ICT for Civic Data — Crash Course 2026

Verification and Cleaning

Self-Paced Review — Section D

civicliteraci.es

Why This Matters

> Make data trustworthy

Make data trustworthy so that charts, maps, and arguments built on it are correct. These are the **Verify** and **Clean** steps of the pipeline.

Verification checks that the data is what you think it is. Cleaning fixes errors, standardises values, and consolidates categories. Both steps must happen before analysis, or every conclusion you draw could be wrong.

Before You Start

> Why cleaning matters

The FloodArchive dataset has:

- **5 ways to spell Philippines:** Philippines, Phillipines, Phillippines, Philipines, philippine
- **9 ways to say Heavy rain:** Heavy Rain, Heavy rain, heavy rain, Monsoonal rain, Monsoon, Torrential rain, Tropical storm, Brief torrential rain, Rain

Each spelling variant becomes a separate category in a chart or filter. A bar chart of flood causes shows 9 bars instead of 3. A country filter misses events because the name doesn't match. A COUNTIF formula returns 12 instead of 47.

Messy data does not produce errors. It produces **wrong answers that look right**.

> Who is invisible?

After designing your data table, ask: **what can't this data capture? Who is missing from the dataset?**

- Missing data is not the same as zero
- People sleeping rough but not in contact with services don't appear in homelessness data
- Communities in regions where no one is counting don't appear in disaster data
- Events that happened before systematic recording began don't appear in historical data

This is a critical question from Lecture 1. It applies at every stage, but it matters most during verification: when you look at the data, ask who and what is **not** in it.

> Eight principles keep your cleaning auditable

#	Principle	Why
1	**Never modify originals**	Keep the raw file untouched. Work on a copy.
2	**Never let AI modify content directly**	AI can write formulas and scripts. It should not edit cell values.
3	**Look at the data yourself**	Sort, scroll, spot-check. Build intuition for what's wrong.
4	**Create new columns, don't overwrite**	Keep originals alongside cleaned versions for comparison.
5	**One problem at a time**	Fix spelling first. Then categories. Then missing values.
6	**Right tool for the job**	Sheets for transparency, OpenRefine for scale, scripts for automation.
7	**Document every decision**	Why did you merge these categories? Write it down.
8	**Verify after each step**	Check the result before moving to the next problem.

> When to use which tool

Google Sheets

- Under ~50,000 rows
- You want to **see every step** happening
- Formulas are transparent and auditable
- Best for learning and small datasets
- Good for collaborative work (share the sheet)

OpenRefine

- Any dataset size (handles millions of rows)
- **Bulk standardisation** of messy categories
- Clustering algorithms find similar values automatically
- Every action is recorded and undoable
- Best for repetitive, large-scale cleaning

Scripts (Python, R) are best for automated pipelines that run repeatedly. For one-off cleaning in this course, Sheets and OpenRefine cover everything.

> Data format shapes what you can ask

The FloodArchive stores multi-country floods as **one row** with an OtherCountry column:

flood-row.csv

ID	Country	OtherCountry	Date	Dead
4567	Brazil	"Argentina, Paraguay"	2019-03-15	12

A more expressive format would be **hierarchical**: each country gets its own record with country-specific dates and impacts. But CSV is flat: one row, one record.

When you encounter limitations in your data, ask: **is this a data quality problem or a data modeling problem?** The Philippines misspellings are data quality. The OtherCountry column is data modeling. Different problems need different solutions.

Walkthrough

> Three methods get external data into Sheets

Three methods:

1. IMPORTDATA formula — pulls live data from a URL directly into the sheet:

```
=IMPORTDATA("https://example.com/data.csv")
```

The data updates when the URL changes. Useful for APIs and live datasets.

2. Upload CSV — File → Import → Upload. Google Sheets asks which separator to use: comma, semicolon, or tab. If the data looks wrong (everything in one column), try a different separator.

3. Paste special (values only) — if you have formula-generated data (from IMPORTDATA or calculations), copy it and use Edit → Paste special → Values only. This detaches the values from formulas so you can edit them freely.

