Laboratory 10

Cost-Benefit Analysis (CBA) for Transport Projects

Excel & Python Practical Session

Module: Sustainable Transport and Policy Analysis

Duration: 180–200 Minutes

Contents

Objective

In this lab you will:

- Explore a professional **USDOT CBA Excel template** and identify inputs, outputs, and key indicators (NPV, BCR).
- Implement a **basic Python CBA model** using route, trip and service data.
- Extend the Python model to include **capital costs, operating costs, and additional benefits** (emissions, safety, congestion, economic development).
- Test **project viability** under different assumptions using sensitivity analysis.
- Redesign a project scenario (by changing benefits, costs or subsidies) to make it economically viable and interpret NPV/BCR results.

This manual focuses on clear, step-by-step instructions that match the Lab 10 slides and the practical notebooks.:contentReference[oaicite:0]index=0

1 Files & Scenario Overview

1.1 Files Used in This Lab

- **Excel template**
  - `USDOT BCA Spreadsheet Template 11.18.24.xlsx`
- **Python notebooks**
  - `Transport_CBA_Practical_with_less_componenets.ipynb` (basic model)
  - `CBA_Practical_Notebook_with_extra_components.ipynb` (extended model)
- **CSV data**
  - `Routes_Table.csv`
  - `Trip_Table.csv`
  - `Service_Table.csv`

1.2 Transport CBA Scenario (Python Part)

The Python notebooks implement a CBA for a public transport project using real-like data:

- Each row in the merged dataset represents **one trip** or service.
- Key variables per trip include:
  - distance travelled,
  - scheduled and actual travel time (delay),
  - cost per km,
  - per-trip benefit (e.g. time savings).
- From these, the notebooks calculate:
  - transport cost,
  - delay cost,
  - total cost,
  - benefits,
  - Net Present Value (NPV) and Benefit-Cost Ratio (BCR) over multiple years.

In the extended notebook, additional *project-level* costs and benefits are added (capital, operating, emissions, safety, etc.), reflecting the "included vs. missing" components discussed in the handout.

Reflection & Insight

Think of the Excel file as the "professional black box", and the Python notebooks as your way to open that box and re-create the logic step by step.

2 Activity 1 – Exploring the USDOT Excel Tool

Task 1: Open and Explore the Template

Goal: Understand the structure of a professional CBA spreadsheet.
Steps:

1. Open `USDOT BCA Spreadsheet Template 11.18.24.xlsx` in Excel (or Google Sheets if necessary).
2. Identify and click through the main sheets. Typically there will be:
   - input sheets (project description, costs, benefits),
   - calculation sheets (discounting, intermediate results),
   - output/summary sheets (NPV, BCR, IRR, decision).
3. On the *cost inputs* sheet, look for:
   - capital costs (construction, land acquisition),
   - annual operating & maintenance costs (staff, energy, repairs).
4. On the *benefit* sheet, locate:
   - travel time savings,
   - vehicle operating cost savings,
   - safety, emissions, congestion or economic development benefits (if included).
5. On the *summary/output* sheet, note:
   - Net Present Value (NPV),
   - Benefit-Cost Ratio (BCR),
   - any traffic-light or recommendation field (e.g. "Economically Viable" if BCR > 1).

Record:

- Which sheet(s) show capital costs?
- Which sheet(s) show time savings?
- Where exactly are NPV and BCR displayed (sheet name, cell reference)?

Task 2: Link Excel Components to the Python Model

Goal: Connect what you see in Excel with what you will implement in Python.
Steps:

1. Make a two-column list in your notes:
   - Column 1: components visible in Excel (e.g. "Capital costs", "Time savings", "Emission benefits").
   - Column 2: note whether these are:
     - already included in the *basic* Python notebook,
     - only included in the *extended* Python notebook,
     - or not included at all (yet).
2. Use the "included vs missing" CBA handout as a guide: transport costs and delay costs are included; capital, operating, environmental and safety items initially are not.:contentReference[oaicite:2]index=2

Outcome: You should now have a clear idea of which CBA components you will model in Python and which ones are missing in the basic version.

