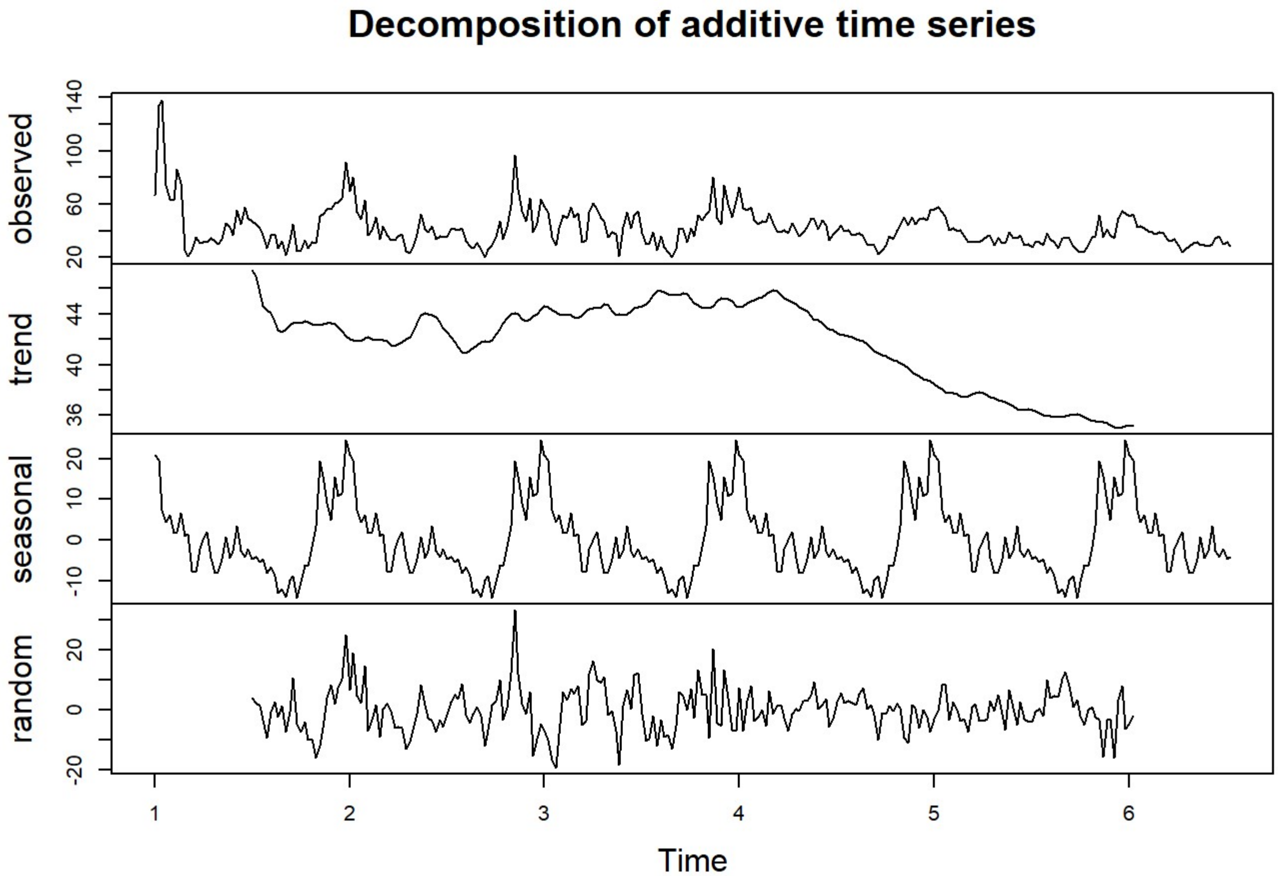

```
plot(dfDecom.NO2)
```



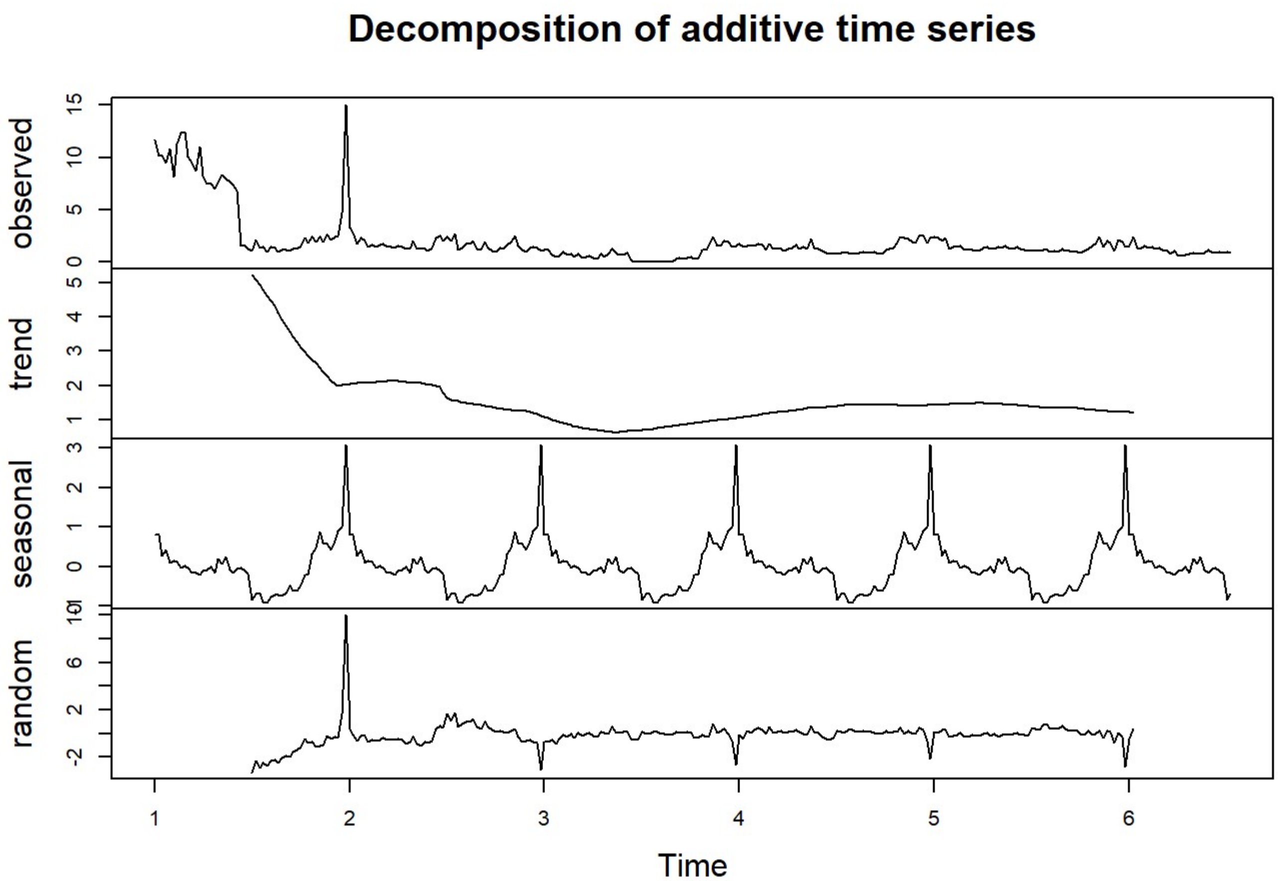

```
plot(dfDecom.NOx)
```



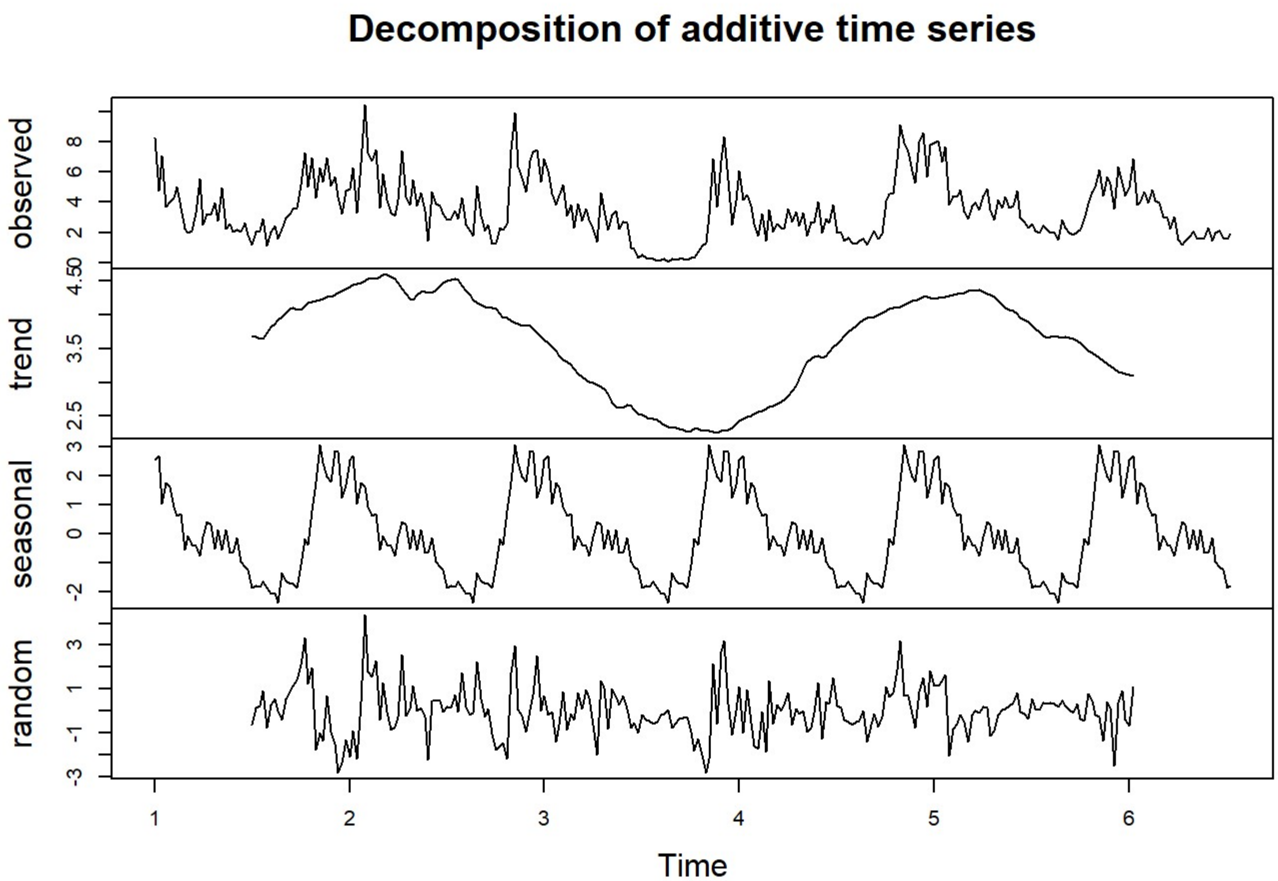

```
plot(dfDecom.NH3)
```



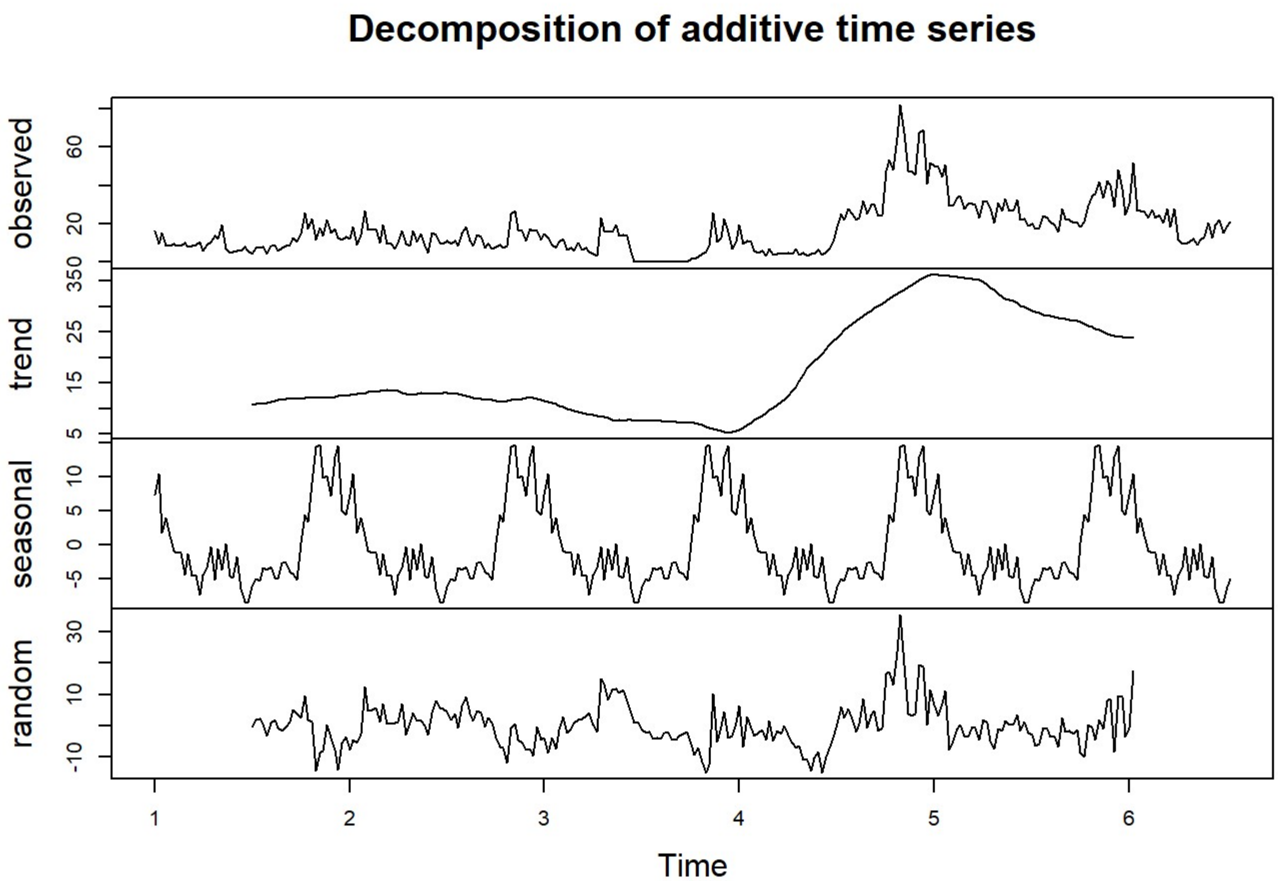

