To clean and make consistent the college names in your dataset, we can use the dplyr package for data manipulation and stringr for string handling. The script below uses a dictionary-based approach to replace variations of college names with consistent names.

Here’s an R script:

R

Copy code

# Load required libraries

library(dplyr)

library(stringr)

# Sample dataset

data <- data.frame(

ID = 1:24,

College = c(

"UNIVERSITY OF MUMBAI",

"SHIVAJI UNIVERSITY, KOLHAPUR",

"UNIVERSITY",

"DR.BAMU AURANGABAD",

"UNIVERSITY OF MUMBAI",

"UNIVERSITY OF MUMBAI, MUMBAI",

"GOA UNIVERSITY",

"DR BABASAHEB AMBEDKAR MARATHWADA UNIVERSITY",

"KARNATAKA UNIVERSITY DHARWAD",

"DR. BABA SAHEB AMBEDKAR MARATHWADA UNIVERSITY",

"DR. BABA SAHEB AMBEDKAR MARATHWADA UNIVERSITY, CH. SAMBHAJINAGAR",

"DR. BABASAHEB AMBEDKAR MARATHWADA UNIVERSITY AURANGABAD",

"SOLAPUR UNIVERSITY SOLAPUT",

"DR BAMU UNIVERSITY",

"NORTH MAHARASHTRA UNIVERSITY JALGAON",

"SHIVAJI UNIVERSITY, KOLHAPUR",

"DR. BABASAHEB AMBEDKAR MARATHWADA UNIVERSITY",

"PUNJABI UNIVERSITY",

"DR BAMU AURANGABAD",

"SHIVAJI UNIVERSITY KOLHAPUR",

"MUMBAI UNIVERSITY",

"SRTMU NANDED",

"DR BABASAHEB AMBEDKAR MARATHWADA UNIVERSITY, AURANGABAD",

"DR B A M U AURANGABAD"

)

)

# Create a dictionary for replacements

replacement\_dict <- list(

"UNIVERSITY OF MUMBAI" = c("UNIVERSITY OF MUMBAI", "MUMBAI UNIVERSITY", "UNIVERSITY OF MUMBAI, MUMBAI"),

"SHIVAJI UNIVERSITY, KOLHAPUR" = c("SHIVAJI UNIVERSITY, KOLHAPUR", "SHIVAJI UNIVERSITY KOLHAPUR"),

"DR. BABASAHEB AMBEDKAR MARATHWADA UNIVERSITY" = c(

"DR BABASAHEB AMBEDKAR MARATHWADA UNIVERSITY",

"DR. BABA SAHEB AMBEDKAR MARATHWADA UNIVERSITY",

"DR. BABA SAHEB AMBEDKAR MARATHWADA UNIVERSITY, CH. SAMBHAJINAGAR",

"DR. BABASAHEB AMBEDKAR MARATHWADA UNIVERSITY AURANGABAD",

"DR BABASAHEB AMBEDKAR MARATHWADA UNIVERSITY, AURANGABAD",

"DR B A M U AURANGABAD",

"DR.BAMU AURANGABAD",

"DR BAMU AURANGABAD",

"DR BAMU UNIVERSITY"

),

"GOA UNIVERSITY" = c("GOA UNIVERSITY"),

"KARNATAKA UNIVERSITY DHARWAD" = c("KARNATAKA UNIVERSITY DHARWAD"),

"NORTH MAHARASHTRA UNIVERSITY JALGAON" = c("NORTH MAHARASHTRA UNIVERSITY JALGAON"),

"SOLAPUR UNIVERSITY" = c("SOLAPUR UNIVERSITY SOLAPUT"),

"PUNJABI UNIVERSITY" = c("PUNJABI UNIVERSITY"),

"SRTMU NANDED" = c("SRTMU NANDED")

)

# Function to replace inconsistent names

replace\_college\_names <- function(college\_name, dict) {

for (key in names(dict)) {

if (college\_name %in% dict[[key]]) {

return(key)

}

}

return(college\_name) # Return the original name if no match is found

}

# Apply the function to clean the data

data <- data %>%

mutate(College\_Cleaned = sapply(College, replace\_college\_names, dict = replacement\_dict))

# View the cleaned data

print(data)

**How It Works:**

1. **Dictionary-based Replacement:** The replacement\_dict maps all variations of a college name to a consistent version.
2. **String Matching:** The replace\_college\_names function iterates through the dictionary, checks for matches, and replaces inconsistent names.
3. **Cleaned Column:** The new column College\_Cleaned contains the consistent college names.

**Output Example:**

| **ID** | **College** | **College\_Cleaned** |
| --- | --- | --- |
| 1 | UNIVERSITY OF MUMBAI | UNIVERSITY OF MUMBAI |
| 3 | DR. BABA SAHEB AMBEDKAR MARATHWADA UNIVERSITY | DR. BABASAHEB AMBEDKAR MARATHWADA UNIVERSITY |

You can adapt the dictionary by adding more entries for other inconsistencies as needed.

