# Data Format for Wind Energy Applications

## Organizing data

Describe best practices for organizing data so computers can make the best use of data sets.

Spreadsheets are used for data entry, but in reality we tend to use spreadsheet programs for much more than data entry. We use them to create data tables for publications, to generate summary statistics, and make figures. Using spreadsheets to generate tables for publications is not ideal due to the need for special formatting, which can lead to unpredictable results when transferring to document software. Additionally, spreadsheets' graphical nature makes it difficult to replicate complex calculations, increasing the risk of inconsistencies. Command-line based programs like Python or R offer more reliability by ensuring uniform application of calculations across the dataset.

When working with data in programming languages like Python, proper formatting is crucial to avoid issues with data manipulation and analysis. Formatting your data in a consistent and clean manner ensures that your code runs smoothly without errors related to data types or mismatched formats. In this section, we will explore how to format your data in text format for seamless use in Python, including tips on data cleaning and preparation.

Ensuring your data is properly formatted before importing it into Python can save a significant amount of time and prevent potential errors. We will guide you through the process of preparing your data in a text-based format that is ready for analysis, highlighting techniques to check data quality and generate preliminary summary statistics.

## Formatting Data for Programming Languages

When working with data in programming languages like Python, proper formatting is crucial to avoid issues with data manipulation and analysis. Ensuring your data is consistently formatted before importing it into Python can save a significant amount of time and prevent potential errors. This guide will walk you through best practices for data entry and formatting to ensure your data is ready for seamless use in Python.

### Importance of Proper Data Formatting

The most common mistake is treating data files like lab notebooks, relying on context, notes, and spatial layout to convey information. While humans can interpret these cues, computers require explicitly defined structures. Properly formatted data allows you to harness the power of programming languages for effective and efficient data analysis.

### Best Practices for Data Formatting

#### Save each data file separately

Ensure each text file contains only one table. Each row should represent a distinct observation, and each column should represent a variable.

If the data saved in speadsheet format has many tabs, combine data from all tabs into a single text file, adding additional columns to differentiate data where necessary. Or separate each tab and saved in a single text file.

#### Including Metadata in Data Table

Adding metadata directly into the data table disrupts data parsing. Store metadata in a separate file.

ADD raw data from daragh here

#### Not Filling in Zeros

Leaving cells blank instead of filling in zeros can lead to misinterpretation of data as missing rather than zero. Therefore you should xxplicitly enter zeros where applicable to indicate a measured value of zero.

Date\_and\_time,Wind\_speed,Wind\_speed\_Standard\_deviation

2022-01-01 00:00:00,6.781222248077393,0

2022-01-01 00:10:00,6.936052060127258,1.2872216897689304

#### Using Problematic Null Values

Using arbitrary values like -999 or 999 to indicate missing data can cause issues in analysis. Use a consistent and appropriate null indicator like NA or leave cells blank. Common placeholders include empty fields or specific markers like ‘NA’ or ‘null’:

Date\_and\_time,Wind\_speed,Wind\_speed\_Standard\_deviation

2022-01-01 00:00:00,6.781222248077393,1.182439280264067

2022-01-01 00:10:00, ,1.2872216897689304

#### Consistent Delimiters:

Ensure your text data uses consistent delimiters (e.g., commas, tabs) to separate fields. CSV (Comma-Separated Values) is a widely accepted format:

Date\_and\_time,Wind\_speed,Wind\_speed\_Standard\_deviation

2022-01-01 00:00:00,6.781222248077393,1.182439280264067

2022-01-01 00:10:00,6.936052060127258,1.2872216897689304

#### Uniform Data Types:

Ensure each column contains data of the same type. For example, an Wind\_speed column should only have numeric values:

Date\_and\_time,Wind\_speed,Wind\_speed\_Standard\_deviation

2022-01-01 00:00:00,6.781222248077393,1.182439280264067

2022-01-01 00:10:00,6.936052060127258,1.2872216897689304

#### Avoid Special Characters:

Using spaces, special characters, or starting field names with numbers. Remove or escape special characters that might interfere with data parsing. Use descriptive, alphanumeric field names with underscores or camel case within text fields:

# Date and time,Wind speed (m/s),"Wind speed, Standard deviation (m/s)"

2022-01-01 00:00:00,6.781222248077393,1.182439280264067

2022-01-01 00:10:00,6.936052060127258,1.2872216897689304

#### Standardize Date Formats:

Avoid Single Column Date Formats: Storing dates in a single column may seem natural, but it can introduce ambiguities and compatibility issues. Different programs might interpret the same date differently, leading to errors.

Separate Year, Month, and Day: Storing dates as separate fields eliminates ambiguity and makes it easier to manipulate dates programmatically.

Year,Month,Day

2015,03,24

Use Unambiguous Date Formats: If storing dates as strings, use a consistent and sortable format like YYYYMMDD for dates and YYYYMMDDhhmmss for datetimes.

