Self-Organising Maps for Customer Segmentation

Theory and worked examples using census and customer data sets.

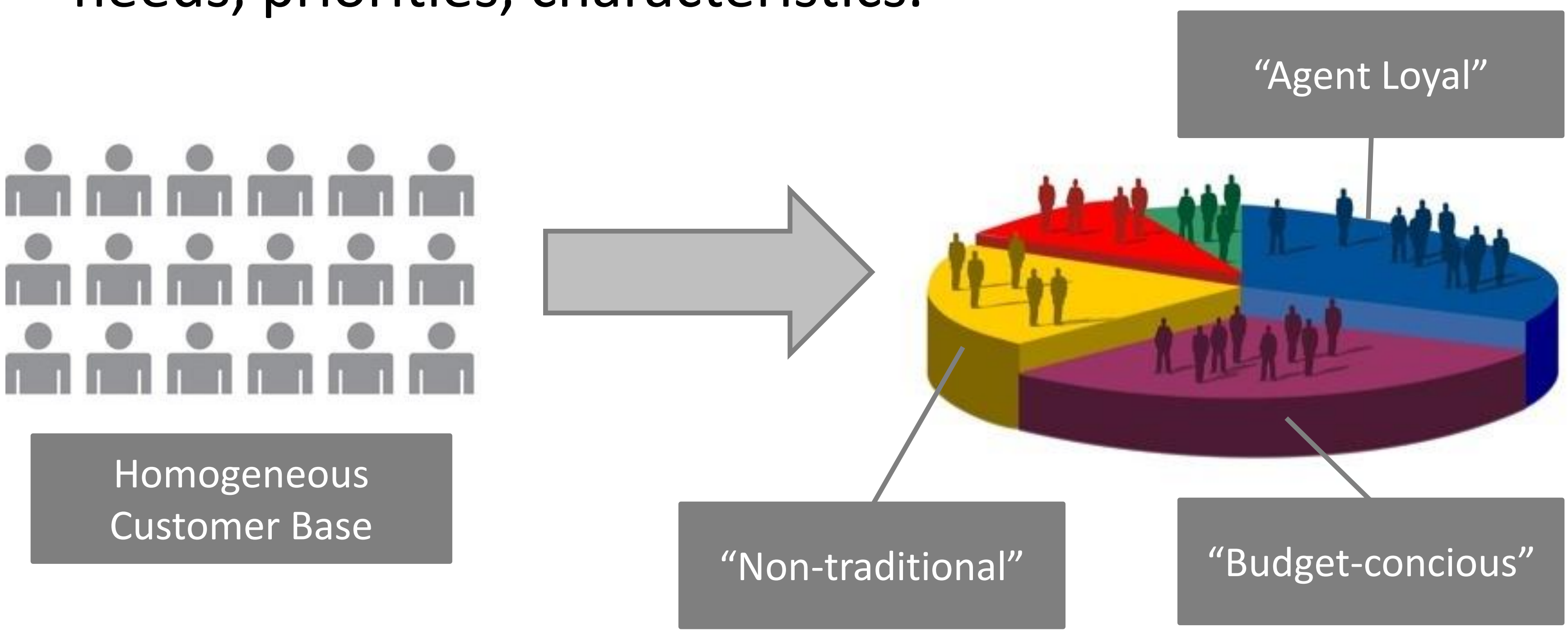
Talk for Dublin R Users Group
20/01/2014
Shane Lynn Ph.D. – Data Scientist
www.shanelynn.ie / @shane_a_lynn

Overview

1. Why customer segmentation?
2. What is a self-organising map?
   - Algorithm
   - Uses
   - Advantages
3. Example using Irish Census Data
4. Example using Ta-Feng Grocery Shopping Data

Customer Segmentation

- Customer segmentation is the application of clustering techniques to customer data
- Identify cohorts of "similar" customers – common needs, priorities, characteristics.



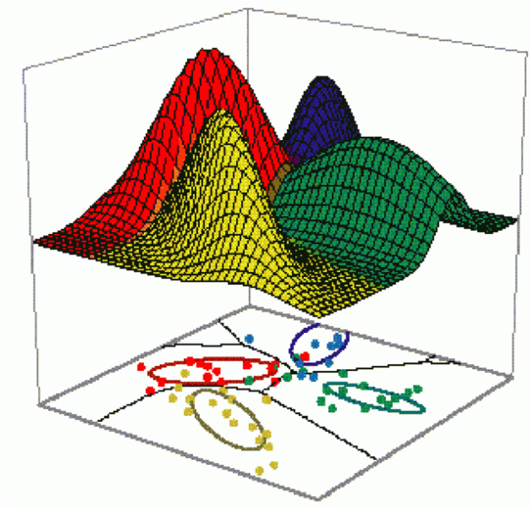
Customer Segmentation

Typical Uses

- Targeted marketing
- Customer retention
- Debt recovery

Techniques

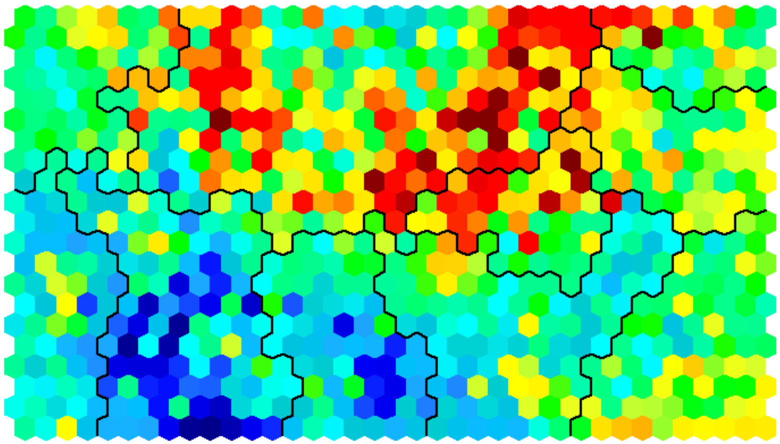
- Single discrete variable
- K-means clustering
- Hierarchical clustering
- Finite mixture modelling
- Self Organising Maps



Self-Organising Maps

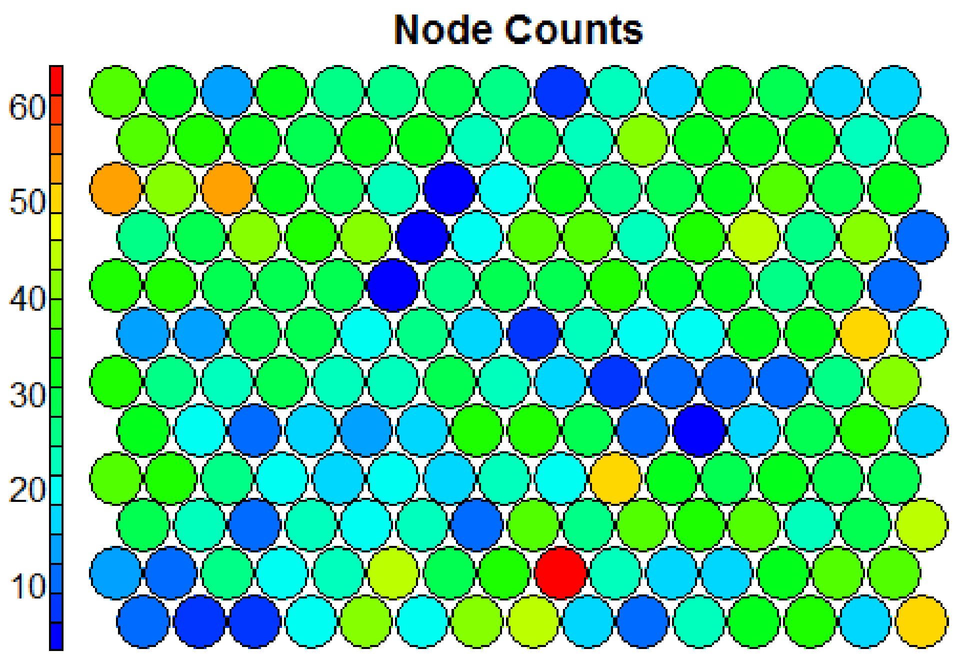
A Self-Organising Map (SOM) is a form of unsupervised neural network that produces a low (typically two) dimensional representation of the input space of the set of training samples.

- First described by Teuvo Kohonen (1982) ("Kohonen Map")
- Over 10k citations referencing SOMs – most cited Finnish scientist.
- Multi-dimensional input data is represented by a 2-D "map" of nodes
- Topological properties of the input space are maintained in map



Self-Organising Maps

- The SOM visualisation is made up of several nodes
- Input samples are "mapped" to the most similar node on the SOM. All attributes in input data are used to determine similarity.
- Each node has a weight vector of same size as the input space
- There is no variable / meaning to the x and y axes.

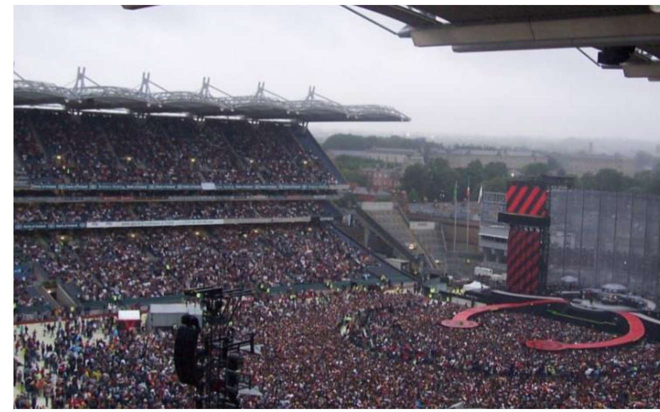


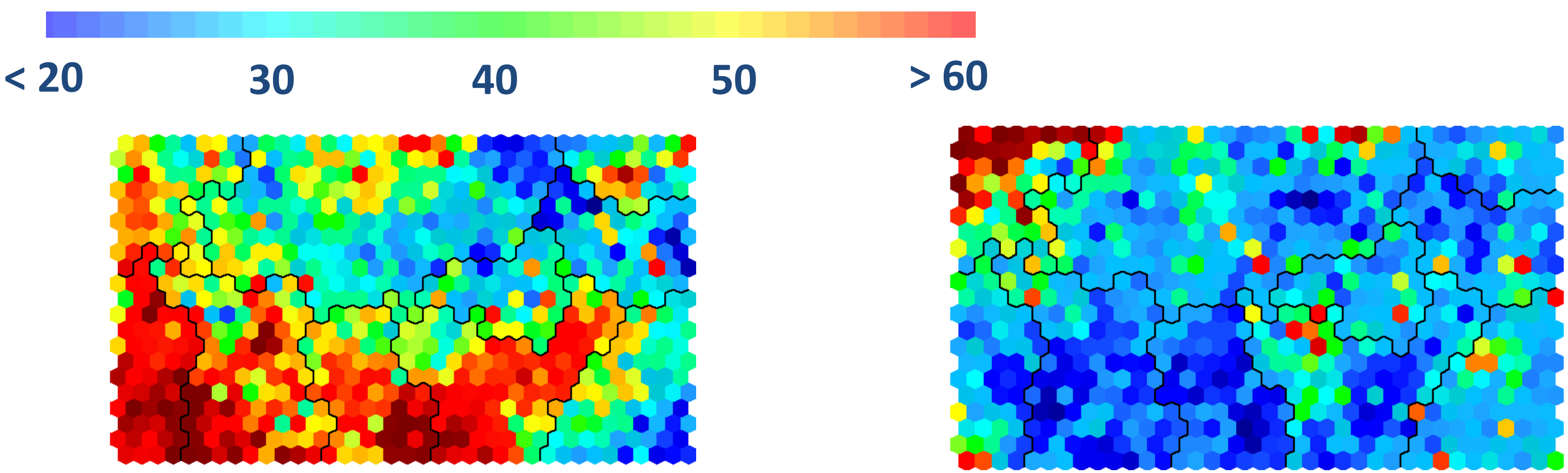
All nodes have:

- Position
- Weight Vector
- Associated input samples

Self-Organising Maps

- Everyone in this room stands in Croke Park.
- Each person compares attributes – e.g. age, gender, salary, height.
- Everyone moves until they are closest to other people with the most similar attributes. (using a Euclidean distance)
- If everyone holds up a card indicating their age – the result is a SOM heatmap





Self-Organising Maps

Algorithm

First Choose: Map size, map type

1. Initialise all weight vectors randomly.
2. Choose a random datapoint from training data and present it to the SOM
3. Find the "Best Matching Unit" (BMU) in the map – the most similar node.
4. Determine the nodes within the "neighbourhood" of the BMU.
   - The size of the neighbourhood decreases with each iteration.
5. Adjust weights of nodes in the BMU neighbourhood towards the chosen datapoint.
   - The learning rate decreases with each iteration
   - The magnitude of the adjustment is proportional to the proximity of the node to the BMU.
6. Repeat Steps 2-5 for N iterations / convergence.

$$\sigma(t)=\sigma_0 e^{(-t/\lambda)}$$

t = current iteration
λ = time constant
σ_0 = radius of the map

$$\lambda = numIterations / mapRadius$$

…nisin…

$$DistFromInput^2=\sum_{i=0}^{i=n}(I_i-W_i)^2$$

I = current input vector
W = node's weight vector
n = number of weights

…rst Choose…

1. Initialise all weight v…s randomly.
2. Choose a random datap… from training …a and present it to the SOM
3. Find the "Best Matching Unit" (…MU) in…
similar node.

New weight of a node.

$$W(t+1)=W(t)+\Theta(t)L(t)(I(t)-W(t))$$

4. Determine the nodes within the "neigh…
   – The size of the neighbourhood decreases wit… …ch iteration.
5. Adjust weights of nodes in the BMU neighbourhood towards the chosen datapoint.
   – The learning rate decreases with each iteration
   – The magnitude of the adjustment is proportional to the proximity of the node to the BMU.
6. Repeat Steps 2-5 for N iterations / converge…

Learning rate.

$$L(t)=L_0 e^{(-t/\lambda)}$$

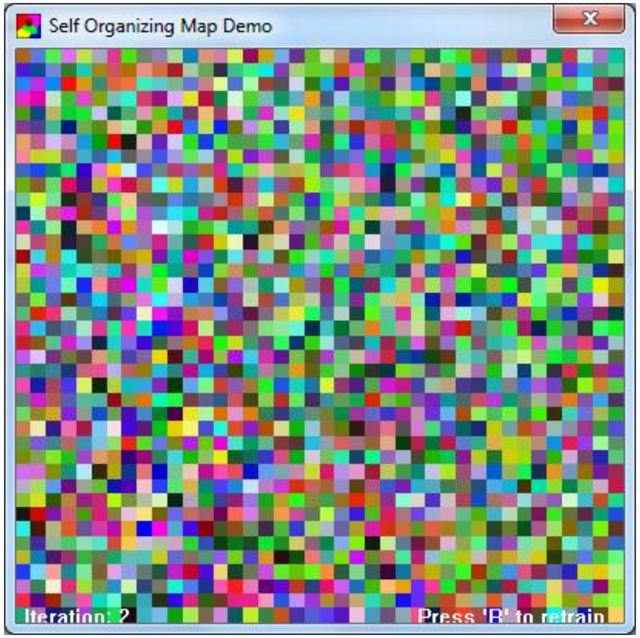
Distance from BMU.

