The Use of Computer Programming in a Secondary Mathematics Class

JaCoya Thompson, Northwestern University

JaCoya Thompson is a PhD student in the Computer Science Department at Northwestern University. She is a research assistant on the Computational Thinking in STEM project. Her research interests include the use of computational tools to support novice learner's exploration of mathematics concepts in formal learning environments.

Dr. Sally Wu, Northwestern University

Sally P.W. Wu is the Director of Curriculum Development at Northwestern University. Her work on the Computational Thinking in STEM project focuses on helping high school teachers integrate computer science principles into their mathematics or science classrooms. She received her PhD from University of Wisconsin-Madison in Educational Psychology-Learning Sciences. Her research broadly examines how to help students learn complex visual-spatial content in introductory STEM courses through the design of technology-enhanced interventions for the classroom. Her work thus far has investigated the effects of drawing, collaboration, reflection, and other active learning strategies in undergraduate chemistry and electrical engineering.

Jacob Mills, Evanston Township High School

Jacob Mills is a mathematics teacher at Evanston Township High School. He participated in Computational Thinking in STEM project summer institute and codesigning a Python-infused lesson that he implemented in his mathematics class.

Introduction

The National Council of Teachers of Mathematics [24] Technology Principle states that "[t]echnology is essential in teaching and learning mathematics; it influences the mathematics that is taught and enhances student learning'' (p. 24). Indeed, prior research suggests that integrating appropriate forms of technology into mathematics classrooms supports students' learning experiences [9, 11, 24], enhances student attitudes towards the subject of mathematics [2, 9] and brings real-world problems into the classroom for students to explore and solve [18, 29]. This can be done by creating an environment that promotes understanding, rather than memorization, through visualizations [5, 24], automated calculations [15, 24] and providing students the opportunity to explore various questions such as "what happens if...?" [5].

In this study, we investigate the effects of integrating a technology used by professionals in an Advanced Placement (AP) Statistics course, which is designed to provide students with college-level work while still in high school [10]. Technologies in AP Statistics course typically involve (1) spreadsheets for analyzing data and constructing visual representations of data; (2) multimedia materials to teach, tutor, and/or test students' statistical knowledge and skills; (3) web or computer-based tools/simulations to demonstrate and visualize statistical content; and (4) graphing calculators for computation, graphing, or simulation. These technologies have some affordances in helping students engage with content. However, they usually have a narrow functionality which reduces opportunities for open-ended exploration and statistical power. Several statistical software programs have been developed exclusively for helping K-12 students explore statistics, such as Fathom [8], TinkerPlots [19], and CODAP [31]. These educational programs are designed to be less overwhelming to students by utilizing a point-and-click, drag-and-drop functionality [5, 23].

Similar to the ways in which text-based programming allows professionals to analyze open-ended problems, text-based programming may also give students new ways to express mathematical ideas [7, 26, 30]. However, K-12 students rarely use text-based programming software such as R and Python in formal classrooms. Much of the studies about how students explore statistics using text-based programming software focus on introductory statistics and data science courses at the undergraduate level, for students who pursue mathematics or computer science degrees [3,16, 28]. There is little research on the role that text-based programming can play in the teaching of high school AP Statistics, even though the AP Statistics course equates to a semester introductory college course [10]. The integration of programming into AP Statistics can help students who may not pursue computer science develop computational thinking skills to solve real-world problems, thereby broadening participation in computer sciences while students develop mathematics skills. Prior work shows that text-based programming helps students gain familiarity with appropriate graphical and numerical tools for data representation and explore how to interpret and communicate data [20].

To better identify the effects of integrating text-based programming in the AP Statistics classroom, we investigated how students used Python as a tool to understand mathematics content. Python is a text-based open source programming software, with an extensive list of well-

documented packages for use in a broad variety of data-analysis activities [33]. In addition, it is a suitable beginning language for students due to the readability of its syntax [25]. We expect that the use of Python can support students understanding of mathematics content and create an environment where students are excited to engage with mathematics as shown by research on block-based programming [12, 13, 14]. However, we do not know how students will choose to engage in a text-based programming language such as Python. Text-based programming can change the atmosphere of a formal mathematics classroom from one where students are not accustomed to making mistakes, to one in which mistakes lead to reflection and investigation through debugging code [22, 29]. While studies have explored learning using text-based programming in an informal setting [6], few studies have addressed students' understanding of mathematics content when using text-based programming in formal classrooms. Further, little work has focused on students' perceptions of a text-based programming for learning and the impact these perceptions have on their learning [34]. To address these gaps in prior work, we investigate whether students can understand targeted mathematics content using a text-based programming language (research question 1) and how students perceive text-based programming as a tool to learn mathematics content (research question 2).

Methods

In this study, a high school statistics teacher and computer education researcher built a Python-integrated AP statistics unit together over the summer. The teacher identified a lesson on descriptive statistics that could be tailored to coding activities. The teacher showed a willingness to learn coding and was comfortable with debugging when a code error occurred. During the fall, the teacher implemented the unit in two class periods over two weeks. A total of 53 students participated. 39% of students were female (21 female, 30 male, and 2 non-binary) and 22% of students identified as either African American or Hispanic (9 Black, 3 Hispanic, 37 White, 3 Asian, 1 Other). During each 42-minute class period, each student used their own laptop and worked in groups of 3-4.

