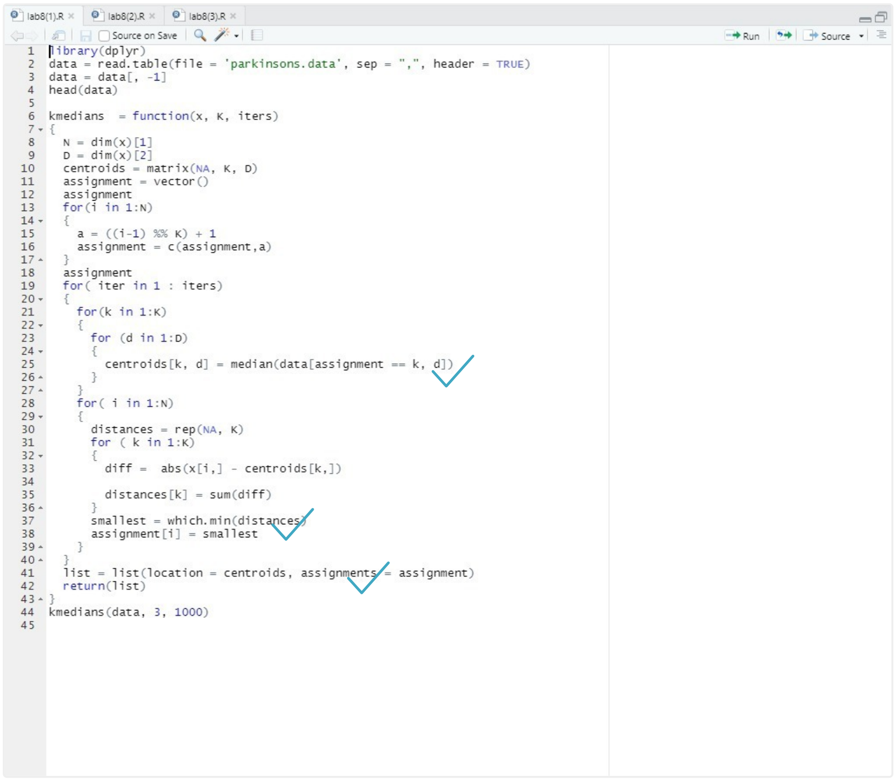
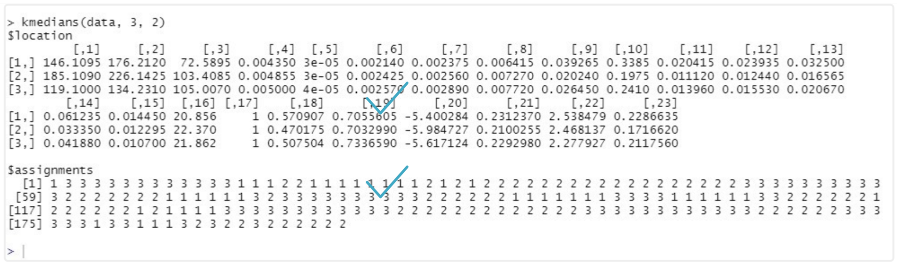
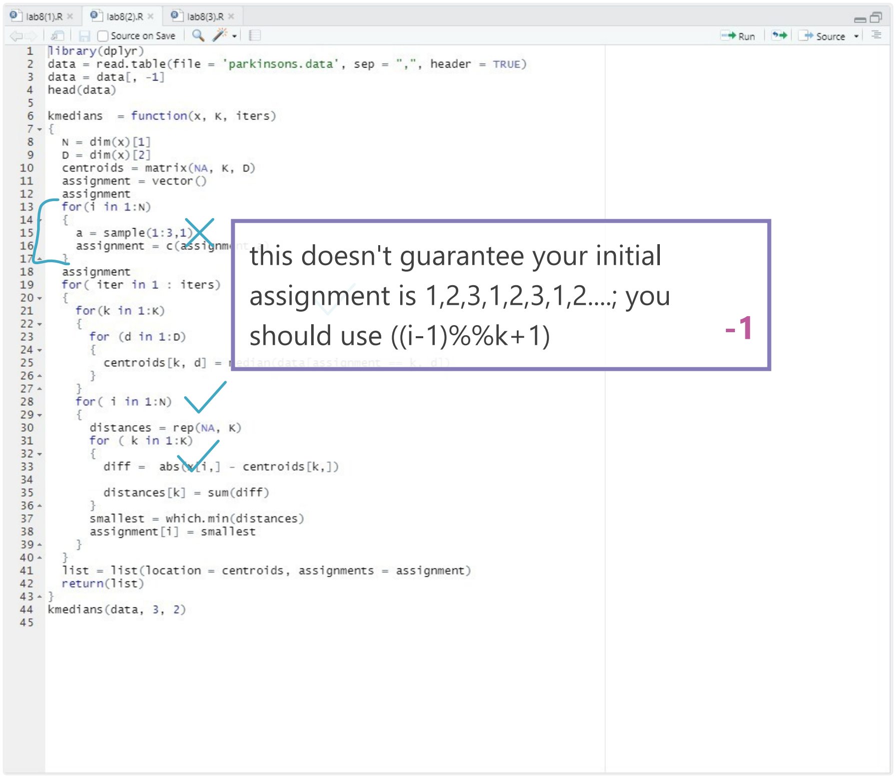


```r
library(dplyr)
data = read.table(file = 'parkinsons.data', sep = ",", header = TRUE)
data = data[, -1]
head(data)

kmedians  = function(x, K, iters)
{
  N = dim(x)[1]
  D = dim(x)[2]
  centroids = matrix(NA, K, D)
  assignment = vector()
  assignment
  for(i in 1:N)
  {
    a = ((i-1) %% K) + 1
    assignment = c(assignment,a)
  }
  assignment
  for( iter in 1 : iters)
  {
    for(k in 1:K)
    {
      for (d in 1:D)
      {
        centroids[k, d] = median(data[assignment == k, d]) ✓
      }
    }
    for( i in 1:N)
    {
      distances = rep(NA, K)
      for ( k in 1:K)
      {
        diff =  abs(x[i,] - centroids[k,])

        distances[k] = sum(diff)
      }
      smallest = which.min(distances) ✓
      assignment[i] = smallest
    }
  }
  list = list(location = centroids, assignments ✓ = assignment)
  return(list)
}
kmedians(data, 3, 1000)
```



