Clustering Electric Vehicle Charging Pattern with DTW and EV Charging energy demand Prediction with LSTM

Hyeyoung Sim

PhD Candidate

Seoul National University

content

1. The importance of EV charging Pattern
2. Clustering time-series– DTW
3. Using the ‘dtwclust’ package in R for time-series clustering
4. Prediction on time-series data – LSTM
5. Applied ‘keras’, ‘tensorflow’ library in Python for time-series prediction

1. The importance of EV charging Pattern

- Why identifying EV charging Pattern is important?

  1. Transportation has a huge responsibility for CO2 emission

  - EV is considered the most effective tool for CO2 emission reduction

  - 8,151 thousand units (2022) to 39,208 thousand units (2030)

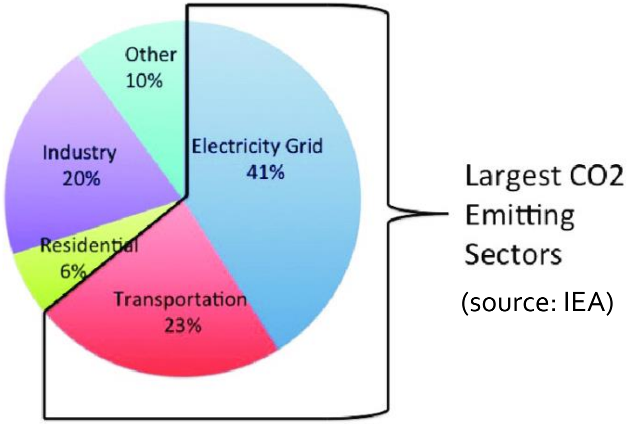
  2. EV charging pattern is related to CS and energy demand

  - Predicting EV charging patterns can expect energy demand

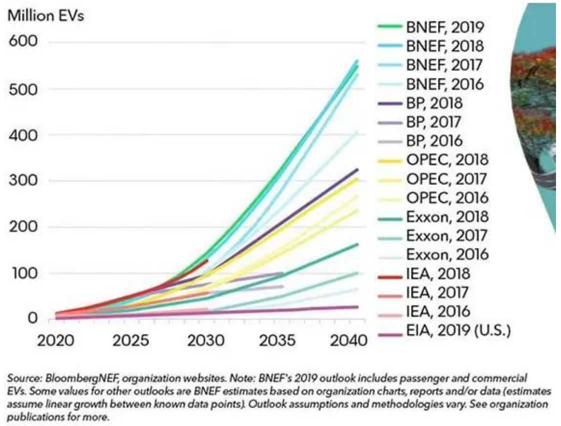
  - Still receive energy from conventional grids support fossil energy

→EV has not yet "greened" energy consumption, but it will be

→required to accurately model the effect of nationwide EV charging demands for the future infrastructure needs and the power gird



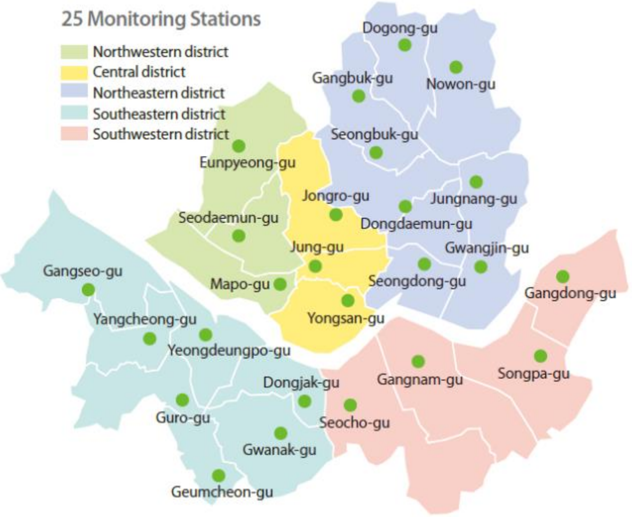
(source: IEA)



1. The importance of EV charging Pattern

- The goal of this study
  1. Clustering EV charging pattern
  2. Predicting EV charging pattern

