Tools for automating project-based data science classes with hundreds of students

Jason Fleischer, PhD
Asst Teaching Prof
Cognitive Science, UC San Diego
https://jgfleischer.com/

Pressure to teach at scale

UC San Diego Fall enrollments:
30k in 2013, 42k in 2023, goal 50k in 2030

Courses in CS / AI / ML / DS are VERY popular across majors

COGS 108 Data Science in Practice is a top 5 most waitlisted class

Typically 400 - 750 students per quarter, 1 TA and 1 undergrad assistant per 100 students

COGS 108
Data Science in Practice

Dual role: A first (or only!) course in data science

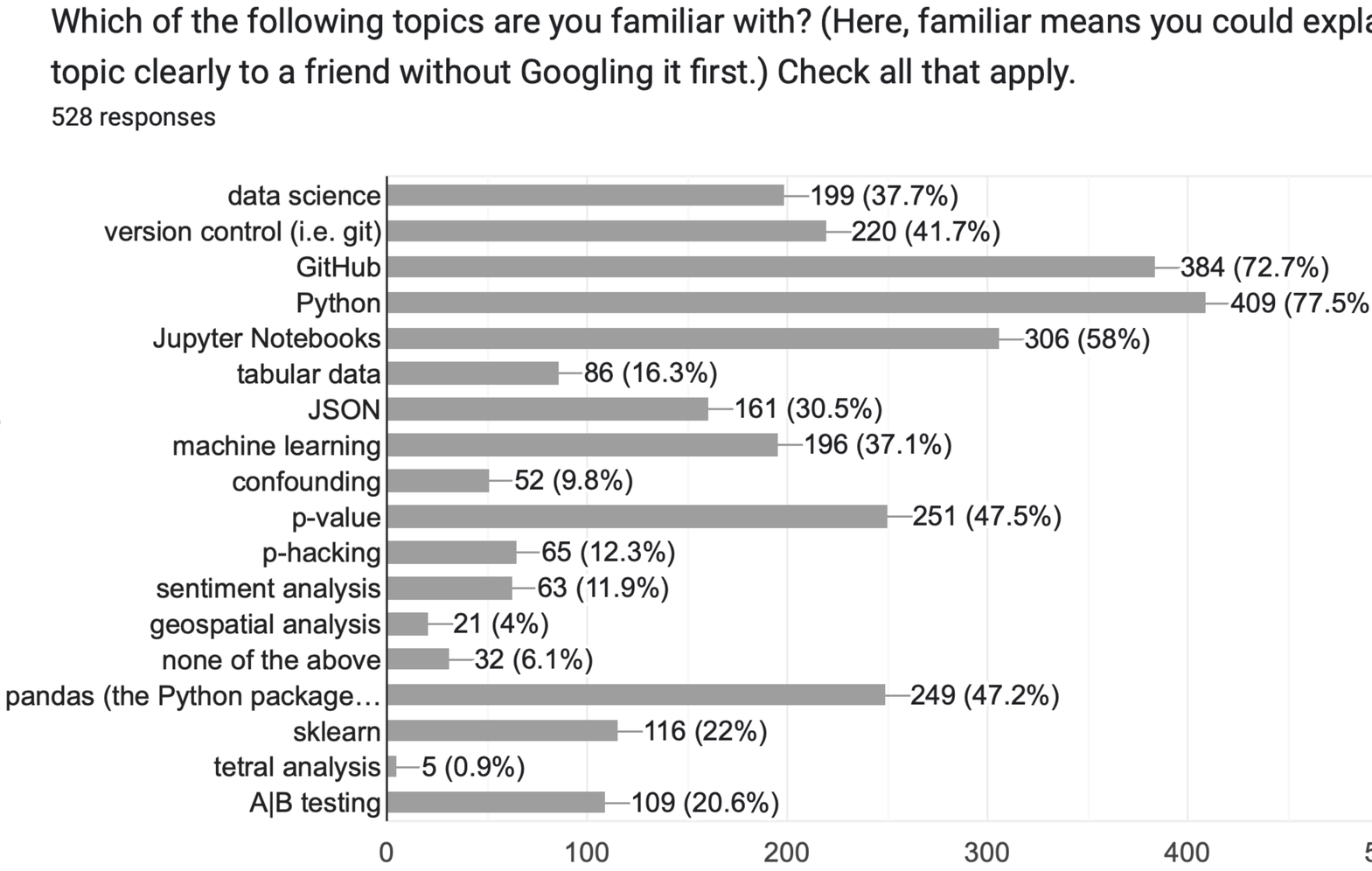
Pre-requisite: some sort of intro programmi

