MONASH University

ETX2250/ETF5922: Data Visualization and Analytics

K means clustering

Lecturer: *Lauren Kennedy*

Department of Econometrics and Business Statistics

lauren.kennedy1@monash.edu

Week 8

Partitional Clustering

1. Random assignment of points in the midst of cluster
2. Repeatedly re-assign points cluster
3. Minimise within-cluster distances
4. Maximise between-cluster distances

One example is k-means, which we focus on today

Comparing clustering algorithms

Hierarchical clustering

- Suitable for small data set (<500 observations)
- Easily examine solutions with increasing numbers of clusters
- Convenient method if you want to observe how clusters are nested

K-means clustering

- Suitable if you know how many clusters you want
- Large data set
- Good for summarising data with k "average" observations that describe the data with the minimum amount of error

Obtaining clusters: K means method

- Non-hierarchical clustering method
- Require number of clusters be specified before assigning observations
- No single objective method for determining the number of clusters

Obtaining clusters: K means method

Consider the following

- Want cohesion with, diversity between
- How much improvement does one more cluster provide?
- Single member or very small clusters are not usually useful
- All clusters should be significantly different across the set of clustering variables
- The utility of the clusters ultimately depends on external validation
  - Other variables not used in the analysis
  - External information, e.g. location, geopolitical information

K-means Clustering

- Given the value of K, the K-means algorithm **randomly** partitions the observations into K clusters
- After all observations have been assigned to a cluster, the resulting cluster centroids are calculated
- Using the updated cluster centroids, all observations are reassigned to the cluster with the closest centroid (usually Euclidean distance)
- Continue until the assignments no longer change or a predetermined maximum number of iterations is reached

K-means Clustering

- Number of K clusters needs to be specified, usually advisable to try several values of K
- Initial K clusters are chosen randomly and the results do depend strongly on this particular starting point – procedure should be repeated a number of times (>10) with different starting points
- The “best” result is chosen

The algorithm in pictures

The algorithm in pictures

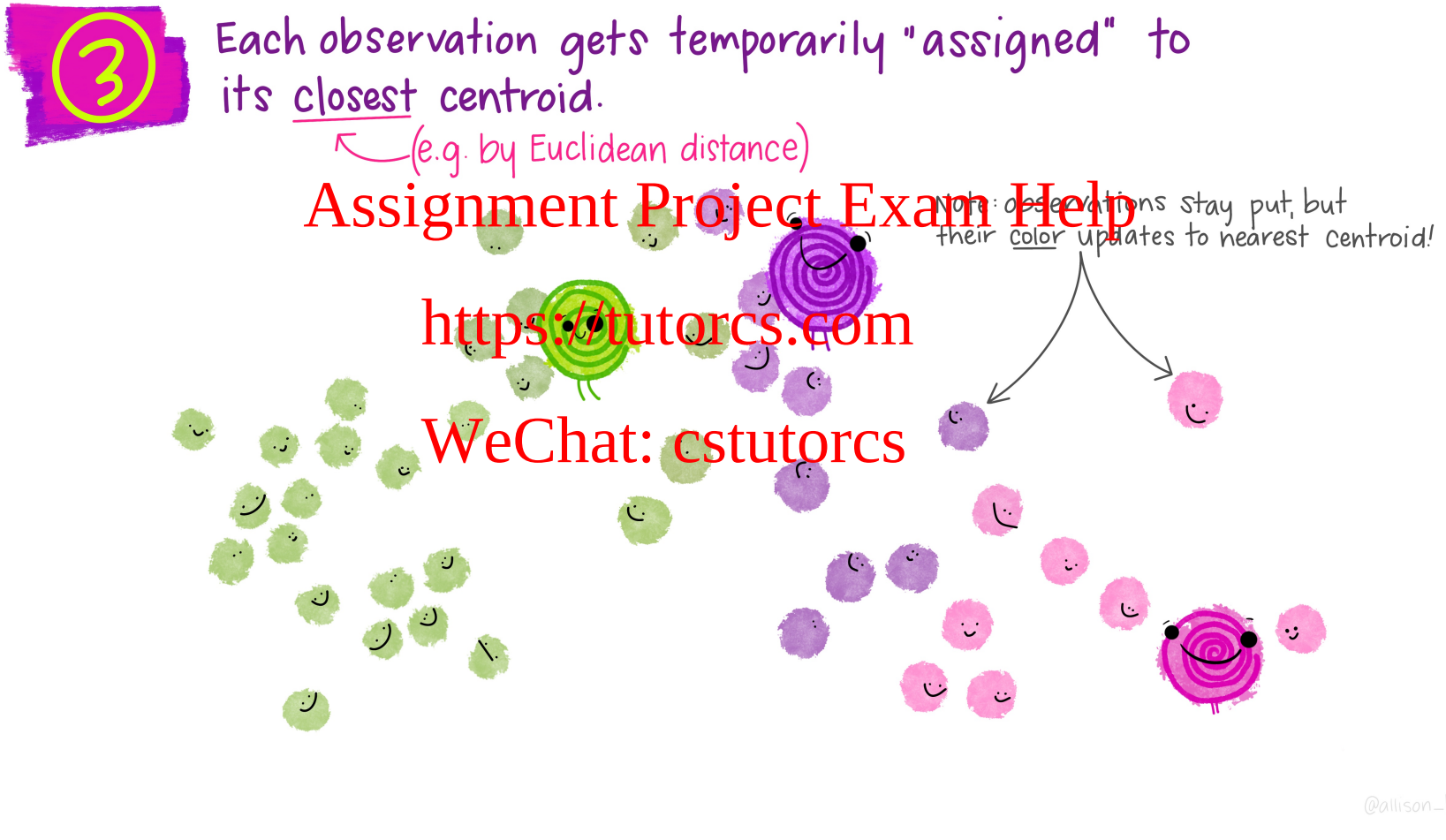
Specify the number of clusters (in this example, k = 3).

Then imagine k cluster centroids are created.

