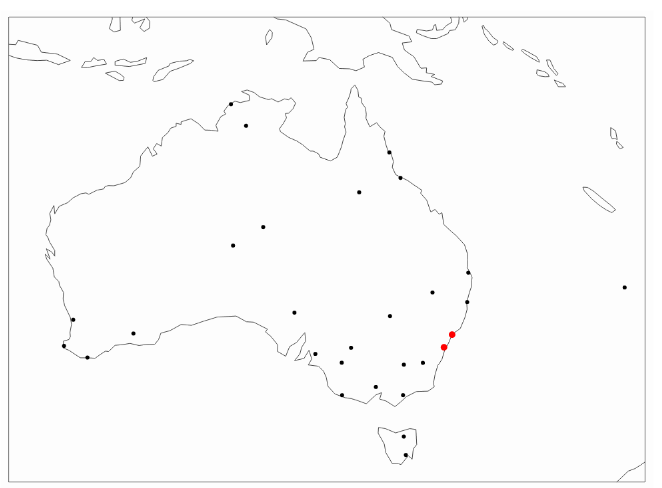
**HAC: Hierarchical Agglomerative Clustering — Is It Better Than K-Means?**

# Types of clustering algorithms

Not all clustering algorithms are created equal. Different clustering algorithms implement different ideas on how to best cluster your data. There are 4 main categories:

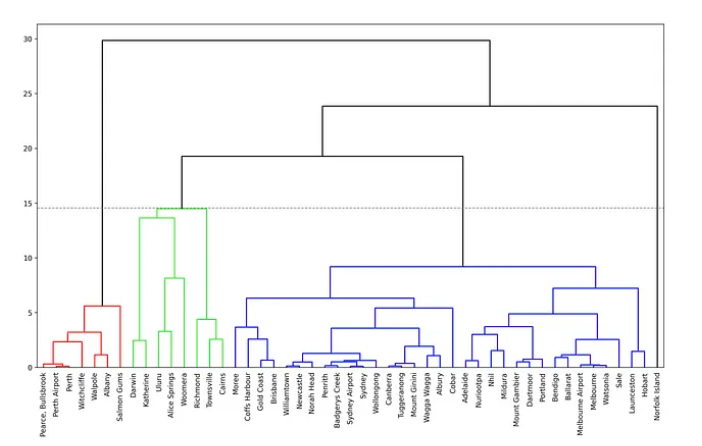
* **Centroid-based** — uses Euclidean distance to assign every point to the nearest cluster center. Example: [K-Means](https://towardsdatascience.com/k-means-clustering-a-comprehensive-guide-to-its-successful-use-in-python-c3893957667d)
* **Connectivity-based**— assumes that nearby objects (data points) are more related than far away objects. Example: [Hierarchical Agglomerative Clustering (HAC)](https://towardsdatascience.com/hac-hierarchical-agglomerative-clustering-is-it-better-than-k-means-4ff6f459e390).
* **Density-based** — defines clusters as dense regions of space separated by low-density regions. Example: Density-Based Spatial Clustering of Applications with Noise (DBSCAN).
* **Distribution-based**— assumes the existence of a specified number of distributions within the data. Each distribution with its own mean (μ) and variance (σ²) / covariance (Cov). Example: [Gaussian Mixture Models (GMM)](https://towardsdatascience.com/gmm-gaussian-mixture-models-how-to-successfully-use-it-to-cluster-your-data-891dc8ac058f).
* HAC is not as well-known as K-Means, but it is quite flexible and often easier to interpret. It uses a “bottom-up” approach, which means that each observation starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy.
* To help illustrate this, I have created a few graphs. The first one demonstrates how the algorithm merges closest points one step at a time until there is only one cluster remaining.

****

****

Note, the above implementation of HAC uses ‘average’ linkage, which I will explain later in this section.

