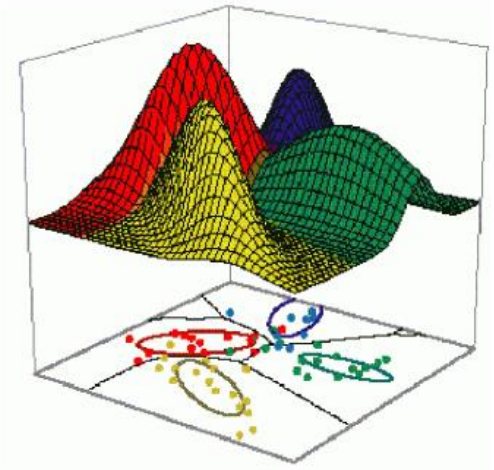
동국대학교
dongguk university

실습 5강
자기조직화지도

Codes adapted from “https://www.shanelynn.ie/self-organising-maps-for-customer-segmentation-using-r/”

실습 데이터

- 아일랜드 더블린 지역의 센서스 데이터 (2011)
- 4046 records
- 14 variables



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	id	avr_age	avr_household_size	avr_education_level	avr_num_cars	avr_health	rented_percent	unemployment_percent	internet_percent	single_percent	married_percent	separated_percent	divorced_percent	widow_percent
2	267123023	40.02811245	2.524752475	3.038461538	1.03960396	4.385542169	6.930693069	15.34391534	71	53.41365462	33.33333333	4.819277108	2.81124498	5.62248996
3	267016001	35.67365967	3.320610687	3.597701149	1.983739837	4.509433962	4.87804878	12.46105919	72.95081967	49.41724942	44.98834499	1.398601399	0.233100233	3.962703963
4	267016002	35.88235294	3.324324324	4.295302013	1.905405405	4.596638655	1.351351351	10.40462428	83.78378378	47.4789916	43.69747899	3.361344538	0	5.462184874
5	267002034	38.51666667	3.088607595	3.871794872	1.730769231	4.530172414	3.896103896	8.108108108	78.94736842	47.08333333	48.33333333	0.833333333	1.25	2.5
6	267002029	24.67800454	3.512	3.93373494	1.112	4.510344828	20.8	21.81069959	81.30081301	67.12018141	26.98412698	2.267573696	1.587301587	2.040816327
7	267002044	25.13953488	3.136363636	4.496855346	1.109090909	4.419161677	29.35779817	24.15458937	81.9047619	64.24418605	30.23255814	1.453488372	3.488372093	0.581395349
8	267002030	25.05882353	3.441176471	4.726872247	1.558823529	4.68988764	23.52941176	8.680555556	94.11764706	55.55555556	36.81917211	3.703703704	2.832244009	1.089324619
9	267002015	24.43733333	2.861538462	3.973509934	0.969230769	4.580555556	4.6875	19.45701357	81.25	70.4	21.33333333	2.933333333	3.466666667	1.866666667
10	267002017	23.50530035	2.958762887	4.674242424	1.391752577	4.685920578	20.6185567	14.04494382	91.75257732	62.54416961	32.86219081	2.473498233	1.766784452	0.35335689
11	267002018	24.67355372	2.928571429	4.530769231	1.404761905	4.579831933	14.81481481	10.8974359	82.92682927	66.52892562	29.75206612	1.652892562	1.652892562	0.41322314
12	267002007	22.14421252	2.977653631	4.18	1.05027933	4.587548638	8.426966292	22.87822878	83.8150289	71.91650854	21.63187856	3.605313093	2.277039848	0.569259962
13	267002016	22.97482838	2.913333333	4.373563218	1.14	4.563380282	1.379310345	19.4214876	89.51048951	71.39588101	23.798627	2.517162471	1.830663616	0.457665904
14	267002014	26.06349206	2.3875	5.089285714	1.1875	4.60989011	27.27272727	13.86861314	80.26315789	67.1957672	28.04232804	2.645502646	1.058201058	1.058201058
15	267002033	31.42060086	3.178082192	4.125	1.767123288	4.600858369	10.95890411	15.88235294	93.15068493	54.93562232	41.63090129	2.145922747	0.429184549	0.858369099
16	267002012	23.52536232	3.26744186	4.621621622	1.302325581	4.590405904	31.76470588	19.10828025	89.41176471	60.14492754	33.33333333	2.173913043	3.623188406	0.724637681

