Clustering Code - PennState

#R #cluster #StatisticalModeling

the following are clustering codes from an example problem developed by PennState

---

Cluster Analysis: An Example

The following is an example of applying cluster analysis in R by Penn State[1][2]

Overview

Cluster analysis is an exploratory, descriptive, “bottom-up” approach to structure heterogeneity. From a “data mining” perspective cluster analysis is an “unsupervised learning” approach.

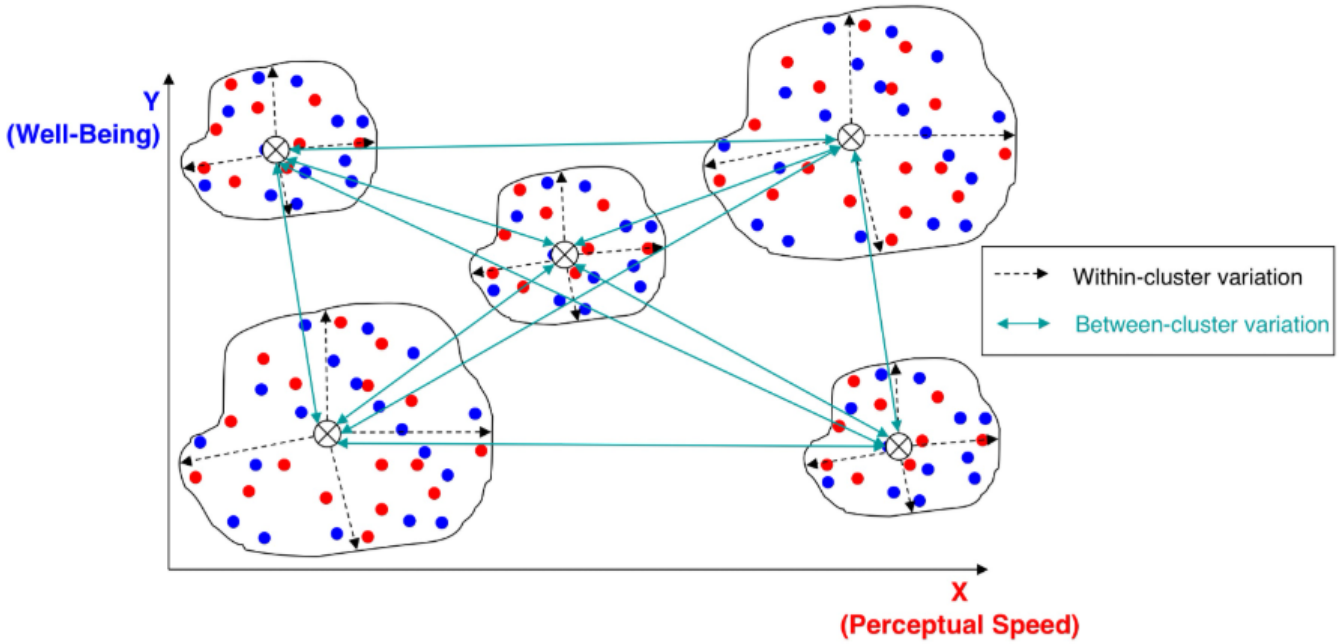
A key underpinning of cluster analysis is an assumption that a sample is NOT homogeneous. The method is used to examine and describe distinct sub-populations in the sample.

Can groups of individuals (observations) be identified whose members (a) are similar on group-defining variables, and (b) differ from members of other groups?

Often cluster analysis (and other “mixture” methods) are considered as a person-oriented approach - where a research objective is to identify “types of persons”. A contrast is made with variable-oriented approaches, such as factor analysis and regression - where the research objective is to identify groups of variables or relations among variables. An intuitive representation of the contrast is shown by whether one is interested in data reduction across columns (= variable-oriented) or rows (= person-oriented) of a persons x variables data matrix.

Note: Generally, the “person-specific” terminology used here at PSU is fundamentally different than the “person-oriented” or “person-centered” terminology used elsewhere. (We would rather label the approach used elsewhere as a “sub-group-oriented” approach.)

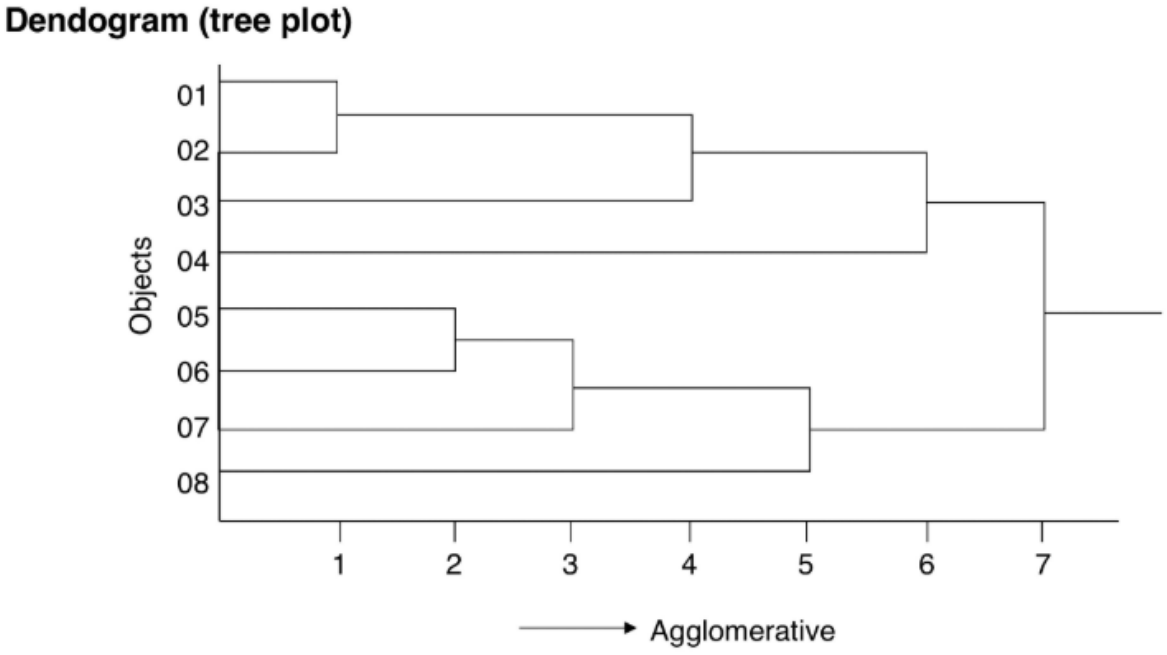
This is the general idea …



There are, of course, a variety of ways to do this … with techniques generally falling into two categories

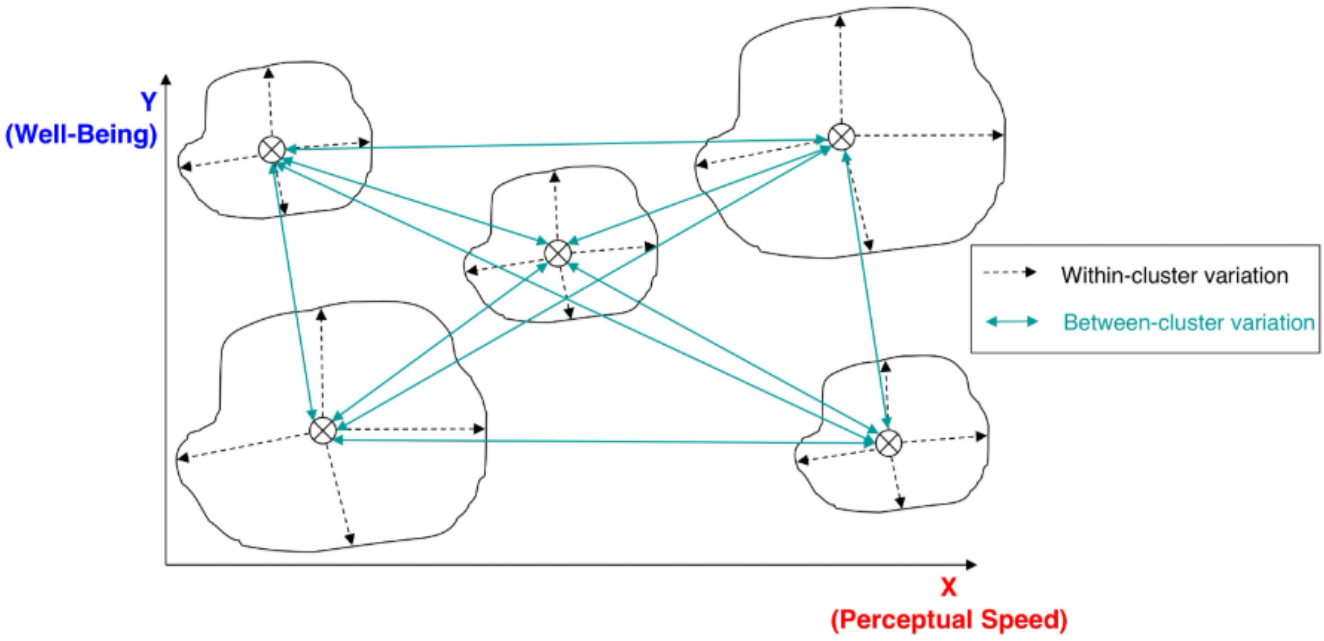
Hierarchical methods … where each object initially forms its own cluster and objects are grouped together based on similarity/difference.

Hierarchical Clustering Methods: Each object initially forms its own cluster



Non-hierarchical methods … where number of clusters are specified in advance (prior to launching algorithm) and objects are moved in and out of clusters ("ANOVA in reverse"). For example, the k-means clustering algorithm assigns objects (persons) to clusters so as to maximize the difference

among the means of the clusters on all variables.



Preliminaries

Libraries used in this script. There are two main libraries that we will use `cluster` (namesake) and `fpc` (Flexible Procedures for Clustering). Some other functions are in the base package.

```
#general packages
library(ggplot2)
library(psych)

#cluster packages
library(cluster) #clustering
library(fpc) #flexible procedures for clustering
library(clusterCrit) #cluster criteria
```

Our example makes use of one our experience sampling data sets, but treats these data as though they are cross-sectional.
Getting the data and doing a bit of data management (new id variable)

```
#set filepath for data file
filepath <-
"https://quantdev.ssri.psu.edu/sites/qdev/files/AMIBbrief_raw_daily1.csv"

#read in the .csv file using the url() function
daily <- read.csv(file=url(filepath),header=TRUE)

#clean-up of variable names
var.names.daily <- tolower(colnames(daily))
colnames(daily)<-var.names.daily
```

```
#creating a new "id" variable
daily$id <- daily$id*10+daily$day

names(daily)
```

```
##  [1] "id"      "day"     "date"    "slphrs"  "weath"   "lteq"    "pss"
##  [8] "se"      "swls"    "evalday" "posaff"  "negaff"  "temp"    "hum"
## [15] "wind"    "bar"     "prec"
```

```
#reducing down to variable set
daily <- daily[ ,c("id","slphrs","weath","lteq","pss","se","swls","evalday",
"posaff","negaff","temp","hum","wind","bar","prec")]

#names of variables
names(daily)
```

```
##  [1] "id"      "slphrs"  "weath"   "lteq"    "pss"     "se"      "swls"
##  [8] "evalday" "posaff"  "negaff"  "temp"    "hum"     "wind"    "bar"
## [15] "prec"
```

```
#looking at data
head(daily,10)
```

id	slphrs	weath	lteq	pss	se	swls	evalday	posaff	negaff	temp	hum	wind	bar	prec
1010	6.0	1	10	2.50	2	3.8	1	3.9	3.0	28.0	0.79	11.0	29.40	0.20
1011	2.0	2	10	2.75	3	4.2	0	3.8	2.3	20.8	0.62	3.6	30.17	0.00
1012	9.0	3	10	3.50	4	5.0	1	5.1	1.0	29.1	0.51	1.9	30.35	0.02
1013	7.5	2	9	3.00	4	5.0	1	5.6	1.3	30.2	0.58	2.7	30.23	0.00
1014	8.0	1	18	2.75	3	4.0	1	4.3	1.1	22.7	0.55	2.4	30.46	0.00
1015	8.0	2	19	2.75	3	4.2	1	3.9	1.0	21.4	0.54	0.7	30.54	0.00
1016	8.0	3	21	3.50	4	4.6	1	5.1	1.2	31.4	0.49	1.0	30.51	0.00
1017	7.0	NA	14	2.75	3	4.6	1	4.8	1.1	45.3	0.52	1.1	30.30	0.00
1020	7.0	0	12	3.50	5	5.6	0	6.3	1.4	28.0	0.79	11.0	29.40	0.20
1021	6.0	0	20	4.00	5	6.6	0	7.0	1.6	20.8	0.62	3.6	30.17	0.00

Cluster Analysis: General

The aim of cluster analysis is to identify groups of similar observations - formally, forming groups so that:
(a) within a group, the observations are most similar to each other,
(b) and between groups the observations are most dissimilar to each other.

Cluster analysis is a form of unsupervised classification: no pre-defined classes, and can be considered descriptive data mining.

Humans, as a society, have been “clustering” for a long time in attempts to understand (and simplify) the environment we live in:

Clustering the animal and plant kingdoms into a hierarchy of similarities.
Clustering chemical structures.
Day-by-day we see grocery items clustered into similar groups.
We cluster student populations into similar groups of students from similar backgrounds or studying similar combinations of subjects.

Similarity / Dissimilarity
The concept of similarity is often operationlized via measurement of distance. The task in cluster analysis is to identify groups of observations, so that the distance between the observations within a group is minimised and between the groups the distance is maximised. The fundamental information that is used in clustering is distance. In the same way that other methods work from similarity measures (e.g., correlation matrix), cluster analysis methods work from dissimilarity measures (e.g., distance matrix).
There are some caveats to performing automated cluster analysis using distance measures. We often observe, particularly with large datasets, that a number of interesting clusters will be generated, and that one or two clusters will account for the majority of the observations. It is as if these larger clusters simply lump together those observations that don’t fit elsewhere. As with all exploratory methods, there is a lot of “art” involved. That is, researchers must be prepared to make decisions themselves - they must be actively involved in the exploration and description process.

Preparing Data

Missing Data

Note that cluster analysis does NOT generally work with missing data. Here we simply delete incomplete cases. Other possibilities include imputation, and calculation of distances using most complete subsets.

```
#removing observations with NA
dailysub <- daily[complete.cases(daily), ]
describe(dailysub)
```

	vars	n	mean	sd	median	trimmed	mad	min	max	range	skew	kurtosis	se
id	1	1376	3276.2819767	1279.8763851	3271.50	3302.2422868	1497.426000	1010.00	5327.00	4317.00	-0.1023726	-1.0402205	34.5031550
slphrs	2	1376	7.1960756	1.8058116	7.00	7.1974138	1.482600	0.00	18.00	18.00	0.1207219	1.9324209	0.0486814
weath	3	1376	2.0007267	1.2941826	2.00	2.0009074	1.482600	0.00	4.00	4.00	-0.0556619	-1.0634078	0.0348888
lteq	4	1376	12.5007267	10.4214464	9.00	11.2359347	8.895600	0.00	58.00	58.00	1.0724521	0.9464013	0.2809434
pss	5	1376	2.6184593	0.6839263	2.75	2.6414096	0.741300	0.00	4.00	4.00	-0.3692700	0.1713646	0.0184374
se	6	1376	3.4287791	0.9927081	3.00	3.4646098	1.482600	1.00	5.00	4.00	-0.4022571	-0.1178189	0.0267616
swls	7	1376	4.1120640	1.2725150	4.20	4.1546279	1.186080	1.00	7.00	6.00	-0.2776101	-0.2196009	0.0343047
evalday	8	1376	0.6845930	0.4648467	1.00	0.7304900	0.000000	0.00	1.00	1.00	-0.7936331	-1.3711413	0.0125314
posaff	9	1376	4.1078670	1.1003100	4.20	4.1387250	1.186080	1.00	7.00	6.00	-0.2430963	-0.3671490	0.0296624
negaff	10	1376	2.4509448	1.0386149	2.20	2.3407668	1.037820	1.00	6.90	5.90	0.9490537	0.6733386	0.0279992
temp	11	1376	40.1818314	7.8759041	42.00	40.5127042	8.895600	20.80	56.00	35.20	-0.3685936	-0.2358267	0.2123201
hum	12	1376	0.6204578	0.1963037	0.66	0.6295735	0.207564	0.24	0.90	0.66	-0.3643819	-1.1017409	0.0052920
wind	13	1376	7.3567587	4.4525749	7.00	6.8075318	4.447800	0.70	20.00	19.30	0.9736384	0.8603566	0.1200334
bar	14	1376	30.0182122	0.3340609	30.00	30.0373049	0.429954	29.32	30.54	1.22	-0.3889538	-0.8964509	0.0090057
prec	15	1376	0.0493387	0.0934249	0.00	0.0253085	0.000000	0.00	0.30	0.30	1.8539266	1.9753796	0.0025186

Scaling

The unit of distance may be different for different variables. For example, one year of difference in age seems like it should be a larger difference than $1 difference in income. Different variables will be "weighted" differently in the distance calculation. To alleviate this, a common approach is to rescale each variable into a standardized, *z-score* variable (i.e., by subtracting the mean and dividing by the standard deviation). Thus, all the variables would then have mean = 0, with differences scales in standard deviation units. Note that this scales everything in relation to the observed sample (which has plusses and minuses).
The R function `scale()` makes it all very easy.

```
#scaling all the variables
dailyscale <- data.frame(scale(dailysub, center=TRUE, scale=TRUE))
#checking and fixing the id variable (which we did not want standardized)
str(dailyscale$id)
```

```
##  num [1:1376] -1.77 -1.77 -1.77 -1.77 -1.77 ...
```

```
dailyscale$id <- dailysub$id
str(dailyscale$id)
```

```
##  num [1:1376] 1010 1011 1012 1013 1014 ...
```

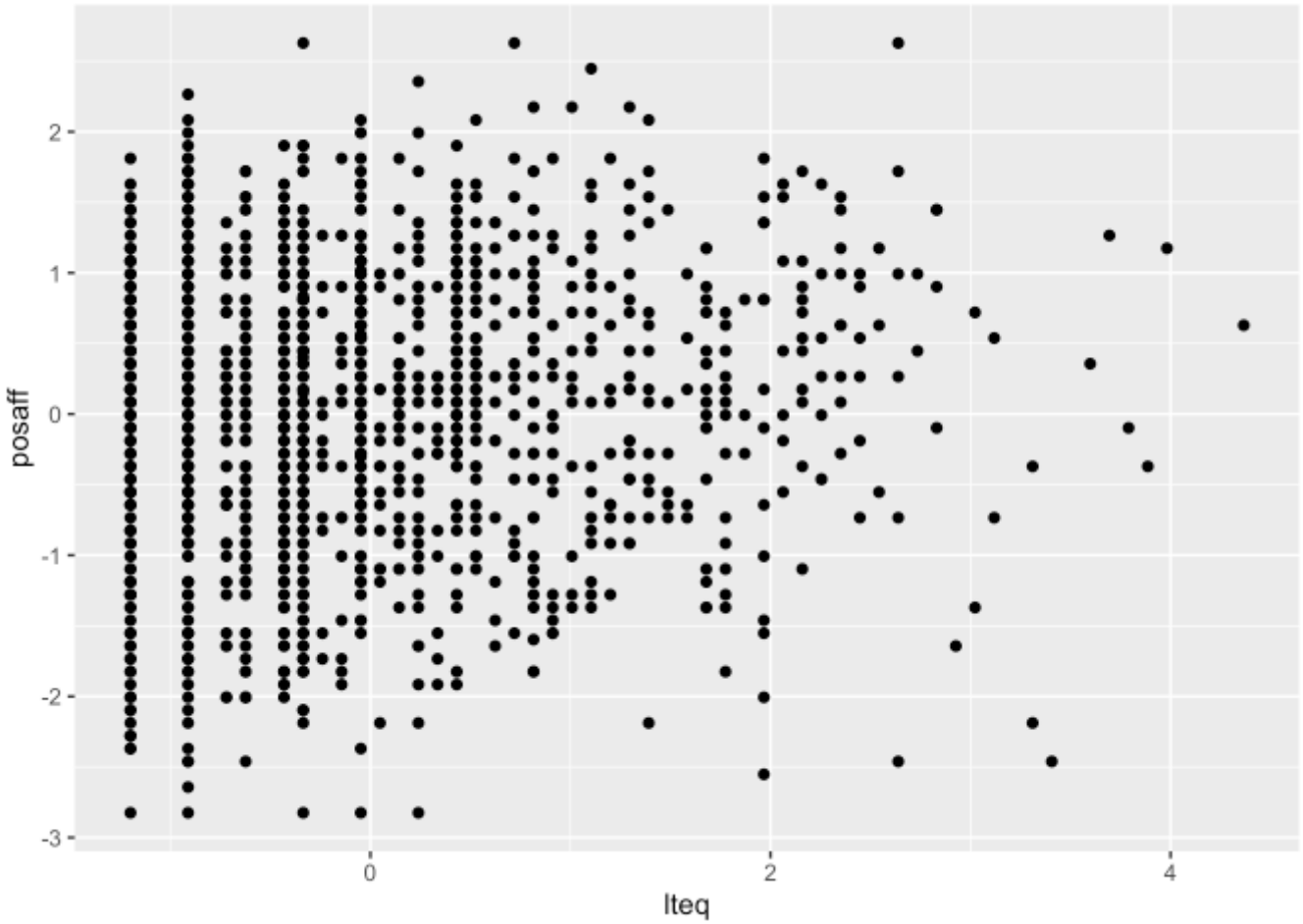
```
describe(dailyscale)
```

	vars	n	mean	sd	median	trimmed	mad	min	max	range	skew	kurtosis	se
id	1	1376	3276.282	1279.876	3271.5000000	3302.2422868	1497.4260000	1010.0000000	5327.0000000	4317.000000	-0.1023726	-1.0402205	34.5031550
slphrs	2	1376	0.000	1.000	-0.1085803	0.0007411	0.8210159	-3.9849537	5.9828636	9.967817	0.1207219	1.9324209	0.0269582
weath	3	1376	0.000	1.000	-0.0005615	0.0001396	1.1455880	-1.5459385	1.5448155	3.090754	-0.0556619	-1.0634078	0.0269582
lteq	4	1376	0.000	1.000	-0.3359156	-0.1213644	0.8535859	-1.1995194	4.3659269	5.565446	1.0724521	0.9464013	0.0269582
pss	5	1376	0.000	1.000	0.1923317	0.0335566	1.0838887	-3.8285695	2.0200140	5.848584	-0.3692700	0.1713646	0.0269582
se	6	1376	0.000	1.000	-0.4319287	0.0360939	1.4934904	-2.4466196	1.5827623	4.029382	-0.4022571	-0.1178189	0.0269582
swls	7	1376	0.000	1.000	0.0691041	0.0334487	0.9320755	-2.4456011	2.2694712	4.715072	-0.2776101	-0.2196009	0.0269582
evalday	8	1376	0.000	1.000	0.6785183	0.0987358	0.0000000	-1.4727286	0.6785183	2.151247	-0.7936331	-1.3711413	0.0269582
posaff	9	1376	0.000	1.000	0.0837337	0.0280449	1.0779507	-2.8245376	2.6284710	5.453009	-0.2430963	-0.3671490	0.0269582
negaff	10	1376	0.000	1.000	-0.2416148	-0.1060816	0.9992346	-1.3969998	4.2836428	5.680643	0.9490537	0.6733386	0.0269582
temp	11	1376	0.000	1.000	0.2308520	0.0420108	1.1294703	-2.4609024	2.0084257	4.469328	-0.3685936	-0.2358267	0.0269582
hum	12	1376	0.000	1.000	0.2014336	0.0464365	1.0573617	-1.9381086	1.4240291	3.362138	-0.3643819	-1.1017409	0.0269582
wind	13	1376	0.000	1.000	-0.0801241	-0.1233504	0.9989276	-1.4950358	2.8395348	4.334570	0.9736384	0.8603566	0.0269582
bar	14	1376	0.000	1.000	-0.0545176	0.0571533	1.2870529	-2.0900749	1.5619543	3.652029	-0.3889538	-0.8964509	0.0269582
prec	15	1376	0.000	1.000	-0.5281103	-0.2572133	0.0000000	-0.5281103	2.6830242	3.211135	1.8539266	1.9753796	0.0269582

Plotting a sample of bivariate observations

We choose a small subset of variables for easy visualization in a bivariate space. We use lteq, a measure of physical activity, and posaff, a measure of positive affect.

```
ggplot(dailyscale,aes(x=lteq,y=posaff)) +
  geom_point()
```



Note that these data are not specifically constructed for cluster analysis - so it is questionable whether this is the approach that would actually be used here.

