


Lab 06 Dashboard using Tableau and Miscellaneous

RE 519 Data Analytics and Visualization | Autumn 2025

- Lab 06-A: Dashboard using Tableau
 -  [TODO: Tableau Dashboard for Project Datasets](#)
 - Lab 06-B: Command Line, Virtual Environment, and SQL (Optional)
 - [Command Line](#)
 - [Virtual Environment](#)
 - [Relational Database and SQL](#)
 - [Code Style](#)
 - Acknowledgement
-

The due day for each lab can be found on the [course website](#). The submission should include Rmd, html, and any other files required to rerun the codes. For Tableau, submit your **Tableau packaged workbook (.tbwx)** This will include any excel files that you attach so that when you share the file anyone with Tableau can open it!

From Lab 4, the use of any generative AI tool (ChatGPT, Copilot, etc.) is **allowed**. But, I still encourage you to write codes by yourself and use AI tools as a way to debug and explore new knowledge. More information about [Academic Integrity](#) and [the Use of AI](#).

Lab 06-A: Dashboard using Tableau

 **TODO: Tableau Dashboard for Project Datasets**

16 points

The purpose of this lab is to use Excel/R for data preprocessing and Tableau Worksheet + Dashboard to conduct exploratory data analysis and initial visualization on the topic you are interested in. The assignment can be seen as the first step for the [Data Analytics and Visualization Projects](#). Note: if you plan to **use Tableau Dashboard for your final project**, you are expected to make significant progress after this lab, both in analytical depth and dashboard design.

Requirements

Basic requirements for this lab:

- Create at least five individual visualizations worksheets
- Create one or two interactive dashboards
 - Add a title, description, and data source citation
 - [Format your work](#) and ensure proper alignment, consistent colors, and readable labels,
 - Make the dashboard interactive using, for example, [parameters](#), [filter](#), [highlights](#), [URL actions](#), [set actions](#).

You are encourage to explore some advanced functions in Tableau. [Tableau Desktop and Web Authoring Help](#) is always the most helpful material to learn Tableau.

- [Create Parameters](#)
- [Create Custom Fields with Calculations](#)
- [Create Sets](#)
- [Dashboards](#)
- [Build Advanced Chart Types](#)

The results of Tableau visualization should be reflected and briefly discussed in your [project proposal](#), providing evidence of your initial data exploration and research direction.

Submission

This is a **group assignment** and each group only need to submit once.

Submit your [Tableau packaged workbook \(.tbwx\)](#) This will include any data files that you

attach so that when you share the file anyone with Tableau can open it!

Or, submit the link to your Tableau Public page via Canvas.

You can also upload to GitHub and submit the link to your repository. Please be sure to make the repository public/visible. For Lab 4-9, the use of any generative AI tool (ChatGPT, Copilot, etc.) is allowed.

Late Days

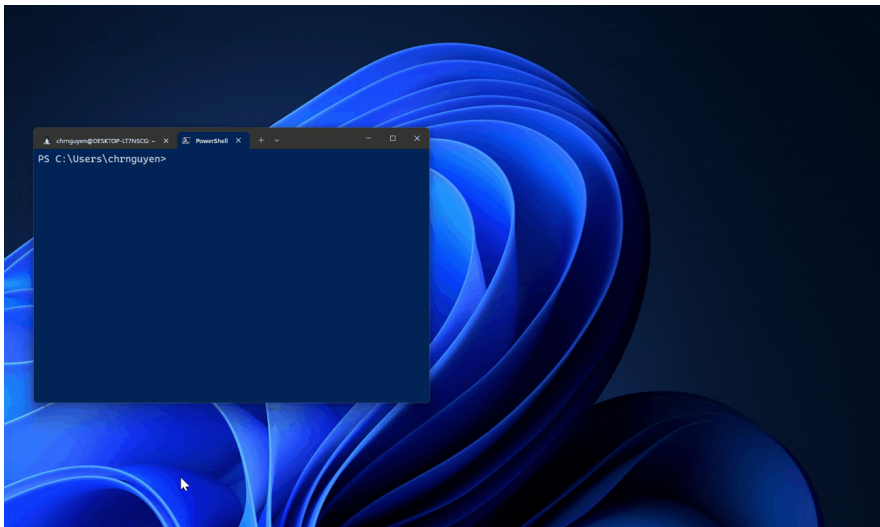
All group members will consume late days if the submission is late.