- Data
  - EV charging (unit: minute)
  - Seoul(Korea)
  - Fast Charger



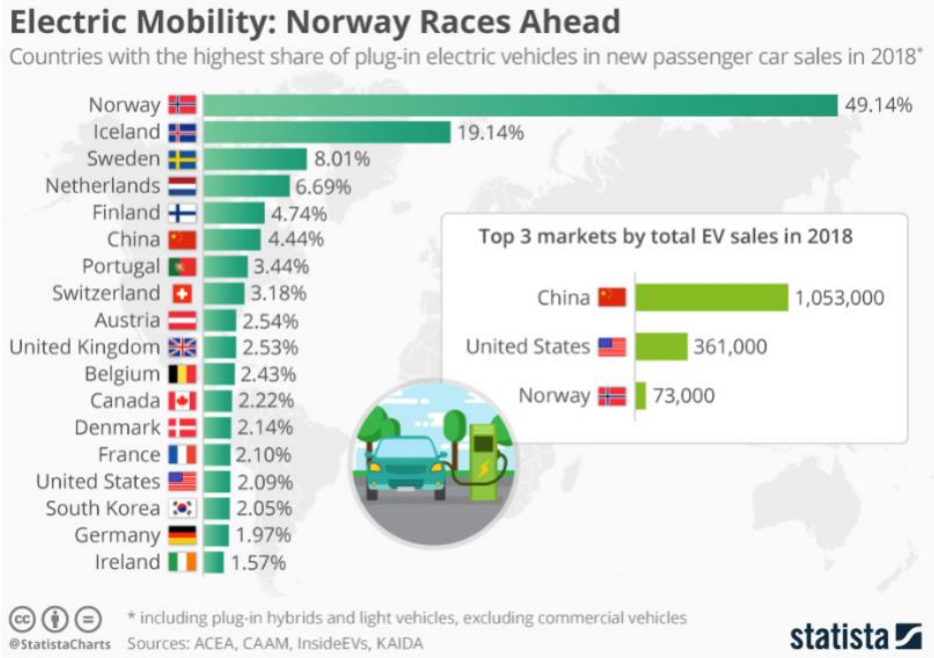


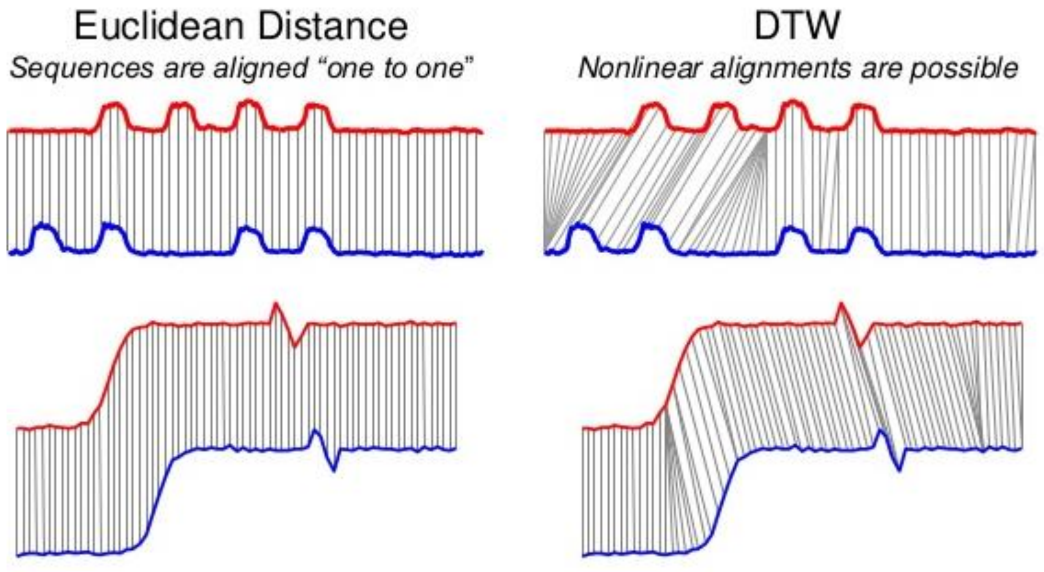
Table 1. Types of Charger

Site \ Time		Fast Charger	Standard Charger
Private Charger		(1) Private & Open type	(2) Private & Closed type
		Privately-operated facility, gas station	House parking
Public Charger		(3) Public & Open type	(4) Public & Closed type
	Destination	-	Workplace parking lot
	Route	Government-owned facility	-
	Emergency		

2. Clustering time-series

Euclidean vs. Dynamic Time Warping

- An example of matched points of two data vectors
  - Clearly these two series follow the same pattern, but the blue curve has a different time speed from the red
  - ED: not perfectly synced up, especially in peak of the blue curve
  - DTW: perfectly matched
- Euclidean
  - one-to-one match
  - Variable have equal time series
  - Pairs of data points and compares them
  - Matches same order of vector
  - 9 to 9, 10 to 10
- DTW
  - a one-to-many match
  - An algorithm for measuring similarity between two temporal sequences, which may vary in speed
  - 9 to (9, 10, 11)
  - Calculates the smallest distance between all points



Dr. Eamonn Keogh http://www.cs.ucr.edu/~eamonn/tutorials.html

$$ED(\vec{x},\vec{y}) = \sqrt{\sum_{i=1}^{m}(x_i - y_i)^2}$$

Euclidean Distance

$$DTW(\vec{x},\vec{y}) = \min \sqrt{\sum_{i=1}^{k} w_i}$$

DTW(1)

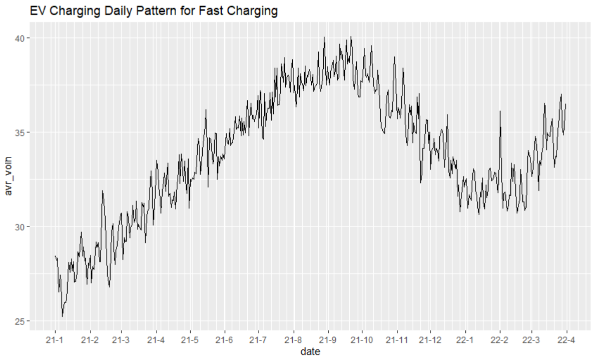
$$\gamma(i,j) = ED(i,j) + \min\{\gamma(i-1,j-1), \gamma(i-1,j), \gamma(i,j-1)\}$$

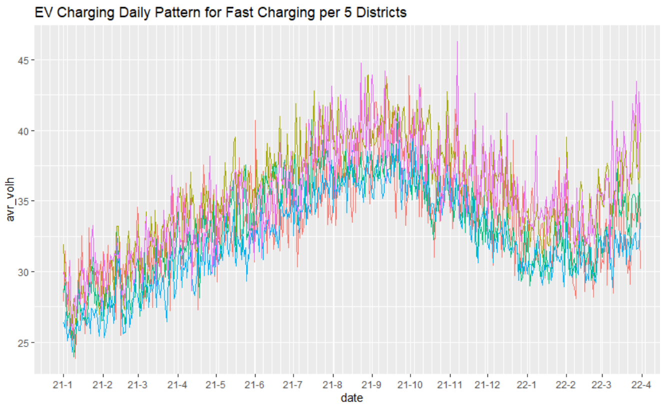
DTW(2)

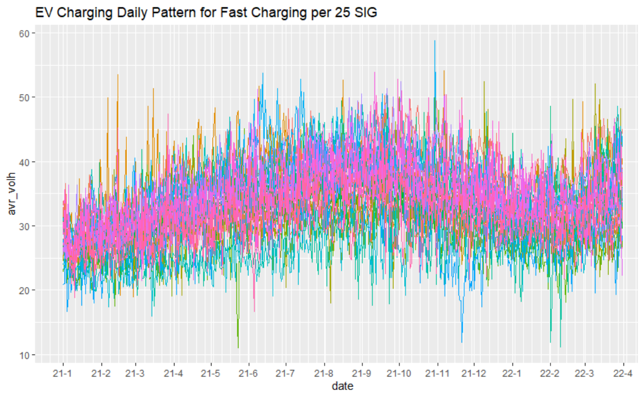
3. Using the 'dtwclust' package in R for time-series clustering

