

# Introduction to clustering

- *k*-means and agglomerative clustering
- Clustering with big data on the NCI

Sonya Fiddes

With help from Kate Saunders &  
acknowledgments to Acacia Pepler &  
Pandora Hope



The Australian Antarctic Program Partnership is led by the University of Tasmania, and includes the following partner agencies



The Australian Antarctic Program Partnership is funded by the Australian Government Department of Industry, Science, Energy and Resources through the Antarctic Sciences Collaboration Initiative.



Australian Government  
Bureau of Meteorology



Australian Government  
Department of Agriculture, Water and the Environment  
Antarctic Antarctic Division

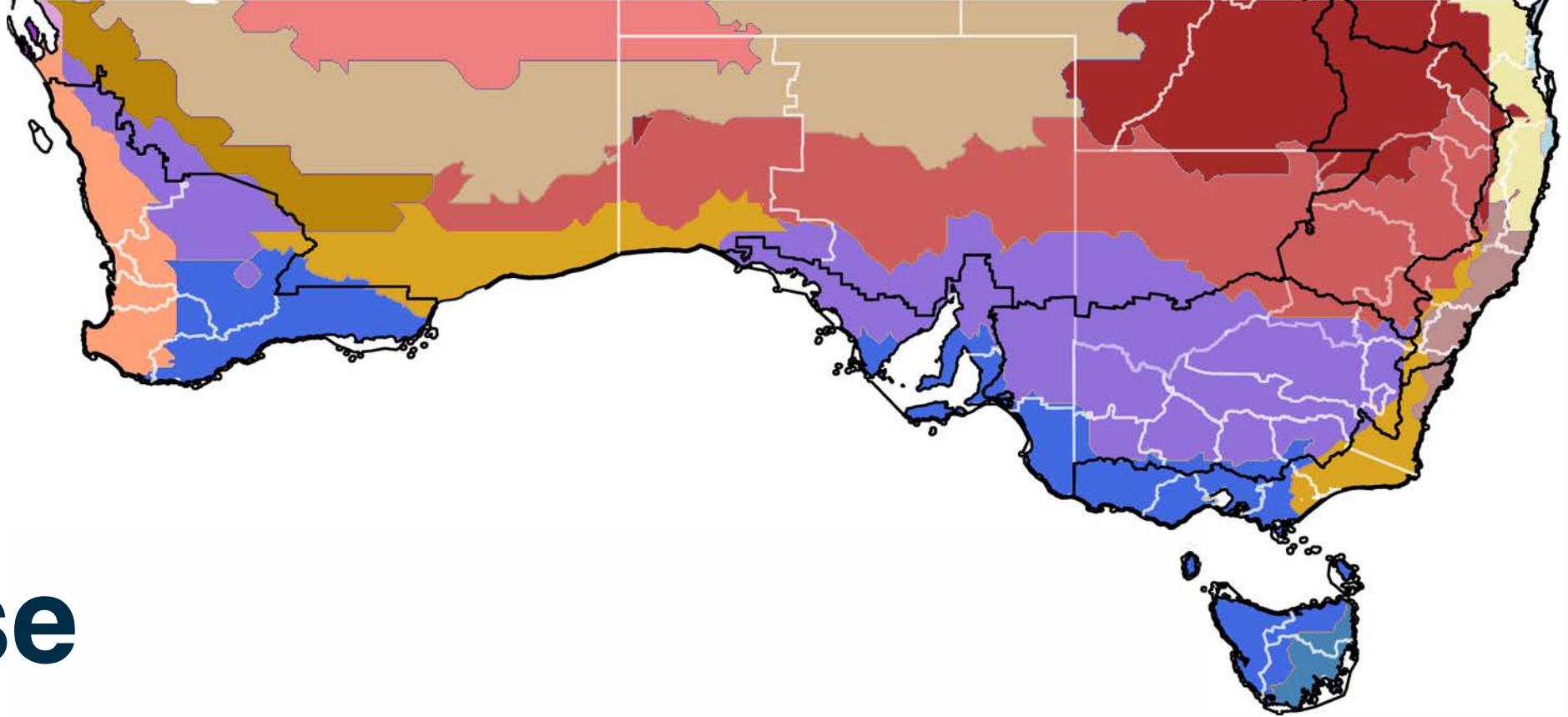


Australian Government  
Geoscience Australia



Tasmanian Government

# Why use clustering?



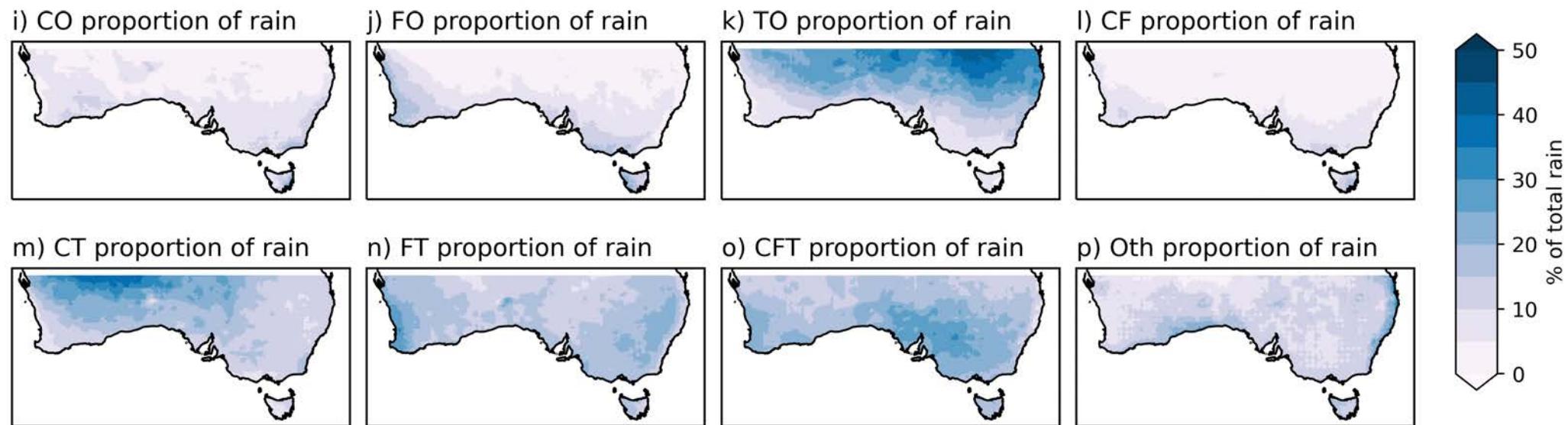
**Want to organise data into ‘clusters’ that are internally (within cluster) similar and externally (between cluster) different**

**Do not know what these clusters look like**

**Unsupervised machine learning**

**Example: Fiddes et al. (2021) wanted to know where different climate regions existed across Southern Australia, based on the proportion of rain produced by weather systems.**

**We did not know where these boundaries would fall, or how many clusters there should be.**



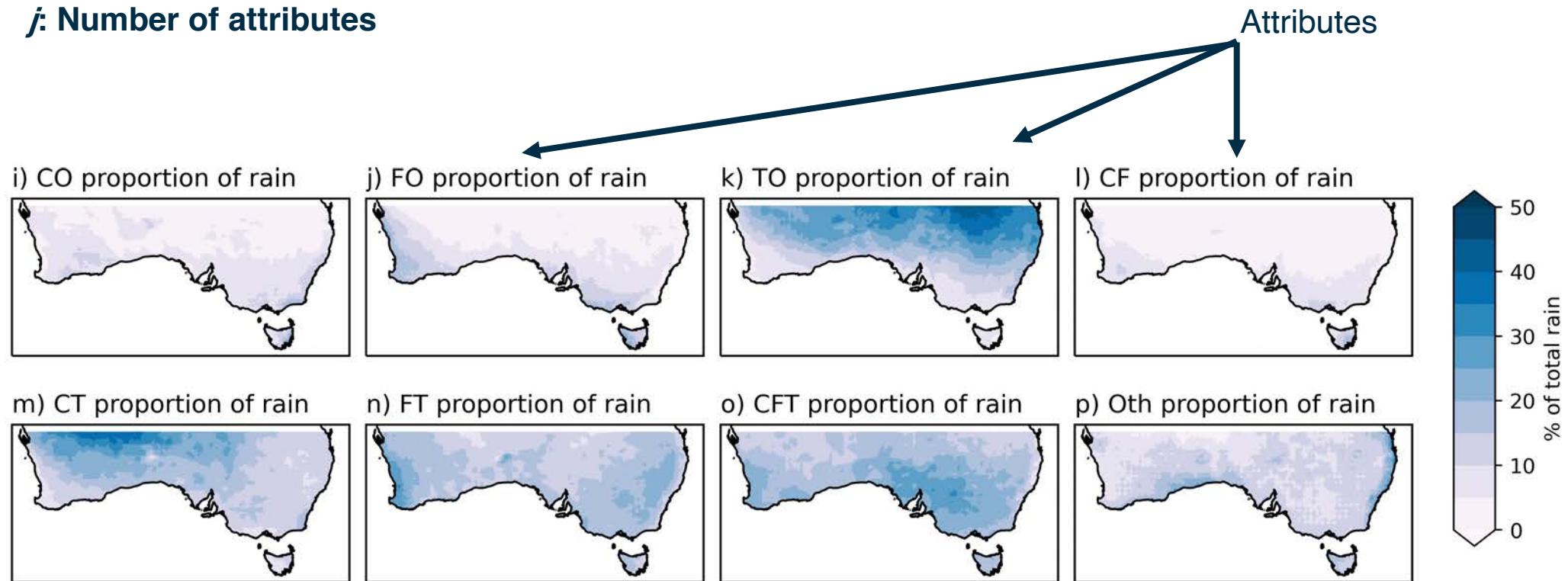
# Some terminology:

*n* : Number of clusters

*i* : Number of observations

*j* : Number of attributes

In this case:  
 $i$  = number of grid points  
 $j = 8$



# How do we define similar?

**The majority of clustering algorithms use a ‘dissimilarity matrix’  $D$ :  
the sum of attribute dissimilarity  $d_j$  between observations  $i$  and  $i'$**

$$D(x_i, x_{i'}) = \sum_{j=1}^p d_j(x_{ij}, x_{i'j})$$

**The most common measure of dissimilarity is the squared distance (but there are a number of other options)**

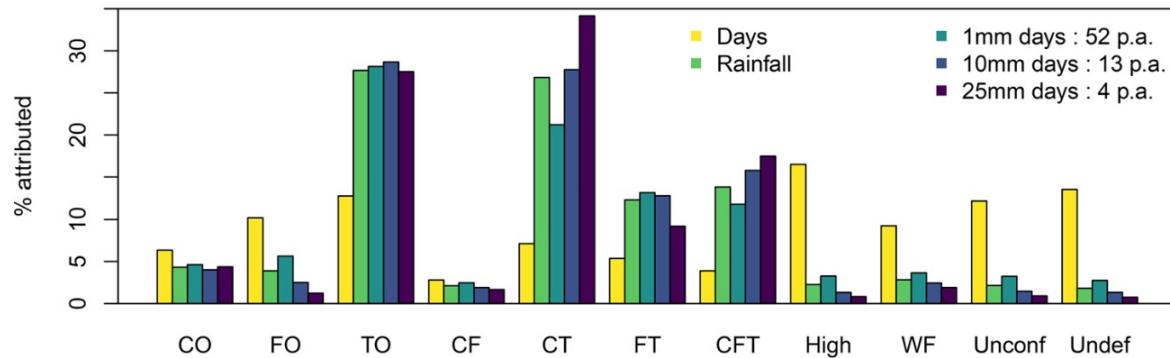
$$d_j(x_{ij}, x_{i'j}) = (x_{ij} - x_{i'j})^2$$