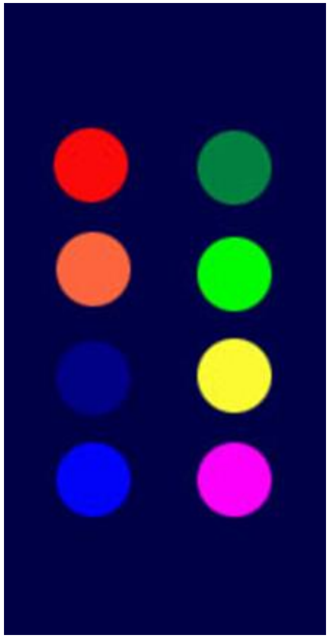
$$\Theta(t)=e^{(-distFromBMU^2/(2\sigma^2(t)))}$$

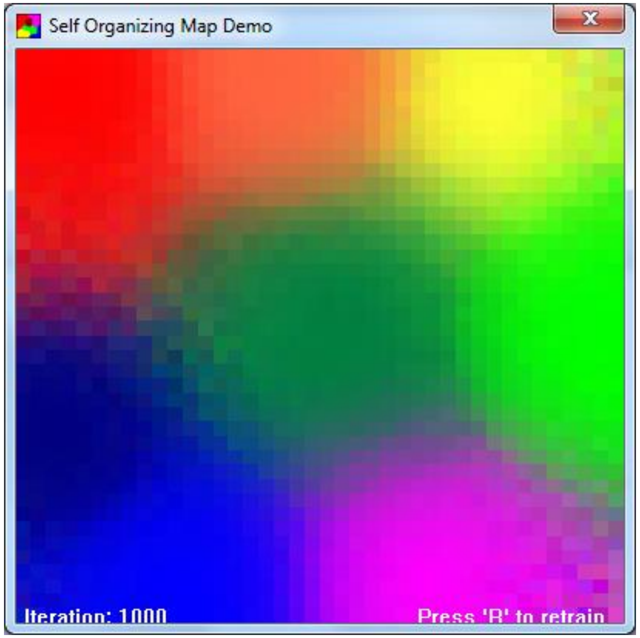
Self-Organising Maps

Example – Color Classification

- SOM training on RGB values. (R,G,B) (255,0,0)
- 3-D dataset -> 2-D SOM representation
- Similar colours have similar RGB values / similar position

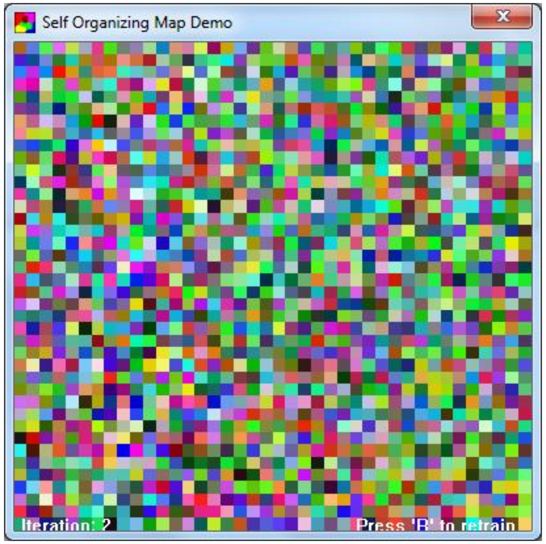


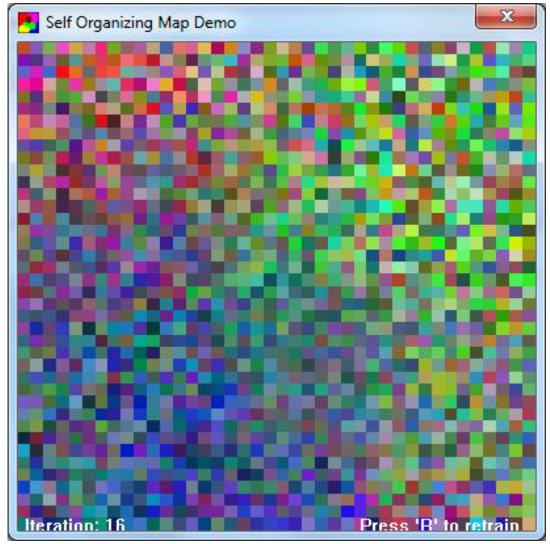


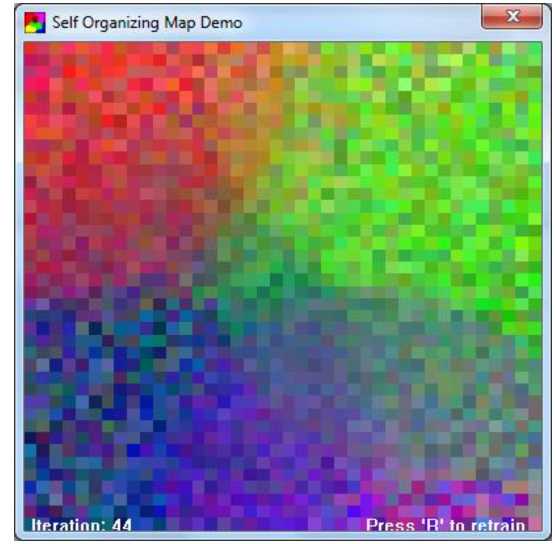


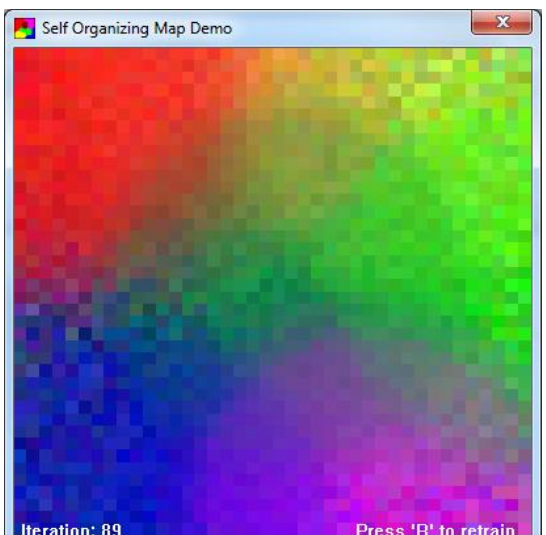
Self-Organising Maps

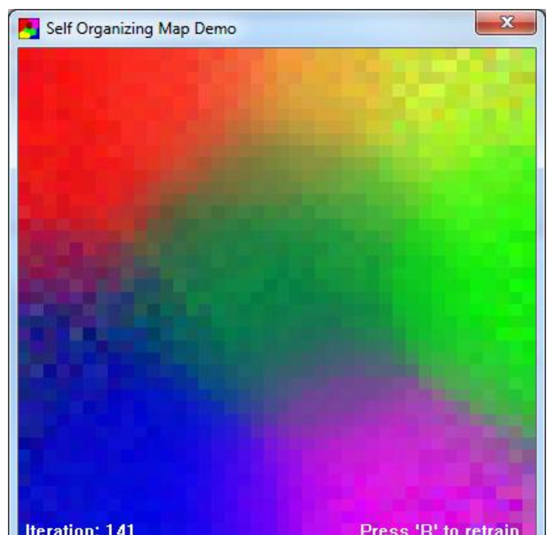
SOM Colors

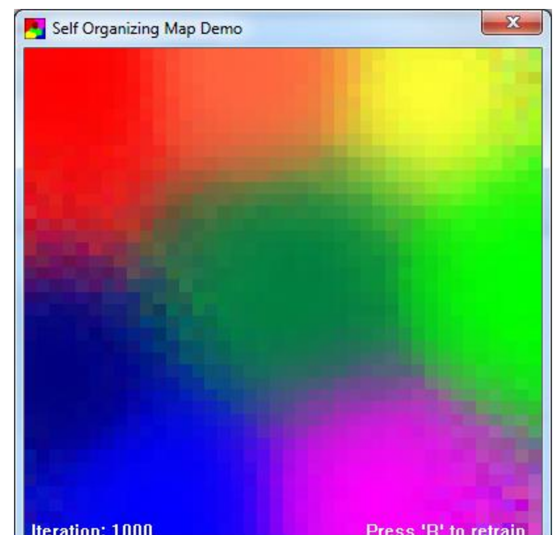








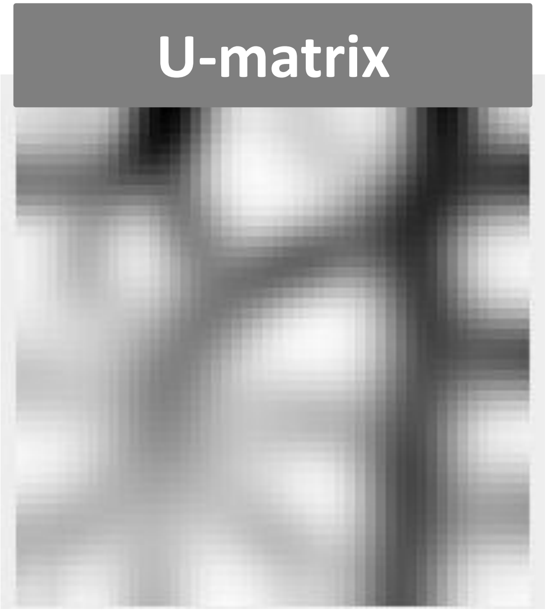




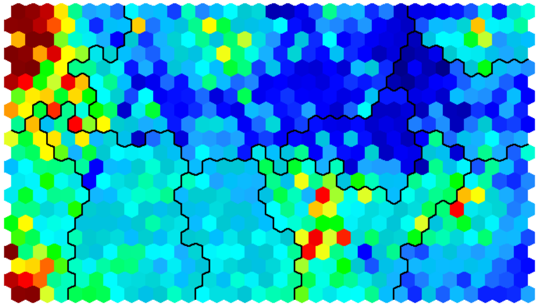
Self-Organising Maps

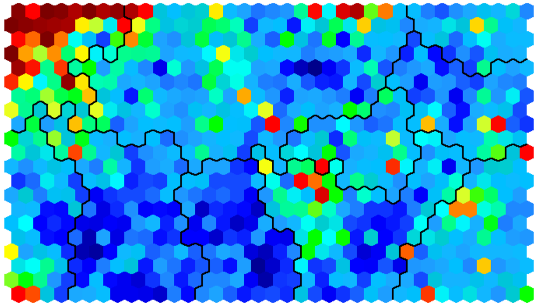
- On the SOM, actual distance between nodes is not a measure for quantitative (dis)similarity
- Unified distance matrix
  - Distance between node weights of neighbouring nodes.
  - Used to determine clusters of nodes on map

http://davis.wpi.edu/~matt/courses/soms/applet.html



Self-Organising Maps





Heatmaps

- Heatmaps are used to discover patterns between variables
- Heatmaps colour the map by chosen variables – Each node coloured using average value of all linked data points
- Can use variables not in in the training set

Dummy Variables

- SOMs only operate with numeric variables
- Categorical variables must be converted to dummy variables.

Gender
Male
Female
Female
Male

Gender_M	Gender_F
1	0
0	1
0	1
1	0

Self-Organising Maps

Data collection

Form data with samples relevant to subject

Data scaling

Center and scale data. Prevents any one variable dominating model

Data “capping”

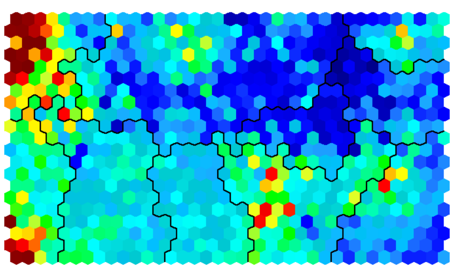
Removing outliers improves heatmap scales

Build SOM

Choose SOM parameters and run iterative algorithm

Examine heatmaps

Examine clusters



Self-Organising Maps

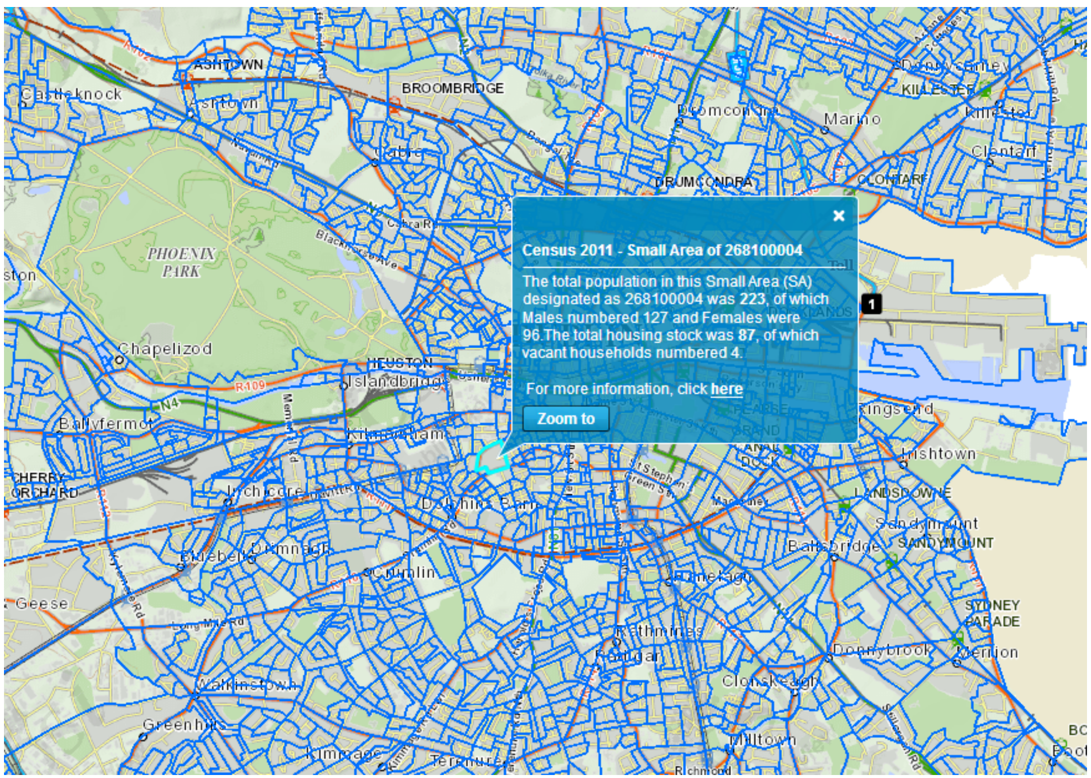
- “Kohonen” package in R
- Well documented with examples
- Key functions:
  - **somgrid()**
    - Initialise a SOM node set
  - **som()**
    - Can change – radius, learning rate, iterations, neighbourhood type
  - **plot.kohonen()**
    - Visualisation of resulting SOM
    - Supports heatmaps, node counts, properties, u-matrix etc.

