Uncertainty Research Lab Summer 2022: Data Science for Engineers

Katelyn Vo, Maeve Stites,
and Zach del Rosario

August 5, 2022

1 Introduction & Context

Before the 1950's, the Air Force believed in the 'average man' [4]. The 'average man' was believed to be the average in human body data as a way to oversimplify the variability in a population of pilot bodies. This was to simplify the design for a cockpit to fit everyone since it would fit the 'average man.' It was only until further investigation by Gilbert Daniels revealed that out of the thousands of Air Force pilots (over 4,000 pilots), no one was average in all dimensions of a pilot. Daniels quoted that "'the average man' is a misleading and illusory concept as a basis for design criteria," illustrating the importance of handling variability in the real world and the consequences of only using a single value (the mean) to represent a distribution [4]. However, this is not the only example of variability negatively affecting the effectiveness of designs, and furthermore, this is not a widely known study.

Despite the results of this study, there is a lack of research regarding how engineers handle variability. Many industries rely on single values (like the mean) when doing engineering design work. This flattens real world variability, and opens up the potential to design shortfalls and probability of failure. For example, if an engineer were building a structural system out of an aluminum alloy, they might use a mean strength value of that alloy when doing calculations for the design. Because real world materials have variability, meaning the samples will exhibit a range of different strength value, this design approach could lead to mechanical failure if the sample used in the construction falls below the mean strength used in the design. This is why it is important that engineers are given adequate tools and knowledge on how to approach real world variability.

In this study, we conducted interviews with engineers to understand how they approach real world variability. The goal of this study is to better understand how engineers reason under uncertainty, and to provide more

baseline research that will help us to better educate engineers on how to approach variability.

1.1 Background

Previously in the realm of this research, Zachary del Rosario first tested the question of how engineers deal with variability on 7 engineering students [1]. In his research, del Rosario investigated how these students identify and treat variability by using real-world examples used in the developmental workshops and through interviews. The findings revealed a disconnect between the "whats" and "whys" of statistical analysis, especially when understanding *variability*.

Del Rosario constructed a theoretical framework in the previous research that helped explain the differences in what the world sees as 'variability' [1].

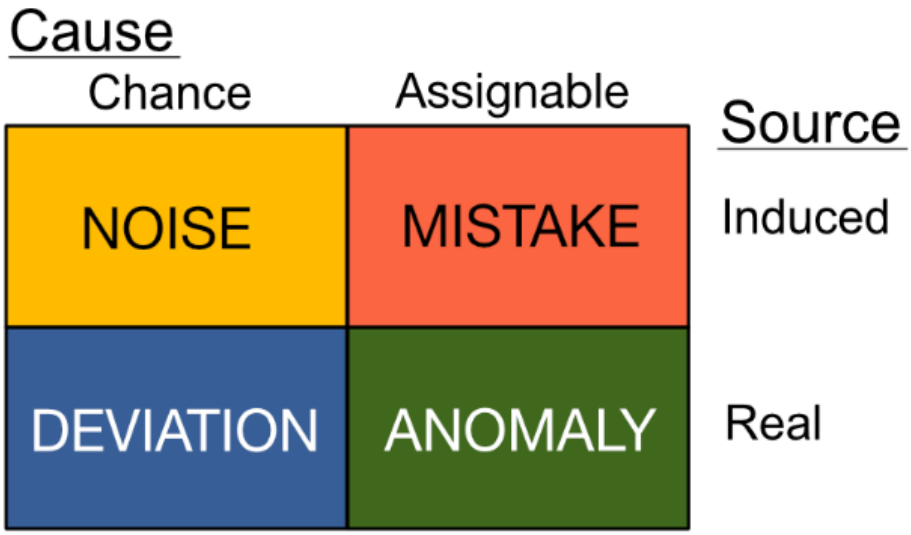

Figure 1: Cause-Source variability quadrant diagram. This diagram illustrates a "cause" and "source" dimension, "cause" being the concept of chance versus assignable (controllable) and "source" being the concept of real versus induced (not real/erroneous) variability.

The "Cause" axis consists of "chance" and "assignable" which determines the practicality of variability. "Assignable" (controllable) means more effort can reduce the variability in the system. While "chance" means variability is impractical to reduce in the situation; an example of chance is attempting to eliminate cracks in all parts.

The "Source" axis consists of "real" and "induced" (not real) variability. Real variability affects material properties while not real variability does not affect material properties. For example, "real" variability can be char-

acterized as a crack in the material affecting its strength, while "not real" variability where imprecise measurements of a material creates variability within a dataset.

Understanding these quadrants and definitions are important since these are used for the coding scheme and analysis in the study. Del Rosario's previous study was a foundation for this line of research, but had its limitations. Although it led to the creation of this framework and brought forth further questions on the topic, the study was primarily conducted on engineering students. Since the participants were only engineering students, they did not have much experience to draw from for the questions asked. Furthermore, the study only interviewed 7 participants, limiting the amount of data and findings for the research.

Comparing the previous study to the one in this paper, the first study tested preconceived notions on the question (focusing on hypothesis testing) while the current study aims to create a theory explaining the behaviors observed during interviews.

2 Data Science for Engineers Study

The Data Science for Engineers Study had 24 participants all of whom were interviewed before starting the course. In the time span of 6 weeks, the engineers participated in the course which helped with further data collection in the research. By utilizing both the transcripts from interviews and the information gathered via course work, we were able to formulate a preliminary theory that explains how engineers react to variability.

2.1 Methodology

2.1.1 Background Research

Grounded Theory observes the idea of building a theory from the data itself as opposed to testing a preconceived hypothesis. Grounded Theory is also more used for qualitative data, making it easier to interpret and make sense of raw data to make discoveries. By piecing these discoveries together, a theory can be developed and improved iteratively until the theory explains the behaviors we see. With Grounded Theory, we can construct an understanding of how engineers handle variability in the real world.

2.1.2 Interviews

We interviewed participants with a semi-structured interview protocol to inquire how these engineers approach variability found in data sets. The interview consisted of 10 prompted questions with notes allowing us researchers to ask any follow up questions necessary to understand the participants'

answers. These were conducted as one-on-one interviews online and lasted between an hour to an hour and a half long. Before conducting the interviews, we asked the participants for their consent to allow the interviews to be recorded.

A total of 24 participants were interviewed, all of whom had more than two years of engineering experience in the industries such as Civil, Mechanical, and Aerospace.

2.1.3 Course

As part of the study, Professor Rosario taught a five week "Data Science for Engineers" course of which the 24 participants participated in. This course was designed to educate engineers to give them modelling and data analysis skills to deal with variability in engineering systems; in this case, examples were done in the particular context of Aerospace engineering and material safety [6].

The interviews described in the above section were conducted at the beginning of the course, with follow-up scheduled after the completion of the course. Comparisons will be drawn between the pre-course and post-course interview to see how education impacts how engineers approach variability. These post-course interviews still need to be conducted.

In addition to the course's role in the study, information gathered from research participants supported software development work and software development research. Read more about this in subsection Grama - Python Package Development.

2.1.4 Coding and Analysis

Coding helps with understanding qualitative data presented in the form of interviews[3]. These codes allow for the transfer of founded ideas between interviews. To make sense of the data from participants, we conducted stages of coding to help conceptualize phrases and actions done by the participants to create a theory.

The first stage of coding was initial coding. The process consisted of tagging sections of the transcripts to help summarize the choices made by the participants. For example, when a participant mentioned taking a statistical quantity or explaining why they would make their decision, we applied tags to the lines to describe their actions. Descriptions concerning how an engineer handled the data set, both statistically and non-statistically, were tagged and reviewed by other researchers in the group before analysis. An example of these tags are shown below in italics using parts of the interview with Participant 6:

PARTICIPANT 6: Here, I think that I would side towards using the

min value as a conservative value to use for design *(statistics: min)*. I would report that the tensile yield strength of this material is whatever the min is, 155.7, I think 155.6 *(number answer), identifying a material by its min.*

The first sentence was tagged as "statistics: min" since the participant specifically said they would use the minimum value for the question asking participants to describe steel from a provided dataset (Q2). This is however different from the second sentence which was tagged as "(number answer)" and "identifying a material by its min" in which the participant specifically stated they would describe a material's property by its minimum. These lines and phrases became important to contrasting and comparing how other participants answered these questions.

After initial coding, researchers began focused coding. Focused coding takes the codes from initial coding and categorizes these ideas into concepts. This particular part of coding elevates the data into the creation of a theory, in this case, a theory explaining how engineers deal with variability. The results from the focused coding clarify the "Singular-to-Targeted" Coding Scheme discussed more in the following subsection Preliminary Results.

2.2 Preliminary Results

For each question asked in the interview protocol, we interpreted the choices made by the participants. These choices were categorized into three groups that help explain how engineers deal with variability, which were deemed the "closed coding" scheme. These categories were:

- Singular – the participant ignored or neglected the variability in their analysis of the data set.

- Acknowledged – the participant acknowledged the variability but failed to design for the variability in the data set.

- Targeted – the participant both acknowledged and specifically designed for variability in the data set. These designs may not be the best in analysis, but if the participant understand the consequences of variability and design to decrease variability, the choice made was labeled as Targeted.

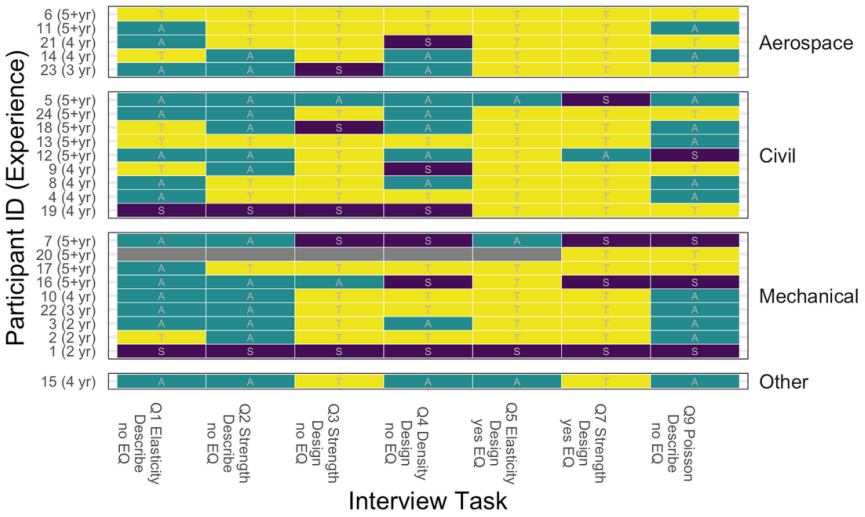

Figure 2: Singular-To-Targeted Coding Scheme. For each participant and sets of questions, each box was assigned a letter to represent how the participant analysed the data set shown in the question. The letters correspond S for Singular (ignores/denies variability), A for Acknowledged (acknowledges but does not design for variability), and T for Targeted (acknowledges and designs for variability). The grey, unlabeled boxes for earlier parts of Participant 20 were due to corrupted recording data, therefore the earlier section of the interview was uncodeable.

These categories describe the types of analysis participants took for particular questions. The following below include examples from interviews that demonstrate the different categories and how they were coded.

Singular analysis consists of the participant ignoring or denying variability in the system. In this case, Participant 1 only used the mean and did not look at any other factors in all of the data sets. The following quote specifically for Q3, a design questions about the provided tensile yield strengths of a cast steel alloy:

PARTICIPANT 1: I guess it's because it's also, the sample is ranging from 1 to 10, and the data they collected, we are doing the same thing on it, the same motion, or same measurement or same ways to collect the data. We can just take the mean of all the 10 samples, and then get a good idea of what is the mean value.

The participant failed to acknowledge or recognize any variability in the data sets. This and only taking the mean resulted in a Singular analysis of the data across the observed questions.

Acknowledged analysis consists of the participant acknowledging the variability but not addressing the consequences of variability by changing their analysis. Participant 22 used the mean despite acknowledging variability in the data set for Q9, a design question about the Poisson ratio of aluminum alloys.

