

Research Software Engineering: A Primer

Dr. Franziska Horn

Research Software Engineering: A Primer

Dr. Franziska Horn

2024-11-25

Table of contents

Preface	1
1 Research Purpose	3
1.1 Types of Analyses	3
1.2 Draw your Why	6
1.3 Evaluation Metrics	7
2 Data & Results	9
2.1 Data Types	9
2.2 Data Analysis Results	11
2.3 Draw your What	16
3 Tools	19
3.1 Programming languages	19
4 Software Design	21
5 Implementation	23
6 From Research to Production	25
References	27

Preface

This book is meant to **empower researchers to code with confidence and clarity**.

If you studied something other than computer science—especially in the natural sciences like physics, chemistry, or biology—it’s likely you were never taught how to properly develop software. Yet, you’re often still expected to write code as part of your daily work. Maybe you’ve taken a programming course like *Python for Biologists* and can put together functional scripts through trial and error (with a little help from ChatGPT). But chances are, no one ever showed you how to write well-structured, maintainable, and reusable code that could make your life—and collaborating with your colleagues—so much easier.

This book is for you if you want to:

- Write functional software more quickly
- Use a structured approach to design better programs
- Reuse your code in future projects
- Feel confident about what your scripts are doing
- Prepare your research code for production
- Share your work with pride.

Whether you’re just beginning your scientific journey—perhaps working on your first major project like a master’s thesis or your first paper—or you’re contemplating a move from academia to industry, the practical advice in this book can guide you along the way.

While the book contains some example code in Python to illustrate the concepts, the general ideas are independent of any programming language.

This is still a draft version! Please write me an email, if you have any suggestions for how this book could be improved!

Enjoy!

Acknowledgments

The texts in this book were edited and refined with the help of ChatGPT, however, all original content is my own.

How to cite

```
@book{horn2025rseprimer,  
  author = {Horn, Franziska},  
  title = {Research Software Engineering: A Primer},  
  year = {2025},  
  url = {https://franziskahorn.de/rsebook/},  
}
```

1 Research Purpose

Before writing your first line of code, it's crucial to have a clear understanding of what you're trying to achieve—specifically, the purpose of your research. This clarity will not only help you reach your desired outcomes more efficiently but will also be invaluable when collaborating with others. Being able to explain your goals effectively ensures everyone is aligned and working toward the same objective.

We'll begin with an overview of common research goals and the types of data analysis needed to achieve them. Then, we'll explore how to visually communicate your research purpose, as visual representations are often the most effective way to convey complex ideas. Finally, we'll discuss how to quantify the outcomes you're trying to achieve.

1.1 Types of Analyses

Most research questions can be categorized into four broad groups, each associated with a specific type of analytics approach (Figure 1.1).

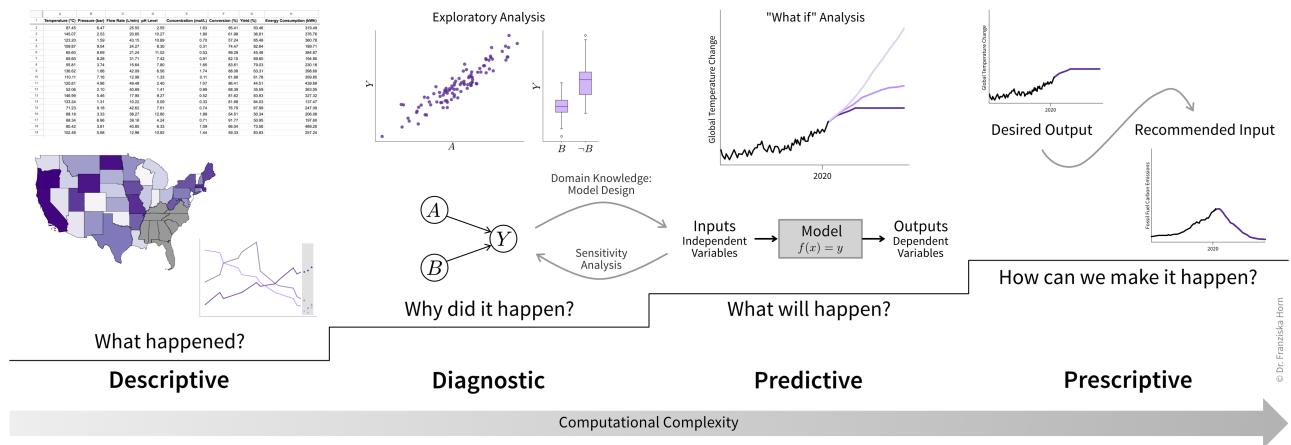


Figure 1.1: Descriptive, diagnostic, predictive, and prescriptive analytics, with increasing computational complexity and need to write custom code.

Descriptive Analytics

This approach focuses on observing and describing phenomena, often for the first time. Examples include:

- Identifying animal and plant species in unexplored regions of the deep ocean.
- Measuring the physical properties of a newly discovered material.

1 Research Purpose

- Surveying the political views of the next generation of teenagers.

Methodology:

- Collect a large amount of data (e.g., samples or observations).
- Calculate summary statistics like averages, ranges, or standard deviations, typically using standard software tools.

Diagnostic Analytics

Here, the goal is to understand relationships between variables and uncover causal chains to explain *why* phenomena occur. Examples include:

- Investigating how CO₂ emissions from burning fossil fuels drive global warming.
- Evaluating whether a new drug reduces symptoms and under what conditions it works best.
- Exploring how economic and social factors influence shifts toward right-wing political parties.

