

SHETH L.U.J AND SIR M.V COLLEGE

PRACTICAL NO:11

11 Reshaping data using pivot_longer()/pivot_wider() (R).

CODE:

```
library(dplyr)
```

```
library(tidyr)
```

```
#
```

```
=====
```

```
=====
```

```
# 1. SETUP: Read Your Dataset
```

```
df <- read.csv("C:/Users/info/Downloads/Retail Product_ - Retail Product.csv",
```

```
    na.strings = c("", "NA")) %>%
```

```
    mutate(ProductID = row_number()) %>%
```

```
    select(ProductID, Category, Price, Discount)
```

```
print("--- 1. Original Wide Data ---")
```

```
print(head(df))
```

```
#
```

```
=====
```

```
=====
```

```
# 2. PIVOT_LONGER (Wide → Long)
```

```
#
```

```
=====
```

```
=====
```

```
long_df <- df %>%
```

```
    pivot_longer(
```

```
        cols = c(Price, Discount),
```

```
        names_to = "Metric",
```

```
        values_to = "Value"
```

```
)
```

```
print("--- 2. Long Format ---")
```

```
print(head(long_df, 6))
```

```
#
```

```
=====
```

```
=====
```

```
# 3. PIVOT_WIDER (Long → Wide)
```

SHETH L.U.J AND SIR M.V COLLEGE

```
#  
=====  
=====  
  
wide_df <- long_df %>%  
pivot_wider(  
  names_from = Metric,  
  values_from = Value  
)  
  
print("--- 3. Wide Format (Back to Original) ---")  
print(head(wide_df))  
  
#  
=====  
=====  
# 4. ADVANCED EXAMPLE (Category Pivot)  
#  
=====  
=====  
  
df_clean <- df %>%  
  mutate(Category = ifelse(is.na(Category), "Unknown", Category))  
  
category_pivot <- df_clean %>%  
  select(ProductID, Category, Price) %>%  
  pivot_wider(  
    names_from = Category,  
    values_from = Price  
)  
  
print("--- 4. Category Pivot (Spreading Categories) ---")  
print(head(category_pivot))
```

OUTPUT:

SHETH L.U.J AND SIR M.V COLLEGE

SHETH L.U.J AND SIR M.V COLLEGE

PRACTICAL NO:12

12 Combining datasets vertically (concatenation) using rbind() (R).

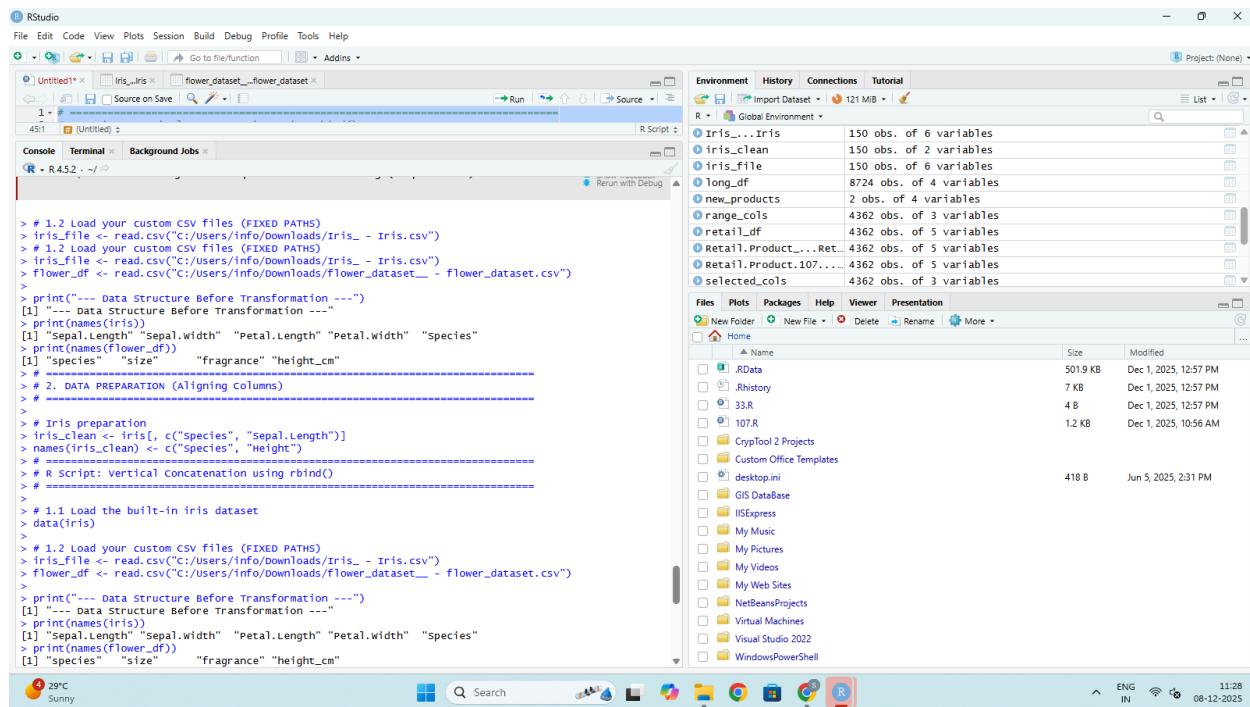
CODE:

```
#  
=====  
=====  
# R Script: Vertical Concatenation using rbind()  
#  
=====  
=====  
  
# 1.1 Load the built-in iris dataset  
data(iris)  
  
# 1.2 Load your custom CSV files (FIXED PATHS)  
iris_file <- read.csv("C:/Users/info/Downloads/Iris_ - Iris.csv")  
flower_df <- read.csv("C:/Users/info/Downloads/flower_dataset__ - flower_dataset.csv")  
  
print("--- Data Structure Before Transformation ---")  
print(names(iris))  
print(names(flower_df))  
  
#  
=====  
=====  
# 2. DATA PREPARATION (Aligning Columns)  
#  
=====  
=====  
  
# Iris preparation  
iris_clean <- iris[, c("Species", "Sepal.Length")]  
names(iris_clean) <- c("Species", "Height")  
  
# Flower dataset preparation  
flower_clean <- flower_df[, c("species", "height_cm")]  
names(flower_clean) <- c("Species", "Height")  
  
# Convert to numeric  
iris_clean$Height <- as.numeric(iris_clean$Height)  
flower_clean$Height <- as.numeric(flower_clean$Height)
```

SHETH L.U.J AND SIR M.V COLLEGE

```
#  
=====  
=====  
# 3. VERTICAL COMBINATION  
#  
=====  
=====  
  
combined_data <- rbind(iris_clean, flower_clean)  
  
print("--- Combined Data Summary ---")  
print(paste("Iris rows:", nrow(iris_clean)))  
print(paste("Flower rows:", nrow(flower_clean)))  
print(paste("Total rows:", nrow(combined_data)))  
print("--- Preview ---")  
print(head(combined_data))  
print(tail(combined_data))
```

OUTPUT:



The screenshot shows the RStudio interface with the following details:

- Console:** Displays the R script and its output. The output shows the loading of CSV files, printing of data structures before transformation, and the final combined dataset.
- Environment:** Shows the global environment with objects like Iris...Iris, iris_clean, iris_file, Tong_df, range_cols, retail_df, Retail.Product..., Retail.Product.107..., and selected_cols.
- File Explorer:** Shows the file system structure including RData, History, 33.R, 107.R, CryTool 2 Projects, Custom Office Templates, desktop.ini, GIS DataBase, IISExpress, My Music, My Pictures, My Videos, My Web Sites, NetBeansProjects, Virtual Machines, Visual Studio 2022, and WindowsPowerShell.
- Bottom Status Bar:** Shows system information including battery level (29%), CPU usage (Sunny), network (ENG IN), and date/time (08-12-2025 11:28).