Other Software

- Viscovery
- SPSS Modeler
- MATLAB (NN toolbox)
- WEKA
- Synapse
- Spice-SOM

Census Data Example

- Data from Census 2011
- Population divided in 18,488 “small areas” (SA)



- 767 variables available on 15 themes.
- Interactive viewing software online.
- Data available via csv download.
- Focusing on Dublin area for this talk.
- Aim to segment based on selected variables.

Census Data Example

```
data <- read.csv("../data/census-data-smallarea.csv")
```

- Data is formatted as person counts per SA
- Need to extract comparable statistics from count data
  - Change count data to percentages / numbers etc.

```
marital_data <- data[, c("T1_2SGLT", "T1_2MART", "T1_2SEPT",
                         "T1_2DIVT", "T1_2WIDT", "T1_2T")]
marital_percents <- data.frame(t(apply(marital_data, 1,
                          function(x) {x[1:5]/x[6]})) * 100)
```

  - Change “ranked” variables to numeric values
    i.e. Education: [None, Primary, Secondary, 3rd Level, Postgrad] changed to [0,1,2,3,4]
    Allows us to calculate “mean education level”

Census Data Example

```
# average household size
household_data <- data[, 331:339] #data on household sizes and totals
mean_household_size <- c(1:8)      #sizes 1-8 persons
results$avr_household_size <- apply(household_data, 1,
         function(x){
         size <- sum(x[1:length(mean_household_size)] *
                     mean_household_size) / x[length(x)]
         }
)
```

- Calculated features for small areas:

```
> names(data)
 [1] "id"                    "avr_age"                "avr_household_size"
"avr_education_level"  "avr_num_cars"
 [6] "avr_health"            "rented_percent"         "unemployment_percent"
"internet_percent"     "single_percent"
[11] "married_percent"       "separated_percent"      "divorced_percent"
"widow_percent"
```

- Filtered for Fingal, Dublin City, South Dublin, & Dun Laoghaire (4806 SAs)

Census Data Example

- SOM creation step

```
require(kohonen)

data_train <- data[, c(2,4,5,8)] #age, education, num cars, unemployment

data_train_matrix <- as.matrix(scale(data_train))

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

som_model <- som(data_train_matrix,
                 grid=som_grid,
                 rlen=100,
                 alpha=c(0.05,0.01),
                 keep.data = TRUE,
                 n.hood="circular" )
```

Census Data Example

Kohonen functions accept numeric matrices

- SOM creation steps

scale() mean centers and normalises all columns

```
require(kohonen)

data_train <- data[, c(2,4,5,8)] #age, education, num cars, unemployment

data_train_matrix <- as.matrix(scale(data_train))

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

som_model <- som(data_train_matrix,
                 grid=som_grid,
                 rlen=100,
                 alpha=c(0.05,0.01),
                 keep.data = TRUE,
                 n.hood="circular" )
```

Creating hexagonal SOM grid with 400 nodes (10 samples per node)

Params:
- Rlen – times to present data
- Alpha – learning rate
- Keep.data – store data in model
- n.hood – neighbourhood shape

Census Data Example

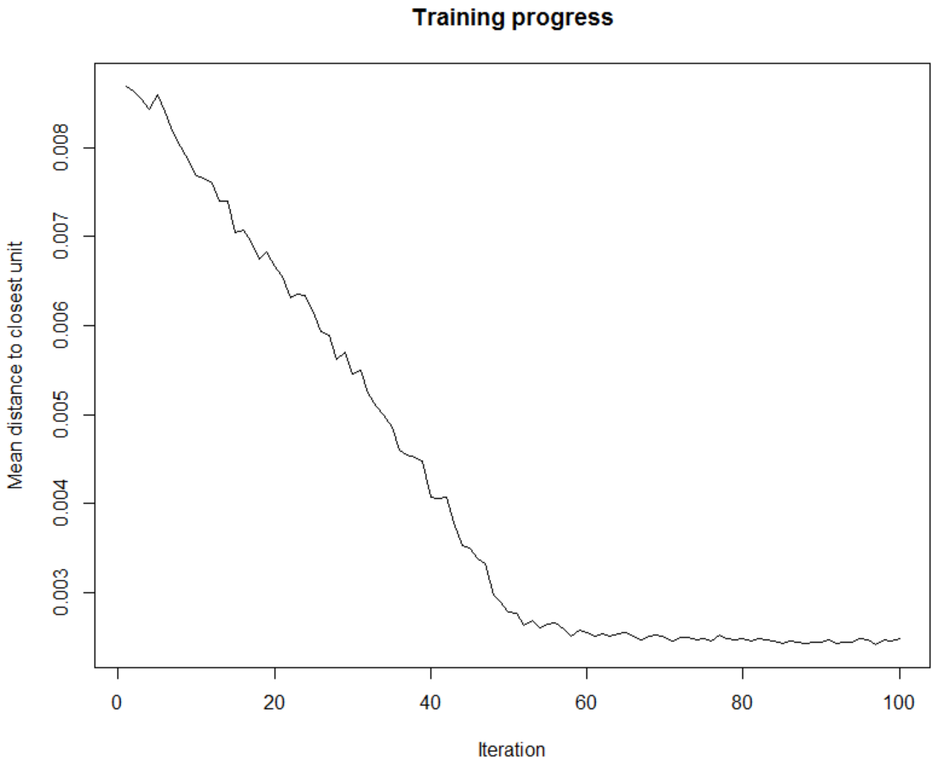
- som_model object

```
>> summary(som_model)
som map of size 20x20 with a hexagonal topology.
Training data included; dimension is 4806 by 4
Mean distance to the closest unit in the map: 0.1158323
```

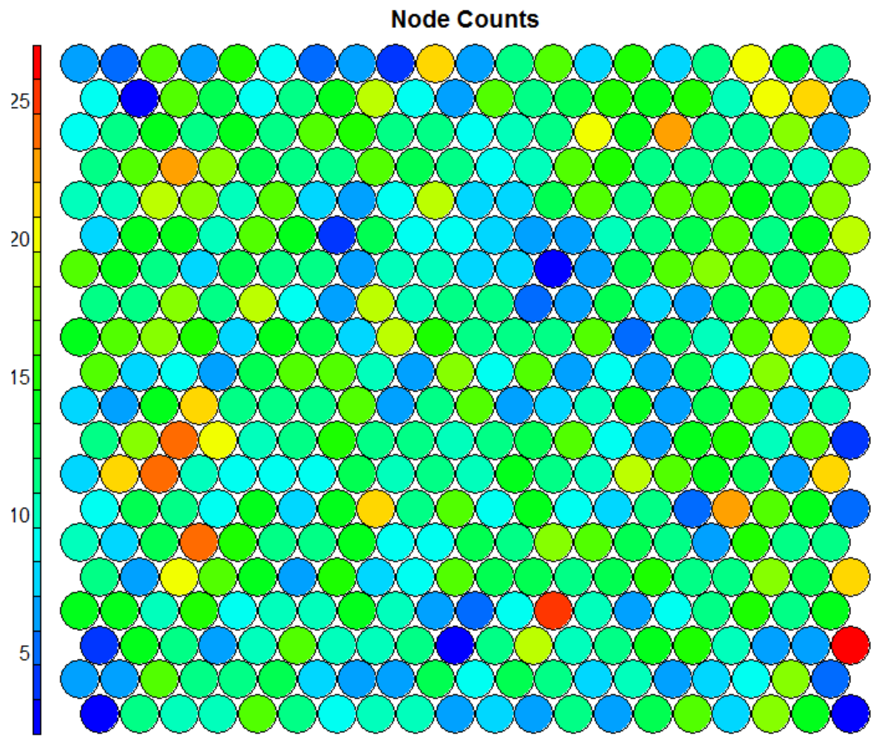
- Contains all mapping details between samples and nodes.
- Accessible via “$” operator
- Various plotting options to visualise map results.

Census Data Example

```
plot(som_model, type = "changes")
```




```
plot(som_model, type = "counts")
```

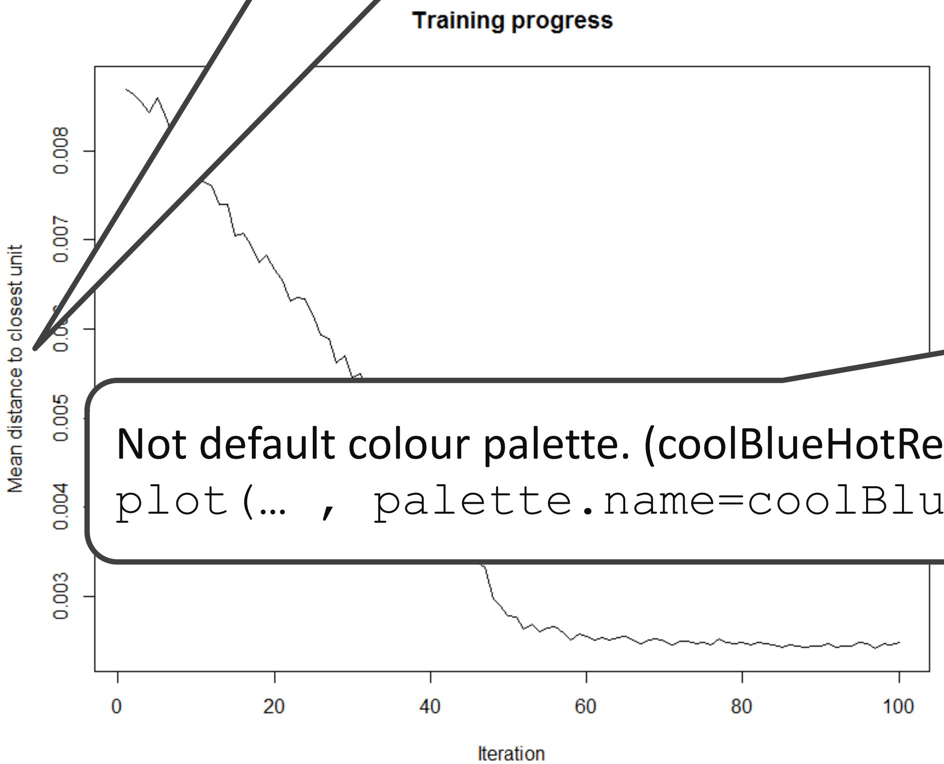


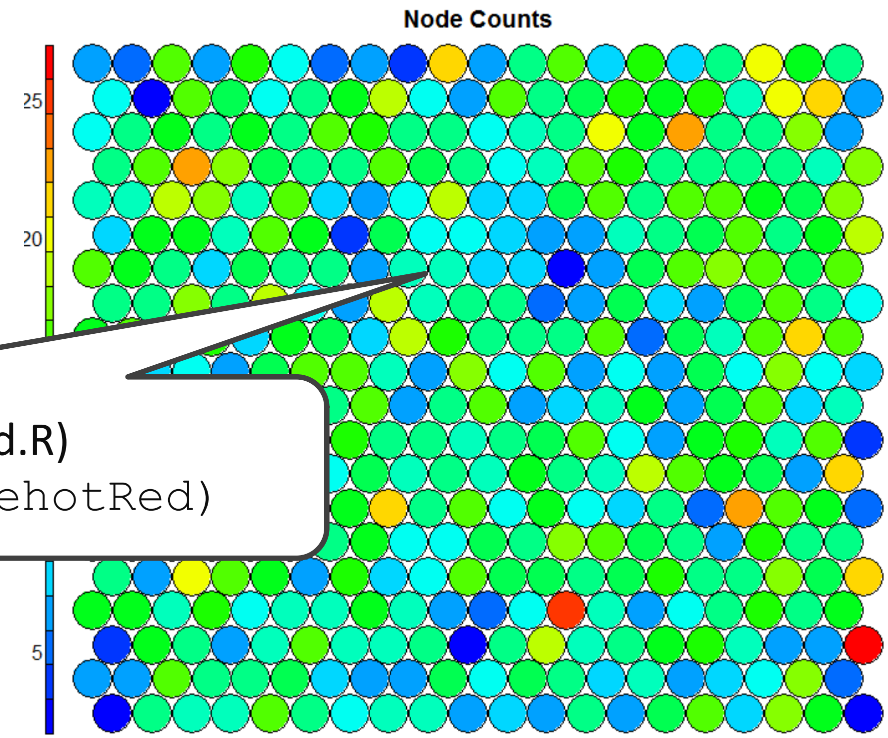
Census Data Example

Minimising “mean distance to closest unit”

`plo` … `ges")`

`plot(som_model, type = “counts")`



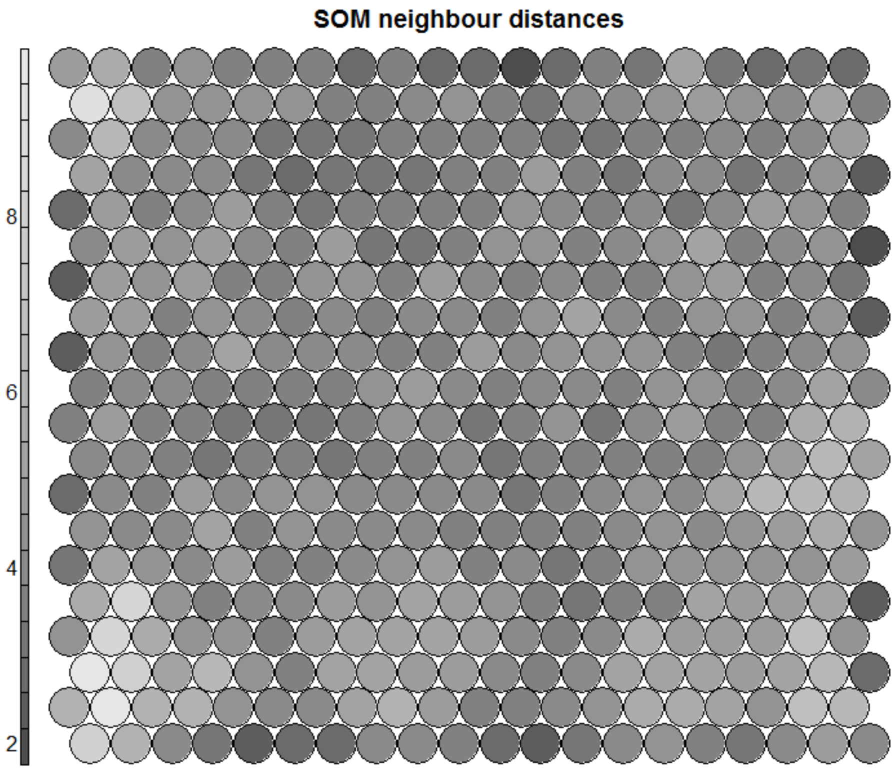


Not default colour palette. (coolBlueHotRed.R)

