Data Wrangling Lab Report

Lab #28 - Data Wrangling Techniques

Ahmad Mukhtar

National University of Sciences and Technology (NUST)
Chip Design Centre (NCDC), Islamabad, Pakistan

October 13, 2024

Contents

0.1 Task 1: Interactive Regex Tutorial

0.1.1 Objective

In this task, the aim was to complete a series of 15 exercises to explore the fundamentals of regular expressions (regex), including character matching, quantifiers, groups, and anchors.

0.1.2 Screenshots of Completed Exercises

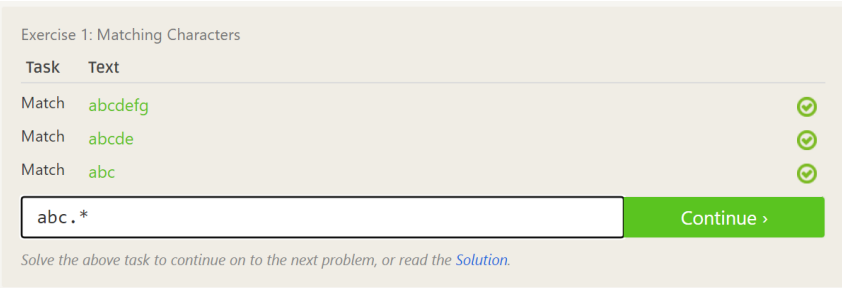

Figure 1: Exercise 1: Matching characters

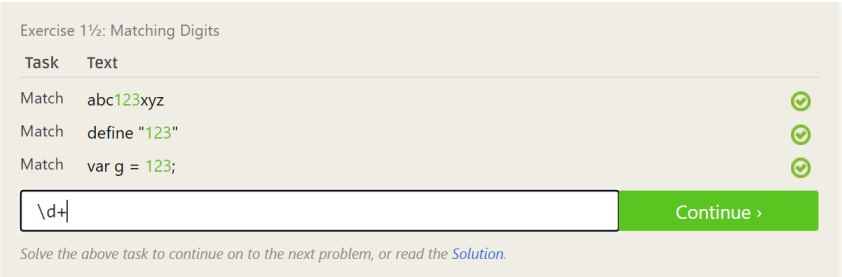

Figure 2: Exercise 2: Character Ranges and Classes

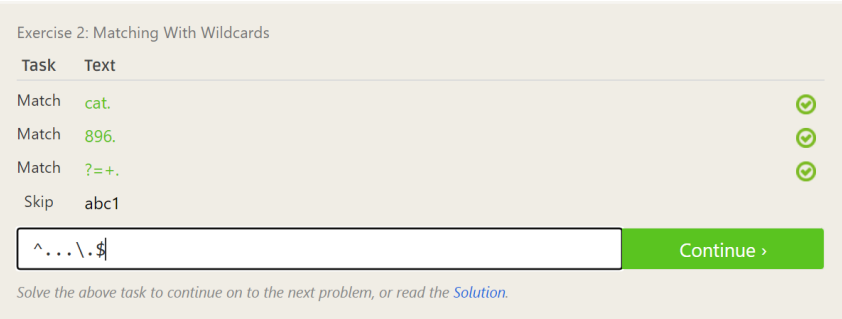

