Laboratory 1
Overview of Transport Data Sources and Tools

Module: Introduction to Transport Research and Analysis

Duration: 180 Minutes

Contents

Objective 2

1 Files & Tools 3

2 Activity 1 – Exploring Transport-Data Categories & Sources 3

3 Activity 2 – Tool Access & "Hello World" 5

4 Activity 3 – MATLAB First Steps with a Transport Dataset 6

5 Activity 4 – Python Basics with `Python_Introduction.ipynb` 7

6 Activity 5 – Excel Basics for Transport Data 9

7 Activity 6 – VISUM and VISSIM Orientation 9

8 Homework – Comparative Analysis of Transport Data in MATLAB and Python 10

Objective

In this lab you will:

- classify major types of transport data and link them to common file formats and portals,
- explore at least one real transport dataset (counts, GTFS, or similar),
- verify access to three core analysis tools: MATLAB, Python/Colab, and Excel,
- perform basic data loading and plotting in MATLAB and Python,
- practice simple spreadsheet operations in Excel on transport data,
- understand the role of VISUM and VISSIM through a GUI walk-through or demonstration,
- prepare for a homework assignment comparing MATLAB vs. Python for data analysis.

The focus is on following the practical steps clearly and getting comfortable with tools and datasets used throughout the rest of the course.

1 Files & Tools

1.1 Files Used in This Lab

The exact filenames may differ between universities, but you will typically have:

- **Example transport datasets**, e.g.
  - a sample traffic-count or speed dataset (`traffic_counts_example.csv`),
  - a public-transport schedule extract or passenger-count file,
  - any small dataset your lecturer provides on Moodle or by e-mail.
- **Python notebook:**
  - `Python_Introduction.ipynb` (basic Python syntax and examples).

1.2 Software Platforms

- **MATLAB** (desktop or MATLAB Online).
- **Python** via Anaconda (local) or **Google Colab** (browser).
- **Excel** (desktop) or Excel Online / Google Sheets.
- **PTV VISUM** and **PTV VISSIM** (usually on lab PCs; some students may install a student version).

Reflection & Insight

This lab is intentionally light on theory: the aim is to touch each tool at least once, so that later labs can focus on modelling rather than installation or "where do I click" issues.

2 Activity 1 – Exploring Transport-Data Categories & Sources

In the introductory slides you saw eight major categories of transport data (network, service, demand, traffic, freight, shared mobility, environment, socio-economic context) and example portals such as OpenStreetMap, GTFS directories, Eurostat, and national traffic-count sites.

Task 1: Map Example Variables to Data Categories

Goal: Connect real-world variables to the correct transport-data category.
Steps:

1. In your notes (or on a printed handout), set up a table with three columns:

Category	Typical variables	Example formats
Network & infrastructure		
Service / supply		
Demand & mobility patterns		
Traffic & operations		
Freight & logistics		
Shared & micromobility		
Environment & asset condition		
Socio-economic / land-use		

2. Using examples from the slides, fill in each row with:
   - 2–3 typical variables (e.g. "hourly volume, AADT" for traffic & operations),
   - 1–2 common formats or specs (e.g. Shapefile, GeoJSON, GTFS, CSV).
3. Compare your answers with your neighbours and resolve discrepancies.

Checkpoint: You should be able to take any given variable (e.g. "bus headway" or "weigh-in-motion tonnage") and put it into the correct category.

Task 2: Explore At Least One Open Data Portal

Goal: Discover where to obtain real transport datasets.
Steps:

1. With your lecturer, briefly visit one or two of the portals mentioned in the slides, such as:
   - OpenStreetMap / Geofabrik downloads,
   - GTFS data directory,
   - a national traffic-count or household-travel survey portal.
2. Note for each portal:
   - what type of data it provides,
   - which formats are available (e.g. zipped GTFS text files, CSV, Shapefile),
   - temporal and spatial coverage (which years, which regions).
3. Write down one dataset that looks interesting for your future project (name, link and a one-line description).

Optional: If time permits, download a small CSV file (e.g. hourly volume at one station) that you can use later in MATLAB and Python.

3 Activity 2 – Tool Access & "Hello World"

Before we start doing transport analysis, we must confirm that MATLAB, Python and Excel work on your device (or on a lab PC).

Task 3: MATLAB / Octave Access

Goal: Confirm you can run a simple MATLAB script.
Steps:
1. Launch MATLAB. If you do not have the desktop version, use:
   - **MATLAB Online** in your browser, or
   - **GNU Octave** as a free alternative with similar syntax.
2. In the Command Window, type:

```
x = linspace(0, 4*pi, 100);
y = sin(x);
plot(x, y);
title('Hello Transport World');
xlabel('x');
ylabel('sin(x)');
```

3. Check that a plot window appears and that you see a sinus curve.

Checkpoint: If you can see the plot without errors, your MATLAB installation is ready for later labs.