Because many students in the high school classroom do not have the prerequisite skills in text-based programming, we scaffolded students' use of Python in a series of lessons using the Python libraries, Pandas [21] and Matplotlib [17], and Jupyter notebooks [32]. The Jupyter notebooks for each lesson served as an interface to modify, debug, and re-execute code on a given data set. We structured the notebooks so that the set-up time between teacher and student would be minimal to support a smooth transition into Python. The notebooks provided comments embedded within code and carefully sequenced problems to scaffold students through each lesson. The lessons that used Python built off of one another, allowing students to gain more confidence with Python and fluency with the mathematics content over time.

The unit focused on descriptive statistics, measures of central tendency, and measures of variability. These topics are from the *Exploring One-Variable Data Unit 1* skills in the College Board AP Statistics standards [1]. The unit consists of five lessons, which are detailed below.

Lesson 1: Students received a dataset of gas prices at various gas stations. Students used calculators and their prior knowledge of descriptive statistics to make decisions about the amount of money they should budget for gasoline. Students were then prompted to explain the reasoning for their budget using the data set.

Lesson 2: Students begin to familiarize themselves with Jupyter notebook and various Python commands/functions. They receive a dataset and run Python code in the notebooks (Figure 1) to generate tables and plots (Figure 2). First, students generate a descriptive table and interpret the statistical values (mean, standard deviation, and quartiles). Then, students generate a boxplot and identify what the length of the "box"" shows as well as what descriptive statistics are not shown on the boxplot. Lastly, they plot a histogram, describe the distribution (shape, center, spread, outliers), and identify the descriptive statistics not shown on the histogram. In order to interpret data, students manipulated the data in their tables and plots by changing the quartile labels (25%, 50%, and 75%) in the descriptive table to the names of those quartiles (lower, median, upper). To further familiarize themselves with the code, students changed the color and bin count of the histogram and adjusted the x-and-y limits on the plots. Students submitted all tables and plots.

Lesson 3: Students engaged in an unplugged activity to learn about the formula for sample standard deviation by writing a "pseudo code" or procedure for calculating sample standard deviation from a given dataset. Students test their pseudocode and compare their output to the output of the graphing calculator. If the outputs do not match, they must "debug their code" until it matches the calculator's output. After writing the correct code, students answered some follow-up questions about standard deviation.

Lesson 4: Students analyze one of five datasets that may relate to their lives such as number of Instagram followers, daily hours spent on cell phone, and hours spent studying. In this lesson, only part of the code is written (see Figure 3). Students have to use code from lesson 2 and debug the code throughout the notebook for results. Students first generate descriptive tables and make a prediction of the histogram shape using the mean and median from the tables. Students then plot the histogram, adjust bin widths, describe various distributions, and explain which bin width best fit the dataset and why. Finally, students produce a boxplot. They interpret the descriptive statistics shown on a boxplot versus a histogram and discuss the pros and cons of boxplots. Students submitted all tables and plots.

Lesson 5: Before this lesson, students from both class periods completed a survey, including questions about their height, Instagram followers, daily hours spent on cell phone, and hours spent studying. The teacher compiled datasets by period and asked students to make comparisons between the two class periods. As in lesson 4, students have to complete and debug the code (Figure 4). Students plotted descriptive tables, histograms, and boxplots to compare the two class periods. Then, they interpreted the standard deviations and distributions (shape, center, spread, outliers) of the two class periods. Finally, students removed values in both of the datasets that are closest and farthest from the mean. After each change, they rerun the code to produce

descriptive tables, to help them observe and interpret changes to standard deviation as well as how standard deviation relates to the mean. Students submitted all tables and plots.


```
File    Edit    View    Insert    Cell    Kernel    Widgets    Help                    Trusted    Python 3  O

In [ ]:   1 import numpy
          2 import pandas as pd
          3 import matplotlib
          4 import matplotlib.pyplot as plt

In [ ]:   1 import numpy as np
          2 ds = pd.read_csv('gas_prices.csv') #read in data set from a csv file
```

Question 2.1 Display a descriptive table

```
In [ ]:   1 #get descriptive about data set
          2 ds.describe()
```

Question 3.1 Display a descriptive table only with information for counts, mean, and standard deviation for the data set

```
In [ ]:   1 ds.describe().loc[['','','']] #insert names between quotation marks
```

Question 4.1 Add the correct quartiles (Q1,Median,Q3) labels to 25%, 50%, 75% listed in the table

```
In [ ]:   1 print("",ds.describe().loc[['25%']] ) #insert the quartile that represents 25% between

In [ ]:   1 print("",ds.describe().loc[['50%']] ) #insert the quartile that represets 50% between t
          2

In [ ]:   1 print("",ds.describe().loc[['75%']] ) #insert the quartile that represents 75% between
```

Question 5.1 Plot a box-plot of the data set

```
In [ ]:   1 import seaborn as sns
          2 sns.set(style="whitegrid")
          3 ax = sns.boxplot(x=ds[""]) # insert column name between quotation marks, run code to pl
          4 plt.show()
```

Question 6.1 Plot a histogram of the data set

```
In [ ]:   1 ds.hist(column="",           # insert the column name to plot between red quotation "" mar
          2             figsize=(8,8),          # Plot size
          3             color="",               # insert color name between red quotation "" marks
          4             bins=4 )                # number of bins
          5 plt.show() #run code to plot histogram
```

Figure 1. Python code embedded in Jupyter Notebook, provided to students in lesson 2.