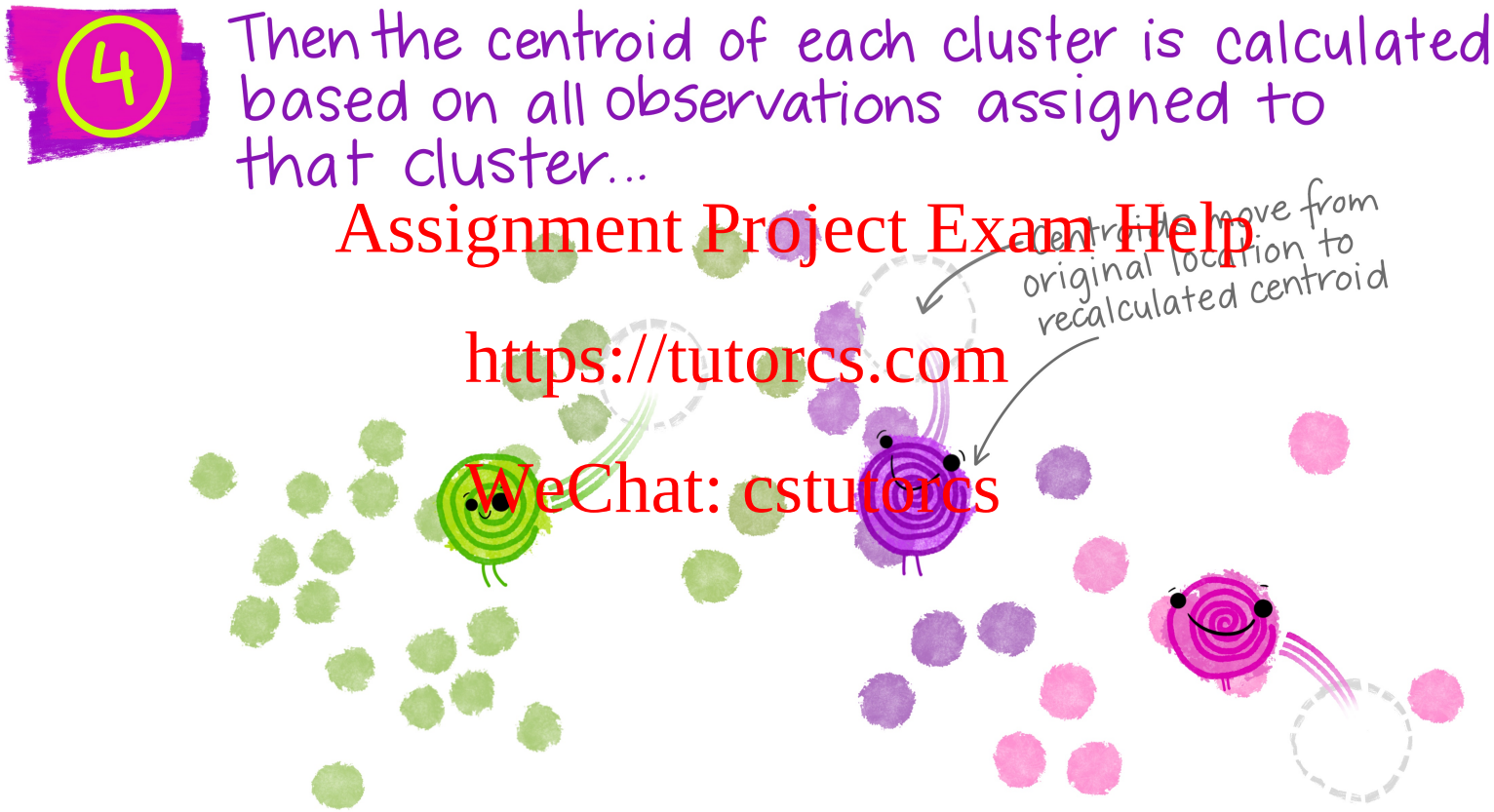
@allison_horst

The algorithm in pictures

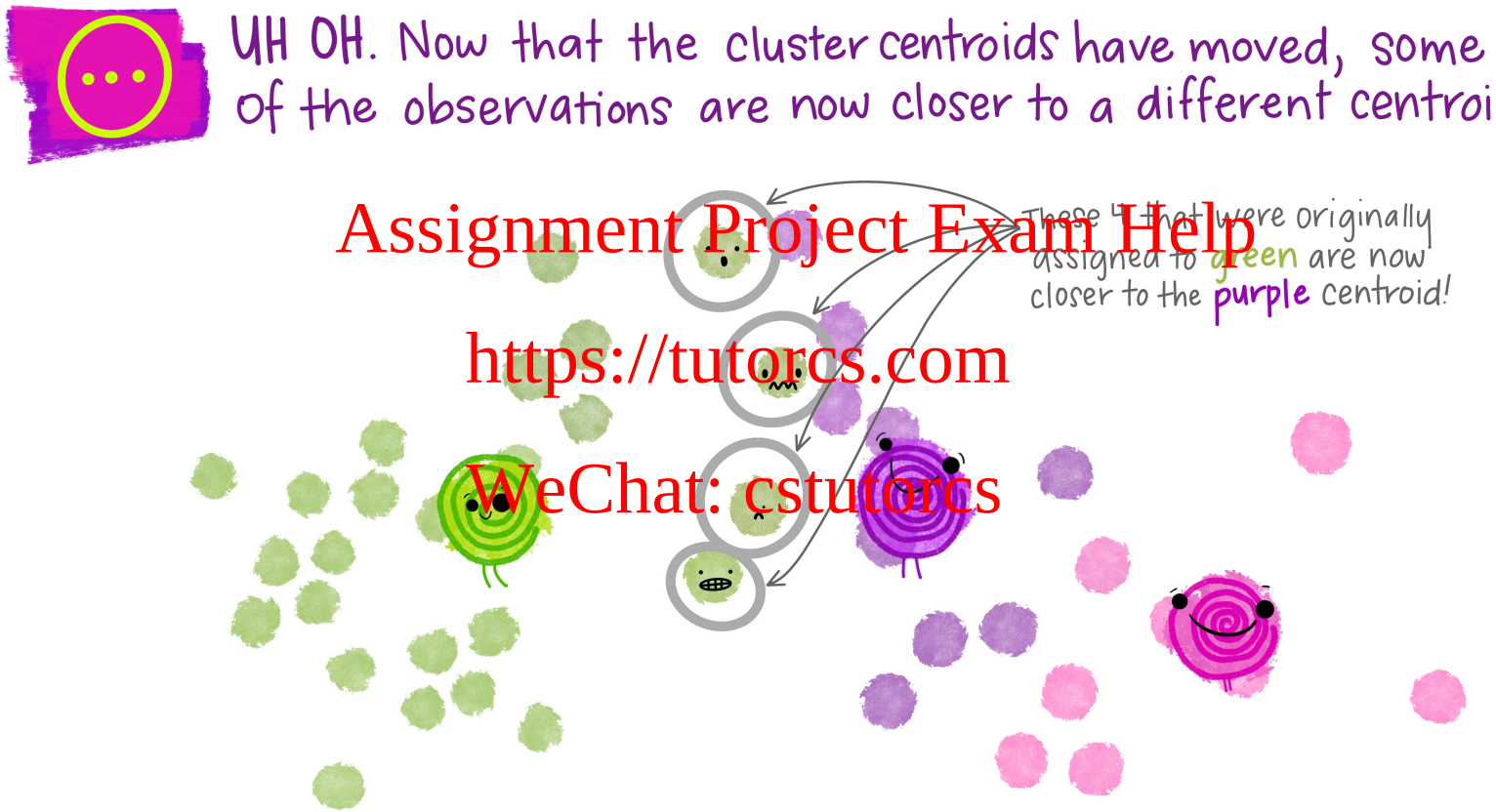
The algorithm in pictures



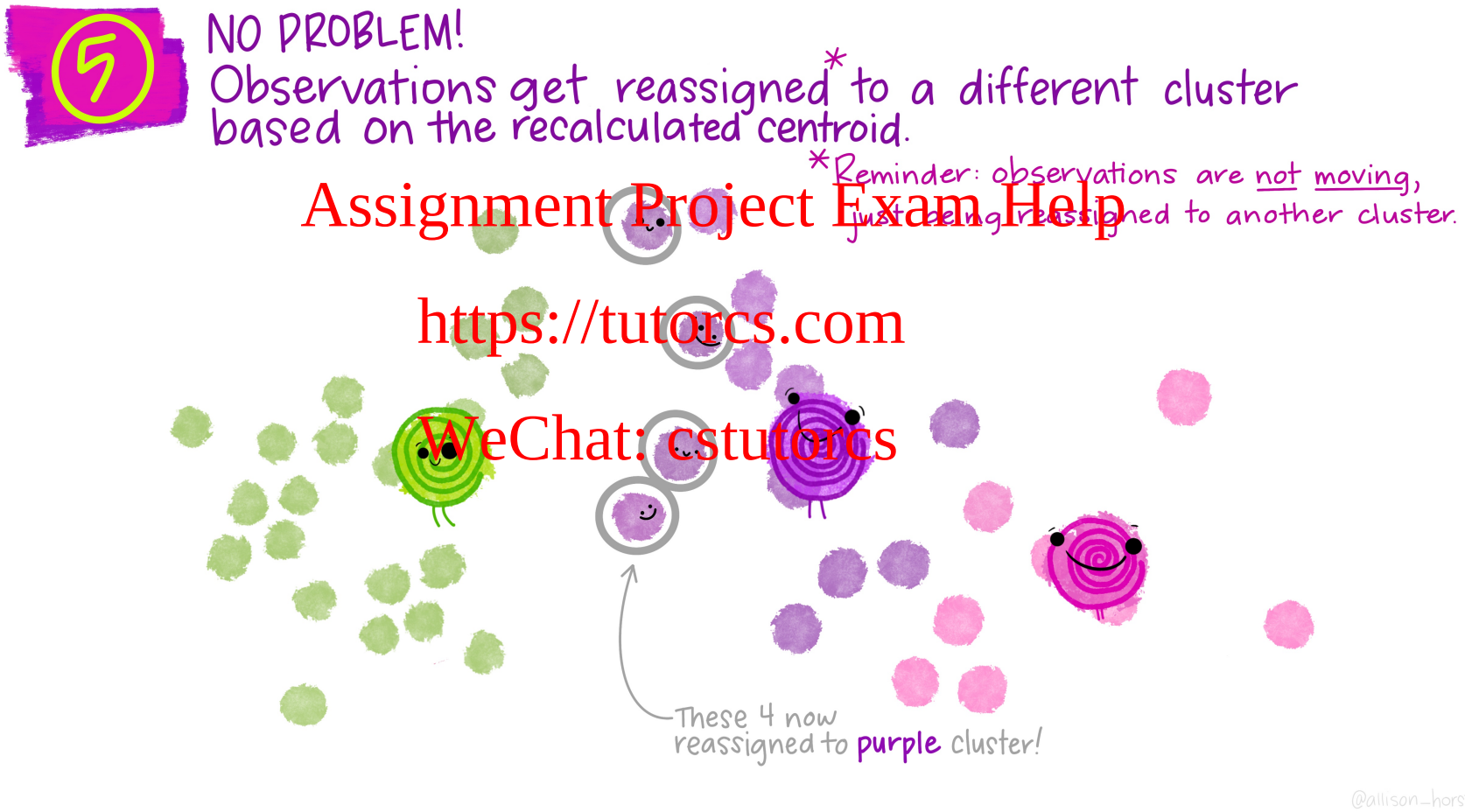
The algorithm in pictures



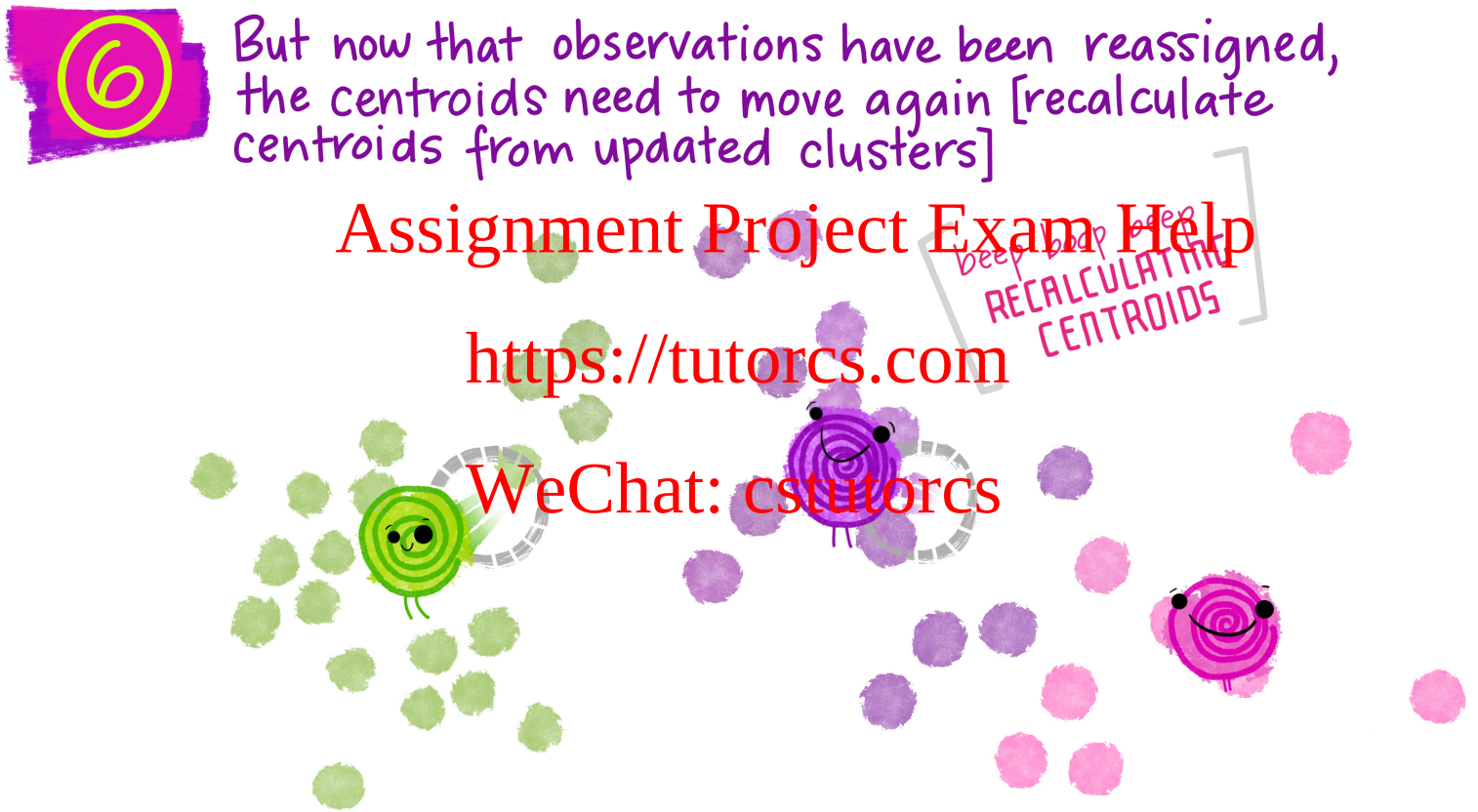
The algorithm in pictures



The algorithm in pictures



The algorithm in pictures



The algorithm in pictures

The algorithm in pictures

Then the centroid for each cluster is recalculated...

...which means observations will be reassigned...

@allison_horst

The algorithm in pictures

That iterative process of

Recalculate cluster centroids
→ Reassign observations to nearest centroid
→ Recalculate cluster centroids
→ Reassign observations to nearest centroid
→ Recalculate cluster centroids
→ Reassign observations to nearest centroid

↓

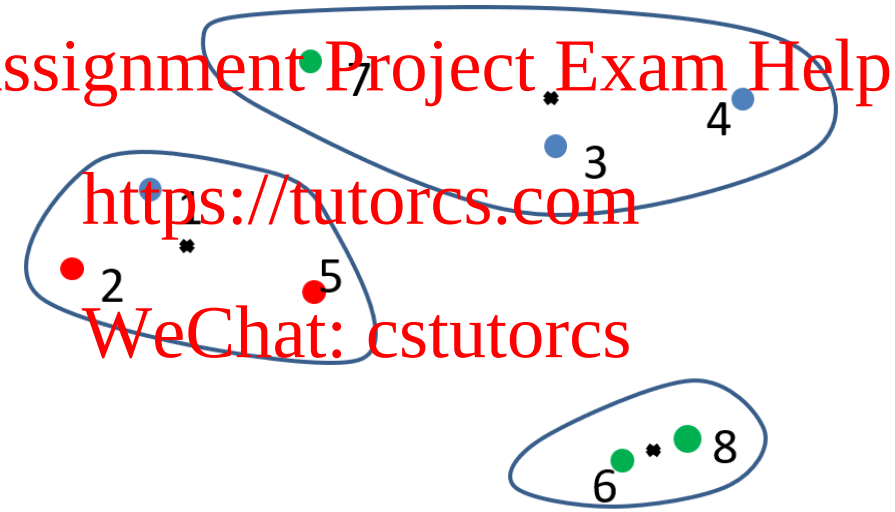
Continues until nothing is moving or being reassigned anymore!

@allison_horst

The algorithm in pictures

Centroid

- A centroid is the average observation of a cluster