Lab 06-B: Command Line, Virtual Environment, and SQL (Optional)

In Section B, we will introduce three topics related to data science: command line, environment and SQL. This section is optional, and we will not go deeper into it. The purpose of this section is to let you know what is command line, terminal, database, SQL, and why we use such tools.

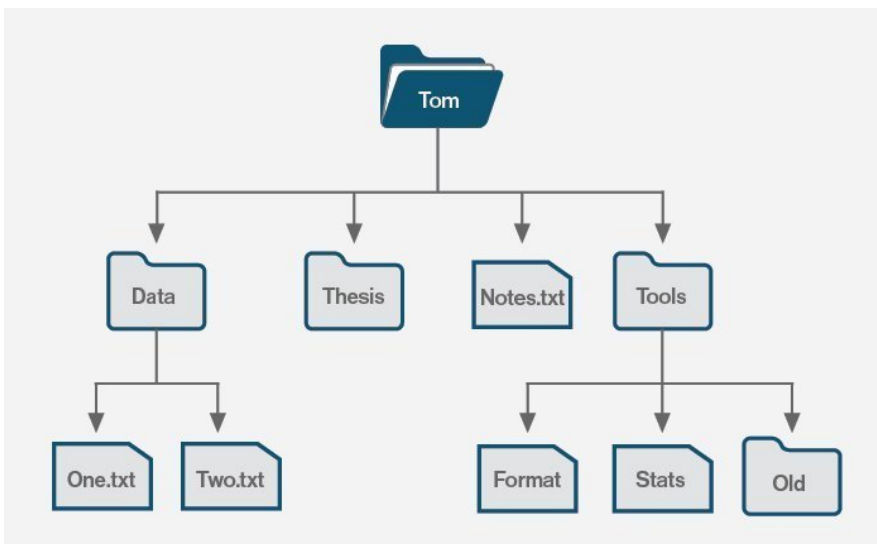
Command Line

We often see those fancy, “geeky” black boxes on computer screens in movies. Those black boxes are usually **terminals**, **shell**, or **command-line interfaces**. They allow users to type **commands** directly to control the computer.



Terminal in Windows System. Source: Microsoft

In any computer, we have a hierarchical structure of folders (also called *directories*) and files within them. The whole file system can be visualized as a tree. Normally, we use a **Graphical User Interface (GUI)** — for example, clicking through Finder on macOS or File Explorer on Windows — to navigate among files, applications, and folders. However, we can also use a **Command Line Interface (CLI)** to navigate the same file system through text commands. For example, when we enter `ls` (means list) in Mac/Linux system terminal, we will get a list of files in the working directory.



File System. Source: www.techtarget.com

Why Do We Sometimes Need Commands?

- We want to have a record of what we did
- We want to repeat the same procedure later — a script can store and rerun our actions automatically
- We want to use some services do not have a graphical interface
- We want to operate systems without a graphical interface, such as a sensor and server

Command Line Tool

Most common command line is `bash`, which is widely used in any Mac, Linux, supercomputer. (Actually, in Mac, the default shell is `zsh` but you can see it as `bash+`). The default command line in Windows is `PowerShell` - their syntax are different. `Bash` is more popular and widely used and we will use it as the example. I believe that is one of the reason that programmers like to use Mac and Linux instead of Windows.

For Windows users, you can learn [PowerShell](#) from Microsoft. Or, you can install a [Linux system using WSL](#). Or, you will likely install `Git` when installing `GitHub Desktop`; `Git` will come up with `bash`.

```
# instead of R, we have bash within the {} above
# we cannot promise the following commands can be run on Windows
```

Most time, we should will terminal instead of `Rmd`. Please try those commands in terminal even through they are same actions. We will only have a look of some simple operations.

```
pwd # print current working directory
ls # list the files/folders in the working directory
mkdir -p new_folder_bash # make a new directory; -p is an argument: no error if existing, make parent c

## /Users/yohaoyu/Repo/data-analytics-visualization/labs/lab-06
## airbnb.sqlite
## bash_example
## create_new_folder_script.sh
## custom.css
## lab-06.html
## lab-06.Rmd
## new_folder_bash
## new_folder_script
```

We open the folder we just created and create a new `txt` file.

```
cd new_folder_bash # change the working directory
ls # list the files/folders in the working directory
touch using_bash.txt # create empty files
rm using_bash.txt # remove the file
```

Script

We can write these commands into a shell script file (with the `.sh` extension), for example `create_new_folder_script.sh`. Then we can run it from the command line using:

```
bash create_new_folder_script.sh
```

Next Step

I don't think we have to go any deeper here. The goal of this section is simply to help you understand what command-line tools are and how they can be used to automate tasks. If you feel it is interesting, there are some comprehensive guide on `bash`:

- [The Linux Command Line](#).
- [The Unix Shell Software Carpentry](#).