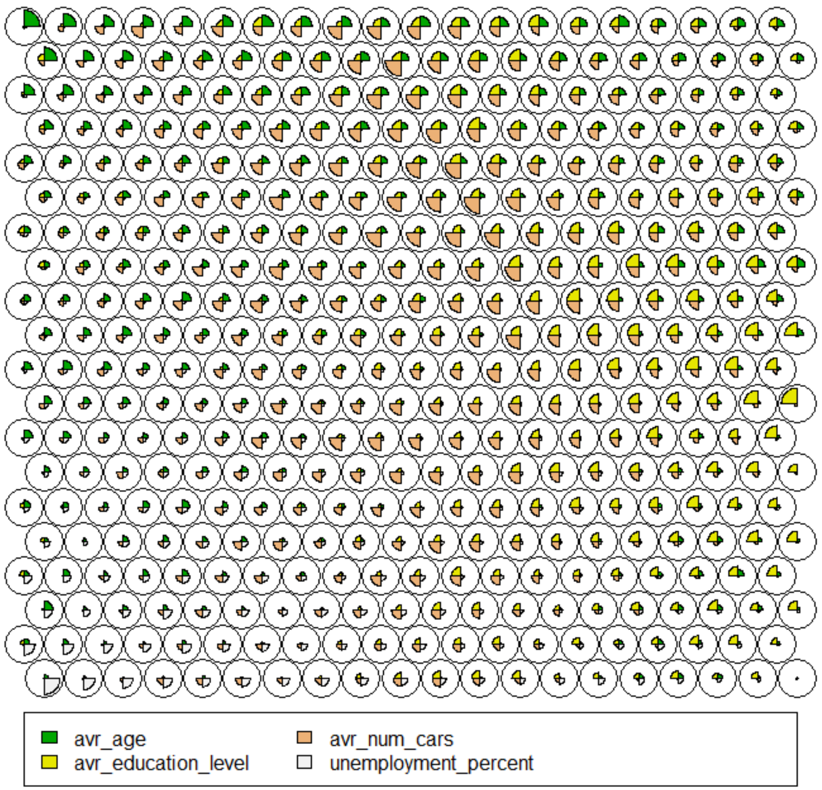
```
plot(… , palette.name=coolBluehotRed)
```

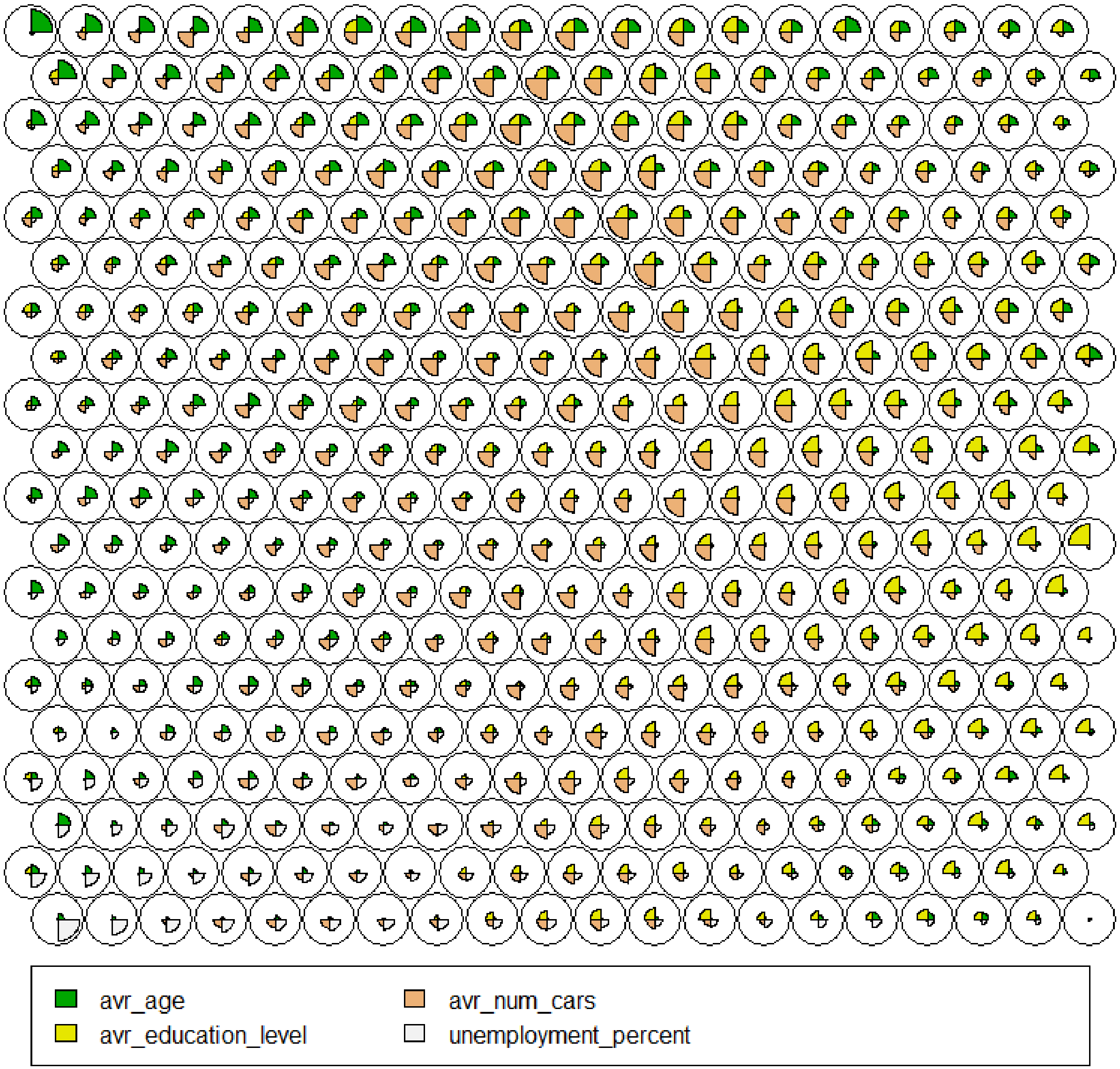
Census Data Example

```
plot(som_model, type =
"dist.neighbours")
```

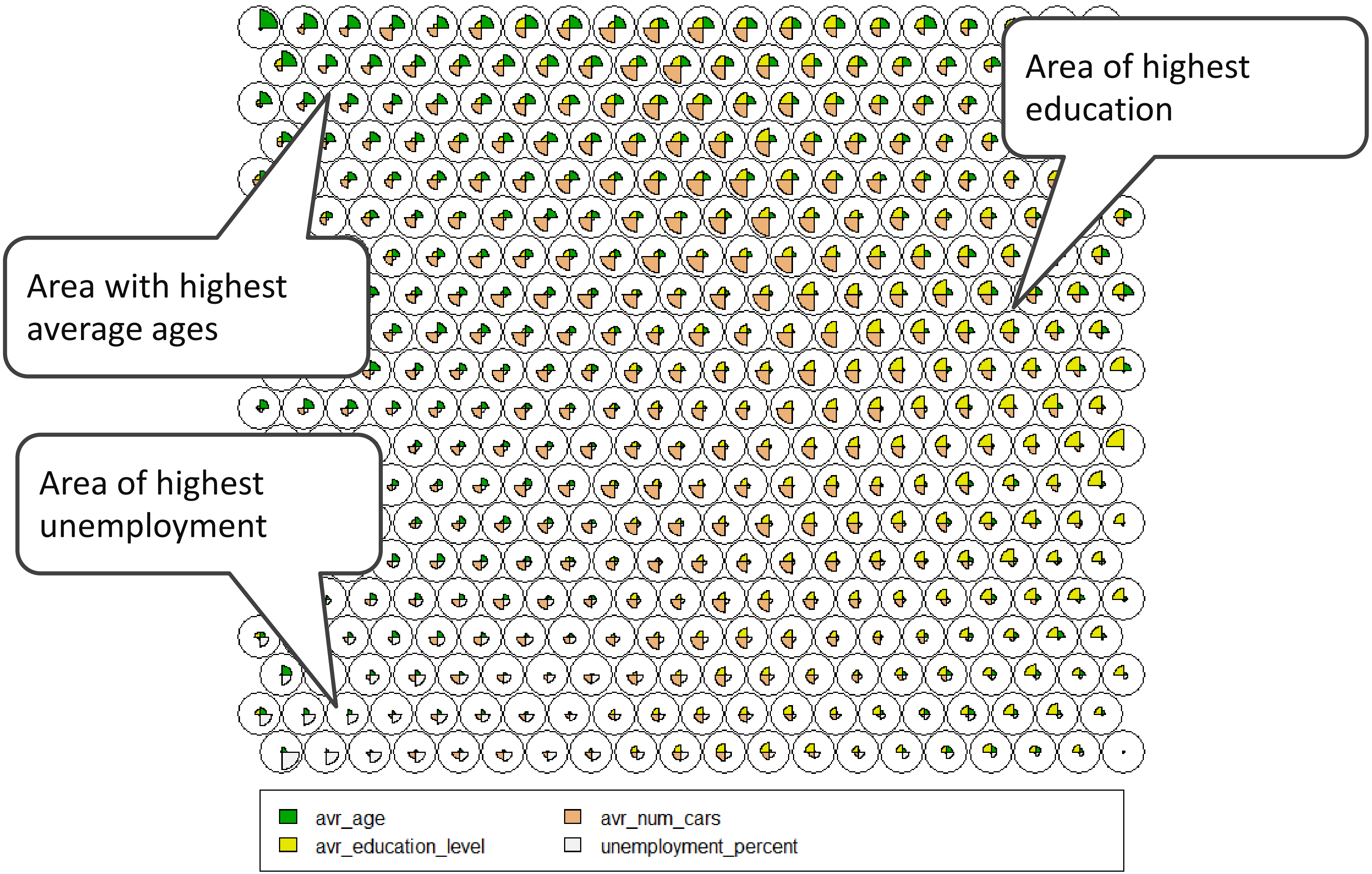



```
plot(som_model, type = "codes")
```



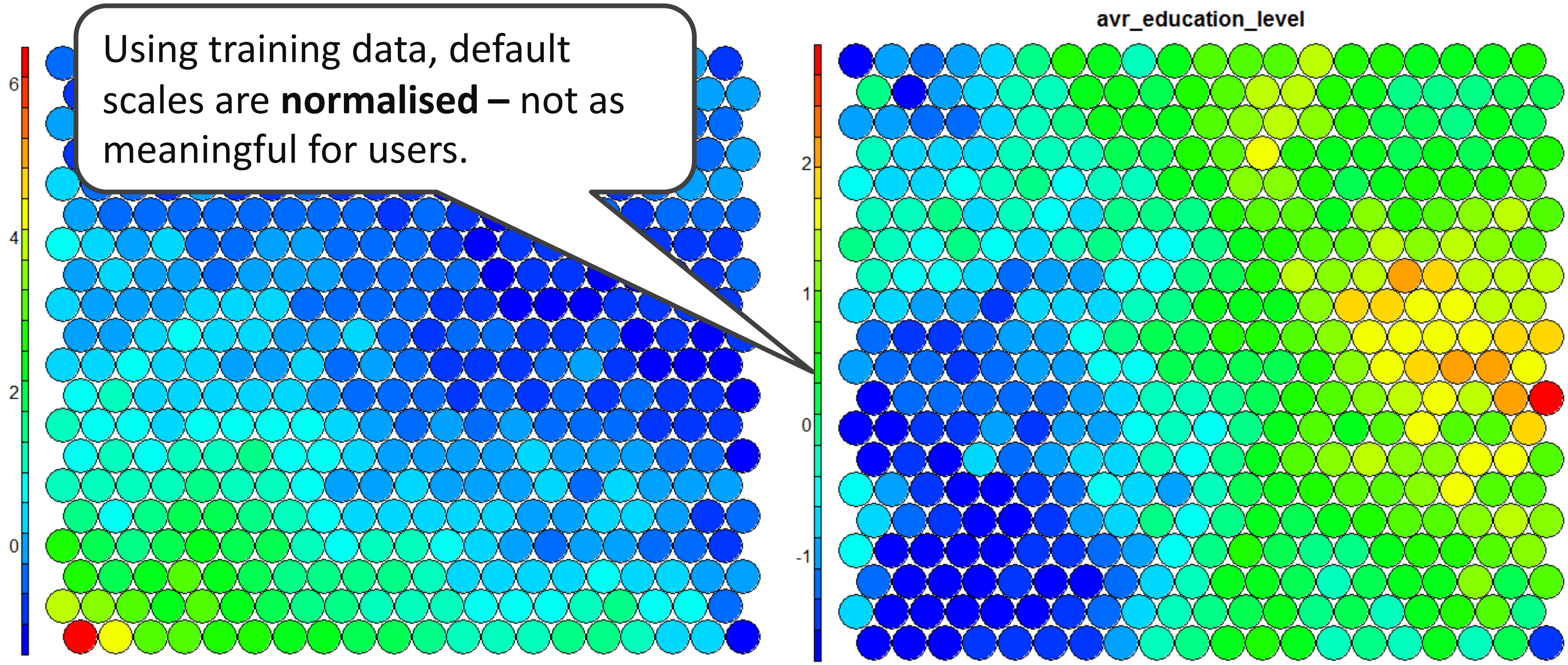


- Fan diagram shows distribution of variables across map.
- Can see patterns by examining dominant colours etc.
- This type of representation is useful for SOMs when the number of variables is less than ~ 5
- Good to get a grasp of general patterns in SOM


Area of highest education
Area with highest average ages
Area of highest unemployment
avr_age
avr_num_cars
avr_education_level
unemployment_percent

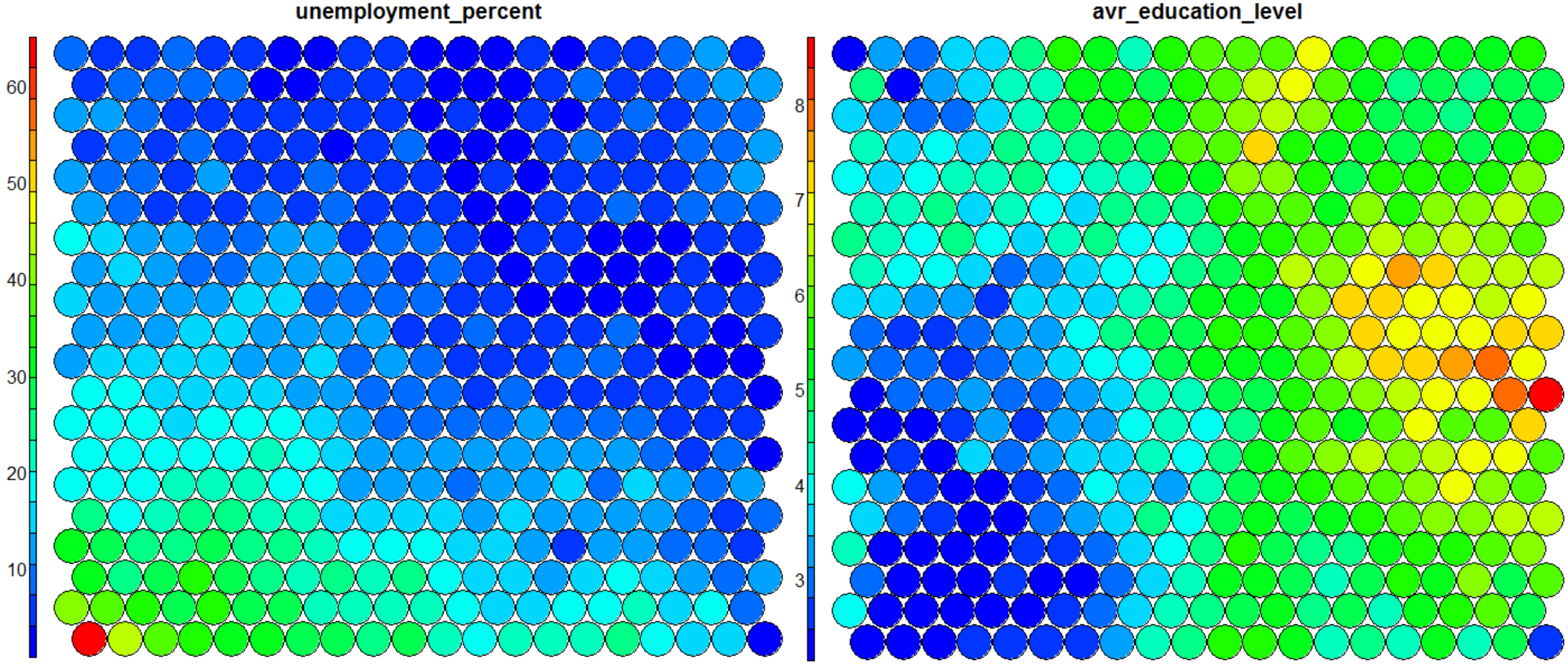
Census Data Example

```
plot(som_model, type = "property", property = som_model$codes[,4],
main=names(som_model$data)[4], palette.name=coolBlueHotRed)
```