Figure 3: Exercise 3: Using Quantifiers

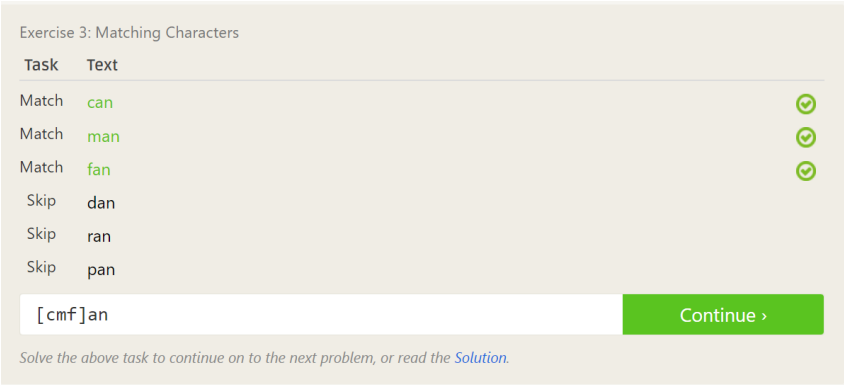

Figure 4: Exercise 4: Escaping Special Characters

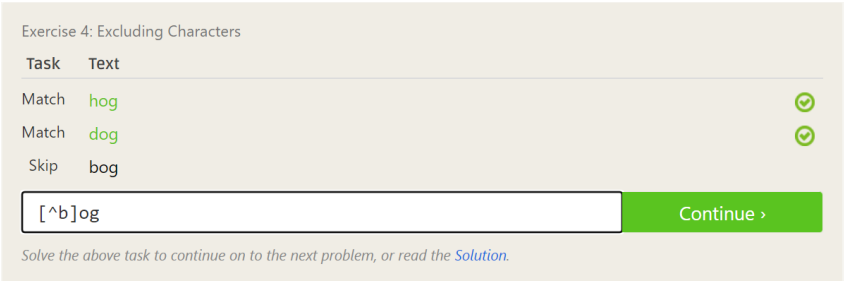

Figure 5: Exercise 5: Anchors

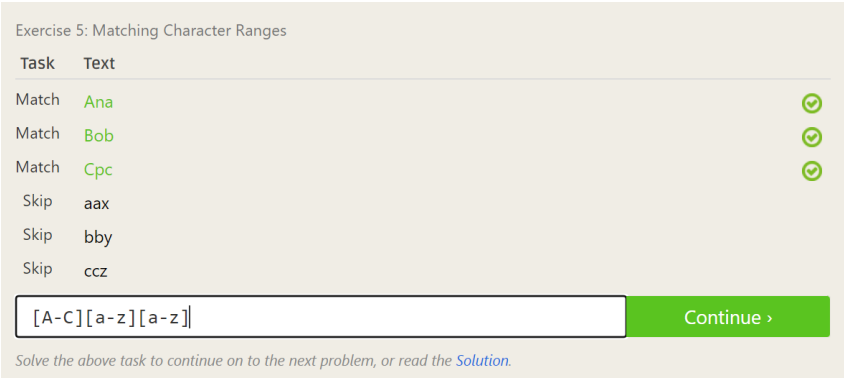

Figure 6: Exercise 6: Word Boundaries

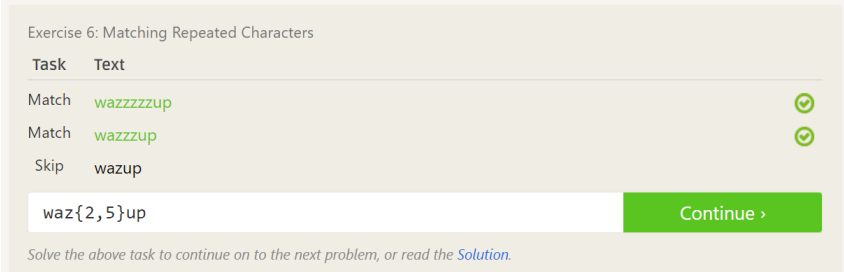

Figure 7: Exercise 6 Part 2: Using Word Boundaries

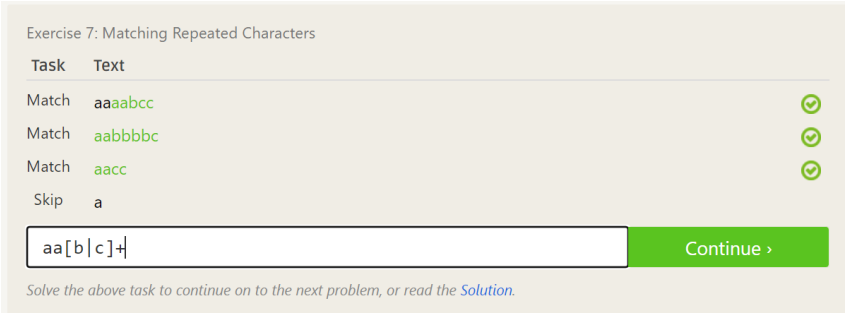

Figure 8: Exercise 7: Matching Digits and Non-Digits

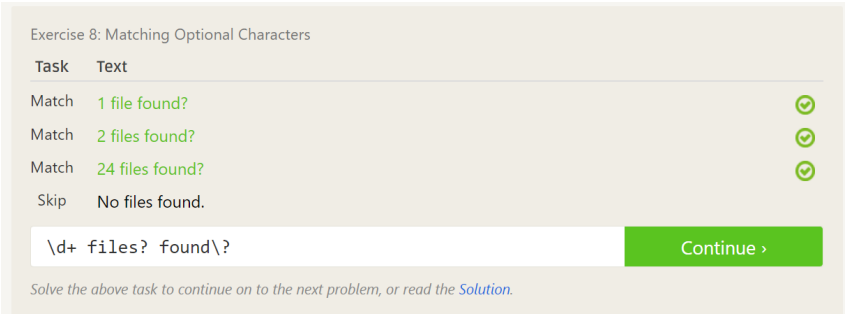

Figure 9: Exercise 8: Matching Whitespace

Figure 10: Exercise 9: Capturing Groups

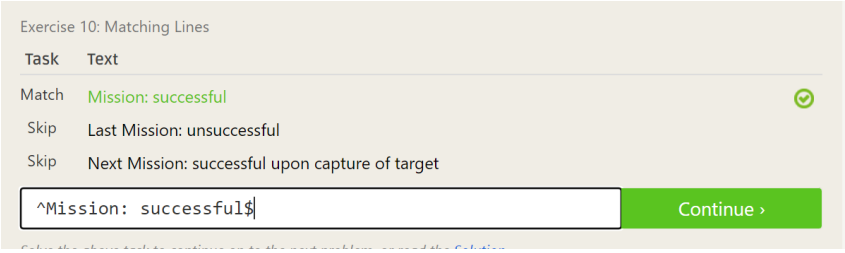

Figure 11: Exercise 10: Repetitions and Quantifiers

Figure 12: Exercise 11: Alternation (OR)