- The average value for each variable across the observations in the cluster
- Centroid does not have to be an actual observation



Recall KTC Example

- Example: continuous variables in KTC data
- These are: Age, Income, Children (columns 2, 4 and 6

ID	Age	Female	Income	Married	Children	CarLoan	Mortgage
1	49	1	17546	0	1	0	0
2	40	0	30085	1	3	1	1
3	51	1	16375	1	0	1	0
4	23	1	20375	1	3	0	0

- Use Euclidean distance: the variables will need standardising so that each of the three variables is on a similar scale. This is easily done in R.

Within Sum of Squares

- Use $\|$value $\|$to represent the Euclidean distance in *n* dimensional space, where *n* is the number of variables
- Suppose we have $k$clusters, call them $S_1, \ldots, S_k$
- We are trying to select the clusters in such a way as to minimise the “within sum of squares”

$$\sum_{j=1}^{k} \sum_{x \in S_j} \|x - \bar{m}_j\|^2$$

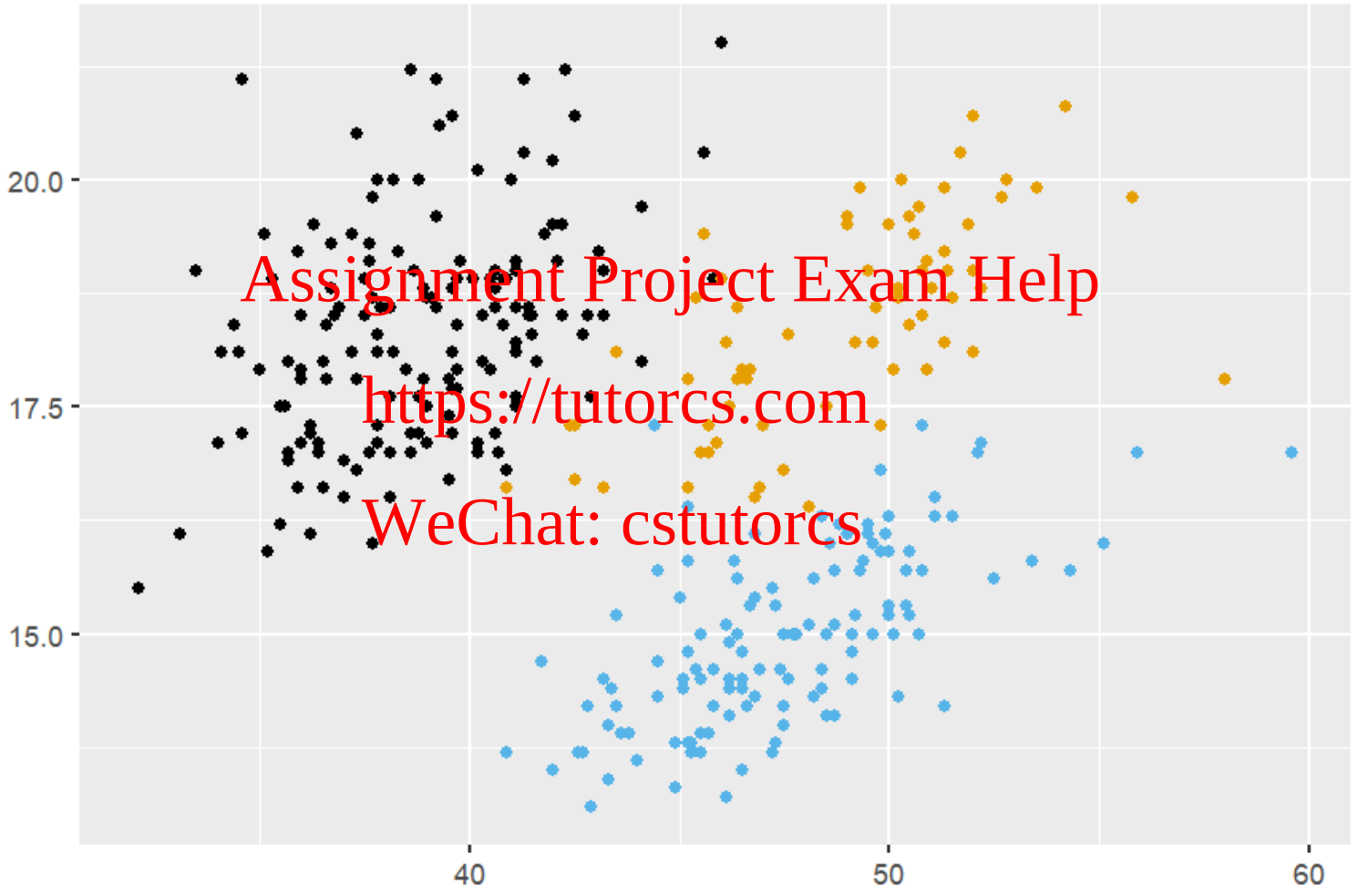
Total SS and between SS

Total SS = the sum of squared distances between each data point to the global sample mean