Census Data Example

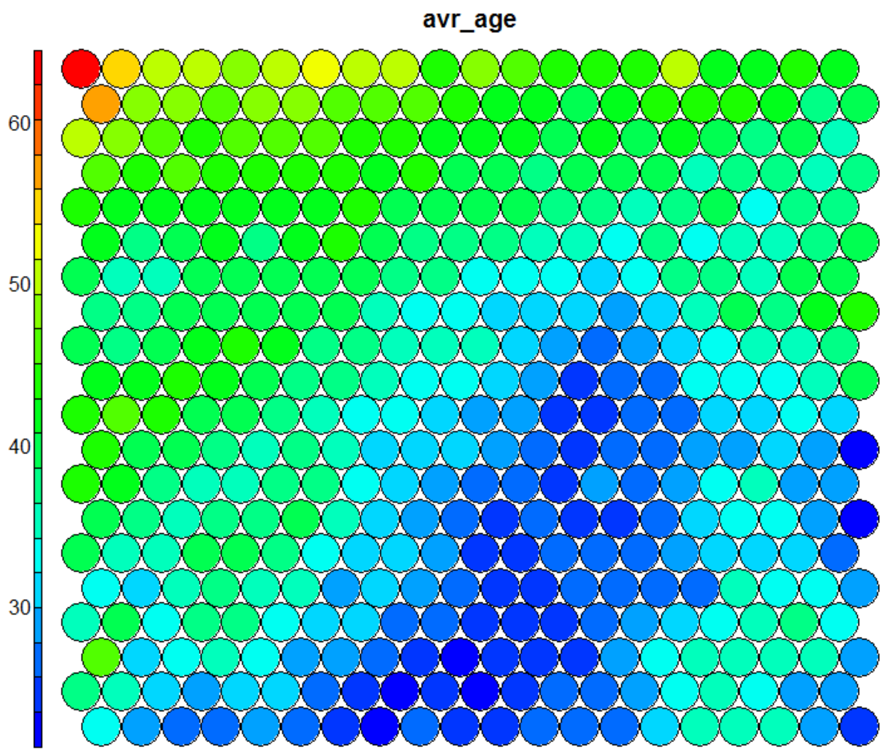
```
var <- 2 #define the variable to plot
var_unscaled <- aggregate(as.numeric(data_train[,var]),
         by=list(som_model$unit.classif),
         FUN=mean, simplify=TRUE)[,2]
plot(som_model, type = "property", property=var_unscaled,
         main=names(data_train)[var], palette.name=coolBlueHotRed)
```

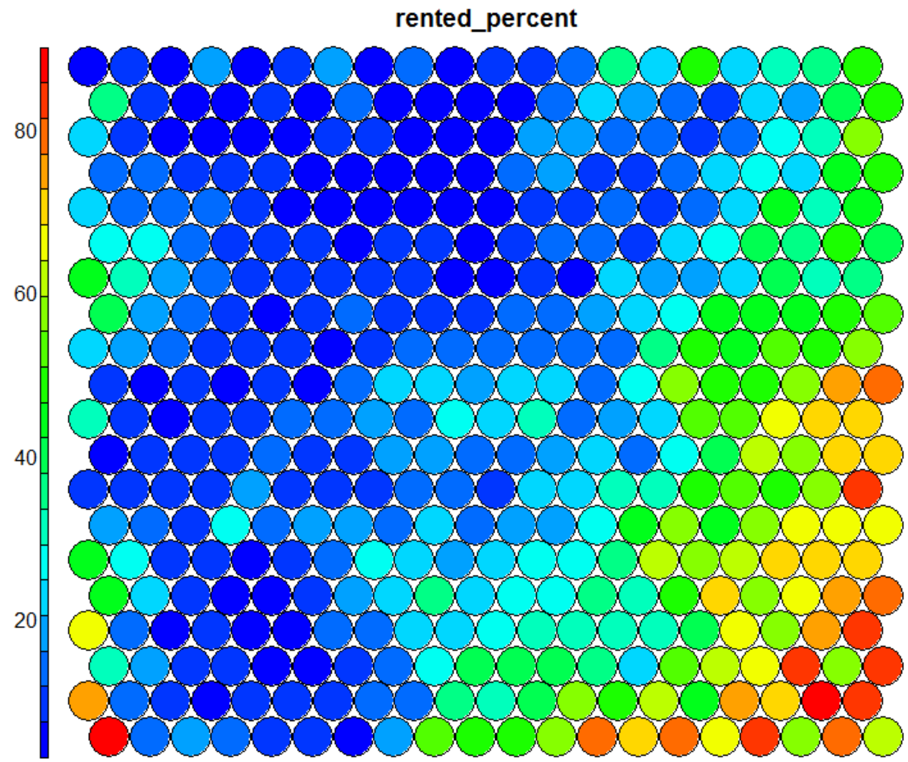


Census Data Example

- Very simple to manually identify clusters of nodes.
- By visualising heatmaps, can spot relationships between variables.
- View heatmaps of variables not used to drive the SOM generation.

```
plotHeatMap(som_model, data, variable=0) #interactive window for selection
```

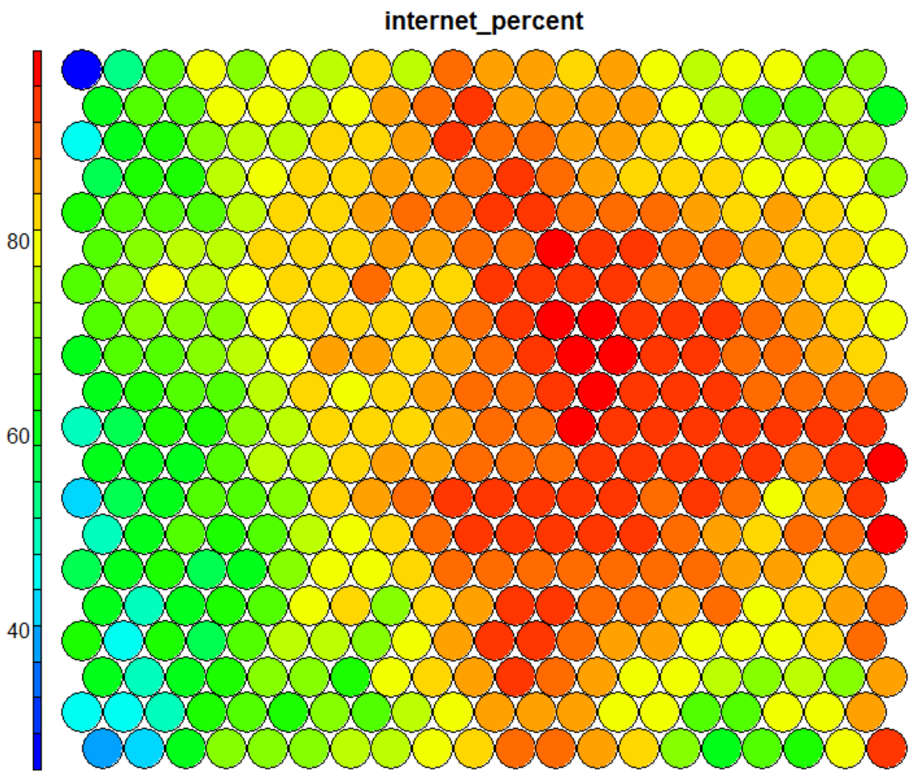


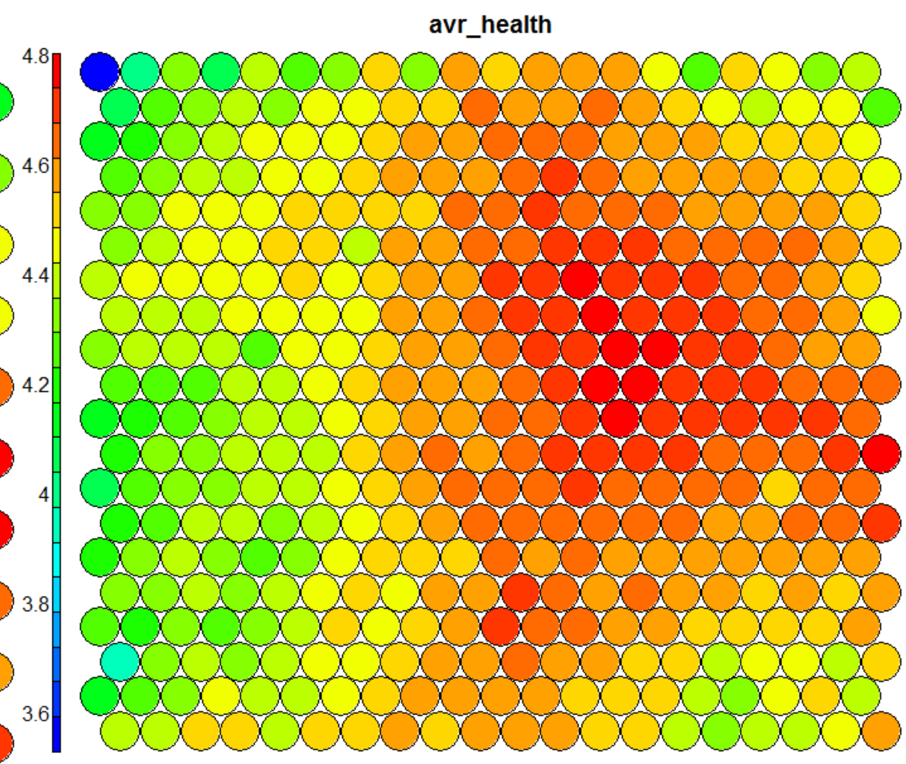


Census Data Example

- Very simple to manually identify clusters of nodes.
- By visualising heatmaps, can spot relationships between variables.
- View heatmaps of variables not used to drive the SOM generation.

```
plotHeatMap(som_model, data, variable=0) #interactive window for selection
```

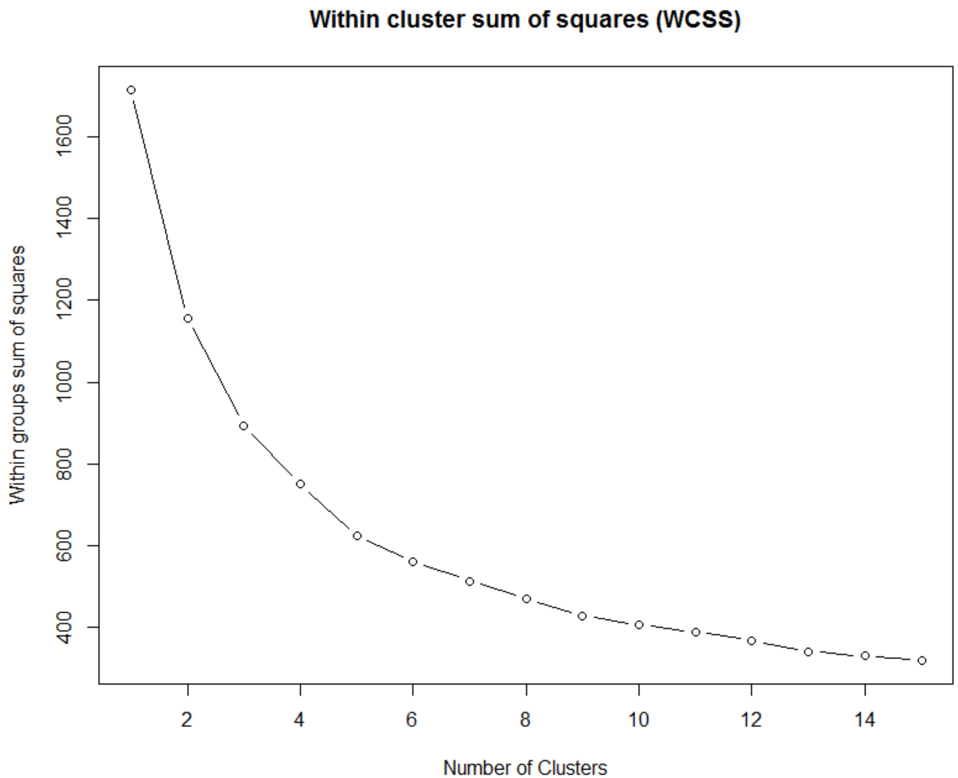




Census Data Example

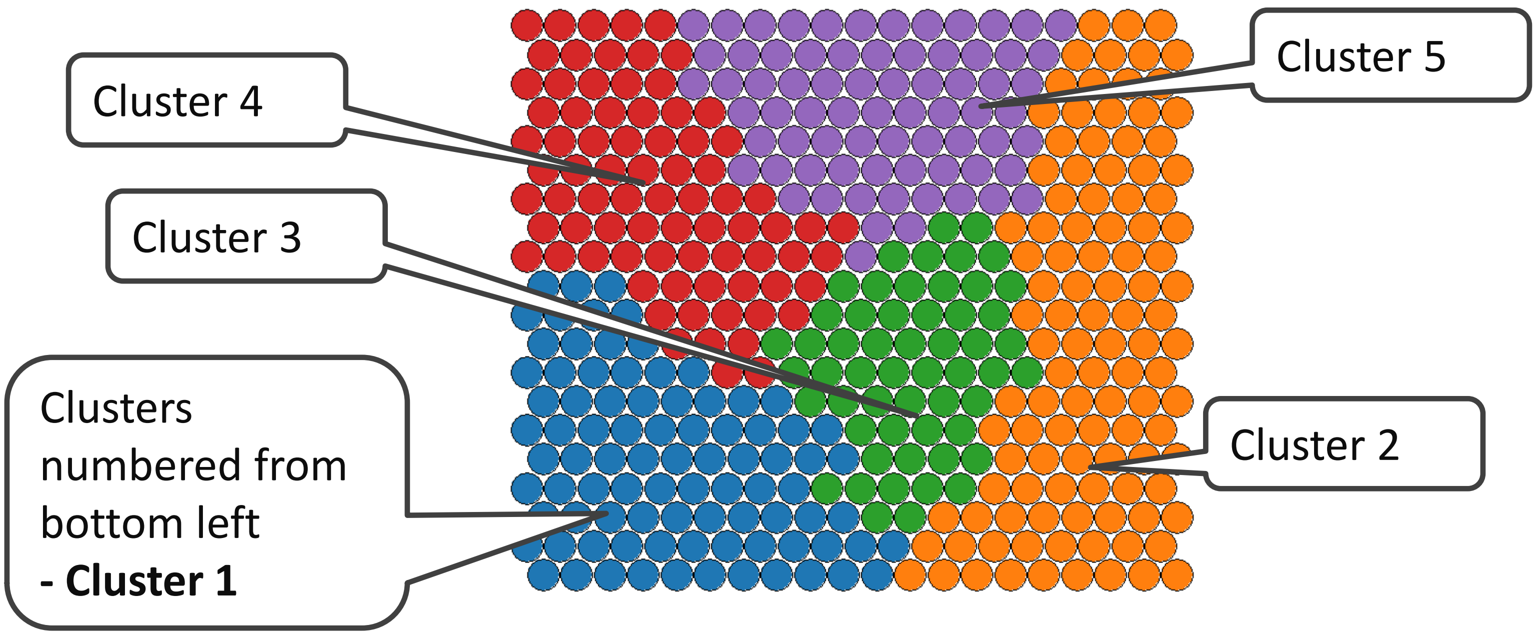
```
mydata <- som_model$codes
wss <- (nrow(mydata)-1)*sum(apply(mydata,2,var))
for (i in 2:15) wss[i] <- sum(kmeans(mydata,
                                     centers=i)$withinss)
```