Methodology:

- Perform exploratory data analysis, such as looking for correlations between variables.
- Conduct statistical tests to support or refute hypotheses (e.g., comparing treatment and placebo groups).
- Design of experiments to control for external factors (e.g., randomized clinical trials).
- Build predictive models to simulate relationships. If these models match real-world observations, it suggests their assumptions correctly represent causal effects.

Predictive Analytics

This method involves building models to describe and predict relationships between independent variables (inputs) and dependent variables (outputs). These models often rely on insights from diagnostic analytics, such as which variables to include in the model and how they might interact (e.g., linear or nonlinear dependence). Examples include:

- Weather forecasting models.
- Digital twin of a wind turbine to simulate how much energy is generated under different conditions.
- Predicting protein folding based on amino acid sequences.

Methodology:

The key difference lies in how much domain knowledge informs the model:

- *White-box (mechanistic) models:* Based entirely on known principles, such as physical laws or experimental findings. These models are often manually designed, with parameters fitted to observed data.
- *Black-box (data-driven) models:* Derived purely from observational data. Researchers usually test different model types (e.g., neural networks or Gaussian processes) and choose the one with the highest accuracy.

- *Gray-box (hybrid) models:* These combine mechanistic and data-driven approaches. For example, the output of a mechanistic model may serve as an input to a data-driven model, or the data-driven model may predict residuals (i.e., prediction errors) from the mechanistic model, where both outputs combined yield the final prediction.

Resources to learn more about data-driven models

If you want to learn more about how to create data-driven models and the machine learning (ML) algorithms behind them, these two free online books are highly recommended:

- [1] [Supervised Machine Learning for Science](#) by Christoph Molnar & Timo Freiesleben; A fantastic introduction focused on applying black-box models in scientific research.
- [2] [A Practitioner's Guide to Machine Learning](#) by me; A broader overview of ML methods for a variety of use cases.

After developing an accurate model, researchers can analyze its behavior (e.g., through a sensitivity analysis, which examines how outputs change with varying inputs) to gain further insights about the system (to feed back into diagnostic analytics).

Prescriptive Analytics

This approach focuses on decision-making and optimization, often using predictive models. Examples include:

- Screening thousands of drug candidates to find those most likely to bind with a target protein.
- Optimizing reactor conditions to maximize yield while minimizing energy consumption.

Methodology:

- *Decision support:* Use models for “what-if” analyses to predict outcomes of different scenarios. For example, models can estimate the effects of limiting global warming to 2°C versus exceeding that threshold, thereby informing policy decisions.
- *Decision automation:* Use models in optimization loops to systematically test input conditions, evaluate outcomes (e.g., resulting predicted material quality), and identify the best conditions automatically.

Please note: These recommendations are only as good as the underlying models. Models must accurately capture causal relationships and often need to extrapolate beyond the data used to build them (e.g., for disaster simulations). Data-driven models are typically better at interpolation (predicting within known data ranges), so results should ideally be validated through additional experiments, such as testing the recommended new materials in the lab.

Together, these four types of analytics form a powerful toolkit for tackling real-world challenges: descriptive analytics provides a foundation of understanding, diagnostic analytics uncovers the causes behind observed phenomena, predictive analytics models future scenarios based on this understanding, and prescriptive analytics turns these insights into actionable solutions. Each step builds on the previous one, creating a systematic approach to answering complex questions and making informed decisions.

1.2 Draw your Why

In research, your goal is to improve the status quo, whether by filling a knowledge gap or developing a new method, material, or process with better properties. When sharing your idea—with collaborators, in a talk, or through a publication—a visual representation of what you’re trying to achieve can be incredibly useful.

One effective way to illustrate the improvement you’re working on is by creating “before and after” visuals, depicting the problem with the status quo and your solution (Figure 1.2).

The “before” scenario might show a lack of data, an incomplete understanding of a phenomenon, poor model performance, or an inefficient process or material. The “after” scenario highlights how your research addresses these issues and improves on the current state, such as refining a predictive model or enhancing the properties of a new material.