# Are all attributes equal?

**No. Most commonly, clustering uses the Euclidean distance (there are other options).**

**If one attribute (and its dissimilarities) is larger in magnitude than another, this attribute will contribute more to the clustering decisions.**

**Fiddes et al. (2021) was interested in how the relationship of rainfall and weather types influence climate. Rainfall and days exist on different scales (note plot is showing %)**



**Fig. 6** Annual percentage contribution of each weather system type to rainfall in the  $0.05^{\circ}$  AWAP gridded analyses between 1979 and 2015, averaged across all land areas of Australia south of  $25^{\circ}$  S. Bars show the percentage of all days influenced by each weather type (days), the

total annual precipitation from each type (rainfall), and the proportion of all days with rainfall exceeding a given threshold (1 mm, 10 mm and 25 mm) that are associated with a given weather type. Legend shows the average number of days p.a. for each rainfall threshold

# Data processing

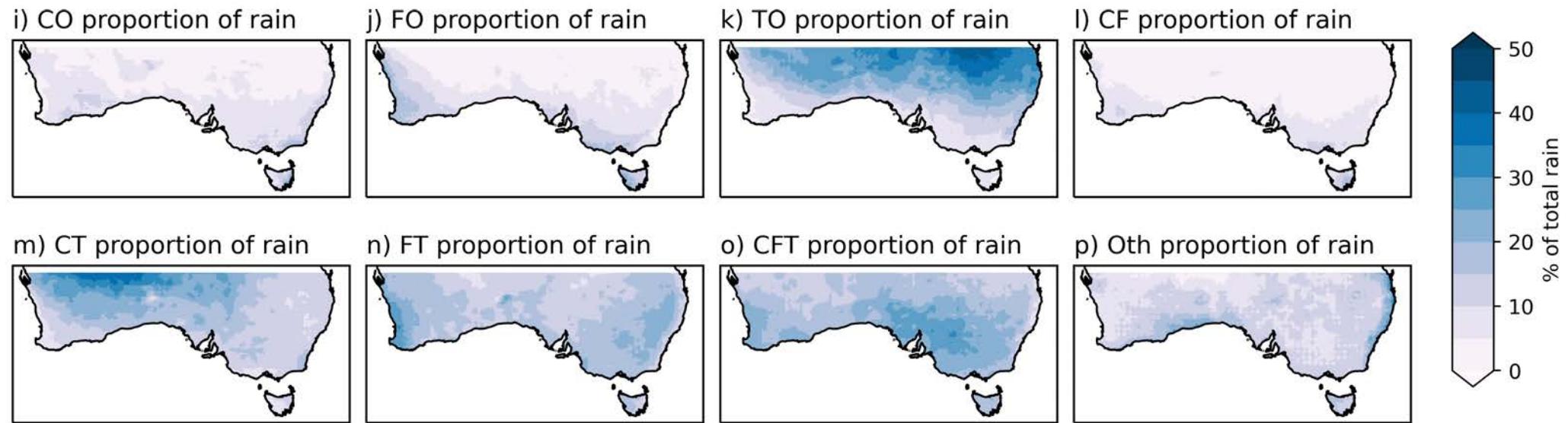
**In order to ensure your attributes contribute equally to the clustering algorithm, you can:**

- **Weight them ( $w$ )**
- **Normalise them**
- **Choose different attributes**
- **Use another method to reduce data ‘dimensionality’ eg. Principal Component Analysis**

$$D(x_i, x_{i'}) = \sum_{j=1}^p w_j \cdot d_j(x_{ij}, x_{i'j});$$

$$\sum_{j=1}^p w_j = 1.$$

In Fiddes et al. (2021), we chose to cluster on the proportion of rain brought by each weather type, instead of using total rainfall amount or number of days for each weather type

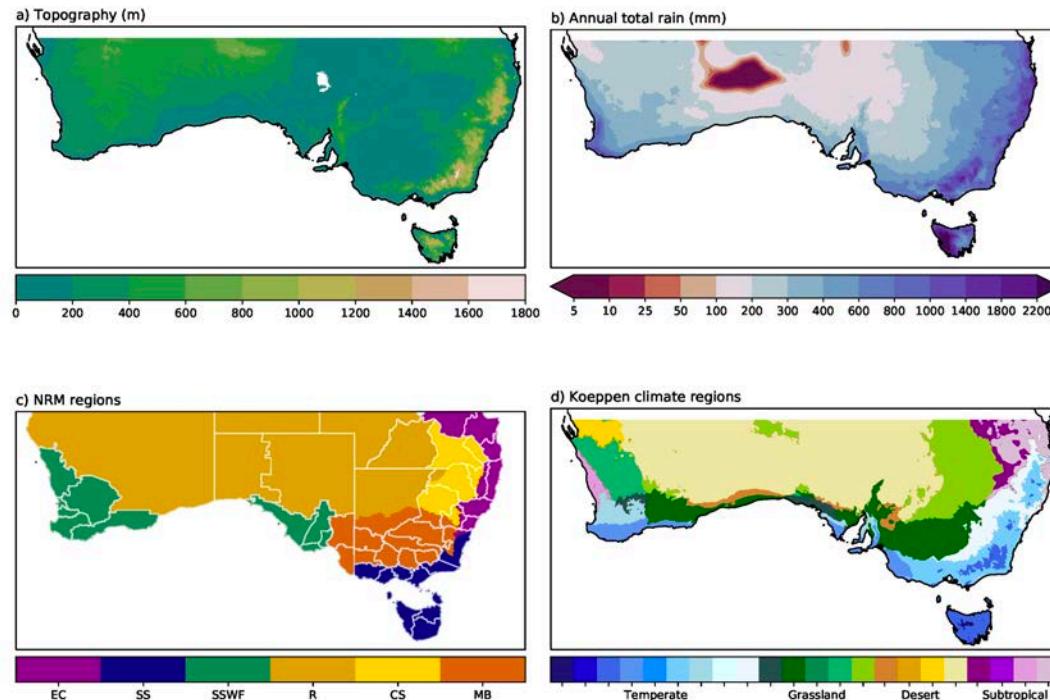


# How many clusters?

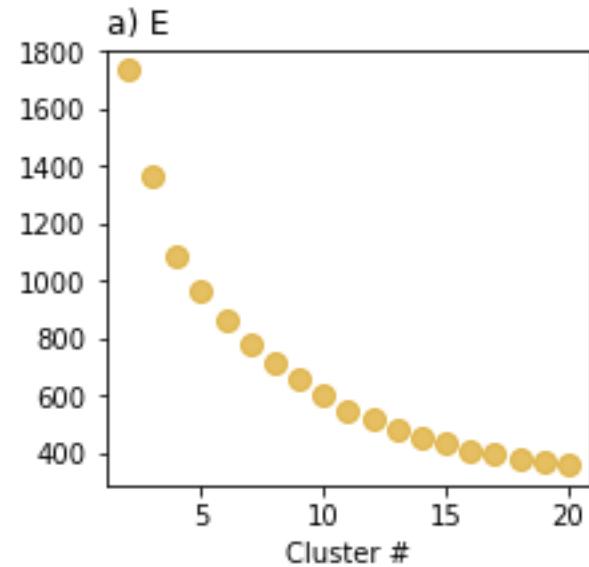


**Ultimately, if you do not know the underlying structure of your data, *there is no 'right' or 'wrong' answer.***

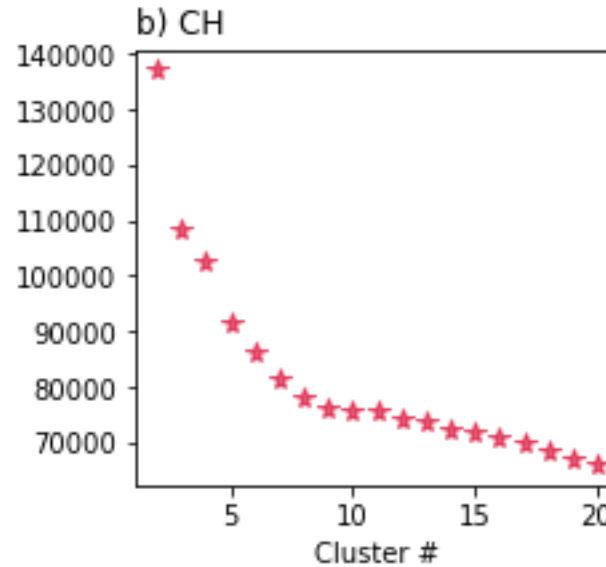
**You need to understand the physical meaning of your data to help make this decision.**



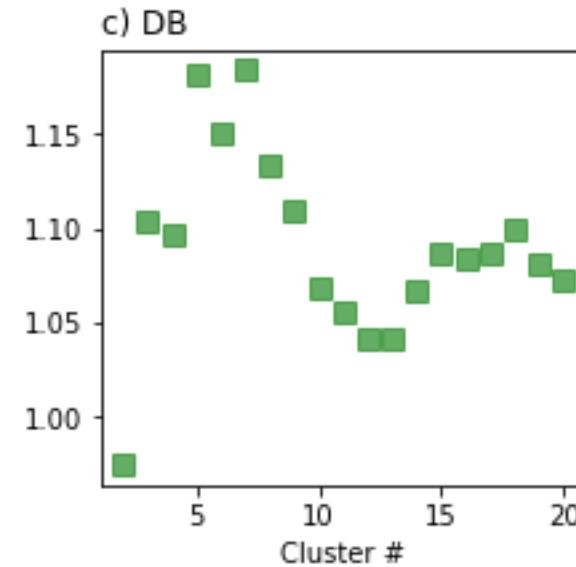
**There are some metrics to help, but be warned: often they don't.**