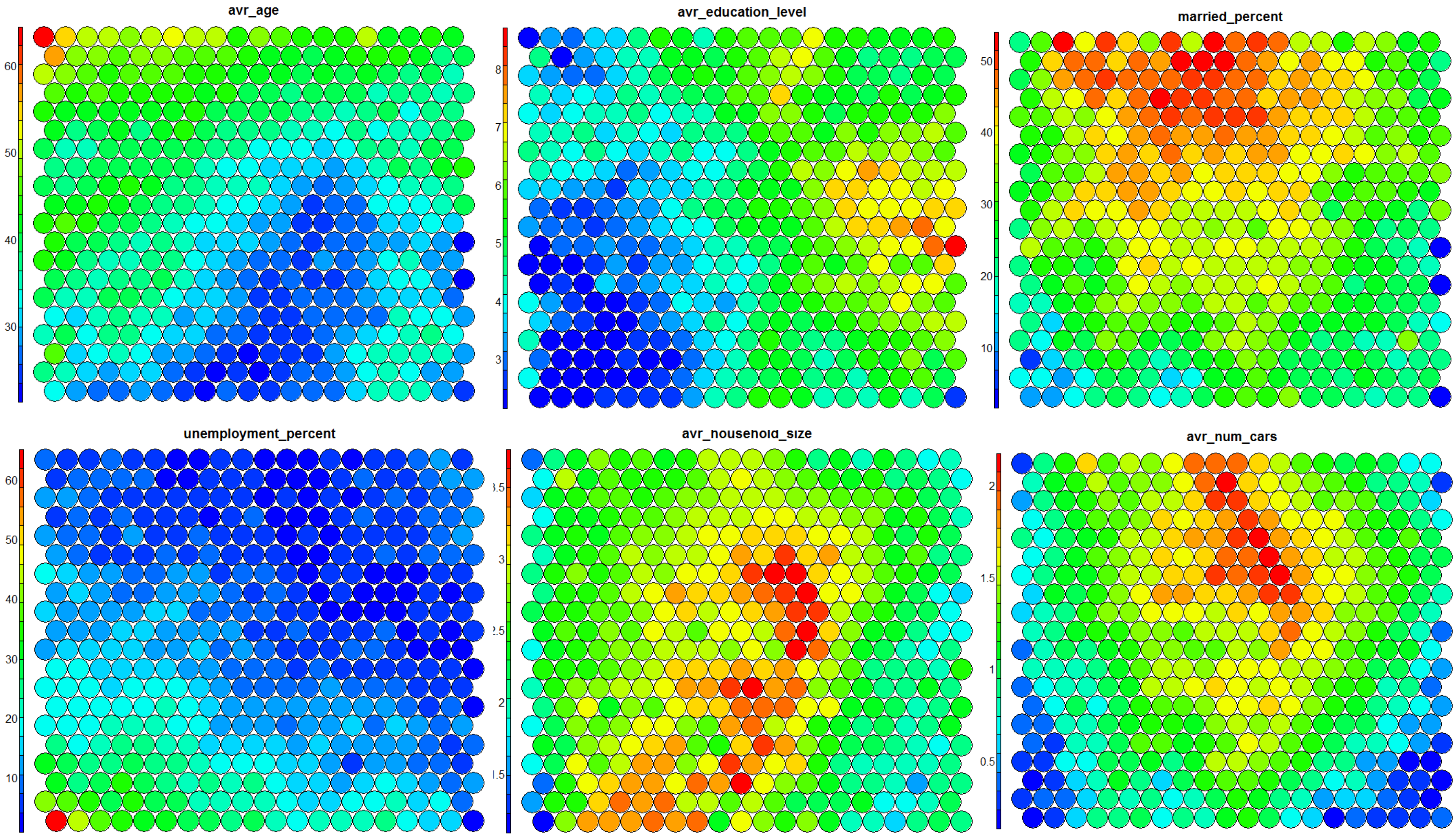
- Cluster on som codebooks
- Hierarchical clustering
  - Start with every node as a cluster
  - Combine most similar nodes (once they are neighbours)
  - Continue until all combined
- User decision as to how many clusters suit application



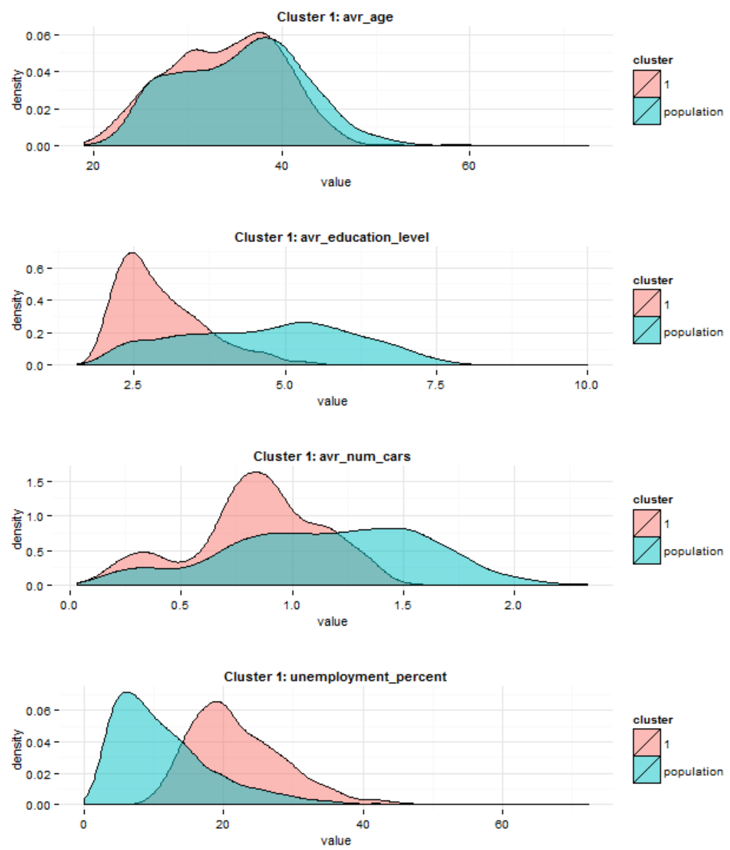
Census Data Example

```
som_cluster <- clusterSOM(ClusterCodebooks=som_model$codes,
                          LengthClustering=nrow(som_model$codes),
                          Mapping=som_model$unit.classif,
                          MapRow=som_model$grid$xdim,
                          MapColumn=som_model$grid$ydim,
                          StoppingCluster=2 )[[5]]
plot(som_model, type="mapping", bgcol = pretty_palette[som_cluster], main =
"Clusters")
add.cluster.boundaries(som_model, som_cluster)
```



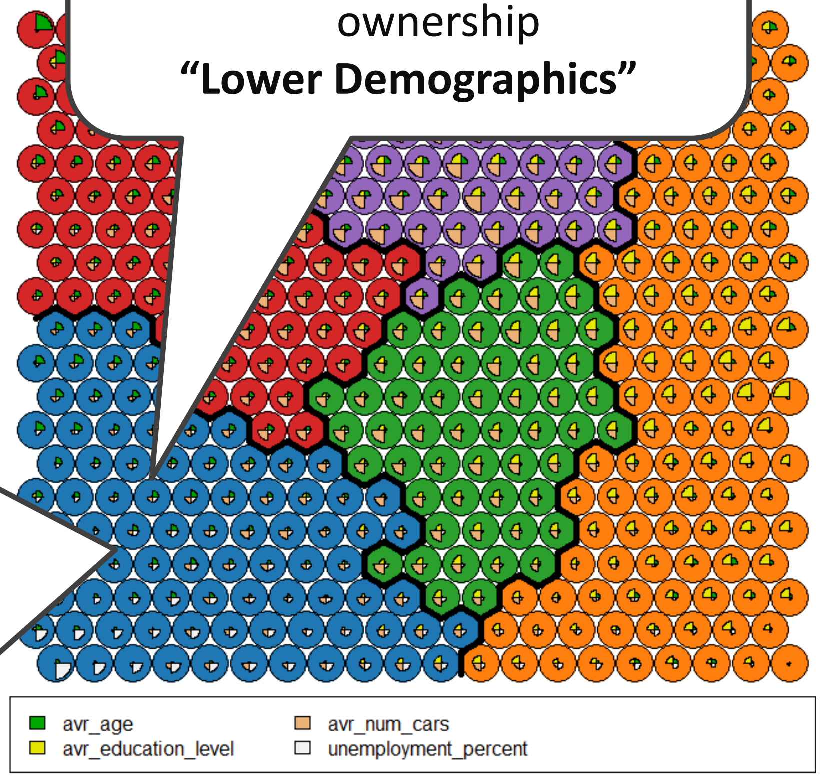

avr_age
60
50
40
30
avr_education_level
8
7
6
5
4
3
married_percent
50
40
30
20
10
unemployment_percent
60
50
40
30
20
10
avr_household_size
3
2
avr_num_cars
2
1.5
1
0.5

Census Data Example



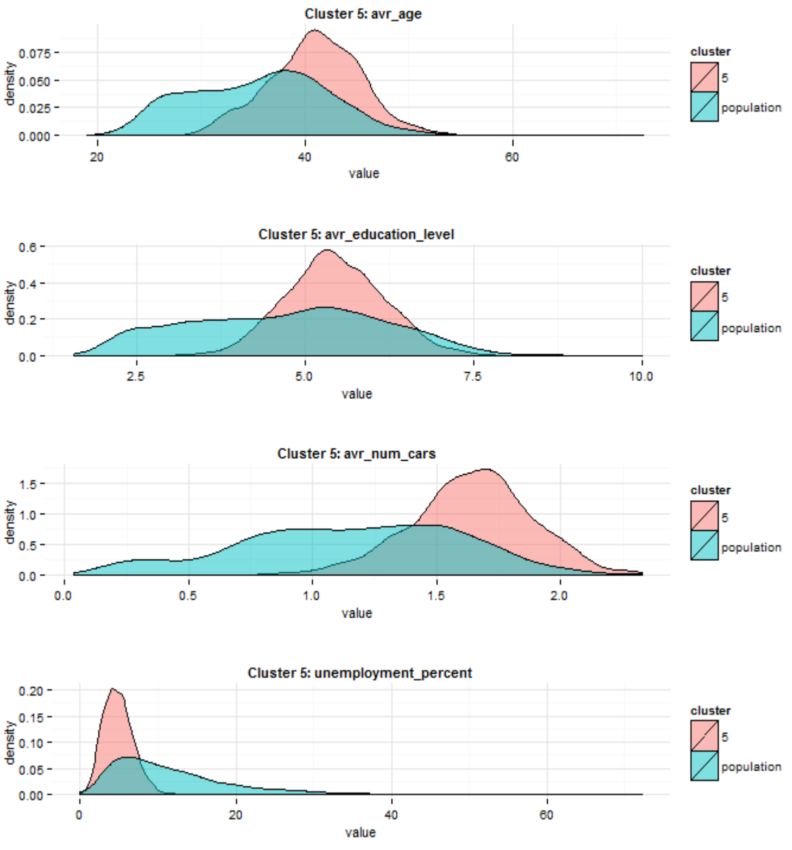
- High level of unemployment
- Low average education
- Lower than average car ownership

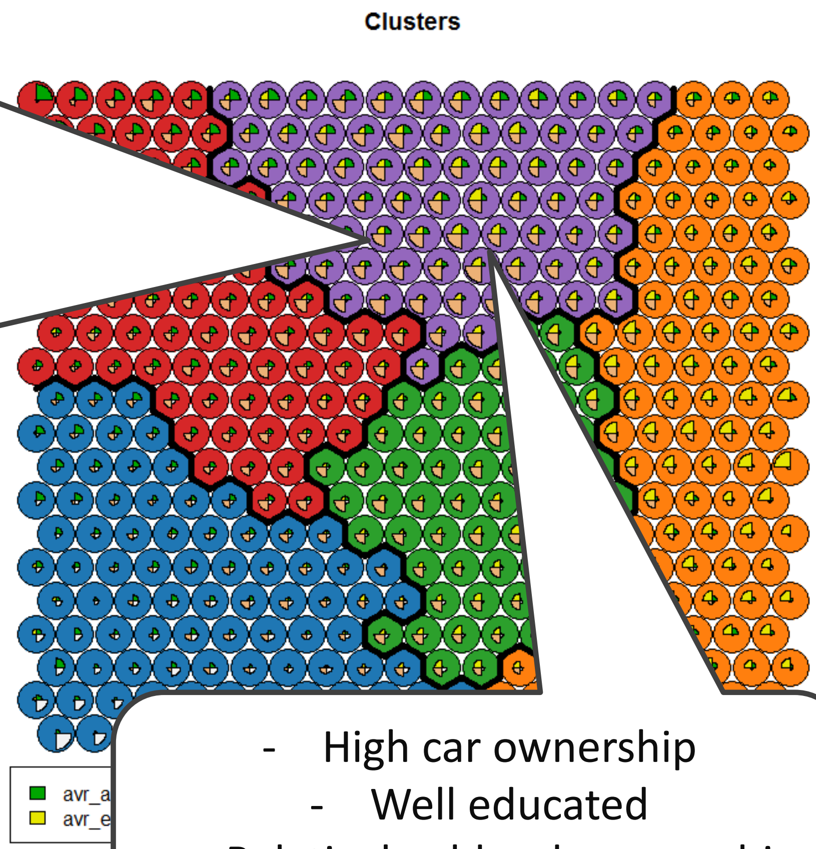
“Lower Demographics”