An Example on Data Processing using `bash`

When you download data from [Dewey Data Platform](#), like the rental data we used in lab 4. You may receive many compressed `csv` files. In this example, we have a folder called `bash_example`, which contains 100 zipped `csv` files. We would like to unzip and merge them for future analysis or visualization. It will take amount of time and energy to do that one by one. `bash` would be super helpful.

Even if we don't know exactly how to do this using the command line, we can simply ask AI tools like OpenAI's GPT or Google's Gemini: "How do I unzip all zip files and merge all CSV files (with the same header) into a single file using `bash`?"

ChatGPT 5 gives me the following commands and it works:

```
for f in *.zip; do
    unzip -o "$f"
done
```

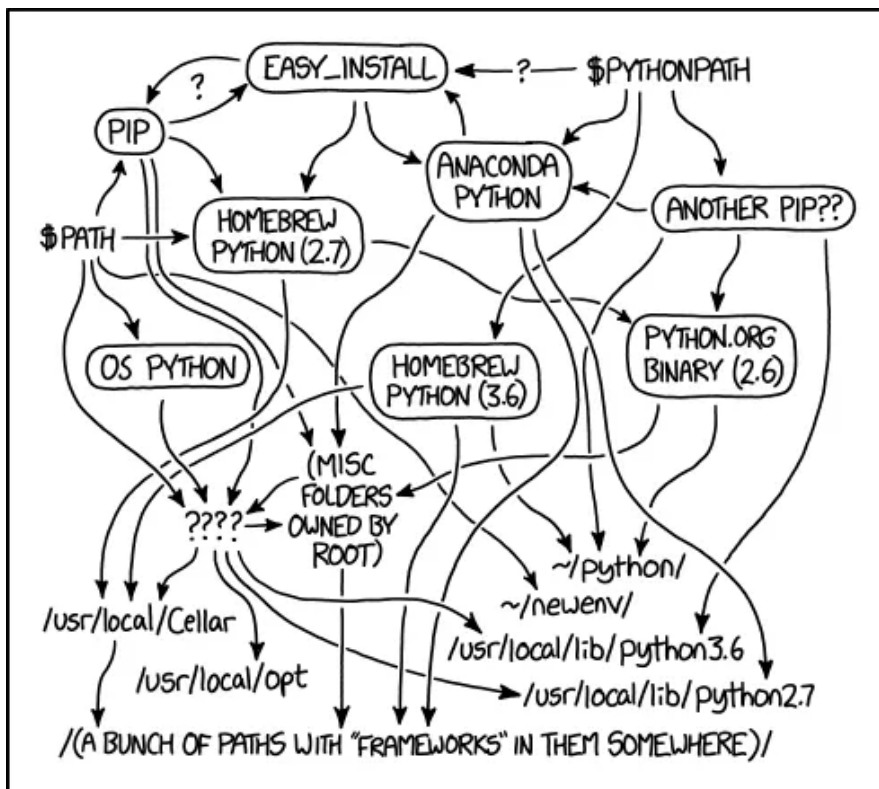
```

header=$(head -n 1 $(ls *.csv | head -1))
echo "$header" > merged.csv
for f in *.csv; do
    tail -n +2 "$f" >> merged.csv
done

```

Virtual Environment

You may realize that we are using many, many, many existing packages. Meanwhile, those packages are changing and those packages are using other packages..... In other words, R (like most programming ecosystems) is built on a web of software dependencies, where one package may only work correctly with specific versions of the others — some people call it **dependency hell**.



MY PYTHON ENVIRONMENT HAS BECOME SO DEGRADED
THAT MY LAPTOP HAS BEEN DECLARED A SUPERFUND SITE.