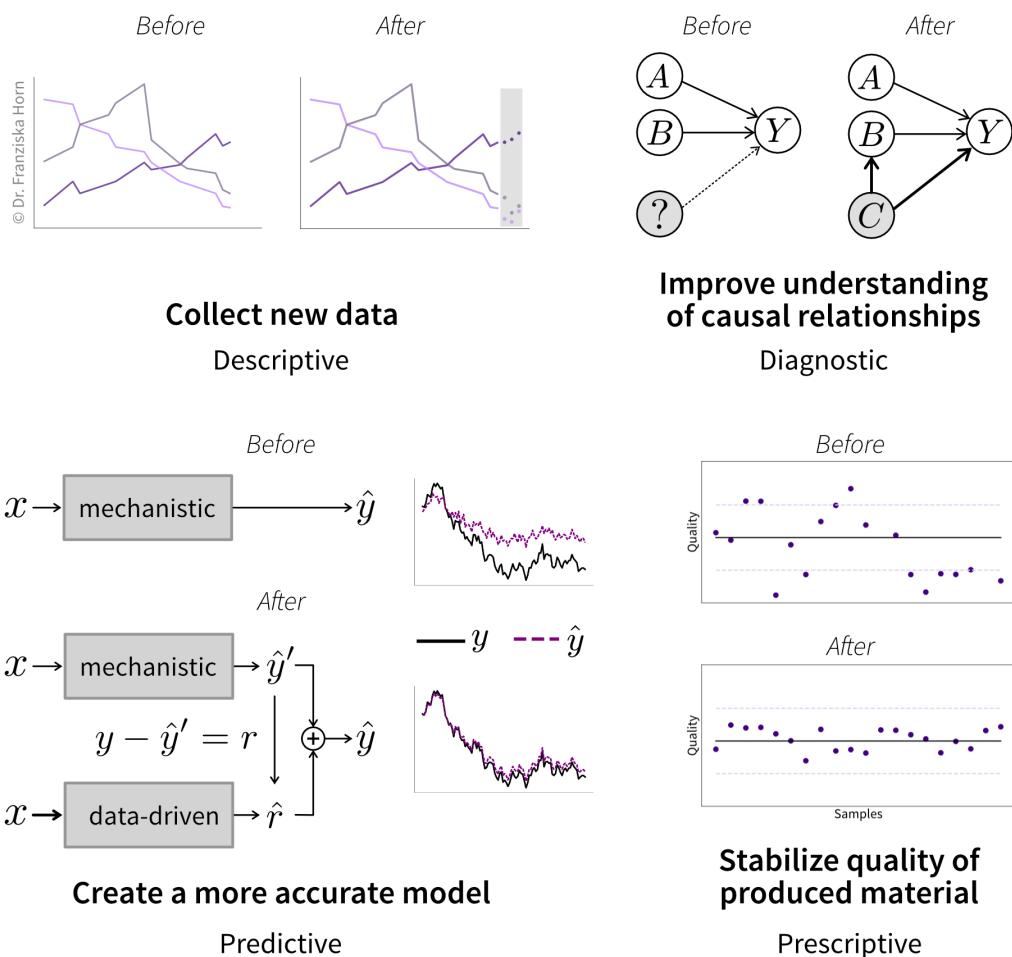


Figure 1.2: Exemplary research goals and corresponding “before and after” visuals for descriptive, diagnostic, predictive, and prescriptive analytics tasks.

At this point, your “after” scenario might be based on a hypothesis or an educated guess about what your results will look like—and that’s totally fine! The purpose of visualizing your goal is to guide your

development process. Later, you can update the picture with actual results if you decide to include it in a journal publication, for example.

Of course, not all research goals are tied directly to analytics. Sometimes the main improvement is more qualitative, for example, focusing on design or functionality (Figure 1.3). Even in these cases, however, you'll often need to demonstrate that your new approach meets or exceeds existing solutions in terms of other key performance indicators (KPIs), such as energy efficiency, speed, or quality parameters like strength or durability.

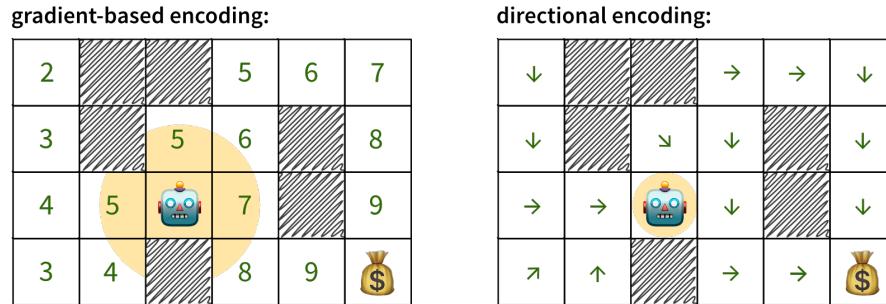


Figure 1.3: This example illustrates a task where a robot must reach its target (represented by money) as efficiently as possible. **Original approach (left):** The robot relied on information encoded in the environment as expected rewards. To determine the shortest path to the target, the robot required a large sensor (shown as a yellow circle) capable of scanning nearby fields to locate the highest reward. **New approach (right):** Instead of relying on reward values scattered across the environment, the optimal direction is now encoded directly in the current field. This eliminates the need for large sensors, as the robot only needs to read the value of its current position, enabling it to operate with a much smaller sensor and thereby reducing hardware costs. **Additional quantitative evaluation:** It still needs to be demonstrated that with the new approach, the robot reaches its target at least as quickly as with the original approach.

Give it a try—does the sketch help you explain your research to your family?

1.3 Evaluation Metrics

The “before and after” visuals help illustrate the improvement you’re aiming for in a qualitative way. However, to make a compelling case, it’s important to back up your findings with quantifiable results that show the extent of your improvement. Common evaluation metrics include:

- **Number of samples:** This refers to the amount of data you’ve collected, such as whether you surveyed 100 or 10,000 people. Larger sample sizes can provide more robust and reliable results.
- **Reliability of measurements:** This evaluates the consistency of your data. For example, how much variation occurs if you repeat the same measurement, e.g., run a simulation with different random seeds. Other factors, like sampling bias (i.e., when your sample is not representative for the whole population), can also affect the validity of your conclusions.
- **Statistical significance:** The outcome of a statistical hypothesis test, such as a p-value that indicates whether the difference in symptom reduction between the treatment and placebo groups is significant.

1 Research Purpose

- **Model accuracy:** This measures how well your model predicts or matches new observational data. Common metrics include R^2 , which indicates how closely the model's predictions align with actual outcomes.
- **Algorithm performance:** This includes metrics like memory usage and the time required to fit a model or make predictions, and how these values change as the dataset size increases. Efficient algorithms are crucial when scaling to large datasets or handling complex simulations.
- **Key Performance Indicators (KPIs):** These are specific metrics tied to the success of your optimized process or product. For example, KPIs might include yield, emissions, energy efficiency, or quality parameters like durability, strength, or purity of a chemical compound.
- **Convergence time:** This refers to how quickly a process (or simulation thereof) reaches optimal results and stabilizes without fluctuating. A shorter convergence time often suggests a more efficient and reliable process.

Your evaluation typically involves multiple metrics. For example, in prescriptive analytics, you need to demonstrate both the accuracy of your model and that the recommendations generated with it led to a genuinely optimized process or product.

Ideally, you should already have an idea of how existing solutions perform on these metrics (e.g., based on findings from other publications) to establish the baseline your solution should outperform (i.e., your “before”). You’ll likely need to replicate at least some of these baseline results (e.g., by reimplementing existing models) to ensure your comparisons are not influenced by external factors. But understanding where the “competition” stands can also help you identify secondary metrics where your solution could excel. For example, even if there’s little room to improve model accuracy, existing solutions might be too slow to handle large datasets efficiently.¹

These results are central to your research (and publications), and much of your code will be devoted to generating them, along with the models and simulations behind them. Clearly defining the key metrics needed to demonstrate your research’s impact will help you focus your programming efforts effectively.

Before you continue

At this point, you should have a clear understanding of:

- The problem you’re trying to solve.
- Existing solutions to this problem, i.e., the baseline you’re competing against.
- Which metrics should be used to quantify your improvement on the current state.

¹For example, currently, a lot of research aims to replace traditional mechanistic models with data-driven machine learning models, as these enable significantly faster simulations. A notable example is the AlphaFold model, which predicts protein folding from amino acid sequences—a breakthrough so impactful it was recognized with a Nobel Prize [3].

2 Data & Results

In the previous chapter, we've gained clarity on the problem you're trying to solve and how to quantify the improvements your research generates. Now it's time to dive deeper into what these results might actually look like and the data on which they are built.

2.1 Data Types

In one form or another, your research will rely on data, both collected or generated by yourself and possibly others.

Structured vs. Unstructured Data

Data can take many forms, but one key distinction is between structured and unstructured data (Figure 2.1).

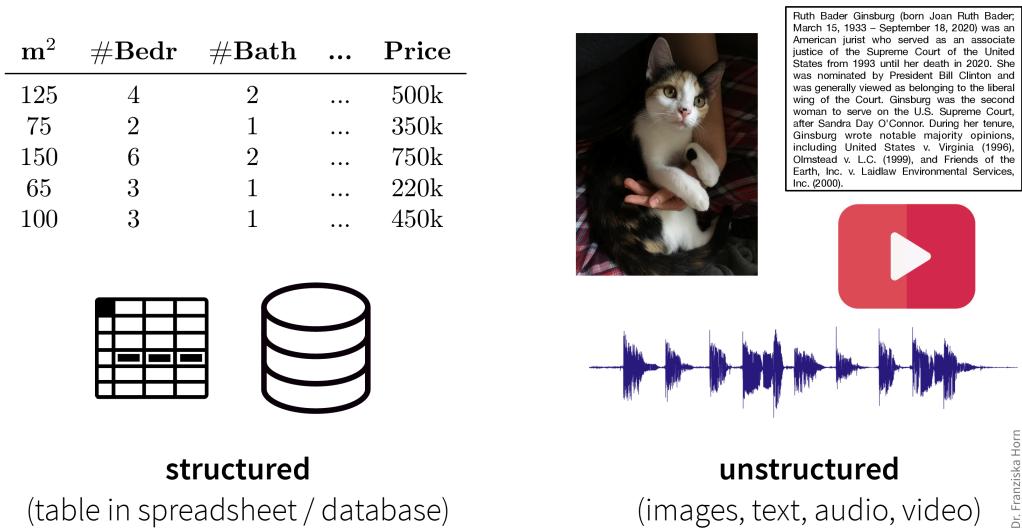


Figure 2.1: Structured and unstructured data.

Structured data is organized in rows and columns, like in Excel spreadsheets, CSV files, or relational databases. Each row represents a sample or observation (a data point), while each column corresponds to a variable or measurement (e.g., temperature, pressure, household income, number of children).

2 Data & Results

Unstructured data, in contrast, lacks a predefined structure. Examples include **images**, **text**, **audio recordings**, and **videos**, typically stored as separate files on a computer or in the cloud. While these files might include structured metadata (e.g., timestamps, camera settings), the data content itself can vary widely—for instance, audio recordings can range from seconds to hours in length.

Structured data is often *heterogeneous*, meaning it includes variables representing different kinds of information with distinct units or scales (e.g., temperature in °C and pressure in kPa). Unstructured data tends to be *homogeneous*; for example, there's no inherent difference between one pixel and the next in an image.

Note

Even though unstructured data is common in science (e.g., microscopy images), for simplicity, this book focuses on structured data. Furthermore, for now we'll assume that your data is stored in an Excel or CSV file, i.e., a spreadsheet with rows (samples) and columns (variables), on your computer. Later in Chapter 6, we'll discuss more advanced options for storing and accessing data, such as databases and APIs.

Programming Data Types

Each variable in your dataset (i.e., each column in your spreadsheet) is represented as a specific data type, such as:

- **Numbers** (integers for whole numbers or floats for decimals)
- **Strings** (text)
- **Boolean values** (true/false)

In programming, these are so-called *primitive data types* (as opposed to composite types, like arrays or dictionaries containing multiple values, or user-defined objects) and define how information is stored in computer memory.

Data types in Python

```
# integer
i = 42
# float
x = 4.1083
# string
s = "hello world!"
# boolean
b = False
```

Statistical Data Types

Even more important than how your data is stored, is understanding what your data *means*. Variables fall into two main categories:

1. **Continuous (numerical) variables** represent measurable values (e.g., temperature, height). These are usually stored as floats or integers.
2. **Discrete (categorical) variables** represent distinct options or groups (e.g., nationality, product type). These are often stored as strings, booleans, or sometimes integers.

Misleading data types

Be cautious: a variable that looks numerical (e.g., 1, 2, 3) may actually represent categories. For example, a `material_type` column with values 1, 2, and 3 might correspond to aluminum, copper, and steel, respectively. In this case, the numbers are IDs, not quantities.

Recognizing whether a variable is continuous or discrete is crucial for creating meaningful visualizations and using appropriate statistical models.

Time Series Data

Another consideration is whether your data points are linked by time. **Time series data** often refers to numerical data collected over time, like temperature readings or sales numbers. These datasets are usually expected to exhibit seasonal patterns or trends over time.

However, nearly all datasets involve some element of time. For example, if your dataset consists of photos, timestamps might seem unimportant, but they could reveal trends—like changes in image quality due to new equipment.

Important

Always include timestamps in your data or metadata to help identify potential correlations or unexpected trends over time.

Sometimes, you may be able to collect truly time-independent data (e.g., sending a survey to 1,000 people simultaneously and they all answer within the next 10 minutes). But usually, your data collection will take longer and external factors—like an election during a longer survey period—might unintentionally affect your results. By tracking time, you can assess and adjust for such influences.

2.2 Data Analysis Results

When analyzing data, the process is typically divided into two phases:

1. **Exploratory Analysis:** This involves generating a variety of plots to gain a deeper understanding of your data, such as identifying correlations between variables. It's often a quick and dirty process to help you familiarize yourself with the dataset.

2 Data & Results

2. **Explanatory Analysis:** This focuses on creating refined, polished plots intended for communicating your findings, such as in a publication or presentation. These visuals are designed to clearly convey your results to an audience that may not be familiar with your data.

Exploratory Analysis

In this initial analysis, the goal is to get acquainted with the data, check if the trends and relationships you anticipated are present, and uncover any unexpected patterns or insights.

- Examine the **raw data**:
 - Is the dataset complete, i.e., does it contain all the variables and samples you expected?
- Examine **summary statistics** (e.g., mean, standard deviation (std), min/max values, missing value count, etc.):
 - What does each variable mean? Given your understanding of the variable, are its values in a reasonable range?
 - Are missing values encoded as NaN (Not a Number) or as ‘unrealistic’ numeric values (e.g., -1 while normal values are between 0 and 100)?
 - Are missing values random or systematic (e.g., in a survey rich people are less likely to answer questions about their income or specific measurements are only collected under certain conditions)? This can influence how missing values should be handled, e.g., whether it makes sense to impute them with the mean or some other specific value (e.g., zero).
- Examine the **distributions of individual (continuous) variables**:

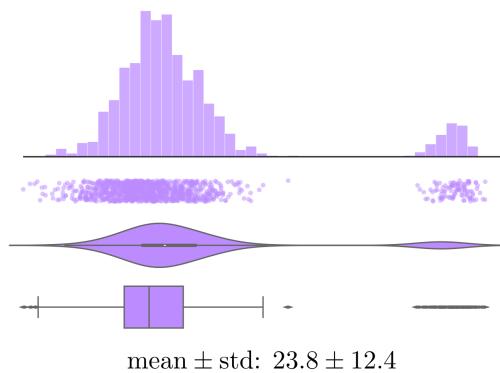


Figure 2.2: Histogram, strip plot, violin plot, box plot, and summary statistics of the same values.

- Are there any outliers? Are these genuine edge cases or can they be ignored (e.g., due to measurement errors or wrongly encoded data)?
- Is the data normally distributed or does the plot show multiple peaks? Is this expected?
- Examine **trends over time** (by plotting variables over time, even if you don’t think your data has a meaningful time component, e.g., by lining up representative images according to their timestamps to see if there is a pattern):

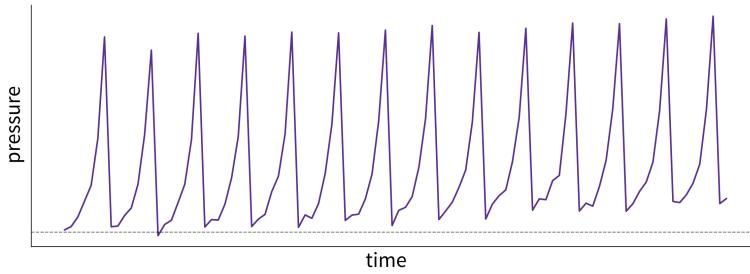


Figure 2.3: What caused these trends and what are their implications for the future? This plot shows fictitious data of the pressure in a pipe affected by fouling—that is, a buildup of unwanted material on the pipe’s surface, leading to increased pressure. The pipe is cleaned at regular intervals, causing a drop in pressure. However, because the cleaning process is imperfect, the baseline pressure gradually shifts upward over time.

- Are there time periods where the data was sampled irregularly or samples are missing? Why?
- Are there any (gradual or sudden) data drifts over time? Are these genuine changes (e.g., due to changes in the raw materials used in the process) or artifacts (e.g., due to a malfunctioning sensor recording wrong values)?

- Examine relationships between two variables:

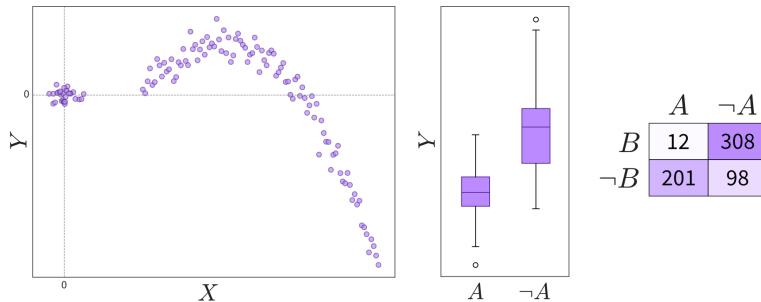


Figure 2.4: Depending on the variables’ types (continuous or discrete), relationships can be shown in scatter plots, box plots, or a table. Please note that not all interesting relations between the two variables can be detected through a high [correlation coefficient](#), so you should always check the scatter plot for details.

- Are the observed correlations between variables expected?
- Examine patterns in multidimensional data (using a parallel coordinate plot):

2 Data & Results

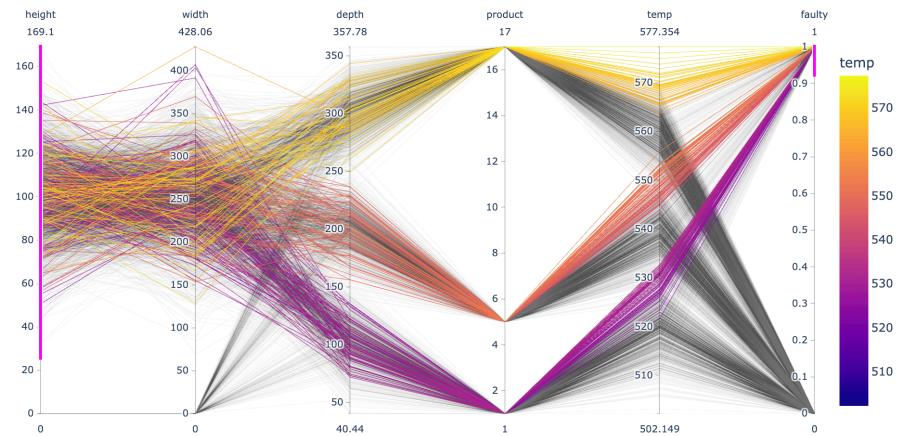


Figure 2.5: Each line in a parallel coordinate plot represents one data point, with the corresponding values for the different variables marked at the respective y-axis. The screenshot here shows an interactive plot created using the Python `plotly` library. By selecting value ranges for the different dimensions (indicated by the pink stripes), it is possible to spot interesting patterns resulting from a combination of values across multiple variables.

- Do the observed patterns in the data match your understanding of the problem and dataset?

Explanatory Analysis

Most of the plots you create during an exploratory analysis are likely for your eyes only. Any plots you do choose to share with a broader audience—such as in a paper or presentation—should be refined to clearly communicate your findings. Since your audience is much less familiar with the data and likely lacks the time or interest to explore it in depth, it’s essential to make your results more accessible. This process is often referred to as *exploratory analysis* [4].

⚠ Warning

Don’t “just show all the data” and hope that your audience will make something of it—understand what they need to answer the questions they have.

Step 1: Choose the right plot type

- Get inspired by visualization libraries (e.g., [here](#) or [here](#)), but avoid the urge to create fancy graphics; sticking with common visualizations makes it easier for the audience to correctly decode the presented information.
- Don’t use 3D effects!
- Avoid pie or donut charts (angles are hard to interpret).
- Use line plots for time series data.
- Use horizontal instead of vertical bar charts for audiences that read left to right.
- Start the y-axis at 0 for area & bar charts.
- Consider using [small multiples](#) or sparklines instead of cramming too much into a single chart.

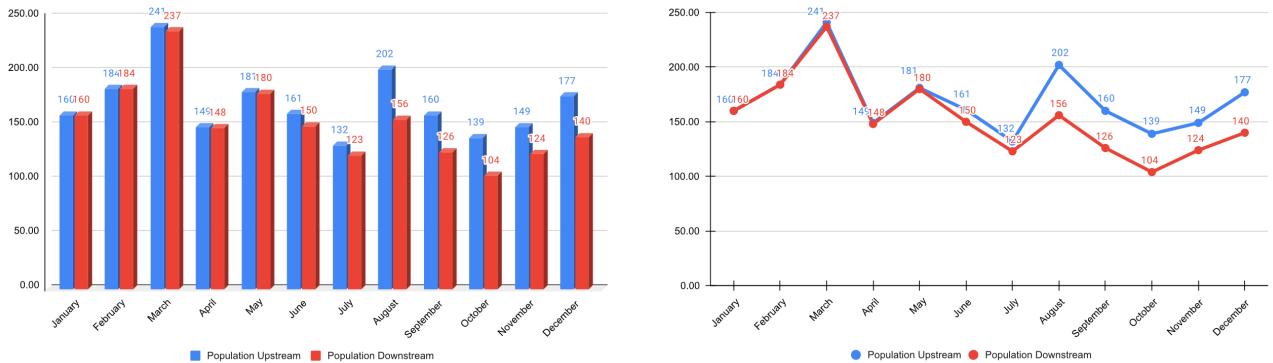


Figure 2.6: *Left:* Bar charts (especially in 3D) make it hard to compare numbers over a longer period of time. *Right:* Trends over time can be more easily detected in line charts. [Example adapted from: *Storytelling with Data* by Cole Nussbaum Knaflic]

Step 2: Cut clutter / maximize data-to-ink ratio

- Remove border.
- Remove gridlines.
- Remove data markers.
- Clean up axis labels.
- Label data directly.