Census Data Example

Cluster 5 details

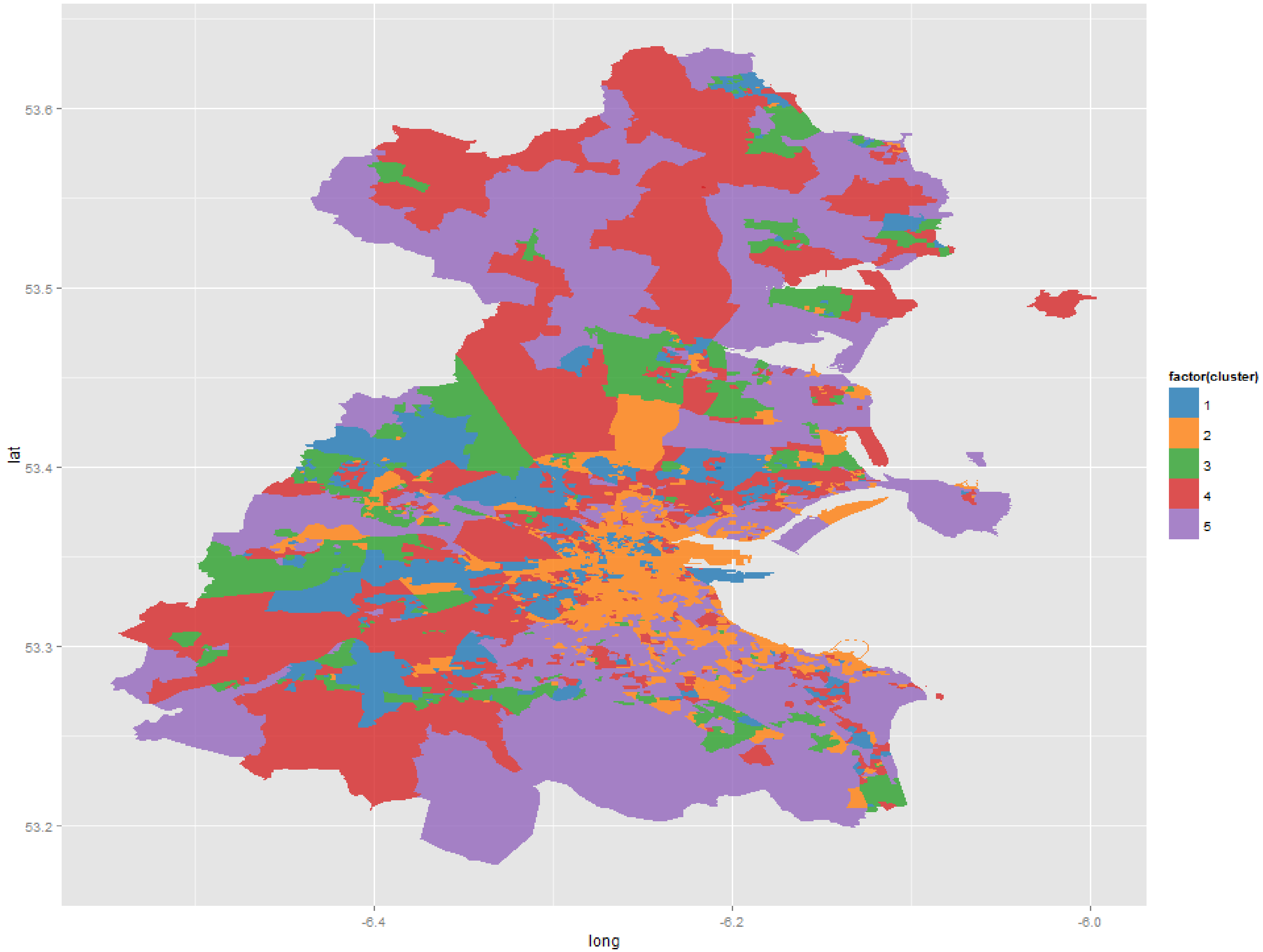




- High car ownership
- Well educated
- Relatively older demographic
- Larger households

"Commuter Belt"

Census Data Example



Census Data Example

Conclusions

- Can build meaningful names / stories for clusters in population
- Multiple iterations with different variable combinations and heatmaps
- Similar to work by commercial data providers “Experian”

Group	Description	Type	Description
A	Established Elites	A01	Elite Executives
		A02	Affluent Empty Nesters
		A03	Professional Urbanites
B	Upwardly Mobile Enclaves	B04	Aspiring Professional Couples
		B05	Evolving Diversity
		B06	Up and Coming
C	City Centre Mix	C07	City Centre Sophisticates
		C08	Inner Ring Cosmopolitans
		C09	Industrious New Comers
		C10	University Influence
D	Struggling Society	D11	Striving Large Families
		D12	Entrenched Hardship
E	Poorer Greys	E13	Ageing Workers
		E14	Community Stalwarts
		E15	Town Centre Singles
F	Industrious Urban Fringe	F16	Settled in Suburbia
		F17	Working Family Commuters
		F18	Small Town Simplicity
G	Careers & Kids	G19	Suburban Progress
		G20	Successful Families
		G21	Upscale Commuters

Source: Experian Mosaic Ireland: New Mosaic Classifications

Ta-Feng Grocery Shopping

- More complex and realistic example of customer data
- Data set contains information on
  - 817,741 grocery shopping transactions.
  - 32,266 customers

date	customer_id	age_group	address	product_subclass	product_id	quantity	asset	price
01/01/2001	141833	F	F	130207	4.71E+12	2	44	52
01/01/2001	1376753	E	E	110217	4.71E+12	1	150	129
01/01/2001	1603071	E	G	100201	4.71E+12	1	35	39
01/01/2001	1738667	E	F	530105	4.71E+12	1	94	119
01/01/2001	2141497	A	B	320407	4.71E+12	1	100	159
01/01/2001	1868685	J	E	110109	4.71E+12	1	144	190

- Similar steps as with census example, but with some additional requirements

Ta-Feng Grocery Shopping

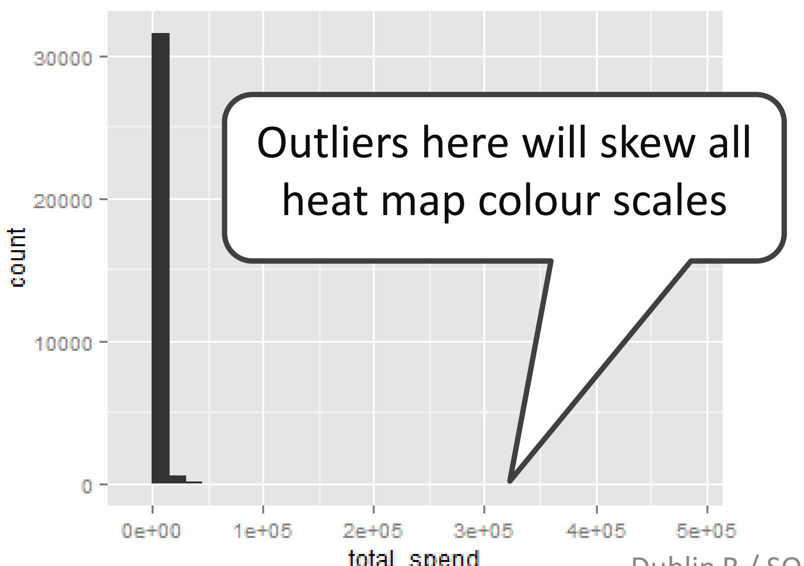
- Feature generation using data.table - Data table has significant speed enhancements over standard data frame, or ddply.

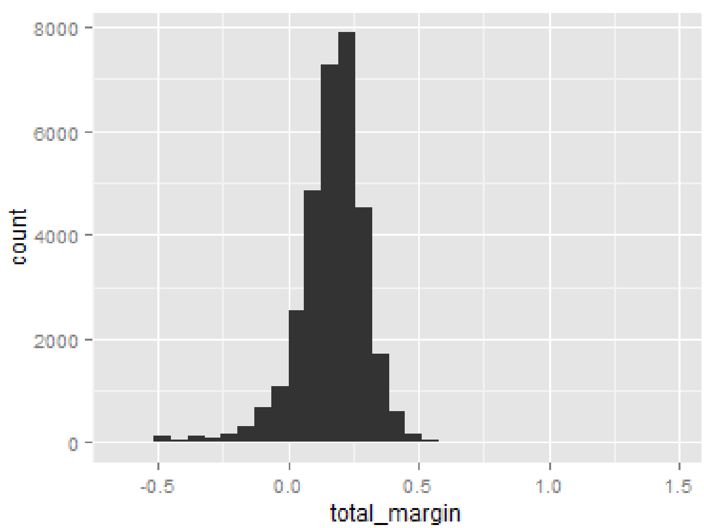
```
> system.time(test1 <- data[, sum(price), by=customer_id])
   user  system elapsed
   0.11    0.00    0.11
> system.time(test2 <- aggregate(data$price,
by=list(data$customer_id), FUN=sum))
   user  system elapsed
   1.81    0.03    1.84
> system.time(test3 <- ddply(data[,c("price", "customer_id"),
with=F],
+                             .variables="customer_id",
+                             .fun=function(x){sum(x$price)}))
   user  system elapsed
   5.86    0.00    5.86
```

Ta-Feng Grocery Shopping

- Customer characteristics generated:

Customer ID	Total Spend	Max Item	Num Baskets
Total Items	Items per basket	Mean item cost	Profit per basket
Total Profit	Sales Items (%)	Profit Margin	Budget items (%)
Premium items (%)	Item rankings	Weekend baskets (%)	Max basket cost



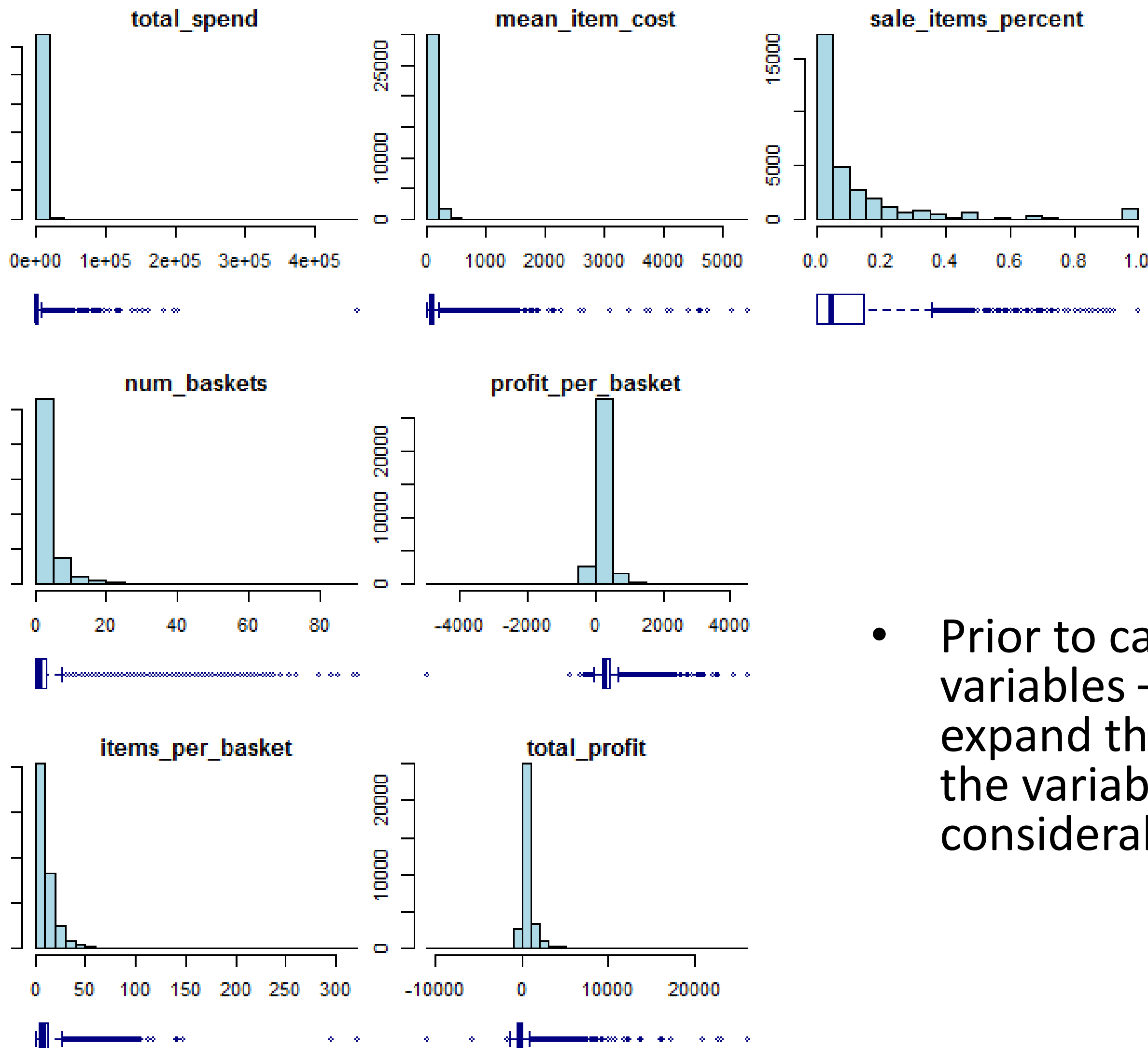


Ta-Feng Grocery Shopping

- Solution is to cap all variables at 98% percentiles
- This removes extreme outliers from dataset and prevents one node discolouring heatmaps