**Elbow plots – find the elbow.**



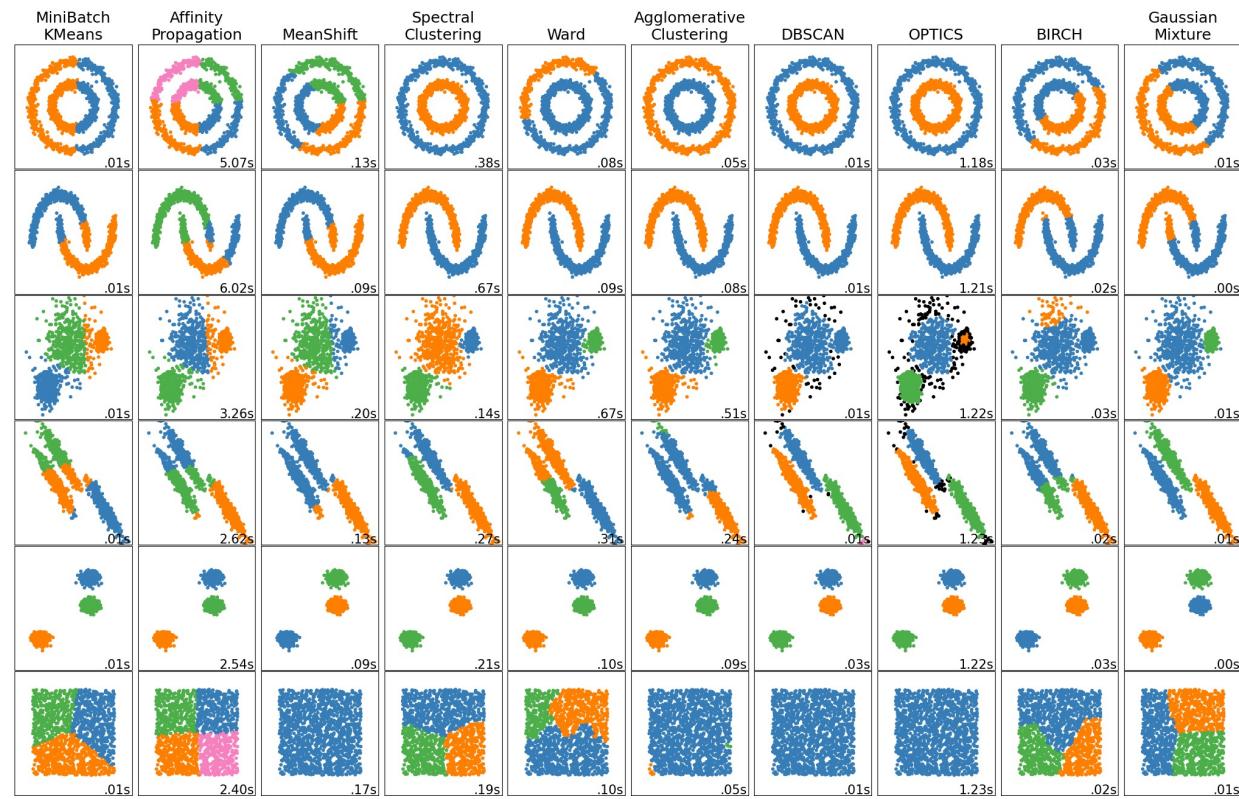
**Calinski-Harabasz score – higher score: better defined clusters**



**Davies-Bouldin index – lower score: clusters with least similarity**

# Types of clustering

## There are many types of clustering available



# *K*-means

**Uses the squared Euclidean distance as it's dissimilarity metric**

**Minimizes with-in cluster variance**

- 1. Randomly assign observations to clusters**
- 2. Work out the cluster means (cluster centres)**
- 3. Update assignment of observations to the nearest cluster centre\***

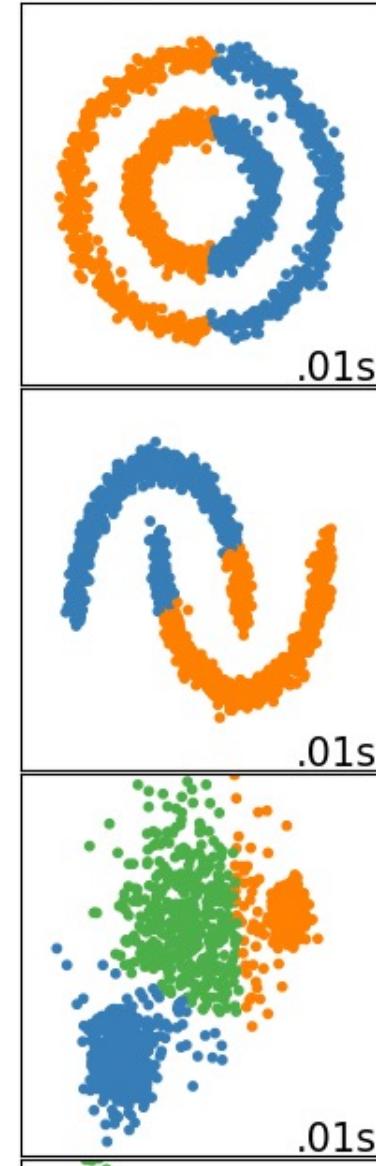
**\* Repeat 2-3 until the change in cluster means is very small (a given threshold) i.e  
the assignments do not change**

# Assumptions

**Clusters are spatially grouped (spherically)**

**Assumes that each cluster has a roughly equal number of observations (ie. equal probability)**

**Does not like ‘high-dimensional’ spaces (because it uses the Euclidean distance). I.e. attributes have the same variance**



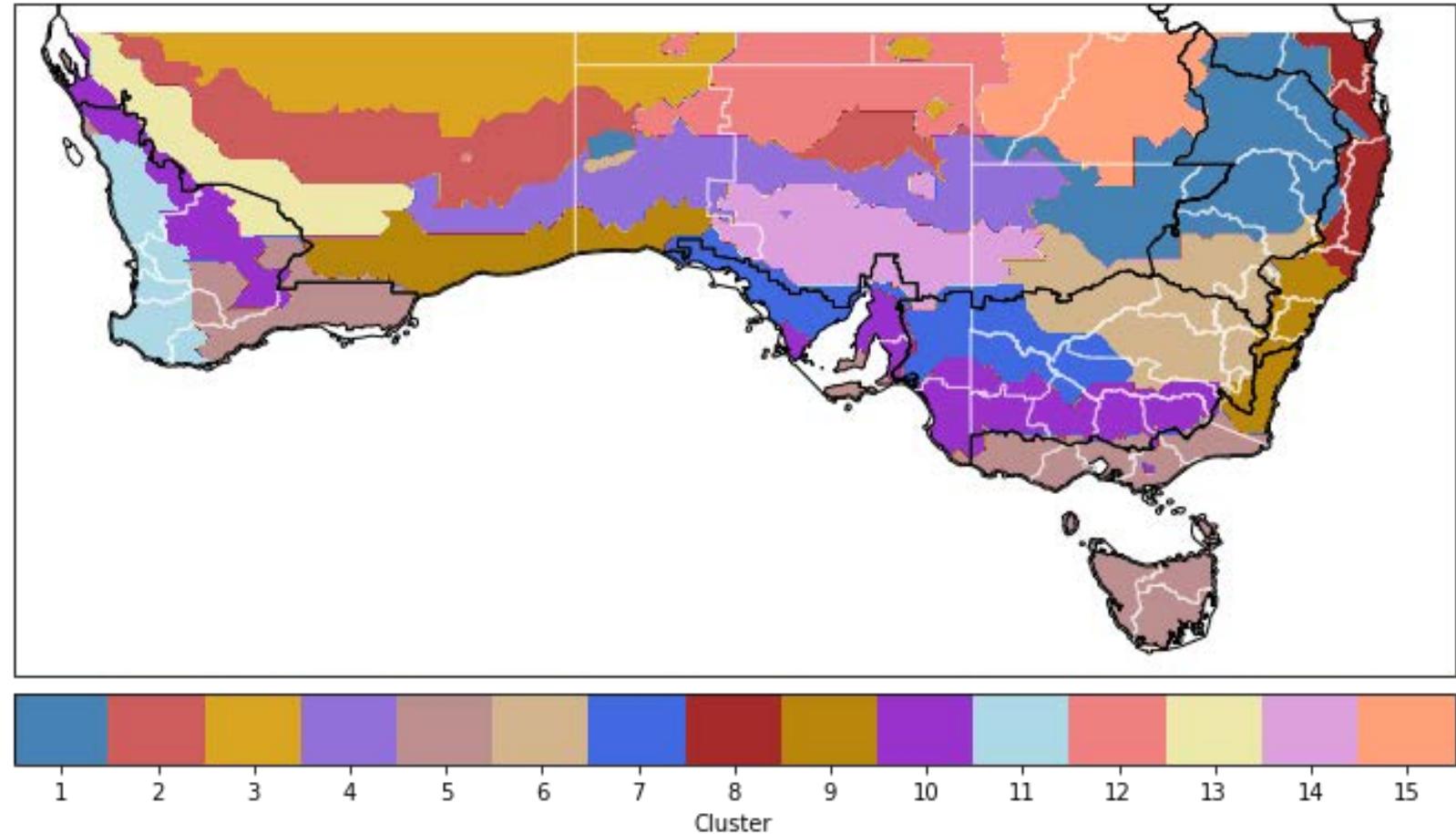
### **Pros:**

- **Scales well to very large data sets**
- **Easy to implement, highly flexible and efficient**
- **Will always converge**
- **Observations are not tied to a cluster (i.e can move depending on centroid means).**

### **Cons:**

- **Does not handle outliers well**
- **Assumptions of even cluster size & attribute variance**
- **Depends on initial values (i.e first cluster assignment)**

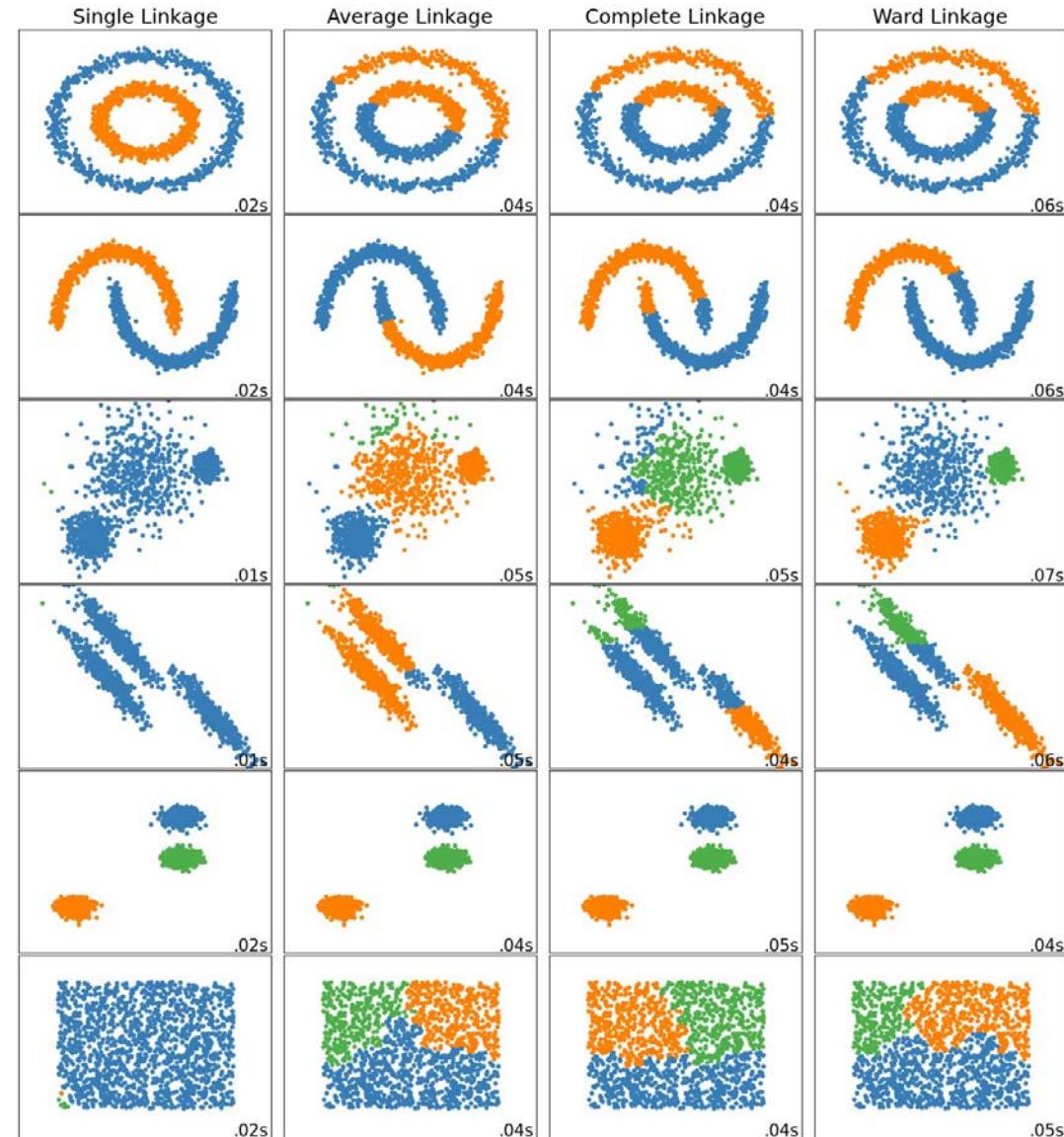
a) Annual clusters



# Agglomerative clustering

**From Sci-Kit learn: four forms of agglomerative clustering available, each using a different ‘linkage’ method (ie. the dissimilarity metric they minimize):**