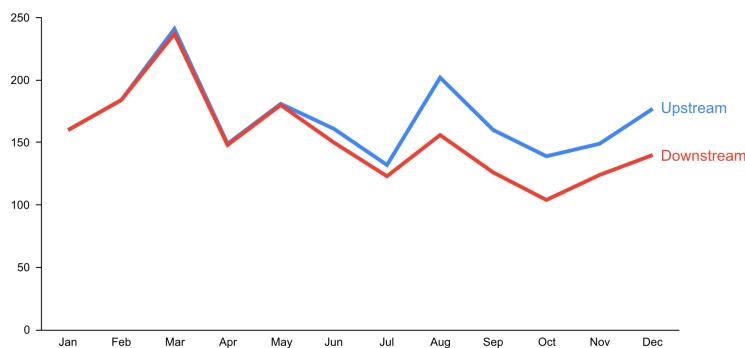


Figure 2.7: Cut clutter! [Example adapted from: *Storytelling with Data* by Cole Nussbaum Knaflic]

Step 3: Focus attention

- Start with gray, i.e., push everything in the background.
- Use pre-attentive attributes like color strategically to highlight what's most important.
- Use data labels sparingly.

2 Data & Results

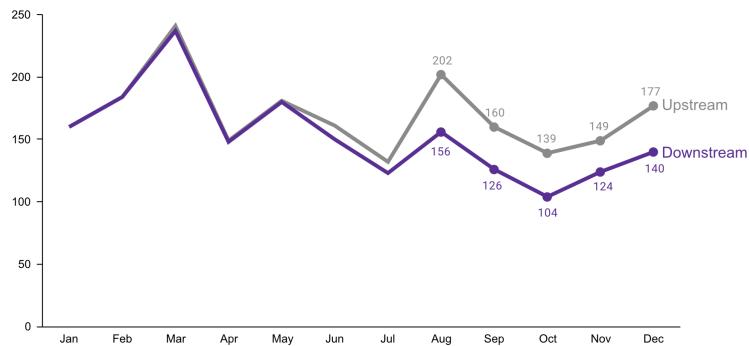


Figure 2.8: Start with gray and use pre-attentive attributes strategically to focus the audience's attention. [Example adapted from: *Storytelling with Data* by Cole Nussbaum Knaflic]

Step 4: Make data accessible

- Add context: Which values are good (goal state), which are bad (alert threshold)? Should the value be compared to another variable (e.g., actual vs. forecast)?
- Leverage consistent colors when information is spread across multiple plots (e.g., data from a certain country is always drawn in the same color).
- Annotate the plot with text explaining the main takeaways (if this is not possible, e.g., in interactive dashboards where the data keeps changing, the title can instead include the question that the plot should answer, e.g., “Is the material quality on target?”).

Fish population declines after chemical plant opens

Further investigation is needed to assess the potential role of thermal pollution.

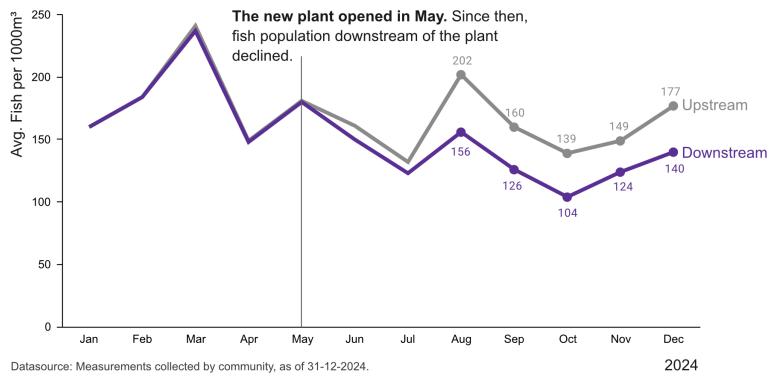


Figure 2.9: Tell a story. [Example adapted from: *Storytelling with Data* by Cole Nussbaum Knaflic]

2.3 Draw your What

You might not have looked at your data yet—or maybe you haven’t even collected it—but it’s important to start programming with the end in mind. Think about how, in software development, a UX designer typically creates mockups of a user interface (like the screens of a mobile app) before

developers begin coding. Similarly, in our case, we want to start with a clear picture of what the output of our program should look like. The difference is that, instead of users interacting with the software themselves, they'll only see the plots or tables that your program generated, maybe in a journal article.¹

Based on your takeaways from the previous chapter—about the problem you're solving and the metrics you should use to evaluate your solution—try sketching what your final results might look like. Ask yourself: *What figures or tables would best communicate the advantages of my approach?*

Depending on your research goals, your results might be as simple as a single number, such as a p-value or the total number of people surveyed. However, if you're reading this, you're likely tackling something that requires a more complex analysis. For example, you might compare your solution's overall performance to several baseline approaches or illustrate how your solution converges over time (Figure 2.10).