Task 4: Python / Colab Access

Goal: Ensure you can run Python code from the `Python_Introduction.ipynb` notebook.
Steps:
1. Open https://colab.research.google.com and sign in.
2. Upload `Python_Introduction.ipynb` to Colab (File → Upload Notebook).
3. Run the first few cells. If the notebook contains `print("Hello, world!")`, ensure it executes successfully.
4. In a new code cell, type:

```
x = 5
y = 3
print("Sum =", x + y)
```

5. Run the cell and confirm that `Sum = 8` is printed.

Task 5: Excel / Sheets Access

Goal: Confirm you can open and edit a spreadsheet.
Steps:
1. Open Excel (desktop) or Excel Online / Google Sheets.
2. Create a small table with three columns:
   - `time_hour`, `volume`, `lane_id`,
   - and 5–10 dummy rows of made-up values.
3. Insert a simple line chart of `volume` vs. `time_hour`.

Checkpoint: You can quickly create, save, reopen and edit a basic spreadsheet file.

4 Activity 3 – MATLAB First Steps with a Transport Dataset

In this activity you will load a small transport dataset and make a basic plot in MATLAB.

Task 6: Load and Inspect a Dataset in MATLAB

Goal: Read a CSV file, inspect its structure, and identify relevant variables.
Steps:
1. Place the example dataset (e.g. `traffic_counts_example.csv`) in your MATLAB working folder.
2. In MATLAB, run:

```
T = readtable('traffic_counts_example.csv');
head(T)
summary(T)
```

3. From the output, identify:
   - the number of rows (observations),
   - the main columns (e.g. time, volume, station or link ID).

Checkpoint: You should now know what each column represents and which one is suitable for plotting as a time series or category-wise bar chart.

Task 7: Create at Least One Plot in MATLAB

Goal: Plot a meaningful relationship from the dataset.

Steps:

1. Choose a numeric variable to visualise (e.g. hourly volume).
2. If there is a time column (e.g. `timestamp` or `hour`), create a line plot:

```
plot(T.timestamp, T.volume);
xlabel('Time');
ylabel('Volume␣(veh/h)');
title('Hourly␣traffic␣volume');
```

3. If your dataset is more categorical (e.g. station IDs), you could aggregate and plot:

```
G = groupsummary(T, "station_id", "sum", "volume");
bar(G.station_id, G.sum_volume);
xlabel('Station');
ylabel('Total␣volume');
title('Total␣volume␣by␣station');
```

4. Adjust axis labels and title to match your dataset context.

Reflection & Insight

MATLAB is particularly strong at matrix operations and quick plotting. Do not worry if your plot is not "perfect" yet—focus on correctly loading data and producing a readable figure.

5 Activity 4 – Python Basics with `Python_Introduction.ipynb`

The slides introduced Python concepts such as variables, data types, input/output, conditionals, and loops. The notebook `Python_Introduction.ipynb` mirrors these ideas in executable code.

Task 8: Run and Modify Basic Python Examples

Goal: Get comfortable running and editing Python cells in Colab.
Steps:
1. In Colab, run all cells in `Python_Introduction.ipynb` once from top to bottom.
2. Locate a cell defining variables such as:

```
my_int = 7
my_float = 7.0
my_list = [1, 2, 3]
```

3. Change the values (e.g. modify the list) and re-run the cell. Confirm the printed output updates accordingly.
4. Find a cell that defines a function, for example:

```
def add(a, b):
    return a + b
```

5. Add a new cell where you call this function several times with different arguments and print the results.

Checkpoint: You should feel comfortable changing simple numbers, strings, lists and re-running the notebook without errors.

Task 9: Input Conditionals and Loops

Goal: Practice core Python control structures that you will reuse in later labs.
Steps:
1. Find the part of the notebook where `input()` and `print()` are used.
2. Modify the code so that it:
   - asks the user for a number,
   - prints whether the number is positive, negative or zero.
3. Next, create a simple loop that prints numbers from 1 to 10 and their squares:

```
for n in range(1, 11):
    print(n, n**2)
```

4. Run your cells and confirm the outputs are as expected.

Reflection & Insight

These basics (variables, functions, conditionals, loops) are the building blocks for everything from simple data cleaning to advanced machine learning pipelines in later courses.