PARTICIPANT 22: It's very steady because the difference between all the samples are minimal. It has a small standard deviation...It has a Poisson's ratio 0.3 mean value. I would just say this aluminum alloy has a Poisson's ratio mean value, but putting some numbers. That's it.

Although the variability of the data set was acknowledged, the participant failed to change their analysis to account for the variability.

Targeted analysis consists of the participant acknowledging the variability and changing their approach to adjust for the variability in the system. For example, Participant 21 used the minimum value of the data set for the steel description question (Q2):

RESEARCHER: How come you are choosing to do the minimum for this particular question as opposed to the other one where you're using the mean? How come you're doing a different process for that?

PARTICIPANT 21: Probably because of elasticity being a slope. It just feels like it is better to have the best estimate possible of what that slope actually is, which I would think best practice is taking a measure of a mean or a median whereas yield strength is a point of no return for a material. I would be much more cautious of just taking an average and I would want to play it safe with worst-case scenario.

Since Participant 21 observed the question in the context of a 'worst-case scenario,' the participant specifically changed their analysis to use a minimum value instead of previously taking the mean.

With the results of the Singular-To-Targeted Coding Scheme and the categorization of the focused codes, we developed the following grounded theory.

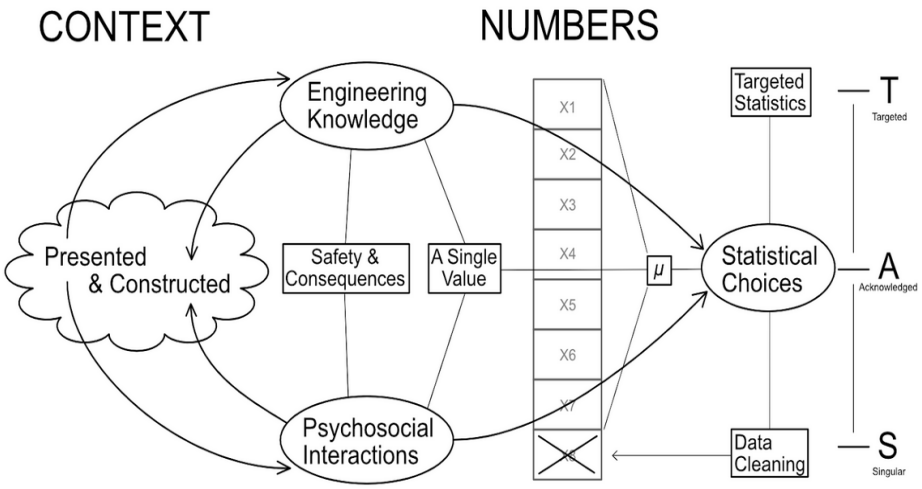

Figure 3: The Context-Engaged Engineering Data Analysis (CEEDA) model. This diagram aims to explain the constructed theory on potential factors that play a role in an engineer's decision on designs facing variability.

The CEEDA model helps explain the trends and observations from the closed coding scheme mentioned earlier. The model was built based on the data from the study in which we manipulated the context and numbers presented to affect what statistical choice participants made. The main categories of the model are: Engineering Knowledge, Psychosocial Interactions, and Statistical Choices. These categories work together to explain the thought process that engineers undergo. When an engineer analyzes a dataset, the context of the data affects their Statistical Choices.

Both Engineering Knowledge and Psychosocial Interactions build and work upon the context presented to the engineer in a feedback loop as illustrated in the model. We further observed both "Safety Consequences" and "A Single Value" when coding through the interviews and determined these ideas as important factors in an engineer's decision-making in design. Many of the participants recognized the consequences of variability in the dataset and connected these consequences to the jeopardized safety of humans (although not explicitly). This idea still affected how the engineers continued with their analyses as safety played a role in their Statistical Choice. This is also the case for "A Single Value" in which a myth in engineering is that a single value is required as a numerical input for analysis. Due to this, many engineers justified taking the mean of the dataset meanwhile others took a range when determining the attributes from the dataset.

The Statistical Choices, referenced earlier in the Singular-to-Targeted closed coding scheme, were influenced by these factors mentioned above. However, this is only an overarching understanding of the observations we

found in the study. There are still some trends found in the Singular-to-Targeted scheme that are not fully explained by the CEEDA model, but will be continued in future iterations and continuations of the research.