The second graph is known as a **Dendrogram**. It gives the full picture of the path taken, moving from all individual points (bottom of the graph) to one single point/cluster (top of the graph):

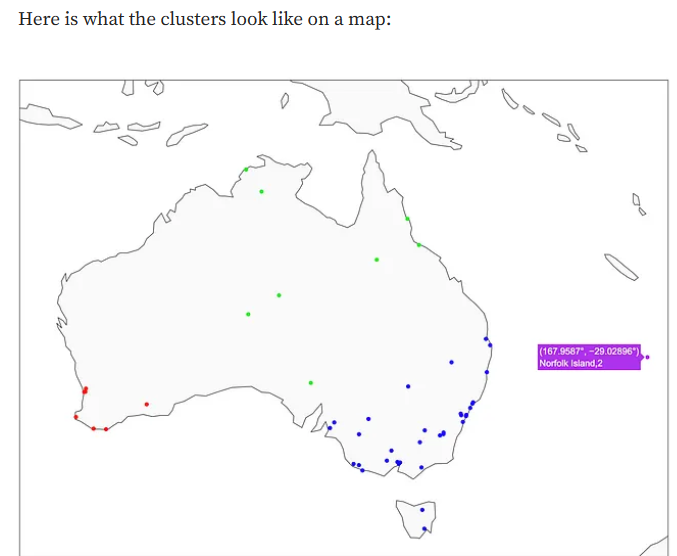


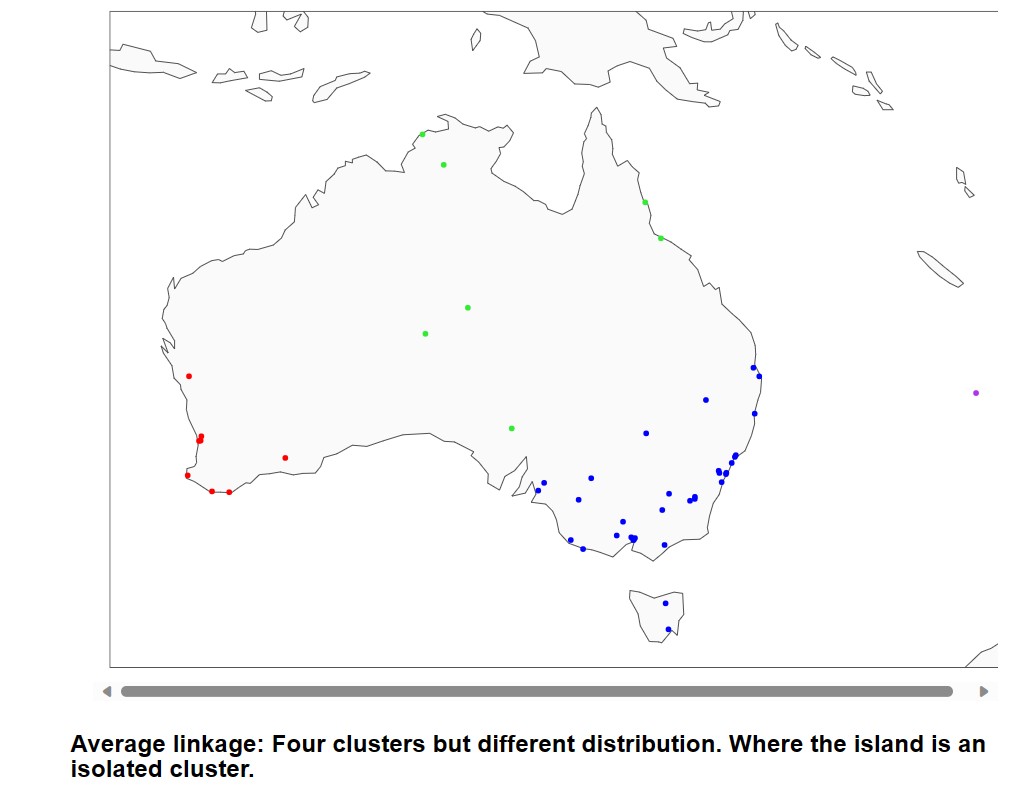
Note, I have added a dotted horizontal line to indicate the number of clusters I have selected. In general, a good rule of thumb is to identify the largest section within the y-axis where you do not have vertical lines intersected by any horizontal lines. Let me explain further.