4o

**You said:**

look there are not 24 rows there are total 20000 rows and each student came from differnt colleges and differnt universities write an appropriate R script

**ChatGPT said:**

ChatGPT

For a dataset with 20,000 rows and many different college and university names, manual dictionary mapping won't be feasible. Instead, we can use an automated approach with **string matching** and **clustering**. Below is an R script that uses the **stringdist** package for clustering similar strings and assigning consistent names.

**R Script:**

R

Copy code

# Load required libraries

library(dplyr)

library(stringdist)

library(stringr)

# Example dataset (replace with your actual dataset)

set.seed(42)

data <- data.frame(

ID = 1:20000,

College = sample(c(

"UNIVERSITY OF MUMBAI",

"SHIVAJI UNIVERSITY, KOLHAPUR",

"DR BABASAHEB AMBEDKAR MARATHWADA UNIVERSITY",

"DR BAMU AURANGABAD",

"GOA UNIVERSITY",

"KARNATAKA UNIVERSITY DHARWAD",

"SOLAPUR UNIVERSITY SOLAPUT",

"SRTMU NANDED",

"MUMBAI UNIVERSITY",

"SHIVAJI UNIVERSITY KOLHAPUR"

), 20000, replace = TRUE)

)

# Step 1: Preprocess college names (remove punctuation, convert to lowercase)

data <- data %>%

mutate(College\_Cleaned = str\_to\_lower(College) %>%

str\_replace\_all("[^a-z0-9 ]", "") %>%

str\_squish())

# Step 2: Generate distance matrix for clustering

distance\_matrix <- stringdistmatrix(data$College\_Cleaned, data$College\_Cleaned, method = "jw")

# Step 3: Perform clustering (hierarchical clustering)

hclust\_res <- hclust(as.dist(distance\_matrix), method = "average")

# Step 4: Cut the dendrogram to group similar colleges

# Adjust `k` (number of clusters) based on your dataset

cluster\_assignments <- cutree(hclust\_res, k = 50)

# Step 5: Map clusters to consistent names

# Use the first name in each cluster as the representative

data <- data %>%

mutate(Cluster = cluster\_assignments) %>%

group\_by(Cluster) %>%

mutate(College\_Consistent = first(College)) %>%

ungroup()

# Step 6: Save the cleaned data

write.csv(data, "cleaned\_college\_data.csv", row.names = FALSE)

# View cleaned data

print(data %>% select(ID, College, College\_Consistent) %>% head(20))

**How It Works:**

1. **Preprocessing:**
   * Convert all college names to lowercase and remove special characters (e.g., punctuation) for standardization.
   * Remove extra spaces.
2. **String Matching with stringdistmatrix:**
   * Calculates the similarity between college names using the Jaro-Winkler distance (method = "jw"), which is effective for detecting similar short text variations.
3. **Clustering:**
   * Uses hierarchical clustering (hclust) to group similar names into clusters.
   * Adjust k (number of clusters) based on the variety in your dataset.
4. **Assign Consistent Names:**
   * Assign the first name in each cluster as the representative consistent name for that cluster.
5. **Save Results:**
   * The cleaned dataset is saved to a CSV file.

**Notes:**

* **Scaling Up:** For very large datasets, consider more efficient clustering techniques, such as k-medoids or approximate matching algorithms.
* **Fine-Tuning Clusters:** You can manually inspect the clusters and refine them further if needed.
* **Representative Name:** The first name in each cluster is chosen as the consistent name; you can modify this to use a more meaningful representative.

This script is efficient for large datasets and helps standardize inconsistent names automatically.

4o

**You said:**

the dataset is csv file but already imported using read.csv() methods and the column name of colleges is "UG.INSTITUTE.COLLEGE"

**ChatGPT said:**

ChatGPT

Given that your dataset is already loaded using read.csv() and the column of interest is UG.INSTITUTE.COLLEGE, the following script integrates directly with your dataset for cleaning and clustering college names. It uses the same clustering-based approach.

**R Script for Cleaning College Names**

R

Copy code

# Load required libraries

library(dplyr)

library(stringdist)

# Assuming your dataset is already loaded and named `data`

# Column name for colleges is `UG.INSTITUTE.COLLEGE`

# Step 1: Preprocess college names

data <- data %>%

mutate(

College\_Cleaned = UG.INSTITUTE.COLLEGE %>%

tolower() %>% # Convert to lowercase

stringr::str\_replace\_all("[^a-z0-9 ]", "") %>% # Remove special characters

stringr::str\_squish() # Remove extra whitespace

)

data <- data %>% filter(!is.na(College\_Cleaned) & College\_Cleaned != "")

# Step 2: Generate a distance matrix for clustering

distance\_matrix <- stringdistmatrix(data$College\_Cleaned, data$College\_Cleaned, method = "jw")

# Step 3: Perform clustering

hclust\_res <- hclust(as.dist(distance\_matrix), method = "average")

# Step 4: Cut the dendrogram into clusters

# Adjust `k` based on the variety of colleges in your data

num\_clusters <- 200 # You can change this based on inspection

cluster\_assignments <- cutree(hclust\_res, k = num\_clusters)