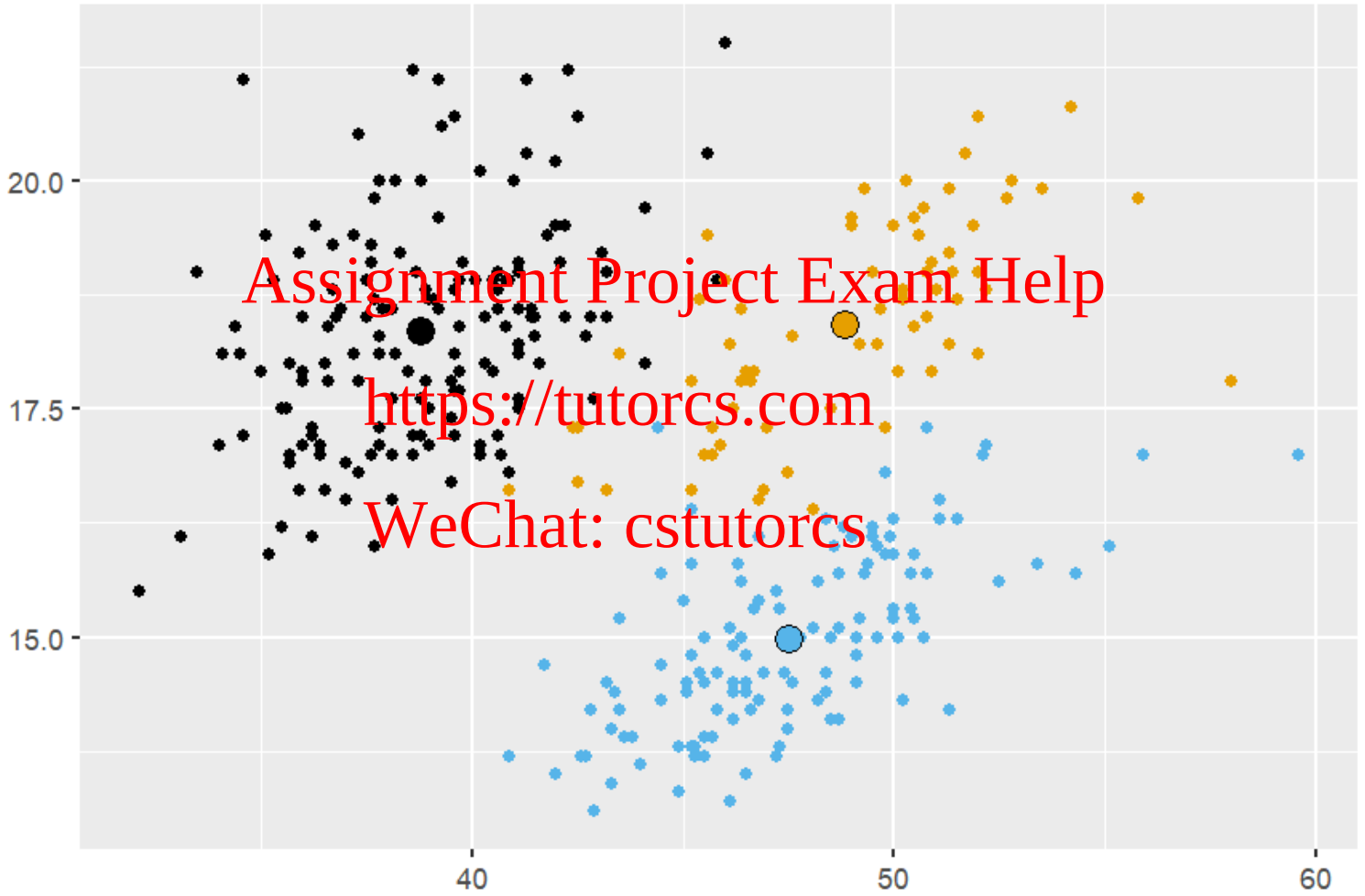
Obtain between SS:

1. For each cluster calculate the squared distance from the centroid of the cluster to the global sample mean
2. Multiply the number of points in the cluster
3. Add up the results for each cluster
4. If there were no discernible pattern of clustering, the three means of the three groups would be close to the global mean, and between SS would be a very small fraction of total SS5

