

EE282 Homework #2 Assignment

Generating 3 HW questions based on the three prompts

Question #1: Bash

Prompt: Ask a problem that requires a student to understand basic manipulation of files. Your question should require an answer using the following commands/ concepts: ls, touch, mv, rm

You are graduate student working on a project looking at different opsin types in *Speyeria mormonia* butterfly eyes. The lab keeps all project files in a shared project/ directory on a server. Your PI has asked you to organize the files for data collection.

You log into the server and go into the project/ directory. Now it's time to organize the files using Bash.

1. List the contents of the project/ directory to see what files are present.
2. Create three empty files named data1.txt, data2.txt, and notes.txt
3. Rename notes.txt to README.txt
4. Organize the data into a subdirectory by creating a new directory called raw_data/ and move data1.txt and data2.txt into it.
5. You find that data2.txt file is corrupted, delete it from inside the raw_data/ directory
6. Check the organization by listing the contents of both project/ and raw data/ to make sure all files are correctly organized.

KEY:

1. List contents of the project directory

```
ls
```

2. Create three empty files

```
touch data1.txt data2.txt notes.txt
```

```
# 3. Rename notes.txt to README.txt
mv notes.txt README.txt
```

```
# 4. Create raw_data directory and move data files
```

```
mkdir raw_data
mv data1.txt data2.txt raw_data/
```

```
# 5. Delete data2.txt from raw_data
```

```
rm raw_data/data2.txt
```

```
# 6. List contents to verify changes
```

```
ls
ls raw_data
```

Question #2: R

Prompt: Ask a question that requires a student understand the difference between accessing a column in a matrix with numeric indices versus accessing a column in a data frame with numeric indices. Your question should require an answer comparing the following: `mymatrix[,1]` vs `mydf[,1]` vs `mydf[1,]` vs `mydf[[1]]`. **Can change file names from mydf to what I want**

1. You are conducting immunohistochemistry experiments on *Speyeria mormonia* butterfly eyes to look at opsin expression for your lab. Your PI gives you two versions of a dataset that recorded measurements from an experiment:

One is stored as a **matrix** (for the use in numerical computations).
The other is stored as a **data frame** (for use in analysis and reporting).

You load both versions into R:

```
mymatrix <- matrix(c(1, 2, 3, 4), nrow = 2)
colnames(mymatrix) <- c("Opsin Types", "Visual Patterns")
Speyeriadf <- data.frame(
  Opsin Types = c(1, 2),
  Visual Patterns = c(blue, uv)
```

Predict the Outputs for the following expressions looking at the given information

`mymatrix[, 1]`

`Speyeriadf[, 1]`

`Speyeriadf[1]`

`Speyeriadf[[1]]`

1. Write the expected output for each expression.
2. State whether each output is a numeric vector or data frame.
3. Indicate whether the column names and dimensions were preserved for each output.

KEY:

1. `mymatrix[, 1]`

Output:

`[1] 1 2`

Type/Class:

Numeric vector

No preserved column names

No dimensions preserved

Explanation:

Matrices are atomic structures. Extracting a single column drops dimensions by default, returning a vector.

2. `Speyeriadf[, 1]`

Output:

`[1] 1 2`

Type/Class:

Numeric vector

No column names preserved.

No dimensions preserved.

Explanation:

Using [, 1] on a data frame behaves like a matrix-style indexing. R simplifies the result and drops the data frame structure.

3. Speyeriadf[1]

Output:

Opsin Types

```
1      1
2      2
```

Type/Class:

Data frame

Column names are preserved.

Dimensions are preserved.

Explanation:

Single brackets [return a subset of the data frame, not the contents of the column.

4. Speyeriadf[[1]]

Output:

```
[1] 1 2
```

Type/Class:

Numeric vector

Column names are **not** preserved.

Dimensions are **not** preserved.

Explanation:

Double brackets [[means the contents of a single column, which removes all of the data frame structure.

Question #3: Making an R script Executable and Secure

Prompt: Ask a question that requires a student to understand how to create a simple script and make it usable by everyone, but only writable by its creator. Your question should require an answer using chmod NNN (using octal permissions).

Everyone in your lab uses a shared Linux server. You write a short R script for an analysis you are conducting on the *Speyeria mormonia* project that prints a summary message after finishing an analysis. Other members in the your lab need to run your script so it must be executable by everyone.

However, to avoid any potential accidental changes to your script, only you should be the sole editor of the script.

You create a file called `summary.Speyeria` with the following contents

```
#!/usr/bin/env Rscript
```

```
print("Speyeria mormonia summary complete")
```

However when you check the file permissions, you notice that the script is not executable.

1. Write the Linux command to change the permissions of `summary.Speyeria` so that
You the owner can read, write, and execute the script.
Other lab members can read and execute the script, but not modify it.
All other users on the system can read and execute the script but not modify it.
2. Your answer must be numeric (octal) permissions with `chmod` in the form:
`chmod NNN summary.Speyeria`
3. Briefly explain what each digit of your chose octal permission setting represents.

KEY:

1. in Bash:

The correct `chmod` Command is:

```
chmod 755 summary.Speyeria
```

2. The three digits 755 correspond to **owner**, **group**, and **others**, in that order. Each digit is the sum of the permission values.

Explanation of Permissions

- **7 (Owner):** read (4) + write (2) + execute (1)
- **5 (Group):** read (4) + execute (1)
- **5 (Others):** read (4) + execute (1)

3. After running the command the file permissions will look like:

-rwxr-xr-x summary.Speyeria

Owner (rwx): read, write, execute

Group (r-x): read, execute

Others (r-x): read, execute