	Year	Average US Gas Price
count	28.000000	28.000000
mean	2004.500000	2.077857
std	8.225975	0.888000
min	1991.000000	1.060000
25%	1997.750000	1.215000
50%	2004.500000	2.015000
75%	2011.250000	2.790000
max	2018.000000	3.640000

Figure 2: Example of tables and plots submitted after running the python code in lesson 2.

```
In [ ]:   1 import numpy as np
          2 import pandas as pd
          3 import matplotlib
          4 import matplotlib.pyplot as plt
```

```
In [ ]:   1 ds = pd.read_csv('')
```

```
In [ ]:   1 #show data in data set
          2 ds
```

Question 1.2 Plot Descriptive Table

```
In [ ]:   1 #complete code to return descriptive table about data set (count, mean, std, etc.)
          2 ds.
          3
```

Question 2.1 Plot Histogram

```
In [ ]:   1 #complete code to plot a histogram of the column of your choosing from the dataset
          2 ds.hist(column="",        # Column to plot
          3             figsize=(8,8),        # Plot size
          4             color="",        # Plot color
          5             bins=8 )        # number of bins
          6 plt.show()
```

Question 3.1 Plot Histogram with bin count less than 8

```
In [ ]:   1 #complete code to plot a histogram and changing bin count to a number less than 8
          2 #for the column of your choosing from the dataset
          3
          4 ds.hist(column="",        # Column to plot
          5             figsize=(8,8),        # Plot size
          6             color="green",        # Plot color
          7             bins= )        # number of bins
          8 plt.show()
          9
```

Question 3.2 Plot Histogram with bin count more than 8

```
In [ ]:   1 #complete code to plot a histogram and changing bin count to a number grater than 8
          2 #for the column of your choosing from the dataset
          3
          4 ds.hist(column="",        # Column to plot
          5             figsize=(8,8),        # Plot size
          6             color="",        # Plot color
          7             bins= )        # number of bins
          8 plt.show()
```

Question 4.1 Plot Box-Plot

```
In [ ]:   1 import seaborn as sns
          2 sns.set(style="whitegrid")
          3 ax = sns.boxplot(x=ds[""]) # insert Column to plot between quotation "" marks
          4 plt.show()
```

Figure 3: Partly written Python code embedded in Jupyter Notebook, provided to students in lesson 4.

```python
1 import numpy as np
2 import pandas as pd
3 import matplotlib
4 import matplotlib.pyplot as plt
5 import seaborn as sns
```

Question 1.1 Comparing Different Data Sets

```python
1 #read in csv file
2 #this will be dataset 1
3 ds1 = pd.read_csv('daily_phoneuse_1stperiod.csv')
```

```python
1 #read in same csv file from different class section
2 #this will be dataset 2
3 ds2 = pd.read_csv('daily_phoneuse_2ndperiod.csv')
```

Question 1.2 Plot Descriptive Tables

```python
1 #plot descriptive table for dataset 1
2 ds1.describe()
```

```python
1 #plot descriptive table for dataset 2
2 ds2.describe()
```

Question 2.1 Plot Histograms

```python
1 #complete code to plot histogram for dataset 1
2 ds1.hist(column="screentime",          # Column to plot
3                figsize=(8,8),            # Plot size
4                color="blue",             # Plot color
5                bins=15 )                 # number of bins
6 plt.show()
```

```python
1 #complete code to plot histogram for dataset 2
2
3 ds2.hist(column="screentime",          # Column to plot
4                figsize=(8,8),            # Plot size
5                color="red",              # Plot color (choose a different color to represent
6                bins=15)                  # number of bins
7 plt.show()
```

Question 2.1 Plot Box Plots

```python
1 #complete code to plot box plots for dataset 1
2 #insert column name for dataset 1 between red qoutation marks
3 sns.boxplot(x=ds1["screentime"], color="blue")
4 #adjust xlim to best represent your data
5 #both figures must be on the same x-scale
6 plt.xlim(0,50)
7 plt.show()
8
```

```python
1 #complete code to plot box plots for dataset 2
2 #insert column name for dataset 2 between red qoutation marks
3
4 sns.boxplot(x=ds2["screentime"] , color="red" )
5 #adjust xlim to best represent your data
6 #both figures must be on the same x-scale
7 #plt.xlim(1.5,4.5)
8 plt.show()
9
```

Question 3.1 Remove data from data set

```python
1 #remove values from dataset 1
2 #enter the number closest to mean after == symbol
3 indexNames = ds1[ ds1['screentime'] == 5 ].index
4 ds1_remove = ds1.drop(indexNames)
```

```python
1 # remove values from dataset 2
2 #enter the number closest to mean after == symbol
3 indexNames = ds2[ ds2['screentime'] == 3 ].index
4 ds2_remove = ds2.drop(indexNames)
```

Question 3.1 Plot Descriptive Tables

```python
1 #plot descriptive table for remove data from dataset 1
2 ds1_remove.describe()
```

```python
1 #plot descriptive table for remove data from dataset 2
2 ds2_remove.describe()
```

Question 3.3 Remove data from data set

```python
1 #remove values from dataset 1 by
2 #enter the number furthest away from the mean after == symbol
3 indexNames = ds1[ ds1['screentime'] == 40 ].index
4 ds1_remove = ds1.drop(indexNames)
```

```python
1 # remove values from dataset 2
2 #enter the number furthest away from the mean after == symbol
3 indexNames = ds2[ ds2['screentime'] == 6.5 ].index
4 ds2_remove = ds2.drop(indexNames)
```

Question 3.3 Plot Descriptive Tables

```python
1 #plot descriptive table for remove data from dataset 1
2 ds1_remove.describe()
```

```python
1 #plot descriptive table for remove data from dataset 2
2 ds2_remove.describe()
```

Figure 4: Partly written Python code embedded in Jupyter Notebook, as provided to students in lesson 5.

