K-MEANS CLUSTERING ANALYSIS

in JULIA

by Jesse P. Gutierrez Jr UHD, Data Science

In [12]:
```
using Plots, Statistics
theme(:dark)
```

In [13]:
```
distance(p1,p2) = sqrt((p1[1]-p2[1])^2 + (p1[2] - p2[2])^2)
```

Out[13]:
```
distance (generic function with 1 method)
```

In [14]:
```
cluster1 = [p for p in zip(rand(1:50, 20), rand(1:70, 20))]
cluster2 = [p for p in zip(rand(45:70, 20), rand(45:80, 20))];
```

In [15]:
```
cluster1
```

Out[15]:
```
20-element Array{Tuple{Int64,Int64},1}:
 (43, 47)
 (2, 38)
 (30, 2)
 (39, 42)
 (48, 41)
 (38, 39)
 (50, 65)
 (34, 11)
 (45, 48)
 (2, 49)
 (29, 5)
 (22, 42)
 (2, 62)
 (42, 26)
 (6, 19)
 (10, 10)
 (45, 19)
 (14, 58)
 (43, 22)
 (48, 2)
```

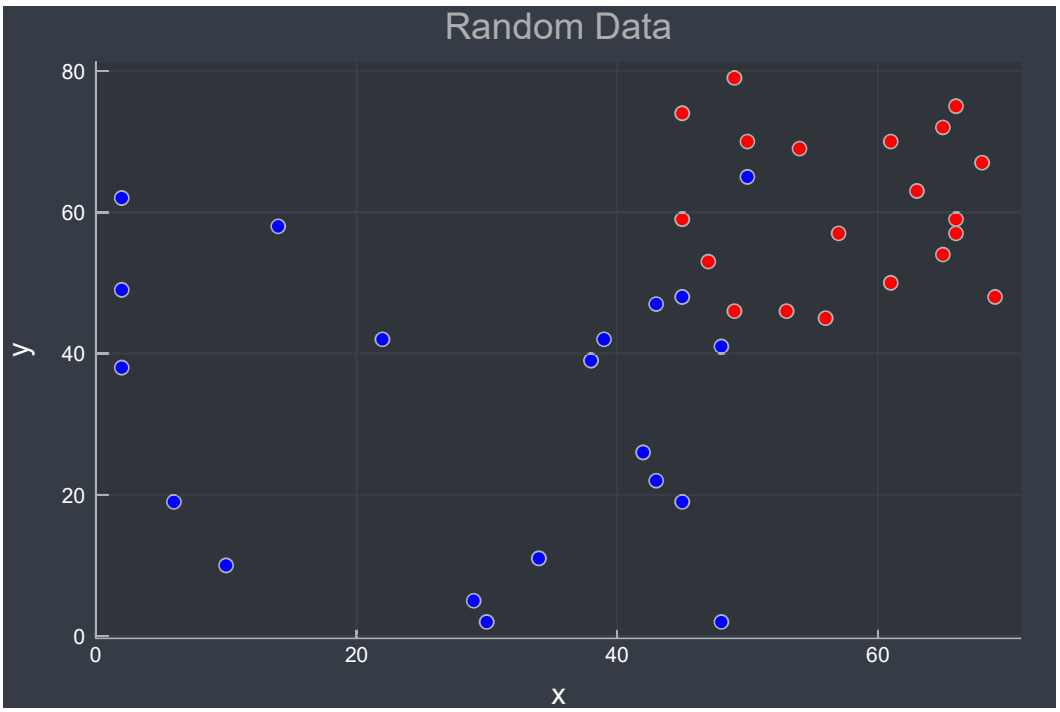
In [16]:
```
cluster2
```

Out[16]:
```
20-element Array{Tuple{Int64,Int64},1}:
 (45, 59)
 (66, 59)
 (66, 75)
 (54, 69)
 (63, 63)
 (56, 45)
 (57, 57)
 (65, 54)
 (65, 72)
 (49, 46)
 (66, 57)
 (45, 74)
 (50, 70)
 (61, 70)
 (61, 50)
 (68, 67)
 (47, 53)
 (69, 48)
 (53, 46)
 (49, 79)
```

In [17]:
```
scatter(cluster1,
    legend = false,
    color = "blue",
    xaxis = "x",
    yaxis = "y",
    title = "Random Data")

scatter!(cluster2,
    color = "red")
```

Out[17]:



In [18]:
```
X = [x for x in cluster1]
for x in cluster2
    push!(X, x)
end
```

In [19]:
```
X   # this is out unlabeled data points
```

Out[19]:
```
40-element Array{Tuple{Int64,Int64},1}:
 (43, 47)
 (2, 38)
 (30, 2)
 (39, 42)
 (48, 41)
 (38, 39)
 (50, 65)
 (34, 11)
 (45, 48)
 (2, 49)
 (29, 5)
 (22, 42)
 (2, 62)
 ⋮
 (65, 72)
 (49, 46)
 (66, 57)
 (45, 74)
 (50, 70)
 (61, 70)
 (61, 50)
 (68, 67)
 (47, 53)
 (69, 48)
 (53, 46)
 (49, 79)
```

