BEAD: Reproducible Computational Research Made Simple

Miklós Koren (Central European University) Krisztián Fekete (Central European University)

RSEcon, September 10, 2025[1]

[1]This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 313164). The views expressed are those of the authors and do not necessarily reflect those of the ERC or the European Commission.

The Editor Says You Have One Week

- Journal editor: “substantial revision invited”
- Reviewers liked Figure 1 (life expectancy vs GDP per capita)
- Concern about health data source
- You need to:
  - Address reviewer concerns
  - Redo analysis with new data
  - Recreate Figure 1
  - Submit within one week

But Your Submission is Months Old

- Research submitted months ago
- Team has been improving data cleaning since then
- Different statistical methods now
- **First question**: How did I actually produce Figure 1?

Research Results are Functions

$$\text{Figure 1} = f(\text{code}, \text{data})$$

- Results depend on both algorithms and data
- Code under version control (Git) -> Yes
- Tagged commit at submission -> Yes
- **But what about the data?**

Data is Also a Function

$$\text{data}_1 = f(\text{code}_2, \text{data}_2)$$

- Data produced by wrangling/cleaning steps
- Which countries dropped?
- What transformations applied?
- Feature engineering details?
- **Chain of data provenance**

Real-World Data Pipelines

- Multiple datasets merged
- Many cleaning steps
- Different versions coexisting
- Green = using latest version
- Red/yellow = outdated dependencies
- Complex dependency graph

The Data Provenance Problem

Why it's complex:

1. **Frequent changes**: Code and data both evolve
2. **Complex pipelines**: Many steps, multiple datasets
3. **Tool heterogeneity**: Python, R, SQL, DuckDB all in one project

Team Dynamics Make it Worse

- Master/PhD students graduate and leave
- Different team members use different tools
- **Every meeting starts with**:
  - “Who knows how to reproduce this?”
  - “Who has the data?”
  - “That person already left. . . ”

Existing Solutions

Version Control (Git)

- Great for code
- Not suitable for large binary data

Data Version Control (DVC)

- Similar spirit to BEAD
- More complex than needed
- dvc.org

Orchestration Tools

- Apache Airflow (Python) - airflow.apache.org
- dbt (SQL) - getdbt.com
- KNIME (no-code) - knime.com
- Too complex for heterogeneous teams

Enter BEAD

A command-line tool that ensures your output is a function of your input

- Much simpler than alternatives
- Language agnostic
- Works with heterogeneous teams
- Different experience levels
- Different operating systems

What BEAD Does NOT Do

Not a code runner

- You run your own code
- Python, R, Stata, SQL - doesn't matter

Not a file delivery system

- File system stores your files
- You copy/move files yourself

Only requirement:

- Works with flat files on file system
- Files not too big (20GB works fine)

What BEAD Enforces

Input data is immutable

- Cannot modify raw data
- Forces good practices
- Preserves data lineage

Core BEAD Concepts

The BEAD

- Self-contained computational unit
- Contains code, data, results
- Packaged as ZIP file
- Remembers exact provenance

Simple Commands

```
bead new my-analysis
bead input add source-data
bead save results
```

Demo Time

Let’s see BEAD in action with a real example. . .

Demo Part 1: Create Analysis with Two Data Sources

```
$ bead new figure1
Created "figure1"

$ cd figure1
$ bead input add life-expectancy
Loading new data to life-expectancy ... Done

$ bead input add gdp-per-capita
Loading new data to gdp-per-capita ... Done
```

Demo Part 2: Workspace Structure

```
$ ls -la
drwxr-xr-x  .bead-meta     # Metadata and provenance
dr-xr-xr-x  input/         # Read-only input data
drwxr-xr-x  output/        # Your results go here
drwxr-xr-x  temp/          # Temporary files
```

Input folder is **read-only** - can't accidentally modify source data!

Demo Part 3: Process Data with SQL

```
$ cat > analyze.sql << 'EOF'
-- Join GDP and life expectancy data
WITH joined_data AS (
    SELECT l.Country, l.Year, l.Life_expectancy,
           g.GDP_per_capita_USD
    FROM read_csv_auto('input/life-expectancy/life_expectancy.csv') l
    JOIN read_csv_auto('input/gdp-per-capita/gdp_per_capita.csv') g
    ON l.Country = g.Country AND l.Year = g.Year
    WHERE l.Year = 2021
)
SELECT Country, GDP_per_capita_USD, Life_expectancy,
       bar(Life_expectancy, 65, 85, 30) as Chart
FROM joined_data ORDER BY GDP_per_capita_USD DESC;
EOF
```

Demo Part 4: Run Analysis

```
$ duckdb < analyze.sql
+----------------+------------+----------+--------------------------------
|    Country     | GDP/capita | Life Exp | Life Expectancy (65-85 years) |
+----------------+------------+----------+--------------------------------
| United States  | $    69288 |   76.3   | ##################
| Germany        | $    50802 |   81.3   | ########################
| United Kingdom | $    47334 |   81.3   | ########################
| China          | $    12556 |   77.1   | ###################
| World          | $    12237 |   71.0   | #########
| India          | $     2257 |   69.7   | #######
+----------------+------------+----------+--------------------------------
```

Demo Part 5: Save as BEAD

```
$ duckdb < analyze.sql > output/figure1.txt
$ bead save
Successfully stored bead at figure1_20250825T184236645231+0200.zip
```

Every bead has: - Unique timestamp - Complete provenance - All code and results

Demo Part 6: Data Update Scenario

Editor asks: “Please update with 2022-2023 data”

```
$ cd ../life-expectancy
$ echo "World,2022,71.3" >> output/life_expectancy.csv
$ echo "World,2023,71.5" >> output/life_expectancy.csv
$ bead save
Successfully stored bead at life-expectancy_20250825T184416025424+0200.zip
```

Demo Part 7: Clean Up Workspace

```
$ bead zap
Deleted workspace life-expectancy

$ ls
figure1/    gdp-per-capita/    bead-box/
```

Workspace gone but bead preserved!

Demo Part 8: Update Analysis

```
$ cd figure1
$ bead input update life-expectancy
Removing current data from life-expectancy
Loading new data to life-expectancy ... Done

$ duckdb < analyze.sql > output/figure1.txt
$ bead save
Successfully stored bead at figure1_20250825T184443082049+0200.zip
```

Analysis automatically uses latest data version!

How BEAD Solves Our Problems

Problem	BEAD Solution
“What data did we use?”	Every bead remembers exact version
“It worked on my machine”	Exact same setup for everyone
“That person left”	Work stays reproducible
Team uses different tools	Language agnostic
Complex pipelines	Chain beads together

Real Research Example

- Multiple datasets connected
- Many cleaning steps
- Green = using latest data version
- Some steps outdated
- BEAD tracks entire dependency graph

BEAD in Practice

Step 1: Create workspace

```
bead new health-analysis
```

Step 2: Load inputs

```
bead input add wdi-data
bead input add health-metrics
```

Step 3: Run analysis

```
python clean_data.py
R --file=analyze.R
```

Step 4: Save snapshot

```
bead save figure1-v2
```

Why BEAD is Different

- **Simple**: 4 commands to learn
- **Universal**: Any language, any tool
- **Portable**: Just ZIP files
- **Secure**: Data stays on your servers
- **Transparent**: Open source, no vendor lock-in

For Research Software Engineers

- Minimal learning curve for researchers
- No infrastructure requirements
- Works with existing workflows
- Complements version control
- Enables true reproducibility

Get Started

Installation

```
pip install bead
```

Documentation

codedthinking.github.io/bead.zip

Source Code

github.com/codedthinking/bead.zip

Key Takeaways

1. **Data provenance is hard** - especially with changing teams
2. **Existing tools too complex** - for heterogeneous research teams
3. **BEAD keeps it simple** - focuses on one thing well
4. **Reproducibility becomes automatic** - not an afterthought

Thank You!

Questions?

Contact

- Web: bead.zip
- GitHub: github.com/codedthinking/bead.zip

Funded by the European Union

European Research Council
Established by the European Commission

References

- **World Development Indicators**: data.worldbank.org/indicator
- **DVC (Data Version Control)**: dvc.org
- **Apache Airflow**: airflow.apache.org
- **dbt**: getdbt.com
- **KNIME**: knime.com