데이터 전처리

```
# 패키지 읽기
library(kohonen)
library(dummies)

# 팔레트 불러오기
pretty_palette <- c("#1f77b4", '#ff7f0e', '#2ca02c',
'#d62728', '#9467bd', '#8c564b', '#e377c2')

# 데이터 불러오기
data <- read.csv('dublin.csv', header=TRUE)

# 분석에 사용할 변수 선정
data_train <- data[, c(2,4,5,8)]

# 정규화 후, 매트릭스 만들기
data_train_matrix <- as.matrix(scale(data_train))
```

SOM 학습

```
# 20 x 20

Som_grid <- somgrid(xdim = 20, ydim=20, topo="hexagonal")
```

※ somgrid() 함수의 주요 요소

- xdim: x 차원 수
- ydim: y 차원 수
- Topo: 형태 (“hexagonal”, “rectangular”)

SOM 학습

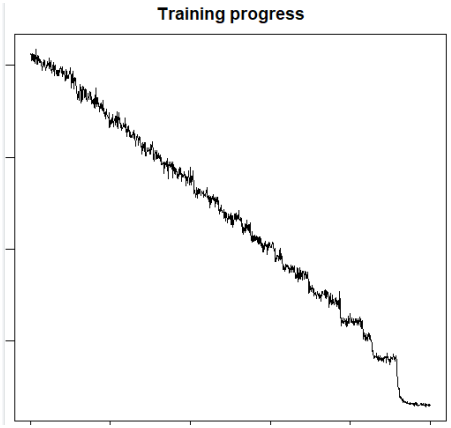
```
# Train the SOM model
set.seed(31)
som_model <- som(data_train_matrix,
                 grid=som_grid,
                 rlen=1000,
                 alpha=c(0.3,0.01),
                 keep.data = TRUE )
```

※ som() 함수의 주요 요소

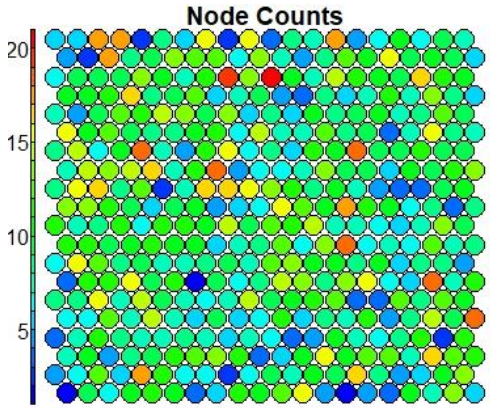
- data: 학습할 데이터
- grid: 출력층 표현 (보통 somgrid()함수의 출력값)
- rlen: 학습횟수
- alpha: 학습율 (보통 0.5 기준으로 학습횟수가 증가할수록 점진적 감소)
- radius: 이웃 노드의 반경
- Keep.data: 원 자료를 같이 저장할지 여부

시각화

```
## custom palette as per kohonen package (not compulsory)
source('coolBlueHotRed.R')
plot(som_model, type = "changes")
```

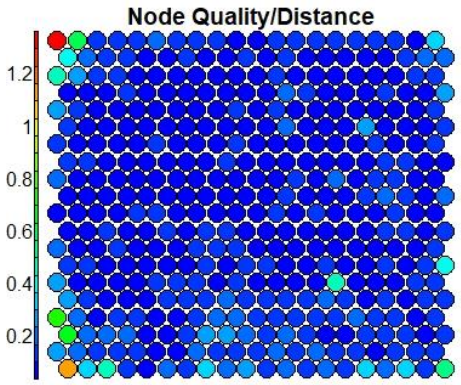



```
#counts within nodes
plot(som_model, type = "counts", main="Node Counts",
palette.name=coolBlueHotRed)
```