Students come from many STEM majors: CS, COGS, DS, ECON, EE, MATH, etc

Half of students are taking this for a major requirement

All materials open source:
https://github.com/COGS108

Precourse survey of relevant skills



COGS 108 curriculum

Formal lecture coverage is a mile wide and an inch deep

Project based: challenge students to learn how to learn on their own

Week	Topic(s)
1	Data Science Tools
2	Data Wrangling & Intuition
3	Asking Well-Formed Questions & Data Ethics
4	Basic Data Viz & Exploratory Data Analysis
5	Inference
6	Text Analysis
7	Machine Learning
8	Nonparametric Analysis
9	Geospatial Analysis
10	Data Science Communication & Jobs

COGS 108 Data Science in Practice

Tom Donoghue
Postdoc
Biomedical Engineering
Columbia

Shannon Ellis
Assoc Prof
Cognitive Science
UC San Diego

Bradley Voytek
Prof
Cognitive Science
UC San Diego

Scott (Yuanjia) Yang
MS student
Computer Science
UC San Diego

COGS 108 class structure

	Frequent, low stakes	Less frequent, high stakes
Individual	Weekly labs and quizzes	Biweekly assignments
Group project	Biweekly checkpoints	Final project

Group project

- Groups of 4 or 5 students, very open ended, last about 8 weeks

- Frequent checkpoints to keep students on track! Points lost on one checkpoint can be regained by fixing the problem at the next one

    1. Review and rate previous projects

    2. Define a question that is answerable by data analysis with an appropriate scope
       Too vague: “Who does well at university?”
       More like it: “For students in COGS 108 from 2022 - 2024, does financial aid status or first gen status predict final course grade when controlled for major?”

    3. Identify, obtain, and wrangle relevant data sources

    4. Exploratory data analysis

    5. Final project

Examples of projects

Modeling the pricing of airbnb rental listings in San Diego
The demographics of Adderall usage
Prediction of musical genre from musical features
Analysis and Prediction on California Wildfire and Climate Variables
Bachelor's Degrees & Career Paths within the Context of COVID-19
Access to Fast Food and Lifestyle Diseases
Analysis of Previous COGS 108 Projects

Lessons Learned from 1,000 Data Science Projects

on right now in session Building Better Data Analyses: Theory, Methods, and Lessons Learned

---

The process of Building Better Data Analyses often begins in the classroom, and increasingly, due to growing enrollments, this is happening at scale. Since 2018, we have taught COGS 108 Data Science in Practice at UC San Diego every single term to 400+ students at a time. This large-enrollment, project-based course aims to teach the critical skills needed to pursue a technical data-focused career. Throughout this course, students complete a term-long group data science project on a topic of the students' choosing. Groups carry out the entire data science process: formulating a question; finding, cleaning and analyzing data; answering their question of interest; and finally, communicating their process and findings in both a detailed, technical data science report and short, oral presentation. Having advised 4,000 students through more than 1,000 projects, we summarize the key lessons we've learned in how to teach using and analyzing data at scale. Our findings highlight the importance of clear instruction, project scaffolding, regular checkpoints, detailed & project-specific feedback, and careful consideration of the technical stack used.

.

Shannon Ellis

Assignments and Labs

- Individual completion
- Lab exercises bring up a topic with low stakes. Its ok if they talk with each other or seek extensive help from a TA.
- 1hr long discussion sections are limited to 60 students at a time size, offer focussed individual help on labs
- Assignments are worth 4x the points of labs. They are intended to demonstrate mastery, students are told not to talk with each other. TA help is available but limited in scope
- Scaling class size via **autograding**!