```
plot(dfDecom.CO)
```




```
plot(dfDecom.Benzene)
```

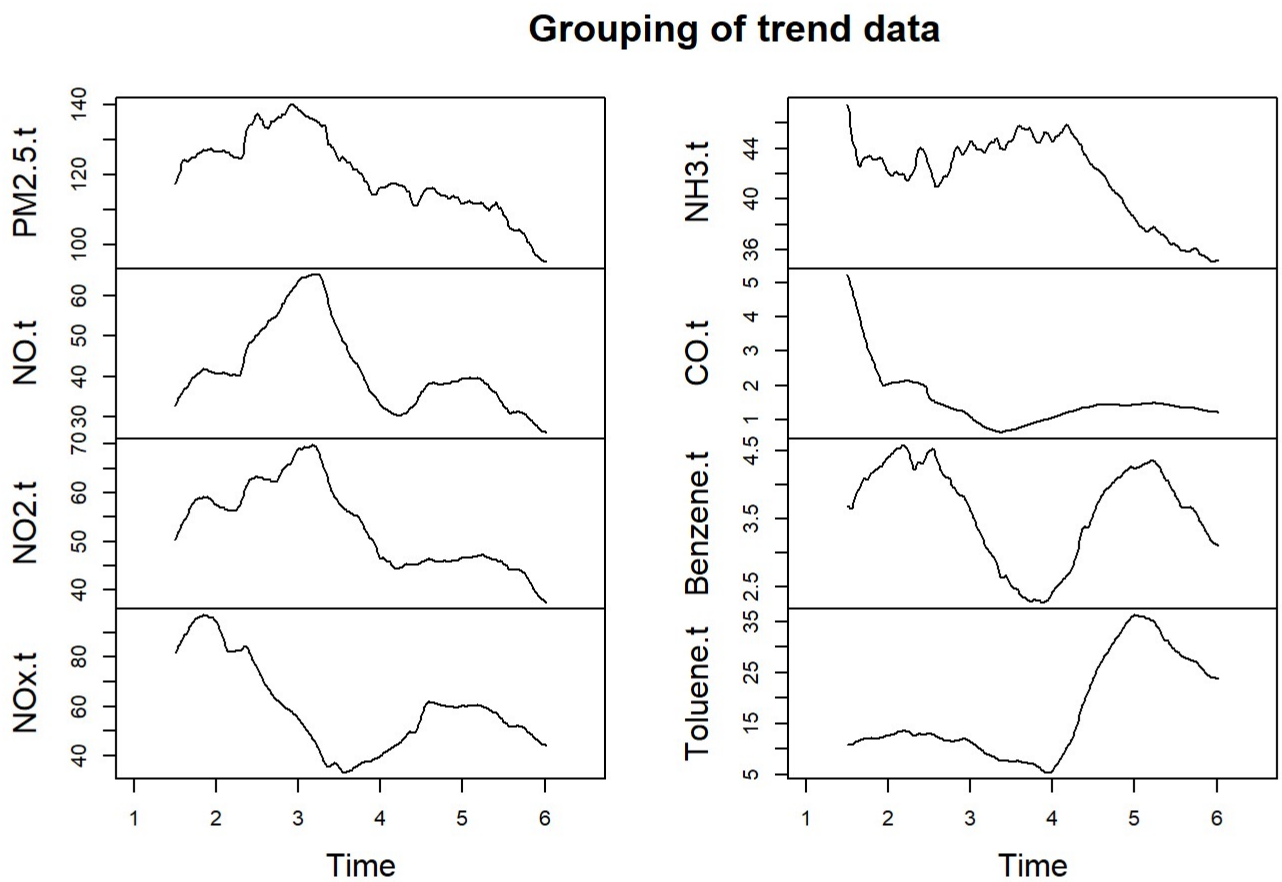



```
plot(dfDecom.Toluene)
```

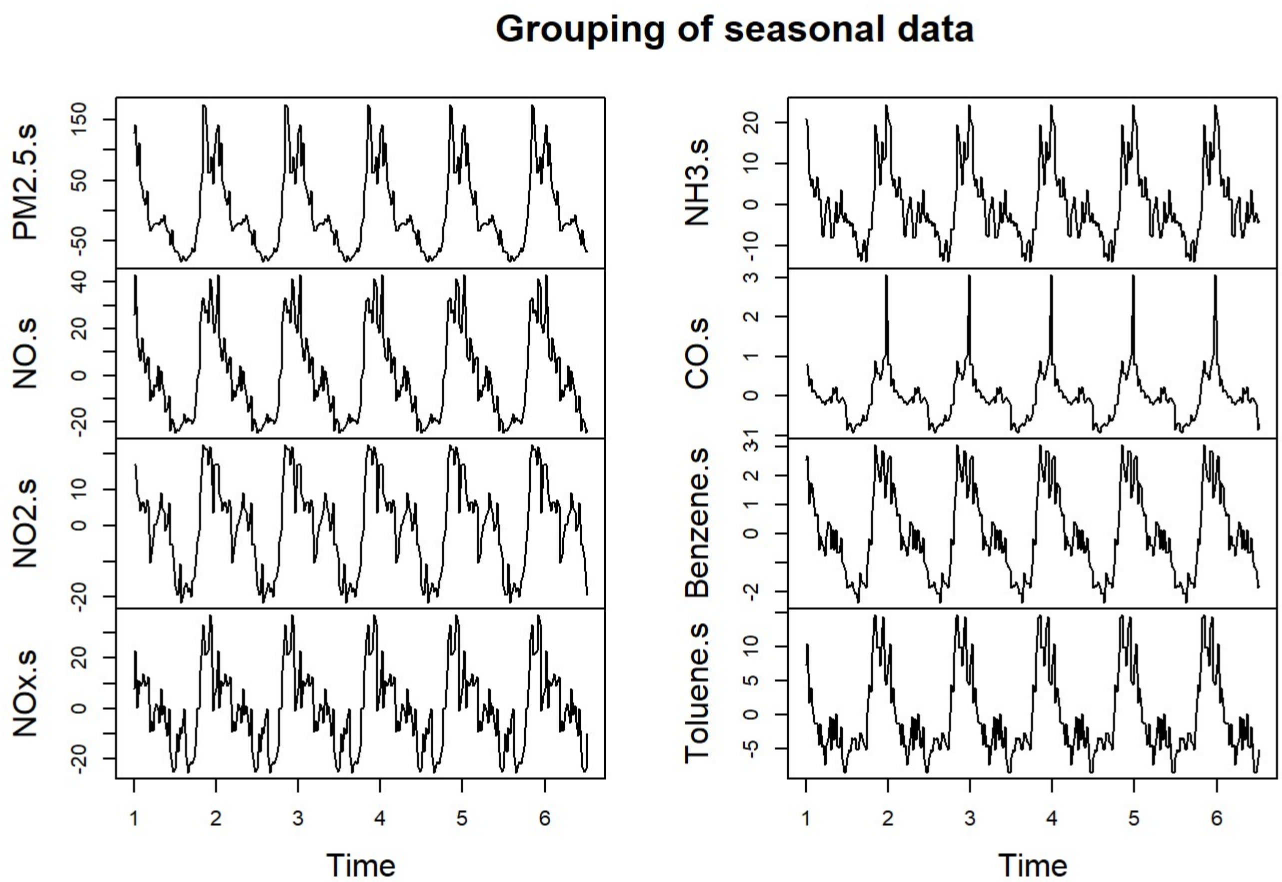


Grouping up of trend, seasonal and random components of decomposed time series.