Date

20150324

Handle Missing Dates Properly: Ensure missing dates are clearly indicated to avoid misinterpretation by programs. Use NA or leave cells blank.

Year,Month,Day

2015,03,24

 ,,

#### Key Points

#### Treat dates as multiple pieces of data (year, month, day) rather than a single entry.

#### Use consistent, unambiguous formats like YYYYMMDD for dates.

#### Clearly indicate missing dates to avoid misinterpretation.

#### When handling dates in programming, use built-in functions to extract and manipulate date components.

#### By following these best practices, you can ensure that your dates are handled correctly and consistently in programming environments, minimizing errors and enhancing the reliability of your data analysis.

#### Maintain Raw Data Integrity:

Never modify your raw data directly. Always create a copy for any cleaning or analysis to ensure reproducibility.

#### Keep Track of Data Cleaning Steps:

Document all data cleaning steps in a plain text file stored in the same folder as the data file. This practice ensures that your analysis is reproducible and transparent.

#### Using Formatting to Convey Information

Using cell formatting (e.g., color, bold text) to convey information that isn't captured in the data values themselves. Use additional columns to encode any extra information.

Date\_and\_time,Wind\_speed,Wind\_speed\_Standard\_deviation, Status

2022-01-01 00:00:00,6.781222248077393,1.182439280264067, checked

2022-01-01 00:10:00,6.936052060127258,1.2872216897689304, unchecked

#### Placing Comments or Units in Cells

Including comments or units directly in data cells. Use separate columns for comments and ensure all measurements in a column are in the same unit, specifying units in the column header, or you can save the unit of each column in a separate file, keeping track of data cleaning steps.

Date\_and\_time,Wind\_speed,Wind\_speed\_Standard\_deviation, Failure

2022-01-01 00:00:00,6.781222248077393,1.182439280264067,

2022-01-01 00:10:00,6.936052060127258,1.2872216897689304, “Gearbox error”

#### Entering Multiple Pieces of Information in One Cell

Combining multiple pieces of information within a single cell. Separate data into distinct columns.

Date\_and\_time,Wind\_speed,Wind\_speed\_Standard\_deviation, Failure, Component

2022-01-01 00:00:00,6.781222248077393,1.182439280264067, ,

2022-01-01 00:10:00,6.936052060127258,1.2872216897689304, Yes, “Gearbox”

#### Structuring Data for Programming

The cardinal rule for data structure is to keep it “tidy”:

Columns = Variables: Each column should represent a variable (e.g., Wind\_speed,Power).

Rows = Observations: Each row should represent a single observation or record.

Cells = Values: Each cell should contain a single data value.

Example of well-structured data:

Date\_and\_time,Wind\_speed,Wind\_speed\_Standard\_deviation

2022-01-01 00:00:00,6.781222248077393,1.182439280264067

2022-01-01 00:10:00,6.936052060127258,1.2872216897689304

#### Key Points

By following these practices, you ensure that your data is ready for analysis in programming languages like Python, avoiding common pitfalls related to data formatting and quality.

* Never modify your raw data directly. Always work on a copy.
* Use consistent formatting and delimiters.
* Clearly indicate missing values.
* Maintain uniform data types within columns.
* Document all data cleaning steps for reproducibility.

### Exporting Data

When preparing data for analysis in programming environments such as Python, exporting it from spreadsheets in a way that is both universal and reliable is crucial. This guide will help you understand why using common spreadsheet formats is problematic and how to properly export your data to formats like CSV or TSV to ensure compatibility and accuracy.

Storing your data in proprietary formats such as Excel's default (\*.xls or \*.xlsx) can lead to several issues: Proprietary nature: Future technology might not support these formats. Compatibility: Other software might not open these files correctly. Version differences: Different Excel versions might handle data differently. epository requirements: Most data repositories do not accept proprietary formats.

Recommended Formats: CSV and TSV. To avoid these issues, use universal, open, and static formats like CSV (Comma-Separated Values) or TSV (Tab-Separated Values). These formats are plain text files that can be opened and read by almost any software, ensuring maximum portability and longevity.

Exporting Data to CSV from Excel: To save a file in CSV format from Excel:

* Open your file in Excel.
* Select "File" > "Save As".
* Choose "Comma Separated Values" (\*.csv) from the format list.
* Check the file name and save location, then click "Save".

You can open CSV files in Excel, but ensure compatibility with other programs by verifying the format after saving.

Example of problematic CSV data:

Date\_and\_time,Wind\_speed,Wind\_speed\_Standard\_deviation

2022-01-01 00:00:00,6.781222248077393,1.182439280264067

2022-01-01 00:10:00,6.936052060127258,1.2872216897689304

TSV as a solution:

Date\_and\_time Wind\_speed Wind\_speed\_Standard\_deviation

2022-01-01 00:00:00 6.781222248077393 1.182439280264067

2022-01-01 00:10:00 6.936052060127258 1.2872216897689304

By following these practices, you ensure your data is in a robust format for analysis in programming environments, minimizing errors and maximizing compatibility.