Week 10:

K- Means Clustering

```
using StatsBase

function kmeans_simple(X, k, max_iter = 100, threshold = 0.001)


    # Let's pick k points from X without replacment

    centroids = X[:, sample(1:size(X,2), k, replace = false)]


    # create a copy. This is used to check if the centroids are moving or not.

    new_centroids = copy(centroids)


    # start an empty array for our cluster ids. This will hold the cluster assignment

    # for each point in X

    cluster_ids = zeros(Float64, size(X, 2))


    for _ in 1:max_iter

        for col_idx in 1:size(X, 2) # iterate over each point


            # let's index the ponts one by one

            p = X[:, col_idx]


            # calculate the distance between the point and each centroid

            point_difference = mapslices(x -> x .- p, centroids, dims=[1])


            # we calculate the squared Euclidian distance
```

```
            distances = mapslices(sum, point_difference .^ 2, dims=[1])

            # now find the index of the closest centroid
            cluster_ids[col_idx] = findmin(distances)[2][2]
            # this gives the index of the minimum

            # you can uncomment this line to see how the loop progresses
            # println("p: $p diff: $point_difference dist: $distances $cluster_ids")
        end

        # you can uncomment this line to see the internal workings of the funtion
        # println("old: $centroids new: $new_centroids")

        # Iterate over each centroid
        for cluster_id in 1:size(centroids, 2)

            # find the mean of the assigned points for that particluar cluster
            mask = [i for (i, m) in enumerate(cluster_id .== cluster_ids) if m]
            new_centroids[:, cluster_id] = mapslices(mean, X[:, mask], dims=[2])
        end

        # You can uncomment this line to see how the centers move after each update
        # println("old_centroids: $centroids new_centroids: $new_centroids point assignemnts:
$cluster_ids")

        # now measure the total distance that the centroids moved
```

```
        center_change = sum(mapslices(x -> sum(x.^2), new_centroids .- centroids, dims=[2]))

        centroids = copy(new_centroids)

        # if the centroids move negligably, then we're done
        if center_change < threshold
            # println(i)
            break
        end
    end

    # we'll send back both the location of the centroids as well as the cluster ids for each point
    return centroids, cluster_ids
end

julia> kmeans_simple(data_simple, 2)
([0.7999999999999999 0.20000000000000004], [2.0 2.0 … 1.0 1.0])

data_complex = [
    0.1 0.1 0.2 0.4 0.5 0.5 0.9;
    0.1 0.2 0.1 0.4 0.3 0.4 1.0
]

julia> complex_result = kmeans_simple(data_complex, 3)
([0.13333333333333333 0.9 0.4666666666666666; 0.13333333333333333 1.0
0.3666666666666667], [1.0, 1.0, 1.0, 3.0, 3.0, 3.0, 2.0])
```

using Plots

gr()

draw original points

scatter(data_complex[1, :], data_complex[2, :])

draw the centroids

scatter!(complex_result[1][1, :], complex_result[1][2, :])