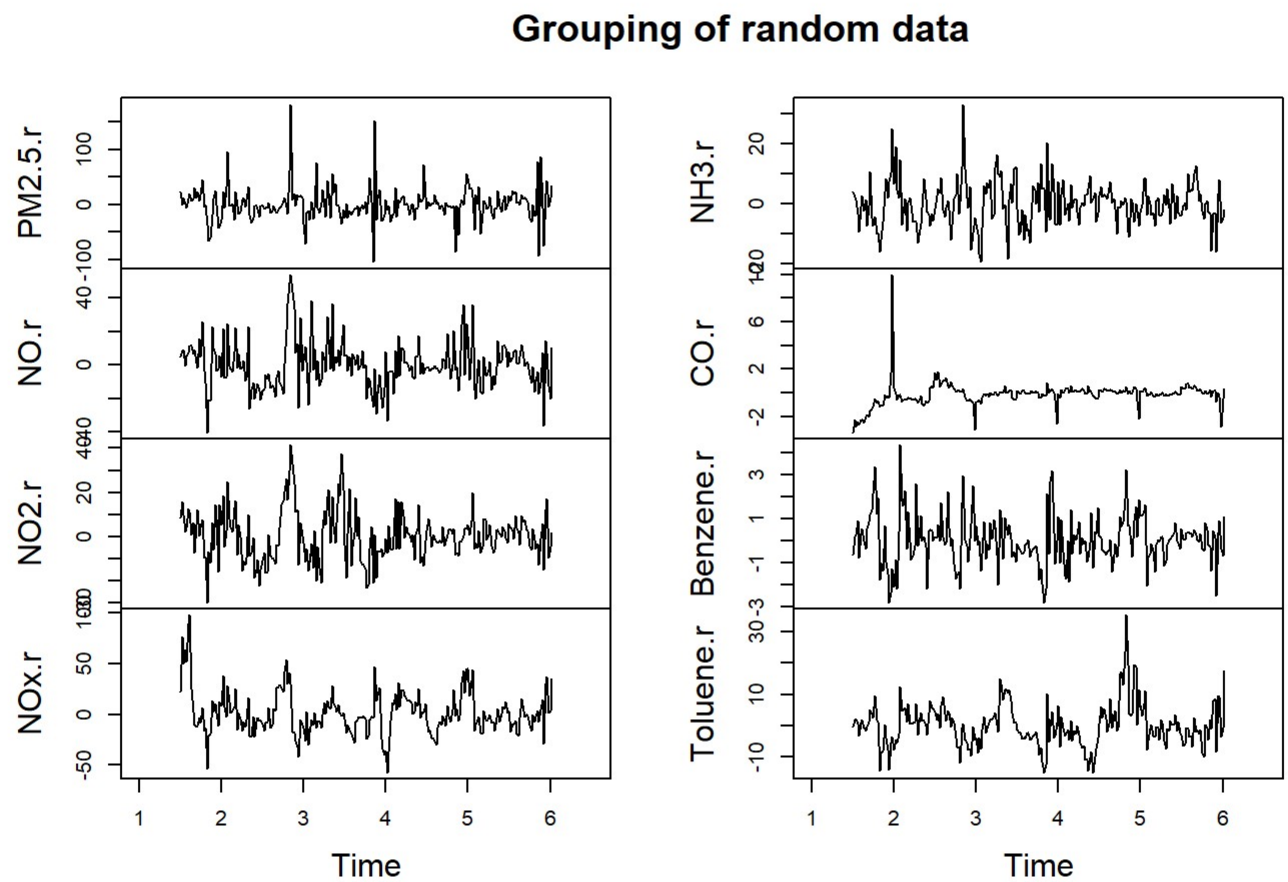
```
dfDecom.trend <- data.frame(PM2.5.t = dfDecom.PM2.5$trend,
                            NO.t = dfDecom.NO$trend,
                            NO2.t = dfDecom.NO2$trend,
                            NOx.t = dfDecom.NOx$trend,
                            NH3.t = dfDecom.NH3$trend,
                            CO.t = dfDecom.CO$trend,
                            Benzene.t = dfDecom.Benzene$trend,
                            Toluene.t = dfDecom.Toluene$trend

)

plot.ts(dfDecom.trend, main="Grouping of trend data")
```




```
dfDecom.seasonal <- data.frame(PM2.5.s = dfDecom.PM2.5$seasonal,
                               NO.s = dfDecom.NO$seasonal,
                               NO2.s = dfDecom.NO2$seasonal,
                               NOx.s = dfDecom.NOx$seasonal,
                               NH3.s = dfDecom.NH3$seasonal,
                               CO.s = dfDecom.CO$seasonal,
                               Benzene.s = dfDecom.Benzene$seasonal,
                               Toluene.s = dfDecom.Toluene$seasonal


)

plot.ts(dfDecom.seasonal, main="Grouping of seasonal data")
```




```
dfDecom.random <- data.frame(PM2.5.r = dfDecom.PM2.5$random,
                             NO.r = dfDecom.NO$random,
                             NO2.r = dfDecom.NO2$random,
                             NOx.r = dfDecom.NOx$random,
                             NH3.r = dfDecom.NH3$random,
                             CO.r = dfDecom.CO$random,
                             Benzene.r = dfDecom.Benzene$random,
                             Toluene.r = dfDecom.Toluene$random


)

plot.ts(dfDecom.random, main="Grouping of random data")
```



Decomposition of time series produces NaNs. So we need to remove rows of data corresponding to NaNs.

```
st <- 27
en <- 262
num <- en - st + 1
ratio <- 0.7
mid <- as.numeric(num*ratio)
df <- df[st:en, ]
df_aqi <- df[, c("AQI")]
df = subset(df, select = -c(AQI) )
df_aqi_train <- df_aqi[1:mid]
df_aqi_test <- df_aqi[(mid+1):num]
df_train <- df[1:mid, ]
df_test <- df[(mid+1):num, ]
dfDecom.trend <- dfDecom.trend[st:en, ]
dfDecom.seasonal <- dfDecom.seasonal[st:en, ]
dfDecom.random <- dfDecom.random[st:en, ]
print(head(df_aqi_train))
```

```
## [1] 217.8571 223.2857 204.4286 178.4286 171.1429 222.4286
```

```
print(head(df_aqi_test))
```

```
## [1] 103.2857 108.5714 107.1429 131.2857 140.1429 199.1429
```

```
y <- df_aqi_train
y0 <- df_aqi_test
```

Original LM

This is regression model on base data.

```
lm.1 = lm(df_aqi_train ~ df_train[, 1] + df_train[, 2] + df_train[, 3] + df_train[,
4] + df_train[, 5] +
            df_train[, 6] + df_train[, 7] + df_train[, 8])
yhat.1 = predict(lm.1, data.frame(df_train[, 1:8]))

train.err.1 = mean((y-yhat.1)^2)
y0hat.1 = predict(lm.1, data.frame(df_test[, 1:8]))
```

```
## Warning: 'newdata' had 70 rows but variables found have 165 rows
```

```
test.err.1 = mean((y0-y0hat.1)^2)
```

```
## Warning in y0 - y0hat.1: longer object length is not a multiple of shorter
## object length
```

```
print(summary(lm.1))
```

```
##
## Call:
## lm(formula = df_aqi_train ~ df_train[, 1] + df_train[, 2] + df_train[,
##     3] + df_train[, 4] + df_train[, 5] + df_train[, 6] + df_train[,
##     7] + df_train[, 8])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -162.115  -20.504   -0.627   21.106  145.763
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   98.42864   13.51399   7.283 1.51e-11 ***
## df_train[, 1]  1.14984    0.08175  14.066  < 2e-16 ***
## df_train[, 2]  0.10748    0.22488   0.478 0.633347
## df_train[, 3]  0.69751    0.30567   2.282 0.023845 *
## df_train[, 4]  0.40967    0.10705   3.827 0.000187 ***
## df_train[, 5] -1.17214    0.36740  -3.190 0.001718 **
## df_train[, 6]  3.31026    2.98399   1.109 0.268989
## df_train[, 7]  3.64903    2.77912   1.313 0.191106
## df_train[, 8] -0.56897    0.58101  -0.979 0.328959
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 39.61 on 156 degrees of freedom
## Multiple R-squared:  0.8816, Adjusted R-squared:  0.8755
## F-statistic: 145.2 on 8 and 156 DF,  p-value: < 2.2e-16
```

