test

Huanyu Chen

2024-04-22

```r
# Load necessary libraries
library(mclust)
```

```
## Warning: package 'mclust' was built under R version 4.3.2
```

```
## Package 'mclust' version 6.1
## Type 'citation("mclust")' for citing this R package in publications.
```

```r
library(tidyverse)
```

```
## Warning: package 'tidyr' was built under R version 4.3.2
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.4.4     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x purrr::map()    masks mclust::map()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```r
# Read the data
df <- read.csv("./data/diabetes.csv") |>
  janitor::clean_names() |>
  mutate(across(c(glucose, blood_pressure, skin_thickness, insulin, bmi), ~na_if(.x, 0)))

# Handle missing values
df[is.na(df)] <- 0

# Extract features
X <- df[, c("pregnancies", "glucose", "blood_pressure", "skin_thickness", "insulin", "bmi",
            "diabetes_pedigree_function", "age")]

# Define the EM algorithm function
EM_algorithm <- function(X, num_clusters, max_iter = 100, tol = 1e-6) {
  # Initialize parameters
  model <- Mclust(X, G = num_clusters)

  # Initialize variables to track convergence
  prev_log_likelihood <- -Inf

  for (iter in 1:max_iter) {
    # E-step
    responsibilities <- matrix(0, nrow = nrow(X), ncol = num_clusters)
    for (k in 1:num_clusters) {
      responsibilities[, k] <- model$z[, k] * model$parameters$pro[k]
    }
    responsibilities <- responsibilities / rowSums(responsibilities)

    # M-step
    model <- Mclust(X, G = num_clusters, z = responsibilities)

    # Calculate log-likelihood
    log_likelihood <- sum(log(apply(model$z * model$parameters$pro, 1, sum)))

    # Check for convergence
    if (abs(log_likelihood - prev_log_likelihood) < tol) {
      break
    }

    # Update previous log-likelihood
    prev_log_likelihood <- log_likelihood
  }

  return(model)
}

# Call the EM algorithm
num_clusters <- 2  # Set the number of clusters
gmm_model <- EM_algorithm(X, num_clusters)

# Get the cluster assignments for each sample
clusters <- predict(gmm_model)

# View the feature patterns of the clusters
cluster_means <- t(apply(gmm_model$parameters$mean, 2, function(x) round(x, 2)))
print(cluster_means)
```

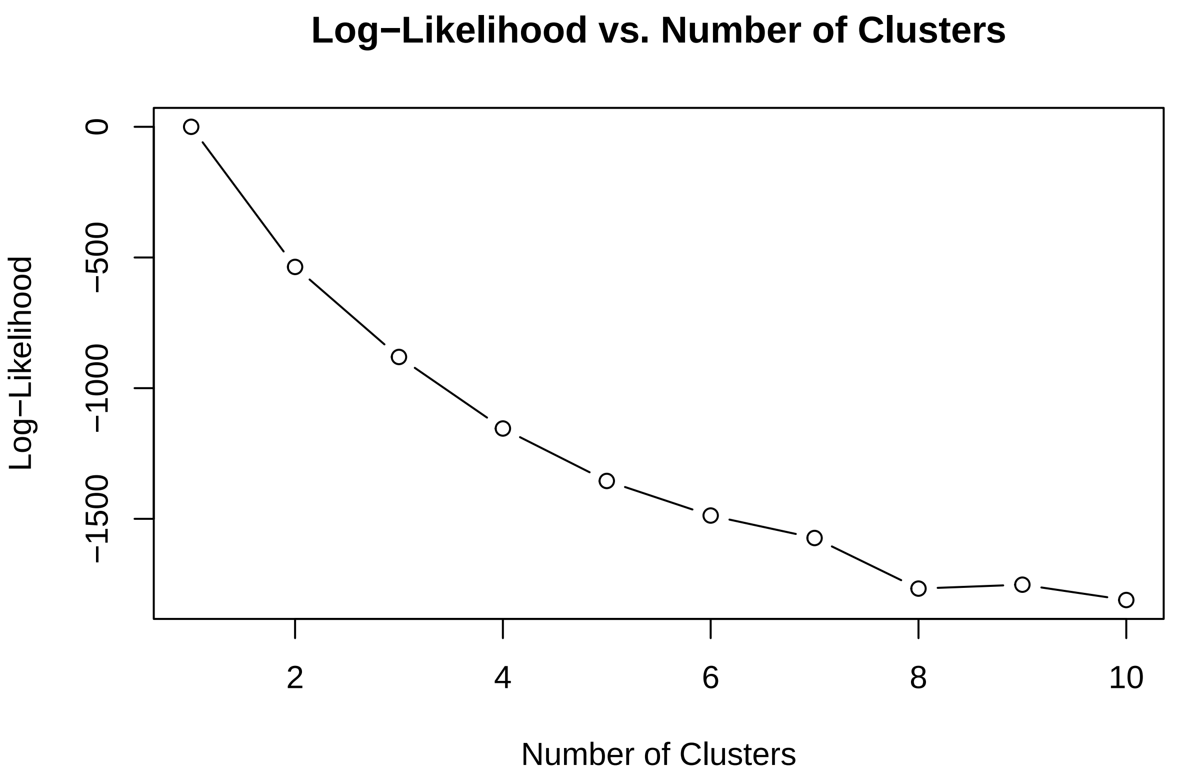
```
##      pregnancies glucose blood_pressure skin_thickness insulin   bmi
## [1,]        5.31  130.41          69.31          19.32   94.14 32.99
## [2,]        2.06  109.29          68.86          22.02   62.32 30.77
##      diabetes_pedigree_function   age
## [1,]                       0.55 39.64
## [2,]                       0.38 25.45
```

```r
# View the mixing weights of the clusters
mixing_weights <- round(gmm_model$parameters$pro, 2)
print(mixing_weights)
```