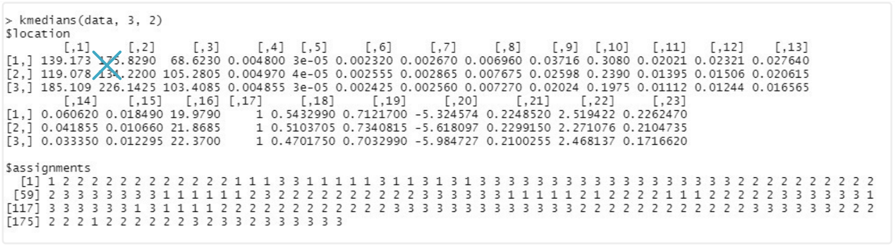
```
> kmedians(data, 3, 2)
$location
         [,1]     [,2]     [,3]     [,4]  [,5]     [,6]     [,7]     [,8]     [,9]  [,10]    [,11]    [,12]    [,13]
[1,] 146.1095 176.2120  72.5895 0.004350 3e-05 0.002140 0.002375 0.006415 0.039265 0.3385 0.020415 0.023935 0.032500
[2,] 185.1090 226.1425 103.4085 0.004855 3e-05 0.002425 0.002560 0.007270 0.020240 0.1975 0.011120 0.012440 0.016565
[3,] 119.1000 134.2310 105.0070 0.005000 4e-05 0.002570 0.002890 0.007720 0.026450 0.2410 0.013960 0.015530 0.020670
        [,14]    [,15]  [,16] [,17]    [,18]     [,19]     [,20]     [,21]    [,22]     [,23]
[1,] 0.061235 0.014450 20.856     1 0.570907 0.7055605 -5.400284 0.2312370 2.538479 0.2286635
[2,] 0.033350 0.012295 22.370     1 0.470175 0.7032990 -5.984727 0.2100255 2.468137 0.1716620
[3,] 0.041880 0.010700 21.862     1 0.507504 0.7336590 -5.617124 0.2292980 2.277927 0.2117560

$assignments
  [1] 1 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 2 2 1 1 1 1 1 1 1 1 2 1 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3
 [59] 3 2 2 2 2 2 2 2 1 1 1 1 1 1 3 2 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 1 1 1 1 1 1 1 3 3 3 3 1 1 1 1 1 1 3 3 2 2 2 2 2 2 1
[117] 2 2 2 2 2 2 1 2 1 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 3 3 3
[175] 3 3 3 1 3 3 1 1 1 3 2 3 2 2 3 2 2 2 2 2 2

> |
```