3 Activity 2 – Basic Python CBA Model

This part uses `Transport_CBA_Practical_with_less_componenets.ipynb` to build a simplified CBA based mainly on transport and delay costs and a flat per-trip benefit.:contentReference[oaicit

Task 3: Load Data and Inspect Tables

Goal: Load the core datasets and understand what each table contains.
Steps:

1. Open `Transport_CBA_Practical_with_less_componenets.ipynb` in Google Colab.
2. Run the first cells that import libraries (`pandas`, etc.).
3. Locate the cell that loads the CSV files, typically something like:

```
routes_df  = pd.read_csv("Routes_Table.csv")
trips_df   = pd.read_csv("Trip_Table.csv")
service_df = pd.read_csv("Service_Table.csv")
```

4. Run this cell and print the `head()` of each dataframe.
5. For each table, note:
   - the key columns (route IDs, distance, delay, etc.),
   - the key identifier used to merge tables (e.g. route or trip ID).

Checkpoint: You should be able to say in one sentence what each of the three tables represents.

Task 4: Merge Data and Compute Costs

Goal: Build one merged table and compute transport and delay costs per trip.
Expected logic (exact column names may differ):

- **Transport cost** per trip:

$$\text{Transport Cost} = \text{distance\_km} \times \text{cost\_per\_km}$$

- **Delay cost** per trip:

$$\text{Delay Cost} = \max(0, \text{delay\_minutes}) \times 0.5$$

  (assuming \$0.5 per minute of delay)
- **Total cost** per trip:

$$\text{Total Cost} = \text{Transport Cost} + \text{Delay Cost}$$

Steps:

1. Run the merge cell (look for something like `merged_df = ...` joining the three tables).
2. Confirm the merged dataframe has the necessary columns for distance, delay and cost per km.
3. Run the cell that computes:
   - transport cost,
   - delay cost,
   - total cost.
4. Print a few rows to confirm the calculations look reasonable (no negative delays, etc.).

Record:

- The formula used for delay cost (e.g. which penalty per minute is applied).
- A sample row showing distance, delay and total cost.

Task 5: Add Per-Trip Benefits and Compute NPV/BCR

Goal: Add a simple benefit assumption and compute NPV and BCR over multiple years.

Typical steps in the notebook:

1. Add a column for per-trip benefit, for example:

```
merged_df["Benefit"] = 10.0   # USD per trip (baseline)
```

2. Aggregate annual totals:
   - annual_costs,
   - annual_benefits,
   - choose analysis period (e.g. 10 years),
   - choose discount rate (e.g. 5%).
3. Compute present values and then:

$$\text{NPV} = \sum_t \frac{\text{Benefits}_t - \text{Costs}_t}{(1+r)^t}$$

$$\text{BCR} = \frac{\sum_t \frac{\text{Benefits}_t}{(1+r)^t}}{\sum_t \frac{\text{Costs}_t}{(1+r)^t}}$$

What to do:

1. Run the cell that defines the benefit column (check the default value, e.g. \$10 per trip).
2. Run the cells that compute annual totals and discount them.
3. Find and run the cell that prints the base **NPV** and **BCR** of the project.

Record (Baseline Basic Model):

- NPV (basic model, with current per-trip benefit),
- BCR (basic model),
- whether the project is classified as viable (BCR > 1) or not.

Task 6: Sensitivity Analysis – Increase Delay Penalty

Goal: Test how sensitive NPV/BCR are to higher delay costs.
Steps:
1. Locate the line where delay cost per minute is set (e.g. 0.5 USD/min).
2. Create a copy of that cell below and increase the penalty, for example:

```
delay_penalty = 1.0   # instead of 0.5, for high-delay scenario
```

   (or adjust the equivalent expression used in your notebook).
3. Re-run the cells that recompute delay cost, total cost, annual values and NPV/BCR.
4. Note the new NPV and BCR values for the *high-delay* scenario.

Compare:
- Does NPV go down? By how much?
- Does BCR drop below 1, making the project non-viable?

Reflection & Insight

Sensitivity analysis is exactly about this: "If reality is worse than planned, does the project still survive?":contentReference[oaicite:4]index=4

4 Activity 3 – Extended Python CBA Model

This part uses `CBA_Practical_Notebook_with_extra_components.ipynb` to add missing costs and benefits: capital cost, operating cost, emissions, safety, congestion, and economic development.

Task 7: Run the Extended Notebook with Default Values

Goal: Get baseline NPV and BCR for the extended model.
Steps:
1. Open `CBA_Practical_Notebook_with_extra_components.ipynb` in Colab.
2. Use **Runtime** → **Restart and run all** to execute the notebook with default assumptions.
3. Let the notebook finish. Scroll to the end and locate:
   - the printed values of NPV,
   - the printed value of BCR,
   - any final message like "Project is economically viable" or "Not viable".

Record (Extended Baseline):
- NPV (extended model, default values),
- BCR (extended model),
- Viability recommendation (Viable / Not Viable).