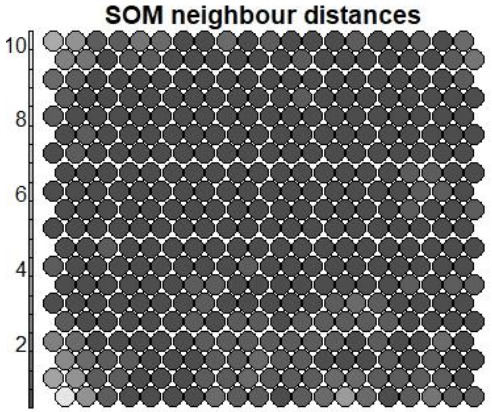


시각화

```
#map quality
plot(som_model, type = "quality", main="Node
Quality/Distance", palette.name=coolBlueHotRed)
```

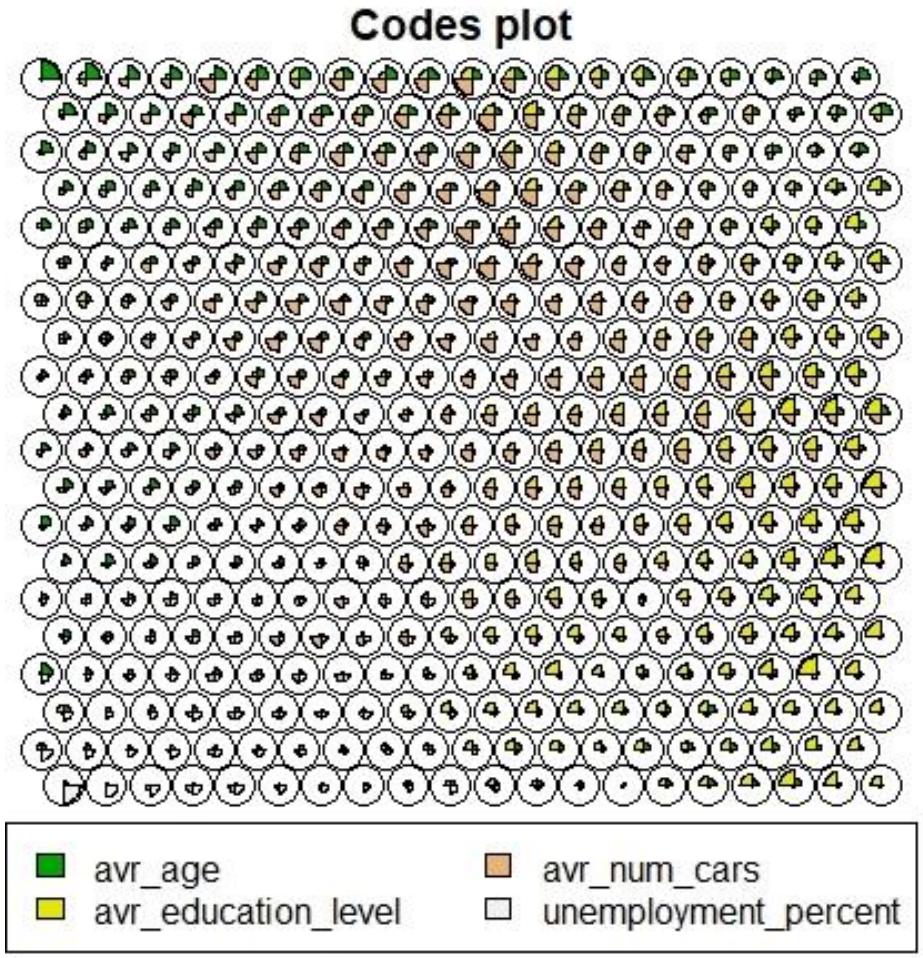



```
#neighbour distances
plot(som_model, type="dist.neighbours", main = "SOM
neighbour distances", palette.name=grey.colors)
```