Dependency Hell. Source: Montana Low

This is why **virtual environments** are so important. They help us *record exactly which package versions were used for a project*, so that our code remains reproducible and stable, even if the global R ecosystem keeps evolving.

In R, we use `renv` [package](#) to manage that for each project. Also, we can use general-purpose environment managers, such as [Anaconda](#), which can handle dependencies across multiple languages (e.g., Python, R).

Relational Database and SQL

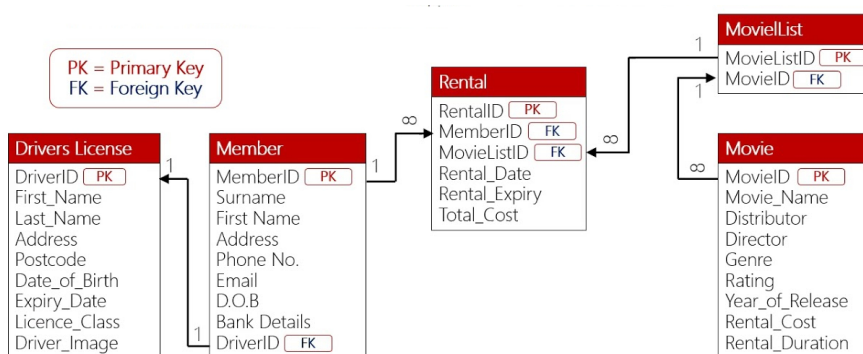
For handling small datasets, like all the datasets we have used so far, storing them in simple `.csv` files works perfectly fine. However, when we start dealing with larger datasets, or data that need to be shared, updated, or queried efficiently, it becomes difficult to manage everything as separate files — imagine the disaster of running DXD Capital’s data system with thousands of `CSVs`.

So, people are using **Database Management System (DBMS)** to create and manage a large amount of data. For commonly used row/column data, **relational databases** are the most widely used type of database. It organizes data into tables, where each table contains rows and columns. Different tables can be linked together through **keys**.

- Primary Key [PK]: unique attributes.
- Foreign Key [FK]: attributes that can be used to uniquely identify a row in **another table**.

Relationships are connections among two or more entity sets.

- one-to-one: SSN - a person who has SSN
- many-to-one: SSN - phone numbers
- many-to-many: stores - products



Entity/Relationship Diagrams for a Relational Database. Source: Josh Erickson, Statistics, University of Michigan

A Quick Look of SQL

SQL (Structured Query Language) is the most commonly used query language for relational databases. Let's try to use SQL to get query for Airbnb data within `airbnb.sqlite`, a [SQLite](#) database.

```
# install.packages("DBI") # you only need to install once
# install.packages("RSQLite") # you only need to install once
library(DBI)
# connect to the airbnb database
airbnb_db <- dbConnect(RSQLite::SQLite(), "airbnb.sqlite")

# check the tables and fields (not important points for SQL)
dbListTables(airbnb_db)

## [1] "airbnb"

dbListFields(airbnb_db, "airbnb")

## [1] "id" "name"
## [3] "host_id" "host_name"
## [5] "neighbourhood_group" "neighbourhood"
## [7] "room_type" "price"
## [9] "minimum_nights" "number_of_reviews"
## [11] "last_review" "reviews_per_month"
## [13] "calculated_host_listings_count" "availability_365"
## [15] "number_of_reviews_ltm" "license"
## [17] "geometry"
```

We use queries to get the data we want from a database. A query is simply a command written in SQL that asks the database to return specific information.

SQL Command	Purpose	Example
SELECT	Retrieve data from one or more tables	SELECT * FROM my_table;
WHERE	Filter rows based on a condition	SELECT * FROM my_table WHERE population > 100000;
ORDER BY	Sort results	SELECT * FROM my_table ORDER BY income DESC;
LIMIT	Restrict the number of returned rows	SELECT * FROM my_table LIMIT 10;
GROUP BY	Aggregate rows by a category	SELECT state, AVG(income) FROM my_table GROUP BY state;