- Description of EV charging behavior

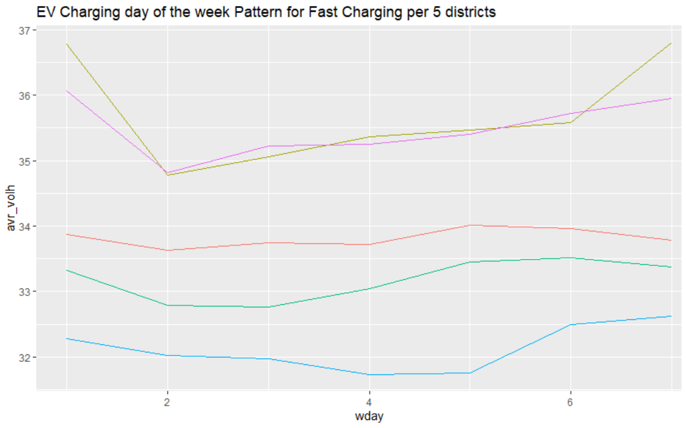
EV charging Daily pattern

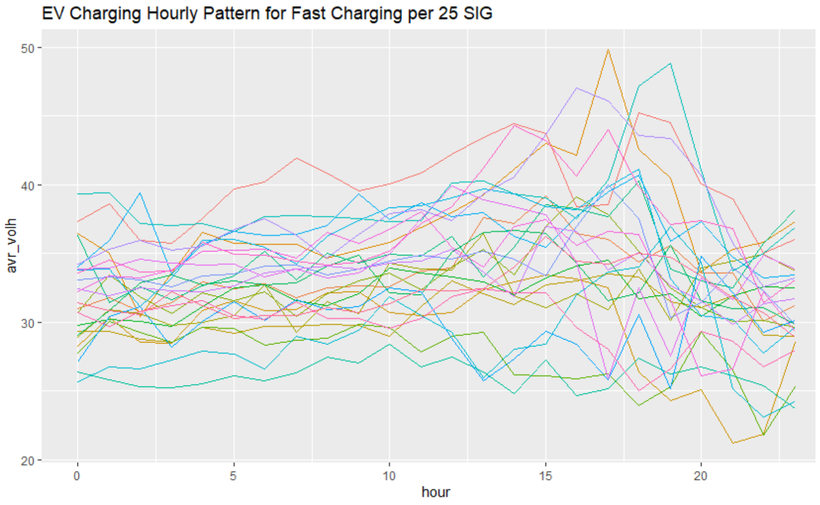






EV charging Hourly pattern





3. Using the ‘dtwclust’ package in R for time-series clustering

- Descriptive of EV charging Hourly behavior

```
> tst
# A tibble: 600 × 4
# Groups:   SIG [25]
   SIG     hour acvol_mean volh_mean
   <chr>  <int>      <dbl>     <dbl>
 1 강남구     0       19.9      37.3
 2 강남구     1       20.4      38.6
 3 강남구     2       19.4      35.9
 4 강남구     3       18.9      35.7
 5 강남구     4       19.7      37.5
 6 강남구     5       21.1      39.7
 7 강남구     6       21.2      40.2
 8 강남구     7       21.9      41.9
 9 강남구     8       21.4      40.8
10 강남구     9       21.0      39.6
# … with 590 more rows
```

```
> head(df_tst)
  hour   강남구   강동구   강북구   강서구   관악구   광진구   구로구   금천구
1    0 37.32609 31.03527 36.49691 28.27002 29.32545 28.90702 30.70376 27.74535
2    1 38.64754 31.81690 35.04841 30.38512 29.36039 30.93325 33.44139 30.05981
3    2 35.93716 30.77701 30.42445 28.57700 28.77432 30.65193 31.86153 29.24430
4    3 35.74551 31.43125 33.58905 28.50965 28.53442 29.75436 30.64135 28.54363
5    4 37.52800 32.94061 36.53247 30.82586 29.60246 30.02291 32.12669 29.65447
6    5 39.68677 32.45573 35.71703 31.56065 29.19326 30.52747 31.59994 29.58667
    노원구   도봉구 동대문구   동작구   마포구 서대문구   서초구   성동구   성북구
1 29.75686 29.06716 36.29696 26.41637 39.32538 25.68855 33.81772 33.99892 27.13219
2 30.17624 30.88536 31.45506 25.85462 39.41996 26.72690 33.86191 35.99095 30.45133
3 30.09186 32.84438 32.58642 25.29894 37.16499 26.62386 31.11681 39.43695 31.14641
4 29.72868 33.41925 31.65279 25.24893 37.04550 27.25316 33.22778 33.51372 28.23152
5 31.18159 32.70873 32.68372 25.49998 37.20877 27.89757 35.94446 35.66104 30.08507
6 32.51177 33.05141 33.36067 26.10843 36.58070 27.65958 36.07159 36.61171 31.49722
    송파구   양천구 영등포구   용산구   은평구   종로구     중구   중랑구
1 33.09027 34.27139 32.47523 33.84920 32.22540 33.57177 30.71302 31.44357
2 33.32412 35.34497 31.94765 33.91677 33.38172 34.50815 29.72430 30.88378
3 33.05294 36.00217 32.51932 34.57377 33.22198 33.68877 30.75853 30.58739
4 32.54990 35.22966 32.30884 34.28469 33.81481 33.76627 31.20808 32.24770
5 33.37782 35.52679 32.21692 34.15640 35.16207 35.81360 31.54183 31.23393
6 33.52006 36.79176 32.63904 34.26030 35.27265 34.95400 30.50904 30.29836
```