Data Analysis

We investigated our research questions using students' work in the three lessons that involve Python (lesson 2, 4, and 5). First, we analyzed students' understanding of mathematics content using Python (research question 1) through artifacts of student work (e.g., plots) and students' responses to questions about statistics. Second, we analyzed student's perception of using Python to understand mathematics content (research question 2) using students' reflective responses from lesson 4 and 5. The reflective responses asked students to summarize what they learned, enjoyed, and did not enjoy during the lessons. One researcher qualitatively coded 43 reflective responses based on response similarity. All reflective responses were first coded into three categories: *Like Python*, *Did Not Like Python*, or *Neither*, if responses do not fall in either of the Like/Did Not Like Python categories. In addition, students expressed two more themes, specifically that they *Learned Content* and felt *Confused/Frustrated* when engaging with Python for mathematics content. The code *Learned Content* is applied if students explicitly made reference to a concept from the lesson and/or correctly explained what they learned. *Confused/Frustrated* is coded if the students specifically mention any confusion or frustration.

Results

Below, we first report students' responses to questions in lesson 2, 4, and 5 to assess students' understanding of mathematics content (research question 1). In our analysis, we found that some students were absent from some lessons or did not complete lessons even if they were in class. Thus, for each of the lessons below, we analyzed data for submitted questions and report percent completion for each lesson.

Students' Understanding of Mathematics Content

In lesson 2, of 51 students in class that day, 92% of students completed more than 70% of the lesson, or about 10 out of the 14 questions. All students submitted correct descriptive tables, box-plots, and histograms. Figure 5 shows all correct and incorrect student responses by question. Results showed that most students correctly interpreted the values for mean, standard deviation, and quartiles. Most students were also able to describe the distribution (shape, center, spread, outliers) of the histogram. However, students struggled on questions about the boxplot. More students were unable to identify the information (descriptive statistics) shown on a boxplot and what the box length of boxplot represents.
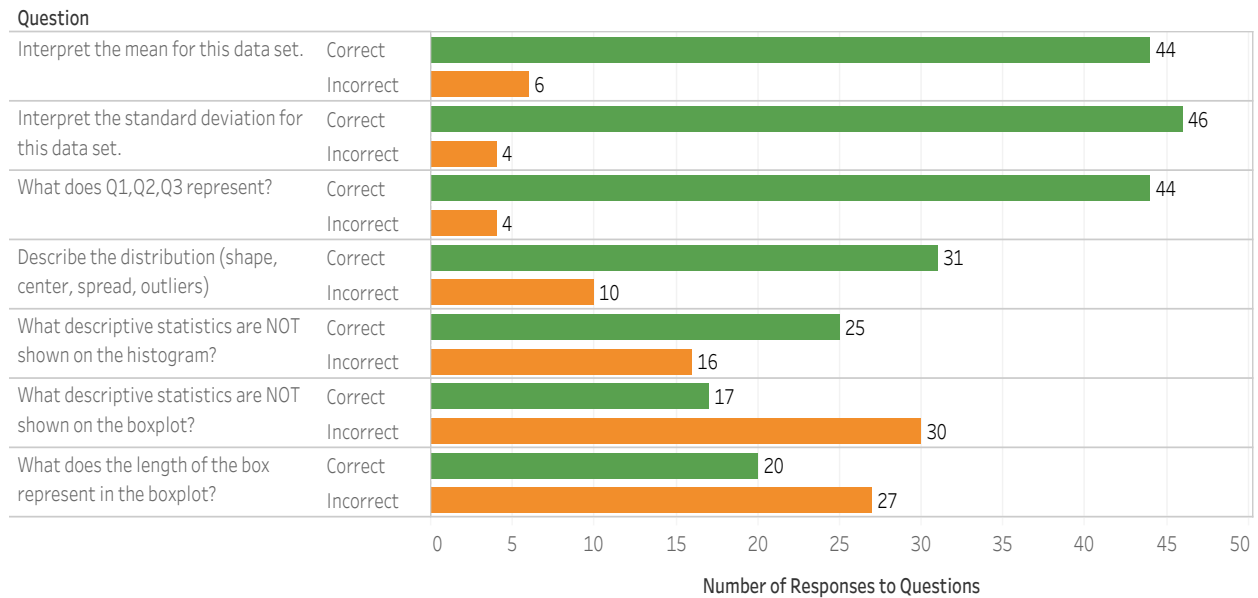
Figure 5: Lesson 2 correct and incorrect responses to questions regarding measures of spread and central tendency, the shape of histograms distribution, and interpretations of plots (table, boxplot and histograms).

