GCAM CSV Extraction Script Documentation

Overview

The `gcam_extract_csv_from_project_files.R` script is designed to extract specific variables from GCAM (Global Change Analysis Model) project files and export them as CSV files. This script processes multiple scenarios simultaneously and combines their data into unified CSV outputs for analysis and visualization.

Purpose

This script automates the extraction of GCAM output data by: - Reading GCAM project files (.dat format) - Extracting specified variables (e.g., CO2 emissions, land allocation, commodity prices) - Combining data from multiple scenarios into single CSV files - Organizing output for downstream analysis

Requirements

R Packages

The script requires the following R packages: - `devtools` - Development tools for R packages - `dplyr` - Data manipulation and transformation - `rgcam` - GCAM-specific data reading and processing functions - `rjson` - JSON file parsing

Install missing packages using:

```r
install.packages(c("devtools", "dplyr", "rjson"))
# rgcam may require special installation from GCAM repository
```

Input Files

1. JSON Configuration File

The script uses a JSON configuration file to specify: - Which variables to extract - Output file locations - Scenario names - Project file paths

JSON Structure

```json
[
    {
        "variable": "variable name as it appears in GCAM",
        "outputFile": "path/to/output.csv",
        "scenarios": ["Scenario1", "Scenario2"],
        "projectFiles": ["path/to/scenario1.dat", "path/to/scenario2.dat"]
    }
]
```

JSON Fields Explained

- **variable**: The exact name of the GCAM query/variable to extract (must match GCAM's internal variable names)
- **outputFile**: The destination path and filename for the CSV output
- **scenarios**: Array of descriptive scenario names (used as labels in the output)
- **projectFiles**: Array of GCAM project file paths (must correspond to scenarios in the same order)

2. GCAM Project Files

Binary `.dat` files generated by GCAM containing model output data. These files are produced by loading GCAM XML outputs into the GCAM project database system.

Example Usage

Example JSON Configuration

Based on the provided example file `gcam_extract_csv_from_project_files.json`:

```json
[
    {
        "variable": "ag commodity prices",
        "outputFile": "./../2025_DiVittorio_et_al_gcam/ag_commodity_prices.csv",
        "scenarios": ["Control", "Full feedback"],
        "projectFiles": [
            "./../2025_DiVittorio_et_al_gcam/20240730_SSP245_ZATM_control.dat",
            "./../2025_DiVittorio_et_al_gcam/20240730_SSP245_ZATM_full_feedback.dat"
        ]
    },
    {
        "variable": "CO2 emissions by sector",
        "outputFile": "./../2025_DiVittorio_et_al_gcam/co2_emissions_sectors.csv",
        "scenarios": ["Control", "Full feedback"],
        "projectFiles": [
            "./../2025_DiVittorio_et_al_gcam/20240730_SSP245_ZATM_control.dat",
            "./../2025_DiVittorio_et_al_gcam/20240730_SSP245_ZATM_full_feedback.dat"
        ]
    }
]
```

This configuration will: 1. Extract agricultural commodity prices from both Control and Full feedback scenarios 2. Create `ag_commodity_prices.csv` containing combined data 3. Extract CO2 emissions by sector from the same scenarios 4. Create `co2_emissions_sectors.csv` containing combined emissions data

Running the Script

Command line execution:

```
Rscript gcam_extract_csv_from_project_files.R path/to/config.json
```

Multiple JSON files:

```
Rscript gcam_extract_csv_from_project_files.R config1.json config2.json config3.json
```

Using the example file:

```
Rscript gcam_extract_csv_from_project_files.R gcam_extract_csv_from_project_files.json
```

Output Files

CSV Structure

Each output CSV file contains: - All columns from the GCAM query result (varies by variable) - A `scenario` column identifying which scenario each row belongs to - Combined data from all specified scenarios in a single file

Example Output Structure

For "ag commodity prices", the CSV might look like:

```
region, sector, subsector, year, value, scenario
USA, Corn, ..., 2020, 150.5, Control
USA, Corn, ..., 2025, 155.2, Control
USA, Corn, ..., 2020, 148.3, Full feedback
...
```

Script Workflow

1. **Parse Command Line Arguments**: Reads JSON file path(s) from command line
2. **Validate Input**: Checks that at least one JSON file was provided
3. **Process Each JSON File**: Iterates through all specified JSON files
4. **Process Each Variable Block**: For each variable specification in the JSON:
   - Loads each project file
   - Extracts the specified variable using `getQuery()`
   - Converts to dataframe and adds scenario label
   - Combines all scenario dataframes
   - Writes combined dataframe to CSV
5. **File Extension Handling**: Automatically appends `.csv` if not present in output filename

Error Handling

The script will stop execution with an error message if: - No JSON configuration file is provided on the command line - Project files cannot be found or loaded - Variable names don't match available queries in the project files - Invalid JSON formatting

Best Practices

1. **Use Relative Paths**: The example uses relative paths (`../`) for portability across systems
2. **Consistent Scenario Order**: Ensure scenarios array order matches projectFiles array order
3. **Descriptive Scenario Names**: Use clear scenario labels for easier data interpretation
4. **Variable Name Accuracy**: Verify variable names match GCAM's internal query names exactly
5. **Directory Structure**: Ensure output directories exist before running the script

Common Variables to Extract

Based on the example, common GCAM variables include: - `ag commodity prices` - Agricultural commodity price trajectories - `CO2 emissions by sector` - Sectoral CO2 emissions - `CO2 emissions by region` - Regional CO2 emissions - `detailed land allocation` - Land use and land cover change data

Consult GCAM documentation for complete list of available queries.

Troubleshooting

"No such file or directory" errors: Check that all project file paths are correct and files exist

"Query not found" errors: Verify that the variable name exactly matches a query available in your GCAM project files

Package loading errors: Install missing packages and ensure rgcam is properly installed

Empty CSV files: Verify that project files contain data for the specified variables and scenarios

Additional Notes

- The script preserves all original columns from GCAM queries
- Data from different scenarios is stacked (row-wise concatenation)
- The `scenario` column is added for filtering and grouping in subsequent analyses

- CSV files use standard comma-separated format with headers
- Row names are included in the CSV output (first column)