> Explore before you clean

Before cleaning, explore:

- **Sort columns** to see what values exist. Sort the Country column alphabetically and scan for near-duplicates.
- **Filter** to isolate subsets. Filter Country = "Philippines" (but you'll miss "Phillipines").
- **Scroll** through the data. AI reads the first 50 rows carefully and assumes the rest are similar. You need to check the middle and end.

Pivot tables for quick summaries: Insert → Pivot table. Set Rows = the column you want to group by, Values = COUNTA (counts non-empty cells). A pivot table on Country immediately shows you all 5 Philippines variants and their counts.

Conditional formatting with a colour scale on numeric columns reveals visual patterns: outliers, clusters, and gaps.

> Cleaning: filter and overwrite

A controlled alternative to Find and Replace:

1. Click the column filter dropdown
2. Uncheck "Select all," then check only the wrong value (e.g., "Phillipines")
3. The sheet now shows only rows with that value
4. Select the visible cells in that column
5. Type the correct value ("Philippines") and press **Ctrl+Enter** (fills all selected cells)
6. Remove the filter to see all rows again

This is safer than Find and Replace because you **see exactly which rows will change** before changing them. Find and Replace can modify values inside longer strings unexpectedly.

> Cleaning: invisible spaces

Google Sheets catches **trailing** spaces with its automated suggestion ("Did you mean Philippines?"). But it misses **leading** spaces.

The TRIM function removes both leading and trailing spaces:

Workflow: create a helper column with `=TRIM(A2)`, drag down, copy the column, paste as values over the original, delete the helper column.

Why it matters: `" Philippines"` and `"Philippines"` look identical on screen but are different values to formulas. COUNTIF, FILTER, and pivot tables all treat them as separate categories.

> Categorisation with FILTER lookup

When many raw values map to a few categories, use a **mapping sheet**:

1. In a new sheet ("Mapping"), use `=UNIQUE(Data!E:E)` to list all distinct values from the Cause column
2. Sort the list alphabetically
3. Manually assign a category to each value in an adjacent column (e.g., "Heavy Rain" → "Rain", "Monsoonal rain" → "Rain")
4. Back in the data sheet, use FILTER to look up the category:

Formula: `=FILTER(Mapping!B:B, Mapping!A:A = E2)`. Transparent: anyone can open the Mapping sheet and see every decision.

> Categorisation with IFS and SEARCH

An alternative for pattern-based categories. SEARCH finds text within a cell:

```
=IFS(
  IFERROR(SEARCH("rain", E2), 0),  "Rain",
  IFERROR(SEARCH("storm", E2), 0), "Storm",
  IFERROR(SEARCH("snow", E2), 0),  "Snow",
  TRUE,                             "Other"
)
```

The gotcha: SEARCH returns an **error** when it doesn't find the text, not FALSE. This blocks the entire IFS chain. IFERROR wraps each SEARCH to convert the error to 0 (falsy), letting IFS proceed to the next condition.

Without IFERROR: the formula crashes on the first non-match. With IFERROR: the formula tests each pattern in order and falls through to "Other."

> Zero and blank mean different things

The OtherCountry column contains three kinds of values:

- **0** — no other country affected
- **Blank** — unknown or not recorded
- **Country names** — "Argentina, Paraguay"

Create a helper column to classify each row:

Formula: `=IF(OR(A2=0, A2="0"), "Single country", IF(A2="", "Unknown", "Multi-country"))`.
Alternative with IFS (flatter): `=IFS(OR(A2=0, A2="0"), "Single country", A2="", "Unknown", TRUE, "Multi-country")`

> Validating with the REST Countries API

Multi-country flood events: do the paired countries actually share a border?

- Brazil + Argentina → **plausible** (shared border)
- Brazil + Vietnam → **suspicious** (no shared border)

The REST Countries API has a `borders` field listing each country's neighbours by ISO code:

```
https://restcountries.com/v3.1/name/brazil?fields=borders
→ ["ARG", "BOL", "COL", "GUF", "GUY", "PRY", "PER", "SUR", "URY", "VEN"]
```

This is geographic knowledge that formulas can't provide. You need an external data source. Use OpenRefine's "Add column by fetching URLs" feature or write a script that checks each country pair against the API.

> Extract the year with LEFT

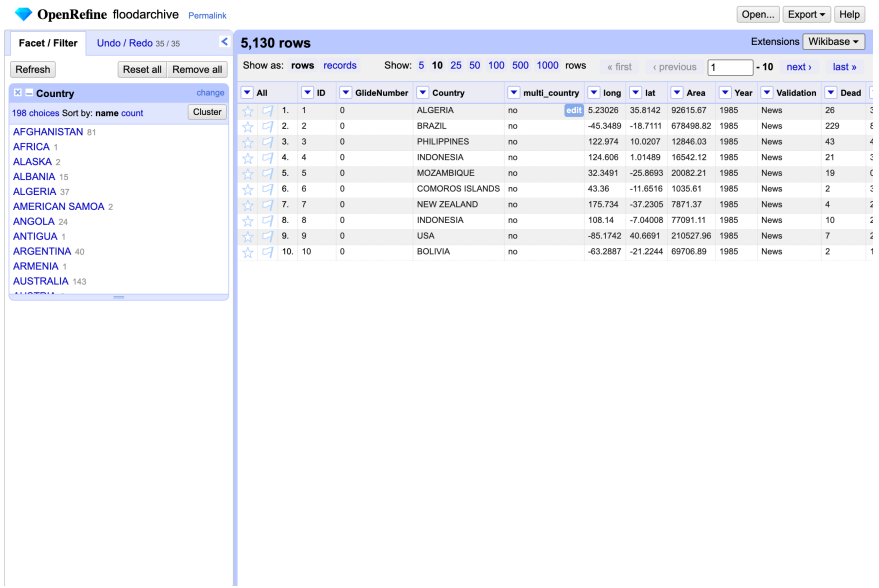
The LEFT function extracts characters from the start of a text string:

Formula: `=LEFT(A2, 4)` extracts the first 4 characters, giving you the year from an ISO date string. Useful when you need to group events by year but the date column contains full dates.

Helper column workflow: create a "Year" column with `=LEFT(A2, 4)`, drag down. Now you can pivot by year, filter by decade, or chart trends over time.

> OpenRefine groups similar values automatically

OpenRefine finds and fixes inconsistencies across thousands of values:



A **text facet** on a column shows all unique values and counts. Click **Cluster** to group similar values.

> Three clustering algorithms find different problems

Algorithm	What it finds	Example
Key collision	Duplicates after normalisation (case, whitespace)	"Heavy Rain" = "heavy rain"
Nearest neighbour	Near-matches by edit distance	"Philippines" ≈ "Phillipines"
Phonetic (Metaphone)	Words that sound alike	"Fillipines" ≈ "Philippines"

> OpenRefine: advanced features

Beyond text facets and clustering:

- **Timeline facet** on date columns → filter by date range using a slider. Useful for identifying time periods with sparse or suspicious data.
- **Trim whitespace** → Edit cells → Common transforms → Trim leading and trailing whitespace. Bulk version of the Sheets TRIM approach.
- **Fetch URLs** → Add column by fetching URLs. Send API requests for each row (e.g., REST Countries API for border validation). Rate-limited automatically.
- **Parse JSON** → use GREL `parseJson()` to extract fields from API responses into new columns.
- **GREL substring** → `value[0,4]` extracts the first 4 characters (like LEFT in Sheets).

> Prompt: asking AI for formula syntax

« I need to cluster similar values in a text column in OpenRefine. Walk me through the steps to apply nearest-neighbour clustering. I'm a beginner using OpenRefine for the first time. »

« I have a column with dates in "YYYY-MM-DD" format in OpenRefine. What GREL expression extracts just the year? »

Objective Steps Output shape Reasoning

Use the **chatbot** for syntax questions like these. Save CLI tokens for tasks that need to touch your files.

> Paste as values before exporting

After cleaning, prepare the data for analysis:

1. Create a **new sheet** in the same workbook
2. Copy only the **columns you need** for analysis (original + cleaned columns)
3. **Paste as values** (Edit → Paste special → Values only) to detach from formulas
4. Review the result: are all cleaning formulas resolved to final values?
5. **Download as CSV** (File → Download → Comma-separated values)
6. Upload the CSV to your Codespace for scripting and map-building

Why a separate sheet: your cleaning sheet has helper columns, formulas, and intermediate steps. Your analysis needs a clean, flat CSV with only the final values.