In lesson 4, of 44 students in class that day, 80% of students completed more than 70%, or 7 out of the 10 questions in the lesson. All students submitted correct descriptive tables, box-plots, and histograms. Figure 6 shows all correct and incorrect student responses by question. The majority of students ($n = 21$) were able to correctly predict the shape of the histogram distribution before plotting using values of the mean and median from the descriptive table. In their responses to open-ended questions, students could describe the distribution of a histogram and decide on an appropriate number of bins for the given data set. Students could also identify how a histogram can be interpreted in various ways depending on the bin count. For example, one student stated: "I could see how the data changes when you change the bins, which can deceive people about the data." In addition, the majority of students ($n = 26$) were able to identify the information shown on a boxplot, even though most students were unable to answer this question in lesson 2. Students could also identify pros and cons of boxplots. For example, a student stated: "The pro is that you can see the min, max, median, IQR, outliers, Q1, and Q3 in a fairly simple graph. The con is that you can't see the frequency of each data point or the variability within the IQR. You also can't see the mean and standard deviation."
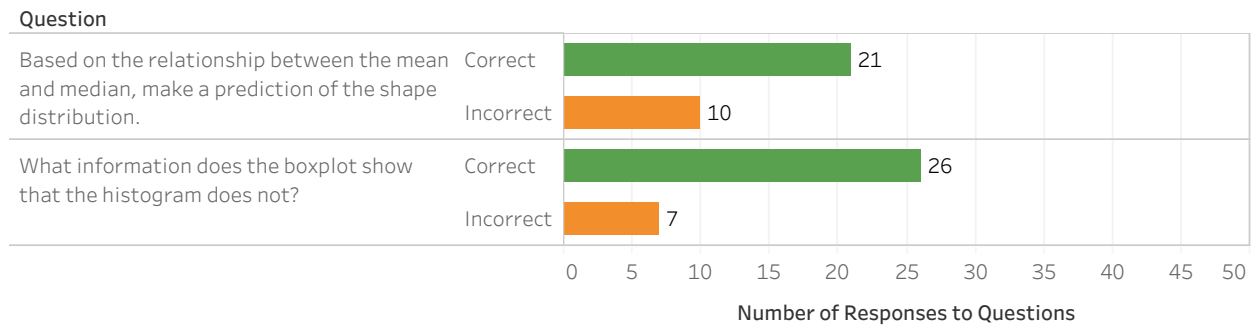
Figure 6. Lesson 4 correct and incorrect responses to questions regarding the shape of histogram distribution based on median and mean and interpretations of plots (table, boxplot and histograms).

In lesson 5, of 47 students in class that day, 43% of students were able to complete more than 70% of the lesson, or 7 out of the 10 questions. Log data shows that the students who completed lesson 5 spent more than 42 minutes, the time of a class period, which means that they completed the lesson after class. All submitted descriptive tables, box-plots, and histograms were correct. Figure 7 shows most students were able to accurately answer the questions in lesson 5. Students could describe distributions (shape, center, spread, outliers) and compare their data to another class as exemplified by one student: "The spread in period one is much larger due to an outlier of 25. This caused the graph of period one to be skewed to the right. In period two the spread is smaller and skewed to the left. There are no outliers." Further, most students could accurately discuss how adding and removing a point closest to the mean affected the standard deviations. For example, one student stated: "If [a data point] was close to the mean it wouldn't have a big effect. However, if we had removed an outlier, like 40 the STD would change".
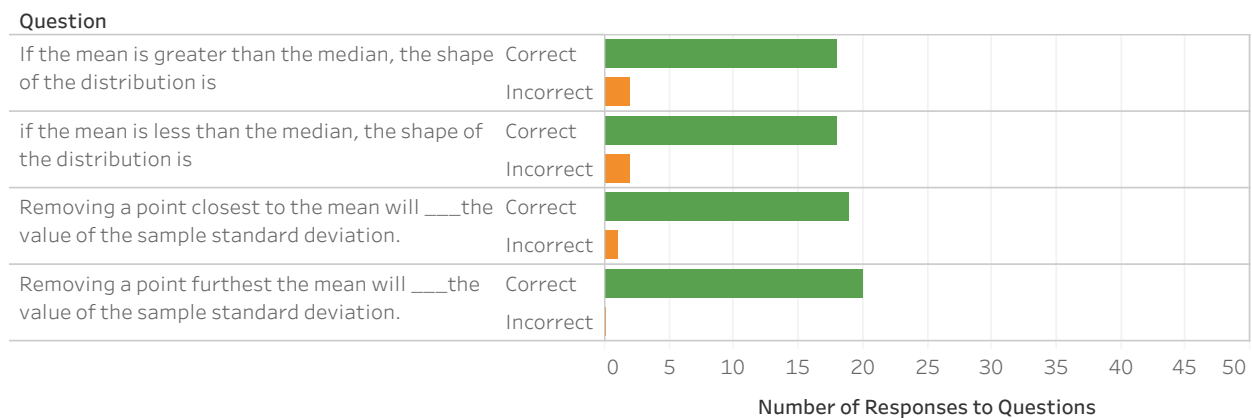
Figure 7. Lesson 5 correct and incorrect responses to questions regarding the shape of the distribution and effect of removing data values.

Students' Perceptions of Programming

In this section, we discuss reflective responses from lesson 4 and 5 to identify how students perceived programming with Python to learn mathematics content (research question 2). Recall that in these responses, students summarized what they learned and discussed what they liked and did not like about the lesson.