Distances

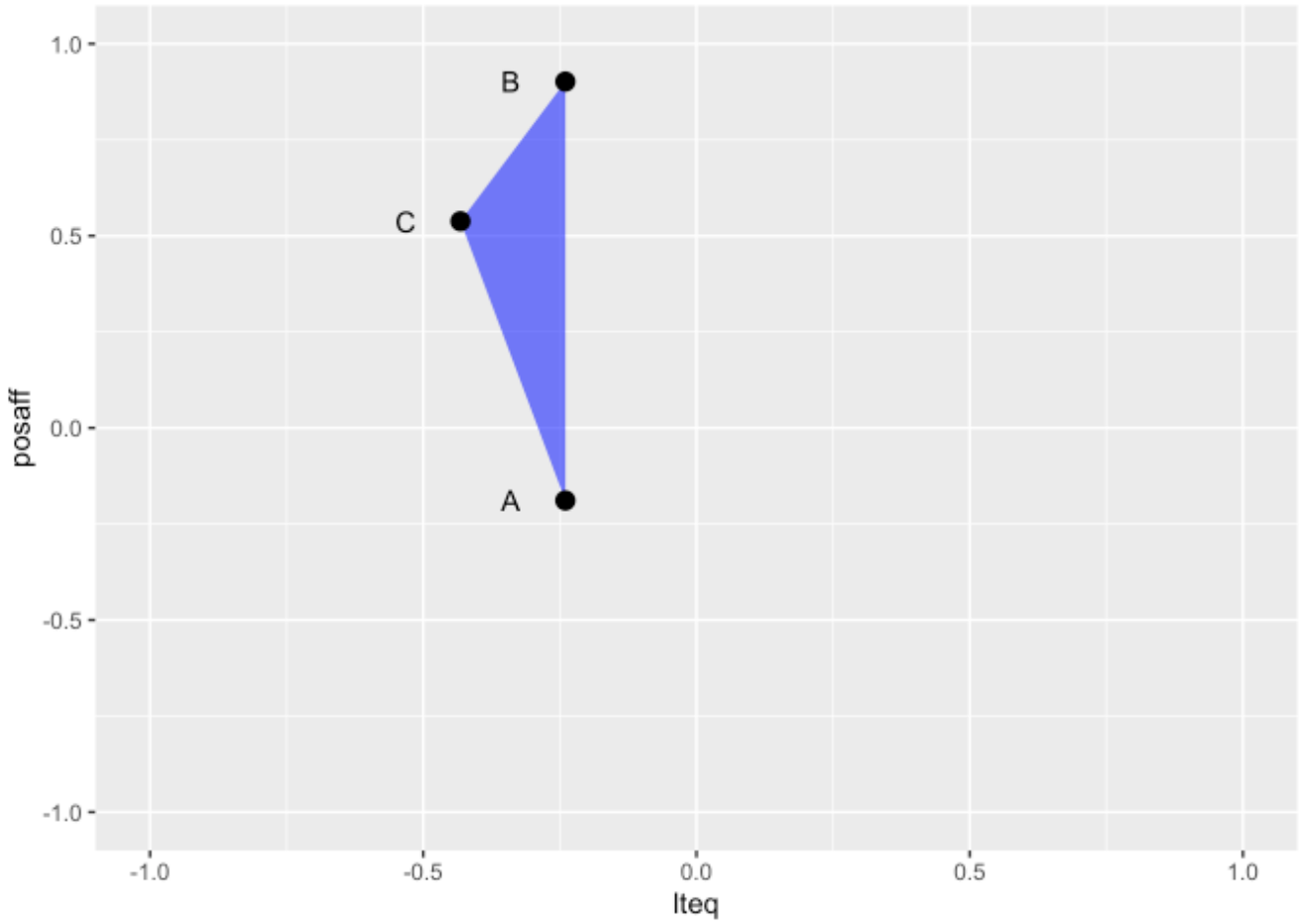
Each individual is concptualized as a point in a multivariate space.
For example, lets look at the first three individuals.

```
data1 <- dailyscale[c(1,3,12),c("id","lteq","posaff")]
head(data1,3)
```

	id	lteq	posaff
1	1010	-0.2399597	-0.1889168
3	1012	-0.2399597	0.9016850
14	1025	-0.4318716	0.5381510

```
labels.abc <-c("A","B","C")
ggplot(data1,aes(x=lteq,y=posaff)) +
  geom_polygon(fill="blue",alpha=.6) +
  geom_point(size=3) +
```

```
  geom_text(aes(x=lteq-.1,label=labels.abc)) +
  ylim(-1,1) + xlim(-1,1)
```



'

Lets look at the distances. Euclidean Distance is calculated as

$$EuclidianDistance_{A,B} = \sqrt{(x_a - x_b)^2 + (y_a - y_b)^2}$$

and easily implemented using the `dist()` function.

```
dist.abc <- dist(data1[1:3,2:3],method="euclidean",diag=TRUE,upper=FALSE)
dist.abc
```

```
##            1         3        14
## 1  0.0000000
## 3  1.0906017 0.0000000
## 14 0.7519693 0.4110804 0.0000000
```

Might also use a different distance measure, such as Manhattan Distance … The distance between two points in a grid based on a strictly horizontal and/or vertical path (that is, along the grid lines), as opposed to the diagonal or “as the crow flies” distance. The Manhattan distance is the simple sum of the horizontal and vertical components, whereas the diagonal distance might be computed by

applying the Pythagorean theorem.

$$ManhattanDistance_{A,B} = |x_a - x_b| + |y_a - y_b|$$

```
dist.abc2 <- dist(data1[1:3,2:3],method="manhattan",diag=TRUE,upper=FALSE)
dist.abc2
```

```
##            1         3        14
## 1  0.0000000
## 3  1.0906017 0.0000000
## 14 0.9189797 0.5554458 0.0000000
```

The great thing about the distances is that they scale up to distance in many dimensions! Yay for geometry!

K-Means Cluster Analysis

Basic clustering in the social sciences often makes use of the *K-means* procedure. The k-means algorithm is a traditional and widely used clustering algorithm. In brief, the algorithm begins by specifying the number of clusters we are interested in. This is the *k*. Each of the *k* clusters is identified by the vector of the average (i.e., the mean) value of each of the variables for observations within a cluster. A random clustering is constructed (random set of mean vectors). The *k* means are calculated. Then, using the distance measure, we gravitate each observation to its nearest mean. The means are then recalculated and the points re-gravitate. And so on until there is no further change to the means.

For illustration - Lets look at a result. We use the R function `kmeans()`.

```
#there are random starts involved so we set a seed
set.seed(1234)
#running a cluster analysis
model <- kmeans(dailyscale[,c("lteq","posaff")], centers=4)
model
```

```
## K-means clustering with 4 clusters of sizes 377, 578, 240, 181
##
## Cluster means:
##         lteq     posaff
## 1 -0.6558537 -1.0705106
## 2 -0.4408363  0.6996343
## 3  0.7763740 -0.5678819
## 4  1.7443674  0.7485388
##
## Clustering vector:
##    1    2    3    4    5    6    7    9   10   11   13   14   15   17   18   19
##    2    1    2    2    3    3    4    2    4    2    2    2    1    2    2    2
```

```
##   20   21   22   23   27   28   29   30   31   32   37   38   39   40   41   42
##    1    1    1    4    1    1    1    1    2    2    2    2    2    2    1    2
##   43   44   47   48   49   51   52   53   54   55   56   57   58   59   60   61
##    2    1    2    2    2    2    2    2    2    1    3    3    1    1    1    3
##   62   63   64   66   67   68   70   71   72   73   74   75   76   77   78   79
##    3    2    4    3    1    3    2    2    2    2    2    2    2    1    2    3
##   81   84   85   86   87   88   89   90   91   92   93   95   96   97   98   99
##    2    1    3    1    2    2    2    4    4    4    4    2    2    2    2    4
##  101  102  103  105  106  107  108  110  111  112  113  114  115  116  117  118
##    4    3    2    3    4    4    3    2    4    4    4    4    4    2    4    4
##  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134
##    4    3    1    1    1    1    1    1    1    2    2    2    2    2    2    2
##  135  136  137  138  140  141  143  144  145  146  147  149  150  151  152  153
##    2    4    4    2    3    3    3    4    3    4    4    4    4    4    1    3
##  154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169
##    2    3    2    3    3    2    2    2    2    2    2    2    2    2    2    2
##  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185
##    1    2    3    1    1    1    1    1    1    1    1    2    2    2    3    3
##  186  187  188  189  190  191  192  193  194  196  197  198  199  201  202  203
##    2    4    4    4    3    3    4    1    1    2    2    2    2    2    4    4
##  204  205  206  208  209  210  211  212  213  214  215  216  217  218  219  220
##    2    2    4    4    4    2    2    2    2    2    2    1    2    3    3    1
##  221  222  223  224  225  226  228  229  230  231  232  233  234  235  236  237
##    1    3    3    3    3    2    2    2    1    1    2    3    3    2    3    1
##  238  239  240  241  242  243  244  245  246  247  248  249  250  251  252  253
##    2    2    1    1    3    3    2    3    3    2    3    2    3    3    3    3
##  254  255  256  257  258  259  260  261  262  263  264  265  266  267  268  269
##    3    4    3    3    2    2    2    2    1    2    2    2    2    1    2    1
##  270  271  272  273  274  275  276  277  278  279  280  281  282  283  285  286
##    3    3    2    2    2    2    1    1    1    2    1    1    1    3    3    1
##  287  288  289  290  291  292  293  294  295  296  297  298  299  300  301  302
##    1    2    2    1    1    1    1    1    1    2    2    2    3    1    2    1
##  303  304  305  306  307  308  309  312  313  314  315  316  317  318  319  320
##    3    2    2    2    2    2    1    2    2    2    2    3    3    1    1    2
##  321  323  324  325  326  327  328  329  330  331  332  333  334  335  343  344
##    2    2    2    2    2    2    2    4    2    2    2    2    2    2    3    2
##  345  346  347  348  349  350  352  353  354  355  356  358  359  360  361  364
##    1    2    1    2    2    2    2    2    2    1    1    1    1    2    4    4
##  365  366  367  368  369  370  371  372  373  374  376  377  378  379  380  381
##    4    4    2    2    2    2    3    2    2    3    1    1    1    2    2    1
##  382  383  384  385  386  387  388  389  390  391  392  393  394  395  396  397
##    1    2    3    2    3    2    2    3    3    3    3    4    1    1    4    3
##  398  399  400  401  402  403  404  405  406  407  408  409  410  411  412  413
##    4    4    4    1    1    4    4    3    4    1    1    1    2    1    1    1
##  414  415  416  417  418  419  422  423  424  425  426  427  428  429  430  431
##    1    2    2    2    2    2    1    2    3    3    4    3    3    4    4    2
```

```
##  432  433  434  435  436  437  438  439  441  442  443  444  445  446  447  448
##    2    2    2    2    3    1    1    3    2    2    3    3    3    3    2    2
##  449  450  451  452  453  454  455  456  457  458  459  460  461  462  463  464
##    2    2    2    2    2    2    1    1    1    1    3    3    1    3    2    2
##  465  466  467  468  469  470  471  472  473  474  475  476  477  478  479  480
##    1    1    2    2    2    1    2    2    3    3    2    1    1    1    1    1
##  481  483  485  486  487  488  489  490  491  492  493  494  495  496  497  498
##    2    1    2    2    2    1    1    1    1    1    1    2    1    2    1    1
##  499  500  501  502  503  504  505  506  507  508  509  510  511  512  513  514
##    1    1    2    2    3    3    3    3    3    2    2    2    2    2    2    2
##  515  516  517  518  519  520  521  522  523  524  525  526  527  528  529  530
##    1    1    4    3    2    2    4    2    4    4    1    1    1    2    3    3
##  531  533  534  535  536  537  538  539  540  541  542  543  544  545  546  548
##    3    3    4    4    4    1    1    1    1    1    1    1    4    4    2    4
##  549  550  551  552  553  554  555  556  557  558  559  560  561  562  563  564
##    4    2    2    2    4    3    3    4    1    3    4    2    3    1    1    2
##  565  566  567  568  569  570  571  572  573  574  575  576  577  578  579  580
##    3    1    1    2    2    2    2    2    2    2    2    1    3    4    4    4
##  581  582  583  584  585  586  587  588  589  590  591  592  593  594  595  596
##    3    1    1    1    2    2    1    1    2    1    1    2    1    1    1    2
##  597  598  599  601  602  603  604  605  606  607  609  610  611  612  613  614
##    1    1    1    2    1    1    2    1    1    1    2    2    2    2    1    1
##  615  616  617  618  619  620  621  622  623  624  625  626  628  629  630  631
##    2    2    2    3    2    2    3    1    1    2    2    2    2    2    2    2
##  632  633  634  635  636  637  638  639  640  641  642  643  644  645  646  647
##    1    1    1    3    1    2    1    1    2    3    2    3    1    1    1    1
##  648  649  650  651  652  653  654  655  656  657  658  659  660  661  662  663
##    1    2    3    1    1    1    3    4    3    1    1    1    1    1    1    3
##  664  665  666  667  668  669  670  671  672  673  674  675  676  677  678  679
##    3    4    4    4    4    4    4    3    4    4    4    2    2    1    4    2
##  680  681  682  684  685  686  687  688  689  690  691  692  693  694  696  697
##    2    2    2    1    3    2    2    4    4    2    2    2    4    4    2    1
##  698  699  700  701  702  703  704  705  706  707  708  709  710  711  712  713
##    1    2    1    1    1    3    3    3    3    3    3    1    3    1    1    1
##  714  715  716  717  719  720  721  722  723  724  725  726  727  728  731  735
##    2    1    1    1    1    1    3    1    1    1    1    3    4    4    4    2
##  736  737  738  739  740  741  742  743  744  745  746  747  748  749  750  751
##    4    2    4    4    3    3    3    4    3    3    2    4    3    3    2    2
##  752  753  754  755  757  758  759  760  761  762  763  764  765  766  767  768
##    3    3    2    2    1    2    2    1    1    2    2    1    2    2    2    2
##  769  770  771  772  773  774  775  776  777  778  779  780  781  782  783  784
##    2    1    2    2    3    1    3    3    3    1    1    1    3    1    1    1
##  785  786  787  788  789  790  791  792  793  794  795  796  797  798  799  800
##    1    1    1    1    2    2    1    1    4    4    4    1    3    2    4    4
##  801  802  803  804  805  806  807  808  809  810  811  812  813  814  815  816
##    4    4    4    1    3    1    2    2    2    2    2    2    2    1    1    1
```

```
##  817  819  820  821  822  823  824  825  826  827  828  829  830  831  832  833
##    2    1    1    1    1    2    2    2    1    2    1    4    1    1    2    2
##  834  835  836  837  838  839  840  841  842  843  844  845  846  847  848  849
##    1    4    1    2    2    2    1    2    2    2    3    3    3    1    3    2
##  850  851  852  853  854  855  856  857  858  859  860  861  862  863  864  866
##    2    1    2    1    3    2    2    4    4    2    4    2    4    4    3    4
##  867  868  869  871  872  873  874  875  876  877  878  879  880  881  882  883
##    3    4    4    3    4    3    4    1    1    4    1    1    2    1    2    1
##  884  885  886  887  888  889  890  891  892  893  894  895  896  897  898  899
##    1    2    2    1    1    1    2    1    2    2    3    3    2    2    1    2
##  900  901  902  903  904  905  906  907  908  909  910  911  912  913  914  915
##    2    2    2    2    2    1    3    1    2    2    3    3    3    1    3    3
##  916  917  918  919  920  921  922  923  924  925  926  927  928  929  930  931
##    3    3    2    3    2    3    2    1    2    1    2    2    1    2    2    2
##  932  933  934  935  936  937  938  940  942  943  944  945  946  947  948  949
##    1    2    4    4    4    4    4    2    2    2    2    2    1    3    2    2
##  950  951  952  953  956  957  958  959  960  961  962  963  964  965  966  967
##    3    3    2    1    1    1    2    2    2    3    3    3    3    2    3    3
##  968  969  970  971  972  973  974  976  977  978  979  980  982  983  984  985
##    2    1    1    3    4    3    1    2    2    3    3    3    2    2    2    2
##  986  987  988  989  990  991  992  993  994  995  996  997  998  999 1000 1001
##    1    3    3    1    3    3    2    2    4    3    3    3    1    2    3    1
## 1002 1003 1004 1005 1006 1007 1008 1009 1011 1012 1013 1014 1015 1016 1017 1018
##    3    4    3    4    1    2    2    2    1    1    2    3    3    3    4    4
## 1019 1020 1021 1022 1023 1024 1025 1026 1027 1028 1029 1030 1031 1032 1033 1034
##    2    2    2    1    3    3    3    4    4    1    2    1    3    1    1    1
## 1035 1036 1037 1038 1039 1040 1041 1042 1043 1044 1045 1046 1047 1048 1049 1050
##    1    1    1    2    2    2    2    2    2    2    2    4    2    2    2    2
## 1051 1052 1053 1054 1055 1056 1057 1058 1059 1060 1061 1062 1064 1065 1066 1067
##    4    2    4    2    2    2    2    1    1    1    1    3    3    1    3    2
## 1068 1069 1070 1071 1072 1073 1074 1075 1076 1077 1078 1079 1080 1081 1082 1083
##    2    3    2    4    2    3    3    3    4    2    2    2    2    1    2    1
## 1084 1085 1086 1087 1088 1089 1090 1091 1092 1093 1094 1095 1096 1097 1098 1099
##    3    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2
## 1100 1101 1102 1103 1104 1105 1106 1107 1108 1109 1110 1111 1112 1113 1114 1115
##    2    2    1    2    1    1    2    1    2    2    4    2    4    2    2    2
## 1116 1117 1118 1119 1120 1121 1122 1123 1124 1125 1126 1127 1128 1129 1130 1131
##    2    2    2    3    3    3    1    3    2    2    2    2    2    2    4    4
## 1132 1133 1134 1135 1136 1137 1138 1139 1140 1141 1142 1143 1144 1145 1146 1147
##    4    2    2    4    2    4    4    3    4    4    2    2    2    3    2    2
## 1148 1149 1150 1151 1152 1153 1154 1156 1157 1158 1160 1161 1162 1163 1164 1165
##    2    2    2    2    1    2    2    1    2    1    2    2    2    1    1    1
## 1166 1167 1168 1169 1170 1171 1172 1173 1174 1175 1176 1177 1178 1179 1180 1181
##    2    4    4    1    3    1    4    2    4    3    3    2    3    3    4    4
## 1182 1183 1184 1185 1186 1187 1188 1189 1190 1191 1192 1193 1194 1195 1196 1197
##    1    2    2    1    2    2    3    1    3    3    4    2    2    4    2    4
```

```
## 1198 1199 1200 1201 1202 1203 1204 1205 1207 1208 1209 1210 1211 1212 1213 1215
##    2    2    2    2    2    2    1    1    2    1    2    2    2    2    2    2
## 1216 1217 1218 1219 1220 1221 1222 1223 1224 1225 1226 1227 1228 1229 1230 1231
##    2    1    2    2    2    2    1    2    1    2    2    1    2    1    4    4
## 1232 1233 1234 1235 1236 1237 1238 1239 1240 1241 1242 1243 1244 1245 1246 1247
##    4    3    4    4    3    3    2    2    1    2    2    2    1    1    2    2
## 1248 1249 1250 1251 1252 1253 1254 1255 1256 1257 1258 1259 1260 1261 1262 1263
##    2    3    1    2    4    2    4    4    2    4    4    4    3    4    2    2
## 1264 1265 1266 1267 1268 1269 1270 1271 1272 1273 1274 1275 1276 1277 1278 1279
##    2    2    3    2    1    3    2    3    4    4    1    2    2    4    2    2
## 1280 1281 1282 1283 1284 1285 1286 1287 1288 1289 1290 1291 1292 1293 1294 1295
##    2    2    1    1    1    1    2    1    1    1    2    2    2    2    2    2
## 1296 1297 1298 1299 1300 1301 1302 1304 1305 1306 1307 1308 1309 1310 1311 1312
##    3    1    3    2    2    2    2    1    1    1    1    2    2    2    2    2
## 1313 1314 1315 1316 1317 1318 1319 1320 1321 1322 1323 1324 1325 1326 1327 1328
##    2    2    2    2    1    2    2    2    2    1    2    2    2    1    2    1
## 1329 1330 1331 1332 1333 1334 1335 1336 1337 1338 1339 1340 1341 1342 1343 1344
##    1    1    1    4    1    3    2    3    3    3    3    2    2    2    2    2
## 1345 1346 1347 1348 1349 1350 1351 1352 1353 1354 1355 1356 1357 1358 1359 1360
##    1    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2
## 1361 1362 1363 1364 1365 1366 1367 1368 1369 1370 1371 1372 1373 1374 1375 1376
##    2    2    2    2    2    2    2    2    1    2    2    4    4    3    1    4
## 1377 1378 1379 1380 1381 1382 1383 1384 1385 1386 1387 1388 1389 1390 1391 1392
##    3    3    3    4    1    2    2    4    4    4    4    1    1    1    1    1
## 1393 1394 1395 1396 1397 1398 1400 1401 1402 1403 1404 1405 1406 1407 1408 1409
##    1    1    1    3    3    3    2    3    3    4    1    1    2    2    1    2
## 1410 1411 1412 1413 1414 1415 1416 1417 1418 1419 1420 1421 1422 1423 1424 1425
##    3    2    1    2    2    1    1    1    1    2    1    2    1    1    1    1
## 1426 1427 1429 1430 1431 1432 1433 1434 1435 1436 1437 1438 1439 1440 1441 1442
##    1    2    4    4    4    3    1    3    4    3    1    1    1    1    1    1
## 1443 1444 1445 1446 1447 1448 1449 1450 1451 1452 1453 1454 1455 1456 1457 1458
##    2    1    1    1    1    1    1    1    3    1    2    2    1    3    2    2
##
## Within cluster sum of squares by cluster:
## [1] 201.2980 329.1644 175.7229 180.1350
##  (between_SS / total_SS =  67.8 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

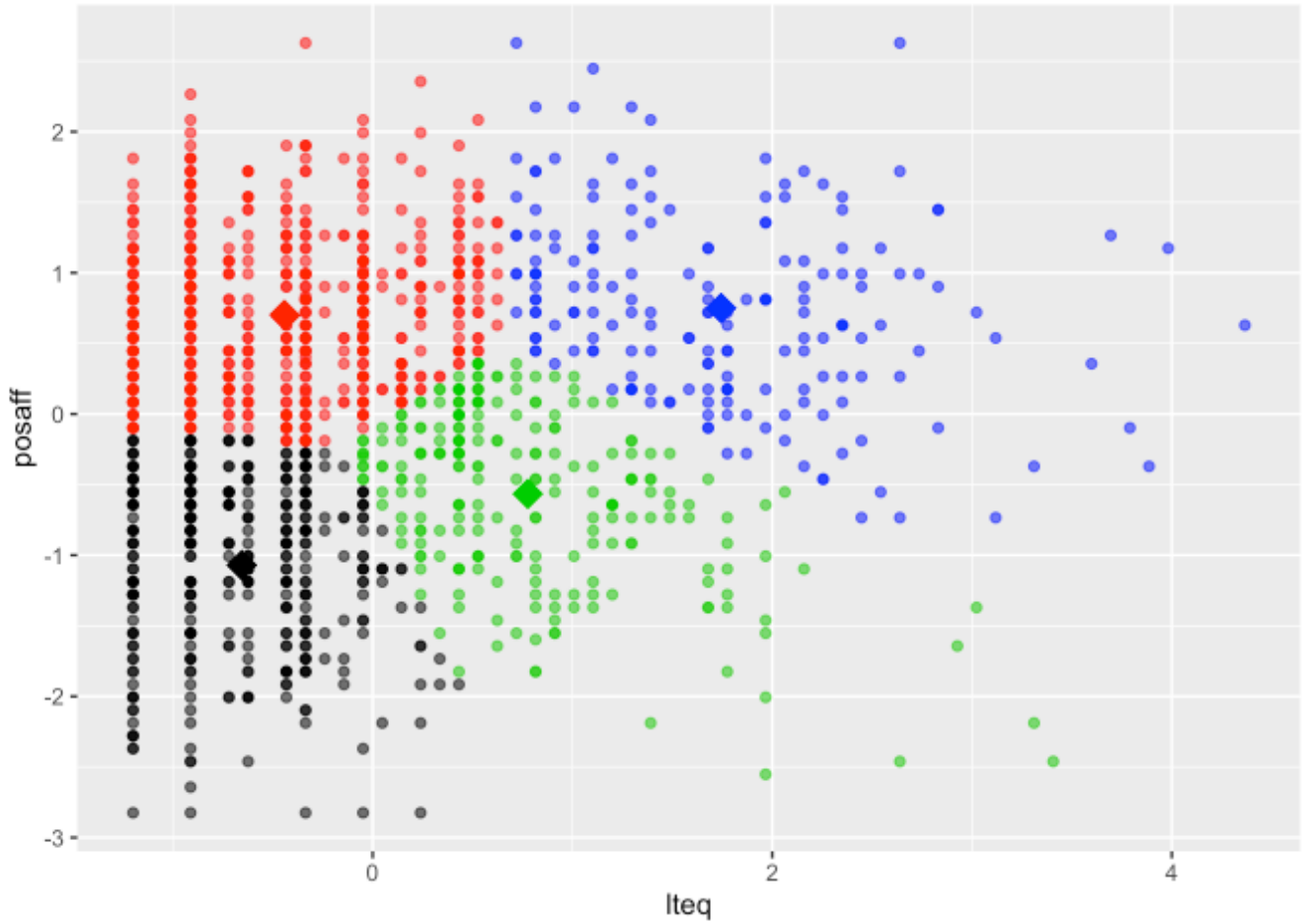
That is a lot of output! - but pretty easy to walk through and understand.

Lets extract the mean vectors and plot for a more intuitive understanding of the results.