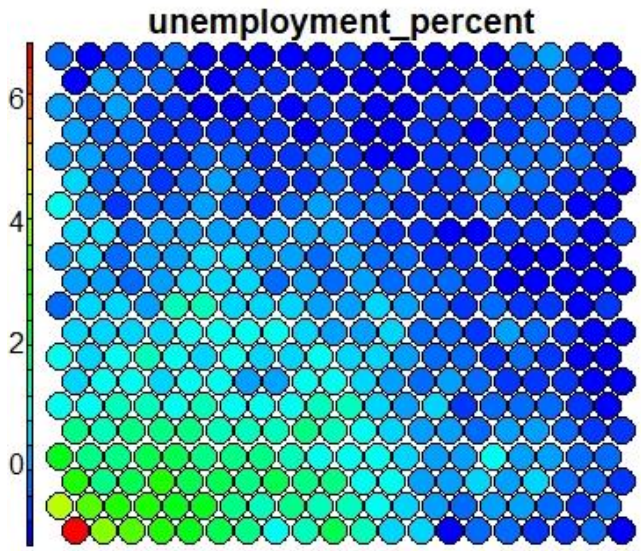
시각화

```
#code spread
plot(som_model, type = "codes")
```



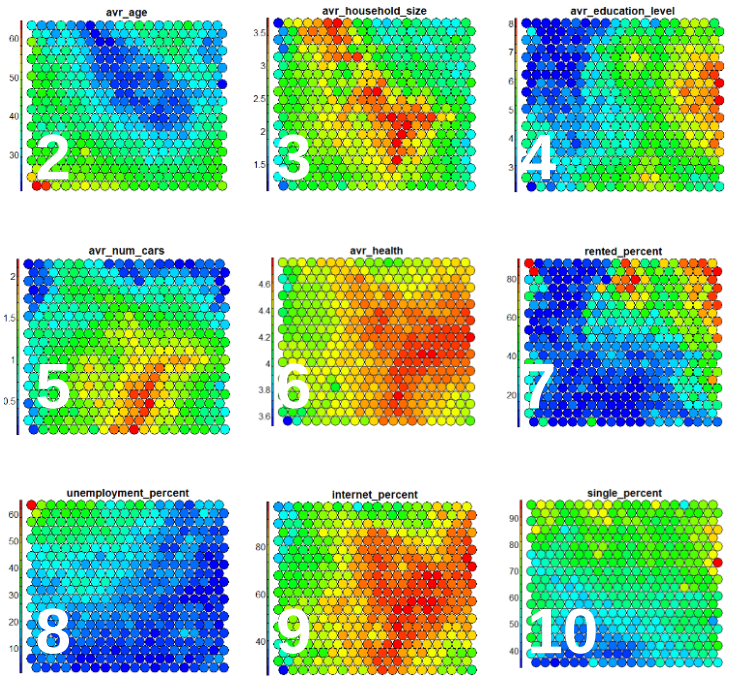
시각화

```
# Plot the heatmap for a variable at scaled / normalised
values

var <- 4 #define the variable to plot

plot(som_model,
     type = "property",
     property = getCodes(som_model)[,var],
     main=colnames(getCodes(som_model))[var],
     palette.name=coolBlueHotRed)
```