Task 10: (Optional) Load the Same Dataset in Python

Goal: Mirror your MATLAB analysis using Python and pandas.
Steps:
1. Upload the same CSV file you used in MATLAB to Colab.
2. In a new notebook cell, run:

```
import pandas as pd

df = pd.read_csv('traffic_counts_example.csv')
print(df.head())
print(df.describe())
```

3. Create a basic plot, e.g. total volume by station:

```
grouped = df.groupby('station_id')['volume'].sum()
grouped.plot(kind='bar');
```

4. Compare this plot to your MATLAB version.

6 Activity 5 – Excel Basics for Transport Data

Task 11: Open and Explore a Dataset in Excel

Goal: Perform simple cleaning and visualisation using a spreadsheet.
Steps:
1. Open your example dataset (CSV or XLSX) in Excel.
2. Check that:
   - the header row is correctly interpreted as column names,
   - numeric columns (e.g. volumes) are not stored as text.
3. Turn on Filters (Data → Filter) and:
   - filter to one station or route,
   - sort by time or date.
4. Insert a simple chart (Insert → Line or Column) to show volume vs. time or counts by station.

Optional: If you know PivotTables, build a PivotTable summarising total volume by station and day.

7 Activity 6 – VISUM and VISSIM Orientation

VISUM and VISSIM are specialised tools that you will see more in later modules. In this lab the goal is only to understand *what* they are used for and how their interfaces look.

Task 12: Observe the VISUM GUI

Goal: Recognise the main parts of the VISUM interface.
Steps (often lecturer-led):
1. On a lab PC (or via a projector), open a small VISUM project.
2. Identify:
   - the network editor window showing links and nodes,
   - the object list (links, nodes, zones, lines),
   - where OD matrices and assignment results are displayed.
3. Note one example question VISUM is good at answering (e.g. link volumes, V/C ratios in a peak hour).

Task 13: Observe the VISSIM GUI

Goal: Distinguish VISUM (macroscopic) from VISSIM (microscopic).
Steps (often lecturer-led):
1. Open a simple VISSIM corridor model.
2. Watch a short simulation run and note:
   - individual vehicles moving, stopping, and queuing,
   - signalised intersections and lane changes.
3. Discuss in one sentence:
   - When would you choose VISUM instead of VISSIM?
   - When is a microscopic view (VISSIM) necessary?

Reflection & Insight

Rough rule of thumb: VISUM for big-picture network planning (flows on many links); VISSIM for detailed operations at corridors and intersections.

8 Homework – Comparative Analysis of Transport Data in MATLAB and Python

For homework, you will repeat part of the in-lab work more systematically and write a short report.

Task 14: Dataset Selection and Description

Goal: Obtain a real transport dataset from an open portal.
Steps:
1. Choose one data portal (e.g. GTFS feeds, Eurostat GISCO, FHWA traffic counts, national open-data site).
2. Download a dataset that includes:
   - either traffic volumes or speeds, *or*
   - public-transport schedules or passenger counts, *or*
   - origin-destination / travel-demand information.
3. Save the dataset in CSV format if possible.
4. Write a short paragraph describing:
   - what the dataset contains (variables, spatial/temporal coverage),
   - where you obtained it (portal and URL).

Task 15: Data Visualisation in MATLAB

Goal: Produce at least two useful plots in MATLAB.
Steps:
1. Load the dataset using `readtable()` or a similar function.
2. Perform any minimal cleaning (e.g. remove missing values, filter to one route or station).
3. Create at least two plots, for example:
   - a time-series line plot (e.g. volume or headway over time),
   - a bar chart comparing categories (e.g. stations, routes, vehicle types).
4. Export the plots as images (PNG or JPEG) for use in your report.

Task 16: Data Visualisation in Python

Goal: Repeat a similar analysis using Python/pandas.
Steps:
1. In a new Colab notebook, import the dataset using `pandas.read_csv()`.
2. Use `df.head()`, `df.info()`, and `df.describe()` to understand the structure.
3. Create at least two plots that mirror (as closely as possible) your MATLAB plots (using `matplotlib` or `pandas` plotting).
4. Save the notebook and export the plots as images.

Task 17: Compare MATLAB and Python

Goal: Reflect on tool differences and report your findings.
Steps:

1. Write a short comparative discussion (approx. 1–2 pages) addressing:
   - Which tool did you find easier to use for this dataset?
   - Which offered more flexibility for plotting and cleaning?
   - Were there any differences in how results were visualised or interpreted?
2. Follow the suggested report structure:

1	Title page (lab title, course, your name, ID, date)
2	Objectives (what you set out to do)
3	Introduction (brief background on the dataset and tools)
4	Methodology (dataset acquisition, MATLAB steps, Python steps)
5	Results (plots and key numerical findings)
6	Discussion (comparison of MATLAB vs. Python, challenges)
7	Conclusion (main lessons learned)
8	References (datasets, software, tutorials)
9	Appendices (code listings, extra plots, screenshots)

3. Submit the PDF report and your Python notebook (and MATLAB script, if requested by the lecturer).

Reflection & Insight

Do not try to make the "perfect" plot. Focus on telling a clear story: what your dataset shows and how MATLAB and Python each helped you uncover that story.