```
#getting centers
model$centers
```

```
##         lteq     posaff
## 1 -0.6558537 -1.0705106
## 2 -0.4408363  0.6996343
## 3  0.7763740 -0.5678819
## 4  1.7443674  0.7485388
```

```
#plotting clustered data points with k means
ggplot(dailyscale,aes(x=lteq,y=posaff)) +
  geom_point(color=model$cluster, alpha=.6) +#plotting alll the points
  #plotting the centroids
  geom_point(aes(x=model$centers[1,1],y=model$centers[1,2]),color=1,size=5,shape=18)
+
  geom_point(aes(x=model$centers[2,1],y=model$centers[2,2]),color=2,size=5,shape=18)
+
  geom_point(aes(x=model$centers[3,1],y=model$centers[3,2]),color=3,size=5,shape=18)
+
  geom_point(aes(x=model$centers[4,1],y=model$centers[4,2]),color=4,size=5,shape=18)
```



There is a funky animation function for demonstrating the steps in the k-means algorithm. Random Cluster! Average! Move! Average! Cluster! Move! Average! Cluster! …

```
library(animation)
kmeans.ani(dailyscale[,c("lteq","posaff")], centers = 4, pch=c(15,16,17,18),
col=c(1,2,3,4))
```

5.0.0.0.1 Evaluation of Clustering Quality

Numerous measures are available for evaluating a clustering. Many are stored within the model object returned by `kmeans()` A basic concept for evaluating the quality of the clusters is the *sum of squares*. This is typically a sum of the square of the distances between observations.

```
model$totss
```

```
## [1] 2750
```

```
model$withinss
```

```
## [1] 201.2980 329.1644 175.7229 180.1350
```

```
model$tot.withinss
```

```
## [1] 886.3203
```

```
model$betweenss
```

```
## [1] 1863.68
```

Evaluation of the sum of squares can help us both evalaute the quality of any given solution, as well as help us choose the number of clusters, *k*, needed to describe the data.

Evaluation: Within Sum of Squares
The within sum of squares is a measure of how *close* the observations are within the clusters. For a single cluster this is calculated as the average squared distance of each observation within the cluster from the cluster mean. Then the total within sum of squares is the sum of the within sum of squares over all clusters. The total within sum of squares generally decreases as the number of clusters increases. As we increase the number of clusters they individually tend to become smaller and the observations closer together within the clusters. As *k* increases, the changes in the total within sum of squares would be expected to reduce, and so it flattens out. A good value of *k* might be where the reduction in the total weighted sum of squares begins to flatten. General rule of thumb: Aim to minimise the total within sum of squares (achieve within-group similarity).

Evaluation: Between Sum of Squares The between sum or squares is a measure of how *far* the clusters are from each other. General rule of thumb: Aim to maximise the between sum of squares (achieve between-group dissimilarity).

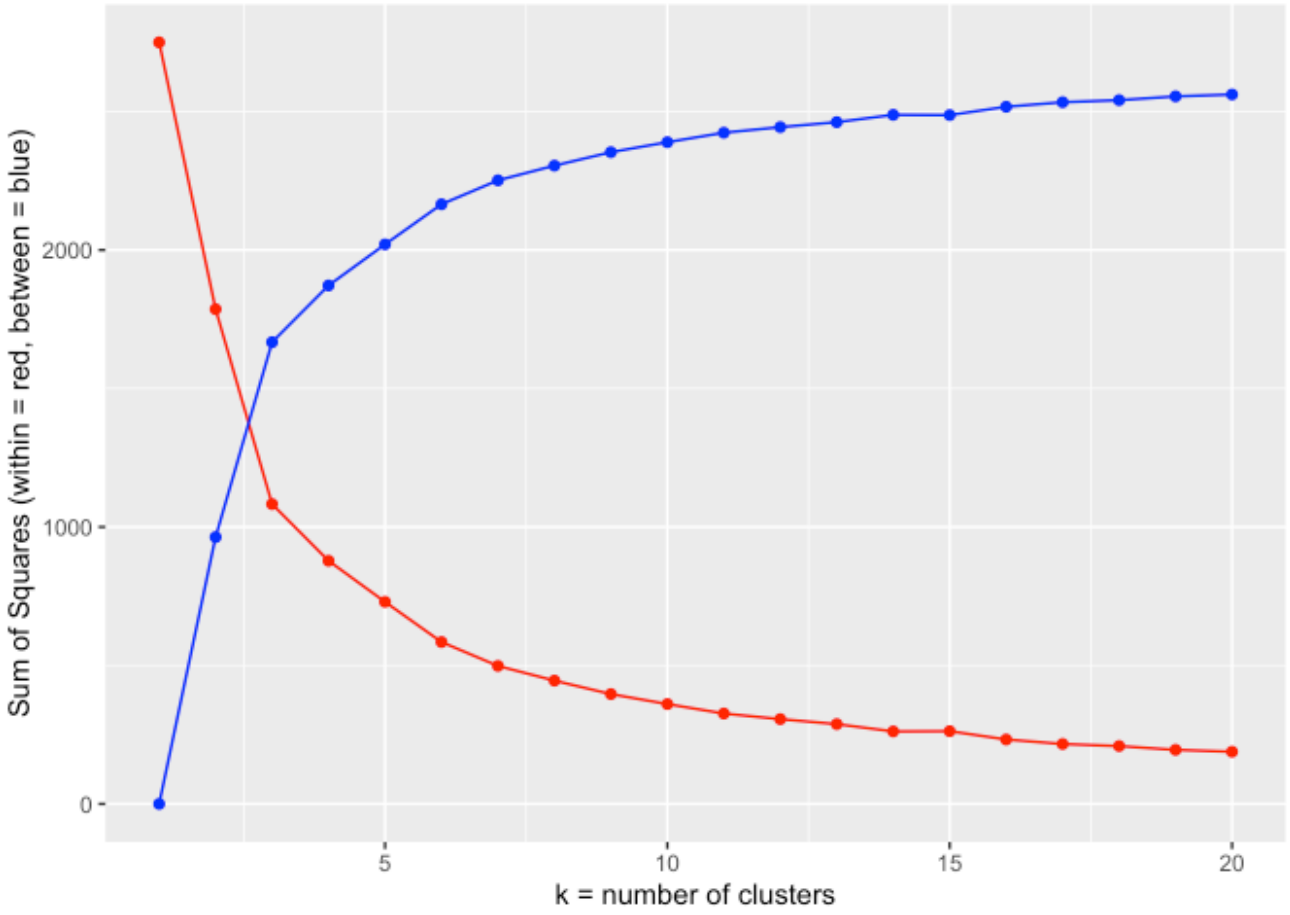
A good clustering will have a small within sum of squares and a large between sum of squares.

So, we need to have a range of solutions to see how the within and between sum of squares looks with different *k*.

```
#making a empty dataframe
criteria <- data.frame()
#setting range of k
nk <- 1:20
#loop for range of clusters
for (k in nk) {
model <- kmeans(dailyscale[,c("lteq","posaff")], k)
criteria <- rbind(criteria,c(k,model$tot.withinss,model$betweenss,model$totss))
}
#renaming columns
names(criteria) <- c("k","tot.withinss","betweenss","totalss")

#scree plot
```

```
ggplot(criteria, aes(x=k)) +
  geom_point(aes(y=tot.withinss),color="red") +
  geom_line(aes(y=tot.withinss),color="red") +
  geom_point(aes(y=betweenss),color="blue") +
  geom_line(aes(y=betweenss),color="blue") +
  xlab("k = number of clusters") + ylab("Sum of Squares (within = red, between = blue)")
```




```
#looking at criteria
round(criteria,2)
```

k	tot.withinss	betweenss	totalss
1	2750.00	0.00	2750
2	1786.49	963.51	2750
3	1082.70	1667.30	2750
4	877.93	1872.07	2750
5	729.55	2020.45	2750
6	584.64	2165.36	2750
7	498.26	2251.74	2750
8	445.17	2304.83	2750
9	396.60	2353.40	2750
10	360.53	2389.47	2750
11	326.35	2423.65	2750
12	306.13	2443.87	2750
13	288.19	2461.81	2750
14	261.78	2488.22	2750
15	262.64	2487.36	2750
16	232.66	2517.34	2750
17	216.25	2533.75	2750
18	208.83	2541.17	2750
19	195.18	2554.82	2750
20	188.38	2561.62	2750

From the scree plot, we might look for 6 clusters (but it is really hard to see any “elbow”).

There are also additional quantitative criteria that can be used to inform selection. For example, the The Calinski-Harabasz criteria, also known as the variance ratio criteria, is the ratio of the between sum of squares (divided by k - 1) to the within sum of squares (divided by n - k) The relative values can be used to compare clusterings of a single dataset, with higher values being better clusterings. The criteria is said to work best for spherical clusters with compact centres (as with normally distributed data) using k-means with Euclidean distance.

And of course this is a well-trodden area of research so there are many many criteria - and packages that calculate them for you - and make automated choices.

```
#from library(fpc)
model.manyCH <- kmeansruns(dailyscale[,c("lteq","posaff")], krange=c(2:20),
criterion="ch",critout = TRUE, plot=FALSE) #better to leave the plot FALSE
```

```
## 2  clusters  745.4072
## 3  clusters  1057.359
## 4  clusters  975.2011
## 5  clusters  978.5366
## 6  clusters  1019.07
## 7  clusters  1033.765
```

```
## 8  clusters  1016.999
## 9  clusters  1015.787
## 10  clusters  1010.407
## 11  clusters  1016.261
## 12  clusters  1025.676
## 13  clusters  1016.533
## 14  clusters  1010.001
## 15  clusters  1007.922
## 16  clusters  1010.128
## 17  clusters  1010.531
## 18  clusters  1019.647
## 19  clusters  1025.941
## 20  clusters  1023.964
```

```
model.manyCH
```

```
## K-means clustering with 3 clusters of sizes 484, 261, 631
##
## Cluster means:
##         lteq     posaff
## 1 -0.4338542 -1.0041922
## 2  1.6409244  0.1951617
## 3 -0.3459522  0.6895275
##
## Clustering vector:
##    1    2    3    4    5    6    7    9   10   11   13   14   15   17   18   19
##    1    1    3    3    3    2    2    3    3    3    3    3    1    3    3    3
##   20   21   22   23   27   28   29   30   31   32   37   38   39   40   41   42
##    1    1    1    3    1    1    1    1    3    3    3    3    3    3    1    3
##   43   44   47   48   49   51   52   53   54   55   56   57   58   59   60   61
##    3    1    3    3    3    3    3    3    3    1    2    1    1    1    1    1
##   62   63   64   66   67   68   70   71   72   73   74   75   76   77   78   79
##    2    3    3    1    1    3    3    3    3    3    3    3    3    1    3    3
##   81   84   85   86   87   88   89   90   91   92   93   95   96   97   98   99
##    3    1    1    1    3    3    3    2    2    2    2    3    3    3    3    2
##  101  102  103  105  106  107  108  110  111  112  113  114  115  116  117  118
##    2    2    3    1    2    2    2    3    2    2    2    2    2    3    2    2
##  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134
##    2    1    1    1    1    1    1    1    1    3    3    3    3    3    3    3
##  135  136  137  138  140  141  143  144  145  146  147  149  150  151  152  153
##    3    2    2    3    2    3    2    2    3    2    2    2    2    2    1    2
##  154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169
##    3    3    3    2    3    3    3    3    3    3    3    3    3    1    3    3
##  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185
##    1    3    3    1    1    1    1    1    1    1    1    3    3    3    1    3
```

```
##  186  187  188  189  190  191  192  193  194  196  197  198  199  201  202  203
##    3    2    2    2    1    2    2    1    1    3    3    3    3    3    2    2
##  204  205  206  208  209  210  211  212  213  214  215  216  217  218  219  220
##    3    3    2    2    2    3    3    3    3    3    3    1    3    1    1    1
##  221  222  223  224  225  226  228  229  230  231  232  233  234  235  236  237
##    1    2    1    1    1    3    3    3    1    1    3    3    3    3    1    1
##  238  239  240  241  242  243  244  245  246  247  248  249  250  251  252  253
##    3    1    1    1    3    3    3    2    2    3    2    3    2    1    2    2
##  254  255  256  257  258  259  260  261  262  263  264  265  266  267  268  269
##    1    2    2    2    3    3    3    3    1    3    3    3    3    1    3    1
##  270  271  272  273  274  275  276  277  278  279  280  281  282  283  285  286
##    2    3    3    3    3    3    1    1    1    3    1    1    1    1    1    1
##  287  288  289  290  291  292  293  294  295  296  297  298  299  300  301  302
##    1    1    3    1    1    1    1    1    1    3    3    3    1    1    3    1
##  303  304  305  306  307  308  309  312  313  314  315  316  317  318  319  320
##    3    3    3    3    3    3    1    3    3    3    3    3    2    1    1    3
##  321  323  324  325  326  327  328  329  330  331  332  333  334  335  343  344
##    3    3    3    3    3    3    3    3    3    3    3    3    3    3    1    3
##  345  346  347  348  349  350  352  353  354  355  356  358  359  360  361  364
##    1    3    1    3    3    3    3    3    3    1    1    1    1    3    3    2
##  365  366  367  368  369  370  371  372  373  374  376  377  378  379  380  381
##    2    2    3    3    3    3    3    3    3    3    1    1    1    3    3    1
##  382  383  384  385  386  387  388  389  390  391  392  393  394  395  396  397
##    1    3    1    3    2    3    3    1    1    3    2    2    1    1    2    2
##  398  399  400  401  402  403  404  405  406  407  408  409  410  411  412  413
##    2    2    2    1    1    2    2    1    2    1    1    1    3    1    1    1
##  414  415  416  417  418  419  422  423  424  425  426  427  428  429  430  431
##    1    3    3    3    3    3    1    3    2    3    2    2    3    2    2    3
##  432  433  434  435  436  437  438  439  441  442  443  444  445  446  447  448
##    3    3    3    3    3    1    1    2    3    3    1    1    2    2    3    3
##  449  450  451  452  453  454  455  456  457  458  459  460  461  462  463  464
##    3    3    3    3    3    3    1    1    1    1    1    1    1    1    3    3
##  465  466  467  468  469  470  471  472  473  474  475  476  477  478  479  480
##    1    1    3    3    3    1    3    3    2    3    3    1    1    1    1    1
##  481  483  485  486  487  488  489  490  491  492  493  494  495  496  497  498
##    3    1    3    3    3    1    1    1    1    1    1    3    1    3    1    1
##  499  500  501  502  503  504  505  506  507  508  509  510  511  512  513  514
##    1    1    3    3    1    1    1    1    1    3    3    3    3    3    3    3
##  515  516  517  518  519  520  521  522  523  524  525  526  527  528  529  530
##    1    1    3    2    3    3    2    3    2    2    1    1    1    3    2    1
##  531  533  534  535  536  537  538  539  540  541  542  543  544  545  546  548
##    2    1    2    2    2    1    1    1    1    1    1    1    3    2    3    2
##  549  550  551  552  553  554  555  556  557  558  559  560  561  562  563  564
##    2    3    3    3    2    2    2    2    1    2    2    3    3    1    1    3
##  565  566  567  568  569  570  571  572  573  574  575  576  577  578  579  580
##    2    1    1    3    3    3    3    3    3    3    3    1    2    2    2    2
```

```
##  581  582  583  584  585  586  587  588  589  590  591  592  593  594  595  596
##    1    1    1    1    3    3    1    1    3    1    1    3    1    1    1    3
##  597  598  599  601  602  603  604  605  606  607  609  610  611  612  613  614
##    1    1    1    3    1    1    3    1    1    1    3    3    3    3    1    1
##  615  616  617  618  619  620  621  622  623  624  625  626  628  629  630  631
##    3    3    3    3    3    3    1    1    1    3    3    3    3    3    3    3
##  632  633  634  635  636  637  638  639  640  641  642  643  644  645  646  647
##    1    1    1    1    1    3    1    1    3    3    3    1    1    1    1    1
##  648  649  650  651  652  653  654  655  656  657  658  659  660  661  662  663
##    1    3    1    1    1    1    3    2    2    1    1    1    1    1    1    1
##  664  665  666  667  668  669  670  671  672  673  674  675  676  677  678  679
##    2    2    2    2    2    2    2    2    2    2    2    3    3    1    2    3
##  680  681  682  684  685  686  687  688  689  690  691  692  693  694  696  697
##    3    3    3    1    1    3    3    2    2    3    3    3    2    2    3    1
##  698  699  700  701  702  703  704  705  706  707  708  709  710  711  712  713
##    1    3    1    1    1    1    2    2    2    2    1    1    2    1    1    1
##  714  715  716  717  719  720  721  722  723  724  725  726  727  728  731  735
##    3    1    1    1    1    1    3    1    1    1    1    1    2    2    2    3
##  736  737  738  739  740  741  742  743  744  745  746  747  748  749  750  751
##    2    3    3    2    1    1    1    2    3    3    3    2    1    2    3    3
##  752  753  754  755  757  758  759  760  761  762  763  764  765  766  767  768
##    1    1    3    3    1    3    1    1    1    3    3    1    3    3    3    3
##  769  770  771  772  773  774  775  776  777  778  779  780  781  782  783  784
##    3    1    3    3    1    1    2    1    2    1    1    1    1    1    1    1
##  785  786  787  788  789  790  791  792  793  794  795  796  797  798  799  800
##    1    1    1    1    3    3    1    1    2    2    2    1    2    3    2    2
##  801  802  803  804  805  806  807  808  809  810  811  812  813  814  815  816
##    2    2    2    1    2    1    3    3    3    3    3    3    3    1    1    1
##  817  819  820  821  822  823  824  825  826  827  828  829  830  831  832  833
##    3    1    1    1    1    3    3    3    1    3    1    2    1    1    3    3
##  834  835  836  837  838  839  840  841  842  843  844  845  846  847  848  849
##    1    2    1    3    3    3    1    3    3    3    3    1    1    1    2    3
##  850  851  852  853  854  855  856  857  858  859  860  861  862  863  864  866
##    3    1    3    1    2    3    3    3    2    3    2    3    2    2    2    2
##  867  868  869  871  872  873  874  875  876  877  878  879  880  881  882  883
##    3    2    2    3    2    2    2    1    1    2    1    1    3    1    3    1
##  884  885  886  887  888  889  890  891  892  893  894  895  896  897  898  899
##    1    3    3    1    1    1    3    1    3    3    3    1    3    3    1    3
##  900  901  902  903  904  905  906  907  908  909  910  911  912  913  914  915
##    3    3    3    3    3    1    1    1    3    3    1    3    1    1    2    2
##  916  917  918  919  920  921  922  923  924  925  926  927  928  929  930  931
##    1    2    3    3    3    3    1    1    3    1    3    3    1    3    3    3
##  932  933  934  935  936  937  938  940  942  943  944  945  946  947  948  949
##    1    3    2    2    2    2    2    3    3    3    3    3    1    1    1    3
##  950  951  952  953  956  957  958  959  960  961  962  963  964  965  966  967
##    2    3    3    1    1    1    3    3    3    1    1    1    1    3    2    2
```

```
##  968  969  970  971  972  973  974  976  977  978  979  980  982  983  984  985
##    3    1    1    2    2    2    1    3    3    2    1    1    3    3    3    1
##  986  987  988  989  990  991  992  993  994  995  996  997  998  999 1000 1001
##    1    2    1    1    3    2    3    3    2    1    2    1    1    3    2    1
## 1002 1003 1004 1005 1006 1007 1008 1009 1011 1012 1013 1014 1015 1016 1017 1018
##    2    2    1    2    1    3    3    3    1    1    3    1    1    2    2    2
## 1019 1020 1021 1022 1023 1024 1025 1026 1027 1028 1029 1030 1031 1032 1033 1034
##    3    3    3    1    2    1    3    2    2    1    3    1    1    1    1    1
## 1035 1036 1037 1038 1039 1040 1041 1042 1043 1044 1045 1046 1047 1048 1049 1050
##    1    1    1    3    3    3    3    3    3    3    3    3    3    3    3    3
## 1051 1052 1053 1054 1055 1056 1057 1058 1059 1060 1061 1062 1064 1065 1066 1067
##    2    3    2    3    3    3    3    1    1    1    1    3    2    1    1    3
## 1068 1069 1070 1071 1072 1073 1074 1075 1076 1077 1078 1079 1080 1081 1082 1083
##    3    3    3    2    3    2    1    2    2    3    3    3    3    1    3    1
## 1084 1085 1086 1087 1088 1089 1090 1091 1092 1093 1094 1095 1096 1097 1098 1099
##    3    3    3    3    3    3    3    3    3    3    3    3    3    3    3    3
## 1100 1101 1102 1103 1104 1105 1106 1107 1108 1109 1110 1111 1112 1113 1114 1115
##    3    3    1    3    1    1    3    1    3    3    2    3    3    3    3    3
## 1116 1117 1118 1119 1120 1121 1122 1123 1124 1125 1126 1127 1128 1129 1130 1131
##    3    3    3    3    2    2    1    2    3    1    3    3    3    3    2    2
## 1132 1133 1134 1135 1136 1137 1138 1139 1140 1141 1142 1143 1144 1145 1146 1147
##    2    3    3    2    3    2    2    2    2    2    3    3    3    3    3    3
## 1148 1149 1150 1151 1152 1153 1154 1156 1157 1158 1160 1161 1162 1163 1164 1165
##    3    3    3    3    1    3    3    1    3    1    3    3    3    1    1    1
## 1166 1167 1168 1169 1170 1171 1172 1173 1174 1175 1176 1177 1178 1179 1180 1181
##    3    2    2    1    1    1    2    3    2    2    1    3    1    3    2    3
## 1182 1183 1184 1185 1186 1187 1188 1189 1190 1191 1192 1193 1194 1195 1196 1197
##    1    3    3    1    3    3    1    1    1    2    2    1    3    2    3    2
## 1198 1199 1200 1201 1202 1203 1204 1205 1207 1208 1209 1210 1211 1212 1213 1215
##    3    3    3    3    3    3    1    1    3    1    3    3    3    3    3    3
## 1216 1217 1218 1219 1220 1221 1222 1223 1224 1225 1226 1227 1228 1229 1230 1231
##    3    1    3    3    3    3    1    3    1    3    3    1    3    1    2    2
## 1232 1233 1234 1235 1236 1237 1238 1239 1240 1241 1242 1243 1244 1245 1246 1247
##    2    2    2    2    1    1    3    3    1    3    3    3    1    1    1    3
## 1248 1249 1250 1251 1252 1253 1254 1255 1256 1257 1258 1259 1260 1261 1262 1263
##    3    3    1    3    2    3    2    3    3    2    2    2    2    2    3    3
## 1264 1265 1266 1267 1268 1269 1270 1271 1272 1273 1274 1275 1276 1277 1278 1279
##    3    3    1    3    1    3    3    3    2    2    1    3    3    2    3    3
## 1280 1281 1282 1283 1284 1285 1286 1287 1288 1289 1290 1291 1292 1293 1294 1295
##    3    3    1    1    1    1    3    1    1    1    3    3    3    3    3    3
## 1296 1297 1298 1299 1300 1301 1302 1304 1305 1306 1307 1308 1309 1310 1311 1312
##    1    1    1    3    3    3    3    1    1    1    1    3    3    3    3    3
## 1313 1314 1315 1316 1317 1318 1319 1320 1321 1322 1323 1324 1325 1326 1327 1328
##    3    1    3    3    1    3    3    3    3    1    3    3    3    1    3    1
## 1329 1330 1331 1332 1333 1334 1335 1336 1337 1338 1339 1340 1341 1342 1343 1344
##    1    1    1    2    1    2    3    1    2    2    2    3    3    3    3    3
```

