



# Facilitating team-based data science

...

Lessons learned from the DSC-WAV Project

Benjamin S. Baumer, Chelsey Legacy, Andrew Zieffler, and Nicholas J. Horton

[shorturl.at/jsvxK](https://shorturl.at/jsvxK)



# While we are getting started

To help us better engage can you please add the following into the Zoom chat?

Your name

Your affiliation

Some good news that you'd like to share





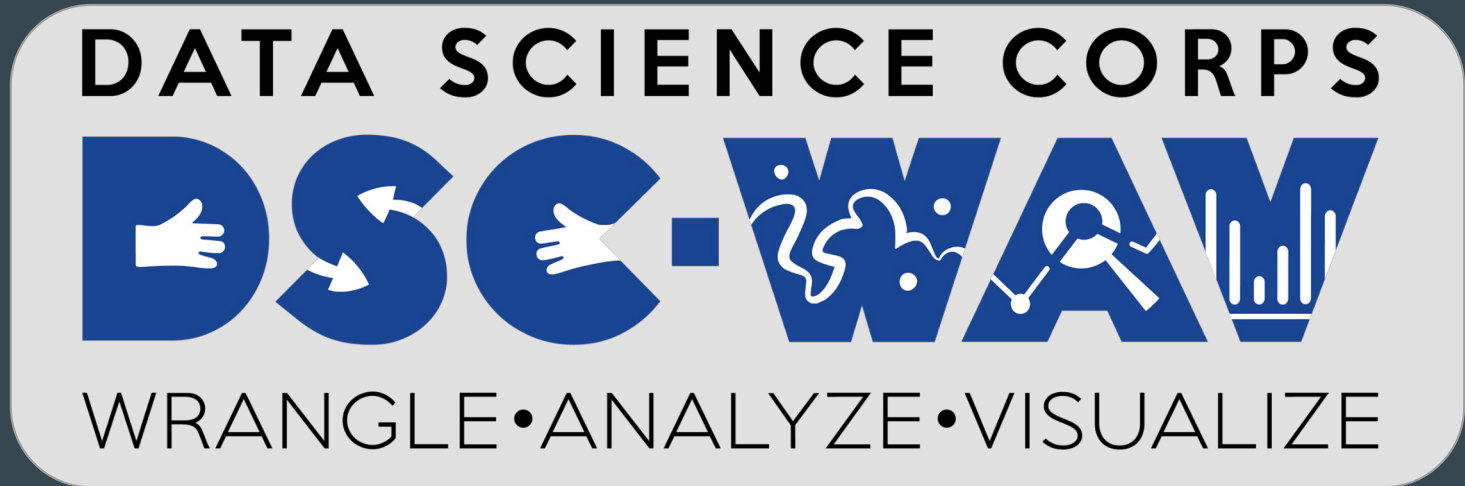
# Outline

- What is the DSC-WAV project?
- What are Agile, Scrum, and Kanban
- Why are these methodologies useful for data science education?
- Interactive Exercises:
  - Writing user stories
  - Prioritizing user stories on a Kanban Board
- What is code review and how does it work? (as time permits)
- What have we learned from this project?



# DSC-WAV (Wrangle-Analyze-Visualize)

- NSF funded effort from the Harnessing the Data Revolution (HDR) Data Science Corps (DSC) initiative





# DSC-WAV (Wrangle-Analyze-Visualize)

- <https://dsc-wav.github.io/www>
- Collaborative project with:
  - Five Colleges (Amherst, Smith, Hampshire, Mount Holyoke, and UMass/Amherst)
  - Greenfield Community College, Holyoke Community College, Springfield Technical Community College,
  - University of Minnesota