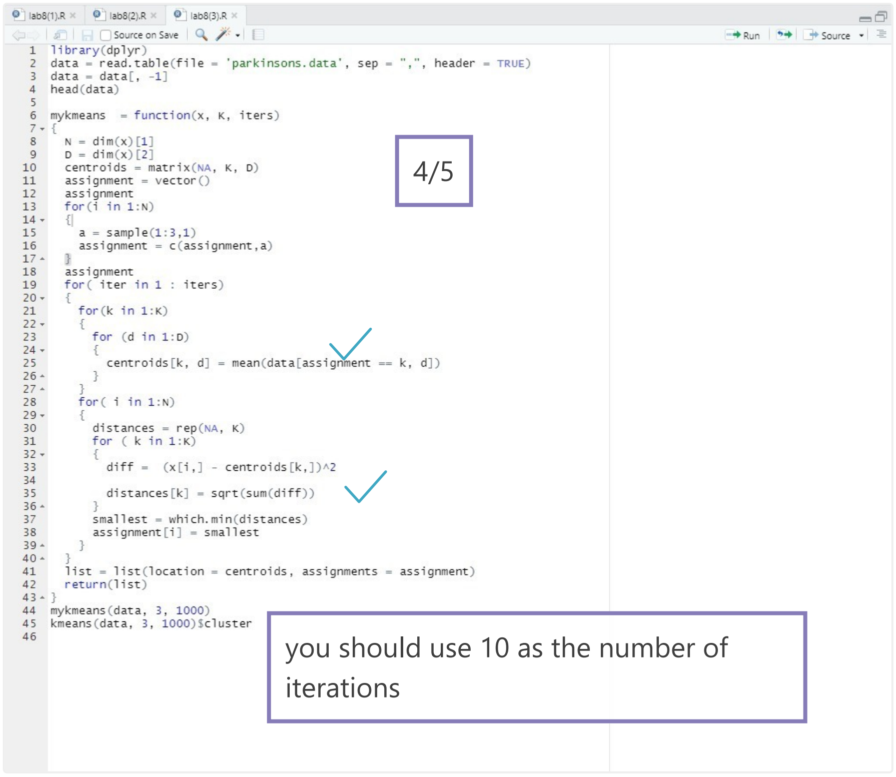
```r
library(dplyr)
data = read.table(file = 'parkinsons.data', sep = ",", header = TRUE)
data = data[, -1]
head(data)

kmedians  = function(x, K, iters)
{
  N = dim(x)[1]
  D = dim(x)[2]
  centroids = matrix(NA, K, D)
  assignment = vector()
  assignment
  for(i in 1:N)
  {
    a = sample(1:3,1)
    assignment = c(assignment, a)
  }
  assignment
  for( iter in 1 : iters)
  {
    for(k in 1:K)
    {
      for (d in 1:D)
      {
        centroids[k, d] = median(data[assignment == k, d])
      }
    }
    for( i in 1:N)
    {
      distances = rep(NA, K)
      for ( k in 1:K)
      {
        diff =  abs(x[i,] - centroids[k,])

        distances[k] = sum(diff)
      }
      smallest = which.min(distances)
      assignment[i] = smallest
    }
  }
  list = list(location = centroids, assignments = assignment)
  return(list)
}
kmedians(data, 3, 2)