```
## 1345 1346 1347 1348 1349 1350 1351 1352 1353 1354 1355 1356 1357 1358 1359 1360
##    1    3    3    3    3    3    3    3    3    3    3    3    3    3    3    3
## 1361 1362 1363 1364 1365 1366 1367 1368 1369 1370 1371 1372 1373 1374 1375 1376
##    3    3    3    3    3    3    3    3    1    3    3    2    2    1    1    2
## 1377 1378 1379 1380 1381 1382 1383 1384 1385 1386 1387 1388 1389 1390 1391 1392
##    2    2    2    2    1    3    3    2    3    2    3    1    1    1    1    1
## 1393 1394 1395 1396 1397 1398 1400 1401 1402 1403 1404 1405 1406 1407 1408 1409
##    1    1    1    2    1    2    3    2    1    2    1    1    3    3    1    3
## 1410 1411 1412 1413 1414 1415 1416 1417 1418 1419 1420 1421 1422 1423 1424 1425
##    1    3    1    3    3    1    1    1    1    3    1    3    1    1    1    1
## 1426 1427 1429 1430 1431 1432 1433 1434 1435 1436 1437 1438 1439 1440 1441 1442
##    1    3    2    2    2    2    1    2    2    3    1    1    1    1    1    1
## 1443 1444 1445 1446 1447 1448 1449 1450 1451 1452 1453 1454 1455 1456 1457 1458
##    3    1    1    1    1    1    1    1    2    1    3    3    1    1    3    3
##
## Within cluster sum of squares by cluster:
## [1] 334.8595 334.4631 413.2618
##  (between_SS / total_SS =  60.6 %)
##
## Available components:
##
##  [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
##  [6] "betweenss"    "size"         "iter"         "ifault"       "crit"
## [11] "bestk"
```

```
#another criteria
model.manyASW <- kmeansruns(dailyscale[,c("lteq","posaff")], krange=c(2:20),
criterion="asw",critout = TRUE, plot=FALSE) #better to leave the plot FALSE
```

```
## 2  clusters  0.3315195
## 3  clusters  0.3884696
## 4  clusters  0.3543075
## 5  clusters  0.3425395
## 6  clusters  0.3502535
## 7  clusters  0.3569903
## 8  clusters  0.3417595
## 9  clusters  0.347614
## 10  clusters  0.3521761
## 11  clusters  0.3606332
## 12  clusters  0.3620871
## 13  clusters  0.3577258
## 14  clusters  0.3577506
## 15  clusters  0.3558161
## 16  clusters  0.3615249
## 17  clusters  0.360369
```

```
## 18  clusters  0.3470803
## 19  clusters  0.3568075
## 20  clusters  0.3531087
```

```
model.manyASW
```

```
## K-means clustering with 3 clusters of sizes 631, 484, 261
##
## Cluster means:
##         lteq     posaff
## 1 -0.3459522  0.6895275
## 2 -0.4338542 -1.0041922
## 3  1.6409244  0.1951617
##
## Clustering vector:
##    1    2    3    4    5    6    7    9   10   11   13   14   15   17   18   19
##    2    2    1    1    1    3    3    1    1    1    1    1    2    1    1    1
##   20   21   22   23   27   28   29   30   31   32   37   38   39   40   41   42
##    2    2    2    1    2    2    2    2    1    1    1    1    1    1    2    1
##   43   44   47   48   49   51   52   53   54   55   56   57   58   59   60   61
##    1    2    1    1    1    1    1    1    1    2    3    2    2    2    2    2
##   62   63   64   66   67   68   70   71   72   73   74   75   76   77   78   79
##    3    1    1    2    2    1    1    1    1    1    1    1    1    2    1    1
##   81   84   85   86   87   88   89   90   91   92   93   95   96   97   98   99
##    1    2    2    2    1    1    1    3    3    3    3    1    1    1    1    3
##  101  102  103  105  106  107  108  110  111  112  113  114  115  116  117  118
##    3    3    1    2    3    3    3    1    3    3    3    3    3    1    3    3
##  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134
##    3    2    2    2    2    2    2    2    2    1    1    1    1    1    1    1
##  135  136  137  138  140  141  143  144  145  146  147  149  150  151  152  153
##    1    3    3    1    3    1    3    3    1    3    3    3    3    3    2    3
##  154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169
##    1    1    1    3    1    1    1    1    1    1    1    1    1    2    1    1
##  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185
##    2    1    1    2    2    2    2    2    2    2    2    1    1    1    2    1
##  186  187  188  189  190  191  192  193  194  196  197  198  199  201  202  203
##    1    3    3    3    2    3    3    2    2    1    1    1    1    1    3    3
##  204  205  206  208  209  210  211  212  213  214  215  216  217  218  219  220
##    1    1    3    3    3    1    1    1    1    1    1    2    1    2    2    2
##  221  222  223  224  225  226  228  229  230  231  232  233  234  235  236  237
##    2    3    2    2    2    1    1    1    2    2    1    1    1    1    2    2
##  238  239  240  241  242  243  244  245  246  247  248  249  250  251  252  253
##    1    2    2    2    1    1    1    3    3    1    3    1    3    2    3    3
##  254  255  256  257  258  259  260  261  262  263  264  265  266  267  268  269
##    2    3    3    3    1    1    1    1    2    1    1    1    1    2    1    2
```

```
##  270  271  272  273  274  275  276  277  278  279  280  281  282  283  285  286
##    3    1    1    1    1    1    2    2    2    1    2    2    2    2    2    2
##  287  288  289  290  291  292  293  294  295  296  297  298  299  300  301  302
##    2    2    1    2    2    2    2    2    2    1    1    1    2    2    1    2
##  303  304  305  306  307  308  309  312  313  314  315  316  317  318  319  320
##    1    1    1    1    1    1    2    1    1    1    1    1    3    2    2    1
##  321  323  324  325  326  327  328  329  330  331  332  333  334  335  343  344
##    1    1    1    1    1    1    1    1    1    1    1    1    1    1    2    1
##  345  346  347  348  349  350  352  353  354  355  356  358  359  360  361  364
##    2    1    2    1    1    1    1    1    1    2    2    2    2    1    1    3
##  365  366  367  368  369  370  371  372  373  374  376  377  378  379  380  381
##    3    3    1    1    1    1    1    1    1    1    2    2    2    1    1    2
##  382  383  384  385  386  387  388  389  390  391  392  393  394  395  396  397
##    2    1    2    1    3    1    1    2    2    1    3    3    2    2    3    3
##  398  399  400  401  402  403  404  405  406  407  408  409  410  411  412  413
##    3    3    3    2    2    3    3    2    3    2    2    2    1    2    2    2
##  414  415  416  417  418  419  422  423  424  425  426  427  428  429  430  431
##    2    1    1    1    1    1    2    1    3    1    3    3    1    3    3    1
##  432  433  434  435  436  437  438  439  441  442  443  444  445  446  447  448
##    1    1    1    1    1    2    2    3    1    1    2    2    3    3    1    1
##  449  450  451  452  453  454  455  456  457  458  459  460  461  462  463  464
##    1    1    1    1    1    1    2    2    2    2    2    2    2    2    1    1
##  465  466  467  468  469  470  471  472  473  474  475  476  477  478  479  480
##    2    2    1    1    1    2    1    1    3    1    1    2    2    2    2    2
##  481  483  485  486  487  488  489  490  491  492  493  494  495  496  497  498
##    1    2    1    1    1    2    2    2    2    2    2    1    2    1    2    2
##  499  500  501  502  503  504  505  506  507  508  509  510  511  512  513  514
##    2    2    1    1    2    2    2    2    2    1    1    1    1    1    1    1
##  515  516  517  518  519  520  521  522  523  524  525  526  527  528  529  530
##    2    2    1    3    1    1    3    1    3    3    2    2    2    1    3    2
##  531  533  534  535  536  537  538  539  540  541  542  543  544  545  546  548
##    3    2    3    3    3    2    2    2    2    2    2    2    1    3    1    3
##  549  550  551  552  553  554  555  556  557  558  559  560  561  562  563  564
##    3    1    1    1    3    3    3    3    2    3    3    1    1    2    2    1
##  565  566  567  568  569  570  571  572  573  574  575  576  577  578  579  580
##    3    2    2    1    1    1    1    1    1    1    1    2    3    3    3    3
##  581  582  583  584  585  586  587  588  589  590  591  592  593  594  595  596
##    2    2    2    2    1    1    2    2    1    2    2    1    2    2    2    1
##  597  598  599  601  602  603  604  605  606  607  609  610  611  612  613  614
##    2    2    2    1    2    2    1    2    2    2    1    1    1    1    2    2
##  615  616  617  618  619  620  621  622  623  624  625  626  628  629  630  631
##    1    1    1    1    1    1    2    2    2    1    1    1    1    1    1    1
##  632  633  634  635  636  637  638  639  640  641  642  643  644  645  646  647
##    2    2    2    2    2    1    2    2    1    1    1    2    2    2    2    2
##  648  649  650  651  652  653  654  655  656  657  658  659  660  661  662  663
##    2    1    2    2    2    2    1    3    3    2    2    2    2    2    2    2
```

```
##  664  665  666  667  668  669  670  671  672  673  674  675  676  677  678  679
##    3    3    3    3    3    3    3    3    3    3    3    1    1    2    3    1
##  680  681  682  684  685  686  687  688  689  690  691  692  693  694  696  697
##    1    1    1    2    2    1    1    3    3    1    1    1    3    3    1    2
##  698  699  700  701  702  703  704  705  706  707  708  709  710  711  712  713
##    2    1    2    2    2    2    3    3    3    3    2    2    3    2    2    2
##  714  715  716  717  719  720  721  722  723  724  725  726  727  728  731  735
##    1    2    2    2    2    2    1    2    2    2    2    2    3    3    3    1
##  736  737  738  739  740  741  742  743  744  745  746  747  748  749  750  751
##    3    1    1    3    2    2    2    3    1    1    1    3    2    3    1    1
##  752  753  754  755  757  758  759  760  761  762  763  764  765  766  767  768
##    2    2    1    1    2    1    2    2    2    1    1    2    1    1    1    1
##  769  770  771  772  773  774  775  776  777  778  779  780  781  782  783  784
##    1    2    1    1    2    2    3    2    3    2    2    2    2    2    2    2
##  785  786  787  788  789  790  791  792  793  794  795  796  797  798  799  800
##    2    2    2    2    1    1    2    2    3    3    3    2    3    1    3    3
##  801  802  803  804  805  806  807  808  809  810  811  812  813  814  815  816
##    3    3    3    2    3    2    1    1    1    1    1    1    1    2    2    2
##  817  819  820  821  822  823  824  825  826  827  828  829  830  831  832  833
##    1    2    2    2    2    1    1    1    2    1    2    3    2    2    1    1
##  834  835  836  837  838  839  840  841  842  843  844  845  846  847  848  849
##    2    3    2    1    1    1    2    1    1    1    1    2    2    2    3    1
##  850  851  852  853  854  855  856  857  858  859  860  861  862  863  864  866
##    1    2    1    2    3    1    1    1    3    1    3    1    3    3    3    3
##  867  868  869  871  872  873  874  875  876  877  878  879  880  881  882  883
##    1    3    3    1    3    3    3    2    2    3    2    2    1    2    1    2
##  884  885  886  887  888  889  890  891  892  893  894  895  896  897  898  899
##    2    1    1    2    2    2    1    2    1    1    1    2    1    1    2    1
##  900  901  902  903  904  905  906  907  908  909  910  911  912  913  914  915
##    1    1    1    1    1    2    2    2    1    1    2    1    2    2    3    3
##  916  917  918  919  920  921  922  923  924  925  926  927  928  929  930  931
##    2    3    1    1    1    1    2    2    1    2    1    1    2    1    1    1
##  932  933  934  935  936  937  938  940  942  943  944  945  946  947  948  949
##    2    1    3    3    3    3    3    1    1    1    1    1    2    2    2    1
##  950  951  952  953  956  957  958  959  960  961  962  963  964  965  966  967
##    3    1    1    2    2    2    1    1    1    2    2    2    2    1    3    3
##  968  969  970  971  972  973  974  976  977  978  979  980  982  983  984  985
##    1    2    2    3    3    3    2    1    1    3    2    2    1    1    1    2
##  986  987  988  989  990  991  992  993  994  995  996  997  998  999 1000 1001
##    2    3    2    2    1    3    1    1    3    2    3    2    2    1    3    2
## 1002 1003 1004 1005 1006 1007 1008 1009 1011 1012 1013 1014 1015 1016 1017 1018
##    3    3    2    3    2    1    1    1    2    2    1    2    2    3    3    3
## 1019 1020 1021 1022 1023 1024 1025 1026 1027 1028 1029 1030 1031 1032 1033 1034
##    1    1    1    2    3    2    1    3    3    2    1    2    2    2    2    2
## 1035 1036 1037 1038 1039 1040 1041 1042 1043 1044 1045 1046 1047 1048 1049 1050
##    2    2    2    1    1    1    1    1    1    1    1    1    1    1    1    1
```

```
## 1051 1052 1053 1054 1055 1056 1057 1058 1059 1060 1061 1062 1064 1065 1066 1067
##    3    1    3    1    1    1    1    2    2    2    2    1    3    2    2    1
## 1068 1069 1070 1071 1072 1073 1074 1075 1076 1077 1078 1079 1080 1081 1082 1083
##    1    1    1    3    1    3    2    3    3    1    1    1    1    2    1    2
## 1084 1085 1086 1087 1088 1089 1090 1091 1092 1093 1094 1095 1096 1097 1098 1099
##    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1
## 1100 1101 1102 1103 1104 1105 1106 1107 1108 1109 1110 1111 1112 1113 1114 1115
##    1    1    2    1    2    2    1    2    1    1    3    1    1    1    1    1
## 1116 1117 1118 1119 1120 1121 1122 1123 1124 1125 1126 1127 1128 1129 1130 1131
##    1    1    1    1    3    3    2    3    1    2    1    1    1    1    3    3
## 1132 1133 1134 1135 1136 1137 1138 1139 1140 1141 1142 1143 1144 1145 1146 1147
##    3    1    1    3    1    3    3    3    3    3    1    1    1    1    1    1
## 1148 1149 1150 1151 1152 1153 1154 1156 1157 1158 1160 1161 1162 1163 1164 1165
##    1    1    1    1    2    1    1    2    1    2    1    1    1    2    2    2
## 1166 1167 1168 1169 1170 1171 1172 1173 1174 1175 1176 1177 1178 1179 1180 1181
##    1    3    3    2    2    2    3    1    3    3    2    1    2    1    3    1
## 1182 1183 1184 1185 1186 1187 1188 1189 1190 1191 1192 1193 1194 1195 1196 1197
##    2    1    1    2    1    1    2    2    2    3    3    2    1    3    1    3
## 1198 1199 1200 1201 1202 1203 1204 1205 1207 1208 1209 1210 1211 1212 1213 1215
##    1    1    1    1    1    1    2    2    1    2    1    1    1    1    1    1
## 1216 1217 1218 1219 1220 1221 1222 1223 1224 1225 1226 1227 1228 1229 1230 1231
##    1    2    1    1    1    1    2    1    2    1    1    2    1    2    3    3
## 1232 1233 1234 1235 1236 1237 1238 1239 1240 1241 1242 1243 1244 1245 1246 1247
##    3    3    3    3    2    2    1    1    2    1    1    1    2    2    2    1
## 1248 1249 1250 1251 1252 1253 1254 1255 1256 1257 1258 1259 1260 1261 1262 1263
##    1    1    2    1    3    1    3    1    1    3    3    3    3    3    1    1
## 1264 1265 1266 1267 1268 1269 1270 1271 1272 1273 1274 1275 1276 1277 1278 1279
##    1    1    2    1    2    1    1    1    3    3    2    1    1    3    1    1
## 1280 1281 1282 1283 1284 1285 1286 1287 1288 1289 1290 1291 1292 1293 1294 1295
##    1    1    2    2    2    2    1    2    2    2    1    1    1    1    1    1
## 1296 1297 1298 1299 1300 1301 1302 1304 1305 1306 1307 1308 1309 1310 1311 1312
##    2    2    2    1    1    1    1    2    2    2    2    1    1    1    1    1
## 1313 1314 1315 1316 1317 1318 1319 1320 1321 1322 1323 1324 1325 1326 1327 1328
##    1    2    1    1    2    1    1    1    1    2    1    1    1    2    1    2
## 1329 1330 1331 1332 1333 1334 1335 1336 1337 1338 1339 1340 1341 1342 1343 1344
##    2    2    2    3    2    3    1    2    3    3    3    1    1    1    1    1
## 1345 1346 1347 1348 1349 1350 1351 1352 1353 1354 1355 1356 1357 1358 1359 1360
##    2    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1
## 1361 1362 1363 1364 1365 1366 1367 1368 1369 1370 1371 1372 1373 1374 1375 1376
##    1    1    1    1    1    1    1    1    2    1    1    3    3    2    2    3
## 1377 1378 1379 1380 1381 1382 1383 1384 1385 1386 1387 1388 1389 1390 1391 1392
##    3    3    3    3    2    1    1    3    1    3    1    2    2    2    2    2
## 1393 1394 1395 1396 1397 1398 1400 1401 1402 1403 1404 1405 1406 1407 1408 1409
##    2    2    2    3    2    3    1    3    2    3    2    2    1    1    2    1
## 1410 1411 1412 1413 1414 1415 1416 1417 1418 1419 1420 1421 1422 1423 1424 1425
##    2    1    2    1    1    2    2    2    2    1    2    1    2    2    2    2
```

```
## 1426 1427 1429 1430 1431 1432 1433 1434 1435 1436 1437 1438 1439 1440 1441 1442
##    2    1    3    3    3    3    2    3    3    1    2    2    2    2    2    2
## 1443 1444 1445 1446 1447 1448 1449 1450 1451 1452 1453 1454 1455 1456 1457 1458
##    1    2    2    2    2    2    2    2    3    2    1    1    2    2    1    1
##
## Within cluster sum of squares by cluster:
## [1] 413.2618 334.8595 334.4631
##  (between_SS / total_SS =  60.6 %)
##
## Available components:
##
##  [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
##  [6] "betweenss"    "size"         "iter"         "ifault"       "crit"
## [11] "bestk"
```

As well, there are other metrics ...

```
#obtaining distance matrix
dist.all <-
dist(dailyscale[,c("lteq","posaff")],method="euclidean",diag=TRUE,upper=FALSE)
#obtaining metrics
cluster.stats(dist.all,clustering=model$cluster)
```

```
## $n
## [1] 1376
##
## $cluster.number
## [1] 20
##
## $cluster.size
##  [1]  93 101 116  44  62  52 100  76  84  30 118  60  75  44  92  42  16  27  55
## [20]  89
##
## $min.cluster.size
## [1] 16
##
## $noisen
## [1] 0
##
## $diameter
##  [1] 0.7929847 0.8109655 1.0261149 2.3029433 1.0584017 1.4089678 0.8651722
##  [8] 0.7263207 0.6982825 1.4798406 1.0403386 1.2808520 1.0261149 1.1512972
## [15] 0.9369996 1.3965650 2.0427084 2.4438887 1.2160490 1.9824618
##
## $average.distance
```

```
##  [1] 0.3400096 0.3413816 0.3801126 0.7562126 0.4309641 0.5409203 0.3444528
##  [8] 0.3184404 0.3136617 0.6138922 0.4269003 0.5356949 0.4191472 0.4703920
## [15] 0.4167709 0.4830194 0.9926368 0.8897549 0.4217370 0.6373993
##
## $median.distance
##  [1] 0.3404514 0.3404514 0.3635339 0.6692652 0.4246881 0.5453009 0.3404514
##  [8] 0.3322502 0.3018737 0.5948243 0.3964924 0.5286565 0.3964924 0.4637077
## [15] 0.3964924 0.4637077 0.9815921 0.7968182 0.3964924 0.6037474
##
## $separation
##  [1] 0.09088348 0.09088348 0.09088348 0.09088348 0.09088348 0.09088348
##  [7] 0.09088348 0.09088348 0.09088348 0.09088348 0.09088348 0.09088348
## [13] 0.09088348 0.13216412 0.09088348 0.09088348 0.13216412 0.13216412
## [19] 0.09088348 0.09088348
##
## $average.toother
##  [1] 1.763301 1.391046 1.426488 2.922640 1.508137 1.997886 1.539863 1.385921
##  [9] 1.604138 2.338607 1.780105 2.039176 1.755922 2.050502 1.723909 2.005250
## [17] 3.120922 2.984987 2.093672 2.444394
##
## $separation.matrix
##             [,1]       [,2]       [,3]       [,4]       [,5]       [,6]
##  [1,] 0.00000000 0.61662059 0.68090281 2.31010541 1.32107833 1.85483081
##  [2,] 0.61662059 0.00000000 0.09088348 2.07926945 0.09595597 0.46370770
##  [3,] 0.68090281 0.09088348 0.00000000 1.11636474 0.09088348 0.82356047
##  [4,] 2.31010541 2.07926945 1.11636474 0.00000000 1.32164119 1.92690709
##  [5,] 1.32107833 0.09595597 0.09088348 1.32164119 0.00000000 0.09088348
##  [6,] 1.85483081 0.46370770 0.82356047 1.92690709 0.09088348 0.00000000
##  [7,] 0.19191194 0.49328019 0.09088348 1.48453873 0.74300153 1.48235567
##  [8,] 0.09088348 0.09088348 0.09595597 1.96279202 0.52865648 1.15617164
##  [9,] 0.09088348 0.09088348 0.67169179 2.63517167 0.89206403 1.32164119
## [10,] 1.18805150 1.68471180 0.86712873 0.33341933 1.37669408 2.21471649
## [11,] 0.81795128 0.09088348 0.85802228 2.88770029 0.67169179 0.88252514
## [12,] 1.57774816 0.97586196 0.09595597 0.09088348 0.20554020 1.04033864
## [13,] 0.86837274 0.87110063 0.09088348 0.55183962 0.63618433 1.45729816
## [14,] 1.96224197 0.95955970 0.37598464 0.86712873 0.13216412 0.13216412
## [15,] 1.27518468 0.09088348 0.57808582 2.44234225 0.09088348 0.09595597
## [16,] 0.33341933 1.27236866 0.75196928 1.45077137 1.50393855 2.29025038
## [17,] 3.11014232 1.87399926 1.53776898 1.63590257 1.07263317 0.48831190
## [18,] 2.78340536 2.01507538 1.15147164 0.20554020 1.15505270 1.31552934
## [19,] 0.09088348 1.10735825 0.92741541 2.30294329 1.59363636 2.26169160
## [20,] 1.81766952 0.86712873 1.53124103 3.23510845 0.90352923 0.19191194
##             [,7]       [,8]       [,9]      [,10]      [,11]      [,12]
##  [1,] 0.19191194 0.09088348 0.09088348 1.18805150 0.81795128 1.57774816
##  [2,] 0.49328019 0.09088348 0.09088348 1.68471180 0.09088348 0.97586196
##  [3,] 0.09088348 0.09595597 0.67169179 0.86712873 0.85802228 0.09595597
```