# DSC-WAV (Wrangle-Analyze-Visualize)

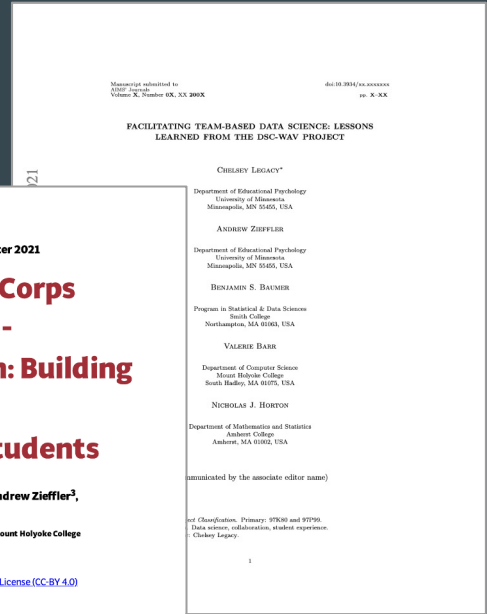
- Goal 1: create opportunities for undergraduate students to work on Data Science for Social Good projects for community organizations





# DSC-WAV (Wrangle-Analyze-Visualize)

- Building data acumen for undergraduate students
  - *HDSR*, <https://hdsr.mitpress.mit.edu/pub/nvflcexe/release/1>
- Facilitating team-based data science: lessons learned
  - *FDS*, under review, <https://arxiv.org/abs/2106.11209>



Harvard Data Science Review • Issue 3.1, Winter 2021

## The Data Science Corps Wrangle-Analyze- Visualize Program: Building Data Acumen for Undergraduate Students

Nicholas J. Horton<sup>1</sup>, Benjamin S. Baumer<sup>2</sup>, Andrew Zieffler<sup>3</sup>,  
Valerie Barr<sup>4</sup>

<sup>1</sup>Amherst College, <sup>2</sup>Smith College, <sup>3</sup>University of Minnesota, <sup>4</sup>Mount Holyoke College

Published on: Feb 25, 2021

License: [Creative Commons Attribution 4.0 International License \(CC-BY 4.0\)](https://creativecommons.org/licenses/by/4.0/)



# Introduction to Agile and Scrum

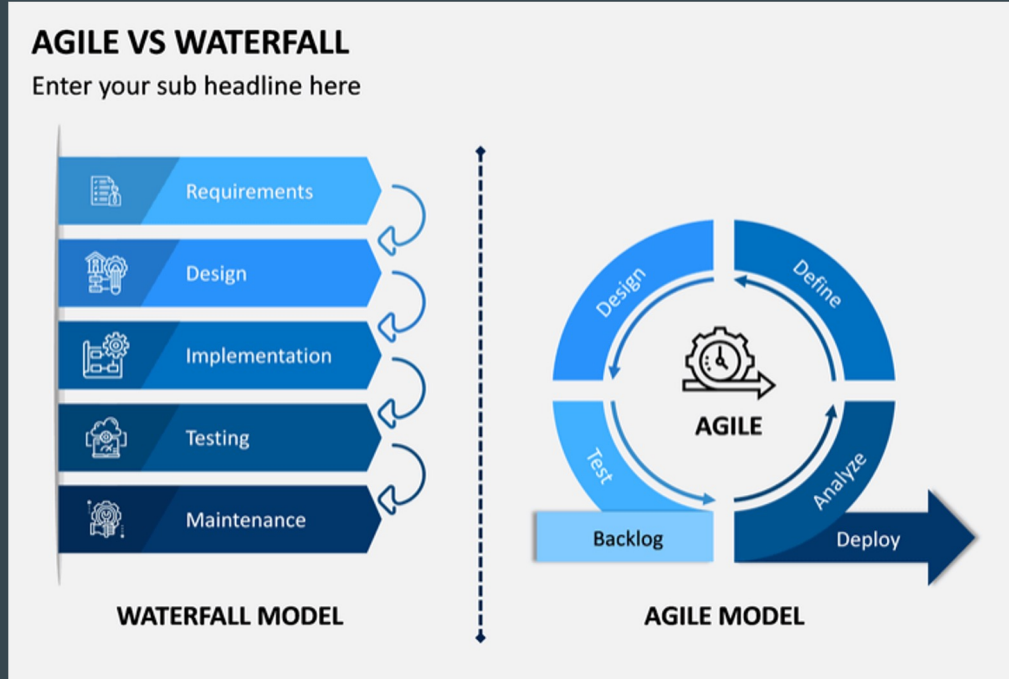


# Agile Philosophy and Scrum Manifesto

1. Individuals and interactions over processes and tools
2. Working software over comprehensive documentation
3. Customer collaboration over contract negotiation
4. Responding to change over following a plan

Q: Can we port these insights from software development into statistics and data science education?