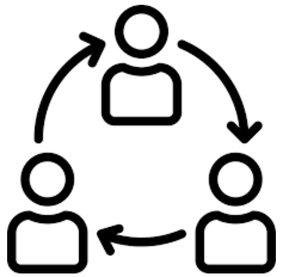
Toolchain and workflow for students

Canvas

A common LMS: Calendar, messaging, online quizzes and discussions, grade management, etc.. **Has an API**

Git

Currently the most common version control tool. Decentralized, collaborative management of a code base



GitHub

Cloud based Git repos. Tools to manage software teams. Home for open source software. Functions as programmer's social media and portfolio. **Has an API**

Git and GitHub for projects

- We create a group repo with templates and manage access control
- Encourage them to use git/GitHub tools for collaboration and version control
- Instruction team uses GitHub Issues to give feedback on checkpoints and answer questions
  - Example: https://github.com/COGS108/Group53_SP24/issues/

Toolchain and workflow for students

JupyterHub

JupyterHub is the best way to serve Jupyter notebook for multiple users. Because JupyterHub manages a separate Jupyter environment for each user, it can be used in a class of students, a corporate data science group, or a scientific research group. It is a multi-user **Hub** that spawns, manages, and proxies multiple instances of the single-user Jupyter notebook server.

NBGrader

`nbgrader` is a tool that facilitates creating and grading assignments in the Jupyter notebook. It allows instructors to easily create notebook-based assignments that include both coding exercises and written free-responses. `nbgrader` then also provides a streamlined interface for quickly grading completed assignments. It is fully integrated with JupyterHub for distributing and collecting student assignments

jupyterhub

Logout | Control Panel

Files | Running | Clusters | Formgrader | Assignments | Courses

Released, downloaded, and submitted assignments for course: COGS108_SP24_A00

Released assignments

A1_COGS108_SP24	COGS108_SP24_A00	Fetch
A2_COGS108_SP24	COGS108_SP24_A00	Fetch
A3_COGS108_SP24	COGS108_SP24_A00	Fetch
A4_COGS108_SP24	COGS108_SP24_A00	Fetch
D1_COGS108_SP24	COGS108_SP24_A00	Fetch
D2_COGS108_Fa22	COGS108_SP24_A00	Fetch
D4_COGS108_SP24	COGS108_SP24_A00	Fetch
D5_COGS108_SP24	COGS108_SP24_A00	Fetch
D6_COGS108_SP24	COGS108_SP24_A00	Fetch
D7_COGS108_SP24	COGS108_SP24_A00	Fetch
D8_COGS108_SP24	COGS108_SP24_A00	Fetch
Practice_COGS108_SP24	COGS108_SP24_A00	Fetch
PythonReview_COGS108_SP24	COGS108_SP24_A00	Fetch

Downloaded assignments

D3_COGS108_SP24	COGS108_SP24_A00	Submit

Submitted assignments

There are no submitted assignments.

Student view of an autograded assignment

jupyterhub D3_dataviz (autosaved) Logout Control Panel

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Workbook : EDA & Data Visualization

We'll continue working with the dataset from the Wrangling workbook here to answer the questions we set out to answer previously:

1. Who cheats more on their significant other - males or females?
2. Are cigarette smokers less likely to skydive?
3. Do people in New England gamble more than other parts of the country?

To do this, we'll explore the data and generate a number of visualizations. Note that we don't have `assert` statements when it comes to visualizations here. Instead, we have hidden test cells. Please do not modify the cells as they are necessary to reward points for your graphs!

Import the following packages using their common shortened name found in parentheses:

- `numpy` (`np`)
- `pandas` (`pd`)
- `matplotlib.pyplot` (`plt`)
- `seaborn` (`sns`)

In [ ]:
```
# YOUR CODE HERE
raise NotImplementedError()
```

In [ ]:
```
assert np
assert pd
assert plt
assert sns
```

Run the following cell code to make things throughout the rest of this workbook a little prettier. (Note: You don't have to edit code here, but are free to and see what changes to be sure you understand each line.)

In [ ]:
```
# Configure libraries
# The seaborn library makes plots look nicer
sns.set(context = 'talk', style='white')

# Don't display too many rows/cols of DataFrames
pd.options.display.max_rows = 7
```

Creating autograded assignments

ID: cell-fa2a938005a3c187 Read-only

1a) Load the data

Import datafile `players.csv` into a DataFrame called `df`.