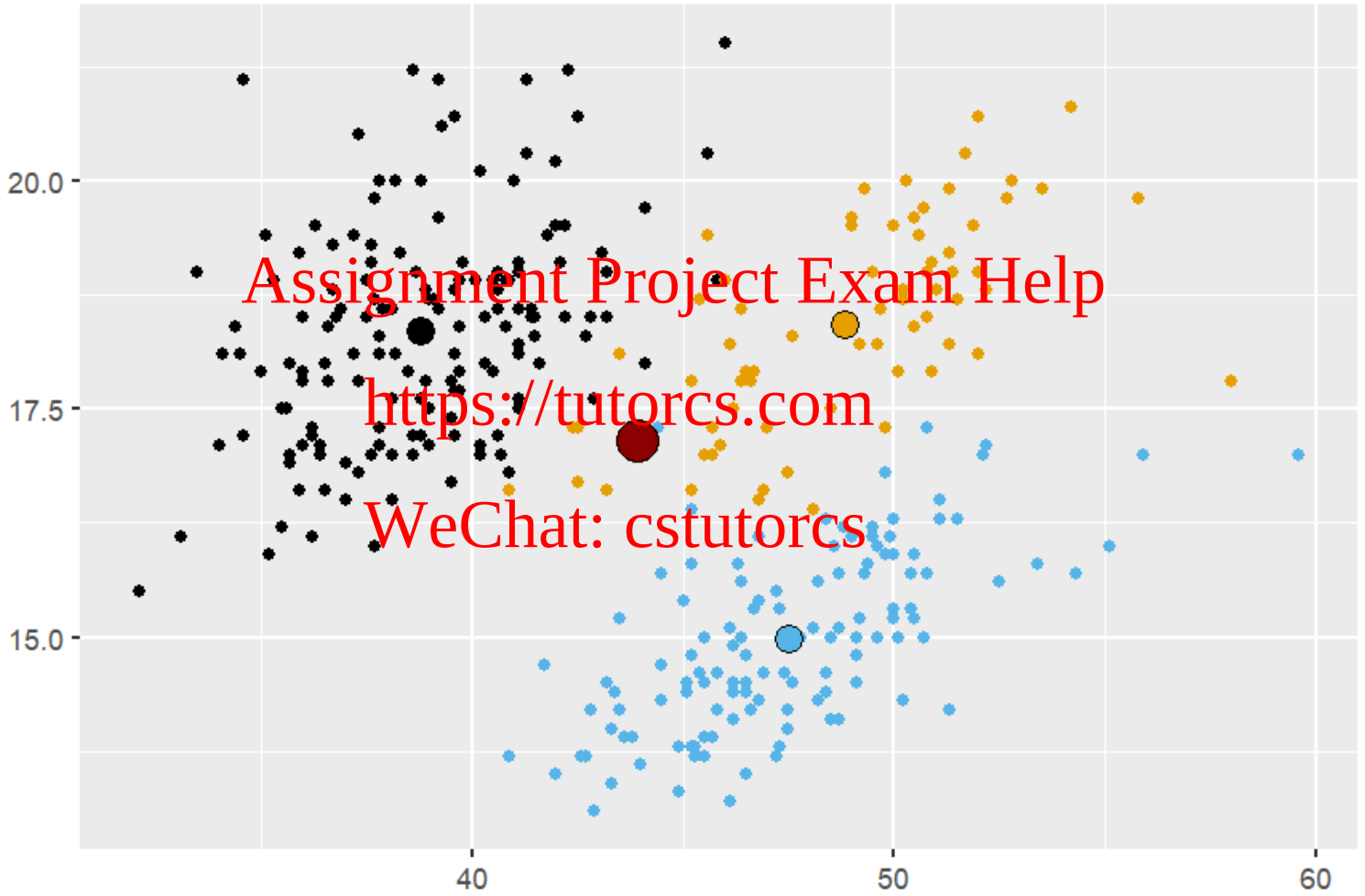
Visualise



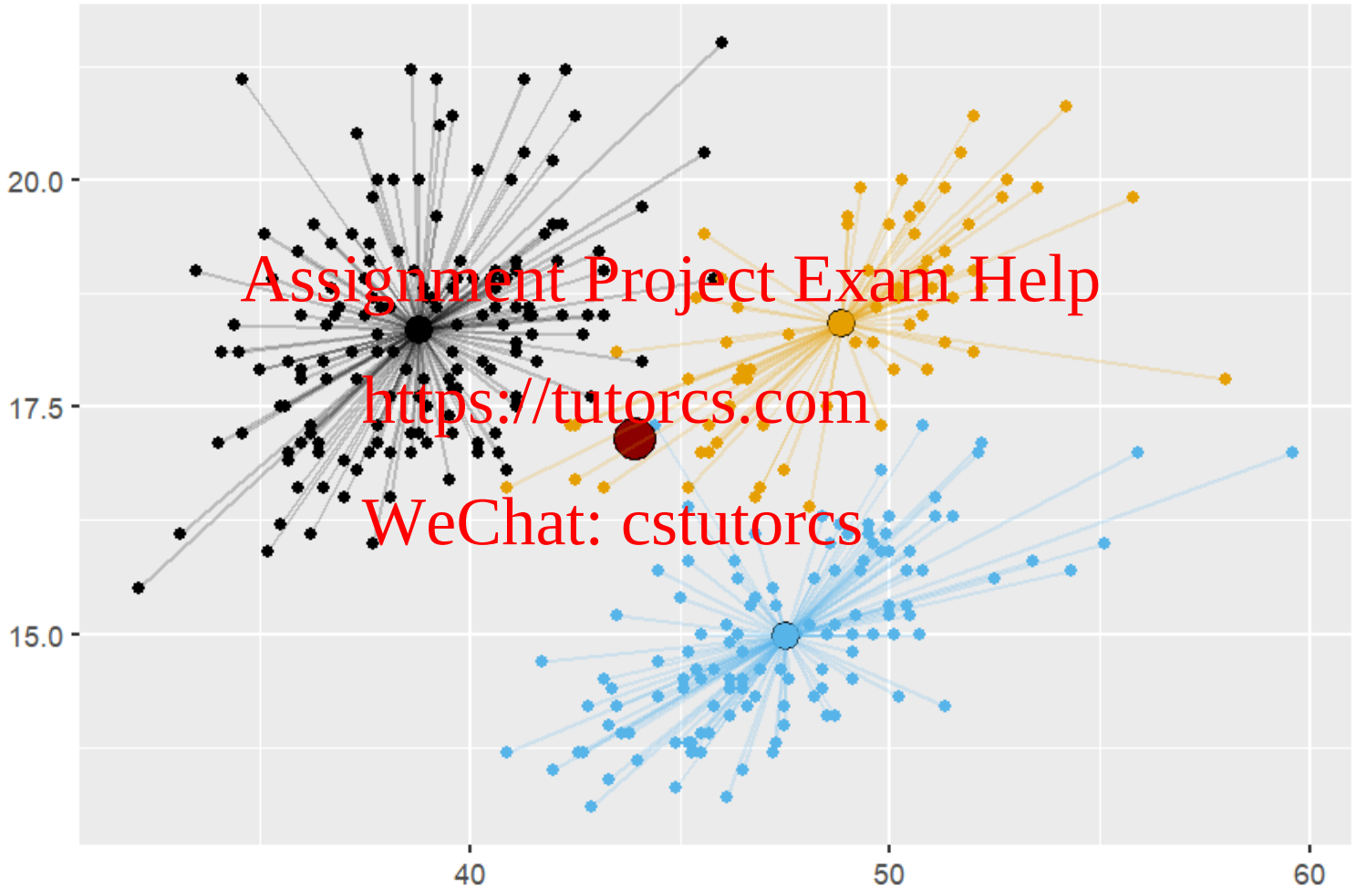
Group centroids



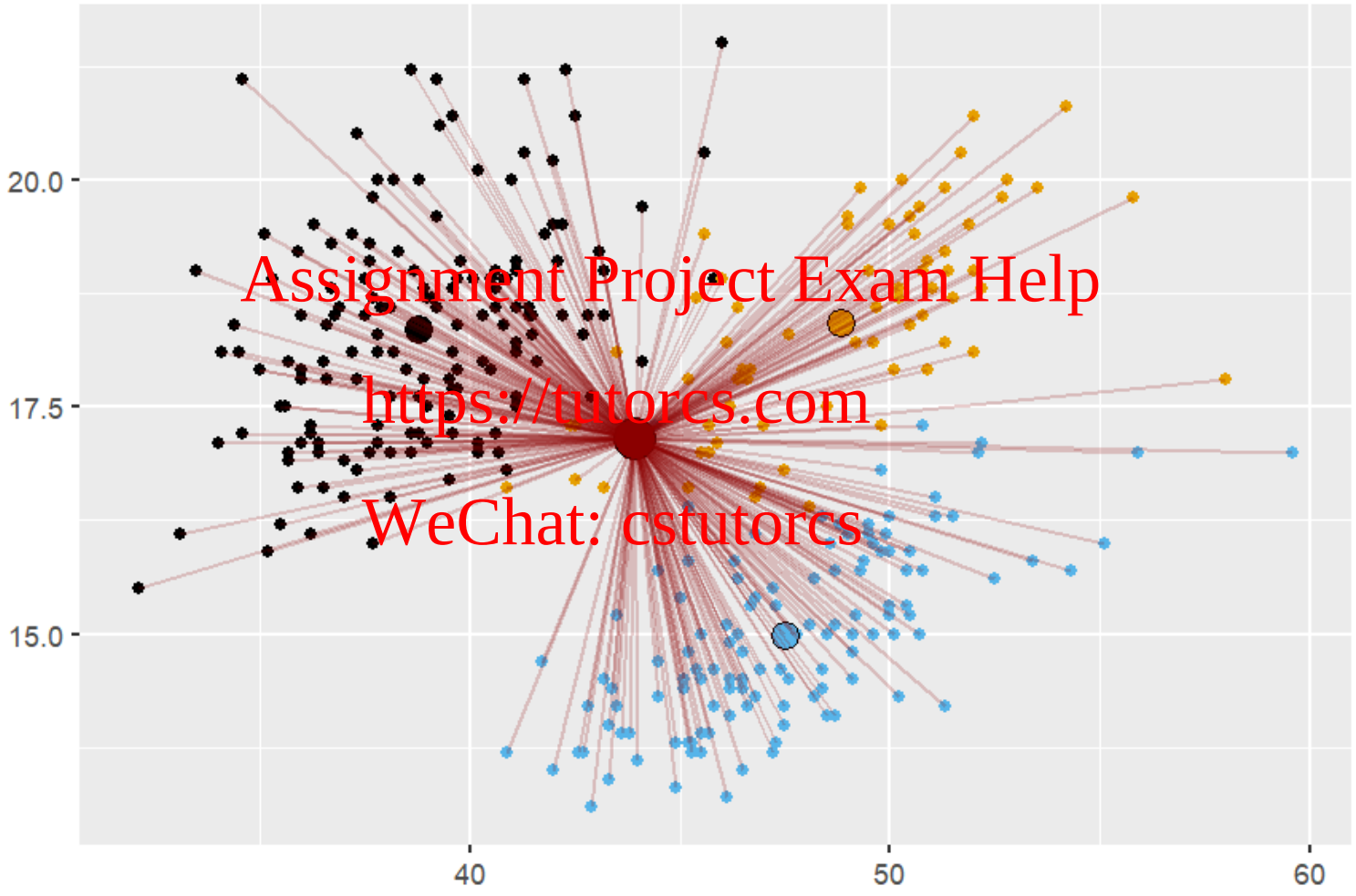
Overall centroid



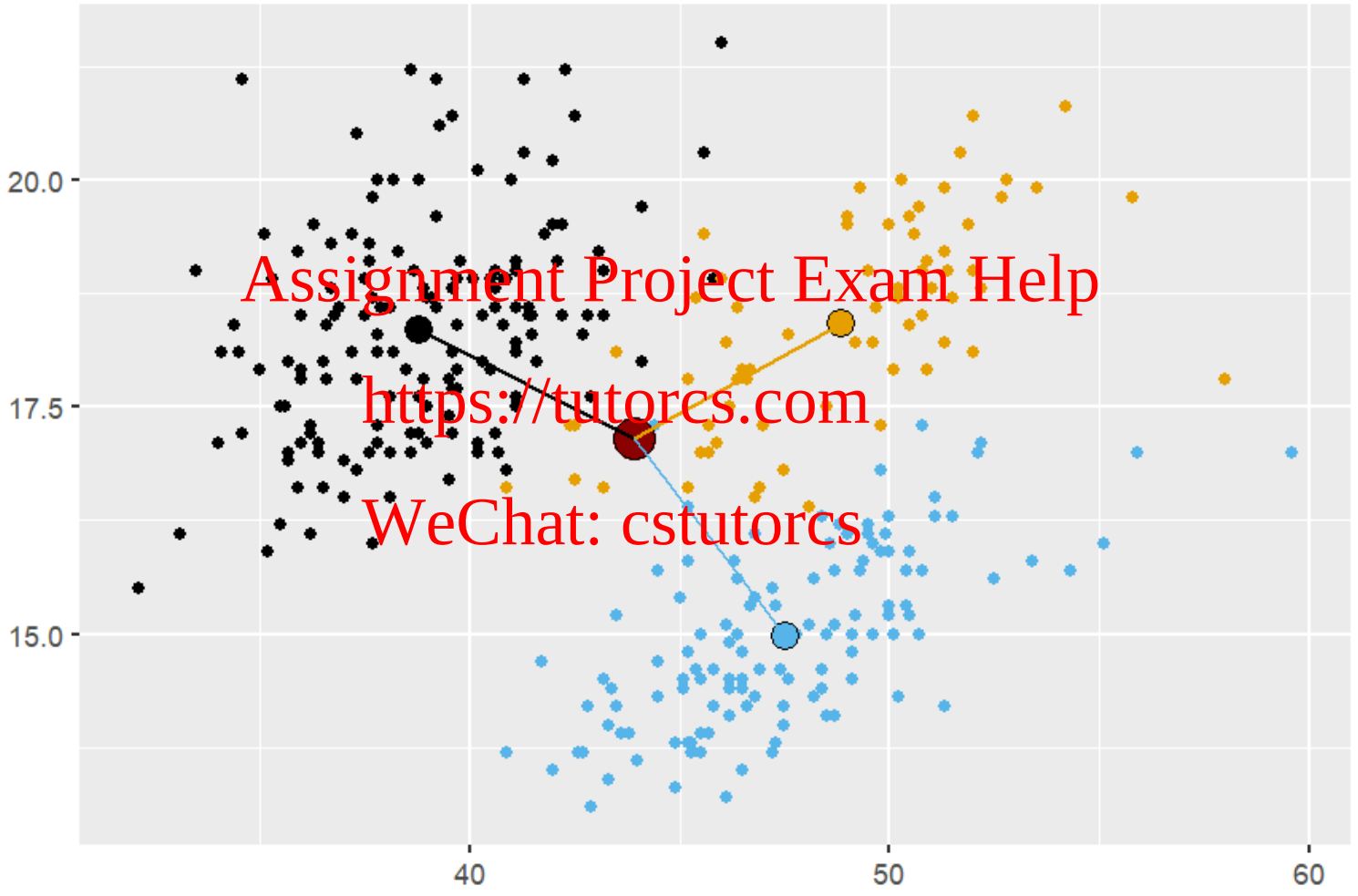
Within sum of squares



Total sum of squares



Between sum of squares



Outcome measure

- The algorithm reapeats this process (calculate cluster centroid, assign observations to the cluster with the nearest cenroid) until there is no change in the clusters or a specified maximum number of iterations has been reached
- Since we want little variation within clusters and a lot of variations between clusters, the ratio

$$\frac{SS_{BTWN}}{SS_{TOT}}$$

- Is a possible measure of how good the clustering that k-means has found

- One rule of thumb is that the ratio

$$\frac{SS_{BTWN}}{SS_{WTHN}}$$

should exceed one (i.e. there is more variation between clusters than within.)