```

this doesn't guarantee your initial assignment is 1,2,3,1,2,3,1,2....; you should use ((i-1)%%k+1)

-1



```
> kmedians(data, 3, 2)
$location
        [,1]     [,2]     [,3]     [,4]  [,5]     [,6]     [,7]     [,8]    [,9]  [,10]   [,11]   [,12]    [,13]
[1,] 139.173 175.8290  68.6230 0.004800 3e-05 0.002320 0.002670 0.006960 0.03716 0.3080 0.02021 0.02321 0.027640
[2,] 119.078 134.2200 105.2805 0.004970 4e-05 0.002555 0.002865 0.007675 0.02598 0.2390 0.01395 0.01506 0.020615
[3,] 185.109 226.1425 103.4085 0.004855 3e-05 0.002425 0.002560 0.007270 0.02024 0.1975 0.01112 0.01244 0.016565
        [,14]    [,15]   [,16] [,17]     [,18]     [,19]     [,20]     [,21]    [,22]     [,23]
[1,] 0.060620 0.018490 19.9790     1 0.5432990 0.7121700 -5.324574 0.2248520 2.519422 0.2262470
[2,] 0.041855 0.010660 21.8685     1 0.5103705 0.7340815 -5.618097 0.2299150 2.271076 0.2104735
[3,] 0.033350 0.012295 22.3700     1 0.4701750 0.7032990 -5.984727 0.2100255 2.468137 0.1716620

$assignments
  [1] 1 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 3 3 1 1 1 1 1 3 1 1 3 1 3 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2 2 2
 [59] 2 3 3 3 3 3 3 3 1 1 1 1 1 1 2 3 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 1 1 1 1 1 2 1 2 2 2 2 1 1 1 2 2 2 2 2 3 3 3 3 3 3 1
[117] 3 3 3 3 3 3 1 3 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 2 2 2
[175] 2 2 2 1 2 2 2 2 2 2 3 2 3 3 2 3 3 3 3 3 3
```



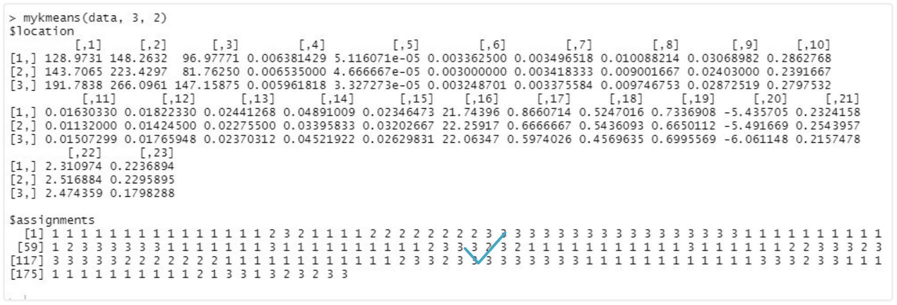
```
library(dplyr)
data = read.table(file = 'parkinsons.data', sep = ",", header = TRUE)
data = data[, -1]
head(data)

mykmeans  = function(x, K, iters)
{
  N = dim(x)[1]
  D = dim(x)[2]
  centroids = matrix(NA, K, D)
  assignment = vector()
  assignment
  for(i in 1:N)
  {
    a = sample(1:3,1)
    assignment = c(assignment,a)
  }
  assignment
  for( iter in 1 : iters)
  {
    for(k in 1:K)
    {
      for (d in 1:D)
      {
        centroids[k, d] = mean(data[assignment == k, d])
      }
    }
    for( i in 1:N)
    {
      distances = rep(NA, K)
      for ( k in 1:K)
      {
        diff =  (x[i,] - centroids[k,])^2

        distances[k] = sqrt(sum(diff))
      }
      smallest = which.min(distances)
      assignment[i] = smallest
    }
  }
  list = list(location = centroids, assignments = assignment)
  return(list)
}
mykmeans(data, 3, 1000)
kmeans(data, 3, 1000)$cluster
```