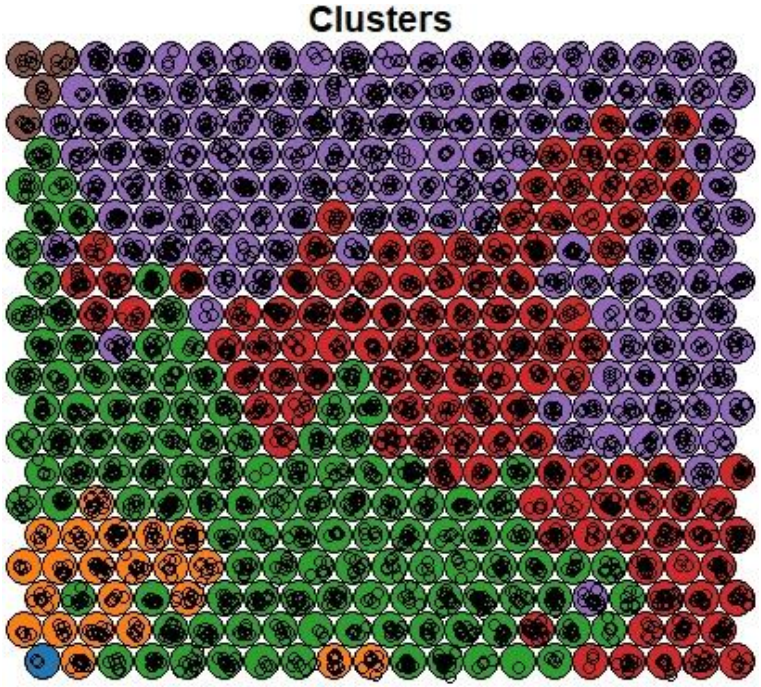
시각화

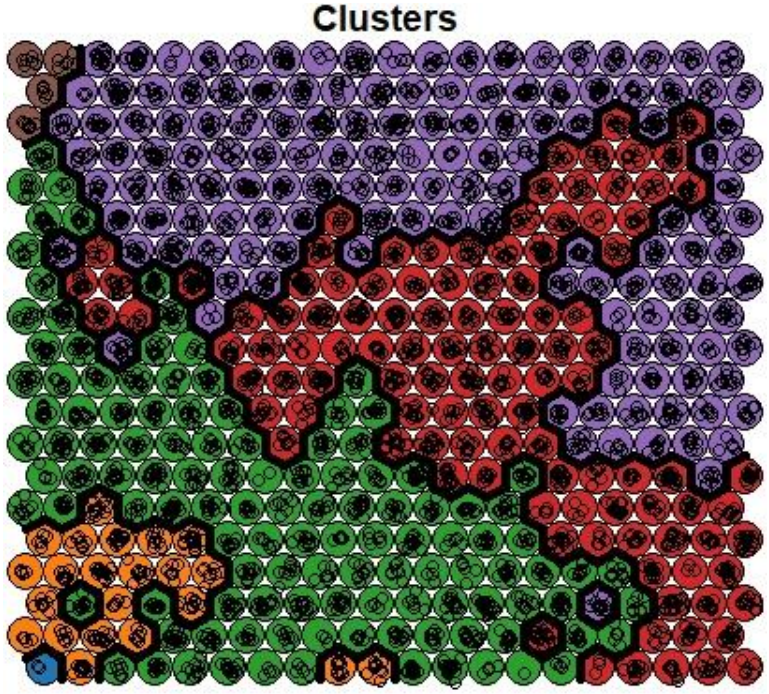
```
#plot a variable from the original data set (will be uncapped etc.)
# This function produces a menu for multiple heatmaps if a factor or
character is chosen
source('plotHeatMap.R')

# A menu of all variables should be displayed if variable=0
# (note on Mac this will required working XQuartz installation.)
plotHeatMap(som_model, data, variable=2)
plotHeatMap(som_model, data, variable=3)
plotHeatMap(som_model, data, variable=4)
plotHeatMap(som_model, data, variable=5)
plotHeatMap(som_model, data, variable=6)
plotHeatMap(som_model, data, variable=7)
plotHeatMap(som_model, data, variable=8)
plotHeatMap(som_model, data, variable=9)
plotHeatMap(som_model, data, variable=10)
plotHeatMap(som_model, data, variable=11)
plotHeatMap(som_model, data, variable=12)
plotHeatMap(som_model, data, variable=13)
plotHeatMap(som_model, data, variable=14)
```