```
##  [4,] 1.48453873 1.96279202 2.63517167 0.33341933 2.88770029 0.09088348
##  [5,] 0.74300153 0.52865648 0.89206403 1.37669408 0.67169179 0.20554020
##  [6,] 1.48235567 1.15617164 1.32164119 2.21471649 0.88252514 1.04033864
##  [7,] 0.00000000 0.09088348 0.60195150 0.46370770 1.15364999 0.72491939
##  [8,] 0.09088348 0.00000000 0.09088348 1.23324119 0.46370770 1.05551567
##  [9,] 0.60195150 0.09088348 0.00000000 1.78901836 0.09088348 1.64134721
## [10,] 0.46370770 1.23324119 1.78901836 0.00000000 2.33393530 0.27265043
## [11,] 1.15364999 0.46370770 0.09088348 2.33393530 0.00000000 1.78412807
## [12,] 0.72491939 1.05551567 1.64134721 0.27265043 1.78412807 0.00000000
## [13,] 0.09088348 0.59482428 1.26087237 0.09088348 1.64978611 0.09595597
## [14,] 1.26044422 1.23789422 1.74859483 1.45413562 1.65388019 0.20554020
## [15,] 1.18148519 0.75196928 0.66082060 2.30729997 0.09088348 1.32107833
## [16,] 0.09088348 0.63618433 1.02770099 0.21234406 1.70480671 1.07263317
## [17,] 2.44942991 2.33204530 2.71544465 2.51583791 2.36662223 1.01797188
## [18,] 1.93097266 2.20885781 2.78272314 1.38570970 2.72492440 0.13216412
## [19,] 0.19191194 0.45441738 0.75196928 1.05551567 1.46674489 1.69875247
## [20,] 2.01239506 1.39331415 1.09060171 3.20354819 0.09088348 2.15169920
##           [,13]     [,14]      [,15]      [,16]     [,17]     [,18]      [,19]
##  [1,] 0.86837274 1.9622420 1.27518468 0.33341933 3.1101423 2.7834054 0.09088348
##  [2,] 0.87110063 0.9595597 0.09088348 1.27236866 1.8739993 2.0150754 1.10735825
##  [3,] 0.09088348 0.3759846 0.57808582 0.75196928 1.5377690 1.1514716 0.92741541
##  [4,] 0.55183962 0.8671287 2.44234225 1.45077137 1.6359026 0.2055402 2.30294329
##  [5,] 0.63618433 0.1321641 0.09088348 1.50393855 1.0726332 1.1550527 1.59363636
##  [6,] 1.45729816 0.1321641 0.09595597 2.29025038 0.4883119 1.3155293 2.26169160
##  [7,] 0.09088348 1.2604442 1.18148519 0.09088348 2.4494299 1.9309727 0.19191194
##  [8,] 0.59482428 1.2378942 0.75196928 0.63618433 2.3320453 2.2088578 0.45441738
##  [9,] 1.26087237 1.7485948 0.66082060 1.02770099 2.7154447 2.7827231 0.75196928
## [10,] 0.09088348 1.4541356 2.30729997 0.21234406 2.5158379 1.3857097 1.05551567
## [11,] 1.64978611 1.6538802 0.09088348 1.70480671 2.3666222 2.7249244 1.46674489
## [12,] 0.09595597 0.2055402 1.32107833 1.07263317 1.0179719 0.1321641 1.69875247
## [13,] 0.00000000 0.8711006 1.44521455 0.09088348 2.0554020 1.2039030 0.86360373
## [14,] 0.87110063 0.0000000 0.95955970 1.87114867 0.1321641 0.2055402 2.18528987
## [15,] 1.44521455 0.9595597 0.00000000 1.99943647 1.5593174 2.0522297 1.82777260
## [16,] 0.09088348 1.8711487 1.99943647 0.00000000 3.0501851 2.2536474 0.19191194
## [17,] 2.05540198 0.1321641 1.55931735 3.05018514 0.0000000 0.4637077 3.36525579
## [18,] 1.20390299 0.2055402 2.05222974 2.25364739 0.4637077 0.0000000 2.89967755
## [19,] 0.86360373 2.1852899 1.82777260 0.19191194 3.3652558 2.8996776 0.00000000
## [20,] 2.39826228 1.4916836 0.09088348 2.67893369 1.3155293 2.6761921 2.46134697
##           [,20]
##  [1,] 1.81766952
##  [2,] 0.86712873
##  [3,] 1.53124103
##  [4,] 3.23510845
##  [5,] 0.90352923
##  [6,] 0.19191194
##  [7,] 2.01239506
```

```
##  [8,] 1.39331415
##  [9,] 1.09060171
## [10,] 3.20354819
## [11,] 0.09088348
## [12,] 2.15169920
## [13,] 2.39826228
## [14,] 1.49168360
## [15,] 0.09088348
## [16,] 2.67893369
## [17,] 1.31552934
## [18,] 2.67619210
## [19,] 2.46134697
## [20,] 0.00000000
##
## $ave.between.matrix
##           [,1]      [,2]      [,3]     [,4]      [,5]      [,6]      [,7]
##  [1,] 0.0000000 1.2814198 1.4809042 3.423008 1.9466986 2.6911161 0.9152320
##  [2,] 1.2814198 0.0000000 0.8807207 3.094794 0.8915917 1.4682081 1.1879449
##  [3,] 1.4809042 0.8807207 0.0000000 2.284587 0.7518602 1.6135772 0.8518119
##  [4,] 3.4230076 3.0947941 2.2845873 0.000000 2.5021223 3.0667793 2.5639580
##  [5,] 1.9466986 0.8915917 0.7518602 2.502122 0.0000000 0.9916272 1.4628966
##  [6,] 2.6911161 1.4682081 1.6135772 3.066779 0.9916272 0.0000000 2.3393522
##  [7,] 0.9152320 1.1879449 0.8518119 2.563958 1.4628966 2.3393522 0.0000000
##  [8,] 0.7415906 0.6969468 0.8552914 2.982048 1.2713189 2.0349267 0.6680151
##  [9,] 0.7254621 0.8249518 1.3838162 3.577932 1.6263249 2.2114589 1.1979022
## [10,] 2.2375845 2.4644078 1.7350057 1.625060 2.2926052 3.1471289 1.4882132
## [11,] 1.5265256 0.8357834 1.6342157 3.860158 1.5266513 1.7653782 1.8096745
## [12,] 2.5619022 1.9896240 1.2303907 1.265139 1.3583369 1.9680604 1.7402728
## [13,] 1.6777258 1.6524612 0.9273351 1.805899 1.4975223 2.3754814 0.8438004
## [14,] 2.8149872 1.8258157 1.3995531 2.036733 1.0287891 1.1635044 2.1556939
## [15,] 1.9968175 0.8470595 1.4717753 3.514970 1.0717698 1.0624622 1.9572743
## [16,] 1.2247208 1.9415944 1.5492060 2.643166 2.1948689 3.0900956 0.8348671
## [17,] 4.1184014 2.9794937 2.7576918 2.983809 2.2313688 1.7341672 3.5369847
## [18,] 3.8247153 3.0752893 2.4204900 1.457763 2.2907661 2.4794878 3.0156154
## [19,] 0.7957739 1.8926654 1.8080719 3.308075 2.3986277 3.2364933 1.0342132
## [20,] 2.7225115 1.7181232 2.4044838 4.384232 1.9543299 1.5538672 2.8498189
##           [,8]      [,9]     [,10]     [,11]    [,12]     [,13]    [,14]
##  [1,] 0.7415906 0.7254621 2.2375845 1.5265256 2.561902 1.6777258 2.814987
##  [2,] 0.6969468 0.8249518 2.4644078 0.8357834 1.989624 1.6524612 1.825816
##  [3,] 0.8552914 1.3838162 1.7350057 1.6342157 1.230391 0.9273351 1.399553
##  [4,] 2.9820477 3.5779324 1.6250600 3.8601580 1.265139 1.8058989 2.036733
##  [5,] 1.2713189 1.6263249 2.2926052 1.5266513 1.358337 1.4975223 1.028789
##  [6,] 2.0349267 2.2114589 3.1471289 1.7653782 1.968060 2.3754814 1.163504
##  [7,] 0.6680151 1.1979022 1.4882132 1.8096745 1.740273 0.8438004 2.155694
##  [8,] 0.0000000 0.6597308 2.0567851 1.2135847 2.001210 1.3212649 2.143955
##  [9,] 0.6597308 0.0000000 2.6257503 0.8648130 2.564590 1.9145003 2.570411
```

```
## [10,] 2.0567851 2.6257503 0.0000000 3.1845931 1.551442 0.9398655 2.442709
## [11,] 1.2135847 0.8648130 3.1845931 0.0000000 2.740902 2.3897909 2.441658
## [12,] 2.0012098 2.5645895 1.5514418 2.7409018 0.000000 1.1417886 1.065257
## [13,] 1.3212649 1.9145003 0.9398655 2.3897909 1.141789 0.0000000 1.829093
## [14,] 2.1439551 2.5704113 2.4427088 2.4416578 1.065257 1.8290927 0.000000
## [15,] 1.4569474 1.3966546 3.1440317 0.8332790 2.360750 2.3175557 1.850589
## [16,] 1.3442191 1.7723930 1.2395972 2.5024700 2.133307 1.0794921 2.768739
## [17,] 3.4356942 3.7554950 3.7744331 3.3917133 2.318846 3.2249765 1.489223
## [18,] 3.2243389 3.7500680 2.5915705 3.7618589 1.376342 2.3848217 1.459359
## [19,] 1.2624788 1.4422990 1.9143496 2.2529259 2.665443 1.6259694 3.145491
## [20,] 2.2998615 2.0569548 4.0753295 1.2708489 3.232918 3.2512401 2.574424
##          [,15]     [,16]    [,17]    [,18]     [,19]    [,20]
##  [1,] 1.9968175 1.2247208 4.118401 3.824715 0.7957739 2.722512
##  [2,] 0.8470595 1.9415944 2.979494 3.075289 1.8926654 1.718123
##  [3,] 1.4717753 1.5492060 2.757692 2.420490 1.8080719 2.404484
##  [4,] 3.5149699 2.6431663 2.983809 1.457763 3.3080747 4.384232
##  [5,] 1.0717698 2.1948689 2.231369 2.290766 2.3986277 1.954330
##  [6,] 1.0624622 3.0900956 1.734167 2.479488 3.2364933 1.553867
##  [7,] 1.9572743 0.8348671 3.536985 3.015615 1.0342132 2.849819
##  [8,] 1.4569474 1.3442191 3.435694 3.224339 1.2624788 2.299862
##  [9,] 1.3966546 1.7723930 3.755495 3.750068 1.4422990 2.056955
## [10,] 3.1440317 1.2395972 3.774433 2.591571 1.9143496 4.075330
## [11,] 0.8332790 2.5024700 3.391713 3.761859 2.2529259 1.270849
## [12,] 2.3607499 2.1333072 2.318846 1.376342 2.6654429 3.232918
## [13,] 2.3175557 1.0794921 3.224977 2.384822 1.6259694 3.251240
## [14,] 1.8505887 2.7687386 1.489223 1.459359 3.1454907 2.574424
## [15,] 0.0000000 2.7238412 2.671744 3.213440 2.6599960 1.029640
## [16,] 2.7238412 0.0000000 4.169549 3.382188 0.8260545 3.599761
## [17,] 2.6717435 4.1695488 0.000000 1.933530 4.5197066 3.020719
## [18,] 3.2134405 3.3821884 1.933530 0.000000 3.9439969 3.933743
## [19,] 2.6599960 0.8260545 4.519707 3.943997 0.0000000 3.437965
## [20,] 1.0296396 3.5997608 3.020719 3.933743 3.4379651 0.000000
##
## $average.between
## [1] 1.837197
##
## $average.within
## [1] 0.4468272
##
## $n.between
## [1] 890805
##
## $n.within
## [1] 55195
##
## $max.diameter
```

```
## [1] 2.443889
##
## $min.separation
## [1] 0.09088348
##
## $within.cluster.ss
## [1] 188.3838
##
## $clus.avg.silwidths
##         1         2         3         4         5         6         7         8
## 0.3855308 0.3627779 0.3488332 0.3178225 0.2862080 0.3327062 0.3766627 0.3132159
##         9        10        11        12        13        14        15        16
## 0.4293266 0.2915247 0.3296523 0.3334063 0.3700007 0.4371085 0.3479506 0.2784984
##        17        18        19        20
## 0.2331334 0.2067116 0.3415744 0.3312162
##
## $avg.silwidth
## [1] 0.3455233
##
## $g2
## NULL
##
## $g3
## NULL
##
## $pearsongamma
## [1] 0.3461202
##
## $dunn
## [1] 0.03718806
##
## $dunn2
## [1] 0.6646246
##
## $entropy
## [1] 2.898485
##
## $wb.ratio
## [1] 0.2432113
##
## $ch
## [1] 970.4576
##
## $cwidegap
##  [1] 0.2878679 0.1919119 0.1321641 0.9088348 0.1817670 0.2726504 0.1817670
##  [8] 0.1919119 0.2878679 0.4246881 0.2878679 0.2878679 0.1817670 0.2123441
```

```
## [15] 0.1919119 0.6166206 0.7249194 0.8146296 0.2878679 0.4544174
##
## $widestgap
## [1] 0.9088348
##
## $sindex
## [1] 0.09088348
##
## $corrected.rand
## NULL
##
## $vi
## NULL
```

And then there are more ...

```
#library(clusterCrit)
#running a cluster analysis
model <- kmeans(dailyscale[,c("lteq","posaff")], centers=4)
#calculating various internal clustering validation or quality criteria
ic <- intCriteria(traj=as.matrix(dailyscale[,c("lteq","posaff")]),
part=model$cluster, crit="all")
ic
```

```
## $ball_hall
## [1] 0.7095635
##
## $banfeld_raftery
## [1] -703.4428
##
## $c_index
## [1] 0.1128338
##
## $calinski_harabasz
## [1] 960.0025
##
## $davies_bouldin
## [1] 0.981463
##
## $det_ratio
## [1] 9.766538
##
## $dunn
## [1] 0.01765742
##
```

```
## $gamma
## [1] 0.7235238
##
## $g_plus
## [1] 0.05400294
##
## $gdi11
## [1] 0.01765742
##
## $gdi12
## [1] 0.133414
##
## $gdi13
## [1] 0.04698696
##
## $gdi21
## [1] 0.7629745
##
## $gdi22
## [1] 5.7648
##
## $gdi23
## [1] 2.030299
##
## $gdi31
## [1] 0.2796567
##
## $gdi32
## [1] 2.113
##
## $gdi33
## [1] 0.7441752
##
## $gdi41
## [1] 0.2295217
##
## $gdi42
## [1] 1.734196
##
## $gdi43
## [1] 0.6107646
##
## $gdi51
## [1] 0.123436
##
## $gdi52
```

```
## [1] 0.9326446
##
## $gdi53
## [1] 0.3284672
##
## $ksq_detw
## [1] 3055637
##
## $log_det_ratio
## [1] 3135.852
##
## $log_ss_ratio
## [1] 0.7415234
##
## $mcclain_rao
## [1] 0.4606744
##
## $pbm
## [1] 1.284308
##
## $point_biserial
## [1] -0.4882913
##
## $ray_turi
## [1] 0.4620738
##
## $ratkowsky_lance
## [1] 0.4115
##
## $scott_symons
## [1] -3472.898
##
## $sd_scat
## [1] 0.3680023
##
## $sd_dis
## [1] 1.497357
##
## $s_dbw
## [1] 3.681673
##
## $silhouette
## [1] 0.325404
##
## $tau
## [1] 0.4522178
```

```
##
## $trace_w
## [1] 887.3456
##
## $trace_wib
## [1] 4.355222
##
## $wemmert_gancarski
## [1] 0.4640056
##
## $xie_beni
## [1] 78.07365
```

Calculation for many possible *k* can be looped and all the possibilites plotted. So much guidance! So much fun!

As far as I can tell, there is not "standard reporting" of cluster analysis resutls in the psychological literature. Different authors report different things - but all are using some metrics to justify the choice of *k*, and to support why the chosen cluster solution is a good description of the data.

Obtaining a stable, replicable solution

Recall that k-means begins the iterations with a random cluster assignment. Different starting points may lead to different solutions. So, it may be useful to start many times to locate a stable solution. This is automated within the `kmeans()` function.

```
#kmeans with nstart = 1
km.res <- kmeans(dailyscale[,c("lteq","posaff")], centers=4, nstart = 1)
km.res$tot.withinss
```

```
## [1] 886.1714
```

```
#kmeans with nstart = 25
km.res <- kmeans(dailyscale[,c("lteq","posaff")], centers=4, nstart = 25)
km.res$tot.withinss
```

```
## [1] 877.9312
```

```
#kmeans with nstart = 50
km.res <- kmeans(dailyscale[,c("lteq","posaff")], centers=4, nstart = 50)
km.res$tot.withinss
```

```
## [1] 877.9312
```

The improvement can be seen over the single random start. Recommended to do 25+ or 50 for stable solutions.

Replication It may also be informative to repeat the procedure on randomly selected portions of the sample. If the cluster solution replicates in (random) subsets of the data - that would be strong evidence that the typology is pervasive and meaningful.

After Clustering

After finding a suitable cluster solution, each individual is placed in a cluster. Formally, we obtain a vector of cluster assignments - a new categorical, grouping variable.
What next?

Well, we can both describe the clusters and use this new cluster variable in some other analysis - ANOVAs to test group differences, Chi-square tests, Multinomial regressions … the cluster variable can be used as a predictor, a correlate, an outcome.

Describing the Clusters

First we merge the vector of cluster assignments back into the data set.

```
dailyscale.clus <- cbind(km.res$cluster,dailyscale)
names(dailyscale.clus)[1] <- "cluster"
head(dailyscale.clus[,c(1:4,6)],4)
```

cluster	id	slphrs	weath	pss
2	1010	-0.6623479	-0.7732500	-0.1732048
2	1011	-2.8774184	-0.0005615	0.1923317
3	1012	0.9989549	0.7721270	1.2889411
3	1013	0.1683035	-0.0005615	0.5578682

We can describe the different clusters - and potentially name the clusters.

```
library(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.0 ──
```

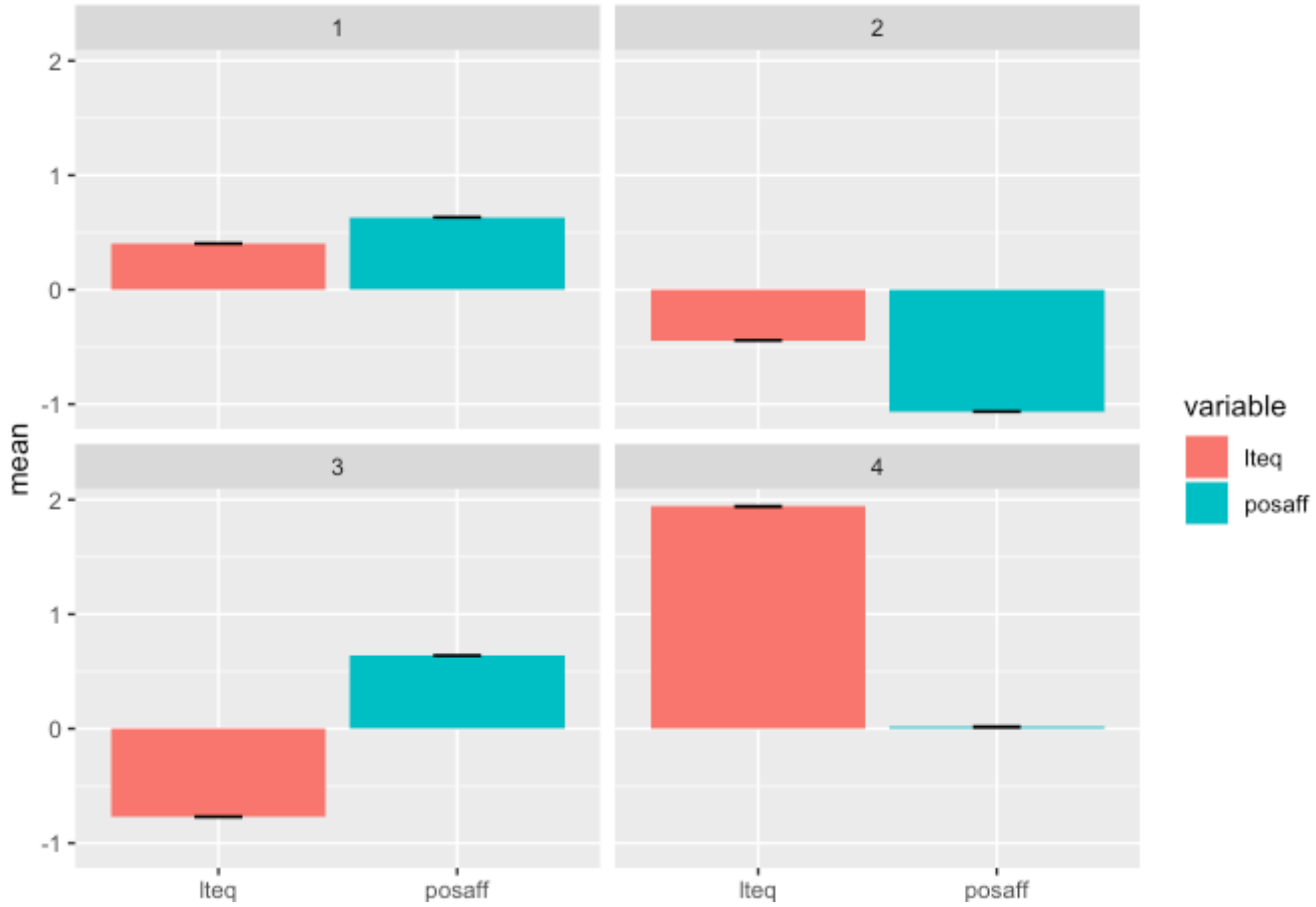
```
## ✓ tibble  2.1.3     ✓ dplyr   0.8.3
## ✓ tidyr   1.0.0     ✓ stringr 1.4.0
## ✓ readr   1.3.1     ✓ forcats 0.4.0
## ✓ purrr   0.3.3
```

```
## ── Conflicts ──────────────────────────────────────────────────
tidyverse_conflicts() ──
## x psych::%+%()    masks ggplot2::%+%()
## x psych::alpha()  masks ggplot2::alpha()
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
# Gather the data to 'long' format so the clustering variables are all in one column
#gather() has been replaced by pivot_longer()
longdata <- dailyscale.clus %>%
  pivot_longer(c(lteq, posaff), names_to = "variable", values_to = "value")

# Create the summary statistics seperately for cluster and variable (i.e. lteq,
posaff)
summary <- longdata %>%
             group_by(cluster, variable) %>%
             summarise(mean = mean(value), se = sd(value) / length(value))

# Plot
ggplot(summary, aes(x = variable, y = mean, fill = variable)) +
  geom_bar(stat = 'identity', position = 'dodge') +
  geom_errorbar(aes(ymin = mean - se, ymax = mean + se),
                 width = 0.2,
                 position = position_dodge(0.9)) +
  facet_wrap(~cluster)
```



From this plot we can see the multivariate "profile" of each cluster - and use that to name the clusters.
1 = Vigorous Exercisers
2 = Happy Sedentary
3 = Happy Exercisers
4 = Unhappy Sedentary

Analyzing the Clusters

Now that we have clusters = groups, we can analyze them. For example, we can take our 4-cluster solution and see if the clusters differ on another variable.

Let's see how the cluster groups differ on `pss`.