- **Ward:** Sum of squares distance (similar to  $k$ -means)
- **Single:** Maximum distance between observations and pairs of clusters
- **Average:** Average distance between all observations and pairs of clusters
- **Complete:** distance between closest observations of pairs of clusters



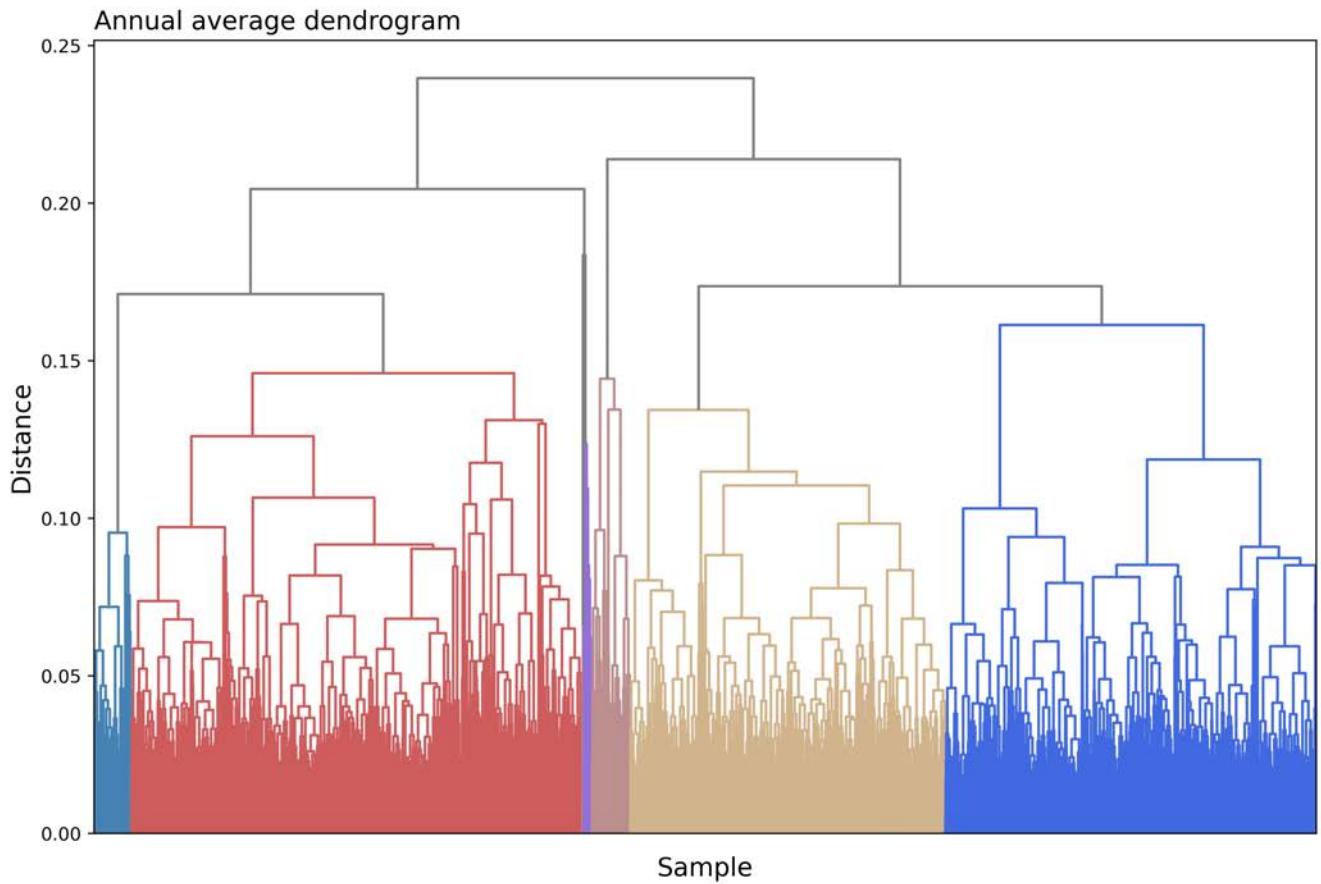
**1. Merge clusters with smallest dissimilarity**

**2. Work out new dissimilarities for new clusters\***

**\* Repeat 1-2 until all observations are merged into a single cluster, or until the desired number of clusters is reached**

**Unlike  $k$ -means, we can gain some idea of the ‘structure’ of the data before hand**

**The dendrogram shows us when/how clusters are merged.**



## Pros:

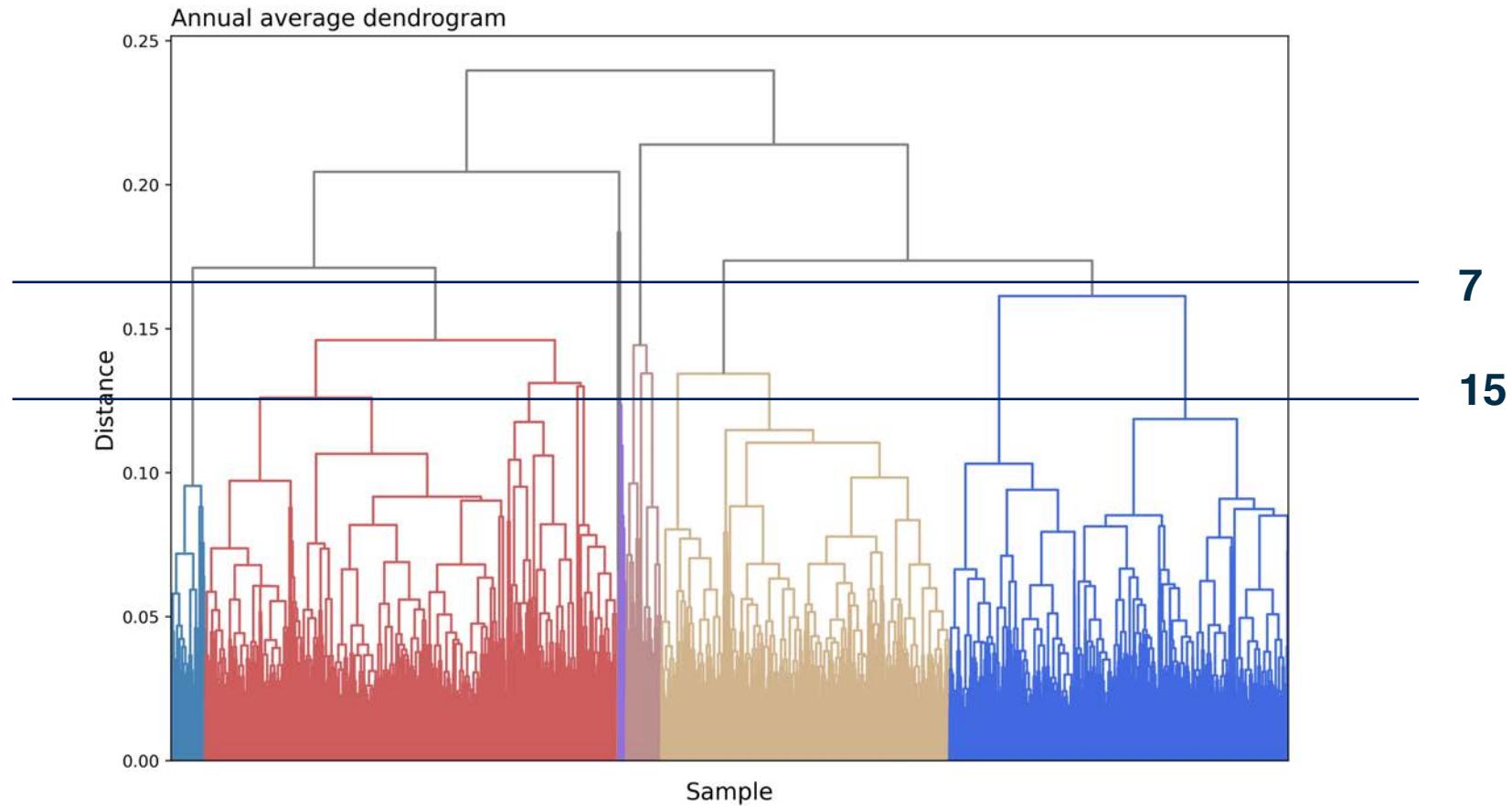
- Scales well to large data sets and large number of clusters
- Does not require clusters to have similar size
- A number of ways to ‘link’ clusters (each with their own pros and cons)
- We can learn about the data structure prior to choosing number of clusters (but ultimately still have to choose number of clusters)

## Cons:

- Less flexible, once groups of observations are assigned to a cluster, they will stay in that cluster
- Less efficient