In lesson 4, 30 students submitted reflective responses. Figure 8 shows that the majority of the students (n = 17) stated they liked using Python in this lesson for example, a student stated: "I used descriptive values to guess at the shape and SOCS. I also compared the usefulness of box-plot and histograms and what info they both give. I liked it. It wasn't that confusing, and it is useful information for the future when I have to decide which one (graph) to use in different situations." Four students mentioned not liking Python, and three of these students explained that Python was frustrating and became confused. For example, one student stated: "I disliked it because it was frustrating and was not properly working for me. However, I learned more about box-plots/histograms." The remaining 9 students did not mention liking Python or not. Regarding other themes, Figure 9 shows that only six students stated they were confused/frustrated by python, and most students (n = 23) stated they learned content and could correctly identify the target content for the lesson (e.g., "We learned [the] value of histogram vs box-plot and how scale [bin counts] of histogram impacts how the data is shown. I liked this lesson.")
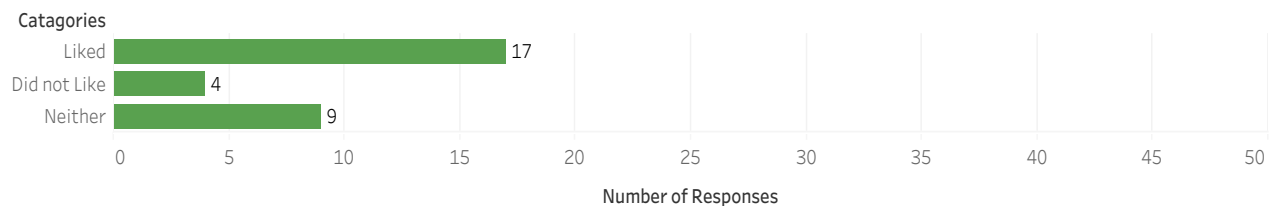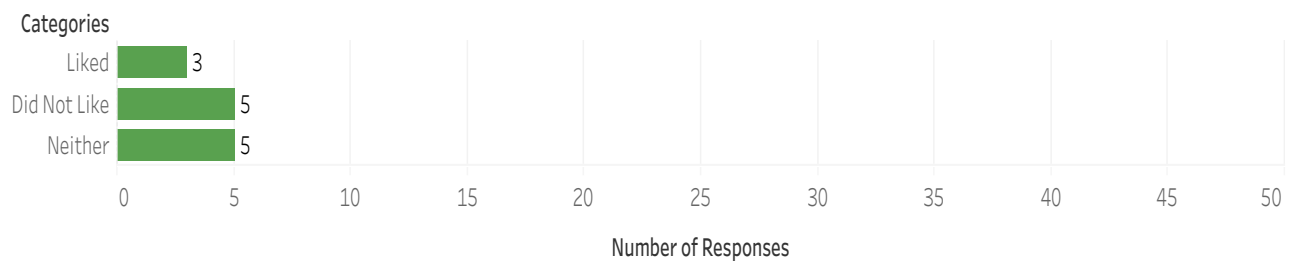

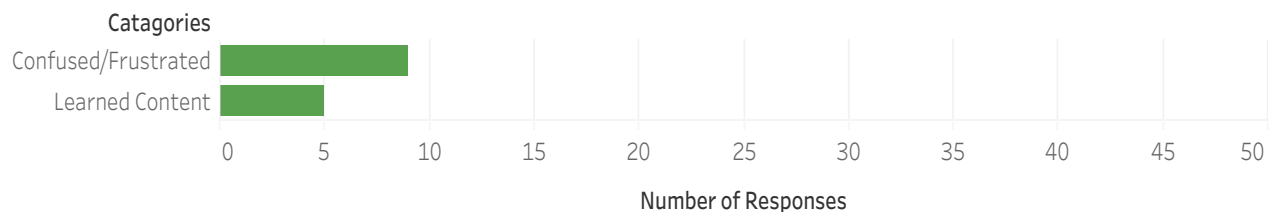Figure 8. Students' reflective responses in lesson 4 on whether they liked or disliked Python
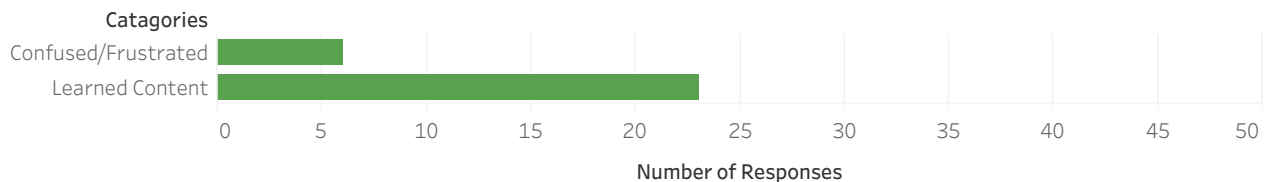
Figure 9. Students' reflective response in lesson 4 on whether they Learned Content or was Confused/Frustrated.

