Guide to 2023 ACLED Column Changes

On 20 March 2023, ACLED made changes to the structure of the dataset by removing three existing columns and adding three new columns. These changes aimed to eliminate redundancies and reduce potential sources of confusion while also introducing more useful variables for analysis. In addition, some existing columns were repositioned within the dataset to improve the readability of the event data.

Three columns – event_id_no_cnty, data_id, and iso3 – have been removed from the dataset. The decision to remove these columns came after careful consideration and internal reviews of their usefulness. Alongside these column removals, ACLED introduced three new variables to facilitate analysis and provide users with additional information about events. The new columns are disorder_type, civilian_targeting, and tags, all of which are described in detail below.

This guide outlines the aforementioned changes to the ACLED dataset and explains how users can adjust to them. Users should review the guide carefully to ensure the updates do not interrupt their workflow.

Column Removals

event_id_no_cnty

ACLED originally created the event_id_no_cnty column so users could sort event IDs within a single country. However, other columns (e.g. event_date or timestamp) can be used to sort events chronologically and do so more accurately. Because of its redundancy and potential inaccuracy when used for chronological sorting, the event_id_no_cnty column was removed from the ACLED dataset. If a user still requires this column, it can easily be reproduced using the event_id_cnty column (see below for details).

data_id

The data_id column contained auto-generated IDs representing each event's row within the
ACLED dataset. Importantly, the values in the data_id column were not static, but changed each time the data were updated. The dynamic nature of the data_id column created confusion, as some users would mistake the values for static, unique IDs. ACLED removed the column from the dataset to reduce this confusion and prevent the use of an incorrect ID variable in the future. To uniquely identify and track events, users should always use the event_id_cnty column, which contains unique IDs that remain static even as the dataset is updated.

**iso3**

The ACLED dataset contained several columns indicating the country in which an event occurred. Two columns in particular – iso and iso3 – allowed users to easily join ACLED data with external datasets. The iso and iso3 columns, respectively, provided users with a country’s unique numeric code and three-letter code from the International Organization for Standardization (ISO). These columns provided different, but entirely interchangeable, country identifiers. ACLED therefore removed the iso3 column from the dataset in order to eliminate this redundancy and create space for more useful columns.

**Column Additions**

**disorder_type**

The new disorder_type column provides users with a broader classification of event types. This new classification system will allow users to more easily identify and filter relevant event categories, particularly those that are often used in ACLED methodology documentation and analysis. Each event will be assigned a disorder type based on the event_type and sub_event_type columns:

<table>
<thead>
<tr>
<th>disorder_type</th>
<th>event_type/sub_event_type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Political violence</td>
<td>• Battles</td>
</tr>
<tr>
<td></td>
<td>• Explosions/remote violence</td>
</tr>
<tr>
<td></td>
<td>• Violence against civilians</td>
</tr>
<tr>
<td></td>
<td>• Mob violence</td>
</tr>
<tr>
<td></td>
<td>• Excessive force against protesters</td>
</tr>
<tr>
<td>Demonstrations</td>
<td>• Protests (all sub-event types, including excessive force against protesters)</td>
</tr>
<tr>
<td></td>
<td>• Violent demonstration</td>
</tr>
<tr>
<td>Strategic developments</td>
<td>• Strategic developments</td>
</tr>
</tbody>
</table>
Note that the `disorder_type` categories are not mutually exclusive, as the `excessive force against protesters` sub-event type (a subset of the `protests` event type) is classified under both `political violence` and `demonstrations`.

### civilian_targeting

The new `civilian_targeting` column denotes that violence in an event mainly or solely targeted civilians. Without this column, users can only identify civilian targeting events by applying a combination of filters across `event_type`, `sub_event_type`, and various actor columns. The `civilian_targeting` column eliminates the need for such complex filtering, as it will contain one of two values: “Civilian targeting,” which indicates that civilians were targeted during the event, or blank (null), which indicates that ACLED found no reports that civilians were the main or sole target in the event. The lack of a civilian targeting designation does not rule out the possibility that civilians were affected by violence in the event, however (e.g. as ‘collateral damage’ in the context of a `battle or explosions/remote violence event`).