In [ ]:
```
# some senior level programming
function k_Means_Clustering(X, k)
    centers = [(rand(0:70), rand(0:80)) for _ = 1:k]     #centers - random collecti
on of centers which is an array of
    #tuples.
    distances = [] # distance is empty but for each point make a temp_distance arra
y, consist of j, distance array
    # first entry is j tuples from centers, second entry is point p to distances to
center
    for p in X
        temp_distances = [(j, distance(p, centers[j])) for j = 1:length(centers)]
        sort!(temp_distances, by = x -> x[2])
        push!(distances, (p, temp_distances[1][1], temp_distances[1][2]))
    end
# starts to change things
```

In [34]:
```
# print the next to slides to see what it does
centers = [(rand(0:70), rand(0:80)) for _ = 1:2]     #centers - random collection o
f centers which is an array of
    #tuples.
distances = [] # distance is empty but for each point make a temp_distance array, c
onsist of j, distance array
    # first entry is j tuples from centers, second entry is point p to distances to
center
for p in X
    temp_distances = [(j, distance(p, centers[j])) for j = 1:length(centers)]
    sort!(temp_distances, by = x -> x[2])
    push!(distances, (p, temp_distances[1][1], temp_distances[1][2]))
end
# starts to change things
```

In [35]:
```
# prints point, index, distances with 2 centers, the centers are randomly changing
distances
```

Out[35]:
```
40-element Array{Any,1}:
 ((43, 47), 2, 27.294688127912362)
 ((2, 38), 2, 68.18357573492314)
 ((30, 2), 2, 57.28001396647874)
 ((39, 42), 2, 31.016124838541646)
 ((48, 41), 2, 22.090722034374522)
 ((38, 39), 2, 32.2490309931942)
 ((50, 65), 1, 15.0)
 ((34, 11), 2, 48.16637831516918)
 ((45, 48), 2, 25.495097567963924)
 ((2, 49), 1, 65.0)
 ((29, 5), 2, 55.90169943749474)
 ((22, 42), 2, 48.010415536631214)
 ((2, 62), 1, 63.071388124885914)
 ⋮
 ((65, 72), 1, 7.0)
 ((49, 46), 2, 21.213203435596427)
 ((66, 57), 1, 8.06225774829855)
 ((45, 74), 1, 21.93171219946131)
 ((50, 70), 1, 15.811388300841896)
 ((61, 70), 1, 6.4031242374328485)
 ((61, 50), 2, 11.40175425099138)
 ((68, 67), 1, 3.605551275463989)
 ((47, 53), 1, 21.633307652783937)
 ((69, 48), 2, 5.0990195135927845)
 ((53, 46), 2, 17.26267650163207)
 ((49, 79), 1, 21.2602916254693)
```

In [25]:
```
temp_p = distances[1] # point in our data set
```

Out[25]:
```
((43, 47), 2, 5.830951894845301)
```

In [26]:
```
centers # a center in data set
```

Out[26]:
```
2-element Array{Tuple{Int64,Int64},1}:
 (62, 35)
 (40, 42)
```

In [27]:
```
# closest to the (6,14), thats why the one above shows up//
println("The distance from (6,14) to u_1 is: ", distance((42,17),(6, 14)), "\n")
println("The distance from (6,14) to u_2 is: ", distance((67,50),(6, 14)), "\n")
```

```
The distance from (6,14) to u_1 is: 36.124783736376884

The distance from (6,14) to u_2 is: 70.83078426785913
```

In [36]:
```
###notes
(6,4).+(24,37)
```

Out[36]:
```
(30, 41)
```

In [ ]:

In [ ]:

In [67]:
```
function k_Means_Assignment(X, centers, k)

    distances = []
    for p in X
        temp_distances = [(j, distance(p, centers[j])) for j = 1:length(centers)]
        sort!(temp_distances, by = x -> x[2])
        push!(distances, (p, temp_distances[1][1], temp_distances[1][2]))
    end

    new_centers = copy(centers)
    for j = 1:k
        new_centers[j] = mean(([x[1][1] for x in distances if x[2] == j]),
                              mean([x[1][2] for x in distances if x[2] == j]))
    end

    return new_centers
end
```

Out[67]:
```
k_Means_Assignment (generic function with 1 method)
```

In [68]:
```
function k_Means_Clustering(X, k, ϵ)
    centers = [(rand(0.0:70.0), rand(0.0:80.0)) for _ = 1:k]
    new_centers = k_Means_Assignment(X, centers, k)
    while maximum([distance(centers[i], new_centers[i]) for i = 1:k]) > ϵ
      centers, new_centers = new_centers, k_Means_Assignment(X, new_centers, k)
    end
    return new_centers
end
```