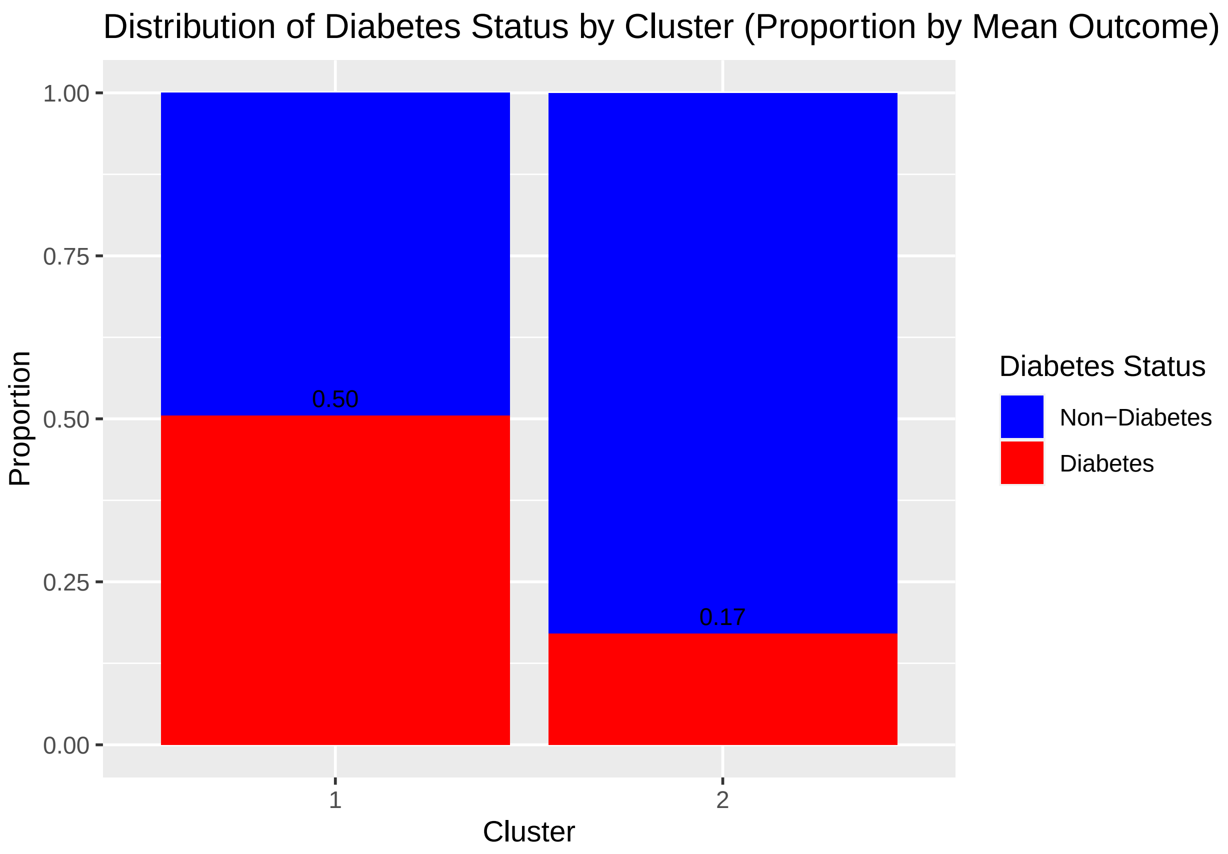
```
## [1] 0.55 0.45
```

```
# Further analysis can be performed based on the cluster assignments,
# for example, analyzing the relationship between cluster assignments and diabetes status
```