In lesson 5, 13 students submitted reflective responses. Figure 10 shows that some students liked the lesson (n = 3), but more students (n = 5) did not like this lesson. For example, a student stated: "I thought that there were a lot of issues with the code and that it was overly time consuming for what we actually got out of it. The code sometimes just didn't work, or we would spend a lot

of time fixing simple issues that didn't teach us anything about statistics, which was frustrating." Figure 11 shows that some students (n = 5) learned content but most students were confused/frustrated (n = 9) with the lesson. For example, one student stated: "Some steps were not working, and I had to ask for a lot of help from outside resources. However, I learned how mean, median, and skewness are all related as well as standard deviation." Overall, students recognized the benefits of this lesson, but was also confused, as stated by this student: "I learned how to graph different real-world data. I liked seeing real information in a visual manner as it helped me understand it better. I was a little confused, however, with some of the instructions for the coding."

Figure 10. Students' reflective responses in lesson 5 on whether they Liked or Disliked Python.

Figure 11. Students' reflective response in lesson 5 on whether they Learned Content or was Confused/Frustrated

Discussion

This study provided some preliminary evidence about integrating a text-based programming language in a formal high school AP statistics class. When scaffolded to novice learners, text-based programming can assist with the teaching of mathematics content. We scaffolded Python-integrated lessons by embedding hints that guide students in generating their results (e.g. tables, boxplots and histograms) and questions that reflect on topics covered in each lesson. This scaffolding may encourage students to conduct their own explorations with less structure and support from the teacher.

Overall, our results suggested that most students showed an understanding of distributions in context (shape, center, spread, outliers), due to students having the opportunity to explore the data sets in depth. For example, allowing students to think more about the effect of bin size in a histogram, through them making adjustments. Through this process students see how different graphs of the same data can provide different information and interpretations [27]. We also

observed that, in Lesson 2, some students struggled with interpreting what is shown or not shown on boxplots, but by Lesson 4, most students could interpret the difference between information shown on a boxplot versus a histogram. This growth of understanding could be due to students being repeatedly prompted to interpret box-plots and histograms after plotting.

Students' reflective responses on lessons 4 and 5 suggest that they enjoyed using Python to create multiple data visuals of real-world datasets. Students also enjoyed the opportunity to make various adjustments to the code to observe changes in data representation. Python allows students to easily make adjustments to data and immediately see the effect of their choices such as bin size in a histogram. Students also liked being able to explore datasets through multiple visual representations (i.e., histogram, boxplot, descriptive table). In the future, additional prompting may help students reflect on the affordances of automatic plotting of visual representations to better understand and experience the practice of doing statistical analysis [4] in a text-based programming language.

Additional prompts may also help students have better perceptions about Python as a tool to understand the mathematics content covered. One student reported that Python did not make statistics easier for them to understand. The students sometimes had difficulty understanding how the Python worked, which led to some frustration, especially when they get syntax errors. Although most of the code in the lesson was written and only required students to make minor modifications, students were not fully confident in debugging and interacting with the code. This is unsurprising because most of the students have not been exposed to programming, so they may require an introduction of the tool. In future implementations we would add questions about the debugging process such as, "how did you determine what to change in the python code when you got the error?" as well as a class discussion about how to debug code. In addition, students could receive opportunities to share their code with each other and provide feedback.

The results overall suggest that a statistics lesson that integrates Python can be an effective technology when implemented in a formal high school classroom. However, because students struggled with Python, we suggest that future iterations include more class time to engage students in programming practices. Recall that the first lesson using Python only required students to click through the code and get automatic results. However, in lesson 4 and 5, students had to write and debug a part of the code to produce results. This may explain why we observed that, as the lessons began to get harder and longer, the students were less likely to finish. Teachers could provide students with additional time to finish or spread lessons across multiple class periods. In addition, teachers can also consider how to engage students in deeper reflections about their understanding at the intersection of mathematics and programming. For example, adding explicit discussion and reflections about mathematics with and without programming may help students recognize the value of using Python for solving real-world problems in statistics.

Conclusion

Taken together, our findings suggest that integrating a text-based programming language, Python, in a high school mathematics course can engage students with real-world data analysis without hindering their understanding of mathematics content. Although Python can support more student-driven and open-ended explorations, teachers must be cautious when using it, particularly in the K-12 classrooms, because students can become frustrated by the debugging process. Our study presents one promising example of how Python can be embedded in a high school statistics course to help students understand mathematics and share the power of programming with students who would not normally engage with Python code.

References

[1.] AP Statistics: AP Central - The College Board. (2019, August 1). Retrieved from https://apcentral.collegeboard.org/courses/ap-statistics?course=ap-statistics

[2.] Barron, A. E., Ivers, K. S., Lilavois, N., & Wells, J. A. (2006). Technologies for education: A practical guide (5th ed.). Santa Barbara, CA: Libraries Unlimited

[3.] Baumer, Ben. 2015.A Data Science Course for Undergraduates: Thinking with Data. The American Statistician, vol. 69, no. 4, pp. 334–342.

[4.] Ben-Zvi, D. (2000). Toward Understanding the Role of Technological Tools in Statistical Learning. Mathematical Thinking and Learning, 2, 127-155.

[5.] Chance, B. L., Ben-Zvi, D., Garfield, J., & Medina, E. (2007). The role of technology in improving student learning. Technology Innovations in Statistics Education,1 (1).

[6.] Datta, Soma, and Veneela Nagabandi. 2017.Integrating Data Science and R Programming at an Early Stage. 2017 IEEE 4th International Conference on Soft Computing & Machine Intelligence (ISCMI), doi:10.1109/iscmi.2017.8279587.