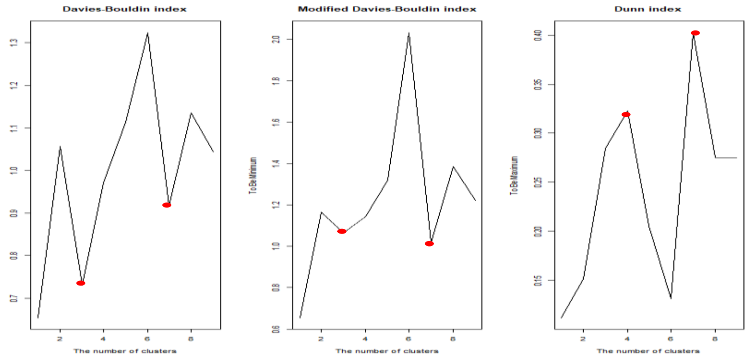
Row: 24 hour, col: 25gu

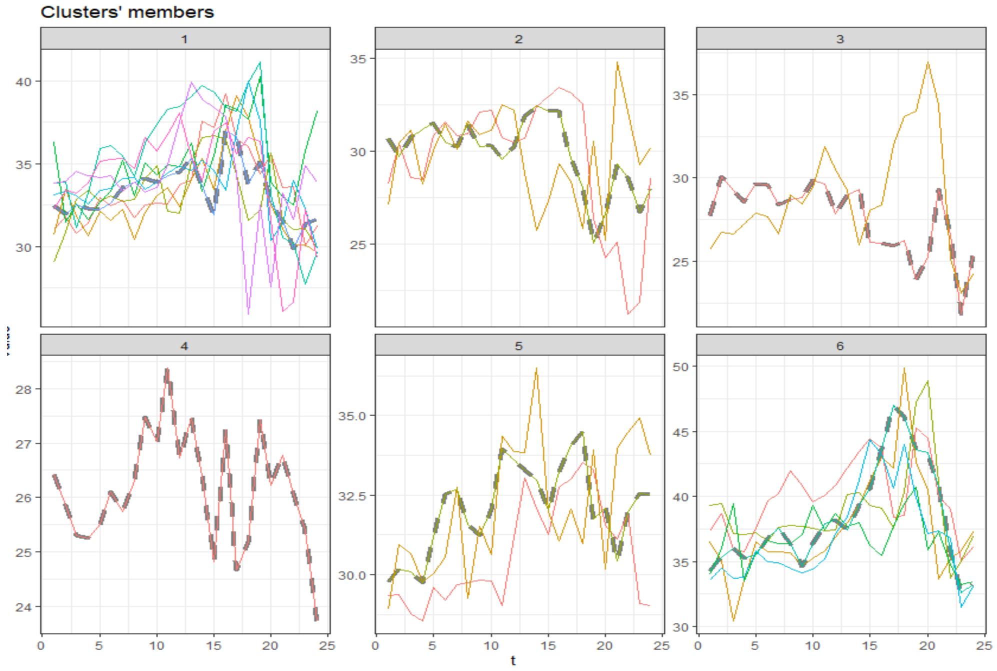
```
> sample <- t(df_tst[,2:25])                        #row; SIG, col; hour
> head(sample)
           [,1]     [,2]     [,3]     [,4]     [,5]     [,6]     [,7]     [,8]
강남구 37.32609 38.64754 35.93716 35.74551 37.52800 39.68677 40.24336 41.93796
강동구 31.03527 31.81690 30.77701 31.43125 32.94061 32.45573 32.76513 31.76634
강북구 36.49691 35.04841 30.42445 33.58905 36.53247 35.71703 35.71365 35.66512
강서구 28.27002 30.38512 28.57700 28.50965 30.82586 31.56065 30.84992 30.94896
관악구 29.32545 29.36039 28.77432 28.53442 29.60246 29.19326 29.67539 29.73371
광진구 28.90702 30.93325 30.65193 29.75436 30.02291 30.52747 32.74797 29.26142
           [,9]    [,10]    [,11]    [,12]    [,13]    [,14]    [,15]    [,16]
강남구 40.84761 39.58919 40.07758 40.87989 42.19778 43.42213 44.42881 43.74550
강동구 32.51345 32.64661 32.55896 33.73542 33.97757 37.58792 37.21073 39.22179
강북구 34.66650 35.23825 35.82605 36.87147 37.99233 39.27729 41.17296 43.02986
강서구 32.13910 32.23443 30.74668 30.51843 30.71027 32.39735 32.95129 33.45451
관악구 29.83371 29.79264 29.02294 31.16151 33.03963 32.09040 31.26627 32.74535
광진구 31.50115 30.62167 34.34603 33.86482 33.82798 36.49043 31.95983 31.06317
          [,17]    [,18]    [,19]    [,20]    [,21]    [,22]    [,23]    [,24]
강남구 38.39855 38.55106 45.25902 44.49727 40.03365 39.01957 35.03352 36.07407
강동구 36.49404 36.06003 34.38098 35.61398 33.57427 33.60899 30.04150 31.22010
강북구 42.17206 49.85111 42.58535 40.48732 33.66403 35.28929 35.80821 37.30213
강서구 33.15096 32.54089 26.38998 24.29042 25.11460 21.21291 21.85868 28.53150
관악구 32.99587 33.50955 33.30357 31.53259 31.10753 31.91493 29.08068 29.01902
광진구 32.05544 30.96314 33.91586 30.15495 33.97897 34.47813 34.93497 33.76247
```

Row: 25gu, col: 24 hour

```
tst <- keco%>%
  group_by(SIG,hour) %>%
  summarise(acvol_mean = mean(acvol),
        volh_mean = mean(volh))      #600 obs.
df_tst <- dcast(tst, hour~SIG, value.var = "volh_mean",
          mean, fill=0)         #row; hour, col; SIG
df_tst$hour <- as.numeric(df_tst$hour)
sample <- t(df_tst[,2:25])          #row; SIG, col; hour
```

3. Using the ‘dtwclust’ package in R for time-series clustering
- EV Hourly charging behavior (partitional)



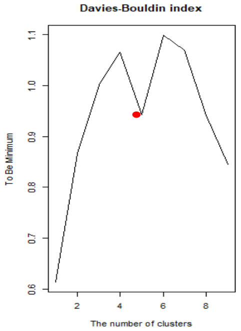


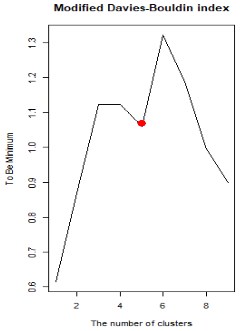

```
        강남구 강동구 강북구 강서구 관악구 광진구 구로구 금천구 노원구 도봉구
cluster      6      1      6      2      5      5      1      3      5      1
        동대문구 동작구 마포구 서대문구 서초구 성동구 성북구 송파구 양천구
cluster        1      4      6        3      1      6      2      1      6
        영등포구 용산구 은평구 종로구 중구
cluster        1      1      1      6    2
> cvi(dtw_cluster)       #cluster validity indices
      Sil        SF        CH        DB    DBstar         D       COP
0.2097529 0.0000000 7.3030805 1.1412385 1.3867687 0.3859013 0.1255462
```

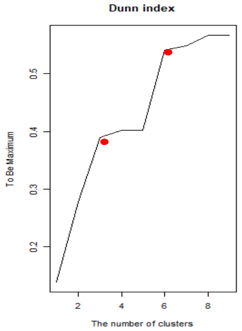
```
# TYPE = partitional
dtw <- tsclust(sample, k = 2L:10L, distance = "dtw_basic", type = "partitional"  )
eval_clust <- sapply(dtw, cvi); eval_clust
par(mfrow = c(1,3))
plot(eval_clust[4,], type = 'l', main= "Davies-Bouldin index", xlab = "The number of clusters",
    ylab = "To Be Minimum")            # 2,4,6
plot(eval_clust[5,], type = 'l', main= "Modified Davies-Bouldin index", xlab = "The number of clusters",
    ylab = "To Be Minimum")           # 2,4,6
plot(eval_clust[6,], type = 'l', main= "Dunn index", xlab = "The number of clusters",
    ylab = "To Be Maximum")            # 2,7,

dtw_cluster = tsclust(sample, k=6L, distance='dtw_basic', type='partitional')
plot(dtw_cluster, type = "sc")
t(cbind(sample[,0], cluster = dtw_cluster@cluster))
cvi(dtw_cluster)       #cluster validity indices
```