Q: Can we use these approaches to improve data analysis and team-based collaboration?



# The Agile Scrum Framework at a glance

Inputs from  
Customers, Team,  
Managers, Execs



Product Owner



The Team



Scrum  
Master



Burn Down/Up  
Chart



Daily Standup  
Meeting

24 Hour  
Sprint

1-4 Week  
Sprint



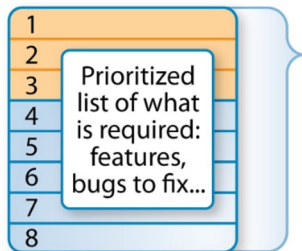
Sprint Review



Finished Work



Sprint  
Retrospective



Product  
Backlog

Team selects starting at top as much as it can commit to deliver by end of Sprint

Sprint  
Planning  
Meeting

Task  
Breakout

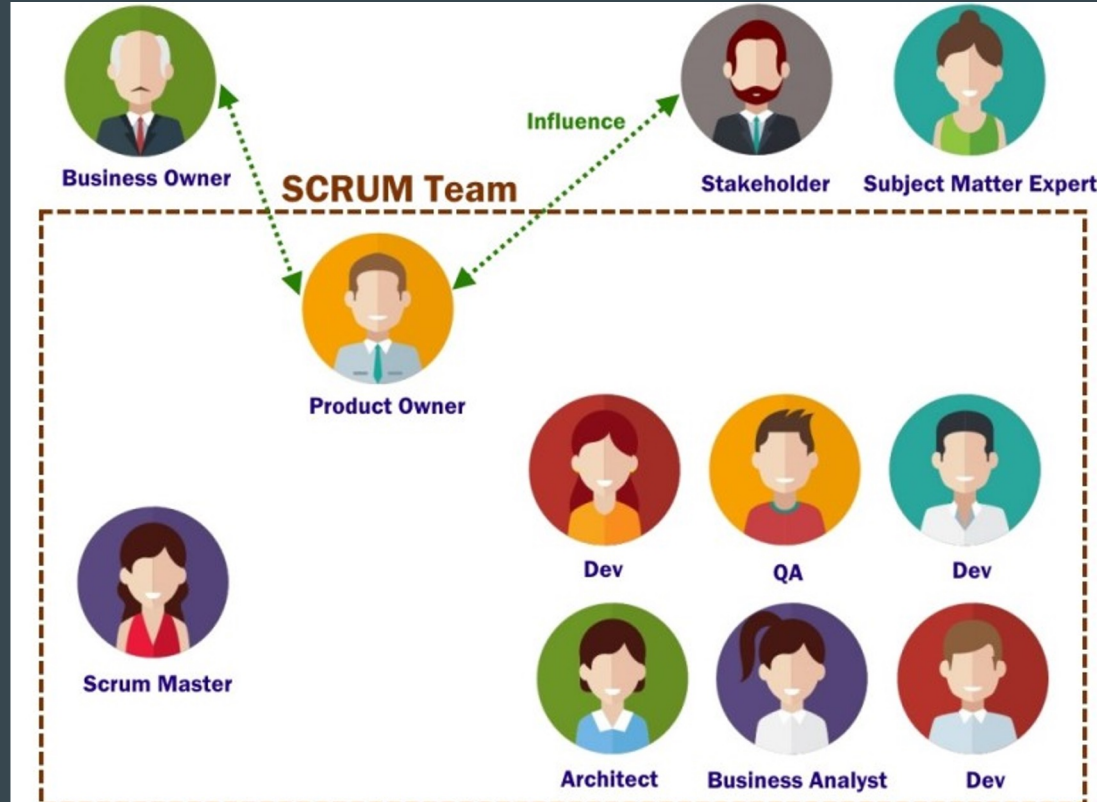
Sprint  
Backlog

Sprint end date and  
team deliverable  
do not change










# Key concepts and terms of Agile/Scrum

- 3 Roles:
  - Product owner
  - Scrum master
  - Development team
- Sprints:
  - Spring planning
  - Sprint demos
  - Sprint retrospectives
- Other concepts:
  - Daily stand-ups
  - Backlogs
  - User stories
  - Kanban board