[7.] DiSessa, A.A. (2018). Computational Literacy and "The Big Picture" Concerning Computers in Mathematics Education. Mathematical Thinking and Learning 20(1), 3-31.

[8.] Erickson, T. (2002). Fifty Fathoms: Statistics demonstrations for deeper understanding. Oakland, CA: EEPS Media.

[9.] Eyyam, R., & Yaratan, H. S. (2014). Impact of use of technology in mathematics lessons on student achievement and attitudes. Social Behavior and Personality: An international journal, 42, S31-S42.

[10.] Franklin, C., Hartlaub, B., Peck, R., Scheaffer, R., Thiel, D., and Freier, K. T. 2011. AP statistics: Building bridges between high school and college statistics education. The American Statistician

[11.] Friel, S. (2007). The research frontier: Where technology interacts with the teaching and learning of data analysis and statistics. In G.W. Blume & M.K. Heid (Eds.), Research on technology and the teaching and learning of mathematics: Cases and Perspectives (Vol. 2, pp. 279- 331). Greenwich, CT: Information Age Publishing, Inc.

[12.] Förster, K.-T. (2015). Programming in Scratch and Mathematics: Augmenting Your Geometry Curriculum. Today! In SIGITE, p. 97–97.

[13.] Foerster, K.-T. (2016). Integrating Programming into the Mathematics Curriculum: Combining Scratch and Geometry in Grades 6 and 7. in Proceedings of the 17th Annual Conference on Information Technology Education, ser. SIGITE '16, D. Boisvert and S. J. Zilora, Eds. New York, NY, USA: ACM, pp. 91–9

[14.] Gadanidis, G., Hughes, J. M., Minniti, L., & White, B. J. G. (2016). Computational thinking, grade 1 students and the binomial theorem. Digital Experiences in Mathematics Education. Advanced online publication. 1-20.

[15.] Garofalo, J., Drier, H. S., Harper, S., Timmerman, M.A., & Shockey, T. (2000). Promoting appropriate uses of technology in mathematics teacher preparation. Contemporary Issues in Technology and Teacher Education.

[16.] Hardin, J., Hoerl, R., Horton, N. J., Nolan, D., Baumer, B., Hall-Holt, O., Ward, M. D. 2015. Data Science in Statistics Curricula: Preparing Students to "Think with Data". The American Statistician, 69(4),343-353.

[17.] Hunter, J. D. (2007). Matplotlib: A 2D Graphics Environment, Computing in Science & Engineering, vol. 9, no. 3, pp. 90-95.

[18.] Kaput, J. (1992) Technology and mathematics education. In A Handbook of Research on Mathematics Teaching and Learning. pp. 515–556.

[19.] Konold, C., & Miller, C.D. (2005). TinkerPlots: Dynamic Data Exploration (Version 1.0) [Computer software]. Emeryville, CA: Key Curriculum Press

[20.] Lazar, N., & Franklin, C. 2015. The Big Picture: Preparing Students for a Data-centric World. Chance,28(4),43-45.

[21.] McKinney,W. 2011. Pandas: a foundational python library for data analysis and statistics. Python for High Performance and Scientific Computing, 1–9.

[22.] Michaeli, T. and Romeike, R. (2017). Addressing teaching practices regarding software quality: Testing and debugging in the classroom. In Proceedings of the 12th Workshop on Primary and Secondary Computing Education, pages 105--106. ACM.

[23.] Monaghan, J., Trouche, L., & Borwein, J. M. (2016). Tools and mathematics - instruments for learning. Switzerland: Springer International Publishing

[24.] National Council of Teachers of Mathematics. (2000). Principles and Standards for School Mathematics. Reston, VA: Author.

[25.] Ozgur, C., Colliau, T., Rogers, G., Hughes, Z., Myer-Tyson B, Matlab vs. python vs. r, Journal of Data Science, vol. 15, no. 3, pp. 355–372, 2017.

[26.] Psycharis, S., & Kallia, M. (2017). The effects of computer programming on high school students' reasoning skills and mathematical self-efficacy and problem solving. Instructional Science, 45(5), 583–602.

[27.] Rau, M. A., Aleven, V., & Rummel, N. (2016). Supporting students in making sense of connections and in becoming perceptually fluent in making connections among multiple graphical representations. Journal of Educational Psychology.

[28.] Satyanarayana, Ashwin. 2013.Software tools for teaching undergraduate data mining course. Smerican Society of Engineering Education Mid- Atlantic Fall Conference.

[29.] Sfard, A. and Leron, U. (1996). Just give me a computer and I will move the earth: Programming as a catalyst of a cultural revolution in the mathematics classroom. International Journal of Computers for Mathematical Learning

[30.] Sherin, BL. (2001) A comparison of programming languages and algebraic notation as expressive languages for physics. Int J Comput Math Learn 6(1):1–61

[31.] The Concord Consortium. (2014). Common online data analysis platform. @Concord, 18(1), 16.

[32.] Toomey, D. (2017). Jupyter for data science: Exploratory analysis, statistical modeling, machine learning, and data visualization with Jupyter. Birmingham: Packt Publishing

[33.] VanderPlas, J. (2016). Python Data Science Handbook. OReilly Media, Inc.

[34.] Weintrop, D. (2015). Comparing Text-based, Blocks-based, and Hybrid Blocks/Text Programming Tools. in Proceedings of the eleventh annual International Conference on International Computing Education Research. ACM., pp. 283–284.