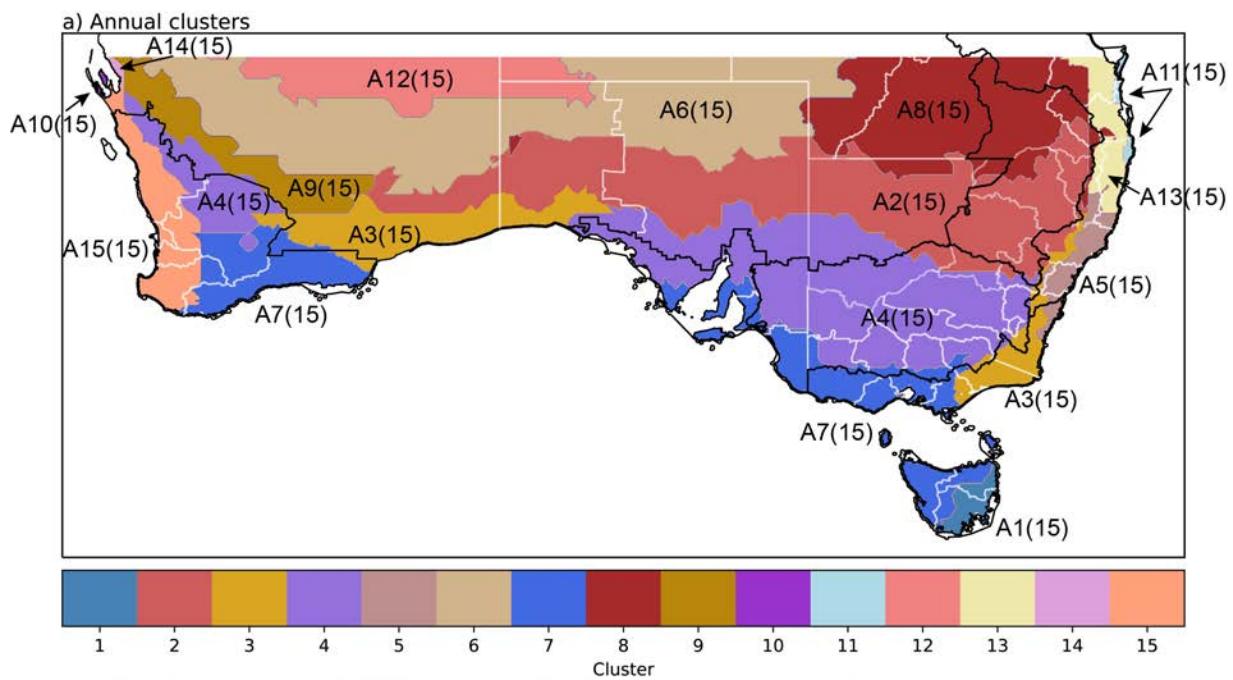
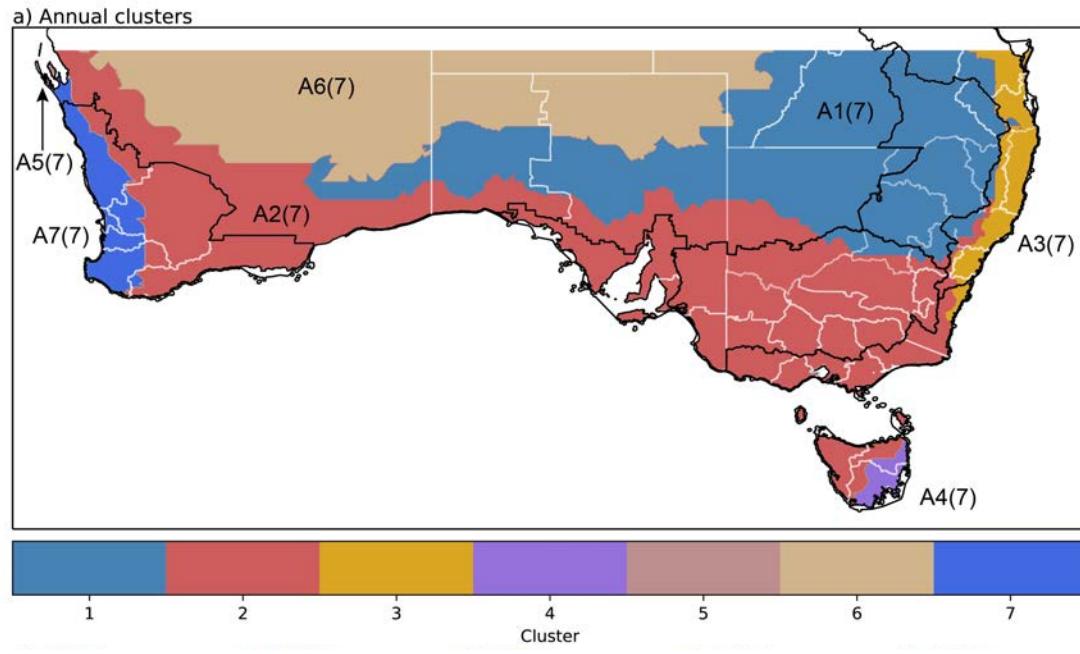
## Average linkage:

We chose 7 and 15 to test, in part based on the dendrogram, and in part due to our knowledge of the system



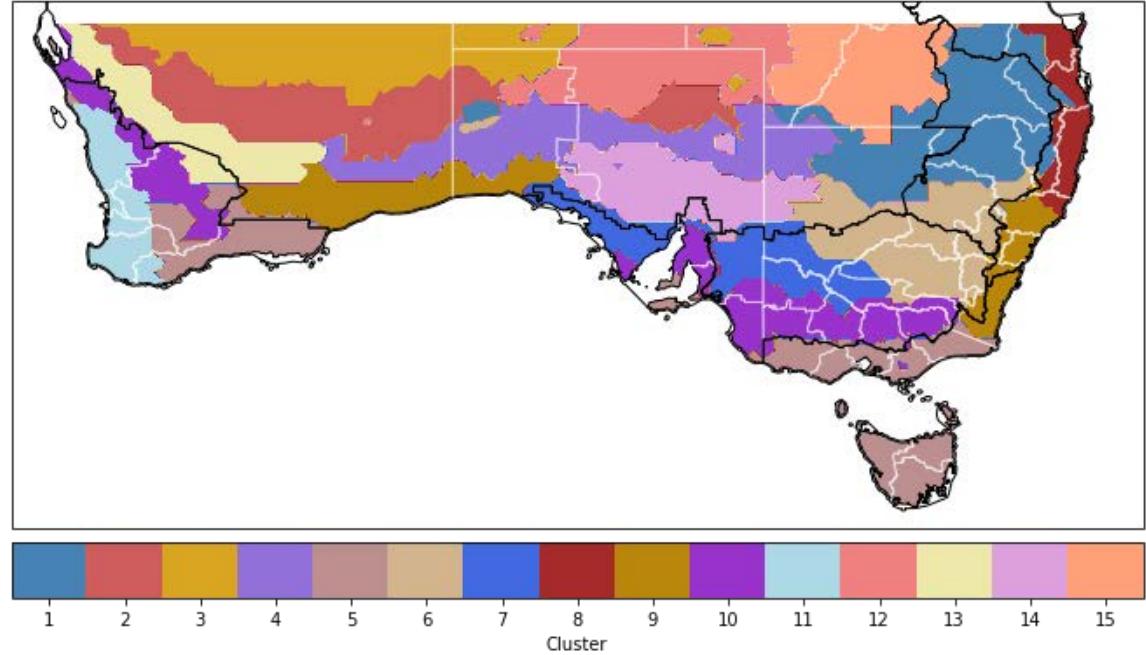
## Average linkage:

Seven clusters didn't capture the differentiations were were after in SE Aus, so 15 clusters were also produced.