Behind the Approach

> Trust the code, not the reasoning

Scripts are deterministic: the same script on the same data produces the same result every time. AI can write scripts that filter rows, create columns, and convert formats. These are trustable because they are testable.

But AI should never directly edit data content. If the logic can't be expressed as a formula or script, do it manually.

Exception: AI writing a formula inside a spreadsheet cell IS trustable. You see the result immediately in the cell and can verify it in place. The formula `=TRIM(A2)` does exactly one thing, and you can check whether the output is correct by looking at it.

> Build good taste in data

Manual spreadsheet work develops intuition for what right data looks like:

- Sort the column. Scroll through. Does anything stand out?
- Check the min and max. Do they make sense?
- Count the categories. Are there too many?

If you rely on AI to do all the data work, you will never develop this judgment. You won't notice when "Dead = 0" could mean zero deaths or unknown. You won't catch that a country column has 47 unique values when only 30 countries are in the dataset.

AI gets lazy with large files: it reads the first 50 rows carefully, then assumes the rest are similar. **Verify values near the end of outputs, not just the top.**

> Core vs peripheral competencies

Core (build skill here)	Peripheral (AI handles well)
> Data cleaning and validation	> Remembering GREL syntax
> Formula logic and spreadsheet work	> Writing HTML and CSS
> Judging whether results make sense	> Memorising git commands
> Understanding what the data represents	> Producing visual polish
> Deciding how to categorise values	> Recalling function names

Best AI use for formulas: ask **"what function would I use for this?"** not "do it for me." You apply the function, verify the result, and build the skill.

FAQ

> Is it ethical to merge Scotland into UK?

Cleaning is not neutral. Decisions about categories are decisions about how to represent the world:

- **Scotland/UK:** do you list Scotland as a separate country or merge it into the UK? Depends on whether your analysis is about political entities or geographic regions.
- **USSR/Russia:** historical data may list the USSR. Do you reclassify as Russia? You lose the Baltic states, Ukraine, and 12 other countries.
- **Palestine/Gaza:** naming and boundary decisions carry political weight.

There is no correct answer. The correct practice is to **document your reasoning** so that anyone reviewing your data understands the choice you made and why.

> What are the most common spreadsheet mistakes?

SUM vs COUNTA in pivot tables: SUM adds up numbers. COUNTA counts non-empty cells. If you want "how many flood events per country," use COUNTA. SUM would add up the Dead column instead of counting rows.

Semicolons vs commas as formula delimiters: Google Sheets uses your locale settings. In most European locales, formulas use semicolons: `=IF(A2>0; "Yes"; "No")`. In English locales, commas: `=IF(A2>0, "Yes", "No")`. If a formula doesn't work, check your delimiter.

Google Sheets limits: maximum ~10 million cells. Performance degrades noticeably above ~100,000 rows. For larger datasets, use OpenRefine or scripts.

> When should I clean: during Get or before Analyse?

Clean after Get, before Analyse. But the pipeline is iterative:

- **Verify** may reveal the data is unsuitable → go back to **Find** for a better source
- **Clean** may reveal structural problems → reconsider your **Define** (maybe your question can't be answered with this data)
- **Analyse** may reveal patterns that look wrong → go back to **Clean** to investigate

The pipeline is a living process. Returning to an earlier step is a feature, not a failure. Document each iteration so you can retrace your path.

> Does zero mean zero or unknown?

In the FloodArchive:

- **Dead = 0** could mean zero deaths, or unknown (no one counted)
- **GlideNumber = 0** is a missing ID, not a real identifier
- **Displaced = 0** might mean no displacement, or the source didn't report it

You must decide how to handle each case and **document the decision**. If you treat 0-as-unknown the same as 0-as-zero, your averages will be wrong (dragged down by false zeros). If you exclude all zeros, you lose genuine zero-death events.

There is no universal rule. The right choice depends on what you use the data for.