SHETH L.U.J AND SIR M.V COLLEGE

```

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Console Terminal Background Jobs
<R> R4.5.2 - ~/Documents
> print(names(iris_clean))
[1] "species"      "size"        "fragrance"    "height_cm"
>
> # =====
> # 2. DATA PREPARATION (Aligning Columns)
> # =====
>
> # Iris preparation
> iris_clean <- iris[, c("Species", "Sepal.Length")]
> names(iris_clean) <- c("Species", "Height")
>
> # convert to numeric
> iris_clean$Height <- as.numeric(iris_clean$Height)
> flower_clean$Height <- as.numeric(flower_clean$Height)
>
> # =====
> # 3. VERTICAL COMBINATION
> # =====
>
> combined_data <- rbind(iris_clean, flower_clean)
>
> print("--- Combined Data Summary ---")
[1] "--- Combined Data Summary ---"
> print(paste("Iris rows:", nrow(iris_clean)))
[1] "Iris rows: 150"
> print(paste("Flower rows:", nrow(flower_clean)))
[1] "Flower rows: 10000"
> print(paste("Total rows:", nrow(combined_data)))
[1] "Total rows: 10150"
> print("--- Preview ---")
[1] "--- Preview ---"
> print(head(combined_data))
  Species Height
1  setosa   5.1
2  setosa   4.9
3  setosa   4.7
4  setosa   4.6
5  setosa   5.0
6  setosa   5.4
> print(tail(combined_data))
  Species Height
10145     rose  87.69
10146   hibiscus 109.52
10147  shoeblack plant 145.23
10148   hibiscus 126.69
10149  shoeblack plant 77.62
10150     rose  88.11
> C

```

29°C Sunny 11:28 08-12-2025


```

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Console Terminal Background Jobs
<R> R4.5.2 - ~/Documents
> print(names(iris_clean)) <- c('Species', 'Height')
>
> # convert to numeric
> iris_clean$Height <- as.numeric(iris_clean$Height)
> flower_clean$Height <- as.numeric(flower_clean$Height)
>
> # =====
> # 3. VERTICAL COMBINATION
> # =====
>
> combined_data <- rbind(iris_clean, flower_clean)
>
> print("--- Combined Data Summary ---")
[1] "--- Combined Data Summary ---"
> print(paste("Iris rows:", nrow(iris_clean)))
[1] "Iris rows: 150"
> print(paste("Flower rows:", nrow(flower_clean)))
[1] "Flower rows: 10000"
> print(paste("Total rows:", nrow(combined_data)))
[1] "Total rows: 10150"
> print("--- Preview ---")
[1] "--- Preview ---"
> print(head(combined_data))
  Species Height
1  setosa   5.1
2  setosa   4.9
3  setosa   4.7
4  setosa   4.6
5  setosa   5.0
6  setosa   5.4
> print(tail(combined_data))
  Species Height
10145     rose  87.69
10146   hibiscus 109.52
10147  shoeblack plant 145.23
10148   hibiscus 126.69
10149  shoeblack plant 77.62
10150     rose  88.11
> C

```

29°C Sunny 11:28 08-12-2025

PRACTICAL NO:13

13. Identifying and handling duplicates using distinct() (R studio).

CODE:

SHETH L.U.J AND SIR M.V COLLEGE

```
#  
=====  
=====  
# R Script: Identifying and Handling Duplicates  
# Function: distinct() from the dplyr package  
#  
=====  
=====  
  
# Load library  
library(dplyr)  
  
#  
=====  
=====  
# 1. SETUP: Create a Dataset with Intentional Duplicates  
  
orders_df <- data.frame(  
  OrderID = c(101, 102, 102, 103, 104, 101, 104),  
  Customer = c("NISHITA", "YUKTA", "SIYA", "NANDINI", "SIMRAN", "PURVI", "GAZALA"),  
  Product = c("Laptop", "Phone", "Phone", "Tablet", "Monitor", "Laptop", "Mouse")  
)  
  
print("--- 1. Original Dataset (Note 7 rows) ---")  
print(orders_df)  
  
#  
=====  
=====  
# 2. IDENTIFYING DUPLICATES  
#  
=====  
=====  
  
duplicates_report <- orders_df %>%  
  group_by(OrderID, Customer, Product) %>%  
  count() %>%      # Count occurrences  
  filter(n > 1)      # Keep only rows that appear more than once  
  
print("--- 2. Identification Report (Rows that are duplicated) ---")  
print(duplicates_report)
```