Out[68]:
```
k_Means_Clustering (generic function with 1 method)
```

In [69]:
```
centers = k_Means_Clustering(X, 2, 0.05)
```

```
MethodError: no method matching mean(::Array{Int64,1}, ::Float64)
Closest candidates are:
  mean(!Matched::Union{Function, Type}, ::Any) at C:\cygwin\home\Administrator\b
uildbot\worker\package_win64\build\usr\share\julia\stdlib\v1.0\Statistics\src\St
atistics.jl:58
  mean(::AbstractArray{T<:Number,N} where N, !Matched::StatsBase.AbstractWeights
{W<:Real,T,V} where V<:AbstractArray{T,1} where T<:Real, !Matched::Int64) where
{T<:Number, W<:Real} at deprecated.jl:53
  mean(::AbstractArray; dims) at C:\cygwin\home\Administrator\buildbot\worker\pa
ckage_win64\build\usr\share\julia\stdlib\v1.0\Statistics\src\Statistics.jl:128
  ...

Stacktrace:
 [1] k_Means_Assignment(::Array{Tuple{Int64,Int64},1}, ::Array{Tuple{Float64,Flo
at64},1}, ::Int64) at .\In[67]:12
 [2] k_Means_Clustering(::Array{Tuple{Int64,Int64},1}, ::Int64, ::Float64) at .\
In[68]:3
 [3] top-level scope at In[69]:1
```

In [53]:
```

```

In [52]:
```

```

In [37]:
```
# sum(x[1][1] for x in distances)
```

Out[37]:
```
1710
```

In [ ]:
```
#[(x[1][1], x[1][2]) for x in distances if x[2] == 1] just to see what we get
```

In [41]:
```
#n, m = sum(x[1][1] for x in distances if x[2] == 1), sum(x[1][2] for x in distance
s if x[2] == 1) to get sum
```

Out[41]:
```
(1234, 1282)
```

In [ ]:
```
n, m = mean([x[1][1] for x in distances if x[2] == 1]), sum([x[1][2] for x in dista
nces if x[2] == 1])
```

In [28]:
```
temp_p = distances[1]
```

Out[28]:
```
((37, 14), 2, 6.708203932499369)
```

In [29]:
```
centers
```

Out[29]:
```
2-element Array{Tuple{Int64,Int64},1}:
 (1, 39)
 (40, 20)
```

In [32]:
```
X
```

Out[32]:
```
40-element Array{Tuple{Int64,Int64},1}:
 (37, 14)
 (45, 65)
 (34, 7)
 (24, 11)
 (4, 30)
 (36, 43)
 (6, 21)
 (24, 14)
 (29, 37)
 (19, 13)
 (13, 63)
 (48, 65)
 (35, 44)
 ⋮
 (51, 56)
 (55, 52)
 (62, 72)
 (61, 71)
 (59, 52)
 (63, 64)
 (48, 59)
 (58, 67)
 (70, 54)
 (69, 56)
 (49, 49)
 (65, 60)
```

This is a work in progress....getting syntax error

In [46]:
```
k_Means_Assignment(X, 2)
```

```
MethodError: no method matching k_Means_Assignment(::Array{Tuple{Int64,Int64},1}
, ::Int64)
Closest candidates are:
  k_Means_Assignment(::Any, ::Any, !Matched::Any) at In[43]:2

Stacktrace:
 [1] top-level scope at In[46]:1
```

In [ ]:

In [ ]:

In [ ]:

In [48]:
```
Centers = k_Means_Clustering(X, 2, 0.05)
```

```
MethodError: no method matching (::Colon)(::Int64, ::Array{Tuple{Int64,Int64},1}
)
Closest candidates are:
  Colon(::T<:Real, ::Any, !Matched::T<:Real) where T<:Real at range.jl:40
  Colon(::A<:Real, ::Any, !Matched::C<:Real) where {A<:Real, C<:Real} at range.j
l:10
  Colon(::T, ::Any, !Matched::T) where T at range.jl:39
  ...

Stacktrace:
 [1] k_Means_Assignment(::Array{Tuple{Int64,Int64},1}, ::Array{Tuple{Int64,Int64
},1}, ::Int64) at .\In[43]:2
 [2] k_Means_Clustering(::Array{Tuple{Int64,Int64},1}, ::Int64, ::Float64) at .\
In[43]:21
 [3] top-level scope at In[48]:1
```

In [47]:
```
for p in X
    temp_distances = [(j, distance(p, Centers[j])) for j = 1:2]
    sort!(temp_distances, by = x -> x[2])
    plot!([p, Centers[temp_distances[1][1]]],
        color = temp_distances[1][1] == 1 ? "blue" : "red")
end
plot!()
```

```
UndefVarError: Centers not defined

Stacktrace:
 [1] (::getfield(Main, Symbol("##119#121")){Tuple{Int64,Int64}})(::Int64) at .\n
one:0
 [2] iterate at .\generator.jl:47 [inlined]
 [3] collect(::Base.Generator{UnitRange{Int64},getfield(Main, Symbol("##119#121"
)){Tuple{Int64,Int64}}}) at .\array.jl:619
 [4] top-level scope at In[47]:2
```

In [ ]:

In [ ]: