DataFrame Serialization

Purpose

The script serializes the processed bathymetry data (with and without smoothing and tide correction) into two formats:

- **Binary format** (`.bin`): For fast, native Julia storage and reloading.
- **CSV format** (`.csv`): For interoperability with Excel, Python, R, or other tools.

Structure of the DataFrame

The DataFrame `df` contains the following columns:

Column	Description
`baths`	Raw bathymetry (depths from unsmoothed bottom picks)
`baths_smoothed`	Smoothed bathymetry (with spatial-temporal filtering)
`tide_correction`	Final bathymetry corrected with tidal data

Each row represents a ping from the selected sonar transect.

Binary Serialization (`.bin`)

Serialization is done using Julia's built-in `Serialization` module. It stores the full internal structure of the DataFrame, preserving types and metadata.

```
using Serialization

# Save DataFrame to binary
open("df_controlados_murcia.bin", "w") do io
    serialize(io, df)
end

# Load DataFrame from binary
df = deserialize(open("df_controlados_murcia.bin"))
```

Advantages:

- Very fast read/write.
- No data loss from rounding or formatting.
- Fully preserves Julia types (e.g., `Float64`, `String`, `Vector{T}`).

Disadvantages:

- Not portable to non-Julia environments.
- Not human-readable.

CSV Export

After serialization to binary, the DataFrame is exported as CSV for compatibility:

```
using CSV

CSV.write("df_controlados_murcia.csv", df)
```

Advantages:

- Readable by Excel, R, Python (Pandas), etc.
- Easy to inspect or share.

Disadvantages:

- Text format, so slower and less precise for large/float-heavy data.
- No support for nested or complex types.