SHETH L.U.J AND SIR M.V COLLEGE

```
#  
=====  
=====  
# 3. HANDLING EXACT DUPLICATES  
#  
=====  
=====  
  
clean_exact <- orders_df %>%  
  distinct()      # Removes only exact row matches  
  
print("--- 3. Removed Exact Duplicates (distinct) ---")  
print(clean_exact)  
  
#  
=====  
=====  
# 4. HANDLING DUPLICATES BASED ON SPECIFIC COLUMN  
#  
=====  
=====  
  
unique_customers <- orders_df %>%  
  distinct(Customer, .keep_all = TRUE)  # Keeps first appearance of each customer  
  
print("--- 4. Unique Customers Only (Partial duplicates removed) ---")  
print(unique_customers)
```

OUTPUT:

SHETH L.U.J AND SIR M.V COLLEGE

The screenshot shows the RStudio interface with the following details:

- File Menu:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Project:** Project (None).
- Source Editor:** Displays an R script for identifying and handling duplicates in a dataset. The script includes code to create a dataset with intentional duplicates, identify them, and filter out rows that appear more than once.
- Environment Tab:** Shows the global environment with various objects and their characteristics.
- File Explorer:** Shows the file structure of the current project, including RData files, history files, and various workspace files.
- System Tray:** Shows the date (08-12-2024), time (11:42), battery status (IN), signal strength (29°C, Sunny), and other system icons.

```
RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Project: (None)

Source
Console Terminal Background Jobs x

R - R4.2.2 : ~/r/r4.2.2/R/x86_64-pc-linux-gnu-library/4.2
2 102 Bebo Phone
3 103 Charlie Tablet
4 104 Dinesh Monitor
> # =====
> # R Script: Identifying and Handling duplicates
> # Function: distinct() from the dplyr package
> #
>
> # Load Library
> library(dplyr)
>
> # =====
> # 1. SETUP: Create a dataset with Intentional Duplicates
>
> orders_df <- data.frame(
+   orderId = c(101, 102, 103, 104, 101, 104),
+   customer = c("NISHITA", "YUKTA", "SIVA", "NANDINI", "SIMRAN", "PURVI", "GAZALA"),
+   product = c("Laptop", "Phone", "Phone", "Tablet", "Monitor", "Laptop", "Mouse")
+ )
>
> print("--- 1. original dataset (Note 7 rows) ---")
[1] "--- 1. original dataset (Note 7 rows) ---"
> print(orders_df)
  orderId customer product
1      101    NISHITA Laptop
2      102     YUKTA  Phone
3      102      SIVA  Phone
4      103    NANDINI Tablet
5      104    SIMRAN Monitor
6      101      PURVI Laptop
7      104      GAZALA Mouse
> #
> # =====
> # 2. IDENTIFYING DUPLICATES
> #
>
> duplicates_report <- orders_df %>%
+   group_by(orderId, customer, product) %>%
+   count() %% # count occurrences
+   filter(n > 1) # keep only rows that appear more than once
>
```

SHETH L.U.J AND SIR M.V COLLEGE

PRACTICAL NO:6

```
#  
=====  
=====  
# R Script: Extracting Date Components using lubridate  
#  
=====  
=====  
  
# Install and load necessary libraries  
install.packages("lubridate")  
library(lubridate)  
library(dplyr)  
  
#  
=====  
=====  
# 1. SETUP: Create Sample Data  
#  
=====  
=====  
  
dates_df <- data.frame(  
  Event_ID = 1:4,  
  Date_String = c("2023-01-15", "2023-10-31", "2024-02-29", "2024-12-25"))
```