✓

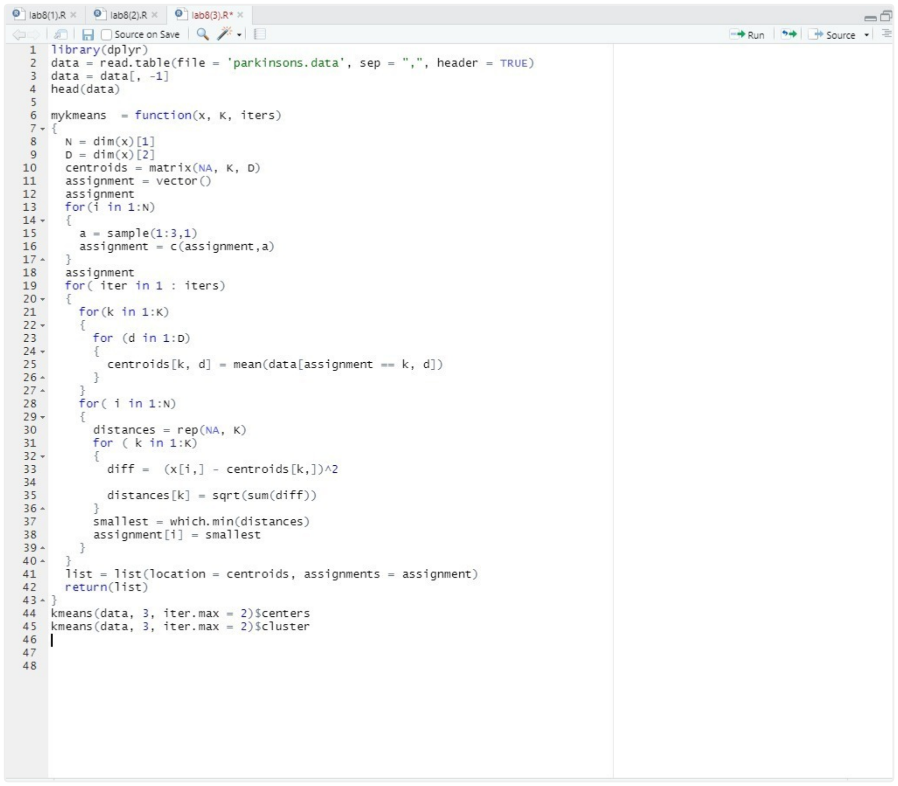
✓

4/5

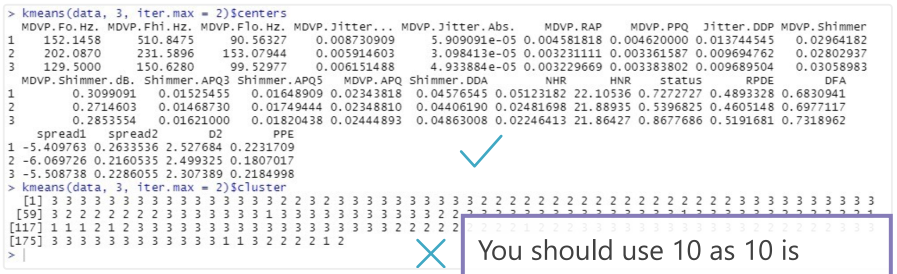
you should use 10 as the number of iterations



```
> mykmeans(data, 3, 2)
$location
         [,1]     [,2]      [,3]        [,4]         [,5]        [,6]        [,7]        [,8]       [,9]     [,10]
[1,] 128.9731 148.2632  96.97771 0.006381429 5.116071e-05 0.003362500 0.003496518 0.010088214 0.03068982 0.2862768
[2,] 143.7065 223.4297  81.76250 0.006535000 4.666667e-05 0.003000000 0.003418333 0.009001667 0.02403000 0.2391667
[3,] 191.7838 266.0961 147.15875 0.005961818 3.327273e-05 0.003248701 0.003375584 0.009746753 0.02872519 0.2797532
          [,11]      [,12]      [,13]      [,14]      [,15]    [,16]     [,17]     [,18]     [,19]     [,20]     [,21]
[1,] 0.01630330 0.01822330 0.02441268 0.04891009 0.02346473 21.74396 0.8660714 0.5247016 0.7336908 -5.435705 0.2324158
[2,] 0.01132000 0.01424500 0.02275500 0.03395833 0.03202667 22.25917 0.6666667 0.5436093 0.6650112 -5.491669 0.2543957
[3,] 0.01507299 0.01765948 0.02370312 0.04521922 0.02629831 22.06347 0.5974026 0.4569635 0.6995569 -6.061148 0.2157478
        [,22]     [,23]
[1,] 2.310974 0.2236894
[2,] 2.516884 0.2295895
[3,] 2.474359 0.1798288

$assignments
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 3 2 1 1 1 1 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1
 [59] 1 2 3 3 3 3 3 3 1 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1 1 1 2 3 3 3 2 3 2 1 1 1 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1 2 2 3 3 3 2 3
[117] 3 3 3 3 3 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 2 3 3 2 3 3 3 3 3 3 3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3 2 3 3 1 1 1
[175] 1 1 1 1 1 1 1 1 1 1 2 1 3 3 1 3 2 3 2 3 3
```