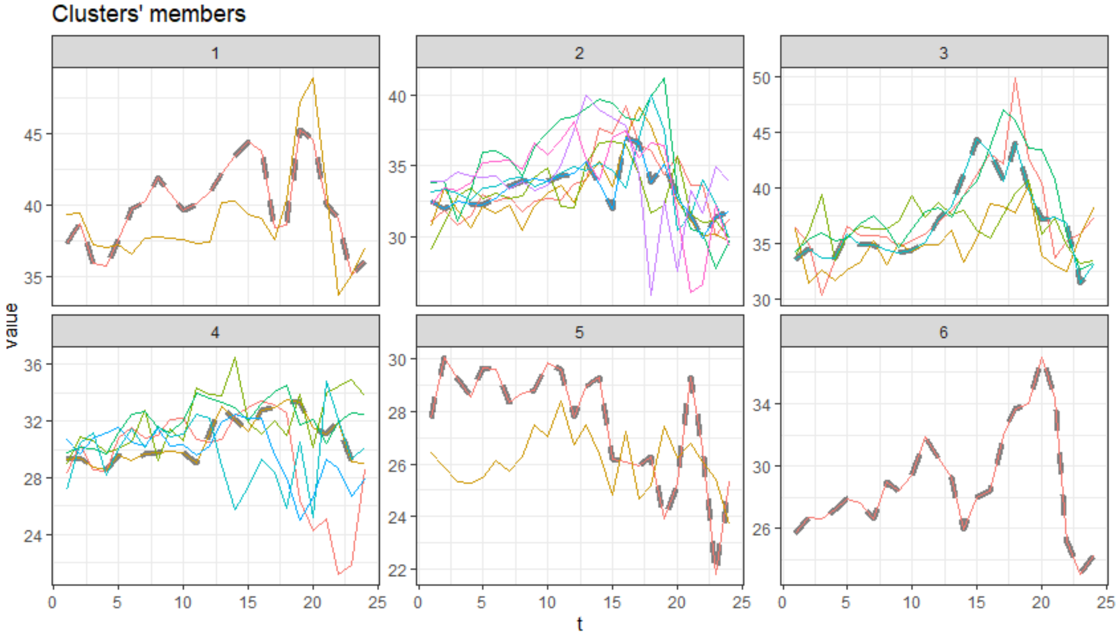
3. Using the 'dtwclust' package in R for time-series clustering

- Hourly charging behavior (hierarchical)







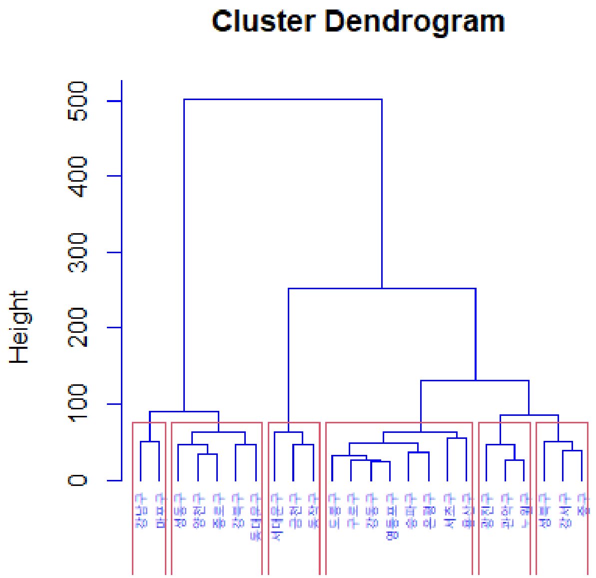


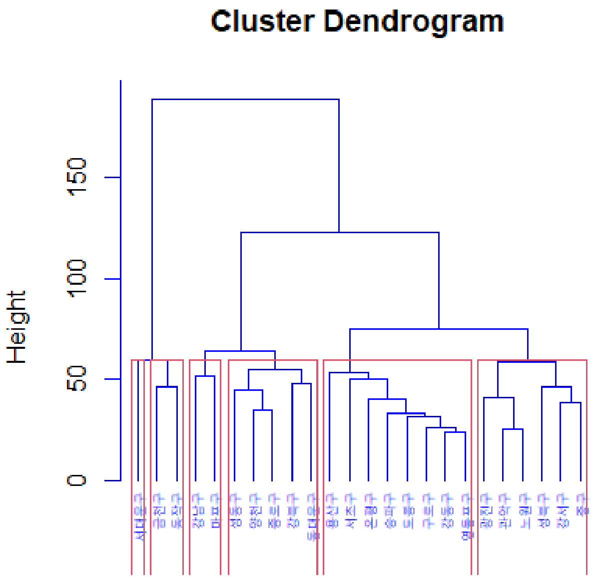

```
> cl= slot(dtw_hr, "cluster");cl        #구별로 그룹지어진것 결과
  강남구   강동구   강북구   강서구   관악구   광진구   구로구   금천구   노원구
       1        2        3        4        4        4        2        5        4
  도봉구 동대문구   동작구   마포구 서대문구   서초구   성동구   성북구   송파구
       2        3        5        1        6        2        3        4        2
  양천구 영등포구   용산구   은평구   종로구     중구
       3        2        2        2        3        4
> cvi(dtw_hr)
      Sil        SF        CH        DB    DBstar         D       COP
0.1859711 0.0000000 6.7875641 0.9424285 1.0592810 0.4025534 0.1348489
```

dtw_hr <- tsclust(sample, k = 2L:10L, distance = "dtw_basic", type = "hierarchical")
eval_clust <- sapply(dtw_hr, cvi)
plot(eval_clust[4,], type = 'l', main= "Davies-Bouldin index", xlab = "The number of clusters",
  ylab = "To Be Minimum")   #5 min
plot(eval_clust[5,], type = 'l', main= "Modified Davies-Bouldin index", xlab = "The number of clusters",
  ylab = "To Be Minimum")   # 5
plot(eval_clust[6,], type = 'l', main= "Dunn index", xlab = "The number of clusters",
  ylab = "To Be Maximum")   # 3 or 6

dtw_hr = tsclust(sample, k=6L, distance='dtw_basic', type='hierarchical')
plot(dtw_hr, type = "sc")
cl= slot(dtw_hr, "cluster");cl
cvi(dtw_hr)

3. Using the ‘dtwclust’ package in R for time-series clustering
- Cluster Dendrogram





#Dendrogram -> Calculate distances
distance <- dist(sample, method = "DTW")   #DTW distance
hccm <- hclust(distance, method = 'complete ') #최장거리법, 완전기준법
hcav <- hclust(distance, method = 'average ')   #평균기준법
par(mfrow = c(1,2))
plot(hccm, cex = 0.7,hang = -1,col = 'blue'); rect.hclust(hccm,k = 6) #5개 최적
plot(hcav, cex = 0.7,hang = -1,col = 'blue'); rect.hclust(hcav,k = 6) #5개 최적