Note these data come from SOFIFA. Similar data have been scraped and shared on Kaggle; however the data we'll use here is distinct from that dataset.

ID: cell-252b5cfd5c8ceff2 Autograded answer

```
### BEGIN SOLUTION
df = pd.read_csv('players.csv')
### END SOLUTION
```

Points: 0.1 ID: 1a_ans Autograder tests

```
assert isinstance(df, pd.DataFrame)
assert df.shape == (18278, 15)
```

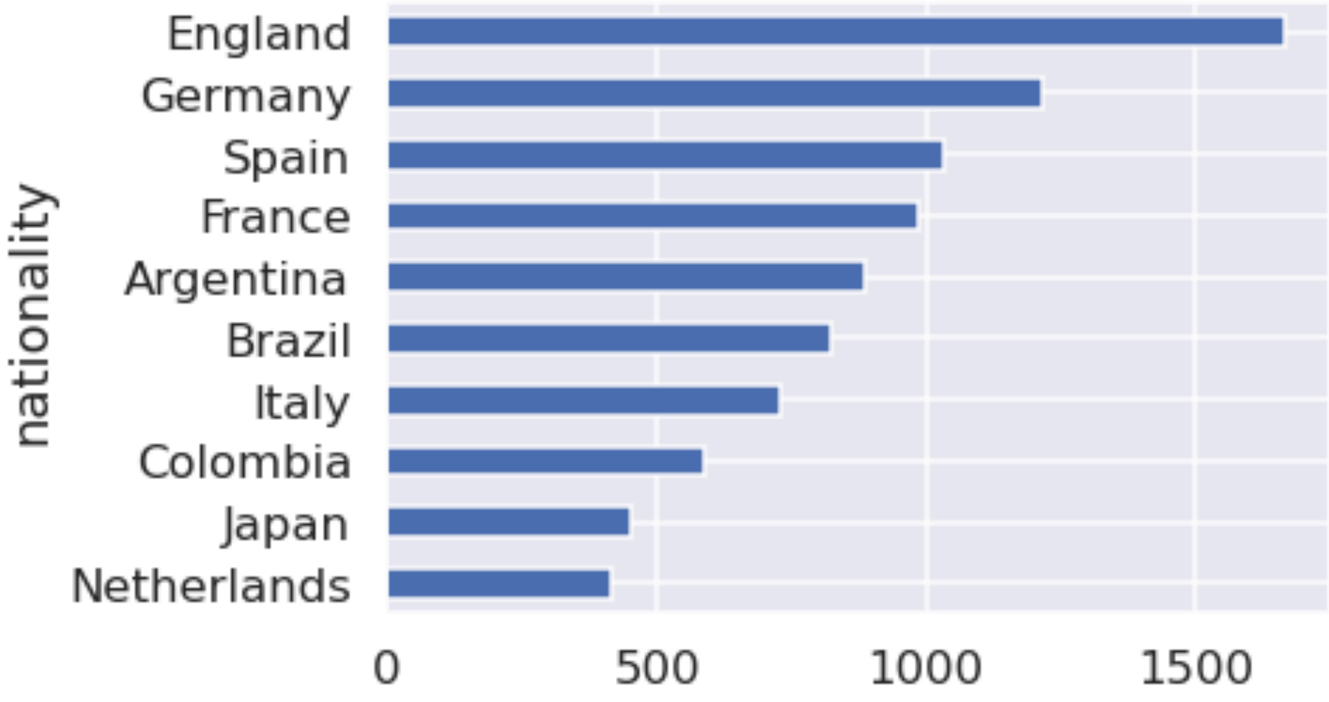
Creating autograded assignments

In [18]:

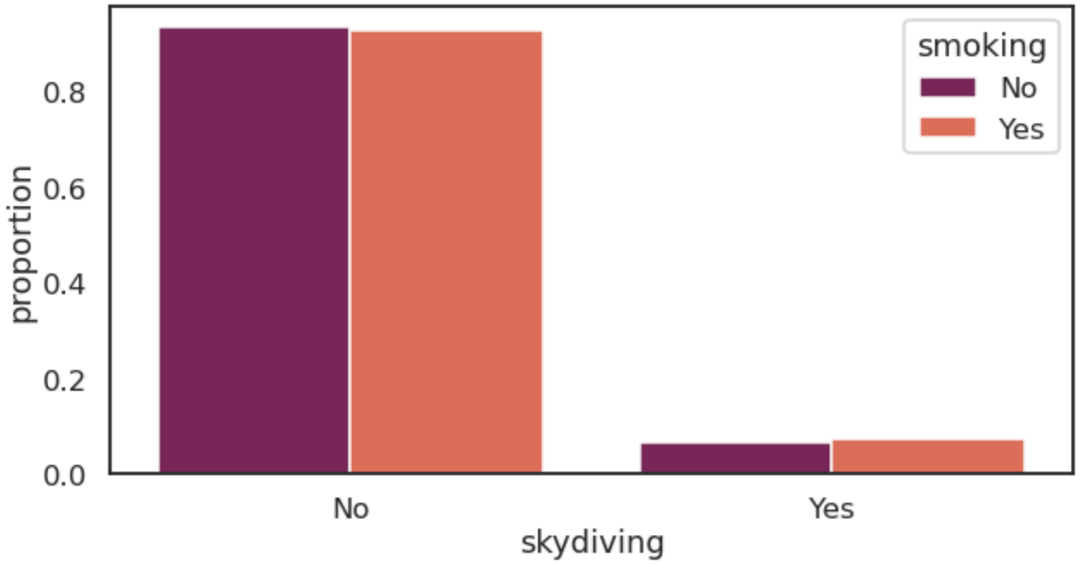
Points: 0.2 ID: cell-f38279d756a7a299 Manually graded answer

```
### BEGIN SOLUTION
# seaborn
# top_10 = df['nationality'].value_counts()[:10].index.tolist()
# df_sub = df[df['nationality'].isin(top_10)]
# sns.countplot(y=df_sub['nationality'])

# pandas
df['nationality'].value_counts().nlargest(10).sort_values().plot(kind =
### END SOLUTION
```



Creating autograded assignments



In [34]:

Points: 0.25 ID: cell-04f94ab3c95ced7b Autograder tests

```
# TESTING CELL, PLEASE DO NOT EDIT
### BEGIN HIDDEN TESTS

from plotchecker import BarPlotChecker

answer_array = [0.93, 0.07, 0.93, 0.07]
pc = BarPlotChecker(plot_skydiving)

# To check the orientation of their barplot, either vertical or horizontal
if [round(item,2) for item in pc.widths].count(pc.widths[0]) == len(pc.widths):
    graph_vals = pc.heights
else:
    graph_vals = pc.widths

assert len(graph_vals) == len(answer_array)

for val in [round(item,2) for item in graph_vals]:
    if val not in answer_array:
        assert 0, "Incorrect values in the bar plot"


### END HIDDEN TESTS
```

Creating autograded assignments

3b) Weight

Extract the weight data for all French players, storing it in `w_france`.

Extract the weight data for all Brazilian players, storing it in `w_brazil`.

In [23]:
```
w_france = df[df['nationality'] == 'France']['weight']
w_brazil = df[df['nationality'] == 'Brazil']['weight']
```

3c) Difference in weight?

Carry out a t-test (using the `ttest_ind()` function) to compare the two distributions. Store the test statstic in the variable `t_val` and the p-value in the variable `p_val`.

In [24]:
ID: cell-8a7bb7133fdd9d77 — Autograded answer
```
### BEGIN SOLUTION
t_val, p_val = ttest_ind(w_france, w_brazil)
### END SOLUTION
```

In [25]:
```
# look at results
t_val, p_val
```

Out[25]:
```
(-1.7586424201253648, 0.07880760827880617)
```

In [26]:
Points: 0.2 — ID: cell-5cd84d5270b2fb06 — Autograder tests
```
assert np.isclose(t_val, -1.76, atol = 0.01)
assert np.isclose(p_val, 0.079, atol = 0.01)
```