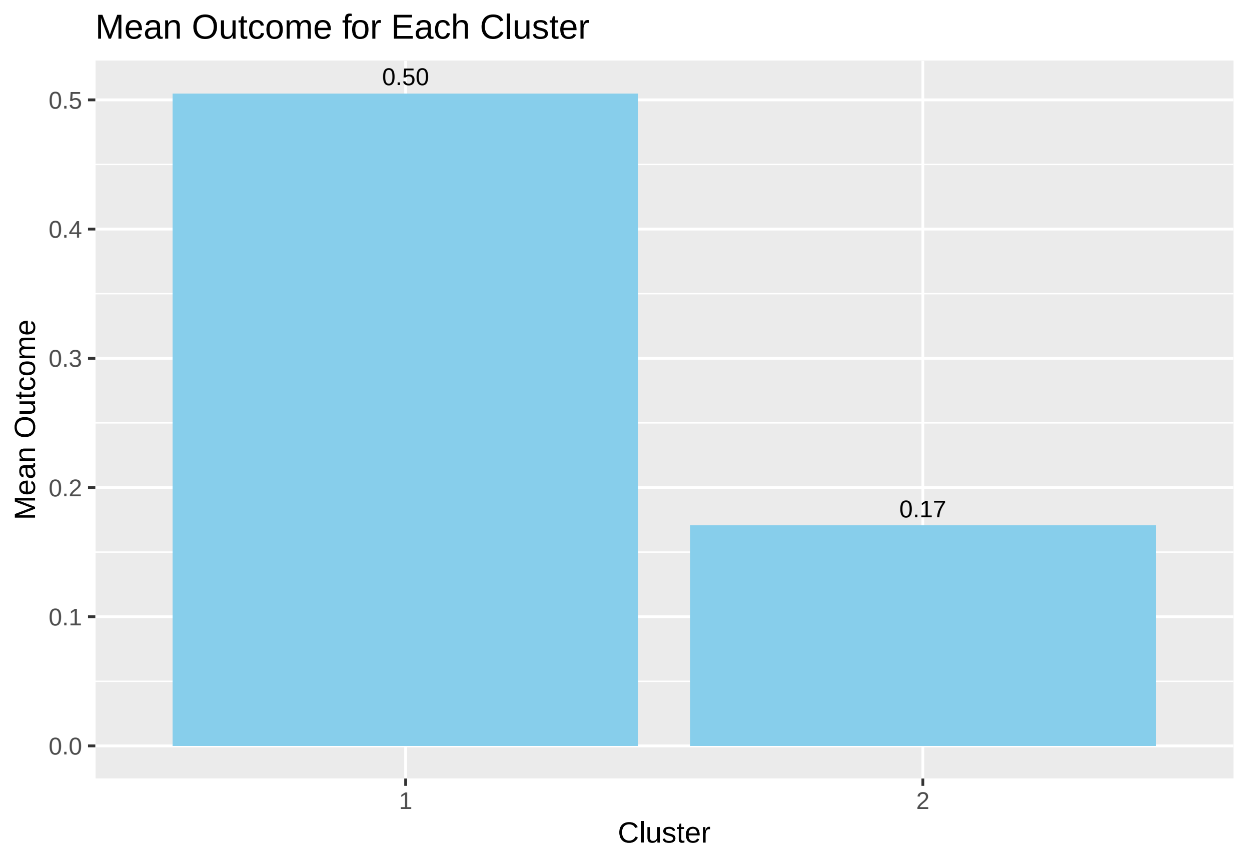
```
# Define a function to run EM algorithm with different numbers of clusters
run_EM_with_different_clusters <- function(X, max_clusters = 10, max_iter = 100, tol = 1e-6) {
  likelihoods <- numeric(max_clusters)

  for (num_clusters in 2:max_clusters) {
    # Run EM algorithm
    model <- EM_algorithm(X, num_clusters, max_iter, tol)

    # Calculate log-likelihood
    log_likelihood <- sum(log(apply(model$z * model$parameters$pro, 1, sum)))

    # Store log-likelihood
    likelihoods[num_clusters] <- log_likelihood
  }

  return(likelihoods)
}

# Call the function to run EM with different numbers of clusters
likelihoods <- run_EM_with_different_clusters(X)

# Plot the likelihoods for different numbers of clusters
plot(1:length(likelihoods), likelihoods, type = "b", xlab = "Number of Clusters", ylab = "Log-Likelihoo
```



```r
library(ggplot2)

# Ensure the cluster and diabetes_status variables are factors
df$cluster <- factor(clusters$classification)
cluster_means <- aggregate(outcome ~ cluster, data = df, FUN = mean)

# Plot the distribution of diabetes status within each cluster
ggplot(df, aes(x = cluster, fill = factor(outcome))) +
  geom_bar(position = "fill") +
  geom_text(data = cluster_means, aes(label = sprintf("%.2f", outcome), y = outcome), vjust = -0.5, colo
  scale_fill_manual(values = c("0" = "blue", "1" = "red"), labels = c("0" = "Non-Diabetes", "1" = "Diabe
  labs(x = "Cluster", y = "Proportion", fill = "Diabetes Status") +
  ggtitle("Distribution of Diabetes Status by Cluster (Proportion by Mean Outcome)")
```