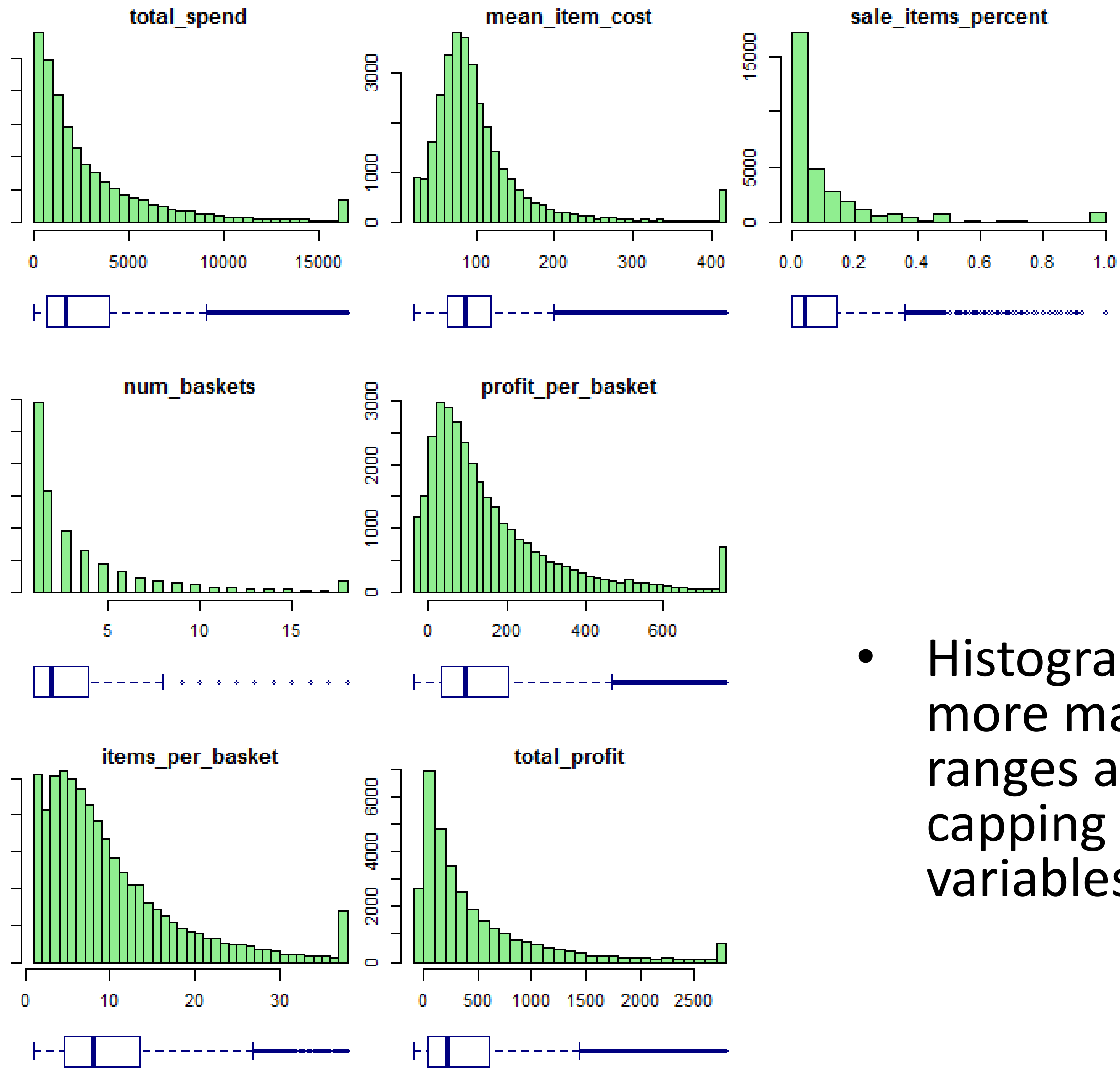
```
capVector <- function(x, probs = c(0.02,0.98)){
  #remove things above and below the percentiles specified
  ranges <- quantile(x, probs=probs, na.rm=T)
  x[x < ranges[1]] <- ranges[1]
  x[x > ranges[2]] <- ranges[2]
  return(x)
}
```

Ta-Feng Grocery Shopping



- Prior to capping the variables – outliers expand the range of the variables considerably

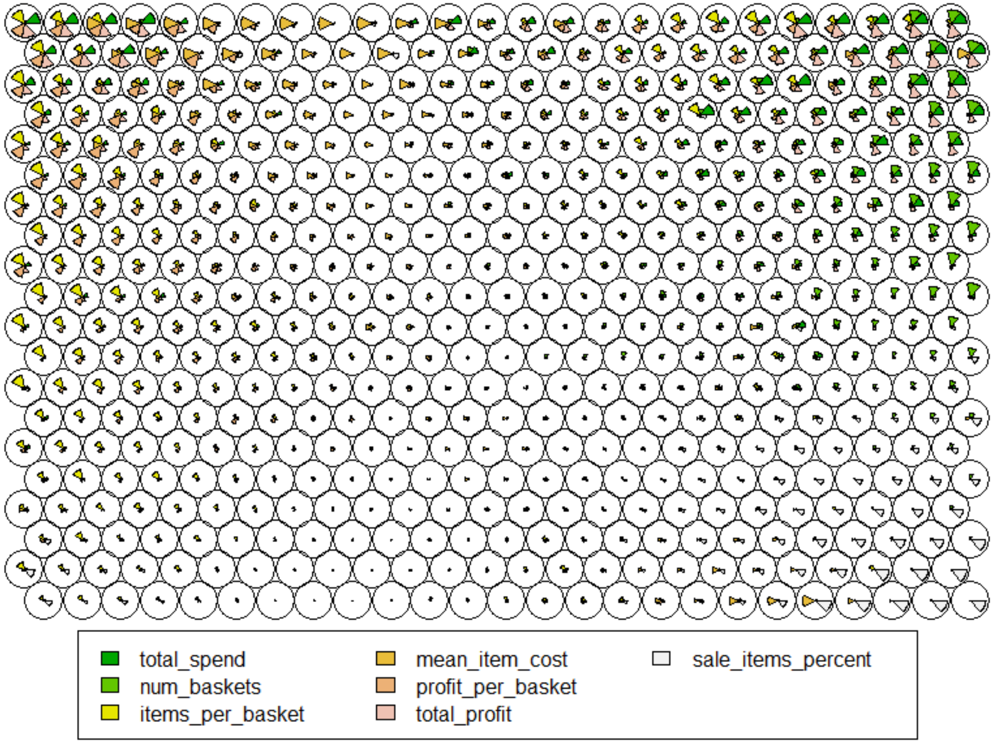
Ta-Feng Grocery Shopping

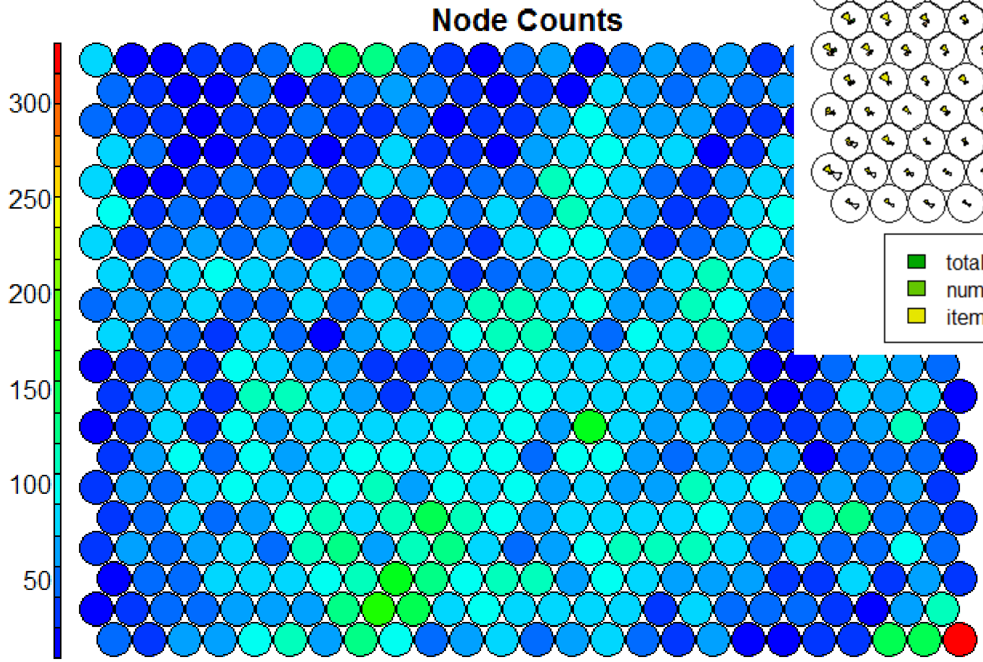


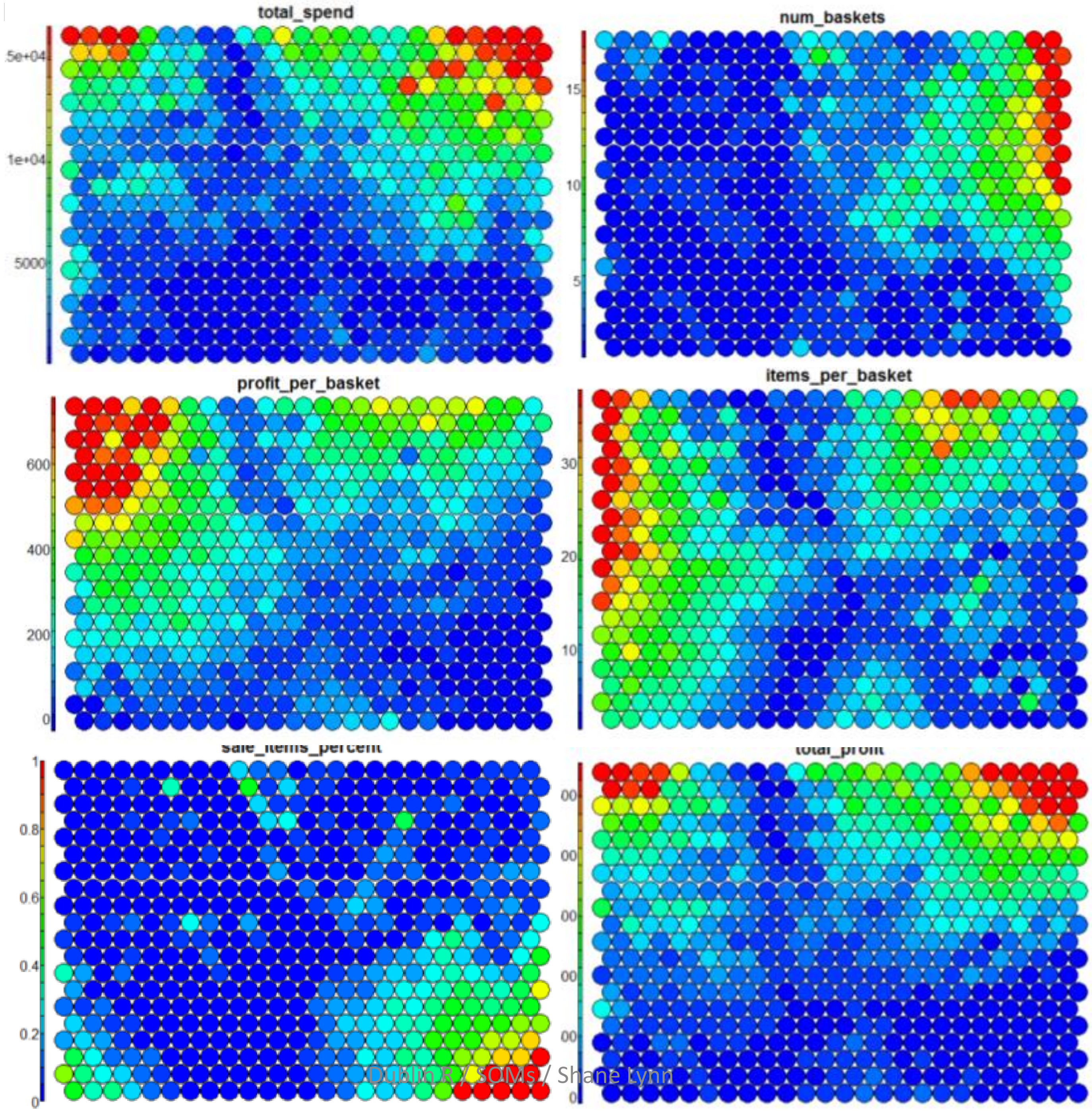
- Histograms have more manageable ranges after capping the variables.

Ta-Feng Grocery Shopping

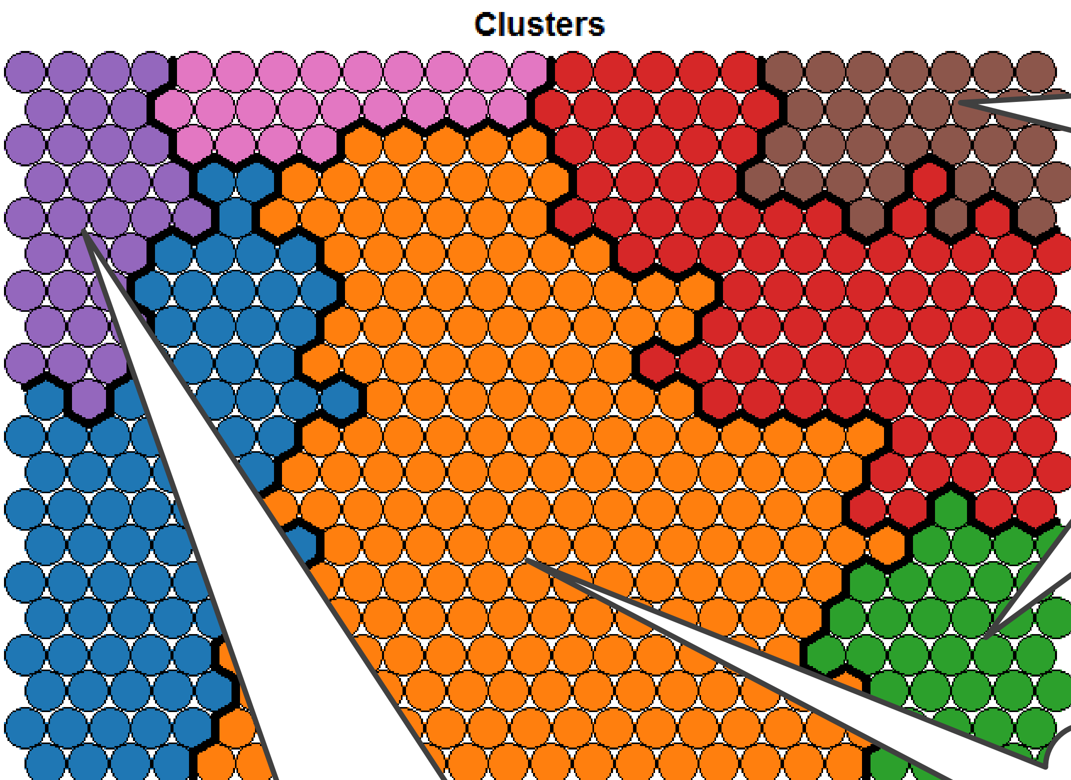
- Explore heatmaps and variable distributions to find patterns






total_spend
1e+04
5000
num_baskets
15
10
5
profit_per_basket
600
400
200
0
items_per_basket
30
20
10
1
0.8
0.6
0.4
0.2
0

Ta-Feng Grocery Shopping



"Steady shoppers"
- Many visits to store
- Profitable overall
- Smaller Baskets

"Bargain Hunters"
- Almost exclusively sale items
- Few visits
- Not profitable

"The Big Baskets"
- Few visits
- Largest baskets
- Large profit per basket

"The great unwashed"
- Single / small repeat visitors
- Small baskets
- Large number of customers

Conclusions

- SOMs are a powerful tool to have in your repertoire.
- Advantages include:
  - Intuitive method to develop customer segmentation profiles.
  - Relatively simple algorithm, easy to explain results to non-data scientists
  - New data points can be mapped to trained model for predictive purposes.
- Disadvantages:
  - Lack of parallelisation capabilities for VERY large data sets
  - Difficult to represent very many variables in two dimensional plane
  - Requires clean, numeric data.
- Slides and code will be posted online shortly.

Questions

Deloitte Analytics.

Shane Lynn

www.shanelynn.ie / @shane_a_lynn

Useful links

- Som package
  - http://cran.r-project.org/web/packages/kohonen/kohonen.pdf
- Census data and online viewer
  - http://census.cso.ie/sapmap/
  - http://www.cso.ie/en/census/census2011smallareapopulationstatisticssaps/
- Ta Feng data set download
  - http://recsyswiki.com/wiki/Grocery_shopping_datasets