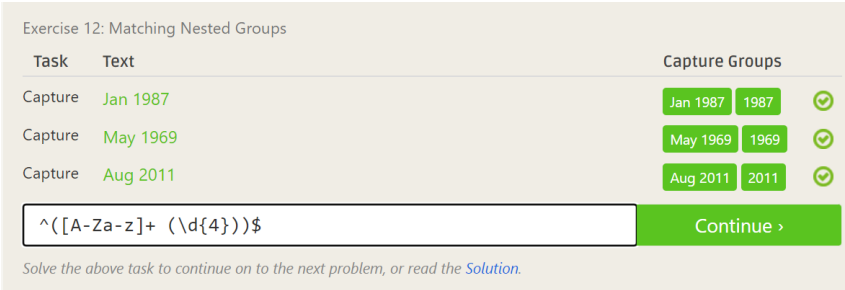

Figure 13: Exercise 12: Matching Patterns with Subgroups

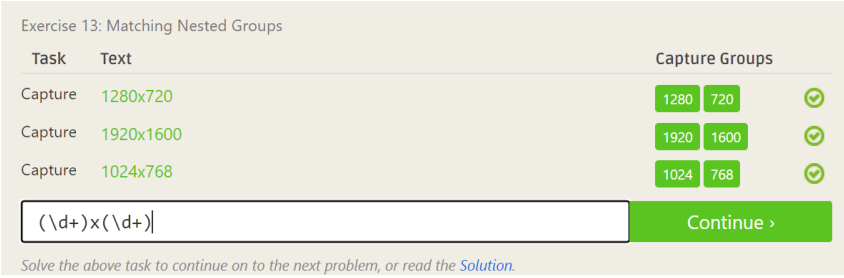

Figure 14: Exercise 13: Non-Capturing Groups

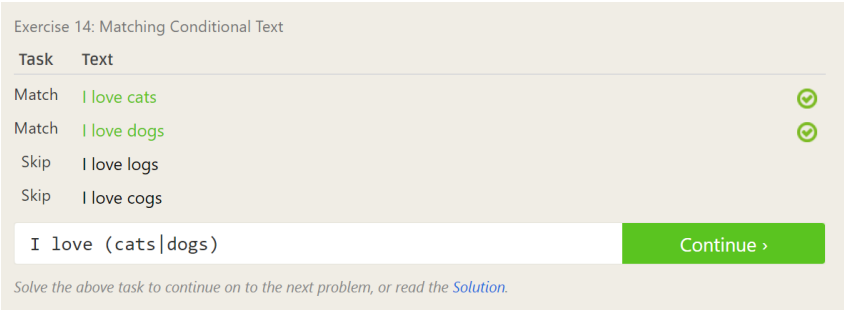

Figure 15: Exercise 14: Greedy vs Lazy Matching

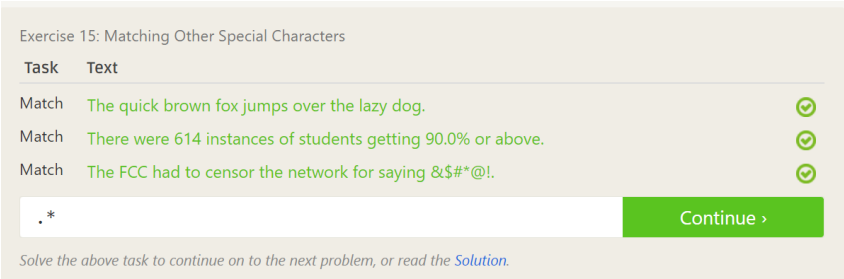

Figure 16: Exercise 15: Final Completion

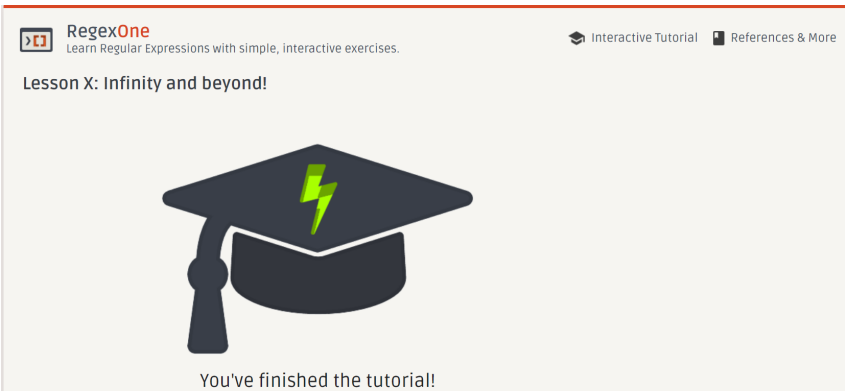

Figure 17: Completion Screenshot: Regex Tutorial Completion

—

0.2 Task 2: Finding Words

0.2.1 Part A: Extracting Assembly Instructions

Objective

The goal of this task was to write a shell script to search for and extract assembly instructions associated with a given PC value from the `core.txt` file.

Bash Script

Listing 1: Task 2: Extracting Assembly Instructions

```
#!/bin/bash

# Check if the user provided a PC value
if [ $# -ne 1 ]; then
  echo "Usage: $0 <PC_value>"
  exit 1
fi

# Store the input PC value
PC_VALUE=$1
```

```
# Search for the instruction in the file
result=$(grep "^${PC_VALUE}," core.txt)

# Check if the result is empty (no matching PC value)
if [ -z "$result" ]; then
  echo "Instruction␣not␣present␣in␣file"
else
  # Extract the full assembly instruction if the PC value is found
  echo "$result" | awk -F',' '{match($0,␣/"([^"]+)"/,␣arr);␣print␣
     arr[1]}'
fi
```

Output

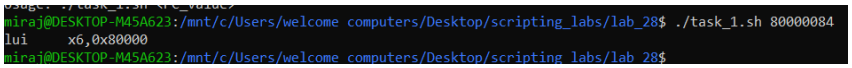

Figure 18: Output for Task 2, Part A - Extracting Assembly Instructions

—

0.2.2 Part B: Processing Words in article.txt

Objective

The goal of this task is to process the `article.txt` file to:

- Find words containing at least one 'a' but not ending in 'i'.
- Identify the three most common two-letter suffixes.
- Count the unique two-letter combinations.
- List the two-letter combinations that do not occur.