Forming student project groups

- Students dislike assigned groups! But they may be good for them. Anecdotally…
  - More professionalism when groupmates are not your friends (Oakley, et al., 2004)
  - Possible to address inequalities with assigned group: when women and members of URM are isolated as solo representatives of their demographic, they often end up in passive roles within the group, losing many of the benefits of interactive group-based learning (Widnall 1988; Heller and Hollabaugh 1992). Clustering two or more of these students per group is the recommended strategy to avoid this isolation (Rosser 1998)
  - There may be benefits to skill or major heterogeneity (Bekele, 2005)

* Github Classroom does but it appears limited to roster import

GroupEng
T.G. Dimiduk & K.C. Dimiduk

GroupEng was written specifically to help instructors follow education research guidelines for improving student, and especially minority, performance in group project based classwork. It allows instructors to think at an abstract level about what properties they want student groups to have; GroupEng handles the details of optimizing the groups to meet those criteria.

GroupEng forms groups of students based on grouping rules supplied by the instructor.

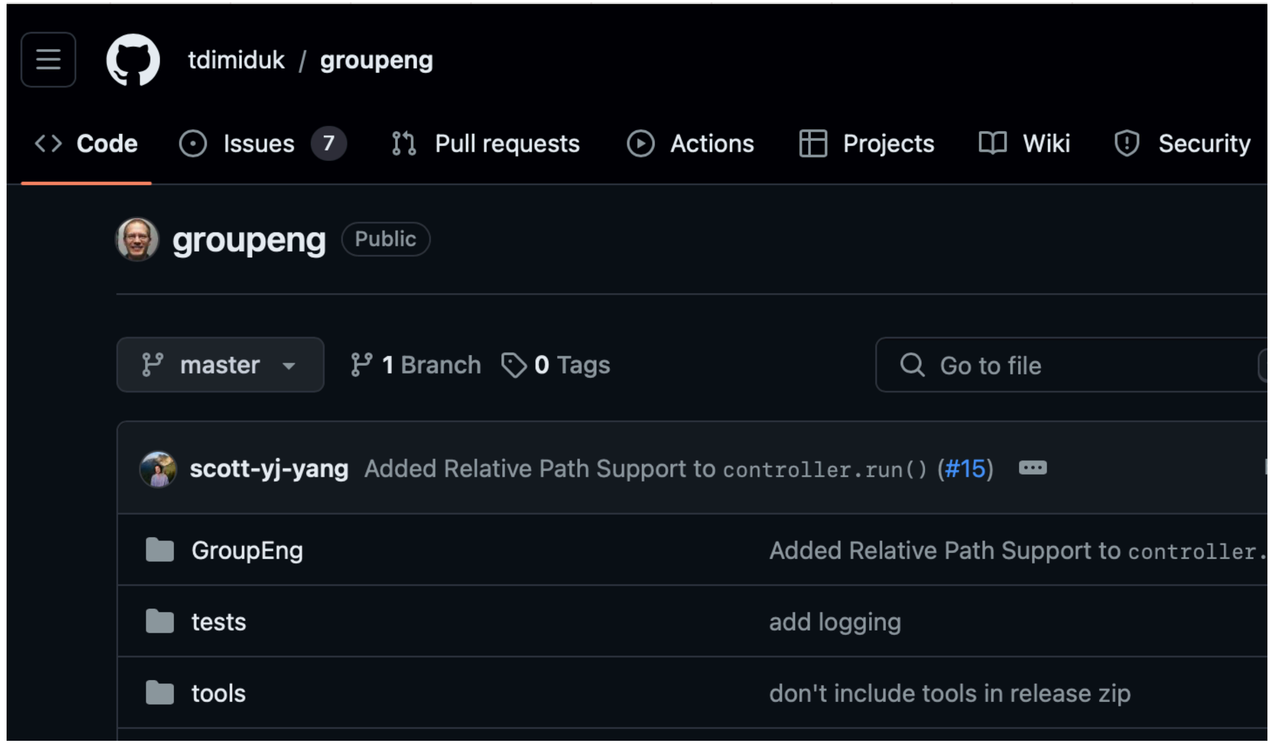
Distribute: Spread students with some attribute (major, a needed skill) across all groups so that each group has about the same number of them

Aggregate: Group students with some attribute (project choice, section number) together in the same groups

Cluster: Ensure that students with some attribute (women, minorities) are not isolated in a group (there will be at least 2 of them in a group)

Balance: Ensure equal strength of groups based on some numeric score (GPA, pretest).