How to in R

Know thy Customer

- Remember from your tutorials last week

```
ktc_df <- read_csv(here::here("data/KTC.csv"))
head(ktc_df)
```

```
## # A tibble: 6 x 8
##      ID   Age Female Income Married Children CarLoan Mortgage
##   <dbl> <dbl>  <dbl>  <dbl>   <dbl>    <dbl>   <dbl>    <dbl>
## 1     1    48      1  17546       0        1       0        0
## 2     2    40      0  30085       1        3       1        1
## 3     3    51      1  16575       1        0       1        0
## 4     4    23      1  20375       1        3       0        0
## 5     5    57      1  50576       1        0       0        0
## 6     6    57      1  37870       1        2       0        0
```

Focus only on the numeric predictors:

```
ktc_numeric <- ktc_df %>%
  select(ID,Age,Income, Children)%>%

  # We move the ID to rownames so we can keep track of the rows but ID
  #doesn't get included in the clustering (which it would if we left
  #it in the data frame)

  column_to_rownames(var = "ID")%>%

# Remember last week? k-means has the same sensitivity to scale,
# so we need to scale all our numeric variables.

  scale
```

```
head(ktc_numeric)
```

```
##          Age     Income    Children
## 1  0.1559026 -0.7637480  0.06169096
## 2 -0.4574848  0.1512843  1.91241969
## 3  0.3859229 -0.8346067 -0.86367341
## 4 -1.7609332 -0.5573020  1.91241969
## 5  0.8459635  1.6466130 -0.86367341
## 6  0.8459635  0.7193939  0.98705533
```

How do we do kmeans?

```
set.seed(145)
km <- kmeans(ktc_numeric, centers = 3, nstart = 25)
```

- centers is the number of clusters. Note that for k-means (unlike hierarchical clustering), we need to specify this. You can also specify a set of cluster centers
- nstart is the number of times you run through the procedure. You only need to specify this if you specify the number of clusters (why? more on that later...)

What does km look like?

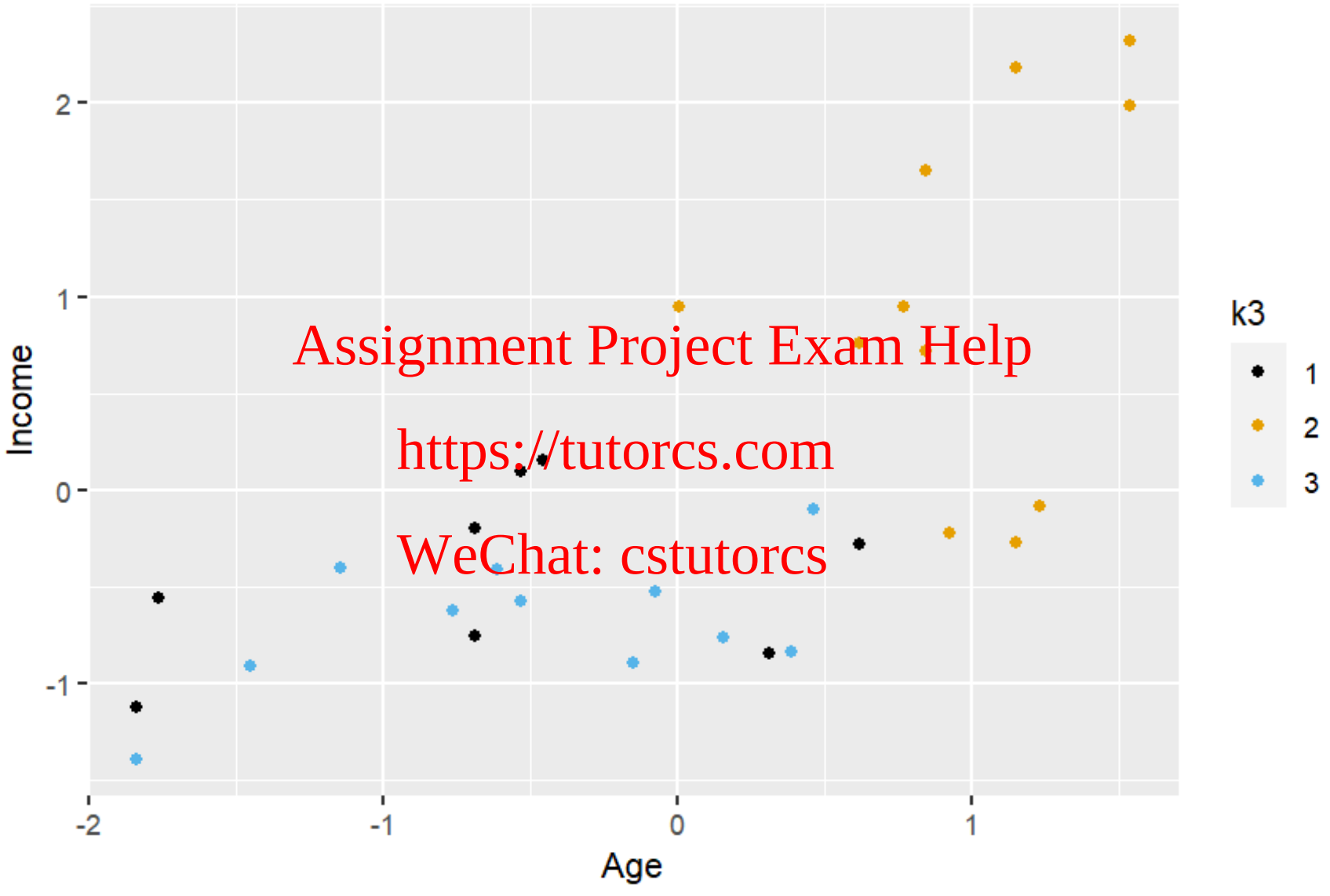
```
km
```

```
## K-means clustering with 3 clusters of sizes 8, 11, 11
##
## Cluster means:
##          Age     Income  Children
## 1 -0.6300000 -0.4368752  1.3340670
## 2  0.9644588  0.9939719 -0.1589292
## 3 -0.5062770 -0.6762445 -0.6113013
##
## Clustering vector:
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
##  3  1  3  1  2  2  3  2  1  1  2  3  3  2  3  3  1  2  2  3  2  1  2  3  1  2
## 27 28 29 30
##  3  3  1  2
##
## Within cluster sum of squares by cluster:
```

What do the clusters look like?

- Let's use our visualisation skills!

```
ktc_numeric %>%
  #This is technically not a dataframe, so let's make it so
  as.data.frame() %>%
  # Add in the cluster allocations
  mutate(k3 = factor(km$cluster))%>%
  ggplot(data = .,
         aes (x =Age,
              y= Income,
              colour = k3))+
  geom_point()+
  ggthemes::scale_color_colorblind()
```



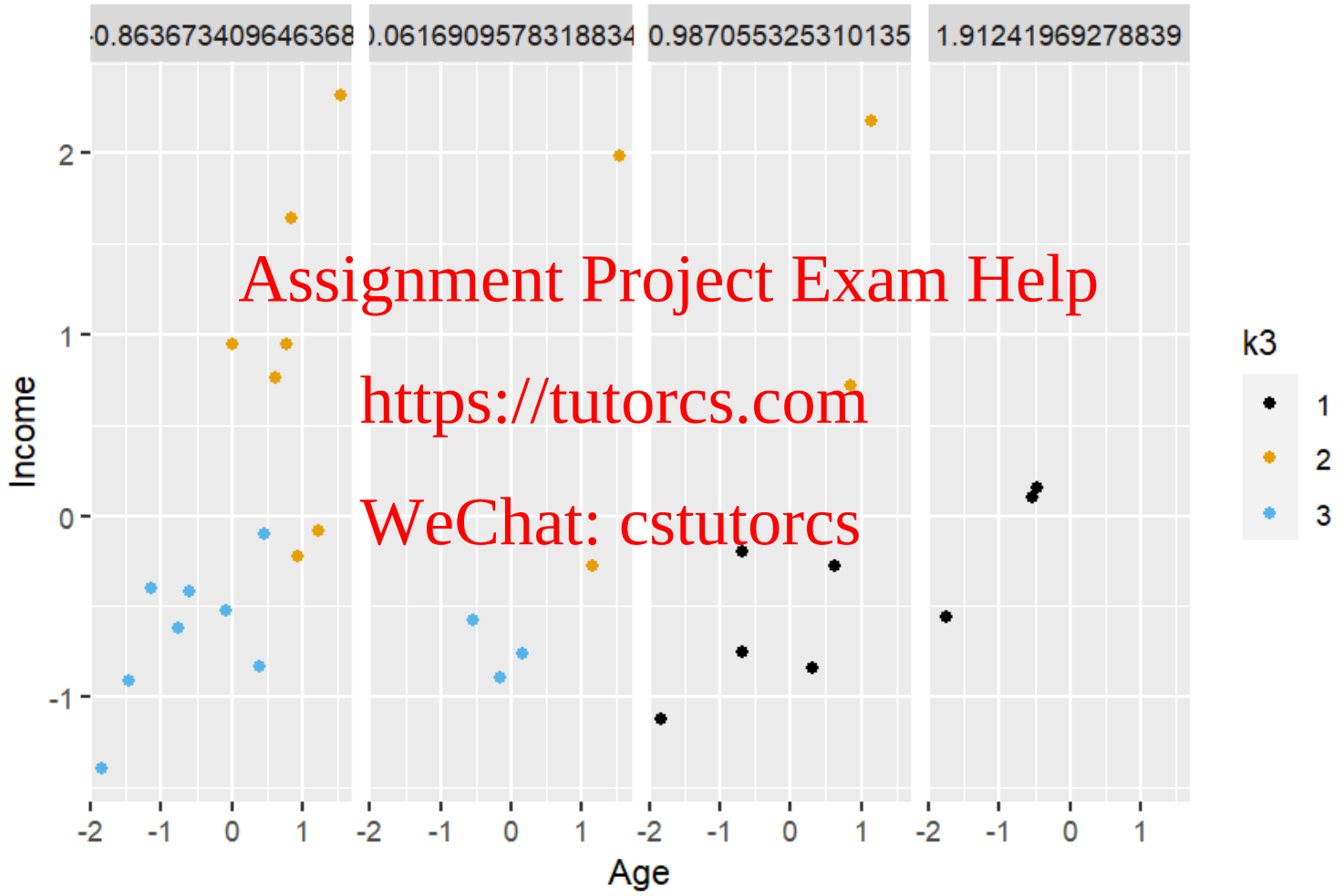
How can we summarise the clusters?