end Original LM

PCA on random part

```
h2o.no_progress()  # turn off progress bars
h2o.init(max_mem_size = "5g")  # connect to H2O instance
```

```
##  Connection successful!
##
## R is connected to the H2O cluster:
##     H2O cluster uptime:         3 days 17 hours
##     H2O cluster timezone:       Asia/Kolkata
##     H2O data parsing timezone:  UTC
##     H2O cluster version:        3.32.0.1
##     H2O cluster version age:    6 months and 15 days !!!
##     H2O cluster name:           H2O_started_from_R_vishw_pcl701
##     H2O cluster total nodes:    1
##     H2O cluster total memory:   4.98 GB
##     H2O cluster total cores:    12
##     H2O cluster allowed cores:  12
##     H2O cluster healthy:        TRUE
##     H2O Connection ip:          localhost
##     H2O Connection port:        54321
##     H2O Connection proxy:       NA
##     H2O Internal Security:      FALSE
##     H2O API Extensions:         Amazon S3, Algos, AutoML, Core V3, TargetEncoder,
Core V4
##     R Version:                  R version 3.6.1 (2019-07-05)
```

```
## Warning in h2o.clusterInfo():
## Your H2O cluster version is too old (6 months and 15 days)!
## Please download and install the latest version from http://h2o.ai/download/
```

```
dfDecom.random.h2o <- as.h2o(dfDecom.random)
```

```
# run basic pca on random component
pca_random <- h2o.prcomp(
  training_frame = dfDecom.random.h2o,
  pca_method = "GramSVD",
  k = ncol(dfDecom.random.h2o),
  transform = "STANDARDIZE",
  impute_missing = TRUE,
  max_runtime_secs = 1000
)
```

```
print(summary(pca_random))
```

```
## Model Details:
## ==============
##
## H2ODimReductionModel: pca
## Model Key:  PCA_model_R_1618906539228_122
## Importance of components:
##                             pc1      pc2      pc3      pc4      pc5      pc6
## Standard deviation     1.781358 1.165017 1.059244 0.860642 0.752150 0.675197
## Proportion of Variance 0.396655 0.169658 0.140250 0.092588 0.070716 0.056986
## Cumulative Proportion  0.396655 0.566313 0.706563 0.799151 0.869867 0.926853
##                             pc7      pc8
## Standard deviation     0.555054 0.526391
## Proportion of Variance 0.038511 0.034636
## Cumulative Proportion  0.965364 1.000000
##
## H2ODimReductionMetrics: pca
##
## No model metrics available for PCA
##
##
##
## Scoring History for GramSVD:
##             timestamp   duration iterations
## 1 2021-04-24 07:08:15  0.003 sec          0
##
## NULL
```

```
print(pca_random@model$importance)
```

```
## Importance of components:
##                             pc1      pc2      pc3      pc4      pc5      pc6
## Standard deviation     1.781358 1.165017 1.059244 0.860642 0.752150 0.675197
## Proportion of Variance 0.396655 0.169658 0.140250 0.092588 0.070716 0.056986
## Cumulative Proportion  0.396655 0.566313 0.706563 0.799151 0.869867 0.926853
##                             pc7      pc8
## Standard deviation     0.555054 0.526391
## Proportion of Variance 0.038511 0.034636
## Cumulative Proportion  0.965364 1.000000
```

```
pca_random_pred <- h2o.predict(pca_random, dfDecom.random.h2o)
```

end PCA on random part

PCA on original data

```
print(head(df))
```

```
##                PM2.5       NO      NO2       NOx      NH3       CO  Benzene
## 2015-07-05 73.90286 13.54571 42.29143  80.54143 46.94143 1.062857 1.208571
## 2015-07-12 50.77000 19.41429 47.53143 149.97429 45.21000 2.110000 2.052857
## 2015-07-19 62.66286 20.15000 42.81000 117.79429 41.75286 1.384286 2.098571
## 2015-07-26 40.78429 12.50286 43.93857 141.57857 36.80714 1.474286 2.921429
## 2015-08-02 32.75571 20.38714 35.87286 128.27714 27.17857 0.990000 1.124286
## 2015-08-09 60.74571 24.30143 47.20286 173.10857 36.79714 1.440000 2.065714
##             Toluene
## 2015-07-05 4.047143
## 2015-07-12 7.458571
## 2015-07-19 7.642857
## 2015-07-26 7.975714
## 2015-08-02 4.117143
## 2015-08-09 8.697143
```

```
dfDecom.df.h2o <- as.h2o(df)
```

```
pca_orig <- h2o.prcomp(
  training_frame = dfDecom.df.h2o,
  pca_method = "GramSVD",
  k = ncol(dfDecom.random.h2o),
  transform = "STANDARDIZE",
  impute_missing = TRUE,
  max_runtime_secs = 1000
)
```

```
print(summary(pca_orig))
```

```
## Model Details:
## ==============
##
## H2ODimReductionModel: pca
## Model Key:  PCA_model_R_1618906539228_123
## Importance of components:
##                             pc1      pc2      pc3      pc4      pc5      pc6
## Standard deviation     2.111503 1.103672 0.944229 0.773621 0.566483 0.495016
## Proportion of Variance 0.557305 0.152262 0.111446 0.074811 0.040113 0.030630
## Cumulative Proportion  0.557305 0.709567 0.821013 0.895824 0.935937 0.966567
##                             pc7      pc8
## Standard deviation     0.383648 0.346811
## Proportion of Variance 0.018398 0.015035
## Cumulative Proportion  0.984965 1.000000
##
## H2ODimReductionMetrics: pca
##
## No model metrics available for PCA
##
##
##
## Scoring History for GramSVD:
##             timestamp   duration iterations
## 1 2021-04-24 07:08:19  0.001 sec          0
##
## NULL
```

```
print(pca_orig@model$importance)
```

```
## Importance of components:
##                             pc1      pc2      pc3      pc4      pc5      pc6
## Standard deviation     2.111503 1.103672 0.944229 0.773621 0.566483 0.495016
## Proportion of Variance 0.557305 0.152262 0.111446 0.074811 0.040113 0.030630
## Cumulative Proportion  0.557305 0.709567 0.821013 0.895824 0.935937 0.966567
##                             pc7      pc8
## Standard deviation     0.383648 0.346811
## Proportion of Variance 0.018398 0.015035
## Cumulative Proportion  0.984965 1.000000
```

```
pca_orig_pred <- h2o.predict(pca_orig, dfDecom.df.h2o)
```

end PCA on original data

GLRM on original data

dimension = 6

```
dfDecom.df.h2o <- as.h2o(df)
```

```
glrm_orig6 <- h2o.glrm(
  training_frame = dfDecom.df.h2o,
  k = 6,
  loss = "Quadratic",
  regularization_x = "None",
  regularization_y = "None",
  transform = "STANDARDIZE",
  max_iterations = 2000,
  seed = 123
)
```

```
print(summary(glrm_orig6))
```

```
## Model Details:
## ==============
##
## H2ODimReductionModel: glrm
## Model Key:  GLRM_model_R_1618906539228_124
## Model Summary:
##   number_of_iterations final_step_size final_objective_value
## 1                 1000         0.00131             106.74490
##
## H2ODimReductionMetrics: glrm
## ** Reported on training data. **
##
## Sum of Squared Error (Numeric):  106.7449
## Misclassification Error (Categorical):  0
## Number of Numeric Entries:  1888
## Number of Categorical Entries:  0
##
##
##
## Scoring History:
##             timestamp   duration iterations step_size objective
## 1 2021-04-24 07:08:24  0.017 sec          0   0.66667 157.86765
## 2 2021-04-24 07:08:24  0.018 sec          1   0.44444 157.86765
## 3 2021-04-24 07:08:24  0.018 sec          2   0.22222 157.86765
## 4 2021-04-24 07:08:24  0.019 sec          3   0.07407 157.86765
## 5 2021-04-24 07:08:24  0.019 sec          4   0.01852 157.86765
##
## ---
##                timestamp   duration iterations step_size objective
## 995  2021-04-24 07:08:25  0.681 sec        994   0.00103 106.75160
## 996  2021-04-24 07:08:25  0.682 sec        995   0.00108 106.74996
## 997  2021-04-24 07:08:25  0.682 sec        996   0.00113 106.74860
## 998  2021-04-24 07:08:25  0.683 sec        997   0.00119 106.74732
## 999  2021-04-24 07:08:25  0.684 sec        998   0.00125 106.74609
## 1000 2021-04-24 07:08:25  0.685 sec        999   0.00131 106.74490
##
## NULL
```

```
print(glrm_orig6@model$importance)
```

```
## Importance of components:
##                             pc1      pc2      pc3      pc4      pc5      pc6
## Standard deviation     2.111570 1.103672 0.944351 0.748049 0.539703 0.354577
## Proportion of Variance 0.557341 0.152262 0.111475 0.069947 0.036410 0.015716
## Cumulative Proportion  0.557341 0.709602 0.821077 0.891024 0.927434 0.943150
```

```
glrm_orig6_pred <- h2o.predict(glrm_orig6, dfDecom.df.h2o)
```

dimension = 5

```
dfDecom.df.h2o <- as.h2o(df)
```

```
glrm_orig5 <- h2o.glrm(
  training_frame = dfDecom.df.h2o,
  k = 5,
  loss = "Quadratic",
  regularization_x = "None",
  regularization_y = "None",
  transform = "STANDARDIZE",
  max_iterations = 2000,
  seed = 123
)
```

```
print(summary(glrm_orig5))
```

```
## Model Details:
## ==============
##
## H2ODimReductionModel: glrm
## Model Key:  GLRM_model_R_1618906539228_126
## Model Summary:
##   number_of_iterations final_step_size final_objective_value
## 1                 1000         0.04330             120.43872
##
## H2ODimReductionMetrics: glrm
## ** Reported on training data. **
##
## Sum of Squared Error (Numeric):  120.4387
## Misclassification Error (Categorical):  0
## Number of Numeric Entries:  1888
## Number of Categorical Entries:  0
##
##
##
## Scoring History:
##             timestamp   duration iterations step_size  objective
## 1 2021-04-24 07:08:30  0.015 sec          0   0.66667 1475.73545
## 2 2021-04-24 07:08:30  0.016 sec          1   0.44444 1475.73545
## 3 2021-04-24 07:08:30  0.016 sec          2   0.22222 1475.73545
## 4 2021-04-24 07:08:30  0.017 sec          3   0.07407 1475.73545
## 5 2021-04-24 07:08:30  0.017 sec          4   0.01852 1475.73545
##
## ---
##                timestamp   duration iterations step_size objective
## 995  2021-04-24 07:08:31  0.585 sec        994   0.05344 120.43872
## 996  2021-04-24 07:08:31  0.586 sec        995   0.05611 120.43872
## 997  2021-04-24 07:08:31  0.586 sec        996   0.05891 120.43872
## 998  2021-04-24 07:08:31  0.587 sec        997   0.03928 120.43872
## 999  2021-04-24 07:08:31  0.588 sec        998   0.04124 120.43872
## 1000 2021-04-24 07:08:31  0.588 sec        999   0.04330 120.43872
##
## NULL
```

```
print(glrm_orig5@model$importance)
```

```
## Importance of components:
##                             pc1      pc2      pc3      pc4      pc5
## Standard deviation     2.111503 1.103672 0.944229 0.773621 0.566483
## Proportion of Variance 0.557305 0.152262 0.111446 0.074811 0.040113
## Cumulative Proportion  0.557305 0.709567 0.821013 0.895824 0.935937
```

```
glrm_orig5_pred <- h2o.predict(glrm_orig5, dfDecom.df.h2o)
```