3 Grama - Python Package Development

In conjunction with the work focusing on understanding how engineers approach variability, we also worked on developing the Grama python package. Grama is an open-source software package of modeling and data science tools designed to make uncertainty quantification[1] more accessible to people regardless of statistical background [5]. This was the package used for teaching the course.

This ease of access to tools is important to consider when thinking of how engineers approach variability. For engineers to be able to use statistical analysis methods to make their design techniques best reflect real world behavior, they need to have access to intuitive and applicable data analysis and modelling tools. This is why the development of packages like Grama is important.

The goal of this research was to:

1. Make Grama a more robust software package for use in future iterations of the study, which should support research efficacy.
2. Make Grama a more accessible and intuitive software package to support wider engineering and general data science efforts.
3. Understand how we can better design software for users.

3.1 Methods

We created a feedback loop with students through office hours, course feedback, and communication over Discord. Synthesizing these experiences enabled us to identify areas where Grama was intuitive (and successful in its design), and areas needing improvement.

We took identified areas of needing improvement, discussed them as a research team, and formalized an issue write-up that enabled us to implement software solutions to improve the design of the package.

3.2 Research

3.2.1 Meaningful Error Messages

Grama uses code patterns that are modelled off of the Tidyverse package and its behavior in the R language; a programming language designed for

[1] Uncertainty Quantification is the science of analyzing uncertainty in scientific problems and using those results to inform decisions.

statistical computing and graphics [9] [8]. Because of this, Grama follows code patterns and syntax that at times replicate R more than they do follow standard Python protocols.

A main example of this is the usage of 'pipes'. Pipes allow us to essentially 'pass' an object (namely a Dataset or Model) to a function. This is helpful for stringing along multiple operations in an intuitive function. An example of pipes:

Listing 1: Grama example of pipes

```
(
    # Select a dataset
    df_dataset
    # Start the plot
    >> gr.ggplot(
        # Map the aesthetics to variables in df_dataset
        gr.aes(
            x="x_variable_name",
            y="y_variable_name",
        )
    )
    # Select a geometry to visualize the aesthetics
    + gr.geom_point()
)
```

In this example, a pipe (the '>>' symbol) is being used to 'pass' a Dataset (df_dataset) to a Grama plotting function (gr.ggplot).

These coding patterns encouraged by Grama can lead to issues that are difficult to debug; this is both because these coding patterns are not native to Python's design, leading to unhelpful built in error messages, and because these coding patterns may be unfamiliar to those with a heavier Python background.

We saw this issue when a student brought an issue where they had accidentally defined a Tuple object, a built-in Python object, when they meant to define a Model object, a Grama object for modelling. They then tried to perform an operation on this model, leading to an unhelpful error message. This whole issue was caused by one mis-placed comma [7].

Listing 2: Erroneous code

```
(
    # model declaration
    md = (
        gr.Model()
        >> gr.cp_vec_function(
            fun=lambda df: gr.df_make(y=df.x),
            var=["x"],
            out=["y"],
        ), # BAD COMMA
    )
    # operation: evaluation of model
```

```
    gr.eval_df(md, df=gr.df_make(x=1))
)
```

When ran, this code returns the rather unhelpful error message: "AttributeError: 'tuple' object has no attribute 'functions'." [7]. This error message provides no intuition to the user as to what is causing the problem (the comma).

Listing 3: Correct code

```
(
    # model declaration
    md = (
        gr.Model()
        >> gr.cp_vec_function(
            fun=lambda df: gr.df_make(y=df.x),
            var=["x"],
            out=["y"],
        )
    )
    # operation: evaluation of model
    gr.eval_df(md, df=gr.df_make(x=1))
)
```

To address this issue, we first thought about the best way to construct a helpful error message for the user. We chose to model our errors off of how the Tidyverse handles errors as the Tidyverse is well regarded for its user intuitiveness, particularly around addressing errors [10][2].