```
fit1 <- aov(pss ~ factor(km.res$cluster), data=dailyscale.clus)
summary(fit1)
```

```
##                          Df Sum Sq Mean Sq F value Pr(>F)
## factor(km.res$cluster)    3    247   82.35   100.2 <2e-16 ***
## Residuals              1372   1128    0.82
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
TukeyHSD(fit1)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = pss ~ factor(km.res$cluster), data = dailyscale.clus)
##
## $`factor(km.res$cluster)`
##             diff        lwr        upr     p adj
## 2-1 -0.937568465 -1.1029011 -0.7722358 0.0000000
## 3-1 -0.007666659 -0.1785830  0.1632497 0.9994508
## 4-1 -0.493702753 -0.7060229 -0.2813826 0.0000000
## 3-2  0.929901806  0.7681609  1.0916427 0.0000000
## 4-2  0.443865711  0.2388594  0.6488720 0.0000002
## 4-3 -0.486036094 -0.6955715 -0.2765007 0.0000000
```

We see that clusters differ from each other on pss, except clusters 3 and 4 (or 3 and 2 if there was label switching).

Differences on **non-clustering** variables provide evidence that, indeed, the cluster solution is providing a meaningful distinction. The typology has value.

In sum, there are variety of ways to justify a cluster solution (e.g., selection of *k*)

1. Conceptual arguments
2. Internal statistical criteria
3. replication of clusters in random halves
4. cluster differentiation on external variables

But, there is an interesting philosophical predicament, as highlighted by Morack, Ram, Fauth, & Gerstorf (2013) ...
"The relative cost/utility of variable-oriented and subgroup-oriented approaches must be considered. Although both types of analysis can be done in the flash of an eye, the time-consuming and effortful nature of multivariate long-term panel data must also be considered. Certainly, a multivariate analysis makes better use of the data than a univariate analysis, but we ourselves still struggle with the added utility of a purely subgroup-oriented approach. Case in point, we interpreted and named the groups from a variable-oriented perspective, and following standard good practices, we evaluated the utility of the profile groups by assessing how the categorical grouping variable was related to a variety of antecedents, correlates, and outcomes in a purely variable-oriented manner. In sum, the field still struggles to take a subgroup-oriented approach without placing it within variable-oriented interpretations and analysis.

A practical benefit of subgroup-oriented interpretation emerges when considering potential interventions. Multivariate profiles may point toward tailoring diagnostic and intervention efforts to individual needs."

Pros and Cons of Clustering

Pros
Structures and summarizes heterogeneity in a given data set
Identifies higher-order, non-linear, or group-specific interactions] Groups individuals in multivariate space using communalities and differences in – level (profile elevation) – dispersion (degree of variability around profile average) – shape (pattern of high and low scores in a profile) Does not assume sample homogeneity, but examines the possibility of distinct sub-populations
Cons
Fuzziness of classification (index for goodness of individual – subgroup fit not available)
No internal statistical criterion available for "difinitive" evaluation of optimal number of clusters
Algorithm always generates clusters irrespective of the true existence of structure
Multicollinearity and outliers can distort cluster solutions

Slight switch here - into the sports car =:-)

Hierarchical Clustering

The `cluster` package provides a whole set of options … including both hierarchical and non-hierarchical methods.
Lets look at a hierarchical method - Good explanations are can be found here http://www.econ.upf.edu/~michael/stanford/maeb7.pdf .[3]

Prelim: we make distance matrix (not totally necessary, but we do here for conceptual value)

```
dist.all <- daisy(dailyscale[,c("lteq","posaff")],metric="euclidean",stand=FALSE)
#loking at distances among first 5 persons
as.matrix(dist.all)[1:5,1:5]
```

```
##           1          2         3         4         5
## 1 0.00000000 0.09088348 1.0906017 1.5479960 0.8493762
## 2 0.09088348 0.00000000 1.1814852 1.6387144 0.8920640
## 3 1.09060171 1.18148519 0.0000000 0.4644381 1.0573130
## 4 1.54799598 1.63871436 0.4644381 0.0000000 1.4634612
## 5 0.84937623 0.89206403 1.0573130 1.4634612 0.0000000
```

Note that `daisy()` does include some treatments for missing data. Be careful!

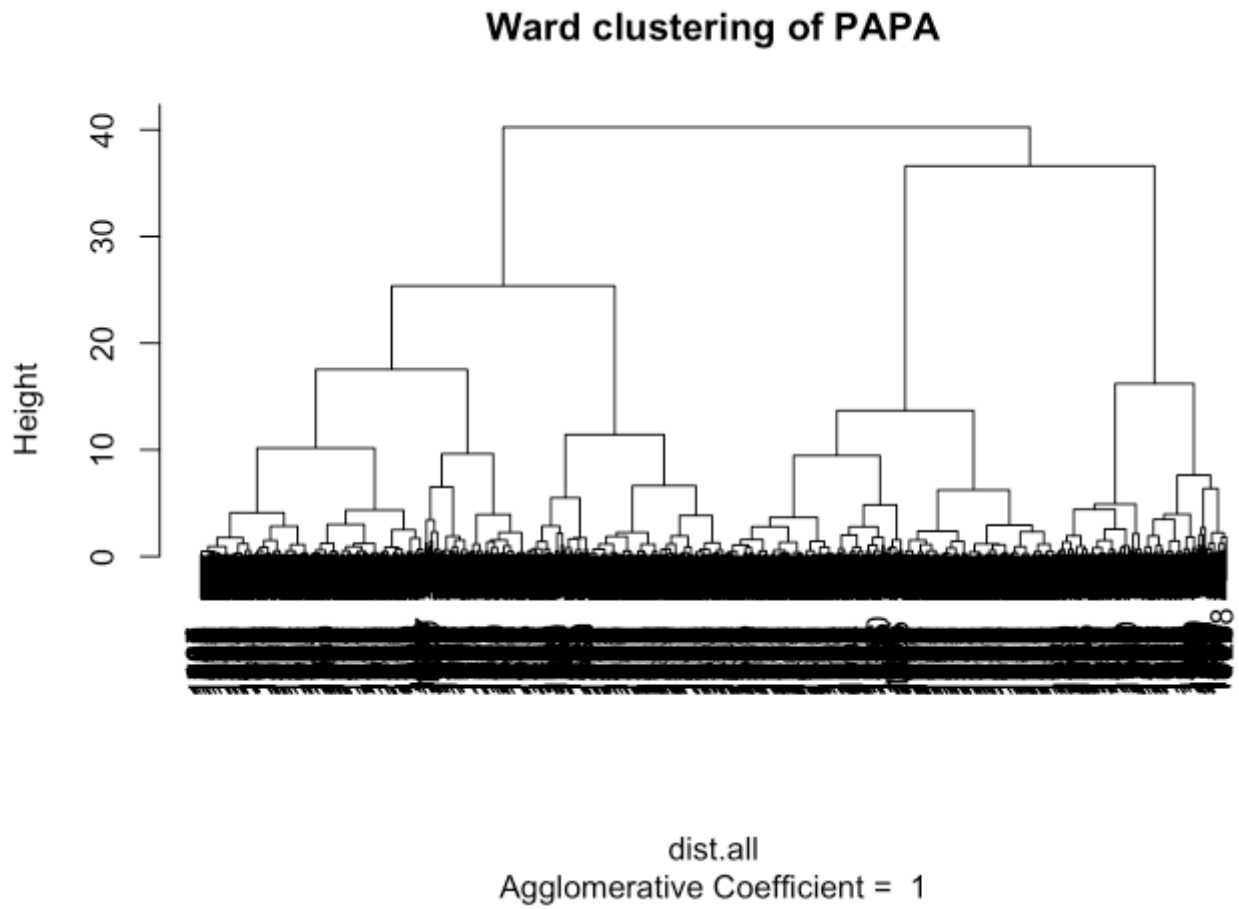
Engage the hierarchical clustering … we use the `agnes()` (agglomerative nesting, aka hierarchical clustering, Ward's method, …) function (which also allows for other linkage options)

```
# Compute Ward clusters for IGother
clusterward.papa <- agnes(dist.all, diss = TRUE, method = "ward")
```

There are many choices for the *linkage method*. We have chosen Ward here, as a classic. Again, this is a well-trodden research area, and one can find recommendations of all types. Read widely to find the best for your specific purpose and data.

Then we visualize it!

```
# Plot
layout(matrix(1))
plot(clusterward.papa, which.plot = 2, main = "Ward clustering of PAPA")
```



This is a **Dendrogram** (basically an organized plot of the distance matrix) that indicates how far apart objects are and when they might be merged together. The y-axis indicates the distance between the clusters. Long vertical lines indicate that there is a lot of between-cluster distance. We determine a level at which to "cut the tree". Generally we are looking for a level above which the lines are long (between-group heterogeneity) and below which the leaves are close (within-group homogeneity).

We see that 4 clusters seems to be a good tradeoff for parsimony.

Lets cut the tree and make cluster assignments.

```
wardcluster4 <- cutree(clusterward.papa, k = 4)
```

And look at some statistical criteria

```
cluster.stats(dist.all, clustering=wardcluster4,
              silhouette = TRUE, sepindex = TRUE)
```

```
## $n
## [1] 1376
##
## $cluster.number
## [1] 4
##
## $cluster.size
## [1] 437 443 229 267
##
## $min.cluster.size
## [1] 229
##
## $noisen
## [1] 0
##
## $diameter
## [1] 4.710322 3.033785 4.361416 2.844837
##
## $average.distance
## [1] 1.0705832 0.8725114 1.2911304 0.8755665
##
## $median.distance
## [1] 0.9638540 0.8580223 1.2044112 0.8146296
##
## $separation
## [1] 0.09088348 0.09088348 0.09088348 0.09088348
##
## $average.toother
## [1] 1.778935 1.969837 2.398617 2.164384
##
## $separation.matrix
##           [,1]       [,2]       [,3]       [,4]
## [1,] 0.00000000 0.09088348 0.09088348 0.09088348
## [2,] 0.09088348 0.00000000 0.09595597 0.09088348
## [3,] 0.09088348 0.09595597 0.00000000 1.65318945
## [4,] 0.09088348 0.09088348 1.65318945 0.00000000
##
## $ave.between.matrix
##          [,1]     [,2]     [,3]     [,4]
## [1,] 0.000000 1.724896 2.024155 1.658275
## [2,] 1.724896 0.000000 2.268755 2.114358
## [3,] 2.024155 2.268755 0.000000 3.226963
## [4,] 1.658275 2.114358 3.226963 0.000000
##
## $average.between
## [1] 2.036311
```

```
##
## $average.within
## [1] 1.005678
##
## $n.between
## [1] 691214
##
## $n.within
## [1] 254786
##
## $max.diameter
## [1] 4.710322
##
## $min.separation
## [1] 0.09088348
##
## $within.cluster.ss
## [1] 948.389
##
## $clus.avg.silwidths
##         1         2         3         4
## 0.1784037 0.4533402 0.2833088 0.4524960
##
## $avg.silwidth
## [1] 0.3375627
##
## $g2
## NULL
##
## $g3
## NULL
##
## $pearsongamma
## [1] 0.4834923
##
## $dunn
## [1] 0.01929453
##
## $dunn2
## [1] 1.284359
##
## $entropy
## [1] 1.345759
##
## $wb.ratio
## [1] 0.4938724
```

```
##
## $ch
## [1] 868.775
##
## $cwidegap
## [1] 0.7249194 0.6166206 0.9088348 0.4544174
##
## $widestgap
## [1] 0.9088348
##
## $sindex
## [1] 0.1292593
##
## $corrected.rand
## NULL
##
## $vi
## NULL
```

A two-step approach

Often times, researchers are using a two-step approach ...

1. Hierarchical Ward's method to ...
   ...evaluate optimal number of clusters
   ...produce starting seeds for subsequent step
2. Non-hierarchical k-means method to ...
   ...determine final case location in the separate subgroups

The two-step approach circumvents some drawbacks of each procedure
...Ward's method does not allow revising assigned membership in later steps tends to produce clusters of similar size ...k-means method produces optimal clusters only if starting seeds are pre-specified

K-medoids

An alternative hierarchical clustering method ... we use the `pam()` (partitioning around mediods) which is like k-means, but a bit more robust.

```
# Compute PAM clustering solution for k=4
clusterpam.papa <- pam(dist.all, k=4, diss = TRUE)
clusterpam.papa
```

```
## Medoids:
##      ID
## [1,] "129" "154"
## [2,] "108" "130"
```

```
## [3,] "676" "739"
## [4,] "233" "262"
## Clustering vector:
##   1   2   3   4   5   6   7   9  10  11  13  14  15  17  18  19
##   1   1   2   2   3   3   3   2   2   2   2   2   4   2   1   2
##  20  21  22  23  27  28  29  30  31  32  37  38  39  40  41  42
##   1   4   4   2   4   4   4   4   2   1   1   1   2   2   4   2
##  43  44  47  48  49  51  52  53  54  55  56  57  58  59  60  61
##   2   1   1   1   1   1   2   1   1   4   3   4   4   4   4   4
##  62  63  64  66  67  68  70  71  72  73  74  75  76  77  78  79
##   3   1   2   3   4   1   2   2   2   1   2   2   1   1   2   1
##  81  84  85  86  87  88  89  90  91  92  93  95  96  97  98  99
##   1   1   4   1   1   2   2   3   3   3   3   2   2   2   2   3
## 101 102 103 105 106 107 108 110 111 112 113 114 115 116 117 118
##   3   3   2   4   3   3   3   2   3   3   3   3   3   3   3   3
## 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134
##   3   4   1   4   4   4   4   4   1   2   2   2   2   1   2   2
## 135 136 137 138 140 141 143 144 145 146 147 149 150 151 152 153
##   2   3   3   2   3   1   3   3   1   3   3   3   3   3   4   3
## 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169
##   1   1   1   3   3   2   2   2   2   2   2   2   1   1   2   2
## 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185
##   1   1   1   4   1   4   4   4   4   1   1   1   2   2   4   1
## 186 187 188 189 190 191 192 193 194 196 197 198 199 201 202 203
##   1   3   3   3   1   3   3   4   4   2   1   1   1   1   3   3
## 204 205 206 208 209 210 211 212 213 214 215 216 217 218 219 220
##   2   2   3   3   3   1   2   2   1   1   1   1   2   4   4   4
## 221 222 223 224 225 226 228 229 230 231 232 233 234 235 236 237
##   4   3   4   4   4   2   2   2   4   4   1   3   1   1   1   4
## 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253
##   1   1   1   4   1   1   2   3   3   1   3   2   3   1   3   3
## 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269
##   4   3   3   3   2   1   2   2   4   1   1   2   1   1   1   1
## 270 271 272 273 274 275 276 277 278 279 280 281 282 283 285 286
##   3   1   1   1   1   2   1   1   1   1   1   1   4   1   4   4
## 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302
##   4   1   2   1   1   1   4   1   4   2   2   2   1   1   1   4
## 303 304 305 306 307 308 309 312 313 314 315 316 317 318 319 320
##   1   2   2   2   1   2   1   1   2   2   1   1   3   4   4   2
## 321 323 324 325 326 327 328 329 330 331 332 333 334 335 343 344
##   2   2   2   2   2   2   2   2   2   2   2   2   2   2   4   2
## 345 346 347 348 349 350 352 353 354 355 356 358 359 360 361 364
##   1   1   1   1   1   2   1   2   1   1   1   1   1   2   2   3
## 365 366 367 368 369 370 371 372 373 374 376 377 378 379 380 381
##   3   3   2   1   1   1   1   1   1   1   4   4   4   2   2   4
## 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397
```

```
##   4   1   4   1   3   1   1   1   1   1   3   3   4   4   3   3
##  398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413
##   3   3   3   1   4   3   3   1   3   4   4   1   1   4   4   4
##  414 415 416 417 418 419 422 423 424 425 426 427 428 429 430 431
##   4   2   1   2   2   2   1   2   3   1   3   3   3   3   3   2
##  432 433 434 435 436 437 438 439 441 442 443 444 445 446 447 448
##   2   2   1   1   1   1   1   3   2   1   1   4   3   3   2   2
##  449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464
##   2   2   2   1   2   2   4   4   4   4   4   4   4   4   1   2
##  465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480
##   1   1   1   1   1   4   2   1   3   3   1   1   1   4   4   4
##  481 483 485 486 487 488 489 490 491 492 493 494 495 496 497 498
##   2   4   2   1   2   4   4   1   4   4   4   2   1   2   1   1
##  499 500 501 502 503 504 505 506 507 508 509 510 511 512 513 514
##   4   4   1   3   1   4   4   4   4   1   2   2   2   2   2   2
##  515 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530
##   4   1   2   3   2   2   3   2   3   3   4   1   1   2   3   1
##  531 533 534 535 536 537 538 539 540 541 542 543 544 545 546 548
##   3   4   3   3   3   4   4   4   4   4   4   4   2   3   2   3
##  549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564
##   3   2   2   1   3   3   3   3   4   3   3   2   3   4   1   2
##  565 566 567 568 569 570 571 572 573 574 575 576 577 578 579 580
##   3   4   1   2   2   2   2   2   2   2   2   4   3   3   3   3
##  581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596
##   4   4   4   1   2   1   1   1   1   1   1   2   1   4   4   1
##  597 598 599 601 602 603 604 605 606 607 609 610 611 612 613 614
##   1   4   4   1   1   4   1   4   4   4   2   1   1   2   1   4
##  615 616 617 618 619 620 621 622 623 624 625 626 628 629 630 631
##   1   2   1   1   2   2   1   4   1   2   2   1   2   2   1   1
##  632 633 634 635 636 637 638 639 640 641 642 643 644 645 646 647
##   4   1   4   4   1   2   4   4   1   3   1   4   4   4   4   4
##  648 649 650 651 652 653 654 655 656 657 658 659 660 661 662 663
##   4   1   1   4   4   4   1   3   3   4   4   4   4   4   4   4
##  664 665 666 667 668 669 670 671 672 673 674 675 676 677 678 679
##   3   3   3   3   3   3   3   3   3   3   3   2   1   4   3   2
##  680 681 682 684 685 686 687 688 689 690 691 692 693 694 696 697
##   2   2   1   4   1   2   2   3   3   2   2   2   3   3   2   4
##  698 699 700 701 702 703 704 705 706 707 708 709 710 711 712 713
##   4   2   4   4   4   3   3   3   3   3   1   4   3   4   1   4
##  714 715 716 717 719 720 721 722 723 724 725 726 727 728 731 735
##   1   1   1   1   4   1   1   4   4   4   4   4   3   3   3   1
##  736 737 738 739 740 741 742 743 744 745 746 747 748 749 750 751
##   3   1   2   3   4   4   1   3   3   3   2   3   1   3   2   2
##  752 753 754 755 757 758 759 760 761 762 763 764 765 766 767 768
##   1   4   3   1   4   1   1   4   4   1   2   1   2   2   1   1
##  769 770 771 772 773 774 775 776 777 778 779 780 781 782 783 784
```

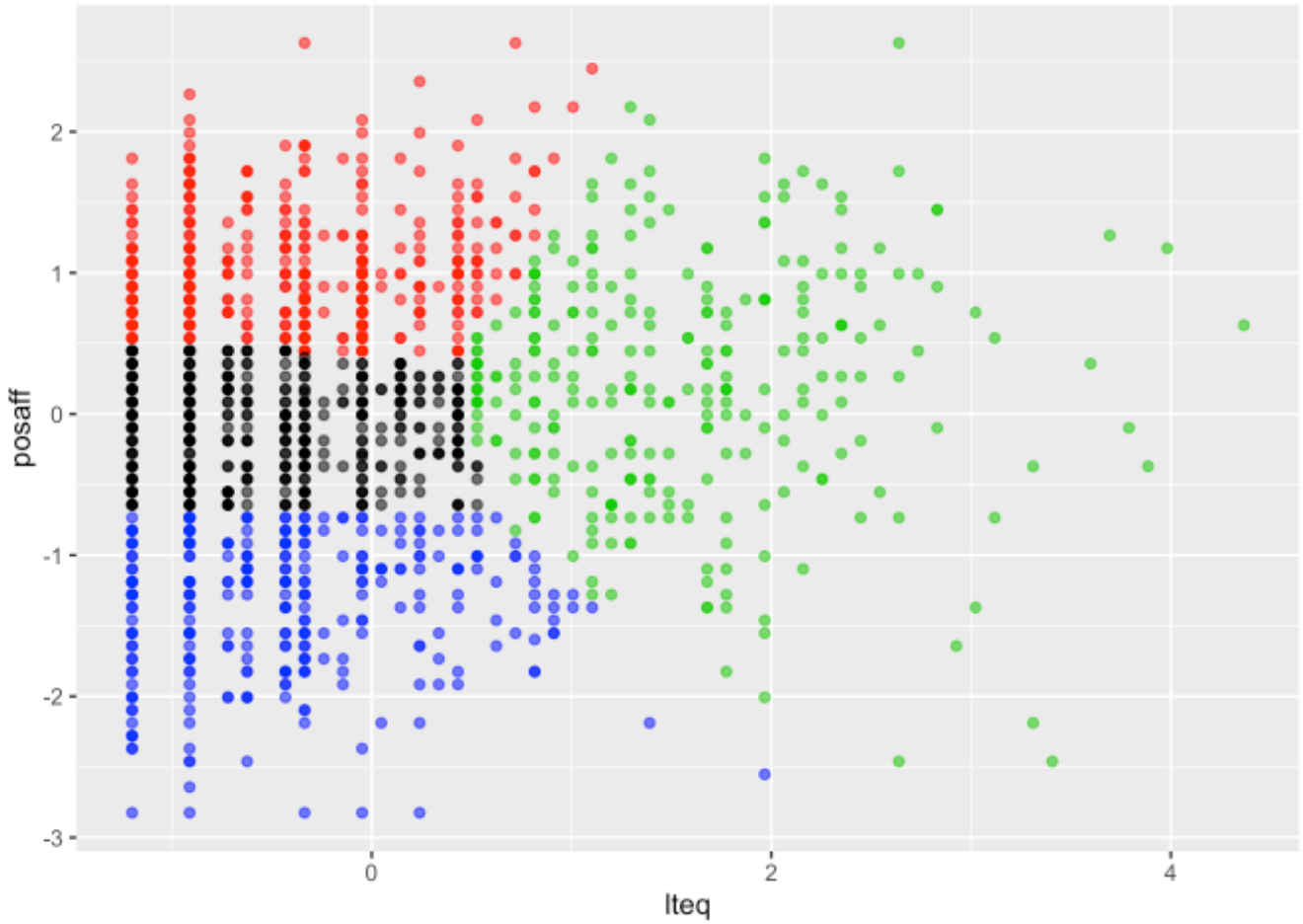
```
##    1    1    2    1    1    1    3    1    3    4    4    4    4    4    4    4
##  785  786  787  788  789  790  791  792  793  794  795  796  797  798  799  800
##    4    1    4    4    1    1    4    4    3    3    3    4    3    2    3    3
##  801  802  803  804  805  806  807  808  809  810  811  812  813  814  815  816
##    3    3    3    4    3    4    2    2    2    2    1    1    1    4    4    4
##  817  819  820  821  822  823  824  825  826  827  828  829  830  831  832  833
##    1    4    4    1    4    2    1    1    4    2    1    3    4    4    2    2
##  834  835  836  837  838  839  840  841  842  843  844  845  846  847  848  849
##    4    3    1    2    1    2    1    1    1    2    3    4    1    1    3    1
##  850  851  852  853  854  855  856  857  858  859  860  861  862  863  864  866
##    1    4    1    1    3    2    2    2    3    2    3    2    3    3    3    3
##  867  868  869  871  872  873  874  875  876  877  878  879  880  881  882  883
##    3    3    3    3    3    3    3    4    1    3    1    4    1    1    1    1
##  884  885  886  887  888  889  890  891  892  893  894  895  896  897  898  899
##    4    2    1    4    4    4    1    4    1    1    1    1    1    1    1    1
##  900  901  902  903  904  905  906  907  908  909  910  911  912  913  914  915
##    1    1    1    1    2    1    1    1    1    3    4    1    4    4    3    3
##  916  917  918  919  920  921  922  923  924  925  926  927  928  929  930  931
##    4    3    2    1    2    1    1    4    1    4    2    2    1    1    1    1
##  932  933  934  935  936  937  938  940  942  943  944  945  946  947  948  949
##    1    2    3    3    3    3    3    2    2    2    1    2    4    4    1    1
##  950  951  952  953  956  957  958  959  960  961  962  963  964  965  966  967
##    3    1    1    1    4    1    2    1    2    1    1    1    1    1    3    3
##  968  969  970  971  972  973  974  976  977  978  979  980  982  983  984  985
##    2    4    4    3    3    3    4    2    1    3    4    4    2    2    2    1
##  986  987  988  989  990  991  992  993  994  995  996  997  998  999 1000 1001
##    4    3    4    1    1    3    1    2    3    1    3    1    4    2    3    4
## 1002 1003 1004 1005 1006 1007 1008 1009 1011 1012 1013 1014 1015 1016 1017 1018
##    3    3    1    3    1    1    1    1    4    4    1    4    1    3    3    3
## 1019 1020 1021 1022 1023 1024 1025 1026 1027 1028 1029 1030 1031 1032 1033 1034
##    2    2    2    4    3    4    1    3    3    4    1    1    1    4    4    1
## 1035 1036 1037 1038 1039 1040 1041 1042 1043 1044 1045 1046 1047 1048 1049 1050
##    4    4    4    2    2    2    2    2    2    2    2    2    2    2    2    2
## 1051 1052 1053 1054 1055 1056 1057 1058 1059 1060 1061 1062 1064 1065 1066 1067
##    3    2    3    2    2    1    1    1    1    1    1    1    3    4    1    2
## 1068 1069 1070 1071 1072 1073 1074 1075 1076 1077 1078 1079 1080 1081 1082 1083
##    2    1    2    3    2    3    4    3    3    2    2    2    2    1    1    1
## 1084 1085 1086 1087 1088 1089 1090 1091 1092 1093 1094 1095 1096 1097 1098 1099
##    1    2    2    2    2    2    2    1    2    2    2    2    2    2    2    2
## 1100 1101 1102 1103 1104 1105 1106 1107 1108 1109 1110 1111 1112 1113 1114 1115
##    2    2    4    2    1    4    1    1    1    2    3    2    2    2    2    2
## 1116 1117 1118 1119 1120 1121 1122 1123 1124 1125 1126 1127 1128 1129 1130 1131
##    2    2    1    1    3    3    1    3    2    1    1    1    2    2    3    3
## 1132 1133 1134 1135 1136 1137 1138 1139 1140 1141 1142 1143 1144 1145 1146 1147
##    3    2    2    3    2    3    3    3    3    3    1    2    2    1    1    2
## 1148 1149 1150 1151 1152 1153 1154 1156 1157 1158 1160 1161 1162 1163 1164 1165
```