# Example of a Kanban Board




Backlog	In Progress (3)	Peer Review (3)	In Test (1)	Done	Blocked
					
<b>Fast Track/ Defect</b>					









As a <young Jedi>

I want to <use The Force>

So that <I can lift my X-wing  
from the swamp>

Acceptance criteria:

<I can fly away from this  
planet>



# Exercise 1: Writing User Stories





# Intro Task: Lending Club personal loans

- Q: Do those who have ever failed to pay incur higher interest rates?
- Worked example



# Breakout Rooms for Kanban Boards

1. Open this [Google Doc](#) and find your breakout room
2. Write **at least 3 user stories** for various steps to answer the question
3. Use the phrasing: “As a \_\_\_\_\_, I want \_\_\_\_\_, so that \_\_\_\_\_”
4. Specify the Acceptance Criteria for each task

## Some ideas:

- a. Exploratory Data Analysis?
- b. Modeling?
- c. Model validation?
- d. Interpret real-world meaning?
- e. How will the pieces fit together?



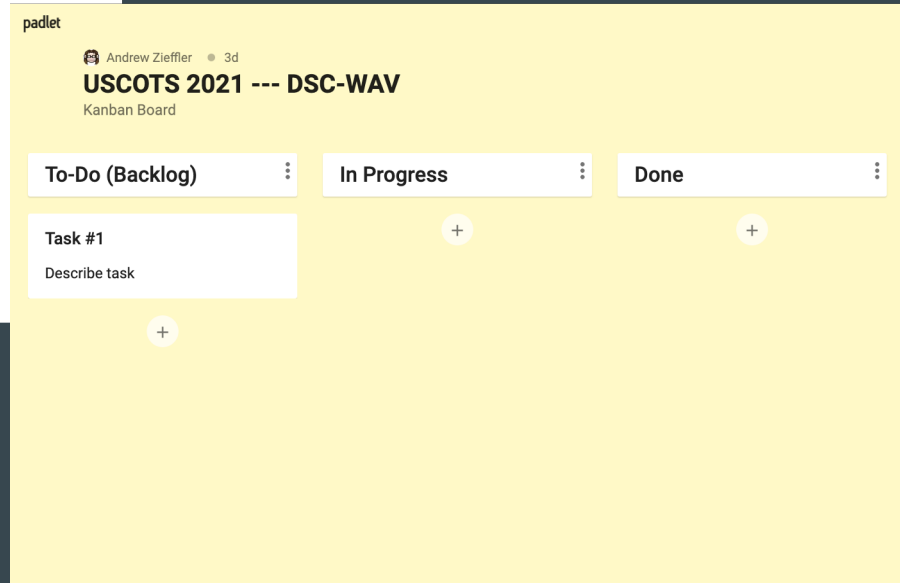
# Exercise 2: Populating and Prioritizing a Kanban Board (using [GitHub Projects](#))



# Alternative Kanban board implementations



[Jamboard \(Google Apps\)](#)



[Padlet \(https://padlet.com\)](https://padlet.com)

Also, [Trello](#)



# Code Review

# What is code review?

- Systematic quality assurance for code
- Dates back to 1976 (Fagan)
- Performed by someone else
- Best practice for improving code quality

```
    }).done(function(response) {
      for (var i = 0; i < response.length; i++) {
        var layer = L.marker(
          [response[i].latitude, response[i].longitude]
          // ,{icon: myIcon}
        );
        layer.addTo(group);

        layer.bindPopup(
          "<p>" + "Species: " + response[i].species + "<br>" +
          "<p>" + "Description: " + response[i].description + "<br>" +
          "<p>" + "Seen at: " + response[i].latitude + " " + response[i].longitude + "<br>" +
          "<p>" + "On: " + response[i].sighted_at + "</p>"
        );
      }

      $('select').change(function() {
        species = this.value;
      });
    });
  });
$.ajax({
  url: queryURL,
  method: "GET"
}).done(function(response) {
  for (var i = 0; i < response.length; i++) {
    var layer = L.marker(
      [response[i].latitude, response[i].longitude]
      // ,{icon: myIcon}
    );
    layer.addTo(group);
  }
});
});
```