3. Using the ‘dtwclust’ package in R for time-series clustering

- Descriptive of EV charging Daily behavior

```
> tst
# A tibble: 11,373 x 4
# Groups:   SIG [25]
   SIG    date       acvol_mean volh_mean
   <chr>  <date>          <dbl>     <dbl>
 1 강남구 2021-01-01       17.2      33.9
 2 강남구 2021-01-02       18.1      31.0
 3 강남구 2021-01-03       18.3      36.6
 4 강남구 2021-01-04       16.6      30.3
 5 강남구 2021-01-05       16.6      30.8
 6 강남구 2021-01-06       15.1      28.2
 7 강남구 2021-01-07       16.4      27.3
 8 강남구 2021-01-08       14.4      27.3
 9 강남구 2021-01-09       15.8      29.7
10 강남구 2021-01-10       14.7      28.2
```

```
> head(df_tst)
                 date   강남구   강동구   강북구   강서구   관악구   광진구
2021-01-01 2021-01-01 33.87746 32.81274 25.92896 25.31416 27.96048 24.40643
2021-01-02 2021-01-02 31.03489 27.03656 33.34322 27.43359 25.24615 27.98035
2021-01-03 2021-01-03 36.63122 30.05095 31.53110 28.30127 28.55091 27.36060
2021-01-04 2021-01-04 30.32488 27.66322 28.91280 24.72973 28.84736 26.94621
2021-01-05 2021-01-05 30.82253 31.81549 30.78244 25.12121 27.01670 26.00182
2021-01-06 2021-01-06 28.18136 29.35350 25.91074 25.11625 26.82866 25.71250
             구로구   금천구   노원구   도봉구 동대문구   동작구   마포구 서대문구
2021-01-01 26.74415 22.78659 31.29753 26.02126 29.99369 23.19677 31.20836 23.81848
2021-01-02 28.79512 24.36549 28.79122 29.92789 28.57524 23.88080 29.38192 25.15036
2021-01-03 25.31820 25.04276 27.66031 27.06801 28.76739 27.70732 32.39284 23.66999
2021-01-04 20.19702 20.70372 25.89898 26.72432 30.87202 21.99573 29.77159 22.80267
2021-01-05 27.88863 24.27527 29.03980 24.57938 26.14269 23.39029 30.91846 23.28399
2021-01-06 25.33519 23.57076 26.53695 25.28728 27.72697 23.64470 31.96796 23.82931
             서초구   성동구   성북구   송파구   양천구 영등포구   용산구   은평구
2021-01-01 31.05159 27.22715 20.91679 29.46378 25.42999 27.38249 25.91743 28.61709
2021-01-02 29.99999 29.76571 21.02115 26.57521 27.62879 25.92134 29.08523 32.40078
2021-01-03 26.90374 27.42288 34.03502 27.53005 24.97308 27.53858 26.13409 28.66374
2021-01-04 26.22469 27.29124 16.62907 25.51526 26.88629 24.74016 24.31556 26.48668
2021-01-05 25.66083 27.77827 19.00717 26.41748 25.17415 26.02343 36.28298 26.70648
2021-01-06 26.81508 29.63462 24.59576 24.60715 27.96416 26.62387 23.18663 31.52403
             종로구     중구   중랑구
2021-01-01 29.04649 26.89533 28.58902
2021-01-02 35.64327 27.43264 28.99104
2021-01-03 31.45767 34.65221 27.14068
2021-01-04 27.53102 26.44487 27.32789
2021-01-05 29.48521 27.38385 25.48561
2021-01-06 27.59535 24.23145 26.31973
```

Row: 365 days, col: 25gu

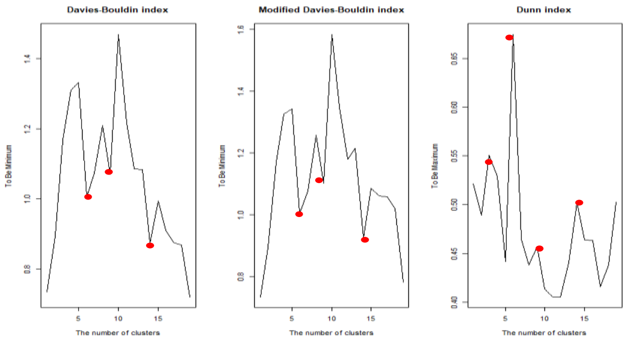
```
> head(sample)
       2021-01-01 2021-01-02 2021-01-03 2021-01-04 2021-01-05 2021-01-06
강남구   33.87746   31.03489   36.63122   30.32488   30.82253   28.18136
강동구   32.81274   27.03656   30.05095   27.66322   31.81549   29.35350
```

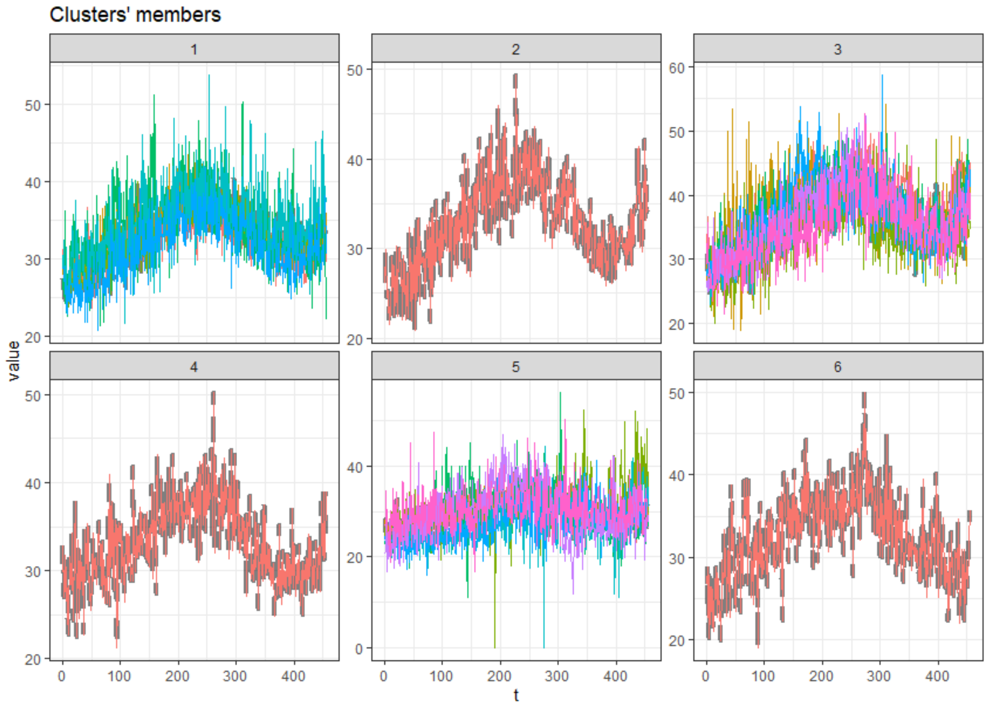
Row: 25gu, col: 365 days

```
tst <- keco%>%
  group_by(SIG,date) %>%
  summarise(acvol_mean = mean(acvol),
           volh_mean = mean(volh))      #11373 obs.
df_tst <- dcast(tst, date~SIG, value.var = "volh_mean",
              mean, fill=0)          #row; hour, col; SIG
rownames(df_tst) <- df_tst[, 1]
sample <- t(df_tst[,-1])            #row; SIG, col; hour
```