You specify class data, desired group size, and an ordered list of grouping rules (earlier rules are given higher priority) in a simple text file, and GroupEng does the rest, outputting a grouping of the students, as well as statistics of how well it was able to meet your rules. GroupEng has a random element, so different runs with the same rules will generate different groups.



Creating and managing student groups with Canvas and GitHub

Canvas has group management abilities…

😀 can form groups (randomly, self-assign, manually, or via API)

😀 group communication

😢 but no ability to systematically manage group composition and no connection to GitHub

GitHub has group management abilities…

😀 great for access control (manually or via API) and therefore forming groups

😀 better group communication

😢 but no ability to systematically manage group composition and no connection to Canvas*

* Github Classroom does but it appears limited to roster import

https://fleischerresearchlab.github.io/CanvasGroupy/

CanvasGroupy

CanvasGroupy

Canvas Grouping Python Script

View our documentation at this link

This module will use `GroupEng` to create canvas and github groups.

Install

```
pip install CanvasGroupy
```

How to use

Please visit GroupEng Official Website to see the documnetation of how to use GroupEng.

```
assign_groups("example/sample_group_specification.groupeng")
```

```
['-', '-', 'B', '-', '-']
['B', '-', 'H', '-', '-']
['H', '-', '-', '-', '-']
['-', 'H', '-', 'B', 'H']
['-', '-', '-', 'B', '-']
['-', '-', 'H', '-', '-']
['nanotech', 'renewable energy', 'nanotech', 'nanotech', 'nanote
['automotive', 'automotive', 'robotics', 'automotive', 'renewable
```


Report an issue

Scott (Yuanjia) Yang
MS student
Computer Science
UC San Diego

Using these tools for a pedagogical experiment in progress

- First large scale RCT on this topic?
- >1200 students enrolled in study over 2 quarters, 2 instructors
- Balanced the number of women in each condition:
  Groups of 5 people with 0 - 5 females
- Outcome variables:
  - Drop rates
  - Measures of contribution via Github
  - Survey instrument assessing
    roles of self and others, team dynamics, individual roles, satisfaction with team and end product

Using these tools for a pedagogical experiment in progress

- No evidence so far for any differences in drop rate
- Evidence for gender based differences in performance that change with gender purity of the group
  - Github contributions
  - Roles in the group

EARLY RESULTS

JUST A TEASER

➡ SOON https://fleischerresearchlab.github.io/ ?????

NeverLate [unreleased]

Imports grades from autograder service to Canvas Gradebook. During import it automatically applies appropriate late penalty once students exceed allowed slip days

v.01 works with JupyterHub/NBGrader
v.02 works with Gradescope

need to generalize and package this!

Scott (Yuanjia) Yang
MS student
Computer Science
UC San Diego

Lessons learned

- Introductory students need a preconfigured cloud based programming env
  Alternative: prerequisite intro to tools for env management and version control

- Manage environments so that the student's working environment and your autograder environment are identical (Docker image / env.yml / requirements.txt)

- Happy to talk about the particular limitations of JupyterHub and NBGrader

- All tools have problems, changing tools just changes problems

Lessons learned

Teaching at scale requires you to think like an executive

- Most student contact hours are not with you. You train and manage the TAs, they do most of the talking with students
- Set clear expectations, have well-documented procedures
- Encourage a culture of
  - Ownership and continuous improvement
  - Communication and teamwork
- **Fundamental limit on scaling:** 10% of students are high maintenance. Problems are your responsibility.

Challenges for the future

- Teaching programming in a world of AI
    - and maybe soon math and problem solving skills
- Continuous updating of tools

No support at your university for JupyterHub?

- Free solution: Google Colab
  well integrated with GitHub
  😢 No NBGrader integration…
    ⌥1 Gradescope to autograde (provide a Docker image)
    ⌥2 Students fork/submit assgns on GitHub and you use NBGrader locally
    ⚠️ Problem with both⌥s: library version consistency with Colab

- Paid solutions:

  - GitHub Classroom + GitHub Spaces (provide a Docker image)

  - EdSTEM (provide a Docker image)