```
##    2    2    2    2    4    1    2    4    2    4    2    1    1    1    4    4
## 1166 1167 1168 1169 1170 1171 1172 1173 1174 1175 1176 1177 1178 1179 1180 1181
##    2    3    3    4    4    1    3    2    3    3    4    1    1    3    3    2
## 1182 1183 1184 1185 1186 1187 1188 1189 1190 1191 1192 1193 1194 1195 1196 1197
##    4    2    2    4    1    1    1    4    1    3    3    1    2    3    3    3
## 1198 1199 1200 1201 1202 1203 1204 1205 1207 1208 1209 1210 1211 1212 1213 1215
##    1    2    2    2    1    2    4    1    2    4    2    1    2    2    2    1
## 1216 1217 1218 1219 1220 1221 1222 1223 1224 1225 1226 1227 1228 1229 1230 1231
##    1    1    2    2    1    2    4    1    1    2    2    1    1    4    3    3
## 1232 1233 1234 1235 1236 1237 1238 1239 1240 1241 1242 1243 1244 1245 1246 1247
##    3    3    3    3    4    4    2    2    1    2    2    1    4    4    1    1
## 1248 1249 1250 1251 1252 1253 1254 1255 1256 1257 1258 1259 1260 1261 1262 1263
##    2    3    4    2    3    1    3    2    2    3    3    3    3    3    2    1
## 1264 1265 1266 1267 1268 1269 1270 1271 1272 1273 1274 1275 1276 1277 1278 1279
##    2    2    1    1    4    1    2    1    3    3    4    1    2    3    2    2
## 1280 1281 1282 1283 1284 1285 1286 1287 1288 1289 1290 1291 1292 1293 1294 1295
##    1    2    1    4    1    1    1    1    4    4    1    1    2    2    2    2
## 1296 1297 1298 1299 1300 1301 1302 1304 1305 1306 1307 1308 1309 1310 1311 1312
##    4    4    4    2    1    2    2    1    4    4    4    2    1    1    1    2
## 1313 1314 1315 1316 1317 1318 1319 1320 1321 1322 1323 1324 1325 1326 1327 1328
##    1    1    2    1    4    2    1    2    1    4    1    2    1    4    1    4
## 1329 1330 1331 1332 1333 1334 1335 1336 1337 1338 1339 1340 1341 1342 1343 1344
##    4    4    1    3    4    3    2    1    3    3    4    2    2    2    2    2
## 1345 1346 1347 1348 1349 1350 1351 1352 1353 1354 1355 1356 1357 1358 1359 1360
##    1    1    2    2    2    2    2    2    2    2    1    2    2    2    2    2
## 1361 1362 1363 1364 1365 1366 1367 1368 1369 1370 1371 1372 1373 1374 1375 1376
##    1    2    2    2    2    2    2    2    4    2    2    3    3    1    4    3
## 1377 1378 1379 1380 1381 1382 1383 1384 1385 1386 1387 1388 1389 1390 1391 1392
##    3    3    3    3    1    2    2    3    2    3    2    1    4    4    4    4
## 1393 1394 1395 1396 1397 1398 1400 1401 1402 1403 1404 1405 1406 1407 1408 1409
##    4    4    4    3    4    3    2    3    4    3    4    1    1    2    4    1
## 1410 1411 1412 1413 1414 1415 1416 1417 1418 1419 1420 1421 1422 1423 1424 1425
##    1    1    1    2    2    4    1    1    4    1    1    2    1    4    4    4
## 1426 1427 1429 1430 1431 1432 1433 1434 1435 1436 1437 1438 1439 1440 1441 1442
##    4    1    3    3    3    3    4    3    3    1    1    4    4    4    1    4
## 1443 1444 1445 1446 1447 1448 1449 1450 1451 1452 1453 1454 1455 1456 1457 1458
##    1    1    4    4    4    4    4    1    3    1    2    2    4    4    1    1
## Objective function:
##     build      swap
## 0.7176465 0.7049830
##
## Available components:
## [1] "medoids"    "id.med"     "clustering" "objective"  "isolation"
## [6] "clusinfo"   "silinfo"    "diss"       "call"
```

```
#Checking length
pamcluster <- clusterpam.papa$clustering
length(pamcluster)
```

```
## [1] 1376
```

```
#binding to originaldata
dailyscale.pam <- cbind(dailyscale,pamcluster)
#plotting clustered data points
ggplot(dailyscale.pam,aes(x=lteq,y=posaff)) +
  geom_point(alpha=.6, color=factor(pamcluster))
```



Lets run the autosearch and see what comes out ...

```
pamauto <- pamk(dist.all,krange=2:10,criterion="asw", usepam=TRUE,
                scaling=FALSE, alpha=0.001, diss=TRUE,
                critout=FALSE, ns=10, seed=NULL)
pamauto
```

```
## $pamobject
## Medoids:
##      ID
## [1,] "598" "653"
## [2,] "63"  "78"
## [3,] "738" "802"
## Clustering vector:
##    1    2    3    4    5    6    7    9   10   11   13   14   15   17   18   19
##    1    1    2    2    3    3    3    2    2    2    2    2    1    2    2    2
##   20   21   22   23   27   28   29   30   31   32   37   38   39   40   41   42
##    1    1    1    2    1    1    1    1    2    2    2    2    2    2    1    2
##   43   44   47   48   49   51   52   53   54   55   56   57   58   59   60   61
##    2    1    2    2    2    2    2    2    2    1    3    1    1    1    1    1
##   62   63   64   66   67   68   70   71   72   73   74   75   76   77   78   79
##    3    2    3    3    1    3    2    2    2    2    2    2    2    1    2    3
##   81   84   85   86   87   88   89   90   91   92   93   95   96   97   98   99
##    2    1    1    1    2    2    2    3    3    3    3    2    2    2    2    3
##  101  102  103  105  106  107  108  110  111  112  113  114  115  116  117  118
##    3    3    2    1    3    3    3    2    3    3    3    3    3    3    3    3
##  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134
##    3    1    1    1    1    1    1    1    1    2    2    2    2    2    2    2
##  135  136  137  138  140  141  143  144  145  146  147  149  150  151  152  153
##    2    3    3    2    3    3    3    3    3    3    3    3    3    3    1    3
##  154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169
##    2    2    2    3    3    2    2    2    2    2    2    2    2    1    2    2
##  170  171  172  173  174  175  176  177  178  179  180  181  182  183  184  185
##    1    2    3    1    1    1    1    1    1    1    1    2    3    3    1    3
##  186  187  188  189  190  191  192  193  194  196  197  198  199  201  202  203
##    2    3    3    3    1    3    3    1    1    2    2    2    2    3    3    3
##  204  205  206  208  209  210  211  212  213  214  215  216  217  218  219  220
##    2    2    3    3    3    2    2    2    2    2    2    1    2    1    1    1
##  221  222  223  224  225  226  228  229  230  231  232  233  234  235  236  237
##    1    3    1    1    3    2    2    2    1    1    2    3    2    2    1    1
##  238  239  240  241  242  243  244  245  246  247  248  249  250  251  252  253
##    2    1    1    1    3    2    2    3    3    2    3    2    3    1    3    3
##  254  255  256  257  258  259  260  261  262  263  264  265  266  267  268  269
##    1    3    3    3    2    2    2    2    1    2    2    2    2    1    2    1
##  270  271  272  273  274  275  276  277  278  279  280  281  282  283  285  286
##    3    2    2    2    2    2    1    1    1    2    1    1    1    1    1    1
##  287  288  289  290  291  292  293  294  295  296  297  298  299  300  301  302
##    1    1    2    1    1    1    1    1    1    2    2    2    1    1    2    1
##  303  304  305  306  307  308  309  312  313  314  315  316  317  318  319  320
##    3    2    2    2    2    2    1    2    2    2    2    3    3    1    1    2
##  321  323  324  325  326  327  328  329  330  331  332  333  334  335  343  344
##    2    2    2    2    2    2    2    3    2    2    2    2    2    2    1    2
##  345  346  347  348  349  350  352  353  354  355  356  358  359  360  361  364
```

```
##   1   2   1   2   2   2   2   2   2   1   1   1   1   2   3   3
##  365 366 367 368 369 370 371 372 373 374 376 377 378 379 380 381
##   3   3   2   2   2   2   3   2   2   3   1   1   1   2   2   1
##  382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397
##   1   2   3   2   3   2   2   1   1   3   3   3   1   1   3   3
##  398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413
##   3   3   3   1   1   3   3   1   3   1   1   1   2   1   1   1
##  414 415 416 417 418 419 422 423 424 425 426 427 428 429 430 431
##   1   2   2   2   2   2   1   2   3   3   3   3   3   3   3   2
##  432 433 434 435 436 437 438 439 441 442 443 444 445 446 447 448
##   2   2   2   2   3   1   1   3   2   2   3   1   3   3   2   2
##  449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464
##   2   2   2   2   2   2   1   1   1   1   3   1   1   1   2   2
##  465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480
##   1   1   2   2   2   1   2   2   3   3   2   1   1   1   1   1
##  481 483 485 486 487 488 489 490 491 492 493 494 495 496 497 498
##   2   1   2   2   2   1   1   1   1   1   1   2   1   2   1   1
##  499 500 501 502 503 504 505 506 507 508 509 510 511 512 513 514
##   1   1   2   3   1   1   1   1   1   2   2   2   2   2   2   2
##  515 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530
##   1   1   3   3   2   2   3   2   3   3   1   1   1   2   3   3
##  531 533 534 535 536 537 538 539 540 541 542 543 544 545 546 548
##   3   1   3   3   3   1   1   1   1   1   1   1   2   3   2   3
##  549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564
##   3   3   3   2   3   3   3   3   1   3   3   2   3   1   1   2
##  565 566 567 568 569 570 571 572 573 574 575 576 577 578 579 580
##   3   1   1   2   2   2   2   2   2   2   2   1   3   3   3   3
##  581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596
##   1   1   1   1   2   2   1   1   2   1   1   2   1   1   1   2
##  597 598 599 601 602 603 604 605 606 607 609 610 611 612 613 614
##   1   1   1   2   1   1   2   1   1   1   2   2   2   2   1   1
##  615 616 617 618 619 620 621 622 623 624 625 626 628 629 630 631
##   2   2   2   2   2   2   1   1   1   2   2   2   2   2   2   2
##  632 633 634 635 636 637 638 639 640 641 642 643 644 645 646 647
##   1   1   1   1   1   2   1   1   2   3   2   1   1   1   1   1
##  648 649 650 651 652 653 654 655 656 657 658 659 660 661 662 663
##   1   2   1   1   1   1   3   3   3   1   1   1   1   1   1   1
##  664 665 666 667 668 669 670 671 672 673 674 675 676 677 678 679
##   3   3   3   3   3   3   3   3   3   3   3   2   2   1   3   2
##  680 681 682 684 685 686 687 688 689 690 691 692 693 694 696 697
##   2   2   2   1   1   2   2   3   3   2   2   2   3   3   2   1
##  698 699 700 701 702 703 704 705 706 707 708 709 710 711 712 713
##   1   2   1   1   1   3   3   3   3   3   1   1   3   1   1   1
##  714 715 716 717 719 720 721 722 723 724 725 726 727 728 731 735
##   2   1   1   1   1   1   3   1   1   1   1   1   3   3   3   2
##  736 737 738 739 740 741 742 743 744 745 746 747 748 749 750 751
```

```
##    3    2    3    3    1    1    3    3    3    3    2    3    1    3    2    2
##  752  753  754  755  757  758  759  760  761  762  763  764  765  766  767  768
##    1    1    3    2    1    2    1    1    1    2    2    1    2    2    2    2
##  769  770  771  772  773  774  775  776  777  778  779  780  781  782  783  784
##    2    1    2    2    1    1    3    1    3    1    1    1    1    1    1    1
##  785  786  787  788  789  790  791  792  793  794  795  796  797  798  799  800
##    1    1    1    1    2    2    1    1    3    3    3    1    3    2    3    3
##  801  802  803  804  805  806  807  808  809  810  811  812  813  814  815  816
##    3    3    3    1    3    1    2    2    2    2    2    2    2    1    1    1
##  817  819  820  821  822  823  824  825  826  827  828  829  830  831  832  833
##    2    1    1    1    1    2    2    2    1    2    1    3    1    1    2    2
##  834  835  836  837  838  839  840  841  842  843  844  845  846  847  848  849
##    1    3    1    3    2    2    1    2    2    2    3    1    1    1    3    2
##  850  851  852  853  854  855  856  857  858  859  860  861  862  863  864  866
##    2    1    2    1    3    2    2    3    3    2    3    2    3    3    3    3
##  867  868  869  871  872  873  874  875  876  877  878  879  880  881  882  883
##    3    3    3    3    3    3    3    1    1    3    1    1    2    1    2    1
##  884  885  886  887  888  889  890  891  892  893  894  895  896  897  898  899
##    1    2    2    1    1    1    2    1    2    2    2    3    2    2    1    2
##  900  901  902  903  904  905  906  907  908  909  910  911  912  913  914  915
##    2    2    2    2    2    1    1    1    2    3    1    3    1    1    3    3
##  916  917  918  919  920  921  922  923  924  925  926  927  928  929  930  931
##    1    3    2    3    3    3    1    1    2    1    2    2    1    2    2    2
##  932  933  934  935  936  937  938  940  942  943  944  945  946  947  948  949
##    1    2    3    3    3    3    3    2    2    2    2    2    1    1    1    2
##  950  951  952  953  956  957  958  959  960  961  962  963  964  965  966  967
##    3    3    2    1    1    1    2    2    2    1    1    1    1    2    3    3
##  968  969  970  971  972  973  974  976  977  978  979  980  982  983  984  985
##    2    1    1    3    3    3    1    2    2    3    1    1    2    2    2    1
##  986  987  988  989  990  991  992  993  994  995  996  997  998  999 1000 1001
##    1    3    1    1    3    3    2    2    3    3    3    1    1    3    3    1
## 1002 1003 1004 1005 1006 1007 1008 1009 1011 1012 1013 1014 1015 1016 1017 1018
##    3    3    3    3    1    2    2    2    1    1    2    1    3    3    3    3
## 1019 1020 1021 1022 1023 1024 1025 1026 1027 1028 1029 1030 1031 1032 1033 1034
##    2    2    2    1    3    1    3    3    3    1    2    1    1    1    1    1
## 1035 1036 1037 1038 1039 1040 1041 1042 1043 1044 1045 1046 1047 1048 1049 1050
##    1    1    1    2    2    2    2    2    2    2    2    3    2    2    2    2
## 1051 1052 1053 1054 1055 1056 1057 1058 1059 1060 1061 1062 1064 1065 1066 1067
##    3    2    3    2    2    2    2    1    1    1    1    3    3    1    3    3
## 1068 1069 1070 1071 1072 1073 1074 1075 1076 1077 1078 1079 1080 1081 1082 1083
##    2    3    2    3    2    3    1    3    3    3    2    2    2    1    2    1
## 1084 1085 1086 1087 1088 1089 1090 1091 1092 1093 1094 1095 1096 1097 1098 1099
##    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2
## 1100 1101 1102 1103 1104 1105 1106 1107 1108 1109 1110 1111 1112 1113 1114 1115
##    2    2    1    2    1    1    2    1    2    2    3    2    3    2    2    2
## 1116 1117 1118 1119 1120 1121 1122 1123 1124 1125 1126 1127 1128 1129 1130 1131
```

```
##    2    2    2    3    3    3    1    3    2    1    2    2    3    2    3    3
## 1132 1133 1134 1135 1136 1137 1138 1139 1140 1141 1142 1143 1144 1145 1146 1147
##    3    2    2    3    2    3    3    3    3    3    2    2    2    3    2    2
## 1148 1149 1150 1151 1152 1153 1154 1156 1157 1158 1160 1161 1162 1163 1164 1165
##    2    2    2    2    1    2    2    1    2    1    2    2    2    1    1    1
## 1166 1167 1168 1169 1170 1171 1172 1173 1174 1175 1176 1177 1178 1179 1180 1181
##    2    3    3    1    1    1    3    2    3    3    1    2    1    3    3    3
## 1182 1183 1184 1185 1186 1187 1188 1189 1190 1191 1192 1193 1194 1195 1196 1197
##    1    2    2    1    2    3    3    1    1    3    3    1    3    3    3    3
## 1198 1199 1200 1201 1202 1203 1204 1205 1207 1208 1209 1210 1211 1212 1213 1215
##    2    2    2    2    2    2    1    1    2    1    2    2    2    2    2    2
## 1216 1217 1218 1219 1220 1221 1222 1223 1224 1225 1226 1227 1228 1229 1230 1231
##    2    1    2    2    2    2    1    2    1    2    2    1    2    1    3    3
## 1232 1233 1234 1235 1236 1237 1238 1239 1240 1241 1242 1243 1244 1245 1246 1247
##    3    3    3    3    1    1    2    2    1    2    2    2    1    1    1    2
## 1248 1249 1250 1251 1252 1253 1254 1255 1256 1257 1258 1259 1260 1261 1262 1263
##    2    3    1    2    3    2    3    3    2    3    3    3    3    3    2    2
## 1264 1265 1266 1267 1268 1269 1270 1271 1272 1273 1274 1275 1276 1277 1278 1279
##    2    2    3    2    1    2    2    3    3    3    1    2    3    3    2    2
## 1280 1281 1282 1283 1284 1285 1286 1287 1288 1289 1290 1291 1292 1293 1294 1295
##    2    2    1    1    1    1    2    1    1    1    2    2    2    2    2    2
## 1296 1297 1298 1299 1300 1301 1302 1304 1305 1306 1307 1308 1309 1310 1311 1312
##    1    1    1    2    2    2    2    1    1    1    1    2    2    2    2    2
## 1313 1314 1315 1316 1317 1318 1319 1320 1321 1322 1323 1324 1325 1326 1327 1328
##    2    1    2    2    1    2    2    2    2    1    2    2    2    1    2    1
## 1329 1330 1331 1332 1333 1334 1335 1336 1337 1338 1339 1340 1341 1342 1343 1344
##    1    1    1    3    1    3    2    3    3    3    3    2    2    2    2    2
## 1345 1346 1347 1348 1349 1350 1351 1352 1353 1354 1355 1356 1357 1358 1359 1360
##    1    2    2    2    2    2    2    2    2    2    2    2    2    2    2    2
## 1361 1362 1363 1364 1365 1366 1367 1368 1369 1370 1371 1372 1373 1374 1375 1376
##    2    2    2    2    2    2    2    2    1    2    2    3    3    1    1    3
## 1377 1378 1379 1380 1381 1382 1383 1384 1385 1386 1387 1388 1389 1390 1391 1392
##    3    3    3    3    1    2    2    3    3    3    3    1    1    1    1    1
## 1393 1394 1395 1396 1397 1398 1400 1401 1402 1403 1404 1405 1406 1407 1408 1409
##    1    1    1    3    1    3    2    3    1    3    1    1    2    2    1    2
## 1410 1411 1412 1413 1414 1415 1416 1417 1418 1419 1420 1421 1422 1423 1424 1425
##    3    2    1    2    2    1    1    1    1    2    1    2    1    1    1    1
## 1426 1427 1429 1430 1431 1432 1433 1434 1435 1436 1437 1438 1439 1440 1441 1442
##    1    2    3    3    3    3    1    3    3    2    1    1    1    1    1    1
## 1443 1444 1445 1446 1447 1448 1449 1450 1451 1452 1453 1454 1455 1456 1457 1458
##    2    1    1    1    1    1    1    1    3    1    2    2    1    1    2    2
## Objective function:
##     build      swap
## 0.8498193 0.7869947
##
## Available components:
```

```
## [1] "medoids"    "id.med"     "clustering" "objective"  "isolation"
## [6] "clusinfo"   "silinfo"    "diss"       "call"
##
## $nc
## [1] 3
##
## $crit
##  [1] 0.0000000 0.3345128 0.3808428 0.3165995 0.3264280 0.3362667 0.3489111
##  [8] 0.3348566 0.3340303 0.3330956
```

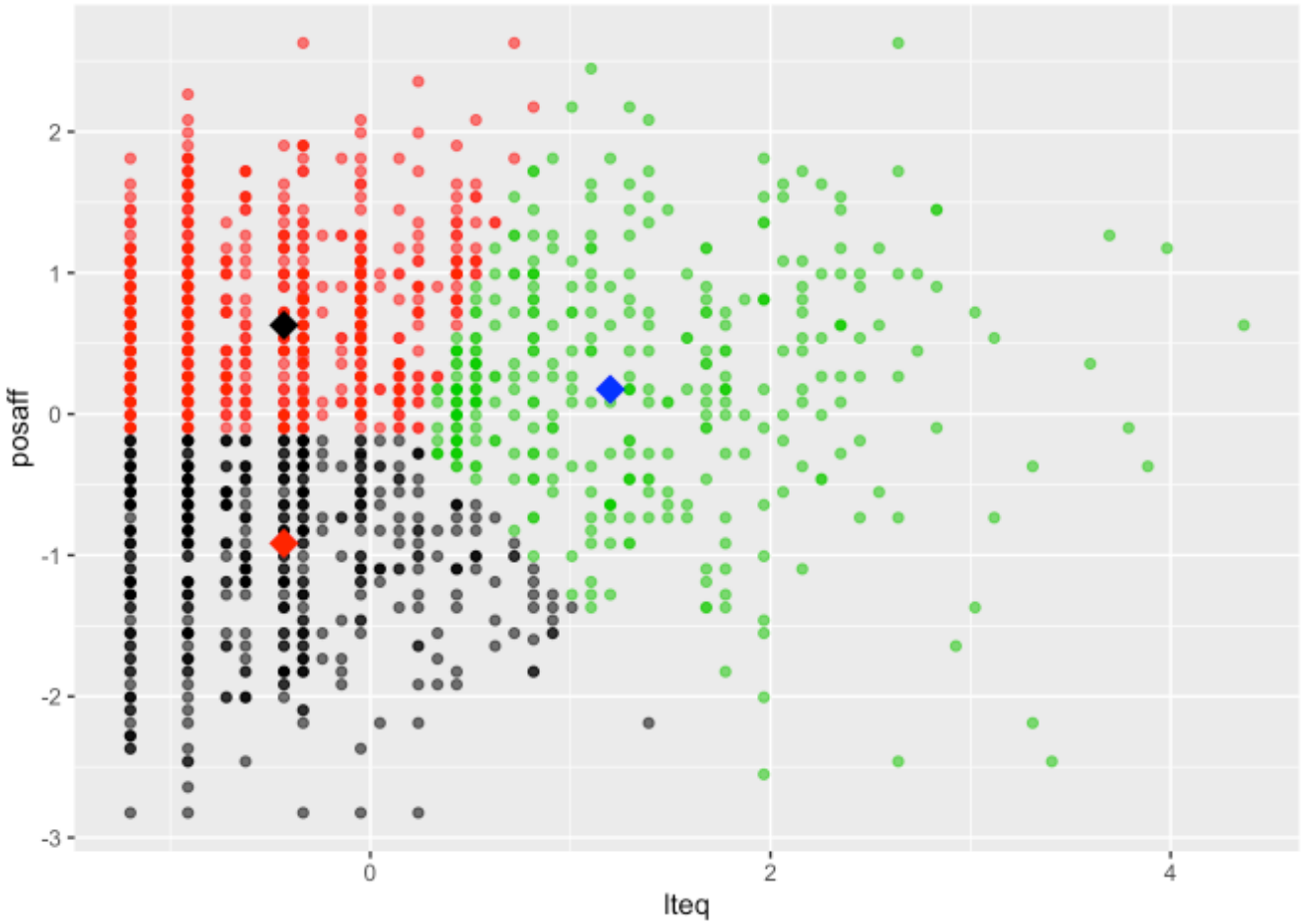
Here, the suggestion is for $k = 3$. Only three clusters.

