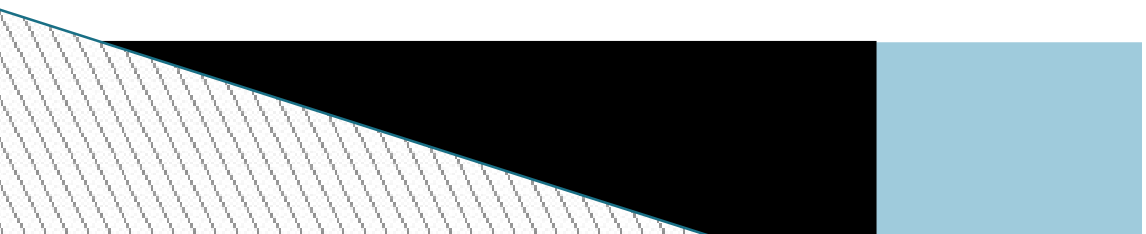
# Preparing math students for careers in industry:
## A perspective from a career changer

AMS Committee on Education
October 30, 2015

Paul Koester
Quantitative Research and Analytics
Allstate

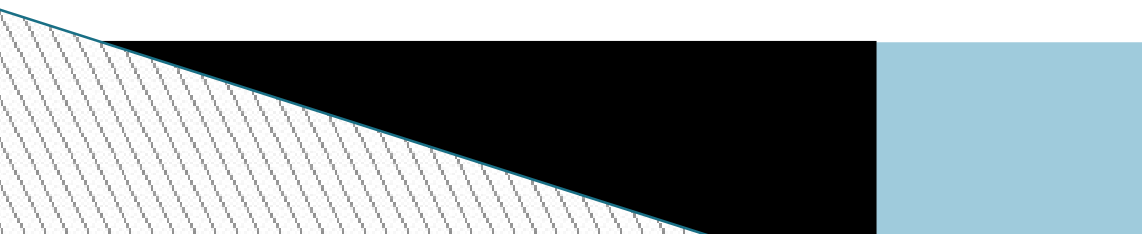
# About Me

- PhD Mathematics, Washington University
- Arithmetic Combinatorics, Number Theory, Harmonic Analysis
- 2007 – 2014:  Teaching positions at Indiana University, St. Louis University, University Kentucky
- 2014 – Present:  Data Scientist at Allstate



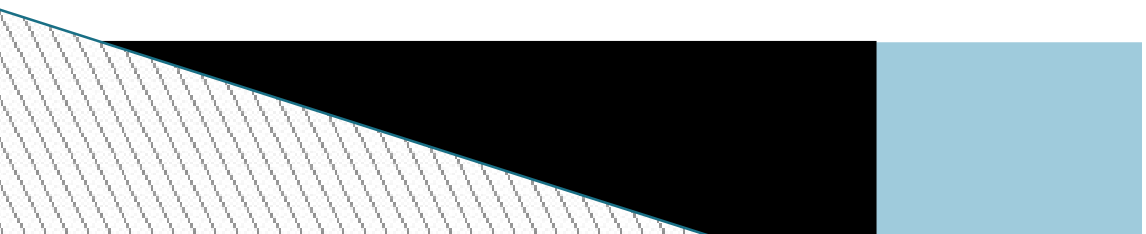
# What is Data Science?

- Extract knowledge and insights from vast amounts of data
- Interdisciplinary field combining techniques from statistics, software development, mathematics, machine learning, etc



# How do I spend my time?

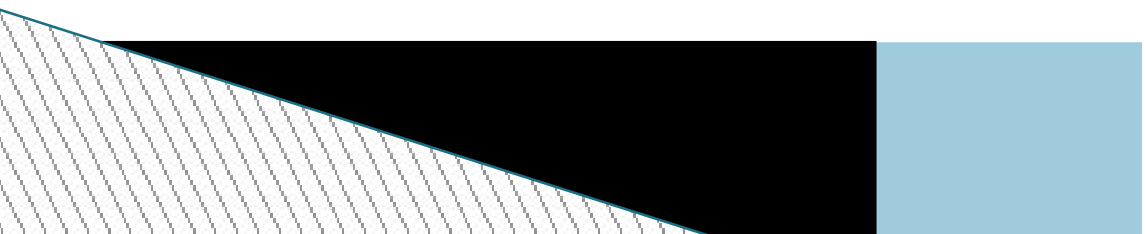
- 25 % of time in data acquisition
- 50 % of time in "data cleaning"
- 10 % of time building and validating models
- 5 % of time interpreting results
- 5 % of time discussing specifications or results with business partners