- Cluster 1 characterised by younger, low income customers
- Cluster 2 characterised by older, high income customers
- Cluster 3 is characterised by younger, low income customers
- It appears that the separation between 1 and 3 is not good because they appear very similar...

What do the clusters look like (pt 2)?

```
ktc_numeric %>%
  #This is technically not a dataframe, so let's make it so
  as.data.frame() %>%
  # Add in the cluster allocations
  mutate(k3 = factor(km$cluster))%>%
  ggplot(data = .,
         aes (x =Age,
              y= Income,
              colour = k3))+
  geom_point()+
  facet_grid(.~Children)+
  ggthemes::scale_color_colorblind()
```

What do the clusters look like (pt 2)?

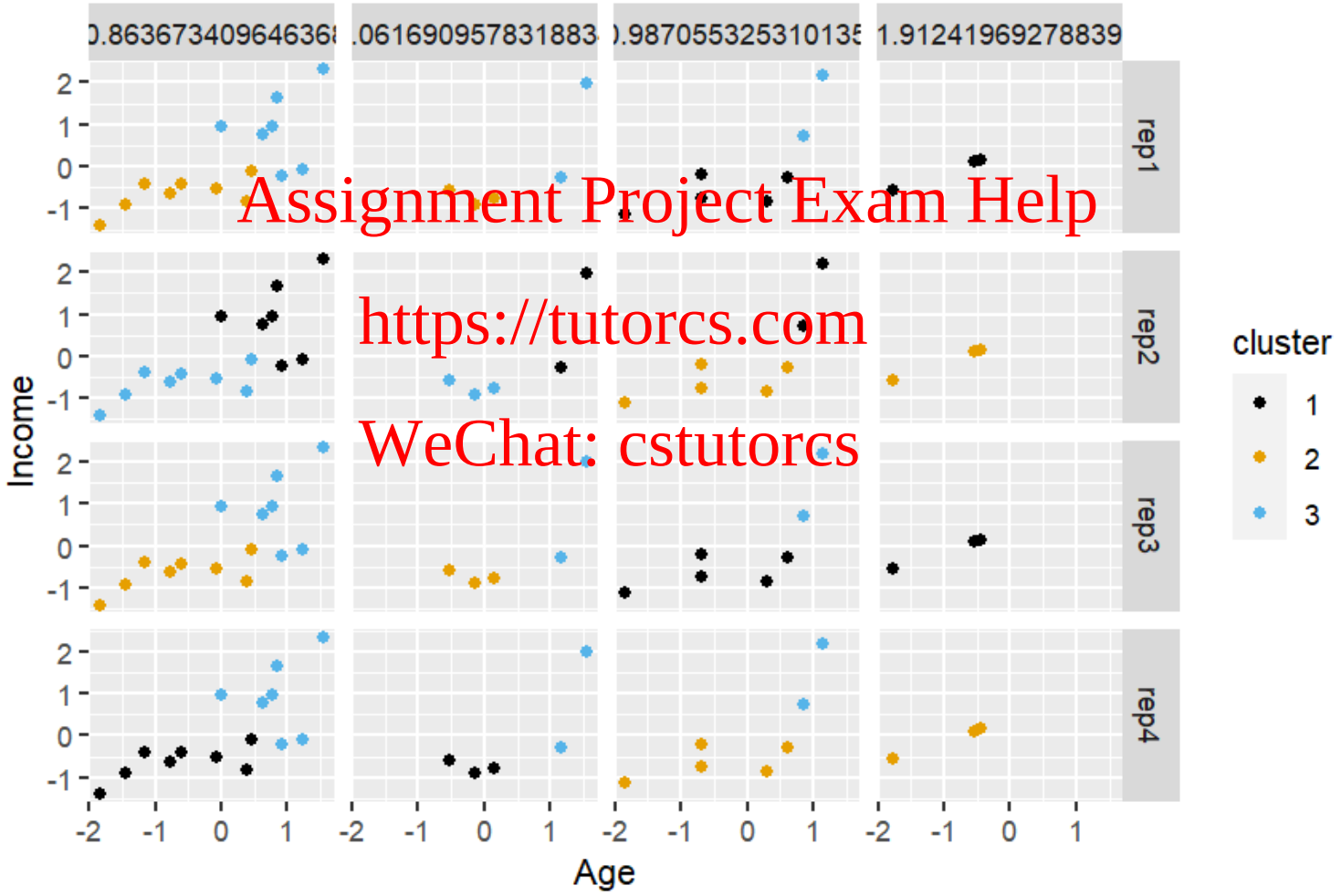


How can we summarise the clusters (pt 2)?

- Cluster 1 characterised by younger, low income customers with more children
- Cluster 2 characterised by older, high income customers with less children
- Cluster 3 is characterised by younger, low income customers with no children
- The seperation appears much better!

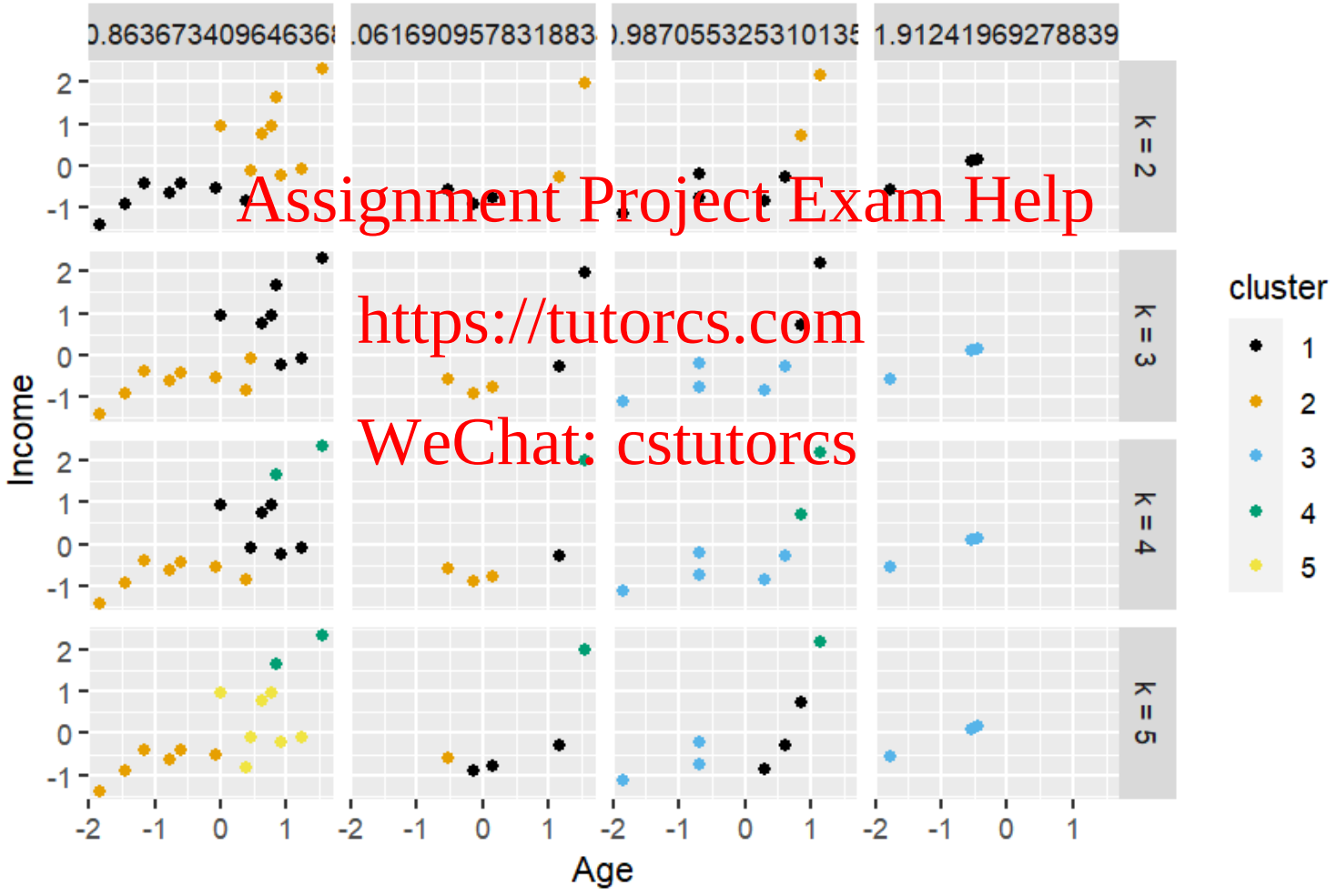
How stable is k-means?