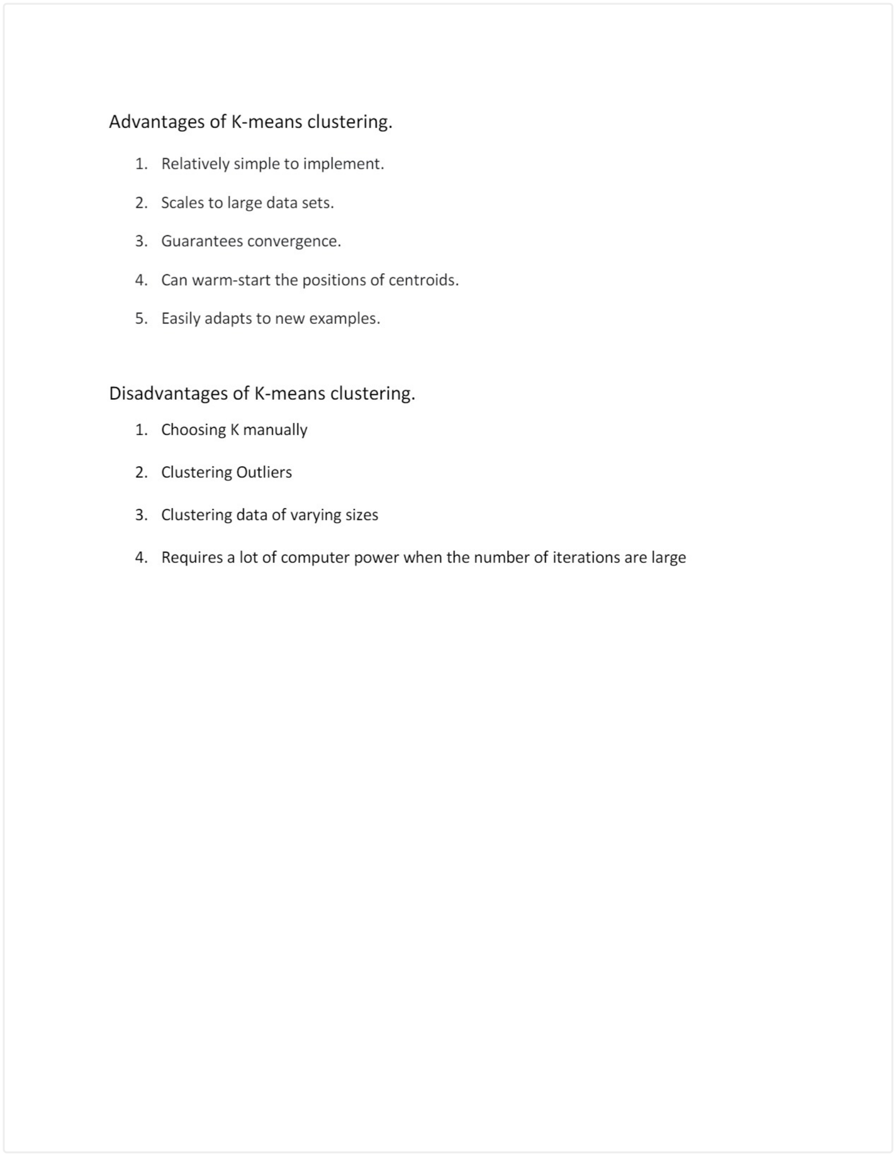


```r
library(dplyr)
data = read.table(file = 'parkinsons.data', sep = ",", header = TRUE)
data = data[, -1]
head(data)

mykmeans  = function(x, K, iters)
{
  N = dim(x)[1]
  D = dim(x)[2]
  centroids = matrix(NA, K, D)
  assignment = vector()
  assignment
  for(i in 1:N)
  {
    a = sample(1:3,1)
    assignment = c(assignment,a)
  }
  assignment
  for( iter in 1 : iters)
  {
    for(k in 1:K)
    {
      for (d in 1:D)
      {
        centroids[k, d] = mean(data[assignment == k, d])
      }
    }
    for( i in 1:N)
    {
      distances = rep(NA, K)
      for ( k in 1:K)
      {
        diff =  (x[i,] - centroids[k,])^2

        distances[k] = sqrt(sum(diff))
      }
      smallest = which.min(distances)
      assignment[i] = smallest
    }
  }
  list = list(location = centroids, assignments = assignment)
  return(list)
}
kmeans(data, 3, iter.max = 2)$centers
kmeans(data, 3, iter.max = 2)$cluster