JOIN

Combine data from multiple tables

SELECT * FROM city JOIN state ON city.state_id = state.id;

```
# query within "*"
# select all columns from airbnb data limit to 3 rows
dbGetQuery(airbnb_db, "SELECT *
```

```
FROM airbnb
LIMIT 3")
```

```
##      id                                name host_id host_name
## 1 6606          Fab, private seattle urban cottage!  14942      Joyce
## 2 9419          Glorious sun room w/ memory foamed  30559 Angielena
## 3 9596 the down home , spacious, central and fab!  14942      Joyce
##  neighbourhood_group neighbourhood      room_type price minimum_nights
## 1 Other neighborhoods      Wallingford Entire home/apt  123              30
## 2 Other neighborhoods      Georgetown Private room    88              2
## 3 Other neighborhoods      Wallingford Entire home/apt   NA              30
##  number_of_reviews last_review reviews_per_month
## 1              161      19973              0.83
## 2              212      20246              1.17
## 3              96      18533              0.56
##  calculated_host_listings_count availability_365 number_of_reviews_ltm
## 1              3              117              1
## 2              10             311             15
## 3              3              1              0
##      license                                geometry
## 1 str-opli-19-002622 01010000206e0b000037e09ba50c61334104259073f3940d41
## 2      Exempt 01010000206e0b0000fc85f387776e3341401016da13ee0841
## 3 STR -OPLI-19-002622 01010000206e0b0000e9c1a7db5a613341a77482de99a70d41
```

```
# select distinct neighbourhood_group column from airbnb data limit to 3 rows
dbGetQuery(airbnb_db, "SELECT DISTINCT neighbourhood_group
```

```
FROM airbnb
LIMIT 3")
```

```
##  neighbourhood_group
## 1 Other neighborhoods
## 2      Ballard
## 3      Magnolia
```

```
# select all columns from airbnb data and order by price limit to 3 rows
dbGetQuery(airbnb_db, "SELECT *
```

```
FROM airbnb
ORDER BY price DESC
LIMIT 3")
```

```
##      id                                name host_id
## 1 1.107914e+18  Seattle Uncovered: Art Scene & Memorable Sights 25138314
```

```
## 2 1.107914e+18 Eclectic Haven: Next to Iconic Seattle Landmarks 25138314
## 3 1.108000e+18 Emerald City - Boutique Gem with Pillow Bar 5615582
##          host_name neighbourhood_group          neighbourhood
## 1 RoomPicks By Antony          Downtown          Belltown
## 2 RoomPicks By Antony          Downtown          Belltown
## 3          Aishat          Downtown Central Business District
##          room_type price minimum_nights number_of_reviews last_review
## 1      Hotel room 50034              1              1      20030
## 2      Hotel room 50034              1              2      20141
## 3 Entire home/apt 50032              1              2      20176
##  reviews_per_month calculated_host_listings_count availability_365
## 1              0.13              13              341
## 2              0.20              13              343
## 3              0.48              12              336
##  number_of_reviews_ltm license
## 1              1 Exempt
## 2              2 Exempt
## 3              2 Exempt
##
##          geometry
## 1 01010000206e0b0000f8f2c8d07f5f3341b9ca6ad7aacb0b41
## 2 01010000206e0b0000f8f2c8d07f5f3341b9ca6ad7aacb0b41
## 3 01010000206e0b0000f1736d5dcb613341a85a06e99eb40b41
```

```
# the average price by neighborhood group and avg price above 250
```

```
dbGetQuery(airbnb_db, "SELECT neighbourhood_group, AVG(price) AS Avg_price
                        FROM airbnb
                        GROUP BY neighbourhood_group
                        HAVING Avg_price > 250")
```

```
##  neighbourhood_group Avg_price
## 1      Cascade      293.697
## 2      Downtown  2190.712
## 3      Queen Anne   294.666
```

```
# disconnect from the database
```

```
dbDisconnect(airbnb_db)
```

SQL is not a complicated language. But similarly to shell, I will recommend to use AI to prepare the queries unless you have a full-time job in data science. Some resources for learning SQL:

- [SQLBolt](#)
- [CSE 414 Introduction to Database Systems \(for non-CSE Majors\)](#)
- [Databases and SQL Software Carpentry.](#)

Code Style

It is important to write efficiently that can be read by computers and **people**. In the community of R, we use to follow the [Tidyverse Style Guide](#).

Acknowledgement

The materials are developed by [Haoyu Yue](#) based materials from [Dr. Feiyang Sun](#) at UC San Diego, Siman Ning and Christian Phillips at University of Washington, [Dr. Charles Lanfear](#) at University of Cambridge.