We took the principles of concise and clear problem statement, followed by a hint for the user as to how to fix the issue. We value this approach in particular as Grama is designed with accessibility in mind, and we want users to be able to quickly resolve issues - especially basic ones like this. The original error message does not provide any intuition into where the problem might be or how to start troubleshooting.

These principles led us to a more relevant error message: "TypeError: Given model argument is type tuple. Have you declared your model with an extra comma after the closing ')'?". Compared to the former error message, this error message much more clearly defines the issue, that the "model argument is type tuple". Additionally, the hint question gives the user a place to start, and potential for a quick fix.

3.2.2 Code Maintainability & Invariant Tests

While working on the issue described in the section above, we observed an issue with code maintainability. The way Grama was written, adding new invariant tests (for more relevant error messages) would require integrating the new error check to each relevant function. We determined the best solution was to group common invariant tests for certain Grama functionality

(in this case Model evaluation) into a centralized location and function. This decision to create a centralized location for adding invariant tests results in a number of advantages:

1. Reduces time and barrier to adding more relevant and specific invariant tests; i.e. messages that are concise and provide user hints for a common bug. This would help ensure more frequent error message updates, something that can easily be left aside with other software development work.

2. Reduces time and barrier to updating invariant test messages.

3. Reduces risk of inconsistent or non-matching error messages across functions that should have the same invariant tests.

4. Reduces the amount of duplicate code. This makes the code base smaller and easier to parse.

3.3 Results

The most direct outcome of this work is a more robust package for use in future studies involving a course, and a more robust package for general modelling and uncertainty quantification work. Informative error messages and improved maintainability should make Grama more intuitive to use; though future observations and feedback should be taken into account to verify this hypothesis. We also come out of this research with meaningful takeaways for designing more intuitive software:

1. We see value in the Tidyverse style of error messages, particularly the concept of 'hinting'. We have observed through personal programming experiences learning the package, and interactions with students, that these 'hints' can make make a function more intuitive to use. This is of course conditional on the relevancy of the hint; as misleading hints can hurt more than they may be able to help [2].

2. Maintainability is an important consideration when it comes to making code more intuitive to use and troubleshoot. Easily update able error messages and documentation lower the barrier to updating based on user feedback.

4 Conclusion

The research concluded with observations of engineers' ability to directly Target the consequences created from variability via the Singular-to-Targeted Analysis coding scheme. The study also ended with the formation of the

Context-Engaged Engineering Data Analysis (CEEDA) model which explains the considerations engineers take when designing for variability. However, to fully explain how engineers react to variability, we still need to collect more data and further build upon the constructed theory. Through this iterative development, we hope to explain the trends we see in our data both in this study and future studies. We plan to continue our research with a new batch of participants following the same protocols and course work and continue developing the theory.

5 Acknowledgement

Thank you to the Clare Booth Luce Program for partial funding for the project. Thank you to AJ Evans, Ellie Ramos, and Emma Fox for their work in the Uncertainty Research Lab for the Data Science for Engineers Study.

6 Footnotes

References

[1] Riya Aggarwal, Mira Flynn, Sam Daitzman, Diane Lam, and Zachary Riggins del Rosario. A qualitative study of engineering students' reasoning about statistical variability. In *2021 Fall ASEE Middle Atlantic Section Meeting*, 2021.

[2] Jenny Bryan. Object of type 'closure' is not subsettable, 2020. Accessed: 2022-07-22.

[3] Kathy Charmaz. *Constructing grounded theory: A practical guide through qualitative analysis*. sage, 2006.

[4] Gilbert S Daniels. *The "Average Man"?* Wright Patterson A.F.B, 1952.

[5] Zachary del Rosario. Grama github repository readme, 2021. Accessed: 2022-08-05.

[6] Zachary del Rosario. Data science for engineers course website, 2022. Accessed: 2022-08-05.

[7] Zachary del Rosario. Grama github repository tuple trap issue, 2022. Accessed: 2022-08-05.

[8] R Foundation. R project, 2022. Accessed: 2022-08-05.

[9] R Studio. Tidyverse, 2022. Accessed: 2022-08-04.

[10] Hadley Wickham. Tidyverse error message style guide, 2022. Accessed: 2022-08-03.