```
#Obtain medoids
pamauto$pamobject$id.med
```

```
## [1] 598  63 738
```

```
#binding new cluster assignment to originaldata
dailyscale.pam$pamnew <- pamauto$pamobject$clustering

#plotting clustered data points with the medoids
ggplot(dailyscale.pam,aes(x=lteq,y=posaff)) +
  geom_point(alpha=.6, color=factor(dailyscale.pam$pamnew)) +
  geom_point(data=dailyscale.pam[598,],aes(x=lteq,y=posaff),color=2,size=5,shape=18)
+
  geom_point(data=dailyscale.pam[63,],aes(x=lteq,y=posaff),color=1,size=5,shape=18)
+
  geom_point(data=dailyscale.pam[738,],aes(x=lteq,y=posaff),color=4,size=5,shape=18)
```



Now we have a zone to play in!

Final Thoughts

Please remember the usual caution. Our intention here has been simple exposure. When using these methods for a paper or project, do the research necessary to engage the method precisely and with good form.
Thank you for clustering!

Code

The following is the code based on the Penn State tutorial[2-1] without all the commentary

```
#2 PRELIMINARIES
#general packages
library(ggplot2)
library(psych)

#cluster packages
library(cluster) #clustering
library(fpc) #flexible procedures for clustering
library(clusterCrit) #cluster criteria

#set filepath for data file
```

```
filepath <-
"https://quantdev.ssri.psu.edu/sites/qdev/files/AMIBbrief_raw_daily1.csv"

#read in the .csv file using the url() function
daily <- read.csv(file=url(filepath),header=TRUE)

#clean-up of variable names
var.names.daily <- tolower(colnames(daily))
colnames(daily)<-var.names.daily

#creating a new "id" variable
daily$id <- daily$id*10+daily$day

names(daily)

#reducing down to variable set
daily <- daily[ ,c("id","slphrs","weath","lteq","pss","se","swls","evalday",
"posaff","negaff","temp","hum","wind","bar","prec")]

#names of variables
names(daily)

#looking at data
head(daily,10)

#4.1 MISSING DATA
#removing observations with NA
dailysub <- daily[complete.cases(daily), ]
describe(dailysub)

#4.2 SCALING
#scaling all the variables
dailyscale <- data.frame(scale(dailysub, center=TRUE, scale=TRUE))
#checking and fixing the id variable (which we did not want standardized)
str(dailyscale$id)

dailyscale$id <- dailysub$id
str(dailyscale$id)

describe(dailyscale)

#4.3 PLOTING A SAMPLE OF BIVARIATE OBSERVATIONS
#choose a small subset of variables for easy visualization in a bivariate space
#use 'lteq' to measure physical activity and positive affect ("posaff")
ggplot(dailyscale,aes(x=lteq,y=posaff)) +
  geom_point()
```

```
#4.4 DISTANCES
#look at the first three as an example
data1 <- dailyscale[c(1,3,12),c("id","lteq","posaff")]
head(data1,3)

#make graph for visualization
labels.abc <-c("A","B","C")
ggplot(data1,aes(x=lteq,y=posaff)) +
  geom_polygon(fill="blue",alpha=.6) +
  geom_point(size=3) +
  geom_text(aes(x=lteq-.1,label=labels.abc)) +
  ylim(-1,1) + xlim(-1,1)

#euclidean distance of A B and C
dist.abc <- dist(data1[1:3,2:3],method="euclidean",diag=TRUE,upper=FALSE)
dist.abc

#Manhattan distance of A B and C
dist.abc2 <- dist(data1[1:3,2:3],method="manhattan",diag=TRUE,upper=FALSE)
dist.abc2

#5 K-MEANS CLUSTER
#there are random starts involved so we set a seed
set.seed(1234)
#running a cluster analysis
model <- kmeans(dailyscale[,c("lteq","posaff")], centers=4)
model

#extract the mean vectors and plot
#getting centers
model$centers
#plotting clustered data points with k means
ggplot(dailyscale,aes(x=lteq,y=posaff)) +
  geom_point(color=model$cluster, alpha=.6) +#plotting alll the points
  #plotting the centroids
  geom_point(aes(x=model$centers[1,1],y=model$centers[1,2]),color=1,size=5,shape=18)
+
  geom_point(aes(x=model$centers[2,1],y=model$centers[2,2]),color=2,size=5,shape=18)
+
  geom_point(aes(x=model$centers[3,1],y=model$centers[3,2]),color=3,size=5,shape=18)
+
  geom_point(aes(x=model$centers[4,1],y=model$centers[4,2]),color=4,size=5,shape=18)

#option to animate the cluster plotting
library(animation)
```

```
kmeans.ani(dailyscale[,c("lteq","posaff")], centers = 4, pch=c(15,16,17,18),
col=c(1,2,3,4))

#5.0.0.0.1 EVALUATING OF CLUSTERING QUALITY
#sum of squares within clusters
model$totss #total sum of squares
model$withinss #how close the observations are within each cluster
model$tot.withinss #sum of the ss from each cluster (smaller the better)
model$betweenss #how far the clusters are from each other (larger the better)

#plot the ss values to determine the best k value
#making a empty dataframe
criteria <- data.frame()
#setting range of k
nk <- 1:20
#loop for range of clusters
for (k in nk) {
  model <- kmeans(dailyscale[,c("lteq","posaff")], k)
  criteria <- rbind(criteria,c(k,model$tot.withinss,model$betweenss,model$totss))
}
#renaming columns
names(criteria) <- c("k","tot.withinss","betweenss","totalss")

#scree plot
ggplot(criteria, aes(x=k)) +
  geom_point(aes(y=tot.withinss),color="red") +
  geom_line(aes(y=tot.withinss),color="red") +
  geom_point(aes(y=betweenss),color="blue") +
  geom_line(aes(y=betweenss),color="blue") +
  xlab("k = number of clusters") + ylab("Sum of Squares (within = red, between =
blue)")

#also consider the quantitative criteria when choosing k
#looking at criteria
round(criteria,2)

#other ways to choose k
#from library(fpc)
model.manyCH <- kmeansruns(dailyscale[,c("lteq","posaff")],
                           krange=c(2:20), criterion="ch",critout = TRUE,
plot=FALSE) #better to leave the plot FALSE
model.manyCH

#another criteria
model.manyASW <- kmeansruns(dailyscale[,c("lteq","posaff")], krange=c(2:20),
                            criterion="asw",critout = TRUE, plot=FALSE) #better to
```

```
leave the plot FALSE
model.manyASW

#obtaining distance matrix
dist.all <-
dist(dailyscale[,c("lteq","posaff")],method="euclidean",diag=TRUE,upper=FALSE)
#obtaining metrics
cluster.stats(dist.all,clustering=model$cluster)

#library(clusterCrit)
#running a cluster analysis
model <- kmeans(dailyscale[,c("lteq","posaff")], centers=4)
#calculating various internal clustering validation or quality criteria
ic <- intCriteria(traj=as.matrix(dailyscale[,c("lteq","posaff")]),
part=model$cluster, crit="all")
ic

#5.1 OBTAINING A STABLE, RELYABLE SOLUTION
#use kmeans() function to autimate different starts and finding the most stable
solution
#25 or 50 are recommended for finding stable solutions
#kmeans with nstart = 1
km.res <- kmeans(dailyscale[,c("lteq","posaff")], centers=4, nstart = 1)
km.res$tot.withinss

#kmeans with nstart = 25
km.res <- kmeans(dailyscale[,c("lteq","posaff")], centers=4, nstart = 25)
km.res$tot.withinss

#kmeans with nstart = 50
km.res <- kmeans(dailyscale[,c("lteq","posaff")], centers=4, nstart = 50)
km.res$tot.withinss

#6.1 DESCRIBING THE CLUSTER
#merge the vector of cluster assignments into the dataset
dailyscale.clus <- cbind(km.res$cluster,dailyscale)
names(dailyscale.clus)[1] <- "cluster"
head(dailyscale.clus[,c(1:4,6)],4)

#describe the different clusters
library(tidyverse)

# Gather the data to 'long' format so the clustering variables are all in one column
#gather() has been replaced by pivot_longer()
longdata <- dailyscale.clus %>%
  pivot_longer(c(lteq, posaff), names_to = "variable", values_to = "value")
```

```
# Create the summary statistics seperately for cluster and variable (i.e. lteq,
posaff)
summary <- longdata %>%
  group_by(cluster, variable) %>%
  summarise(mean = mean(value), se = sd(value) / length(value))

# Plot
ggplot(summary, aes(x = variable, y = mean, fill = variable)) +
  geom_bar(stat = 'identity', position = 'dodge') +
  geom_errorbar(aes(ymin = mean - se, ymax = mean + se),
                width = 0.2,
                position = position_dodge(0.9)) +
  facet_wrap(~cluster)

#6.2 ANALYZING THE CLUSTERS
#analyse how clusters differ on another variable
fit1 <- aov(pss ~ factor(km.res$cluster), data=dailyscale.clus)
summary(fit1)

TukeyHSD(fit1)

#8 HIERARCHICAL CLUSTERING
#make distance matrix
dist.all <- daisy(dailyscale[,c("lteq","posaff")],metric="euclidean",stand=FALSE)
#loking at distances among first 5 persons
as.matrix(dist.all)[1:5,1:5]

# Compute Ward clusters for IGother with aggricultural nesting and ward method
#be sure to try other methods
clusterward.papa <- agnes(dist.all, diss = TRUE, method = "ward")

# Plot dendrogram
layout(matrix(1))
plot(clusterward.papa, which.plot = 2, main = "Ward clustering of PAPA")

#make cluster assignments
wardcluster4 <- cutree(clusterward.papa, k = 4)

#statistical criteria
cluster.stats(dist.all, clustering=wardcluster4,
              silhouette = TRUE, sepindex = TRUE)

#10 K-MEDOIDS
# Compute PAM clustering solution for k=4
clusterpam.papa <- pam(dist.all, k=4, diss = TRUE)
```

```
clusterpam.papa

#Checking length
pamcluster <- clusterpam.papa$clustering
length(pamcluster)

#binding to originaldata
dailyscale.pam <- cbind(dailyscale,pamcluster)
#plotting clustered data points
ggplot(dailyscale.pam,aes(x=lteq,y=posaff)) +
  geom_point(alpha=.6, color=factor(pamcluster))

#run the autosearch
pamauto <- pamk(dist.all,krange=2:10,criterion="asw", usepam=TRUE,
                scaling=FALSE, alpha=0.001, diss=TRUE,
                critout=FALSE, ns=10, seed=NULL)
pamauto

#Obtain medoids
pamauto$pamobject$id.med

#binding new cluster assignment to originaldata
dailyscale.pam$pamnew <- pamauto$pamobject$clustering

#plotting clustered data points with the medoids
ggplot(dailyscale.pam,aes(x=lteq,y=posaff)) +
  geom_point(alpha=.6, color=factor(dailyscale.pam$pamnew)) +
  geom_point(data=dailyscale.pam[598,],aes(x=lteq,y=posaff),color=2,size=5,shape=18)
+
  geom_point(data=dailyscale.pam[63,],aes(x=lteq,y=posaff),color=1,size=5,shape=18)
+
  geom_point(data=dailyscale.pam[738,],aes(x=lteq,y=posaff),color=4,size=5,shape=18)
```

K-means Script

The following is my R script for k-means clustering of WIBS single particle data

```
cluster_fluorescent <- read.csv("SPD_Fluorescent.csv")
cluster_nonfl <- read.csv("SPD_Non_Fluorescent.csv")

df_fl <- cluster_fluorescent
df_nfl <- cluster_nonfl

##############################################################################
############
```

```
#K-MEANS CLUSTER
################################################################################
###########
#general packages
library(ggplot2)
library(psych)

#cluster packages
library(cluster) #clustering
library(fpc) #flexible procedures for clustering
library(clusterCrit) #cluster criteria

#FLUORESCENT

#remove missing data
df_fl <- df_fl[complete.cases(df_fl),]

#scaling all the variables
df_fl_scale <- data.frame(scale(df_fl, center=TRUE, scale=TRUE))

#there are random starts involved so we set a seed
set.seed(50)
#running a cluster analysis
km_fl <- kmeans(df_fl_scale, centers=15)
km_fl

#getting centers
km_fl$centers

#sum of squares within clusters
km_fl$totss #total sum of squares
km_fl$withinss #how close the observations are within each cluster
km_fl$tot.withinss #sum of the ss from each cluster (smaller the better)
km_fl$betweenss #how far the clusters are from each other (larger the better)

#plot the ss values to determine the best k value
#making a empty dataframe
criteria_fl <- data.frame()
#setting range of k
nk <- 1:20
#loop for range of clusters
for (k in nk)
{
  km_fl <- kmeans(df_fl_scale, k)
  criteria_fl <-
rbind(criteria_fl,c(k,km_fl$tot.withinss,km_fl$betweenss,km_fl$totss))
```

```
}
#renaming columns
names(criteria_fl) <- c("k","tot.withinss","betweenss","totalss")

#scree plot
ggplot(criteria_fl, aes(x=k)) +
  geom_point(aes(y=tot.withinss),color="red") +
  geom_line(aes(y=tot.withinss),color="red") +
  geom_point(aes(y=betweenss),color="blue") +
  geom_line(aes(y=betweenss),color="blue") +
  xlab("k = number of clusters") + ylab("Sum of Squares (within = red, between =
blue)")



#NON-FLUORESCENT

#remove missing data
df_nfl <- df_nfl[complete.cases(df_nfl),]

#scaling all the variables
df_nfl_scale <- data.frame(scale(df_nfl, center=TRUE, scale=TRUE))

#there are random starts involved so we set a seed
set.seed(50)
#running a cluster analysis
km_nfl <- kmeans(df_nfl_scale, centers=6)
km_nfl

#getting centers
km_nfl$centers

#sum of squares within clusters
km_nfl$totss #total sum of squares
km_nfl$withinss #how close the observations are within each cluster
km_nfl$tot.withinss #sum of the ss from each cluster (smaller the better)
km_nfl$betweenss #how far the clusters are from each other (larger the better)

#plot the ss values to determine the best k value
#making a empty dataframe
criteria_nfl <- data.frame()
#setting range of k
nk <- 1:75
#loop for range of clusters
for (k in nk)
{
  km_nfl <- kmeans(df_nfl_scale, k)
```

```
  criteria_nfl <-
rbind(criteria_nfl,c(k,km_nfl$tot.withinss,km_nfl$betweenss,km_nfl$totss))
}
#renaming columns
names(criteria_nfl) <- c("k","tot.withinss","betweenss","totalss")

#scree plot
ggplot(criteria_nfl, aes(x=k)) +
  geom_point(aes(y=tot.withinss),color="red") +
  geom_line(aes(y=tot.withinss),color="red") +
  geom_point(aes(y=betweenss),color="blue") +
  geom_line(aes(y=betweenss),color="blue") +
  xlab("k = number of clusters") + ylab("Sum of Squares (within = red, between =
blue)")
```

Hierarchical Script

The following is my R script for Hierarchical clustering of WIBS single particle data

```
cluster_fluorescent <- read.csv("SPD_Fluorescent.csv")
cluster_nonfl <- read.csv("SPD_Non_Fluorescent.csv")

df_fl <- cluster_fluorescent
df_nfl <- cluster_nonfl

##############################################################################################
###########
#Hierarchical Clustering
##############################################################################################
###########
#cluster packages
library(cluster)

#FLUORESCENT
#WARD

#remove missing data
df_fl <- df_fl[complete.cases(df_fl),]

#scaling all the variables
df_fl_scale <- data.frame(scale(df_fl, center=TRUE, scale=TRUE))

#distance variable
dist.all_fl <- daisy(df_fl_scale,metric="euclidean",stand=FALSE)
```

```
# Compute Ward clusters for IGother
clusterward_fl <- agnes(dist.all_fl, diss = TRUE, method = "ward")

# Plot
layout(matrix(1))
plot(clusterward_fl, which.plot = 2, main = "Ward clustering")

#make cluster number assignment
clusterward_kfl <- cutree(clusterward_fl, k = 4)

#cluster stats
cluster.stats(dist.all_fl, clustering=clusterward_kfl,
              silhouette = TRUE, sepindex = TRUE)


#K-MEDOIDS

# Compute PAM clustering solution for k=4
clusterpam_fl <- pam(dist.all_fl, k=4, diss = TRUE)
clusterpam_fl

#Checking length
pamcluster_fl <- clusterpam_fl$clustering
length(pamcluster_fl)

#binding to original data
fl_scale.pam <- cbind(df_fl_scale,pamcluster_fl)

#autostretch
pamauto_fl <- pamk(dist.all_fl,krange=2:10,criterion="asw", usepam=TRUE,
                scaling=FALSE, alpha=0.001, diss=TRUE,
                critout=FALSE, ns=10, seed=NULL)
pamauto_fl

#Obtain medoids
pamauto_fl$pamobject$id.med

#binding new cluster assignment to originaldata
df_fl_scale.pam$pamnew <- pamauto$pamobject$clustering
```

Fuzzy Script

The following is my R script for fuzzy clustering of WIBS single particle data[4]

```
cluster_fluorescent <- read.csv("SPD_Fluorescent.csv")
cluster_nonfl <- read.csv("SPD_Non_Fluorescent.csv")

df_fl <- cluster_fluorescent
df_nfl <- cluster_nonfl
##############################################################################################
###########
#FUZZY CLUSTER ANALYSIS 1
#https://rpubs.com/rahulSaha/Fuzzy-CMeansClustering
##############################################################################################
###########
#install these packages
require(ppclust)
require(factoextra)
require(dplyr)
require(cluster)
require(fclust)
require(psych)

#scale the data
df_fl_scale <- data.frame(scale(df_fl, center=TRUE, scale=TRUE))
describe(df_fl_scale)

#fuzzy cluster code
res.fcm <- fcm(df_fl_scale,centers = 15,nstart = 4)
  #as.data.frame(res.fcm$u)
summary(res.fcm)

#create cluster data frame
fuzzy_cl_fl <- data.frame(res.fcm$cluster)
colnames(fuzzy_cl_fl) <- c("cluster")



#DATA VALIDATION
res.val <- ppclust2(res.fcm, "fclust")

#Fuzzy Silhouette Index:
fsi <- SIL.F(res.val$Xca, res.val$U, alpha = 1)
#Partition Entropy:
pe <- PE(res.val$U)
#Partition Coefficient:
pc <- PC(res.val$U)
#Modified Partition Coefficient:
mpc <- MPC(res.val$U)

data_val <- data.frame(matrix(ncol = 4,nrow = 1))
```

```
data_val[,1] <- fsi
data_val[,2] <- pe
data_val[,3] <- pc
data_val[,4] <- mpc
colnames(data_val) <- c("Fuzzy Silhouette Index","Partition Entropy",
                        "Partition Coefficient","Modified Partition Coefficient")




################################################################################
###########
#FUZZY CLUSTER ANALYSIS 2
#https://search.r-
project.org/CRAN/refmans/fclust/html/Fclust.html#:~:text=Arguments%20%20%20X%20%20%2
0Matrix%20or,cluster%20variant%20...%20%202%20more%20rows%20
################################################################################
###########
require(fclust)

fl_fclust <- Fclust(df_fl_scale, k = 6, type = "standard", noise = TRUE)
summary(fl_fclust)



################################################################################
###########
#FUZZY CLUSTER ANALYSIS 3
#https://www.datanovia.com/en/lessons/fuzzy-clustering-essentials/cmeans-r-function-
compute-fuzzy-clustering/
################################################################################
###########
library(e1071)
#compute fuzzy clustering
  #x: a data matrix where columns are variables and rows are observations
  #centers: Number of clusters or initial values for cluster centers
  #iter.max: Maximum number of iterations
  #dist: Possible values are “euclidean” or “manhattan”
  #m: A number greater than 1 giving the degree of fuzzification.
cm <- cmeans(df_fl_scale, centers = 6, iter.max = 100, dist = "euclidean", m=2)

#Membership coefficient
head(cm$membership)

# Visualize using corrplot
library(corrplot)
```

```
corrplot(cm$membership, is.corr = FALSE)

# Observation groups/clusters
fuzzy_cl_fl <- data.frame(cm$cluster)
colnames(fuzzy_cl_fl) <- c("cluster")

#visualize cluster
library(factoextra)
fviz_cluster(list(data = df, cluster=cm$cluster),
             ellipse.type = "norm",
             ellipse.level = 0.68,
             palette = "jco",
             ggtheme = theme_minimal())
```

---

1. https://quantdev.ssri.psu.edu/sites/qdev/files/11_ClusterAnalysis_2020_0401.html ↩
2. Cluster Tutorial.pdf ↩ ↩
3. Hierarchical Clustering-upfedu michael stanford.pdf ↩
4. https://rpubs.com/rahulSaha/Fuzzy-CMeansClustering ↩