RIYA RANE
S107 SYCS

SHEETH L.U.J AND SIR M.V COLLEGE

```
)  
#  
=====  
=====  
# 2. PARSE AND EXTRACT  
#  
=====  
=====  
processed_data <- dates_df %>%  
  mutate(  
    # A. Parsing: Convert text to actual Date  
    Actual_Date = ymd(Date_String),  
  
    # B. Extraction Components  
    Year_Num      = year(Actual_Date),      # Year (e.g., 2023)  
    Month_Num     = month(Actual_Date),     # Month number (1–12)  
    Month_Name    = month(Actual_Date, label = TRUE),    # Month abbreviation  
    Day_Num       = day(Actual_Date),       # Day (1–31)  
    Weekday_Num   = wday(Actual_Date),     # Day of week (1=Sun)  
    Weekday_Name  = wday(Actual_Date, label = TRUE, abbr = FALSE), # Full Name  
    Quarter       = quarter(Actual_Date),    # Quarter (1–4)  
    Day_of_Year   = yday(Actual_Date)      # Day of the year (1–366)  
)  
  
print("--- Data with Extracted Date Components ---")  
print(processed_data)  
  
#  
=====  
=====  
# 3. SYSTEM DATE & TIME (now)  
#  
=====  
=====  
current_time <- now()  
  
print("--- Current Time Extraction ---")  
print(paste("Current Year:", year(current_time)))  
print(paste("Current Hour:", hour(current_time)))  
print(paste("Current Minute:", minute(current_time)))
```

SHETH L.U.J AND SIR M.V COLLEGE

OUTPUT:

The screenshot shows the RStudio interface with the following details:

- File Menu:** File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, Help.
- Project:** Project: (None)
- Code Editor:** An R script named "Untitled" is open, containing code to extract date components from a dataset. The code includes imports for dplyr and lubridate, data preparation, and extraction logic for year, month, day, and week components.
- Console:** The console output shows the execution of the script, resulting in a data frame with columns: Event_ID, Date_String, Actual_Date, Year_Num, Month_Num, Month_Name, Day_Num, and Day_of_Year.
- Environment:** The environment pane shows various objects: processed_data, range_cols, retail_df, Retail.Product..., selected_cols, shopping_behavior_upd, starts_with_r, and unique_customers, each with their respective dimensions.
- File Explorer:** The file explorer shows the project structure with files like RData, Rhistory, 33.R, 107.R, CrypTool2 Projects, Custom Office Templates, desktop.ini, GIS DataBase, IISExpress, My Music, My Pictures, My Videos, My Web Sites, NetBeansProjects, Virtual Machines, Visual Studio 2022, and WindowsPowerShell.
- Bottom Bar:** Includes icons for file operations, search, and system status.

The screenshot shows the RStudio interface with the following details:

- Code Editor:** Displays R code for extracting date components from a dataset. The code includes functions for calculating the day of the year, week day, quarter, and month, as well as printing the results.
- Console:** Shows the output of the R code, including the extracted date components for four rows of data and current system time information.
- Environment:** A pane listing R objects and their characteristics, such as `processed_data` (4 obs. of 11 variables), `range_cols` (4362 obs. of 3 variables), and `unique_customers` (7 obs. of 3 variables).
- File Explorer:** Shows the local file structure, including RData files, history files, and various projects and files from desktop environments like CrypTool 2, Custom Office Templates, and My Pictures.
- Snipping Tool:** A floating window showing a screenshot of the RStudio interface.

PRACTICAL NO:15

CODE

SHETH L.U.J AND SIR M.V COLLEGE

```
#  
=====  
=====  
# R Script: Generating Basic Summaries  
# Functions: str() and summary()  
# Dataset: Retail Product Data (Simulated)  
#  
=====  
=====  
  
#  
=====  
=====  
# 1. SETUP: Create Sample Data  
#  
=====  
=====  
  
retail_df <- data.frame(  
  ID = 1:6,  
  Category = c("Electronics", "Home", "Electronics", "Clothing", "Home", "Clothing"),  
  Price = c(500.50, 45.00, 900.00, NA, 300.00, 25.00), # Note NA  
  In_Stock = c(TRUE, TRUE, FALSE, TRUE, FALSE, TRUE),  
  Rating = c(4.5, 3.8, 4.9, 4.0, 3.5, 4.2)  
)  
  
print("--- Data Loaded ---")  
print(retail_df)  
  
#  
=====  
=====  
# 2. USING str() — Structure of the Data  
#  
=====  
=====  
  
print("--- OUTPUT OF str() ---")  
str(retail_df)  
  
#  
=====  
=====  
# 3. USING summary() — Statistical Summary
```