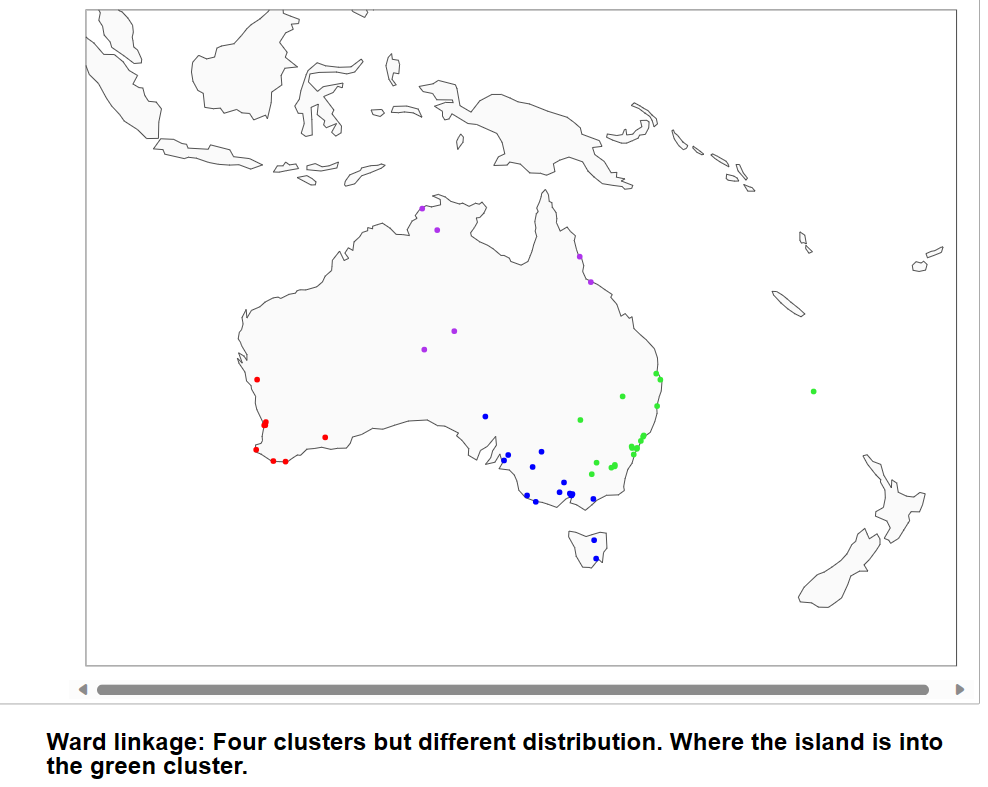
The value on the y-axis is the measure of closeness between points/clusters. The horizontal lines denote places where those points/clusters are being merged.

The above graph indicates that you may want to consider having 2 clusters since the largest distance with no cluster mergers is between 24 and 30.

However, in this case, I decided to choose 4 clusters instead, which is a slightly less optimal solution. Hence, my clusters are red, bright green, blue, and Norfolk Island that sits in the corner on its own. Horizontal line 19.5.  Hence, my clusters are red, bright green, blue, and Norfolk Island that sits in the corner on its own.







## Linkage types

There are multiple ways to link the points together. I used ‘average’ linkage in the above example since it is easy to illustrate and understand. However, it is important to familiarize yourself with other linkage types too. You can choose in Sklearn what measure to use:

**‘Average’:** uses the average of the distances of each observation of the two sets, i.e., finds the mid-point between observations (as shown in the graphs).

* **‘Single’:** uses the minimum of the distances between all observations of the two sets, i.e., looks for the closest point within the cluster of points at that stage (instead of cluster mid-point used in ‘average’).
* **‘Complete’ or ‘Maximum’:**uses the maximum distance between all observations of the two sets. E.g., if the point is closer to the farthest point of Cluster A than the farthest point of Cluster B, then such a point would be added to Cluster A.
* **‘Ward’:** minimizes the variance of the clusters being merged. This is very similar to minimizing Within Cluster Sum of Squares (WCSS) used by K-Means.

‘Ward’ linkage is typically the one that is used most often. It is also the default option for sklearn’s implementation of HAC.

## **HAC advantage over K-Means clustering**

The attractive part of using HAC is the ability to analyze dendrogram, which provides an insight into the similarity level between any two data points.

It only takes one glance at a dendrogram to see that Pearce, Bullsbrook is very ‘distant’ from Norfolk Island, while Sydney and Sydney Airport are quite close together. While this is fairly obvious given our location data, it becomes very beneficial when we use more abstract attributes.

In general, you can generate very similar clusters using HAC and K-Means, but having the ability to see the path of each data point, makes HAC more useful in analyzing similarities and differences between individual data points or clusters.

Steps to create a clusters:

1. At the start, treat each data point as one cluster. Therefore, the number of clusters at the start will be K - while K is an integer representing the number of data points.
2. Form a cluster by joining the two closest data points resulting in K-1 clusters.
3. Form more clusters by joining the two closest clusters resulting in K-2 clusters.
4. Repeat the above three steps until one big cluster is formed.

Parameters:

https://scikit-learn.org/0.21/modules/generated/sklearn.cluster.AgglomerativeClustering.html