Bash Script

Listing 2: Task 2: Processing Words in article.txt

```
#!/bin/bash

# Step 1: Convert the file to lowercase for case insensitivity
cat article.txt | tr '[:upper:]' '[:lower:]' > clean_article.txt

# Step 2: Extract words containing at least one 'a' and not ending
   with 'i'
grep -o '\b\w*a\w*\b' clean_article.txt | grep -v 'i$' >
   filtered_words.txt

# Step 3: Find the three most common last two-letter combinations
echo "Three␣most␣common␣last␣two-letter␣combinations:"
```

```
awk '{print␣substr($0,␣length($0)-1)}' filtered_words.txt | sort |
   uniq -c | sort -nr | head -3

# Step 4: Count how many unique two-letter combinations there are
echo "Number␣of␣unique␣two-letter␣combinations:"
awk '{print␣substr($0,␣length($0)-1)}' filtered_words.txt | sort |
   uniq | wc -l

# Step 5: Find which two-letter combinations do not occur (
   additional challenge)
echo {a..z}{a..z} | tr '␣' '\n' > all_combinations.txt

# Find the combinations that are present in the filtered words
awk '{print␣substr($0,␣length($0)-1)}' filtered_words.txt | sort |
   uniq > present_combinations.txt

# Find which two-letter combinations do not occur
echo "Two-letter␣combinations␣that␣do␣not␣occur:"
grep -v -f present_combinations.txt all_combinations.txt
```

Output

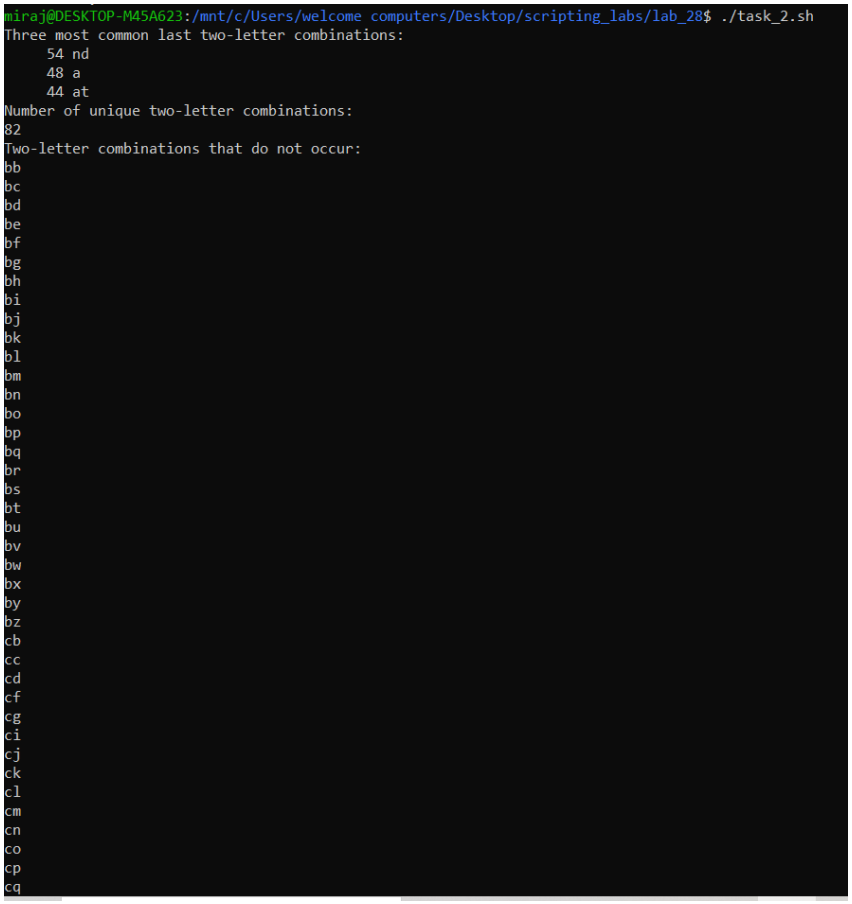

Figure 19: Output for Task 2, Part B - Processing Words in article.txt