Task 8: Identify Additional Costs and Benefits

Goal: See how the extended model fills the "missing" components.
Typical variables in the extended notebook:

- **Project-level costs**
  - `capital_cost` (one-time, e.g. $500,000),
  - `annual_operating_cost` or similar (e.g. $100,000 per year).
- **Additional annual benefits**, often stored in a dictionary like:

```
additional_annual_benefits = {
    "emission_reduction": 200000,
    "safety":             100000,
    "congestion":          50000,
    "economic_dev":       300000,
}
```

- These are added on top of per-trip benefits when computing the total discounted benefits.:contentReference[oaicite:6]index=6

Steps:

1. Find where `capital_cost` and annual operating costs are defined.
2. Find where `additional_annual_benefits` (or similar) is defined.
3. Briefly comment in your notes which components (emissions, safety, etc.) are now included that were missing before.

Task 9: Sensitivity Test in Extended Model

Goal: Compare base vs. high-delay scenario in the extended model.
Steps:

1. Locate the part of the extended notebook where delay cost per minute is defined (similar to Task 6).
2. Adjust the penalty upwards (for example double it) to create a "high-delay" case.
3. Re-run the relevant cells or the full notebook to get updated NPV and BCR.
4. Fill in a small comparison table in your notes:

Scenario	NPV	BCR
Extended baseline	...	...
Extended high-delay	...	...

Questions:

- Does adding extra benefits (emissions, safety, etc.) make the project more robust under high delays?
- Does the BCR stay above 1 in your high-delay case, or does it fall below?

Reflection & Insight

A professional CBA should include most major cost and benefit categories. Leaving out benefits (e.g. emissions, safety) can underestimate viability; leaving out costs (e.g. capital, operating) can overestimate it.:contentReference[oaicite:7]index=7

5 Activity 4 – Homework: Redesign for Viability

In many initial runs, the project may show a **negative NPV** and **BCR ¡ 1**, meaning it is not viable. The homework is to redesign the scenario so that the project becomes viable by adjusting benefits and/or costs in a realistic way.:contentReference[oaicite:8]index=8

Task 10: Baseline Run for Homework

Goal: Establish a clear starting point in the extended model.
Steps:

1. Open `CBA_Practical_Notebook_with_extra_components.ipynb`.
2. Ensure all variables are set to their default values (you can use the version provided by the lecturer).
3. Run all cells once from top to bottom.
4. Record carefully:
   - Baseline NPV,
   - Baseline BCR,
   - Project recommendation (Viable / Not Viable).

Task 11: Redesign the Project Assumptions

Goal: Make the project economically viable by realistic changes to benefits and/or costs.

You must change at least 2–3 items. Example levers (from the guide):

- **Increase per-trip benefit** (time savings):
  - For example, change:

```
merged_df["Benefit"] = 10
```

    to

```
merged_df["Benefit"] = 30
```

    as suggested (representing higher value of time).:contentReference[oaicite:9]index=9
- **Add or increase annual benefits:**
  - Emission reduction (e.g. 200,000/year),
  - Safety benefits (e.g. 100,000/year),
  - Congestion reduction (e.g. 50,000/year),
  - Economic development (e.g. 300,000/year).
  - Adjust `additional_annual_benefits` accordingly.
- **Reduce costs where realistic:**
  - Assume improved operations (lower delay cost),
  - Optimised routes (lower transport cost or cost per km),
  - Reduced annual operating costs.
- **Government subsidy or grant (optional):**
  - Capital cost can be fully or partially subsidised, e.g.:

```
capital_cost = 0   # fully subsidised capital
```

Constraints:

- Changes should be *plausible* for a public transport project.
- Do not artificially change the discount rate unless specifically allowed.

Task 12: Recalculate & Compare Scenarios

Goal: Quantify how your redesign changed project viability.

Steps:

1. After editing the relevant lines, run all cells again from top to bottom.
2. Record the **new** NPV and BCR.
3. Create a simple comparison table in your report or notes:

Scenario	NPV	BCR
Baseline (default)	...	...
Redesigned	...	...

4. Check whether BCR is now above 1 and NPV is positive.

Deliverables (for homework):

- Modified `.ipynb` file with your changes.
- Short report (PDF) that:
  - explains which variables you changed and why,
  - presents before/after NPV and BCR,
  - states clearly whether the project is now economically viable,
  - briefly justifies the redesign from a policy perspective.

Reflection & Insight

CBA in practice is iterative: you rarely get a viable project on the first try. Professionals adjust design, phasing, funding and operations until the numbers and the policy goals make sense together.:contentReference[oaicite:10]index=10