Week 3 Assignment

MD SIMON CHOWDHURY

2024-09-21

1. Normalization in R

Normalization is the process of organizing data to minimize redundancy. In this section, we demonstrate normalization by splitting a single dataframe into three related dataframes.

```
# Original Data
student_data <- data.frame(
  student_id = c(1, 2, 3),
  student_name = c("Alice", "Bob", "Charlie"),
  course = c("Math", "Science", "Math"),
  instructor = c("Dr. Smith", "Dr. Jones", "Dr. Smith")
)

student_data
```

```
##   student_id student_name  course instructor
## 1          1        Alice    Math  Dr. Smith
## 2          2          Bob Science  Dr. Jones
## 3          3      Charlie    Math  Dr. Smith
```

```
# Normalized Data

# Students Table
students <- data.frame(
  student_id = c(1, 2, 3),
  student_name = c("Alice", "Bob", "Charlie")
)

students
```

```
##   student_id student_name
## 1          1        Alice
## 2          2          Bob
## 3          3      Charlie
```

```
# Courses Table
courses <- data.frame(
  course_id = c(1, 2),
  course_name = c("Math", "Science"),
  instructor = c("Dr. Smith", "Dr. Jones")
)

courses
```

```
##   course_id course_name instructor
## 1         1        Math  Dr. Smith
## 2         2     Science  Dr. Jones
```

```
# Enrollments Table
enrollments <- data.frame(
  student_id = c(1, 2, 3),
  course_id = c(1, 2, 1)
)

enrollments
```

```
##   student_id course_id
## 1          1         1
## 2          2         2
## 3          3         1
```

2. Character Manipulation

Using the College Majors dataset from FiveThirtyEight, we identify the majors that contain either “DATA” or “STATISTICS.”

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
# Load data from URL
url <- "https://raw.githubusercontent.com/fivethirtyeight/data/master/college-majors/majors-l
ist.csv"
majors <- read.csv(url)

# Filter majors that contain "DATA" or "STATISTICS"
selected_majors <- majors %>%
  filter(grepl("DATA|STATISTICS", Major, ignore.case = TRUE))

selected_majors
```

```
##   FOD1P                                          Major          Major_Category
## 1  6212 MANAGEMENT INFORMATION SYSTEMS AND STATISTICS               Business
## 2  2101      COMPUTER PROGRAMMING AND DATA PROCESSING Computers & Mathematics
## 3  3702               STATISTICS AND DECISION SCIENCE Computers & Mathematics
```

3. Regular Expressions Descriptions

Below are explanations of what the following regular expressions will match:

1. (.)\1\1: This matches any character repeated three times consecutively, such as “aaa” or “111”. example:

```
example1 <- c("aaa", "bbb", "111", "abc")
grepl("(.)\\1\\1", example1)
```

```
## [1]  TRUE  TRUE  TRUE FALSE
```

2. “(.)(.)\2\1”: This matches a four-character string where the first two characters are mirrored by the last two, like “abba”. example:

```
example2 <- c("abba", "12321", "abcd")
grepl("(.)(.)\\2\\1", example2)
```

```
## [1]  TRUE FALSE FALSE
```

3. (..)\1: This matches any two-character sequence that repeats, like “abab” or “1212”. example:

```
example3 <- c("abab", "1212", "abcd")
grepl("(..)\\1", example3)
```

```
## [1]  TRUE  TRUE FALSE
```

4. “(.).\1.\1”: This matches a five-character string where the first, third, and fifth characters are the same, like “ababa” or “x2x2x”. example:

```
example4 <- c("ababa", "x2x2x", "abcde")
grepl("(.).\\1.\\1", example4)
```

```
## [1]  TRUE  TRUE FALSE
```

5. “(.)(.)(.).*\3\2\1”: This matches any string where the first three characters are repeated in reverse order somewhere later in the string, like “abc123cba”. example:

```
example5 <- c("abc123cba", "xyz321zyx", "abcdef")
grepl("(.)(.)(.).*\\3\\2\\1", example5)
```

```
## [1]  TRUE  TRUE FALSE
```

4. Constructing Regular Expressions with Examples

Here are the regular expressions constructed to match specific patterns, along with examples:

1.Start and end with the same character:

```
pattern1 <- "^(.).*\1$"
example6 <- c("level", "radar", "test", "abba")
grepl(pattern1, example6)
```

```
## [1] FALSE FALSE FALSE FALSE
```

2.Contain a repeated pair of letters:

```
pattern2 <- "(..).*\\1"
example7 <- c("church", "abab", "banana", "test")
grepl(pattern2, example7)
```

```
## [1]  TRUE  TRUE FALSE FALSE
```

3.Contain one letter repeated in at least three places:

```
pattern3 <- "(.).*\\1.*\\1"
example8 <- c("eleven", "banana", "abracadabra", "test")
grepl(pattern3, example8)
```

```
## [1]  TRUE FALSE  TRUE FALSE
```