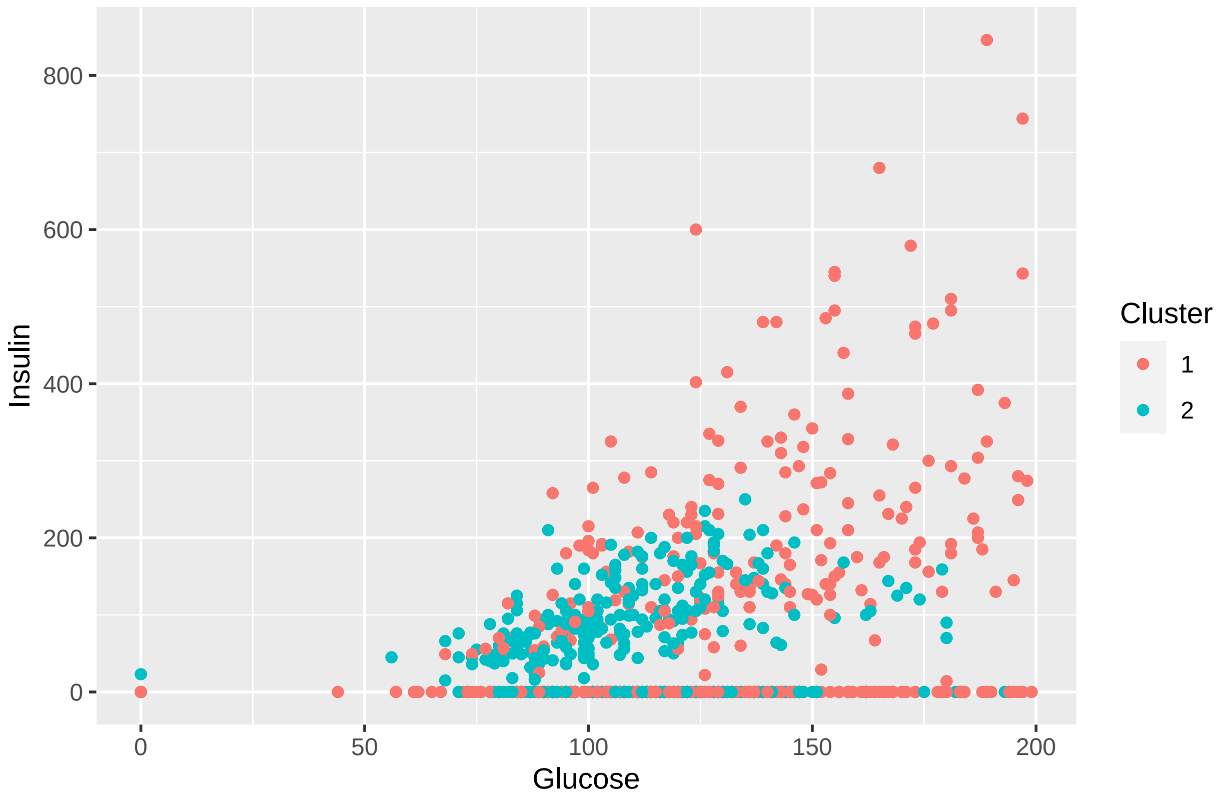


```r
# OR
# Plot the mean outcome for each cluster
ggplot(cluster_means, aes(x = cluster, y = outcome)) +
  geom_bar(stat = "identity", fill = "skyblue") +
  geom_text(aes(label = sprintf("%.2f", outcome)), vjust = -0.5, color = "black", size = 3) +
  labs(x = "Cluster", y = "Mean Outcome") +
  ggtitle("Mean Outcome for Each Cluster")
```



```r
# Plot the relationship between glucose and insulin, colored by cluster
ggplot(df, aes(x = glucose, y = insulin, color = factor(cluster))) +
  geom_point() +
  labs(x = "Glucose", y = "Insulin", color = "Cluster") +
  ggtitle("Relationship between Glucose and Insulin by Cluster")
```

Relationship between Glucose and Insulin by Cluster


Insulin
Glucose
Cluster