군집화

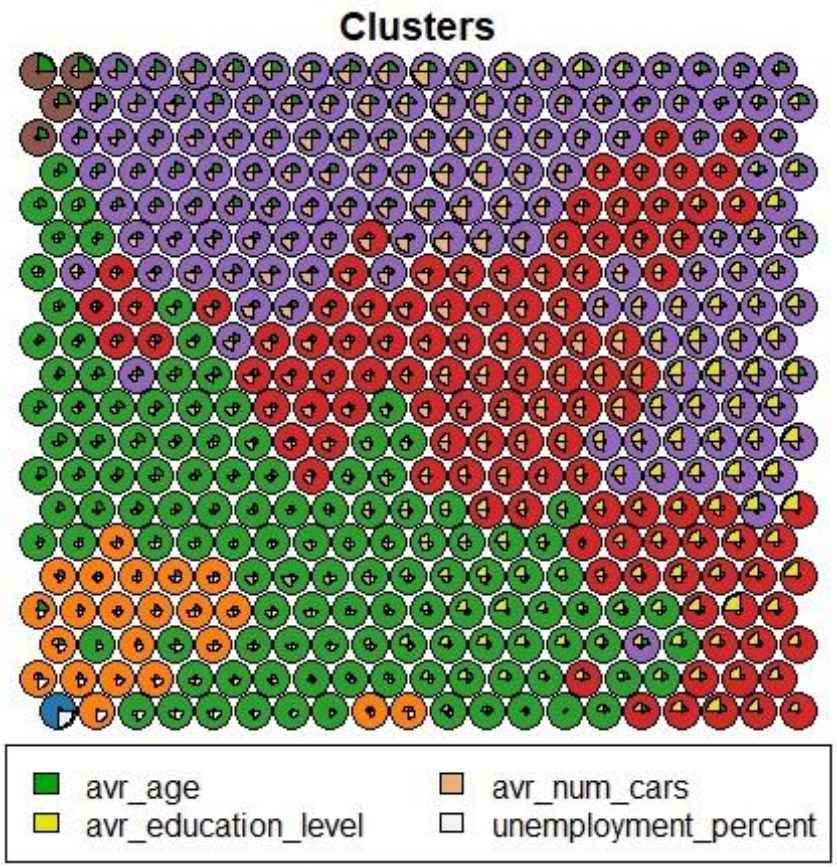
```
# Form clusters on grid
## use hierarchical clustering to cluster the codebook vectors
som_cluster <- cutree(hclust(dist(getCodes(som_model))), 6)

# Show the map with different colours for every cluster
plot(som_model, type="mapping", bgcol = pretty_palette[som_cluster],
main = "Clusters")
add.cluster.boundaries(som_model, som_cluster)
```

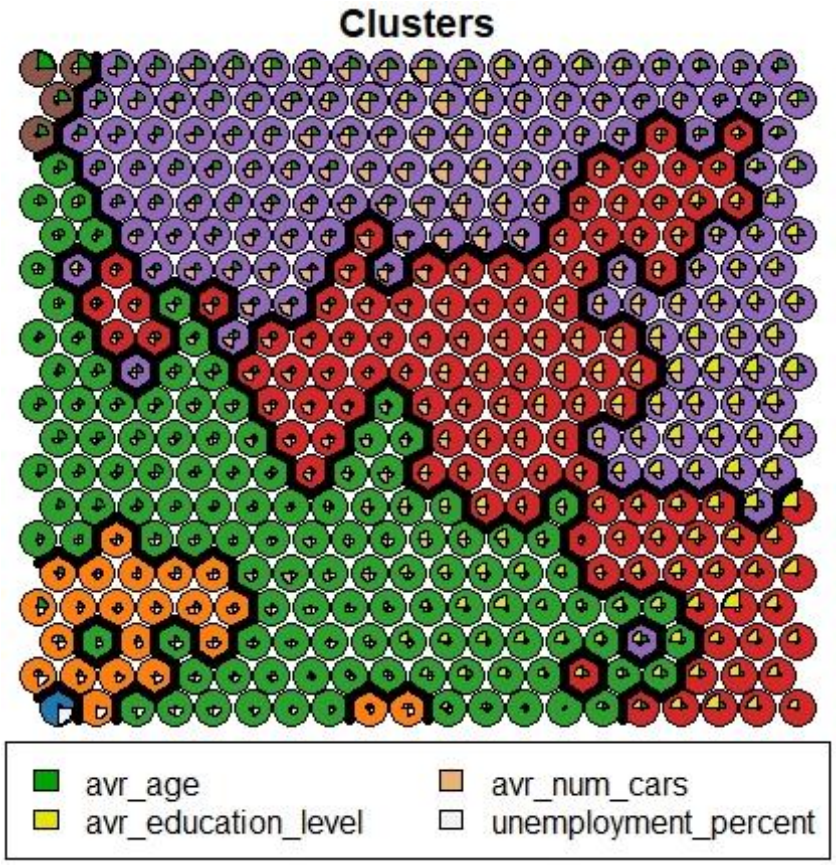




군집화

```
#show the same plot with the codes instead of just colours
plot(som_model, type="codes", bgcol = pretty_palette[som_cluster],
main = "Clusters")
add.cluster.boundaries(som_model, som_cluster)
```





군집화 결과의 시각화

Clustered Map of Dublin (example)