# Step 5: Assign consistent names to each cluster

# Use the first name in each cluster as the representative

data <- data %>%

mutate(Cluster = cluster\_assignments) %>%

group\_by(Cluster) %>%

mutate(Consistent\_College\_Name = first(UG.INSTITUTE.COLLEGE)) %>%

ungroup()

# Step 6: Save the cleaned dataset

write.csv(data, "cleaned\_college\_data.csv", row.names = FALSE)

# Optional: Preview the first few rows of cleaned data

print(data %>% select(UG.INSTITUTE.COLLEGE, Consistent\_College\_Name) %>% head(20))

**Explanation of the Script:**

1. **Preprocessing (College\_Cleaned):**
   * Converts all college names to lowercase.
   * Removes special characters (e.g., punctuation).
   * Removes extra spaces.
2. **String Matching (distance\_matrix):**
   * Uses the stringdistmatrix function with the Jaro-Winkler method (method = "jw"), which is effective for fuzzy matching of text data.
3. **Clustering (hclust):**
   * Performs hierarchical clustering to group similar college names.
   * num\_clusters is set to 200, but you can adjust it depending on your data's variety.
4. **Assigning Consistent Names:**
   * For each cluster, assigns the first college name in the group as the representative name (Consistent\_College\_Name).
5. **Saving the Cleaned Dataset:**
   * The cleaned dataset is saved as cleaned\_college\_data.csv for further analysis.

**Customization:**

* **num\_clusters:** Adjust the number of clusters based on your data. Start with a higher number (e.g., 500) for fine-grained grouping and reduce as needed.
* **Representative Name:** Modify the logic to use a different representative for clusters if needed (e.g., the most frequent name in the cluster).

To standardize the ug.colleges column, which contains inconsistencies due to human input errors, you can use **fuzzy matching** and **string similarity algorithms** to map the raw college names to a consistent set of standard names.

Here’s how you can handle this in R, step by step:

### ****1. Load and Inspect the Data****

First, load your dataset and inspect the column for unique college names.

R

Copy code

# Load the dataset

data <- read.csv("your\_dataset.csv")

# Check unique college names

unique\_colleges <- unique(data$ug.colleges)

length(unique\_colleges) # Count of unique names

### ****2. Create a Standardized Reference List****

Create or load a reference list of the ~500 colleges.

R

Copy code

# Example reference list

reference\_colleges <- c("Harvard University", "MIT", "Stanford University", "UC Berkeley")

### ****3. Fuzzy Matching for Standardization****

Use the **stringdist** package for string similarity measures (e.g., Levenshtein distance) to map inconsistent names to the closest match in the reference list.

#### Install Required Libraries:

R

Copy code

install.packages("stringdist")

install.packages("dplyr")

#### Fuzzy Matching Workflow:

R

Copy code

library(stringdist)

library(dplyr)

# Function to find the closest match

get\_closest\_match <- function(input\_name, reference\_list) {

distances <- stringdist::stringdist(input\_name, reference\_list, method = "jw") # Jaro-Winkler method

reference\_list[which.min(distances)]

}

# Apply the function to the dataset

data$standardized\_colleges <- sapply(data$ug.colleges, get\_closest\_match, reference\_list = reference\_colleges)

### ****4. Handle Abbreviations and Synonyms****

Manually map common abbreviations or known synonyms (e.g., "MIT" → "Massachusetts Institute of Technology").

#### Example:

R

Copy code

# Custom abbreviation mapping

abbreviations <- list(

"MIT" = "Massachusetts Institute of Technology",

"UC Berkeley" = "University of California, Berkeley",

"Stanford" = "Stanford University"

)

# Replace abbreviations

data$standardized\_colleges <- recode(data$standardized\_colleges, !!!abbreviations)

### ****5. Verify and Adjust Matches****

Review the results for mismatches or edge cases.

#### Example:

Identify unmatched colleges:

R

Copy code

unmatched <- setdiff(unique(data$ug.colleges), reference\_colleges)

write.csv(unmatched, "unmatched\_colleges.csv", row.names = FALSE)

You can manually inspect and correct the unmatched entries and rerun the mapping.

### ****6. Automate Improvements with Clustering****

If your dataset is very large, clustering similar names can reduce manual intervention. Use the **stringdistmatrix** function for this.

R

Copy code

# Compute distance matrix

distance\_matrix <- stringdist::stringdistmatrix(unique\_colleges, method = "jw")

# Perform hierarchical clustering

hclust\_result <- hclust(as.dist(distance\_matrix))

plot(hclust\_result) # Visualize clusters

Group names within the same cluster and map them to a standard college name.

### ****7. Save the Cleaned Data****

Export the dataset with standardized college names.

R

Copy code

write.csv(data, "cleaned\_dataset.csv", row.names = FALSE)

### ****Summary****

1. Use fuzzy matching (e.g., stringdist) to find the closest match for each college name.
2. Manually map abbreviations and known synonyms.
3. Review unmatched entries and iteratively improve.
4. Use clustering for further automation in large datasets.