SHETH L.U.J AND SIR M.V COLLEGE

```
#  
=====  
=====  
  
print("--- OUTPUT OF summary() [Before Factor Conversion] ---")  
summary(retail_df)  
  
#  
=====  
=====  
# 4. IMPROVING summary() WITH FACTORS  
#  
=====  
=====  
  
# Convert Category from character → factor for meaningful summary counts  
retail_df$Category <- as.factor(retail_df$Category)  
  
print("--- OUTPUT OF summary() [After Factor Conversion] ---")  
summary(retail_df)  
  
#  
=====  
=====  
# 5. Accessing Specific Summaries  
#  
=====  
=====  
  
avg_rating <- mean(retail_df$Rating)          # Average rating  
max_price <- max(retail_df$Price, na.rm = TRUE)    # Ignore NA  
  
print(paste("Average Rating:", avg_rating))  
print(paste("Highest Price:", max_price))
```

OUTPUT:

SHETH L.U.J AND SIR M.V COLLEGE

The screenshot shows an RStudio interface with several open panes:

- Source** pane: Displays the R code used to create sample data for a retail dataset. The code includes setting up the environment, creating a data frame with columns for ID, Category, Price, In_Stock, and Rating, and then printing the data frame to the console.
- Environment** pane: Shows the global environment with various objects listed, such as range_cols, retail_df, and unique_customers, each with their respective sizes and types.
- Files** pane: Lists the files and projects in the current workspace, including RData, Rhistory, 33.R, 107.R, CrypTool 2 Projects, Custom Office Templates, desktop.ini, GIS DataBase, IISExpress, My Music, My Pictures, My Videos, My Web Sites, NetBeansProjects, Virtual Machines, Visual Studio 2022, and WindowsPowerShell.

The screenshot shows the RStudio interface with the following details:

- Console:** Displays the R session history. The user has run several commands, including printing the summary of the `retail` dataset before and after factor conversion, and printing the summary of the `retail_df` dataset.
- Environment:** Shows the global environment with various objects listed by name, size, and type.
- File Browser:** Shows the local file system structure, including RData files, history files, and various projects and files from desktop environments like CrypTool, Custom Office Templates, and My Pictures.

SHETH L.U.J AND SIR M.V COLLEGE

The screenshot shows a Windows desktop environment. In the foreground, there is a RStudio window displaying R code and its output. The code is related to data summarization and factor conversion. In the background, a File Explorer window is open, showing the contents of a 'Home' folder. The folder contains several files and subfolders, including '.RData', '.History', '33.R', '107.R', 'CryptTool 2 Projects', 'Custom Office Templates', 'desktop.ini', 'GIS DataBase', 'IISExpress', 'My Music', 'My Pictures', 'My Videos', 'My Web Sites', 'NetBeansProjects', 'Virtual Machines', 'Visual Studio 2022', and 'WindowsPowerShell'. The RStudio window has tabs for 'Files', 'Plots', 'Packages', 'Help', 'Viewer', and 'Presentation'. The status bar at the bottom of the RStudio window shows 'ENG' and '11:56'.

```
> # 4. IMPROVING summary() WITH FACTORS
> # =====
> # convert Category from character - factor for meaningful summary counts
> retail_df$Category <- as.factor(retail_df$Category)
>
> print(" --- OUTPUT OF summary() [After Factor Conversion] ---")
[1] " --- OUTPUT OF summary() [After Factor Conversion] ---"
> summary(retail_df)
   ID          Category      Price     In_Stock      Rating
Min. :1.00   Clothing    :2  Min. : 25.0  Mode :logical Min. :3.500
1st Qu.:2.25 Electronics:2  1st Qu.: 45.0  FALSE:2   1st Qu.:3.850
Median :3.50  Home       :2  Median :300.0  TRUE :4    Median :4.100
Mean   :3.50                           Mean  :354.1   Mean  :4.150
3rd Qu.:4.75                           3rd Qu.:500.5 3rd Qu.:4.425
Max.  :6.00                           Max. :900.0   Max.  :4.900
NA's   :1
```

```
> # =====
> # 5. Accessing Specific Summaries
> # =====
>
> avg_rating <- mean(retail_df$Rating)           # Average rating
> max_price <- max(retail_df$Price, na.rm = TRUE) # Ignore NA
>
> print(paste("Average Rating:", avg_rating))
[1] "Average Rating: 4.15"
> print(paste("Highest Price:", max_price))
[1] "Highest Price: 900"
> C
```