### tags

ACLED uses a variety of tags to provide standardized information about events. For example, tags may denote the size of a demonstration, whether women were specifically targeted in a violent incident, or connections to a particular political movement (e.g. “stop the steal” in the United States). Tags were previously included in the `notes` column within square brackets, which could make it difficult to filter events by tag or to extract tag information (e.g. size of a demonstration). The addition of a standalone `tags` column facilitates the extraction and analysis of tagged events. All tags that were in the `notes` column were shifted to the new `tags` column.

### Preparing for Column Updates

Anyone using ACLED data downloaded prior to 20 March 2023 will be using an outdated column structure. Those users will need to take certain steps to ensure that the column changes do not interrupt their workflow, regardless of whether they actively use the specific columns that were removed. Even users who do not currently use the removed columns may be affected by changes in column positions. **All users should follow the steps below to review and update any scripts and/or Excel files used to interact with ACLED data.**

The following examples focus specifically on Excel, R, and Python, but the underlying logic can easily be applied to other platforms and programming languages.
Step 1: Ensure that any references to the column index number (i.e. the column’s position number) are updated.

The position of nearly all ACLED columns changed as a result of the column updates. Therefore, references to column numbers or letters must be updated in all scripts and Excel formulas. These are the new column positions:

<table>
<thead>
<tr>
<th>Column Name</th>
<th>Column Letter (Excel)</th>
<th>Column Number</th>
<th>Positional Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>event_id_cnty</td>
<td>A</td>
<td>1</td>
<td>2 to the left</td>
</tr>
<tr>
<td>event_date</td>
<td>B</td>
<td>2</td>
<td>3 to the left</td>
</tr>
<tr>
<td>year</td>
<td>C</td>
<td>3</td>
<td>3 to the left</td>
</tr>
<tr>
<td>time_precision</td>
<td>D</td>
<td>4</td>
<td>3 to the left</td>
</tr>
<tr>
<td>disorder_type</td>
<td>E</td>
<td>5</td>
<td>New column</td>
</tr>
<tr>
<td>event_type</td>
<td>F</td>
<td>6</td>
<td>2 to the left</td>
</tr>
<tr>
<td>sub_event_type</td>
<td>G</td>
<td>7</td>
<td>2 to the left</td>
</tr>
<tr>
<td>actor1</td>
<td>H</td>
<td>8</td>
<td>2 to the left</td>
</tr>
<tr>
<td>assoc_actor_1</td>
<td>I</td>
<td>9</td>
<td>2 to the left</td>
</tr>
<tr>
<td>inter1</td>
<td>J</td>
<td>10</td>
<td>2 to the left</td>
</tr>
<tr>
<td>actor2</td>
<td>K</td>
<td>11</td>
<td>2 to the left</td>
</tr>
<tr>
<td>assoc_actor_2</td>
<td>L</td>
<td>12</td>
<td>2 to the left</td>
</tr>
<tr>
<td>inter2</td>
<td>M</td>
<td>13</td>
<td>2 to the left</td>
</tr>
<tr>
<td>interaction</td>
<td>N</td>
<td>14</td>
<td>2 to the left</td>
</tr>
<tr>
<td>civilian_targeting</td>
<td>O</td>
<td>15</td>
<td>New column</td>
</tr>
<tr>
<td>iso</td>
<td>P</td>
<td>16</td>
<td>14 to the right</td>
</tr>
<tr>
<td>region</td>
<td>Q</td>
<td>17</td>
<td>Same position</td>
</tr>
<tr>
<td>country</td>
<td>R</td>
<td>18</td>
<td>Same position</td>
</tr>
<tr>
<td>admin1</td>
<td>S</td>
<td>19</td>
<td>Same position</td>
</tr>
<tr>
<td>admin2</td>
<td>T</td>
<td>20</td>
<td>Same position</td>
</tr>
<tr>
<td>admin3</td>
<td>U</td>
<td>21</td>
<td>Same position</td>
</tr>
<tr>
<td>location</td>
<td>V</td>
<td>22</td>
<td>Same position</td>
</tr>
</tbody>
</table>
Examples:

a. Excel

Before the column change, this VLOOKUP formula would have accessed the `event_date` column by referencing column index 3 (i.e. the third column in the selected data range).

```
=VLOOKUP("TTQ971",C2:E11,3,FALSE)
```

Result

```
2018-12-29
```
After the column change, the same VLOOKUP formula now references column index 2 because `event_id_no_cnty` is no longer present, shifting `event_date` to the second column in the selected range.