- We can explore by running the same kmeans, with the same number of clusters a few times over:



But wait...why are we using k = 3?

- What if we tried different values of k



But wait...why are we using k = 3?

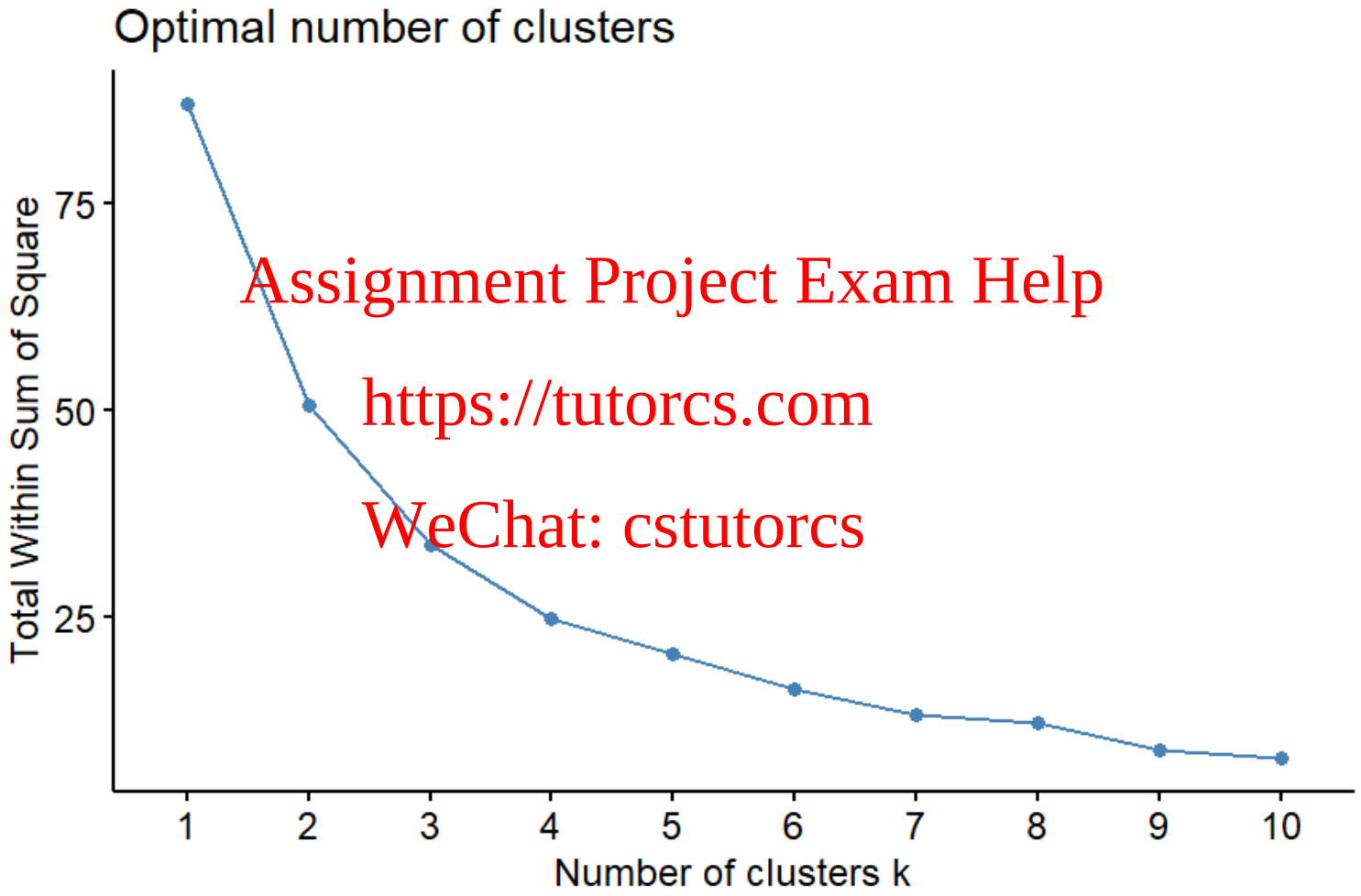
- It's difficult to tell from visual inspection which cluster is best (sometimes it is).
- Can we visualise a diagnostic of some kind instead?
- Consider:
  - The total within sum of squares. This measures the compactness (i.e., goodness) of the clustering. Smaller is better

To do this let's run for a number of different k

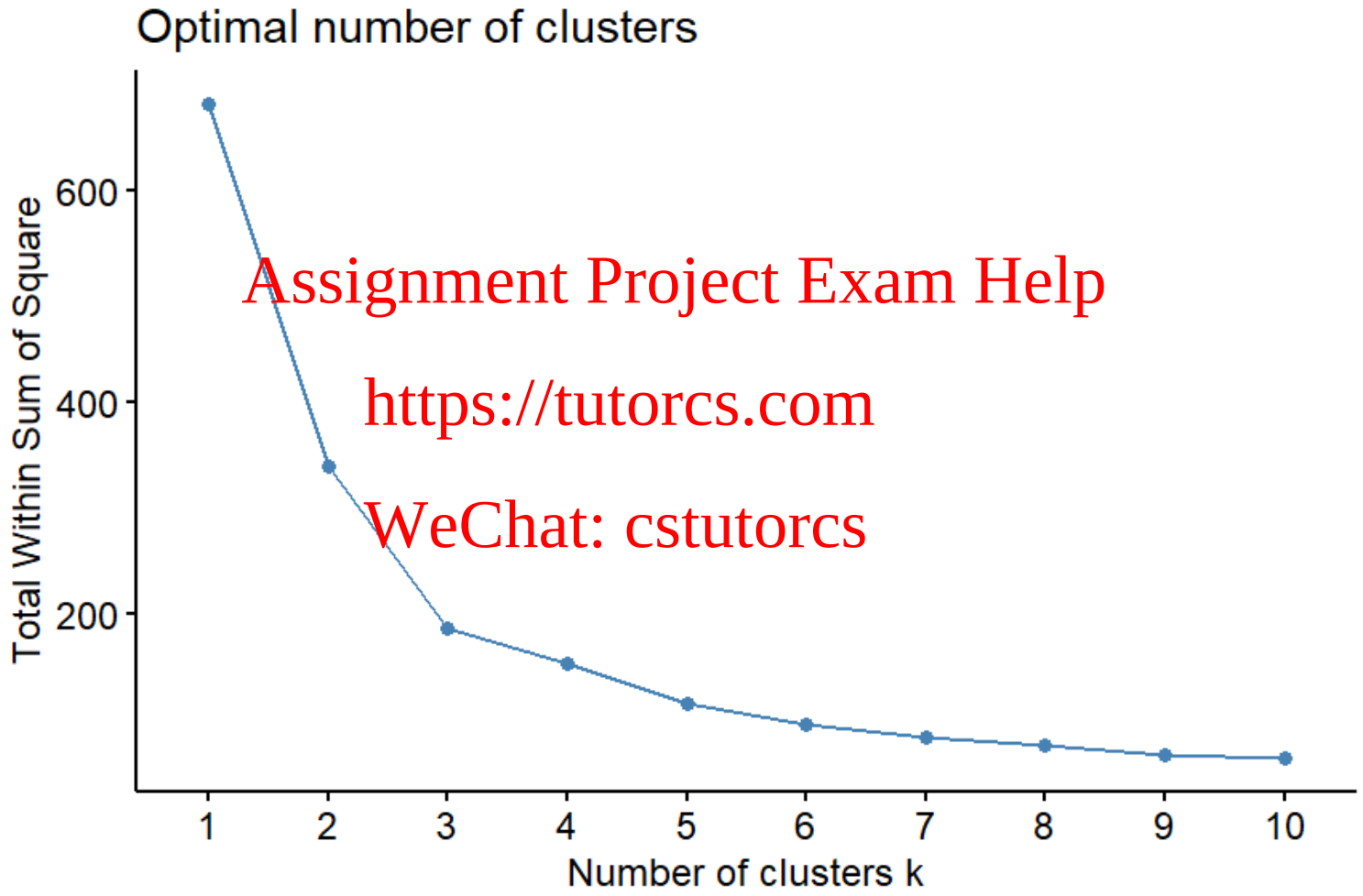
- We could do this by hand, but it is a bit of a pain
- Fortunately there's a function in the `factoextra1` package to help us!

```
library(factoextra)
#specify the dataframe, method (kmeans) and within sum of squares (wss)
fviz_nbclust(ktc_numeric, kmeans, method = "wss")
```

Elbow plot



A sharper elbow



Summary

- Unlike hierarchical clustering, kmeans requires us to specify the number of clusters directly
- kmeans is also iterative, which can mean we are less likely to get strange shaped clusters (see the snake like cluster from the previous lecture)
- Choosing k can be difficult, but we can use the Elbow plot to help to understand what number is best.

MONASH University

Lecturer: *Lauren Kennedy*

Department of Econometrics and Business Statistics

lauren.kennedy1@monash.edu

Week 8