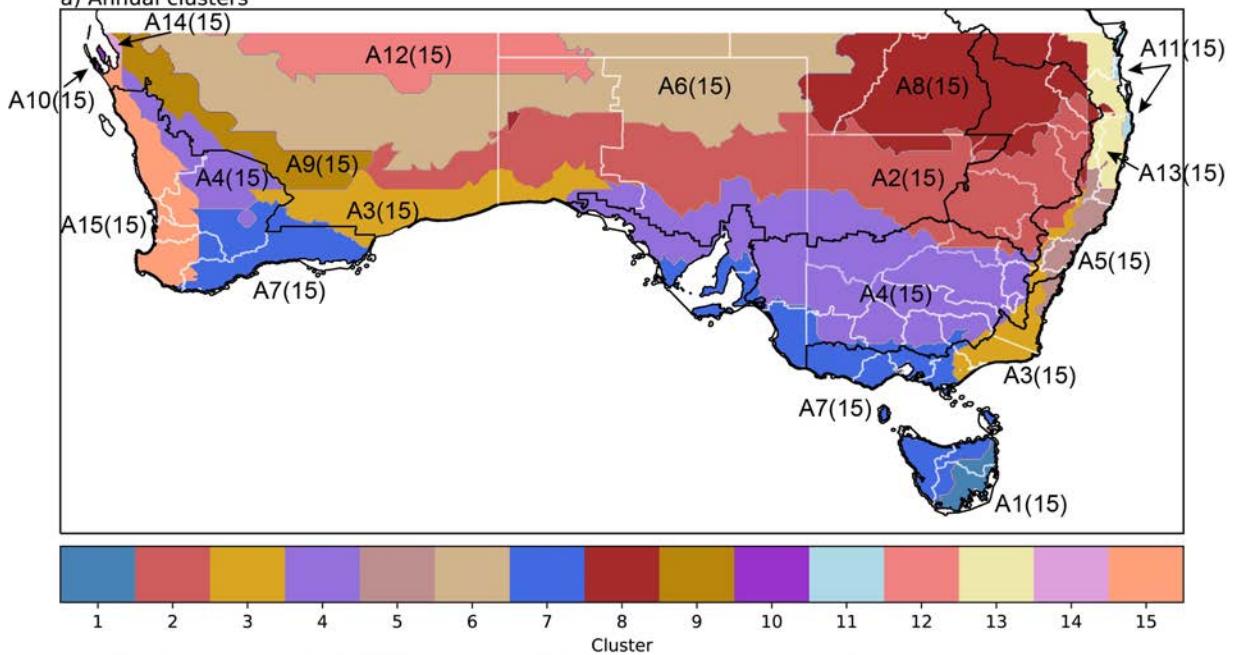
## *K*-means

a) Annual clusters



## Average

a) Annual clusters

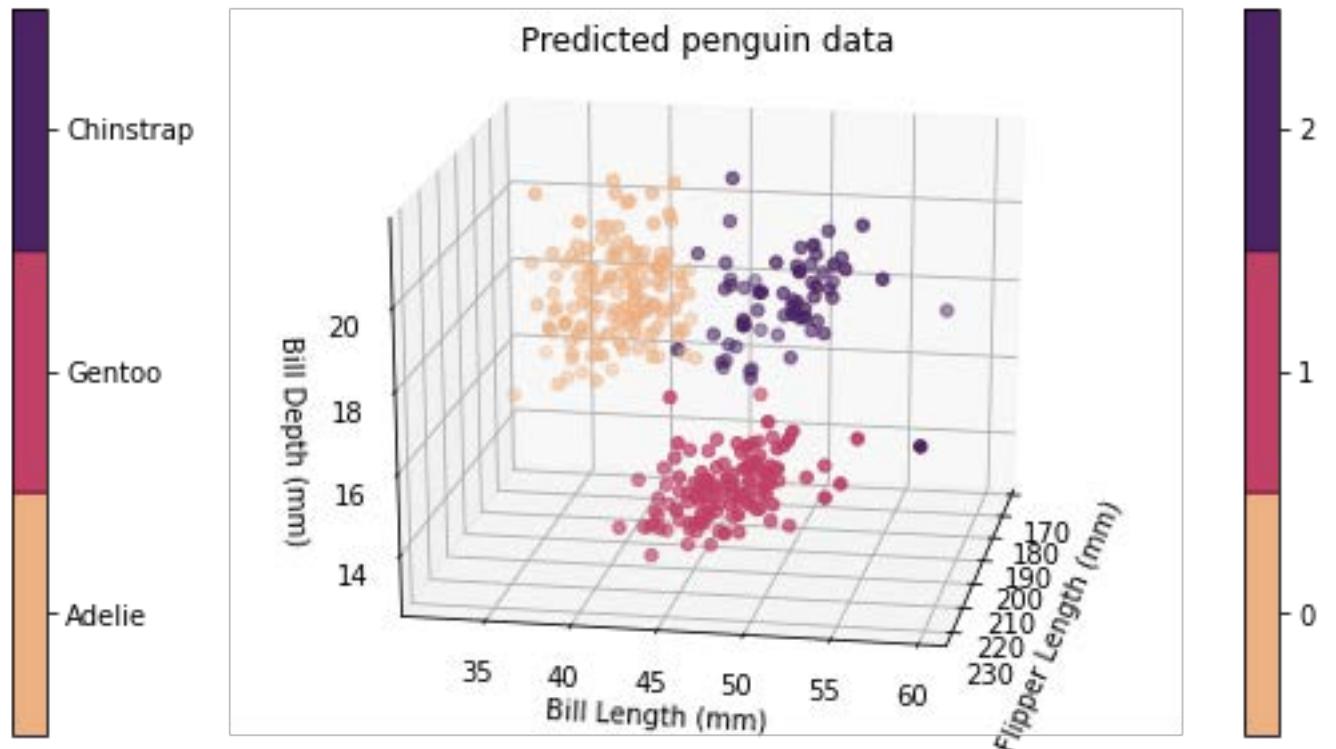
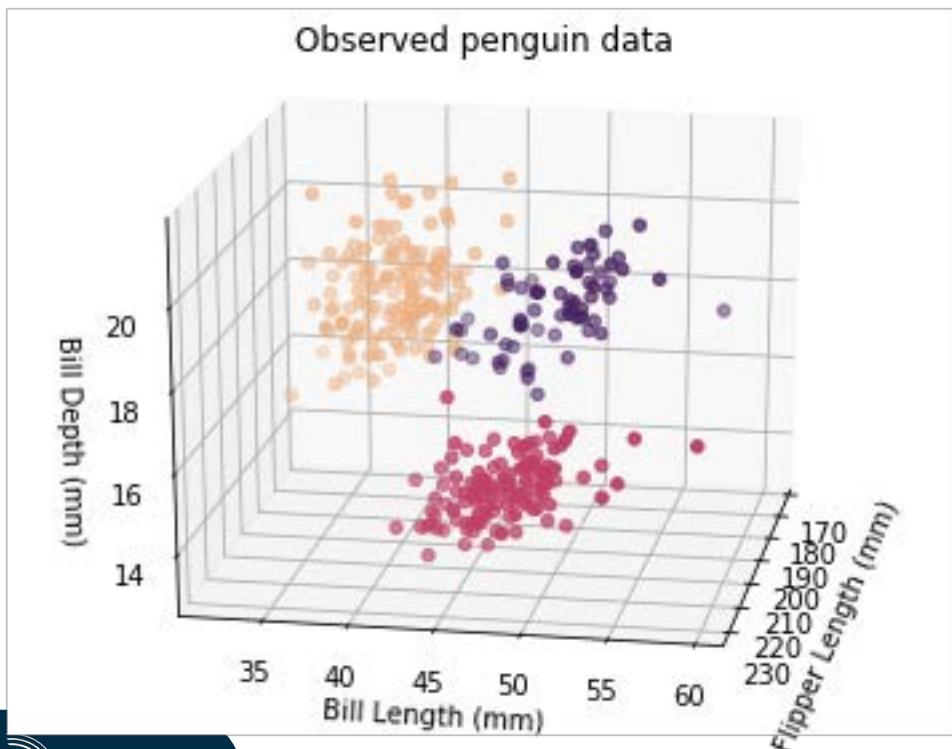


# Tutorial

[https://github.com/sfiddes/teaching/blob/main/2021\\_08\\_ML\\_workshop/Clustering101.ipynb](https://github.com/sfiddes/teaching/blob/main/2021_08_ML_workshop/Clustering101.ipynb)

**Use the Palmer Penguins data set and SK-Learn to perform k-means and agglomerative clustering.**

Note in this instance, we already know the structure of our data, so our lives are *a lot* easier.

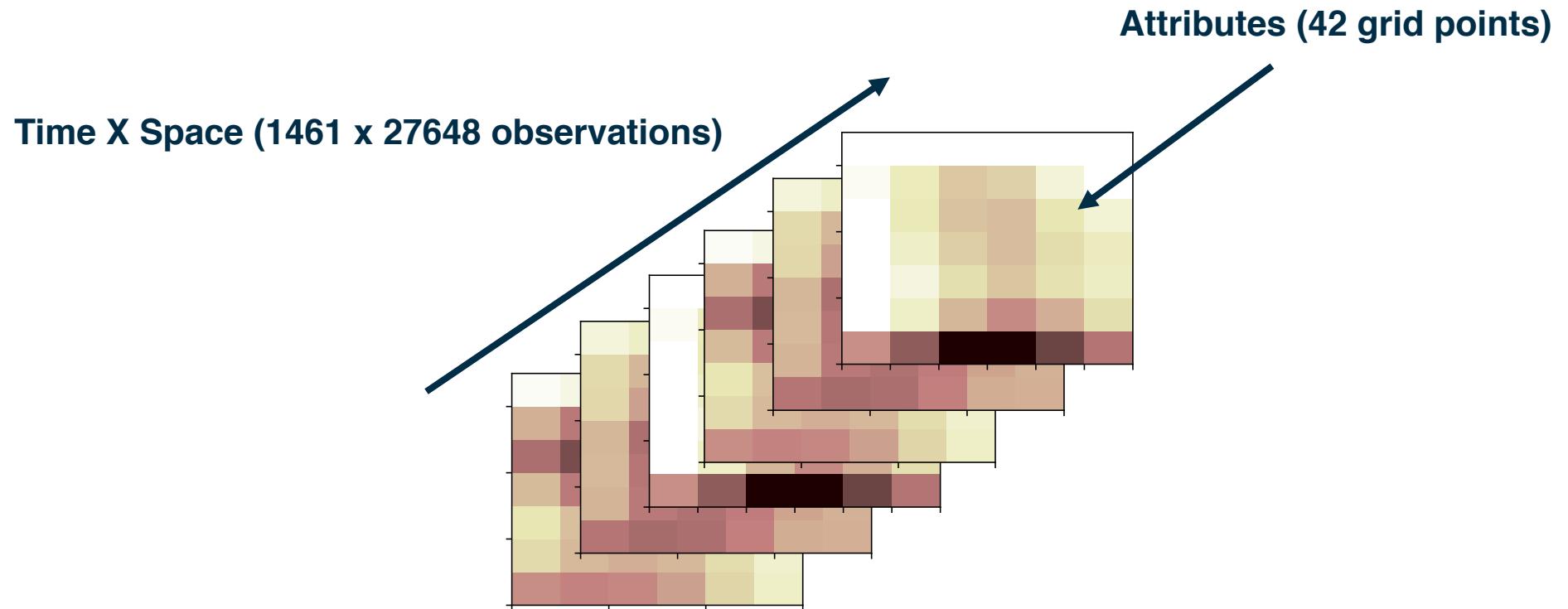


# Clustering with big data sets

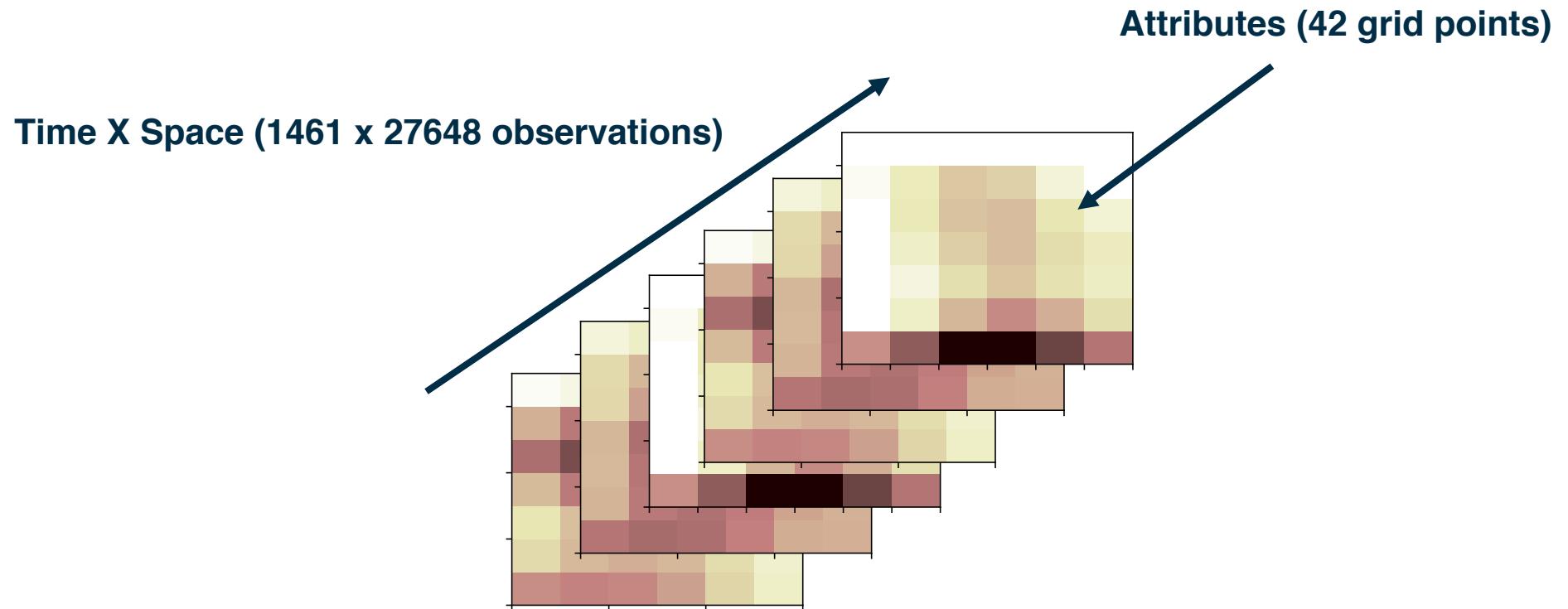
**In many instances in climate science, you are using gridded data over time and space**

**This adds up quickly**

In my current work: using histograms of cloud optical depth vs cloud top pressure (the attributes) for global gridded data, over 4 years.



To perform the clustering, I either need heaps of memory (less feasible)  
or to distribute the computations across a number of cores



# Dask ML

**Thankfully, Dask – Machine Learning, can do this for you, if you are using *k*-means.**

**Set up for *k*-means very similar to SK-Learn, super easy to implement**

**Dask handles the distribution for you**

[https://ml.dask.org/modules/generated/dask\\_ml.cluster.KMeans.html](https://ml.dask.org/modules/generated/dask_ml.cluster.KMeans.html)



The screenshot shows the Dask API Reference page for the `dask_ml.cluster.KMeans` class. The page has a dark header with the Dask logo and navigation links for Get Started, Algorithms, Setup, and Community. The main content includes the class definition, parameters, and a warning message. A sidebar on the left lists various Dask modules and integration options.