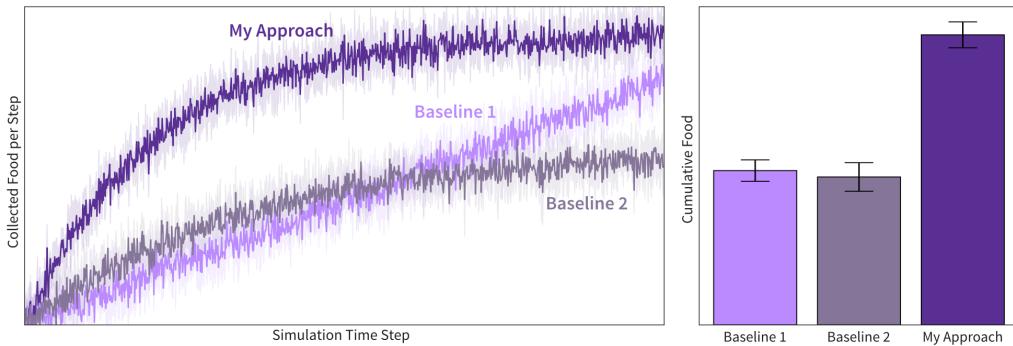


Figure 2.10: Exemplary envisioned results: The plots show the outcome of a multi-agent simulation, where ‘my approach’ clearly outperforms two baseline methods. In this simulation, a group of agents is tasked with locating a food source and transporting the food back to their home base piece by piece. The ideal algorithm identifies the shortest path to the food source quickly to maximize food collection. Each algorithmic approach was tested 10 times using different random seeds to evaluate reliability. The plots display the mean and standard deviations across these runs. *Left:* How quickly each algorithm converges to the shortest path (resulting in the highest number of agents delivering food back to the home base per step). *Right:* Cumulative food collected by the end of the simulation.

It’s important to remember that your actual results might look very different from your initial sketches—they might even show that your solution performs worse than the baseline. This is completely normal. The scientific method is inherently iterative, and unexpected results are often a stepping stone to deeper understanding. By starting with a clear plan, you can generate results more efficiently and quickly pivot to a new hypothesis if needed. When your results deviate from your expectations, analyzing those differences can sharpen your intuition about the data and help you form better hypotheses in the future.

Once you’ve visualized the results you want, work backward to figure out what data you need to create them. This is especially important when you’re generating the data yourself, such as through

¹A former master’s student that I mentored humorously called this approach “plot-driven development,” a nod to test-driven development (TDD) in software engineering, where you write a test for your function first and then implement the function to pass the test. You could even use these sketches of your results as placeholders if you’re already drafting a paper or presentation.

2 Data & Results

simulations. For instance, if you plan to plot how values change over time, you'll need to record variables at every time step rather than just saving the final outcome of a simulation (duh!). Similarly, if you want to report your model's accuracy, you'll need (Figure 2.11):

1. Input variables for each data point to generate predictions (= model output).
2. The actual (true) values for each data point.
3. A way to compute the overall deviation between predictions and true values, such as using an evaluation metric like R^2 .

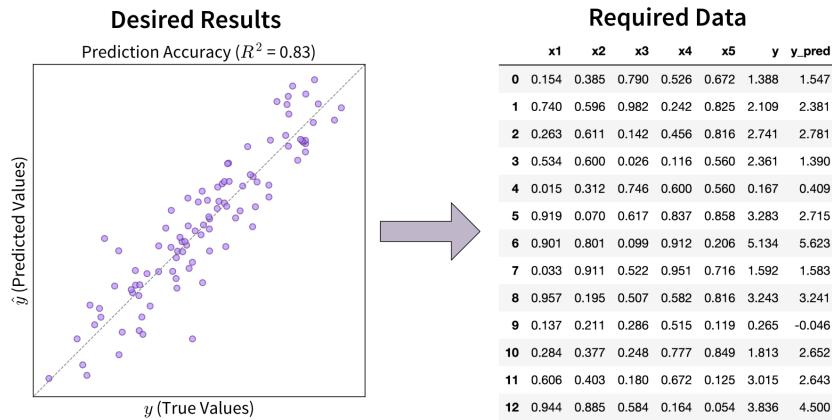


Figure 2.11: Work backwards from the desired results to determine what data is necessary to create them.

By working backward from your desired results to the required data, you can design your code and analysis pipeline to ensure your program delivers exactly what you need.

i Before you continue

At this point, you should have a clear understanding of:

- The specific results (tables and figures) you want to create to show how your solution outperforms existing approaches (e.g., in terms of accuracy, speed, etc.).
- The underlying data needed to produce these results (e.g., what rows and columns should be in your spreadsheet).

3 Tools

Before we continue with creating your results - i.e., actually start with the programming part - here is a quick overview of the tools that could come in handy during your software engineering journey.

While the examples will use the Python programming language, the general principles discussed in the book are equally applicable to other programming languages.

3.1 Programming languages

choose the one that is most common in your field - you'll find more relevant resources and libraries that will come in handy and it's easier to collaborate with co-workers.

i Before you continue

At this point, you should have a clear understanding of:

- How to set up your development environment to code efficiently.
- The fundamental syntax of your programming language of choice (incl. key scientific libraries) to get started.

4 Software Design

5 Implementation

6 From Research to Production

References

1. Freiesleben T, Molnar C (2024) [Supervised machine learning for science: How to stop worrying and love your black box](#).
2. Horn F (2021) [A practitioner's guide to machine learning](#).
3. Callaway E (2024) Chemistry nobel goes to developers of AlphaFold AI that predicts protein structures. *Nature* 634: 525–526.
4. Knaflic CN (2015) Storytelling with data: A data visualization guide for business professionals, John Wiley & Sons.