### Formula

```
-VLOOKUP("TTQ971",B2:F11,2,FALSE)
```

### Result

```
2018-12-29
```

#### b. R

<table>
<thead>
<tr>
<th>Before column change</th>
<th>After column change</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>event_notes &lt;- acled_df %&gt;% select(3, 28)</code></td>
<td><code>event_notes &lt;- acled_df %&gt;% select(1, 28)</code></td>
</tr>
</tbody>
</table>

#### c. Python

<table>
<thead>
<tr>
<th>Before column change</th>
<th>After column change</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>event_notes = acled_df.iloc[0:9,[2,27]]</code></td>
<td><code>event_notes = acled_df.iloc[0:9,[0,27]]</code></td>
</tr>
</tbody>
</table>
Step 2: Ensure that any references to column letters are updated. (Note: This step is specific to Excel.)

As indicated in the previous step, column positions changed, which therefore impacted column letters in Excel (as outlined in the chart in step 1). Users should adjust to these changes by updating any references to column letters in Excel.

Examples:

a. Excel

Before the column change, this COUNTIF formula accessed the `sub_event_type` column (column letter I) by referencing cells I2:I11.

![Excel Chart Before Change](chart_before_change.png)

Formula

`=COUNTIF(I2:I11, "Attack")`

Result

6

After the column change, the same COUNTIF formula accesses the `sub_event_type` column (column letter G) by referencing cells G2:G11.

![Excel Chart After Change](chart_after_change.png)
Step 3: Eliminate all named references to the removed columns.

References to the data_id, event_id_no_cnty, or iso3 column names will produce errors and/or cause formulas and scripts to fail. Users should either delete all references to the removed columns or manually recreate the desired columns (see steps 4-6 for more details on replacing and/or reproducing the removed variables).

Examples:

a. Excel

Before the column change, this FILTER formula directly referenced event_id_no_cnty as the first column in the selected range of columns. (Note: The filter formula does not automatically provide column headers as shown in the result; these were added manually.)
However, after the column changes, this formula returns an error (#REF) because `event_id_no_cnty` no longer exists. To avoid this error, `event_id_no_cnty` is replaced with `event_id_cnty` in the formula.

Before column change

After column change

**b. R**

<table>
<thead>
<tr>
<th>Before column change</th>
<th>After column change</th>
</tr>
</thead>
<tbody>
<tr>
<td># Select a subset of columns</td>
<td># Select a subset of columns</td>
</tr>
<tr>
<td>new_df &lt;- acled_df %&gt;% select(data_id, event_id_cnty, event_date, country, latitude, longitude, notes, iso3)</td>
<td>new_df &lt;- acled_df %&gt;% select(event_id_cnty, event_date, country, latitude, longitude, notes)</td>
</tr>
</tbody>
</table>

**c. Python**

<table>
<thead>
<tr>
<th>Before column change</th>
<th>After column change</th>
</tr>
</thead>
<tbody>
<tr>
<td>#Subsetting some columns</td>
<td>#Subsetting some columns</td>
</tr>
<tr>
<td>new_df=acled_df[&quot;data_id&quot;, &quot;event_id_city&quot;, &quot;event_date&quot;, &quot;country&quot;, &quot;latitude&quot;, &quot;longitude&quot;, &quot;notes&quot;, &quot;iso3&quot;]</td>
<td>new_df=acled_df[&quot;event_id_city&quot;, &quot;event_date&quot;, &quot;country&quot;, &quot;latitude&quot;, &quot;longitude&quot;, &quot;notes&quot;]</td>
</tr>
</tbody>
</table>
Step 4: Begin using the `event_id_cnty` column in place of `event_id_no_cnty`, or regenerate the `event_id_no_cnty` column.

References to `event_id_no_cnty` should be replaced by `event_id_cnty`, which is simply `event_id_no_cnty` with a country abbreviation prefix added. Note that `event_id_no_cnty` is not a unique identifier of an event. Nonetheless, if users still require this column, it can be regenerated by removing the country abbreviation from values in the `event_id_cnty` column.

Examples:

a. Excel

Option 1 - Replace all references to `event_id_no_cnty` with `event_id_cnty`

Before the column change, the COUNTA and UNIQUE formulas shown here referenced the `event_id_no_cnty` column to count the number of unique events in the data. (Note that this approach was prone to errors and, as in this case, did not produce the desired result because `event_id_no_cnty` was not a unique event identifier. This example highlights one of several justifications for removing this column from the ACLED dataset.)
However, after the column change, the same formula returns an error, as *event_id_no_cnty* does not exist. To address the error and correctly count unique events, the reference to *event_id_no_cnty* is replaced by *event_id_cnty*.

**Option 2 - Regenerate *event_id_no_cnty***

1. Create a new column named “*event_id_no_cnty*.”
2. Insert the following formula in the first row of the new column:

\[ =\text{RIGHT}((\text{event\_id\_cnty\ cell\ reference}), \text{LEN}((\text{event\_id\_cnty\ cell\ reference}))-3) \]

3. Fill the column by clicking the autofill icon (the small square in the bottom left corner of the cell) and dragging it down.
Option 1: Replace all the references to `event_id_no_cnty` with `event_id_cnty`

```
# Sort by event_id_no_cnty
acled_df <- acled_df %>%
  arrange(event_id_no_cnty)
```

```
# Sort by event_id_cnty
acled_df <- acled_df %>%
  arrange(event_id_cnty)
```

Option 2: Regenerate `event_id_no_cnty`

```
# Sort by event_id_no_cnty
acled_df <- acled_df %>%
  arrange(event_id_no_cnty)
```

```
# Regenerate event_id_no_cnty and sort using str_extract
acled_df <- acled_df %>%
  mutate(event_id_no_cnty =
        str_extract(event_id_cnty, '\d+'))
  %>%
  arrange(event_id_no_cnty)
```

```
# Sort by event_id_no_cnty
acled_df <- acled_df %>%
  arrange(event_id_no_cnty)
```

```
# Regenerate event_id_no_cnty and sort using gsub
acled_df <- acled_df %>%
  mutate(event_id_no_cnty =
        gsub("\\D", "", event_id_cnty))
  %>%
  arrange(event_id_no_cnty)
```
b. Python

<table>
<thead>
<tr>
<th>Before column change</th>
<th>After column change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Option 1: Replace all references to <code>event_id_no_cnty</code> with <code>event_id_cnty</code></td>
<td></td>
</tr>
</tbody>
</table>
| # Sort by `event_id_no_cnty`  
aclad_df = aclad_df.sort_values("event_id_no_cnty") | # Sort by `event_id_cnty`  
aclad_df = aclad_df.sort_values("event_id_cnty") |
| Option 2: Regenerate `event_id_no_cnty` |  |
| # Sort by `event_id_no_cnty`  
aclad_df = aclad_df.sort_values("event_id_no_cnty") | # Regenerate `event_id_no_cnty`  
aclad_df["event_id_no_cnty"] = aclad_df["event_id_cnty"].str.replace(r’\D+’, ‘’, regex=True)  
# Sort by the regenerated `event_id_no_cnty`  
aclad_df = aclad_df.sort_values("event_id_no_cnty") |

Step 5: Begin using the `event_id_cnty` column in place of the `data_id` column.

As noted previously, the `data_id` column did not provide any analytical information for users and was not a static ID, meaning that it could not be used to uniquely and consistently identify a particular event. The `event_id_cnty` column is the only column that serves as a static, unique event identifier. Any references to the `data_id` column intended to be used as a unique event ID should immediately be replaced with `event_id_cnty`.

Examples:

a. Excel

Before the column change, the COUNTA and UNIQUE formulas shown here referenced the `data_id` column to count the number of unique events in the data. This approach was incorrect, as `data_id` was not a static identifier. If users combined ACLED events retrieved at different times into a single dataset, this may have introduced duplicated `data_id`s, resulting in an incorrect count of the number of unique events. This error is highlighted in the following screenshot, in which the calculation does not produce the correct count of unique events.
After the column change, the absence of the `data_id` column causes the formula to produce an error (#REF). To avoid the error, the reference to `data_id` in the formula is replaced with `event_id_cnty`. In addition, because the `event_id_cnty` is a unique event identifier, the formula returns the correct result.
Step 6: Begin using the iso column instead of the iso3 column.

Users who used the iso3 column, including users who relied on this column to merge ACLED data with external datasets, should replace all references to iso3 in scripts or spreadsheets with references to the iso column. Users should review the examples in step 5 for guidance on how to update column references.