```



```
> kmeans(data, 3, iter.max = 2)$centers
  MDVP.Fo.Hz. MDVP.Fhi.Hz. MDVP.Flo.Hz. MDVP.Jitter... MDVP.Jitter.Abs.    MDVP.RAP    MDVP.PPQ  Jitter.DDP MDVP.Shimmer
1    152.1458     510.8475     90.56327    0.008730909     5.909091e-05 0.004581818 0.004620000 0.013744545   0.02964182
2    202.0870     231.5896    153.07944    0.005914603     3.098413e-05 0.003231111 0.003361587 0.009694762   0.02802937
3    129.5000     150.6280     99.52977    0.006151488     4.933884e-05 0.003229669 0.003383802 0.009689504   0.03058983
  MDVP.Shimmer.dB. Shimmer.APQ3 Shimmer.APQ5   MDVP.APQ Shimmer.DDA        NHR      HNR    status      RPDE       DFA
1        0.3099091   0.01525455   0.01648909 0.02343818  0.04576545 0.05123182 22.10536 0.7272727 0.4893328 0.6830941
2        0.2714603   0.01468730   0.01749444 0.02348810  0.04406190 0.02481698 21.88935 0.5396825 0.4605148 0.6977117
3        0.2853554   0.01621000   0.01820438 0.02444893  0.04863008 0.02246413 21.86427 0.8677686 0.5191681 0.7318962
    spread1   spread2       D2       PPE
1 -5.409763 0.2633536 2.527684 0.2231709
2 -6.069726 0.2160535 2.499325 0.1807017
3 -5.508738 0.2286055 2.307389 0.2184998
> kmeans(data, 3, iter.max = 2)$cluster
  [1] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 3 2 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3
 [59] 3 2 2 2 2 2 2 2 3 3 3 3 3 3 3 1 3 3 3 3 3 3 3 3 3 3 3 3 2 2 [illegible]
[117] 1 1 1 2 1 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2 2 1 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 3 3 3
[175] 3 3 3 3 3 3 3 3 3 3 3 3 1 1 3 2 2 2 2 1 2
> |
```

You should use 10 as 10 is the default maximum number of iterations for the built-in kmeans function. If you use 10, the clustering results should be different, and you need to provide a reason for the difference **-1**



Advantages of K-means clustering.

1. Relatively simple to implement.
2. Scales to large data sets.
3. Guarantees convergence.
4. Can warm-start the positions of centroids.
5. Easily adapts to new examples.

Disadvantages of K-means clustering.

1. Choosing K manually
2. Clustering Outliers
3. Clustering data of varying sizes
4. Requires a lot of computer power when the number of iterations are large

This question wasn't answered