end GLRM on original data

GLRM on trend data

dimension = 3

```
dfDecom.trend.h2o <- as.h2o(dfDecom.trend)
```

```
glrm_trend3 <- h2o.glrm(
  training_frame = dfDecom.trend.h2o,
  k = 3,
  loss = "Quadratic",
  regularization_x = "None",
  regularization_y = "None",
  transform = "STANDARDIZE",
  max_iterations = 2000,
  seed = 123
)
```

```
print(summary(glrm_trend3))
```

```
## Model Details:
## ==============
##
## H2ODimReductionModel: glrm
## Model Key:  GLRM_model_R_1618906539228_128
## Model Summary:
##   number_of_iterations final_step_size final_objective_value
## 1                   57         0.00003             103.84812
##
## H2ODimReductionMetrics: glrm
## ** Reported on training data. **
##
## Sum of Squared Error (Numeric):  103.8481
## Misclassification Error (Categorical):  0
## Number of Numeric Entries:  1888
## Number of Categorical Entries:  0
##
##
##
## Scoring History:
##             timestamp   duration iterations step_size objective
## 1 2021-04-24 07:08:36  0.011 sec          0   0.66667 652.59620
## 2 2021-04-24 07:08:36  0.011 sec          1   0.44444 652.59620
## 3 2021-04-24 07:08:36  0.012 sec          2   0.22222 652.59620
## 4 2021-04-24 07:08:36  0.012 sec          3   0.07407 652.59620
## 5 2021-04-24 07:08:36  0.012 sec          4   0.07778 413.61394
##
## ---
##              timestamp   duration iterations step_size objective
## 52 2021-04-24 07:08:36  0.025 sec         51   0.02035 103.84812
## 53 2021-04-24 07:08:36  0.026 sec         52   0.01017 103.84812
## 54 2021-04-24 07:08:36  0.026 sec         53   0.00339 103.84812
## 55 2021-04-24 07:08:36  0.026 sec         54   0.00085 103.84812
## 56 2021-04-24 07:08:36  0.026 sec         55   0.00017 103.84812
## 57 2021-04-24 07:08:36  0.027 sec         56   0.00003 103.84812
##
## NULL
```

```
print(glrm_trend3@model$importance)
```

```
## Importance of components:
##                             pc1      pc2      pc3
## Standard deviation     1.956401 1.587850 1.099691
## Proportion of Variance 0.478438 0.315159 0.151165
## Cumulative Proportion  0.478438 0.793597 0.944762
```

```
glrm_trend3_pred <- h2o.predict(glrm_trend3, dfDecom.trend.h2o)
```

dimension = 2

```
glrm_trend2 <- h2o.glrm(
  training_frame = dfDecom.trend.h2o,
  k = 2,
  loss = "Quadratic",
  regularization_x = "None",
  regularization_y = "None",
  transform = "STANDARDIZE",
  max_iterations = 2000,
  seed = 123
)
```

```
print(summary(glrm_trend2))
```

```
## Model Details:
## ==============
##
## H2ODimReductionModel: glrm
## Model Key:  GLRM_model_R_1618906539228_130
## Model Summary:
##   number_of_iterations final_step_size final_objective_value
## 1                    8         0.00009            1009.88442
##
## H2ODimReductionMetrics: glrm
## ** Reported on training data. **
##
## Sum of Squared Error (Numeric):  1009.884
## Misclassification Error (Categorical):  0
## Number of Numeric Entries:  1888
## Number of Categorical Entries:  0
##
##
##
## Scoring History:
##             timestamp   duration iterations step_size  objective
## 1 2021-04-24 07:08:40  0.011 sec          0   0.66667 1009.88442
## 2 2021-04-24 07:08:40  0.011 sec          1   0.44444 1009.88442
## 3 2021-04-24 07:08:40  0.011 sec          2   0.22222 1009.88442
## 4 2021-04-24 07:08:40  0.012 sec          3   0.07407 1009.88442
## 5 2021-04-24 07:08:40  0.012 sec          4   0.01852 1009.88442
## 6 2021-04-24 07:08:40  0.012 sec          5   0.00370 1009.88442
## 7 2021-04-24 07:08:40  0.013 sec          6   0.00062 1009.88442
## 8 2021-04-24 07:08:40  0.013 sec          7   0.00009 1009.88442
##
## NULL
```

```
print(glrm_trend2@model$importance)
```

```
## Importance of components:
##                              pc1      pc2
## Standard deviation     1.924219 0.000000
## Proportion of Variance 0.462827 0.000000
## Cumulative Proportion  0.462827 0.462827
```

```
glrm_trend2_pred <- h2o.predict(glrm_trend2, dfDecom.trend.h2o)
```

dimension = 1

```
glrm_trend1 <- h2o.glrm(
  training_frame = dfDecom.trend.h2o,
  k = 1,
  loss = "Quadratic",
  regularization_x = "None",
  regularization_y = "None",
  transform = "STANDARDIZE",
  max_iterations = 2000,
  seed = 123
)
```

```
print(summary(glrm_trend1))
```

```
## Model Details:
## ==============
##
## H2ODimReductionModel: glrm
## Model Key:  GLRM_model_R_1618906539228_132
## Model Summary:
##   number_of_iterations final_step_size final_objective_value
## 1                    8         0.00009            1692.64425
##
## H2ODimReductionMetrics: glrm
## ** Reported on training data. **
##
## Sum of Squared Error (Numeric):  1692.644
## Misclassification Error (Categorical):  0
## Number of Numeric Entries:  1888
## Number of Categorical Entries:  0
##
##
##
## Scoring History:
##             timestamp   duration iterations step_size  objective
## 1 2021-04-24 07:08:44  0.008 sec          0   0.66667 1692.64425
## 2 2021-04-24 07:08:44  0.008 sec          1   0.44444 1692.64425
## 3 2021-04-24 07:08:44  0.008 sec          2   0.22222 1692.64425
## 4 2021-04-24 07:08:44  0.009 sec          3   0.07407 1692.64425
## 5 2021-04-24 07:08:44  0.009 sec          4   0.01852 1692.64425
## 6 2021-04-24 07:08:44  0.009 sec          5   0.00370 1692.64425
## 7 2021-04-24 07:08:44  0.009 sec          6   0.00062 1692.64425
## 8 2021-04-24 07:08:44  0.010 sec          7   0.00009 1692.64425
##
## NULL
```

```
print(glrm_trend1@model$importance)
```

```
## Importance of components:
##                             pc1
## Standard deviation     0.892893
## Proportion of Variance 0.099657
## Cumulative Proportion  0.099657
```

```
glrm_trend1_pred <- h2o.predict(glrm_trend1, dfDecom.trend.h2o)
```

end GLRM on trend data

DFA on original data

dimension = 6

```
dfDecom.df.t <- t(df[, 1:8])
```

Standardizing seasonal data.

```
dfDecom.df.t.mean <- apply(dfDecom.df.t, 1, mean, na.rm = TRUE)
dfDecom.df.t.std <- dfDecom.df.t - dfDecom.df.t.mean
```

```
# create loading matrix
Z_vals <- list("z11", 0, 0, 0, 0, 0, "z21", "z22", 0, 0, 0, 0, "z31", "z32", "z33",
0, 0, 0, "z41", "z42", "z43", "z44", 0, 0,
               "z51", "z52", "z53", "z54", "z55", 0, "61", "z62", "z63", "64", "z65",
"z66", "z71", "z72", "z73", "z74", "z75", "z76", "z81",  "z82", "z83", "z84",  "z85",
"z86")
ZZ <- matrix(Z_vals, nrow = 8, ncol = 6, byrow = TRUE)
## 'aa' is the offset/scaling
aa <- "zero"
## 'DD' and 'd' are for covariates
DD <- "zero"  # matrix(0,mm,1)
dd <- "zero"  # matrix(0,1,wk_last)
## 'RR' is var-cov matrix for obs errors
RR <- "diagonal and unequal"


## number of processes
mm <- 6
## 'BB' is identity: 1's along the diagonal & 0's elsewhere
BB <- "identity"  # diag(mm)
## 'uu' is a column vector of 0's
uu <- "zero"  # matrix(0, mm, 1)
## 'CC' and 'cc' are for covariates
CC <- "zero"  # matrix(0, mm, 1)
cc <- "zero"  # matrix(0, 1, wk_last)
## 'QQ' is identity
QQ <- "identity"  # diag(mm)


## list with specifications for model vectors/matrices
mod_list <- list(Z = ZZ, A = aa, D = DD, d = dd, R = RR, B = BB,
                 U = uu, C = CC, c = cc, Q = QQ)
## list with model inits
init_list <- list(x0 = matrix(rep(0, mm), mm, 1))
## list with model control parameters
con_list <- list(maxit = 3000, allow.degen = TRUE)
```

Fitting the model.

```
## fit MARSS
dfa_orig6 <- MARSS(y = dfDecom.df.t.std, model = mod_list, inits = init_list,
               control = con_list)
```

```
## Warning! Abstol convergence only. Maxit (=3000) reached before log-log convergenc
e.
##
## MARSS fit is
## Estimation method: kem
## Convergence test: conv.test.slope.tol = 0.5, abstol = 0.001
## WARNING: Abstol convergence only no log-log convergence.
##  maxit (=3000) reached before log-log convergence.
##  The likelihood and params might not be at the ML values.
##  Try setting control$maxit higher.
## Log-likelihood: -6076.605
## AIC: 12247.21   AICc: 12249.66
##
##                       Estimate
## Z.z11                 4.26e+01
## Z.z21                 1.31e+01
## Z.z31                 8.39e+00
## Z.z41                 1.40e+01
## Z.z51                 5.76e+00
## Z.61                  3.64e-01
## Z.z71                 1.19e+00
## Z.z81                 5.11e+00
## Z.z22                 6.40e+00
## Z.z32                 4.53e+00
## Z.z42                 1.11e+01
## Z.z52                -7.18e-01
## Z.z62                -1.64e-02
## Z.z72                -2.03e-02
## Z.z82                 1.22e+00
## Z.z33                 2.69e+00
## Z.z43                -4.10e+00
## Z.z53                 2.72e+00
## Z.z63                 1.28e-01
## Z.z73                -2.54e-01
## Z.z83                -1.61e+00
## Z.z44                 8.25e+00
## Z.z54                 2.76e-01
## Z.64                  9.92e-02
## Z.z74                -2.99e-03
## Z.z84                -1.50e-01
## Z.z55                 2.33e+00
## Z.z65                 5.72e-02
## Z.z75                -4.21e-01
## Z.z85                -1.06e+00
## Z.z66                 6.57e-02
## Z.z76                 1.33e-01
## Z.z86                 3.16e+00
## R.(PM2.5,PM2.5)       1.09e+03
## R.(NO,NO)             7.35e+01
## R.(NO2,NO2)           3.55e+01
## R.(NOx,NOx)           4.96e-01
```

```
## R.(NH3,NH3)          2.04e+01
## R.(CO,CO)            6.84e-01
## R.(Benzene,Benzene)  1.30e-01
## R.(Toluene,Toluene)  1.96e+00
## x0.X1               -1.50e+00
## x0.X2               -2.21e+00
## x0.X3                2.40e+00
## x0.X4                9.09e+00
## x0.X5                2.39e-01
## x0.X6                4.12e-01
## Initial states (x0) defined at t=0
##
## Standard errors have not been calculated.
## Use MARSSparamCIs to compute CIs and bias estimates.
##
## Convergence warnings
##  Warning: the  Z.z74  parameter value has not converged.
##  Warning: the  R.(NOx,NOx)  parameter value has not converged.
##  Warning: the  logLik  parameter value has not converged.
##  Type MARSSinfo("convergence") for more info on this warning.
```

```
dfa_pred_orig6 <- t(dfa_orig6$states)
print(head(dfa_pred_orig6))
```

```
##              X1         X2         X3        X4         X5         X6
## [1,] -1.4996663 -2.2098419  2.4028117  9.091306  0.2387484  0.4123392
## [2,] -0.8877632  0.3538429  0.9200842 12.225819  0.3198465 -0.5810277
## [3,] -1.3073919 -0.9929412  0.8161257 10.848227 -0.4568604 -0.1751335
## [4,] -1.1926095 -0.1038362 -0.1821277 11.816507 -0.9645685 -0.8304217
## [5,] -2.2424930  0.2996105 -0.4010850 11.358874 -0.6837383 -0.8748221
## [6,] -1.3525276  1.9800380 -0.9897810 12.702857 -0.2125205 -1.4806423
```

dimension = 5

```
# create loading matrix
Z_vals <- list("z11", 0, 0, 0, 0, "z21", "z22", 0, 0, 0, "z31", "z32", "z33", 0, 0, "
z41", "z42", "z43", "z44", 0,
               "z51", "z52", "z53", "z54", "z55", "61", "z62", "z63", "64", "z65", "z
71", "z72", "z73", "z74", "z75", "z81",  "z82", "z83", "z84",  "z85")
ZZ <- matrix(Z_vals, nrow = 8, ncol = 5, byrow = TRUE)
## 'aa' is the offset/scaling
aa <- "zero"
## 'DD' and 'd' are for covariates
DD <- "zero"  # matrix(0,mm,1)
dd <- "zero"  # matrix(0,1,wk_last)
## 'RR' is var-cov matrix for obs errors
RR <- "diagonal and unequal"


## number of processes
mm <- 5
## 'BB' is identity: 1's along the diagonal & 0's elsewhere
BB <- "identity"  # diag(mm)
## 'uu' is a column vector of 0's
uu <- "zero"  # matrix(0, mm, 1)
## 'CC' and 'cc' are for covariates
CC <- "zero"  # matrix(0, mm, 1)
cc <- "zero"  # matrix(0, 1, wk_last)
## 'QQ' is identity
QQ <- "identity"  # diag(mm)


## list with specifications for model vectors/matrices
mod_list <- list(Z = ZZ, A = aa, D = DD, d = dd, R = RR, B = BB,
                 U = uu, C = CC, c = cc, Q = QQ)
## list with model inits
init_list <- list(x0 = matrix(rep(0, mm), mm, 1))
## list with model control parameters
con_list <- list(maxit = 3000, allow.degen = TRUE)
```

Fitting the model.

```
## fit MARSS
dfa_orig5 <- MARSS(y = dfDecom.df.t.std, model = mod_list, inits = init_list,
               control = con_list)
```

```
## Warning! Abstol convergence only. Maxit (=3000) reached before log-log convergenc
e.
##
## MARSS fit is
## Estimation method: kem
## Convergence test: conv.test.slope.tol = 0.5, abstol = 0.001
## WARNING: Abstol convergence only no log-log convergence.
##  maxit (=3000) reached before log-log convergence.
##  The likelihood and params might not be at the ML values.
##  Try setting control$maxit higher.
## Log-likelihood: -6094.453
## AIC: 12274.91   AICc: 12276.96
##
##                      Estimate
## Z.z11                 42.6144
## Z.z21                 13.1223
## Z.z31                  8.4699
## Z.z41                 13.6294
## Z.z51                  5.9015
## Z.61                   0.3679
## Z.z71                  1.1814
## Z.z81                  5.0979
## Z.z22                  6.3005
## Z.z32                  4.0907
## Z.z42                 11.2763
## Z.z52                 -1.1942
## Z.z62                 -0.0299
## Z.z72                  0.0359
## Z.z82                  1.5499
## Z.z33                  1.2070
## Z.z43                  3.7884
## Z.z53                  1.8764
## Z.z63                  0.0778
## Z.z73                 -0.3326
## Z.z83                 -3.1382
## Z.z44                  8.8555
## Z.z54                 -0.9919
## Z.64                   0.0960
## Z.z74                  0.2712
## Z.z84                  1.7431
## Z.z55                  2.1221
## Z.z65                  0.0699
## Z.z75                 -0.3629
## Z.z85                  0.5134
## R.(PM2.5,PM2.5)     1194.5269
## R.(NO,NO)             67.3047
## R.(NO2,NO2)           60.0944
## R.(NOx,NOx)            0.5676
## R.(NH3,NH3)           20.9649
## R.(CO,CO)              0.7362
## R.(Benzene,Benzene)    0.1201
```

```
## R.(Toluene,Toluene)    1.9237
## x0.X1                 -1.3994
## x0.X2                 -2.7013
## x0.X3                  4.5824
## x0.X4                  5.8498
## x0.X5                  2.0435
## Initial states (x0) defined at t=0
##
## Standard errors have not been calculated.
## Use MARSSparamCIs to compute CIs and bias estimates.
##
## Convergence warnings
##  Warning: the  Z.z72  parameter value has not converged.
##  Warning: the  Z.z85  parameter value has not converged.
##  Warning: the  R.(NOx,NOx)  parameter value has not converged.
##  Warning: the  logLik  parameter value has not converged.
##  Type MARSSinfo("convergence") for more info on this warning.
```

```
dfa_pred_orig5 <- t(dfa_orig5$states)
print(head(dfa_pred_orig5))
```

```
##              X1         X2       X3       X4         X5
## [1,] -1.3993515 -2.7013236 4.582352 5.849679 2.04358012
## [2,] -0.9538943 -0.0760575 6.601955 8.741472 1.59166330
## [3,] -1.3892863 -1.2972903 5.164682 7.998145 0.96869300
## [4,] -1.3963677 -0.3095947 5.665842 9.194415 0.04796077
## [5,] -2.4529139  0.2151170 5.512999 8.739080 0.21931789
## [6,] -1.6418129  1.9175287 6.742903 9.834732 0.26073066
```

end DFA on original data

DFA on seasonal data

```
dfDecom.seasonal.t <- t(dfDecom.seasonal[, 1:8])
```

Standardizing seasonal data.

```
dfDecom.seasonal.t.mean <- apply(dfDecom.seasonal.t, 1, mean, na.rm = TRUE)
dfDecom.seasonal.t.std <- dfDecom.seasonal.t - dfDecom.seasonal.t.mean
```

dimension = 3

```
# create loading matrix
Z_vals <- list("z11", 0, 0, "z21", "z22", 0, "z31", "z32", "z33", "z41", "z42", "z4
3",
               "z51", "z52", "z53", "61", "z62", "z63", "z71", "z72", "z73", "z81",
"z82", "z83")
ZZ <- matrix(Z_vals, nrow = 8, ncol = 3, byrow = TRUE)
## 'aa' is the offset/scaling
aa <- "zero"
## 'DD' and 'd' are for covariates
DD <- "zero"  # matrix(0,mm,1)
dd <- "zero"  # matrix(0,1,wk_last)
## 'RR' is var-cov matrix for obs errors
RR <- "diagonal and unequal"


## number of processes
mm <- 3
## 'BB' is identity: 1's along the diagonal & 0's elsewhere
BB <- "identity"  # diag(mm)
## 'uu' is a column vector of 0's
uu <- "zero"  # matrix(0, mm, 1)
## 'CC' and 'cc' are for covariates
CC <- "zero"  # matrix(0, mm, 1)
cc <- "zero"  # matrix(0, 1, wk_last)
## 'QQ' is identity
QQ <- "identity"  # diag(mm)


## list with specifications for model vectors/matrices
mod_list <- list(Z = ZZ, A = aa, D = DD, d = dd, R = RR, B = BB,
                 U = uu, C = CC, c = cc, Q = QQ)
## list with model inits
init_list <- list(x0 = matrix(rep(0, mm), mm, 1))
## list with model control parameters
con_list <- list(maxit = 3000, allow.degen = TRUE)
```

Fitting the model.

```
## fit MARSS
dfa_seasonal3 <- MARSS(y = dfDecom.seasonal.t.std, model = mod_list, inits = init_lis
t,
               control = con_list)
```

```
## Warning! Reached maxit before parameters converged. Maxit was 3000.
##  neither abstol nor log-log convergence tests were passed.
##
## MARSS fit is
## Estimation method: kem
## Convergence test: conv.test.slope.tol = 0.5, abstol = 0.001
## WARNING: maxit reached at  3000  iter before convergence.
##  Neither abstol nor log-log convergence test were passed.
##  The likelihood and params are not at the ML values.
##  Try setting control$maxit higher.
## Log-likelihood: -4695.14
## AIC: 9454.279   AICc: 9455.418
##
##                           Estimate
## Z.z11                     2.37e+01
## Z.z21                     5.76e+00
## Z.z31                     3.66e+00
## Z.z41                     3.66e+00
## Z.z51                     3.24e+00
## Z.61                      2.11e-01
## Z.z71                     4.78e-01
## Z.z81                     1.79e+00
## Z.z22                     5.09e+00
## Z.z32                     2.27e+00
## Z.z42                     6.69e+00
## Z.z52                    -1.49e+00
## Z.z62                    -8.98e-02
## Z.z72                     2.41e-01
## Z.z82                     2.43e+00
## Z.z33                     1.07e+00
## Z.z43                     5.48e-01
## Z.z53                    -3.56e-04
## Z.z63                     9.37e-03
## Z.z73                     4.15e-02
## Z.z83                    -9.49e-01
## R.(PM2.5.s,PM2.5.s)       3.59e+02
## R.(NO.s,NO.s)             2.05e+01
## R.(NO2.s,NO2.s)           1.36e+01
## R.(NOx.s,NOx.s)           3.12e+01
## R.(NH3.s,NH3.s)           9.75e+00
## R.(CO.s,CO.s)             1.17e-01
## R.(Benzene.s,Benzene.s)   4.67e-02
## R.(Toluene.s,Toluene.s)   5.47e-03
## x0.X1                    -2.87e+00
## x0.X2                    -1.61e+00
## x0.X3                    -2.59e+00
## Initial states (x0) defined at t=0
##
## Standard errors have not been calculated.
## Use MARSSparamCIs to compute CIs and bias estimates.
##
```

```
## Convergence warnings
## 15 warnings. First 10 shown.  Type cat(object$errors) to see the full list.
##  Warning: the  Z.z51  parameter value has not converged.
##  Warning: the  Z.z42  parameter value has not converged.
##  Warning: the  Z.z52  parameter value has not converged.
##  Warning: the  Z.z62  parameter value has not converged.
##  Warning: the  Z.z43  parameter value has not converged.
##  Warning: the  Z.z53  parameter value has not converged.
##  Warning: the  Z.z63  parameter value has not converged.
##  Warning: the  R.(NOx.s,NOx.s)  parameter value has not converged.
##  Warning: the  R.(NH3.s,NH3.s)  parameter value has not converged.
##  Warning: the  R.(Toluene.s,Toluene.s)  parameter value has not converged.
```

```
dfa_pred_seasonal3 <- t(dfa_seasonal3$states)
print(head(dfa_pred_seasonal3))
```

```
##             X1          X2        X3
## [1,] -2.865785 -1.60837069 -2.588218
## [2,] -2.844794 -1.16044236 -2.649078
## [3,] -2.985927 -1.16547012 -2.591991
## [4,] -2.932459 -0.39046971 -2.562785
## [5,] -3.454262 -0.23642561 -2.878204
## [6,] -3.461719 -0.05189357 -2.815201
```

dimension = 2

```
# create loading matrix
Z_vals <- list("z11", 0, "z21", "z22", "z31", "z32", "z41", "z42",
               "z51", "z52", "61", "z62", "z71", "z72", "z81",  "z82")
ZZ <- matrix(Z_vals, nrow = 8, ncol = 2, byrow = TRUE)
## 'aa' is the offset/scaling
aa <- "zero"
## 'DD' and 'd' are for covariates
DD <- "zero"  # matrix(0,mm,1)
dd <- "zero"  # matrix(0,1,wk_last)
## 'RR' is var-cov matrix for obs errors
RR <- "diagonal and unequal"


## number of processes
mm <- 2
## 'BB' is identity: 1's along the diagonal & 0's elsewhere
BB <- "identity"  # diag(mm)
## 'uu' is a column vector of 0's
uu <- "zero"  # matrix(0, mm, 1)
## 'CC' and 'cc' are for covariates
CC <- "zero"  # matrix(0, mm, 1)
cc <- "zero"  # matrix(0, 1, wk_last)
## 'QQ' is identity
QQ <- "identity"  # diag(mm)


## list with specifications for model vectors/matrices
mod_list <- list(Z = ZZ, A = aa, D = DD, d = dd, R = RR, B = BB,
                 U = uu, C = CC, c = cc, Q = QQ)
## list with model inits
init_list <- list(x0 = matrix(rep(0, mm), mm, 1))
## list with model control parameters
con_list <- list(maxit = 3000, allow.degen = TRUE)
```

Fitting the model.

```
## fit MARSS
dfa_seasonal2 <- MARSS(y = dfDecom.seasonal.t.std, model = mod_list, inits = init_lis
t,
               control = con_list)
```

```
## Success! abstol and log-log tests passed at 210 iterations.
## Alert: conv.test.slope.tol is 0.5.
## Test with smaller values (<0.1) to ensure convergence.
##
## MARSS fit is
## Estimation method: kem
## Convergence test: conv.test.slope.tol = 0.5, abstol = 0.001
## Estimation converged in 210 iterations.
## Log-likelihood: -5257.849
## AIC: 10565.7   AICc: 10566.4
##
##                          Estimate
## Z.z11                      0.0756
## Z.z21                      5.0930
## Z.z31                      3.3298
## Z.z41                      3.8810
## Z.z51                      2.1641
## Z.61                       0.1459
## Z.z71                      0.4010
## Z.z81                      1.4719
## Z.z22                      4.6927
## Z.z32                      3.0681
## Z.z42                      3.5761
## Z.z52                      1.9940
## Z.z62                      0.1344
## Z.z72                      0.3695
## Z.z82                      1.3563
## R.(PM2.5.s,PM2.5.s)     5226.6293
## R.(NO.s,NO.s)             22.6009
## R.(NO2.s,NO2.s)           17.5011
## R.(NOx.s,NOx.s)           53.4017
## R.(NH3.s,NH3.s)           22.2160
## R.(CO.s,CO.s)              0.1601
## R.(Benzene.s,Benzene.s)    0.0817
## R.(Toluene.s,Toluene.s)    9.0965
## x0.X1                    -46.3994
## x0.X2                     45.2436
## Initial states (x0) defined at t=0
##
## Standard errors have not been calculated.
## Use MARSSparamCIs to compute CIs and bias estimates.
```

```
dfa_pred_seasonal2 <- t(dfa_seasonal2$states)
print(head(dfa_pred_seasonal2))
```

```
##              X1       X2
## [1,] -46.39365 45.23854
## [2,] -46.25078 45.35902
## [3,] -46.28142 45.31876
## [4,] -45.95916 45.60277
## [5,] -46.24403 45.32631
## [6,] -46.21545 45.33755
```

end DFA on seasonal data

Experiments on dimension = 6

Building lm on PCA of original data.

```
df_pred_pca <- as.data.frame(pca_orig_pred[, 1:6])
df_pred_pca_train <- as.data.frame(df_pred_pca[1:mid, ])
df_pred_pca_test <- as.data.frame(df_pred_pca[(mid+1):en, ])
```

```
lm.2 = lm(df_aqi_train ~ df_pred_pca_train[, 1] + df_pred_pca_train[, 2] + df_pred_pc
a_train[, 3] + df_pred_pca_train[, 4] + df_pred_pca_train[, 5] + df_pred_pca_train[,
6])

yhat.2 = predict(lm.2, data.frame(df_pred_pca_train))
train.err.2.m = mean((y-yhat.2)^2)
y0hat.2 = predict(lm.2, data.frame(df_pred_pca_test))
```

```
## Warning: 'newdata' had 96 rows but variables found have 165 rows
```

```
test.err.2.m = mean((y0-y0hat.2)^2)
```

```
## Warning in y0 - y0hat.2: longer object length is not a multiple of shorter
## object length
```

```
train.err.2.m / train.err.1
```

```
## [1] 1.329139
```

```
test.err.2.m/ test.err.1
```

```
## [1] 0.9897094
```

Building lm on DFA of original data.

```
df_pred_dfa <- as.data.frame(dfa_pred_orig6[, 1:6])
df_pred_dfa_train <- as.data.frame(df_pred_dfa[1:mid, ])
df_pred_dfa_test <- as.data.frame(df_pred_dfa[(mid+1):en, ])
```

```
lm.3 = lm(df_aqi_train ~ df_pred_dfa_train[, 1] + df_pred_dfa_train[, 2] + df_pred_df
a_train[, 3] + df_pred_dfa_train[, 4] + df_pred_dfa_train[, 5] + df_pred_dfa_train[,
6])

yhat.3 = predict(lm.3, data.frame(df_pred_dfa_train))
train.err.3.m = mean((y-yhat.3)^2)
y0hat.3 = predict(lm.3, data.frame(df_pred_dfa_test))
```

```
## Warning: 'newdata' had 96 rows but variables found have 165 rows
```

```
test.err.3.m = mean((y0-y0hat.3)^2)
```

```
## Warning in y0 - y0hat.3: longer object length is not a multiple of shorter
## object length
```

```
train.err.3.m / train.err.1
```

```
## [1] 1.851442
```

```
test.err.3.m/ test.err.1
```

```
## [1] 0.9507366
```

Building lm on GLRM of original data.

```
df_pred_glrm <- as.data.frame(glrm_orig6_pred[, 1:6])
df_pred_glrm_train <- as.data.frame(df_pred_glrm[1:mid, ])
df_pred_glrm_test <- as.data.frame(df_pred_glrm[(mid+1):en, ])
```

```
lm.4 = lm(df_aqi_train ~ df_pred_glrm_train[, 1] + df_pred_glrm_train[, 2] + df_pred_
glrm_train[, 3] + df_pred_glrm_train[, 4] + df_pred_glrm_train[, 5] + df_pred_glrm_tr
ain[, 6])

yhat.4 = predict(lm.4, data.frame(df_pred_glrm_train))
train.err.4.m = mean((y-yhat.4)^2)
y0hat.4 = predict(lm.4, data.frame(df_pred_glrm_test))
```

```
## Warning: 'newdata' had 96 rows but variables found have 165 rows
```

```
test.err.4.m = mean((y0-y0hat.4)^2)
```

```
## Warning in y0 - y0hat.4: longer object length is not a multiple of shorter
## object length
```

```
train.err.4.m / train.err.1
```

```
## [1] 1.434882
```

```
test.err.4.m/ test.err.1
```

```
## [1] 0.9552111
```

Building mixed model tdim = 1, sdim = 3, rdim = 2

```
df_pred_t1_s3_r2 <- bind_cols(as.data.frame(glrm_trend1_pred[, 1:1]), as.data.frame(p
ca_random_pred[, 1:2]), as.data.frame(dfa_pred_seasonal3))
df_pred_t1_s3_r2_train <- as.data.frame(df_pred_t1_s3_r2[1:mid, ])
df_pred_t1_s3_r2_test <- as.data.frame(df_pred_t1_s3_r2[(mid+1):en, ])

lm.5 = lm(df_aqi_train ~ df_pred_t1_s3_r2_train[, 1] + df_pred_t1_s3_r2_train[, 2] +
df_pred_t1_s3_r2_train[, 3] + df_pred_t1_s3_r2_train[, 4] + df_pred_t1_s3_r2_train[,
5] + df_pred_t1_s3_r2_train[, 6])

yhat.5 = predict(lm.5, data.frame(df_pred_t1_s3_r2_train))
train.err.5.m = mean((y-yhat.5)^2)
y0hat.5 = predict(lm.5, data.frame(df_pred_t1_s3_r2_test))
```

```
## Warning: 'newdata' had 96 rows but variables found have 165 rows
```

```
test.err.5.m = mean((y0-y0hat.5)^2)
```

```
## Warning in y0 - y0hat.5: longer object length is not a multiple of shorter
## object length
```

```
train.err.5.m / train.err.1
```

```
## [1] 1.561224
```

```
test.err.5.m/ test.err.1
```

```
## [1] 0.9492149
```

Building mixed model tdim = 2, sdim = 2, rdim = 2

```
df_pred_t2_s2_r2 <- bind_cols(as.data.frame(glrm_trend2_pred[, 1:2]), as.data.frame(p
ca_random_pred[, 1:2]), as.data.frame(dfa_pred_seasonal2))
df_pred_t2_s2_r2_train <- as.data.frame(df_pred_t2_s2_r2[1:mid, ])
df_pred_t2_s2_r2_test <- as.data.frame(df_pred_t2_s2_r2[(mid+1):en, ])

lm.6 = lm(df_aqi_train ~ df_pred_t2_s2_r2_train[, 1] + df_pred_t2_s2_r2_train[, 2] +
df_pred_t2_s2_r2_train[, 3] + df_pred_t2_s2_r2_train[, 4] + df_pred_t2_s2_r2_train[,
5] + df_pred_t2_s2_r2_train[, 6])

yhat.6 = predict(lm.6, data.frame(df_pred_t1_s3_r2_train))
```

```
## Warning in predict.lm(lm.6, data.frame(df_pred_t1_s3_r2_train)): prediction from
## a rank-deficient fit may be misleading
```

```
train.err.6.m = mean((y-yhat.6)^2)
y0hat.6 = predict(lm.6, data.frame(df_pred_t2_s2_r2_test))
```

```
## Warning: 'newdata' had 96 rows but variables found have 165 rows
```

```
## Warning in predict.lm(lm.6, data.frame(df_pred_t2_s2_r2_test)): prediction from
## a rank-deficient fit may be misleading
```

```
test.err.6.m = mean((y0-y0hat.6)^2)
```

```
## Warning in y0 - y0hat.6: longer object length is not a multiple of shorter
## object length
```

```
train.err.6.m / train.err.1
```

```
## [1] 1.532629
```

```
test.err.6.m/ test.err.1
```

```
## [1] 0.97918
```

end Experiments on dimension = 6

Experiments on dimension = 5

Building lm on PCA of original data.

```
df_pred_pca <- as.data.frame(pca_orig_pred[, 1:5])
df_pred_pca_train <- as.data.frame(df_pred_pca[1:mid, ])
df_pred_pca_test <- as.data.frame(df_pred_pca[(mid+1):en, ])
```

```
lm.11 = lm(df_aqi_train ~ df_pred_pca_train[, 1] + df_pred_pca_train[, 2] + df_pred_p
ca_train[, 3] + df_pred_pca_train[, 4] + df_pred_pca_train[, 5])

yhat.11 = predict(lm.11, data.frame(df_pred_pca_train))
train.err.11.m = mean((y-yhat.11)^2)
y0hat.11 = predict(lm.11, data.frame(df_pred_pca_test))
```

```
## Warning: 'newdata' had 96 rows but variables found have 165 rows
```

```
test.err.11.m = mean((y0-y0hat.11)^2)
```

```
## Warning in y0 - y0hat.11: longer object length is not a multiple of shorter
## object length
```

```
train.err.11.m / train.err.1
```

```
## [1] 1.727229
```

```
test.err.11.m/ test.err.1
```

```
## [1] 0.9574493
```

Building lm on DFA of original data.

```
df_pred_dfa <- as.data.frame(dfa_pred_orig5[, 1:5])
df_pred_dfa_train <- as.data.frame(df_pred_dfa[1:mid, ])
df_pred_dfa_test <- as.data.frame(df_pred_dfa[(mid+1):en, ])
```

```
lm.12 = lm(df_aqi_train ~ df_pred_dfa_train[, 1] + df_pred_dfa_train[, 2] + df_pred_d
fa_train[, 3] + df_pred_dfa_train[, 4] + df_pred_dfa_train[, 5])

yhat.12 = predict(lm.12, data.frame(df_pred_dfa_train))
train.err.12.m = mean((y-yhat.12)^2)
y0hat.12 = predict(lm.12, data.frame(df_pred_dfa_test))
```

```
## Warning: 'newdata' had 96 rows but variables found have 165 rows
```

```
test.err.12.m = mean((y0-y0hat.12)^2)
```

```
## Warning in y0 - y0hat.12: longer object length is not a multiple of shorter
## object length
```

```
train.err.12.m / train.err.1
```

```
## [1] 1.888433
```

```
test.err.12.m/ test.err.1
```

```
## [1] 0.9510507
```

Building lm on GLRM of original data.

```
df_pred_glrm <- as.data.frame(glrm_orig5_pred[, 1:5])
df_pred_glrm_train <- as.data.frame(df_pred_glrm[1:mid, ])
df_pred_glrm_test <- as.data.frame(df_pred_glrm[(mid+1):en, ])
```

```
lm.13 = lm(df_aqi_train ~ df_pred_glrm_train[, 1] + df_pred_glrm_train[, 2] + df_pred
_glrm_train[, 3] + df_pred_glrm_train[, 4] + df_pred_glrm_train[, 5])

yhat.13 = predict(lm.13, data.frame(df_pred_glrm_train))
train.err.13.m = mean((y-yhat.13)^2)
y0hat.13 = predict(lm.13, data.frame(df_pred_glrm_test))
```

```
## Warning: 'newdata' had 96 rows but variables found have 165 rows
```

```
test.err.13.m = mean((y0-y0hat.13)^2)
```

```
## Warning in y0 - y0hat.13: longer object length is not a multiple of shorter
## object length
```

```
train.err.13.m / train.err.1
```

```
## [1] 1.727255
```

```
test.err.13.m/ test.err.1
```

```
## [1] 0.9574462
```

Building mixed model tdim = 1, sdim = 3, rdim = 1

```
df_pred_t1_s3_r1 <- bind_cols(as.data.frame(glrm_trend1_pred[, 1:1]), as.data.frame(p
ca_random_pred[, 1:1]), as.data.frame(dfa_pred_seasonal3))
df_pred_t1_s3_r1_train <- as.data.frame(df_pred_t1_s3_r1[1:mid, ])
df_pred_t1_s3_r1_test <- as.data.frame(df_pred_t1_s3_r1[(mid+1):en, ])

lm.14 = lm(df_aqi_train ~ df_pred_t1_s3_r1_train[, 1] + df_pred_t1_s3_r1_train[, 2] +
df_pred_t1_s3_r1_train[, 3] + df_pred_t1_s3_r1_train[, 4] + df_pred_t1_s3_r1_train[,
5])

yhat.14 = predict(lm.14, data.frame(df_pred_t1_s3_r1_train))
train.err.14.m = mean((y-yhat.14)^2)
y0hat.14 = predict(lm.14, data.frame(df_pred_t1_s3_r1_test))
```

```
## Warning: 'newdata' had 96 rows but variables found have 165 rows
```

```
test.err.14.m = mean((y0-y0hat.14)^2)
```

```
## Warning in y0 - y0hat.14: longer object length is not a multiple of shorter
## object length
```

```
train.err.14.m / train.err.1
```

```
## [1] 1.662598
```

```
test.err.14.m/ test.err.1
```

```
## [1] 0.9458615
```

Building mixed model tdim = 1, sdim = 2, rdim = 2

```
df_pred_t1_s2_r2 <- bind_cols(as.data.frame(glrm_trend1_pred[, 1:1]), as.data.frame(p
ca_random_pred[, 1:2]), as.data.frame(dfa_pred_seasonal2))
df_pred_t1_s2_r2_train <- as.data.frame(df_pred_t1_s2_r2[1:mid, ])
df_pred_t1_s2_r2_test <- as.data.frame(df_pred_t1_s2_r2[(mid+1):en, ])

lm.15 = lm(df_aqi_train ~ df_pred_t1_s2_r2_train[, 1] + df_pred_t1_s2_r2_train[, 2] +
df_pred_t1_s2_r2_train[, 3] + df_pred_t1_s2_r2_train[, 4] + df_pred_t1_s2_r2_train[,
5])

yhat.15 = predict(lm.15, data.frame(df_pred_t1_s3_r2_train))
train.err.15.m = mean((y-yhat.15)^2)
y0hat.15 = predict(lm.15, data.frame(df_pred_t1_s2_r2_test))
```

```
## Warning: 'newdata' had 96 rows but variables found have 165 rows
```

```
test.err.15.m = mean((y0-y0hat.15)^2)
```

```
## Warning in y0 - y0hat.15: longer object length is not a multiple of shorter
## object length
```

```
train.err.15.m / train.err.1
```

```
## [1] 1.560261
```

```
test.err.15.m/ test.err.1
```

```
## [1] 0.9653939
```

end Experiments on dimension = 5