# Why do code review?

- Better code quality
  - readability
  - consistency
  - understandability
- Fewer bugs
- Knowledge transfer
- Increased sense of mutual responsibility
- This isn't *your* code, this is *our* code



# When to review code?

- Code reviews should be short!
- Best practices from [2006 Cisco study](#):
  - Lines of code (LOC) under review should be **less than 200**, not exceeding 400, as anything larger overwhelms reviewers and they stop uncovering defects.
  - The total review time should be **less than 60 minutes**, not to exceed 90. Defect detection rates plummet after 90 minutes of reviewing.
- Pull requests should be small!
- Dovetails with our recommendation for more frequent, smaller commits





# How to review code?

- Does this code accomplish the author's purpose?
- Think like an adversary, but be nice about it.
- Think about libraries or existing product code.
- Does the change follow standard patterns?
- Think about your reading experience.
- Does the code adhere to coding guidelines and code style?
- More:
  - [Google Engineering Practices code review documentation](#)
  - [Code Review Best Practices](#)



# Exercise 3: Code Review (Time permitting)



# In practice

- Send a pull request (PR)
  - For R code, run styler beforehand!
  - Tag someone else as a reviewer
- If you're tagged, review the PR
  - Make line-by-line comments, questions, etc.
  - Approve or request changes
- Cycle continues until the PR is merged
  - It's not rude to request changes!!
- This isn't *your* code, this is *our* code



# Examples of code review in the wild

- <https://github.com/rudeboybert/fivethirtyeight/pull/72>
- <https://github.com/rudeboybert/fivethirtyeight/pull/68>
- <https://github.com/rudeboybert/fivethirtyeight/pull/67>
- <https://github.com/rudeboybert/fivethirtyeight/pull/66>
- <https://github.com/beanumber/macleish/pull/41>



# Project Evaluation



# DSC-WAV Implementation

- Students completed surveys at end of project
- Gave insight into:
  - Implementation of these methods
  - Successes
  - Areas for future work



# Agile Implementation

- Students were able to implement Agile
  - Faculty help was crucial
- Many teams made use of sprints to plan work
- Students recreated a sense of community around the work
- Zoom and Slack helped connect students

*“It felt like a simulation of what industry is like as opposed to another group project for class. We didn’t have a professor assigning us tasks. Rather we as a team had to discuss the best way to move forward with the project.”*



# Use of Kanban Boards

- Most teams used Kanban Boards
- User stories made tasks seem clear
- Students noted this was good tool for project management
- Helped students determine and prioritize tasks





# Code Review

- Varied implementation in each team
  - Partners
  - Scrum master
  - Cyclic pattern
- Typically not formal
- Focus on functionality (more than style)



# Final Thoughts & Future Work

- Helped develop technical and non-technical skills
  - Communication
  - Collaboration
  - Leadership
- Student autonomy in their project direction
  - Particularly if faculty mentor got them started
- Still too much “divide and conquer”?
- More structure needed for code review

Thank you!



Questions?



# Learn more...

Videos from Smith capstone course:

- [User stories](#)
- [Kanban boards](#)
- [Scrum roles](#)
- [Sprint mechanics](#)
- [Code review](#)

DSC-WAV publications:

- [Facilitating Team-Based Data Science: Lessons Learned from the DSC-WAV Project](#), under review for special issue on "Data Science Education Research" in *Foundations of Data Science*, 2021
- [The Data Science Corps Wrangle-Analyze-Visualize program: building data acumen for undergraduate students](#), *Harvard Data Science Review*, 2021

DSC-WAV outreach page: <https://dsc-wav.github.io/www/outreach.html>