**Creating Custom Data for the Fashion Reverse Logistics Tool**

This tutorial will guide you through the process of creating your own custom data for the Fashion Reverse Logistics optimization tool based on real-world locations and requirements.

**Option 1: Using the CSV Format Directly**

If you have specific shop locations and requirements, you can create a CSV file directly:

**Step 1: Create a CSV File**

Create a CSV file with the following columns:

* id: Numbering of locations (0 for depot, 1 to n for shops)
* x\_coord: X-coordinate or longitude
* y\_coord: Y-coordinate or latitude
* [demand](vscode-file://vscode-app/c:/Users/jpern/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html): Number of items to collect (0 for depot, 1-3 for shops)

Example format:

*id,x\_coord,y\_coord,demand*

*0,-3.7038,40.4168,0*

*1,-3.7103,40.4230,2*

*2,-3.6883,40.4054,1*

Important notes:

* The first row (id=0) should always be your depot location
* All other rows are shops that need to be visited
* The demand at each shop is how many items need to be collected

**Step 2: Create a Metadata File**

Create a text file with the same name as your CSV but with \_meta.txt suffix. It should contain a single line specifying the vehicle capacity:

*capacity=10*

*alpha=0.15*

*beta=0.02*

Where:

* capacity is the maximum load your vehicle can carry
* alpha is the base CO₂ emissions per km (kg/km)
* beta is the load-dependent emission factor (kg/km/kg)

**Step 3: Save Both Files**

Save both files in the same directory, for example:

* my\_stores.csv
* my\_stores\_meta.txt

**Option 2: Using Google Maps for Real Locations**

For real-world locations, you can use Google Maps to find coordinates:

**Step 1: Locate Your Stores in Google Maps**

1. Open Google Maps (maps.google.com)
2. Find your depot location
3. Right-click on the location and select "What's here?"
4. The coordinates will appear in the info card at the bottom
5. Note these coordinates (latitude, longitude)
6. Repeat for all shop locations

**Step 2: Create a Spreadsheet**

1. Create a spreadsheet in Excel or Google Sheets
2. Add columns: id, x\_coord, y\_coord, demand
3. For the depot (id=0):
   * x\_coord = longitude
   * y\_coord = latitude
   * demand = 0
4. For each shop:
   * x\_coord = longitude
   * y\_coord = latitude
   * demand = number of items to collect (estimate)

**Step 3: Convert to CSV Format**

1. Save the spreadsheet as a CSV file
2. Create the metadata file including emissions parameters:

*capacity=10*

*alpha=0.15*

*beta=0.02*

1. Place both files in the project's directory

**Option 3: Using the Application to Generate Data**

You can also use the application to generate data and then modify it:

**Step 1: Generate a Random Instance**

1. Launch the application with python [main.py](http://\_vscodecontentref\_/0) --gui
2. Click "Create Sample Instances"
3. This creates sample instances in the instances folder

**Step 2: Modify the Generated Data**

1. Open one of the generated CSV files (e.g., small\_instance.csv)
2. Replace the coordinates with your actual store locations
3. Update the demand values based on your actual collection needs
4. Save the file

**Step 3: Update the Metadata**

1. Open the corresponding metadata file (e.g., small\_instance\_meta.txt)
2. Update the vehicle capacity to match your actual vehicle capacity
3. Update the emissions parameters alpha and beta if needed
4. Save the file

**Converting Real-World Distances**

If you're working with real geographical coordinates, you might need to adjust the distance calculation:

**Option 1: Scale the Coordinates**

For small geographical areas, you can use the raw coordinates as they are. The optimizer will find the shortest routes regardless of the absolute distances.

**Option 2: Convert to Kilometers/Miles**

For more accurate distance representation:

1. Open the CSV file in a spreadsheet program
2. Calculate distances between points using the Haversine formula
3. Create a new file with calculated distances
4. Update the application to use these distances

**Tips for Creating Realistic Data**

1. **Vehicle Capacity**: Consider how many items your vehicles can actually carry
2. **Demand Estimation**: Estimate how many items need to be collected from each shop
3. **Emissions Parameters**: Set alpha and beta based on your vehicle's specifications:
   * For a small van: α≈0.15 kg/km, β≈0.02 kg/km/kg
   * For a medium truck: α≈0.25 kg/km, β≈0.03 kg/km/kg
4. **Time Windows**: If certain shops are only open at specific times, note this (for future enhancements)
5. **Depot Location**: The depot should be your actual distribution center or starting point

**Testing Your Data**

After creating your custom data:

1. Run the application: python [main.py](http://\_vscodecontentref\_/1) --gui
2. Select "Browse" and navigate to your custom CSV file
3. Choose your preferred solution method (regular or emissions-optimized)
4. Run the solver
5. Verify the results match your expectations

**Example: Converting From a Store List**

Let's say you have a list of stores with addresses:

1. Use Google Maps or a geocoding service to convert addresses to coordinates
2. Create a CSV file with this structure:

*id,x\_coord,y\_coord,demand*

*0,-3.7038,40.4168,0*

*1,-3.7103,40.4230,2*

*2,-3.6883,40.4054,1*

1. Create a metadata file with your vehicle capacity and emissions parameters:

*capacity=15*

*alpha=0.15*

*beta=0.02*

1. Save both files and use them in the application

**Emissions Optimization**

If you want to optimize for CO2 emissions instead of just distance:

1. Set accurate alpha and beta values in your metadata file
2. When running the solver, select the "CP-Emissions" method in the GUI
3. The solution will minimize CO2 emissions considering both distance and load