**dask\_ml.cluster.KMeans**

```
class dask_ml.cluster.KMeans(n_clusters=8, init='k-means||', oversampling_factor=2, max_iter=300, tol=0.0001, precompute_distances='auto', random_state=None, copy_x=True, n_jobs=1, algorithm='full', init_max_iter=None)
```

Scalable KMeans for clustering

**Parameters:**

- `n_clusters : int, default 8`  
Number of clusters to end up with
- `init : {'k-means||', 'k-means++' or ndarray}`  
'k-means||' : selects the the gg  
'k-means++' : selects the initial cluster centers in a smart way to speed up convergence. Uses scikit-learn's implementation.

**Warning**

If using '`k-means++`', the entire dataset will be read into memory at once.

An array of shape (n\_clusters, n\_features) can be used to give an explicit starting point

**oversampling\_factor : int, default 2**  
Oversampling factor for use in the `k-means||` algorithm.

**max\_iter : int**  
Maximum number EM iterations to attempt.

**init\_max\_iter : int**  
Number of iterations for init step.

**tol : float**  
Relative tolerance with regards to inertia to declare convergence

**algorithm : 'full'**  
The algorithm to use for the EM step. Only "full" (Lloyd's algorithm) is allowed.

**random\_state : int, RandomState instance or None, optional, default: None**

**Clustering**

**API Reference**

- `dask_ml.model_selection: Model Selection`
- `dask_ml.ensemble: Ensemble Methods`
- `dask_ml.linear_model: Generalized Linear Models`
- `dask_ml.naive_bayes: Naive Bayes`
- `dask_ml.wrappers: Meta-Estimators`
- `dask_ml.cluster: Clustering`
  - `dask_ml.cluster.KMeans`
  - `dask_ml.cluster.SpectralClustering`
- `dask_ml.decomposition: Matrix Decomposition`
- `dask_ml.preprocessing: Preprocessing Data`
- `dask_ml.feature_extraction.text: Feature extraction`
- `dask_ml.compose: Composite Estimators`
- `dask_ml.impute: Imputing Missing Data`
- `dask_ml.metrics: Metrics`
- `dask_ml.xgboost: XGBoost`
- `dask_ml.datasets: Datasets`

**INTEGRATION**

- Scikit-Learn & Joblib
- XGBoost & LightGBM
- PyTorch
- Keras and Tensorflow

**DEVELOP**

- Changelog
- Contributing
- Dask-ML Roadmap
- History

# Pangeo

To make your life even easier, the NCI have now set up a Pangeo environment specifically for these types of tasks, using JupyterLab

<https://pangeo.io/>

## About Pangeo

- Motivation
- Mission Statement
- Goals
- Get Involved
- [Edit on GitHub](#)

# ABOUT PANGEO

Pangeo is first and foremost a [community of people](#) working collaboratively to develop software and infrastructure to enable Big Data geoscience research.

Some of the products produced by this community include interconnected [software package](#) and [deployments](#) of this software in cloud and high-performance-computing environments. Such a deployment is sometimes referred to as a *Pangeo Environment*.

## MOTIVATION

There are several building crises facing the geoscience community:

- **Big Data:** datasets are growing too rapidly and legacy software tools for scientific analysis can't handle them. This is a major obstacle to scientific progress.
- **Technology Gap:** a growing gap between the technological sophistication of industry solutions (high) and scientific software (low).
- **Reproducibility:** a fragmentation of software tools and environments renders most geoscience research effectively unrepeatable and prone to failure.

Pangeo aims to address these challenges through a unified, collaborative effort.

## MISSION STATEMENT

Our mission is to cultivate an ecosystem in which the next generation of open-source analysis tools for ocean, atmosphere and climate science can be developed, distributed, and sustained. These tools must be scalable in order to meet the current and future challenges of big data, and these solutions should leverage the existing expertise outside of the geoscience community.

## GOALS

To accomplish this mission, we have identified three specific goals.

1. Foster collaboration around the open source scientific python ecosystem for ocean / atmosphere / land / climate science.
2. Support the development with domain-specific geoscience packages.
3. Improve scalability of these tools to handle petabyte-scale datasets on HPC and cloud platforms.

**The NCI have clear instructions for setting a Pangeo environment, which included standard python packages needed for this type of application (xarray, dask, Jupyter and more!)**

[https://nci-data-training.readthedocs.io/en/latest/\\_notebook/prep/Setup\\_Pangeo\\_environment\\_Gadi.html](https://nci-data-training.readthedocs.io/en/latest/_notebook/prep/Setup_Pangeo_environment_Gadi.html)

NCI data training  
latest

Search docs

Overview

**DATA INFO**

- Where to Find Data
- Where to Get Data
- How to Use Data

**HOW TO RUN JUPYTER NOTEBOOKS**

- On your local machine
- On the VDI
- On Gadi

**NOTEBOOK EXAMPLES GROUPED BY THEME**

- Climate and Environment
- Earth Observation
- Geophysics

**NOTEBOOK EXAMPLES GROUPED BY SERVICE**

- THREDDS
- GSKY

**HELP**

Help



Only 1 type of labeling automation lowers costs. Which type are you using? [Download The Guide](#)

Sponsored · Ads served ethically

Read the Docs v: latest ▾

## Setup for Pangeo Environment on Gadi

In this notebook we will go through:

- Configuring the Pangeo environment using your Gadi account
- Submitting and monitoring multi-node Pangeo jobs to Gadi
- Running Pangeo Jupyter notebooks in batch mode non-interactively

The [Pangeo](#) software ecosystem involves open source tools such as [xarray](#), [iris](#), [dask](#), [Jupyter](#) and many other packages.

This notebook provides instructions on how to use the Pangeo environment to run a multi-node parallel Jupyter notebook within the queues on Gadi and shows how to interact with it from your desktop.

### Configuring your account on Gadi

#### Step 1: Enabling Pangeo in your shell environment

To enable the Pangeo environment, you can use the following command within jobs, or within an interactive environment:

```
$ module load pangeo
Loading pangeo/2020.05
Loading requirement: intel-mkl/2019.3.199 python3/3.7.4 hdf5/1.10.5 netcdf/4.7.3
```

Note that Pangeo has its own Python installation.

#### Step 2: Configuring your JupyterLab password on Gadi

We will use JupyterLab to load notebooks and monitor jobs. JupyterLab is bundled within the Pangeo environment. To setup this environment, run the following two commands:

```
$ jupyter notebook --generate-config
$ jupyter notebook password
```

This is a stand-alone password that you will use later for accessing the JupyterLab server. You can use this command to reset your password at any time.

NCI also have a range of example notebooks using Dask and a Pangeo environment (including one for clustering!):

<https://github.com/NCI-data-analysis-platform/examples-dask>

<https://opus.nci.org.au/display/Help/examples-dask>

## Dask for Machine Learning

- Scaling problems
- Distributed training
- Train larger-than-memory datasets

- Authors: NCI Virtual Research Environment Team
- Keywords: Machine Learning, Dask-ML, Training, Scaling
- Creation Date: 2020-Sep
- Lineage/Reference: This tutorial is referenced to [dask tutorial](#).

### Set up

Choose from the following two options to create a client:

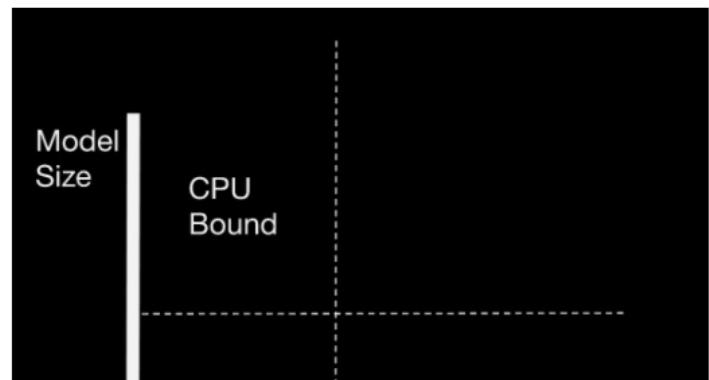
```
In [ ]: # If you run this notebook on your local computer or NCI's VDI instance, you can create cluster
from dask.distributed import Client, progress
client = Client()
client
```

```
In [ ]: # If you run this notebook on Gadi under pangeo environment, you can create cluster using scheduler.json file
from dask.distributed import Client, LocalCluster
client = Client(scheduler_file='scheduler.json')
print(client)
```

### Types of Scaling

There are a couple of distinct scaling problems you might face. The scaling strategy depends on which problem you're facing.

1. CPU-Bound: Data fits in RAM, but training takes too long. Many hyperparameter combinations, a large ensemble of many models, etc.
2. Memory-bound: Data is larger than RAM, and sampling isn't an option.



# Example

## Prep the MODIS data

Rename dims and merge first two tau bins, flatten and normalise. Then save out.

- **Read in the data I need**
- **Organise it into the dimensions I need**
- **Flatten the attributes (so instead of 6x7, it is 1x42) and the observations (1461x144x192 to 40,393,728x1)**
- **Normalise**
- **Ready to be passed to k-means**

```
fdir = '/g/data/p66/slf563/OBS/MCD06COSP_D3_MODIS/'  
modis = xr.open_mfdataset(fdir+'MCD06COSP_D3_MODIS.201*PCL_CL.nc', parallel=True, combine='by_coords').astype('float32')  
modis = modis.drop_vars(['longitude', 'latitude', 'Mean', 'Standard_Deviation', 'Sum', 'Pixel_Counts', 'Sum_Squares'])  
modis = modis.rename_dims({'jhisto_cloud_optical_thickness_total_7': 'tau',  
                           'jhisto_cloud_top_pressure_7': 'pressure',  
                           'lat': 'latitude',  
                           'lon': 'longitude'})  
modis = modis.rename_vars({'JHisto_vs_Cloud_Top_Pressure': 'histo',  
                           'lat': 'latitude',  
                           'lon': 'longitude'})  
modis = modis.transpose('tau', 'pressure', 'time', 'latitude', 'longitude')  
  
data2 = modis.histo[1:,:,:]  
data2[0,:,:].values = data2[0,:,:].values + modis.histo[0,:,:].values  
modis = data2.fillna(0)  
  
dataout = modis.stack(x = ('time', 'longitude', 'latitude'),  
                      y = ('tau', 'pressure')).reset_index(dims_or_levels=['x', 'y'])  
dataout = dataout.sum(dim='y')  
a = np.where(dataout.sum(dim='y').values!=0)[0]  
dataout = dataout.chunk(841536, 1)  
  
dataout.to_netcdf('/g/data/jk72/slf563/ACCESS/output/bx400/modis_clustering_data.nc')  
del dataout
```

modis

Show/Hide data repr Show/Hide attributes

xarray.DataArray

'histo'

- tau: 6
- pressure: 7
- time: 1461
- latitude: 144
- longitude: 192

## K-means:

We have worked out the optimum number of clusters previously.

You can see the actual running of *k*-means is the same!