3. Using the 'dtwclust' package in R for time-series clustering

- Descriptive of EV charging Daily behavior (partitional)






```
> t(cbind(sample[,0], cluster = dtw_cluster@cluster))
        강남구 강동구 강북구 강서구 관악구 광진구 구로구 금천구 노원구 도봉구
cluster      3      4      3      5      5      5      6      5      1      2
        동대문구 동작구 마포구 서대문구 서초구 성동구 성북구 송파구 양천구
cluster        3      5      3        5      3      3      5      1      3
        영등포구 용산구 은평구 종로구 중구 중랑구
cluster        1      1      1      3    5      1
```

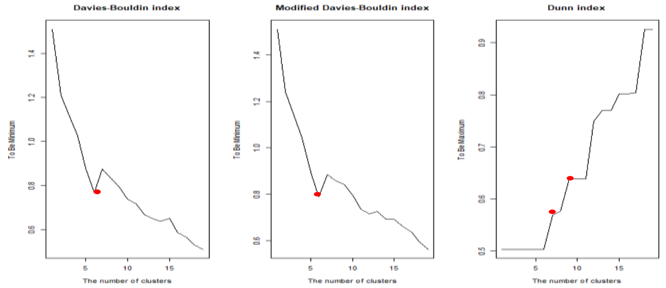
dtw <- tsclust(sample, k = 2L:20L, distance = "dtw_basic", type = "partitional")
eval_clust <- sapply(dtw, cvi)
par(mfrow = c(1,3))
plot(eval_clust[4,], type = 'l', main= "Davies-Bouldin index", xlab = "The number of clusters",
    ylab = "To Be Minimum")   # 6, 9
plot(eval_clust[5,], type = 'l', main= "Modified Davies-Bouldin index", xlab = "The number of clusters",
    ylab = "To Be Minimum")   # 6, 9
plot(eval_clust[6,], type = 'l', main= "Dunn index", xlab = "The number of clusters",
    ylab = "To Be Maximum")   # 6

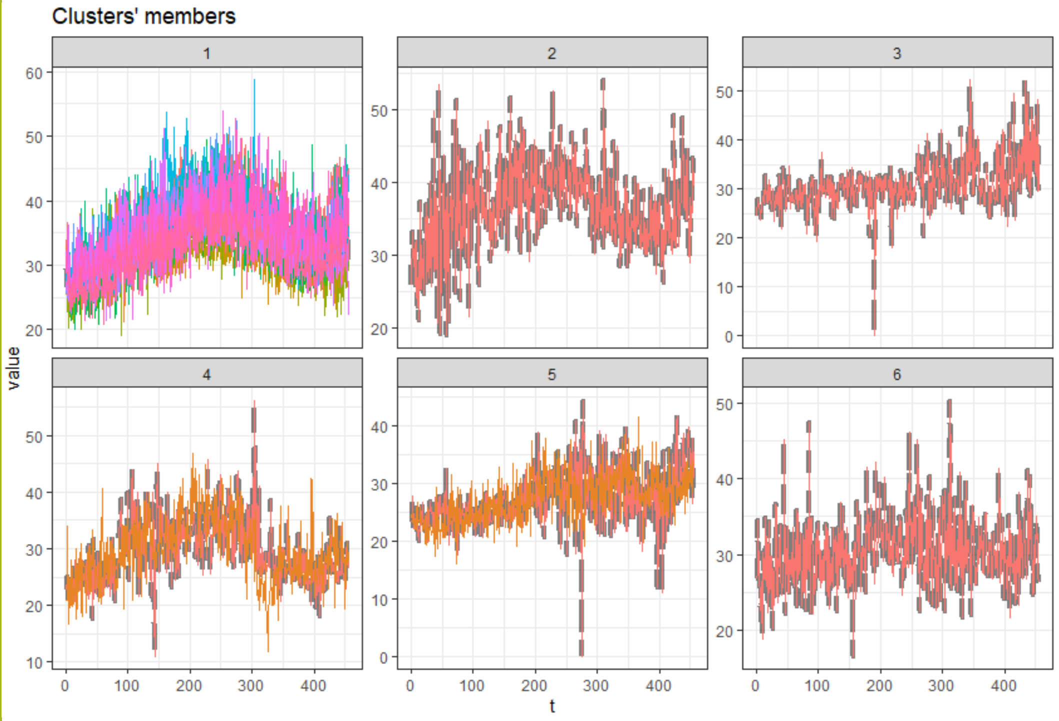
dtw_cluster = tsclust(sample, k=6L, distance='dtw_basic', type='partitional')
plot(dtw_cluster, type = "sc")
t(cbind(sample[,0], cluster = dtw_cluster@cluster))
cvi(dtw_cluster)

```
> cvi(dtw_cluster)
         Sil           SF           CH           DB       DBstar            D
-0.008355573  0.000000000  3.857220176  1.478564370  1.609740689  0.429481553
         COP
 0.479600302
```

3. Using the 'dtwclust' package in R for time-series clustering

- Descriptive of EV charging Daily behavior (hierarchical)





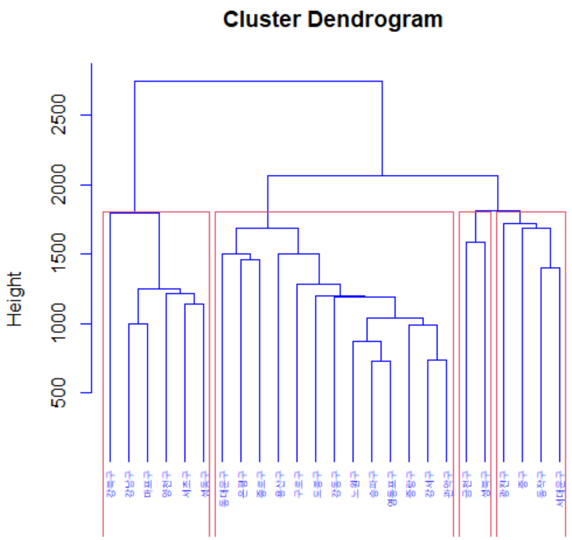

```
> t(cbind(sample[,0], cluster = dtw_hr@cluster))
        강남구 강동구 강북구 강서구 관악구 광진구 구로구 금천구 노원구 도봉구
cluster      1      1      2      1      1      3      1      4      1      1
        동대문구 동작구 마포구 서대문구 서초구 성동구 성북구 송파구 양천구
cluster        1      5      1        5      1      1      4      1      1
        영등포구 용산구 은평구 종로구 중구 중랑구
cluster        1      1      1      1    6      1
```

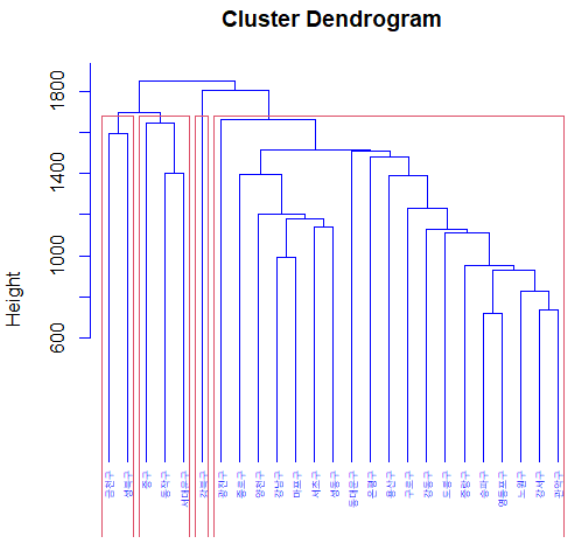
```
> cvi(dtw_hr)
      Sil        SF        CH        DB    DBstar         D       COP
0.1179848 0.0000000 1.9731974 0.8795597 0.8949299 0.5033572 0.5028716
```

```
# TYPE = Hierarchical
dtw_hr <- tsclust(sample, k = 2L:20L, distance = "dtw_basic", type = "hierarchical")
eval_clust <- sapply(dtw_hr, cvi)
plot(eval_clust[4,], type = 'l', main= "Davies-Bouldin index", xlab = "The number of clusters",
    ylab = "To Be Minimum")
plot(eval_clust[5,], type = 'l', main= "Modified Davies-Bouldin index", xlab = "The number of clusters",      ylab
= "To Be Minimum")
plot(eval_clust[6,], type = 'l', main= "Dunn index", xlab = "The number of clusters",
    ylab = "To Be Maximum")

dtw_hr = tsclust(sample, k=6L, distance='dtw_basic', type='hierarchical')
plot(dtw_hr, type = "sc")
t(cbind(sample[,0], cluster = dtw_hr@cluster))
cvi(dtw_hr)
```

```
> cvi(dtw_cluster)
        Sil          SF          CH          DB      DBstar           D         COP
-0.04787554  0.00000000  4.10333357  1.17203268  1.25131341  0.39910721  0.53959709
> cvi(dtw_hr)
      Sil        SF        CH        DB    DBstar         D       COP
0.1179848 0.0000000 1.9731974 0.8795597 0.8949299 0.5033572 0.5028716
```

3. Using the ‘dtwclust’ package in R for time-series clustering
- Cluster Dendrogram






```
#Dendrogram -> Calculate distances
distance <- dist(sample, method = "DTW")     #DTW distance
hccm <- hclust(distance, method = 'complete') #최장거리법, 완전기준법
hcav <- hclust(distance, method = 'average')  #평균기준법
par(mfrow = c(1,2))
plot(hccm, cex = 0.7,hang = -1,col = 'blue'); rect.hclust(hccm,k = 4) #5개 최적
plot(hcav, cex = 0.7,hang = -1,col = 'blue'); rect.hclust(hcav,k = 4) #4개 최적
```

Workflow/ Daily Log

Before the Matera) EV charging, 2021.01.01 ~2021.12.31., 205 charging stations,

-Requested public data open for 4years more

- DTW to cluster EV daily/ hourly charging pattern

In Matera) EV charging, 2018.01~2022.03, 800 charging stations, minute data

- 1) Average Hourly(24) EV charging, 25 Gu districts, Hourly (24*25)
- 2) Average Daily(1551) EV charging, 25 Gu districts, Daily (1551*25)
- 3) Average Daily-Hourly(24*1551) EV charging, 25 Gu (24*1551*25)

Present)

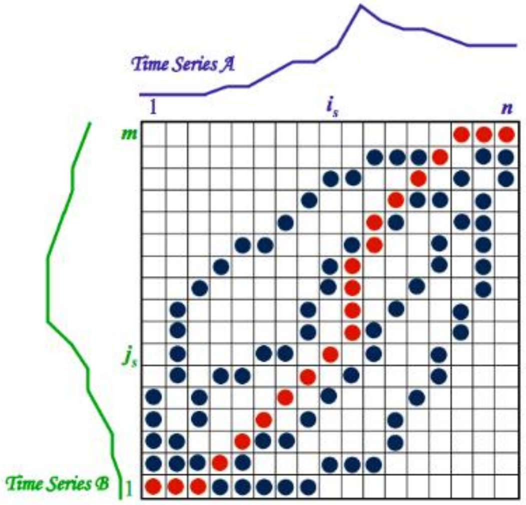
select 1 location(GN), daily(1,551), EV charging prediction by LSTM

- Epoch =50, batch_size = 1, R2=0.5, 1.xx RMSE

Goal) Train set: GA, SB // test set: JG (Group 2: GA, SB, JG)

THANK YOU

Optimisations to the DTW Algorithm



The number of possible warping paths through the grid is exponentially explosive!

reduction of the search space

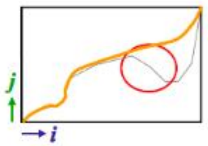
Restrictions on the warping function:

- monotonicity
- continuity
- boundary conditions
- warping window
- slope constraint.

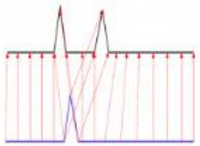
Restrictions on the Warping Function

Monotonicity: $i_{s-1} \le i_s$ and $j_{s-1} \le j_s$.

The alignment path does not go back in "*time*" index.

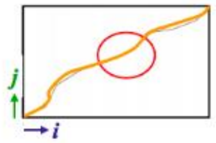


Guarantees that features are not repeated in the alignment.

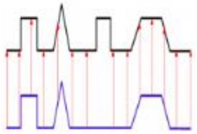


Continuity: $i_s - i_{s-1} \le 1$ and $j_s - j_{s-1} \le 1$.

The alignment path does not jump in "*time*" index.



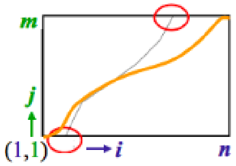
Guarantees that the alignment does not omit important features.



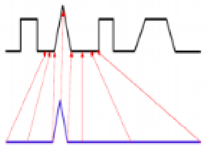
Restrictions on the Warping Function

Boundary Conditions: $i_1 = 1$, $i_k = n$ and $j_1 = 1$, $j_k = m$.

The alignment path starts at the bottom left and ends at the top right.

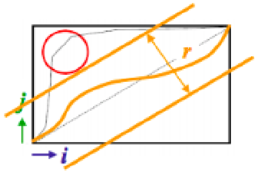


Guarantees that the alignment does not consider partially one of the sequences.



Warping Window: $|i_s - j_s| \le r$, where $r > 0$ is the window length.

A good alignment path is unlikely to wander too far from the diagonal.



Guarantees that the alignment does not try to skip different features and gets stuck at similar features.