# Who are my fellow data scientists

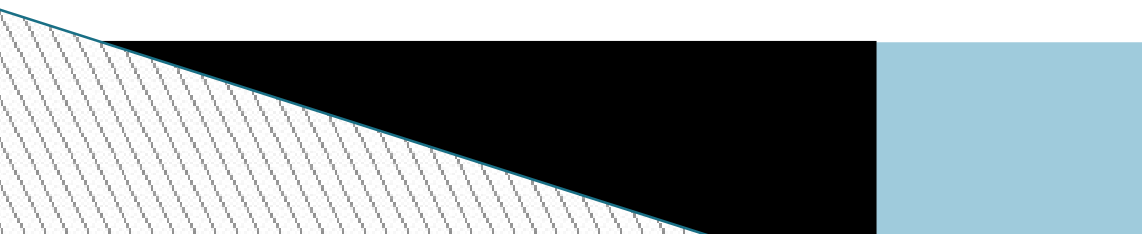
- Almost all have PhDs or MSs. Those without graduate degree have many years experience in business analytics

- Backgrounds include mathematics, physics, statistics, computer science, material science, econometrics, actuarial science, public policy



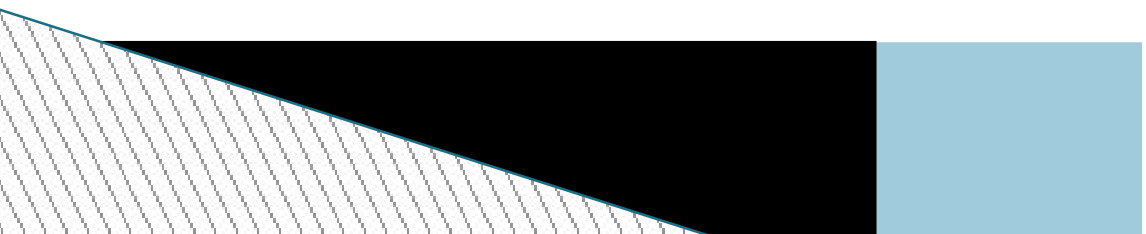
# Challenges in adapting to industry

- You work on what your client deems as valuable
- What your client deems as valuable can change mid-project
- Your client can change mid-project
- Project assignments can change mid-project



# Challenges for mathematicians

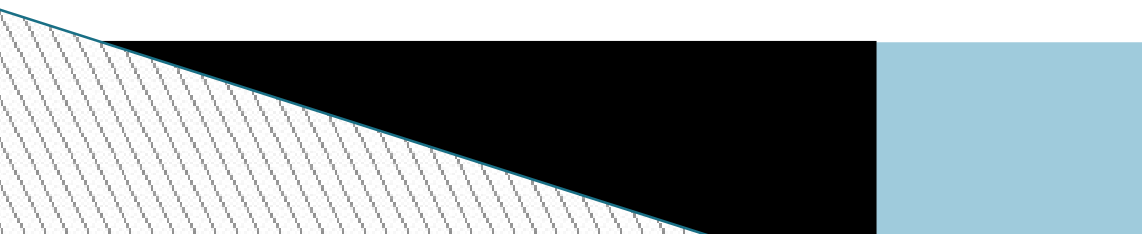
- Real world problems **never** have solutions
- Real world questions are **always** vague and ill-posed
- Real world data is **always** messy
- Computational run time is **always** an issue
- Real world work is experimental, not deductive
  - You can **never** prove your result



# Preparing Students for Jobs in Industry

# Communication Skills

- Ability to determine the appropriate level of detail
- Ability to effectively communicate at that appropriate level
- Ability to communicate effectively to different audiences