I then re-shape the data back into its original form (ie, 6x7 attributes and 1461x144x192 observations)

```
[8]: nclus = 12

[9]: model = KMeans(n_clusters=nclus,random_state=0).fit(data)

/g/data/dk92/apps/anaconda3/2020.12/envs/NCI-data-analysis_2021.03/lib/python3.7/site-packages/distribu
    ("'open_dataset-f0ea9dc1d35bd78f322e113ff995e34h ... , None, None))"

Consider scattering large objects ahead of time
with client.scatter to reduce scheduler burden and
keep data on workers

    future = client.submit(func, big_data)      # bad

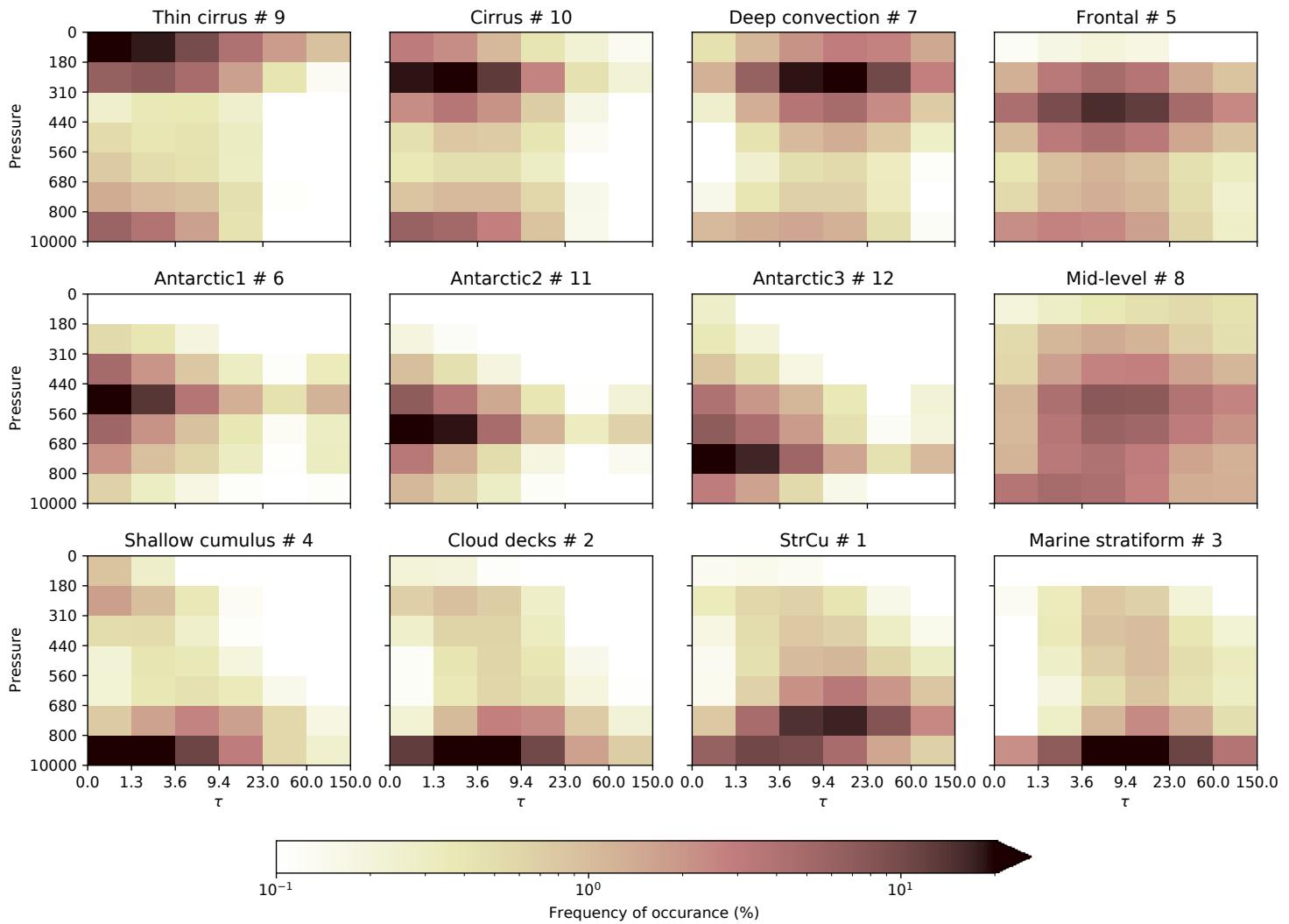
    big_future = client.scatter(big_data)        # good
    future = client.submit(func, big_future)     # good
    % (format_bytes(len(b)), s)

[10]: clusters = model.cluster_centers_
centers = clusters.reshape(nclus,6,7)

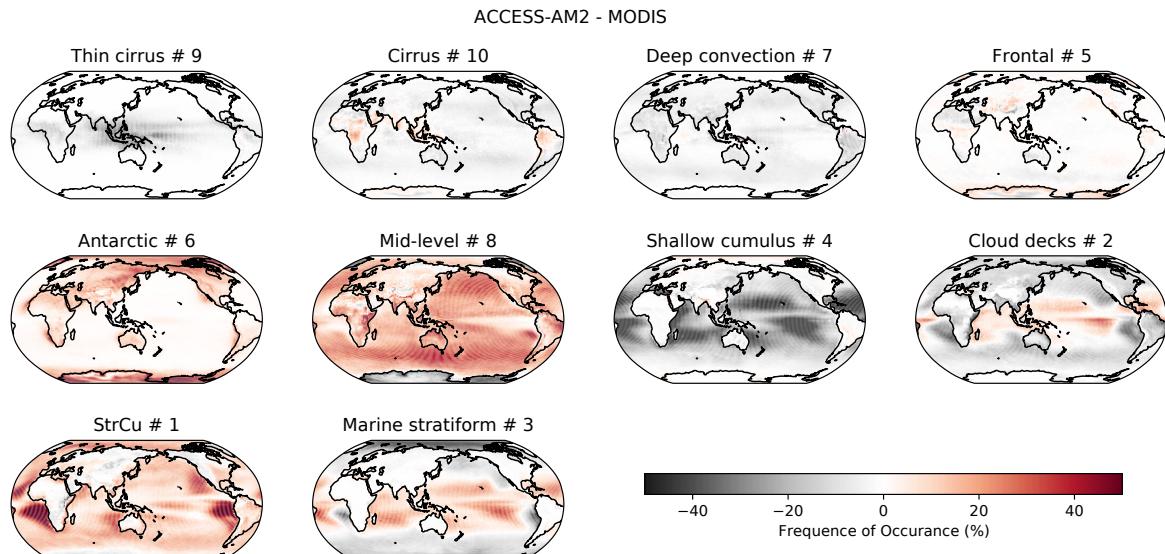
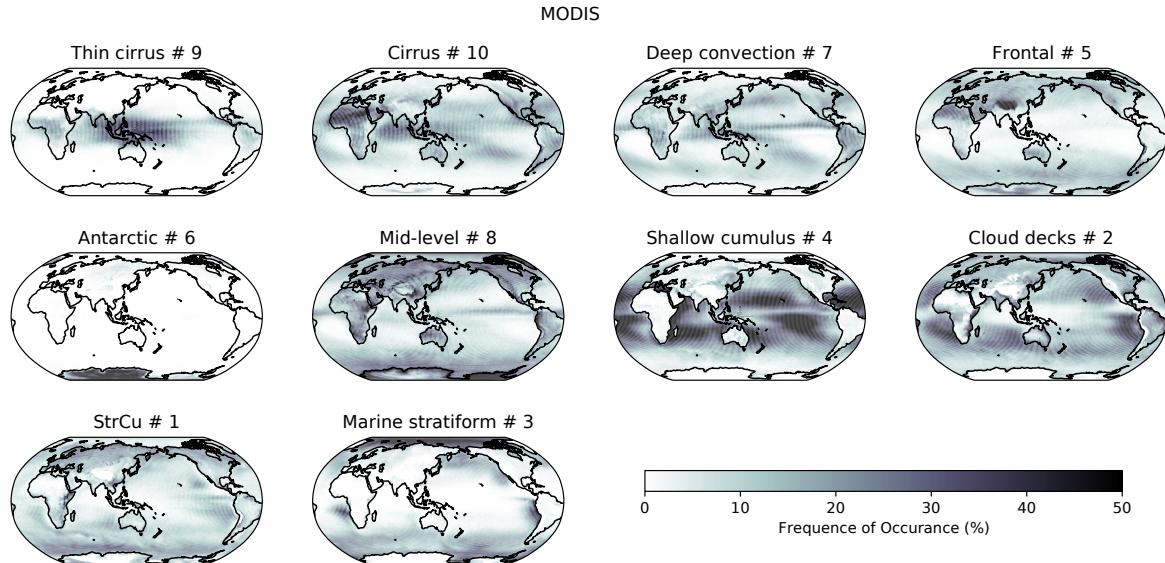
[11]: centers = xr.DataArray(dims=['k','Tau','Pressure'],
                           coords={'k':np.arange(1,nclus+1),'Tau':np.arange(0,6),'Pressure':np.arange(0,7)},
                           attrs={'Cloud top pressure':plevs,'Cloud optical depth':tau})
centers[:] = clusters.reshape(nclus,6,7)
centers = centers.transpose('k','Pressure','Tau')
centers = centers.reindex(Pressure=list(reversed(centers.Pressure)))

[12]: labels = model.labels_
C = xr.DataArray(dims=['time','latitude','longitude'],
                  coords={'time':modis.time,
                          'latitude':modis.latitude,
                          'longitude':modis.longitude}))
C = C.stack(x = ('time','longitude','latitude'))
C[a] = labels
C = C.unstack('x')
```

I end up with 12 cloud regimes based on cloud top pressure (y-axis) and cloud optical depth (x-axis).



I can then use this to evaluate the ACCESS climate model by fitting the learnt *k*-means centroids to the equivalent model data.



# Resources:

Hastie, T., Friedman, J., and Tibshirani, R. (2009). *The Elements of Statistical Learning*. Springer Series in Statistics. (Springer: New York, NY.) **Chapter 14.3**

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* 12(12), 2825–2830

[https://nci-data-training.readthedocs.io/en/latest/\\_notebook/prep/Setup\\_Pangeo\\_environment\\_Gadi.html](https://nci-data-training.readthedocs.io/en/latest/_notebook/prep/Setup_Pangeo_environment_Gadi.html)

<https://opus.nci.org.au/display/Help/examples-dask>

Fiddes, S., Pepler, A., Saunders, K., & Hope, P. (2021). Redefining southern Australia's climatic regions and seasons. *Journal of Southern Hemisphere Earth Systems Science*. <https://doi.org/10.1071/ES20003>

Trevor Hastie  
Robert Tibshirani  
Jerome Friedman

**The Elements of  
Statistical Learning**  
Data Mining, Inference, and Prediction

Second Edition

# THANK YOU

Any questions?



The Australian Antarctic Program Partnership is funded by the Australian Government Department of Industry, Science, Energy and Resources through the Antarctic Science Collaboration Initiative.

The Australian Antarctic Program Partnership is led by the University of Tasmania, and includes the following partner agencies



UNIVERSITY OF  
TASMANIA



IMAS  
INSTITUTE FOR MARINE  
& ANTARCTIC STUDIES



Australian Government  
Department of Agriculture,  
Water and the Environment



Australian Antarctic  
Program



CSIRO



Australian Government  
Bureau of Meteorology



IMOS  
Integrated Marine  
Observing System



Australian Government  
Geoscience Australia



Tasmanian  
Government