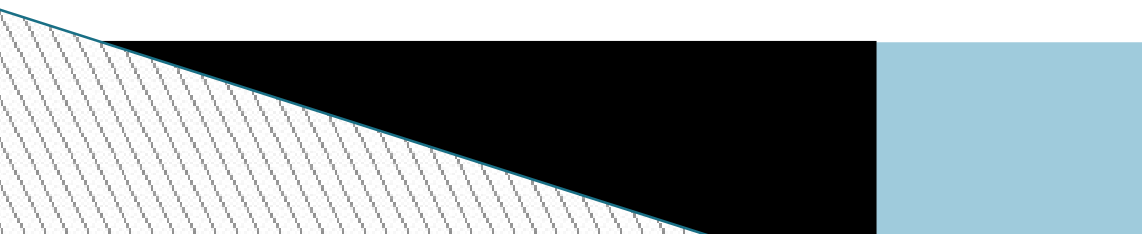
# Problem Solving Skills

In applied work

- Questions are **always** ill-posed
- Assumptions are **always** vague
- Problems can be **valuable** or **solvable,** but never both!!

Progress requires working iteratively:

- Reformulate until problem is solvable, but no longer valuable
- Reformulate until the problem is valuable, but no longer solvable

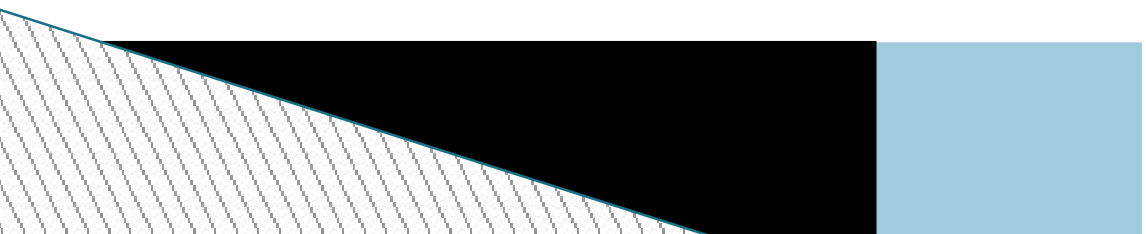


# Defend Assumptions

In applied work

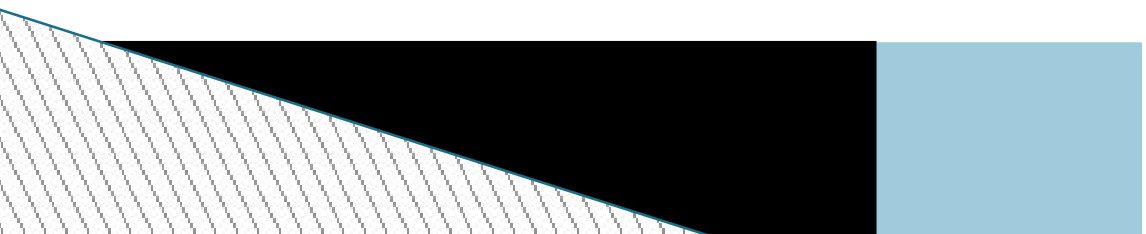
- Questions are **always** ill-posed
- Assumptions are **always** vague
- Problems can be **valuable** or **solvable,** but never both!!

Progress is made by **reformulating** problems.
There is never a "right way" to reformulate a problem.
So you must always be prepared to answer "why did you reformulate it in this way?"



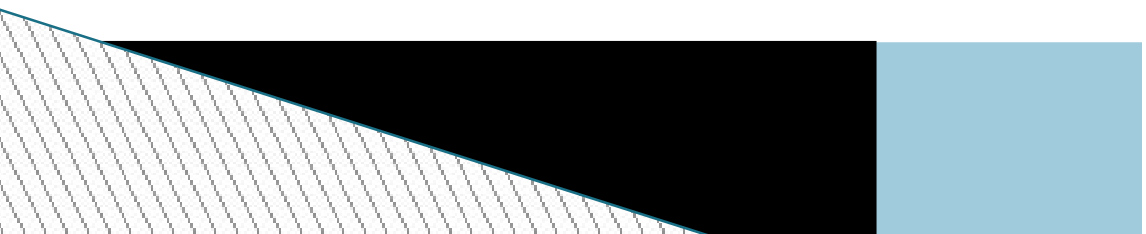
# Written Communication Skills

- Majority of written work in industry job will be writing executable code or writing documentation for code
- There are lots of similarities between writing code and writing a proof
- So training students to write better proofs will also help them write better code



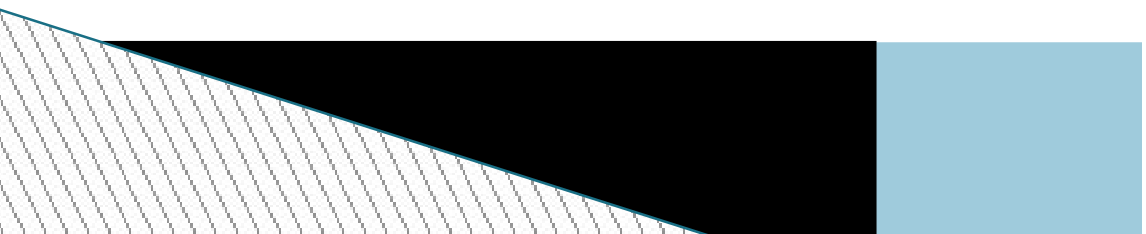
# Lessons Learned from Software Development Community

- Clean Code Principles
  - Variables, methods, and classes should be descriptively named
  - Methods should do one and only one thing
  - One level of abstraction per method
  - Avoid repetition -- encapsulate into methods instead
- Refactoring Code
  - Improving the internal structure of code without altering its external behavior
- Standards for documenting code and assumptions



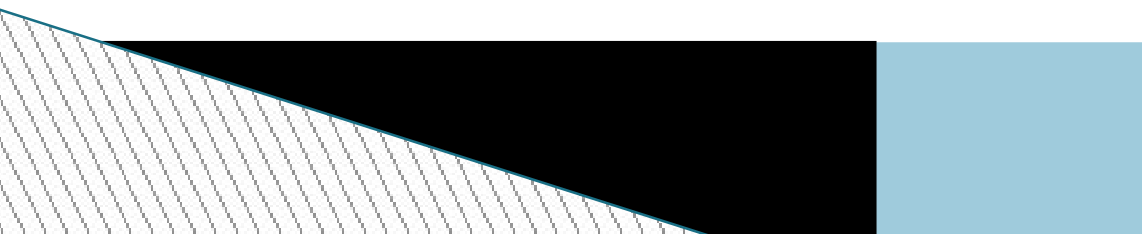
# Adapting Software Development Principles to Proof Writing

- Use descriptive names
  - No more “We say a function has **property A** if **…**”
- Lemmas should encapsulate one and only one significant logical idea
- A proof should operate at a single level of abstraction
  - Multiple levels of detail is a sign that parts of the proof should be encapsulated into a lemma
- When the same argument is repeated multiple times, encapsulate that argument into a lemma



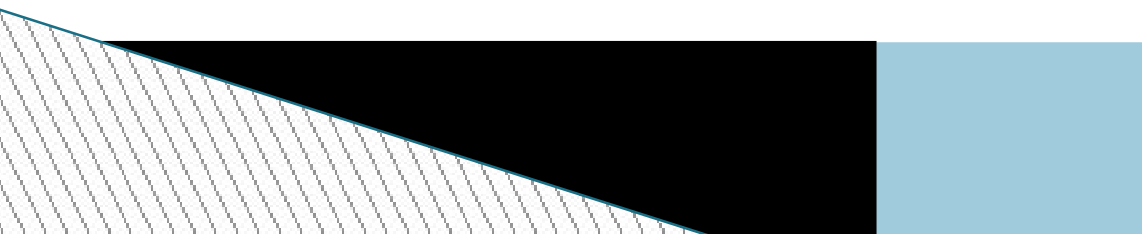
# Adapting Software Development Principles to Proof Writing

- Refactoring in the context of proof writing?
  - Decomposing long complex proof into a simpler series of lemmas
  - Makes proof much easier to follow
  - Isolates where different hypotheses of the theorem are used
  - Makes it easier to modify proof to yield new theorems



# Conclusion

- Improving soft skills will help students succeed in industry jobs
- This can be done directly in math courses
- This can be done in a way that strengthens the current curriculum



# Thank You


- Paul Koester
- paulhkoester@gmail.com
- pkoes@allstate.com