Deconstructing Syntax and ggplot2

Facilitation Guide

Edward Chang
Educ 150: K12 Data Computer and Data Science Education
2020 December 14

Introduction

This activity builds upon the basic coding skills in R taught in introductory data science (IDS) courses or similar courses. Fundamentally, IDS courses mainly focus on introducing the concepts of data science (DS) to students. To do so, they usually take a highly abstracted approach to coding in DS such that no computer science background is required for the class and so that the focus on programming concepts can be lessened to the point of memorization of how to apply functions or other required programming skills. These highly abstracted approaches include the usage of packages designed specifically for the class. This approach, in practice, imposes a high-level natured view of the programming language and obstructs students from doing more than the contents of the course without further courses ("The datascience Package," n.d). This is obviously poor in practice due to it conflicting the interest of a student who aspires to learn long term applicable skills in DS but is not aware of the pedagogical aspect of the course to mainly focus on computational thinking ideas. Most approaches are all extremely similar in that they take away the programming barrier that make DS inaccessible through means of only requiring students to understand code or the functions they use at a high level such as in Gould et al. (n.d.), which also excludes references to the packages that the R code involved in the curriculum uses.

Due to the highly technical aspect of this activity, this activity mainly focuses on the cognitive framing (Kafai et al., 2019), the support of student's comprehension of the programming or CS concepts and ability to perform on tasks to program something. You can see this used in activities developed to help students understand concepts such as loops, recursion, data structures, etc., and for my purposes, I will be using this framing to explore the development of students' mental models on visualization code and evaluation procedure that allow you to write code as it is. Also because of its technical aspect, this activity is useful towards students because it focuses more on these cognitive aspects of data science programming which IDS courses abstract and lack in adequate support for truly understanding data science code.

Audience

This activity is intended for students who have taken or are taking an IDS course, this typically applies to students who are in 11th grade or higher since mathematical knowledge of algebra II is typically required for such students. The activity is designed to cater to students in IDS courses who are interested in tackling more problems in DS through elevating their understanding of the code that they write.

Because of this conflict in interest with IDS courses, this activity is not necessarily part of any (specific) K12 school curriculum, in fact, we would see this out of scope in most K12 curriculums for an IDS course since the purpose of the class would be to introduce the concepts of DS.

Overview

To fully understand this activity, students are expected to have proficiency in the limited programming concepts already taught to them in an IDS course using R. They are also expected to have knowledge of the basic ideas about making visualizations (e.g., the axes, labeling a plot, scales, plotting data). The activity assumes a low ceiling of programming with the student; the student is only expected to know what an IDS course teaches, that is to be able to use code as a tool to achieve a purpose. Specifically, we only explicitly cover examples of data frame manipulation in `dplyr` and visualization in `ggplot2` which students are expected to understand at first glance.

In terms of mental models, the student could start with just seeing that there are set ways to do things in R but with no justification for why it is so. They can simply have blind trust in the code that they write by memorizing the examples that they've already seem. The activity aims to help students to evolve their mental model of coding in R for DS tasks just a bit further to combat the abstracted way they learned ggplot and tidyverse code. They will be supported to develop their mental models of visualization code and evaluation procedures such that they realize why they can write code as it is.

To do all this, our activity aims to have students (1.) apply knowledge of ggplot layers to create visualizations that match given criteria, (2.) explain the reason for the usage of individual ggplot layers, and (3.) identify instances of standard evaluation (SE) and non-standard evaluation (NSE) to differentiate between the syntax used between packages/functions. Since our activity starts with the assumption of the student only needing to understand code through memorization, it is equipped to support students into our intended mental model through relating the tools that they have already seen and shedding light on them by revealing reasons for the previously unidentified rules that they have been utilizing but not fully understanding. The assessments evaluate the student on how well they can utilize the concepts that they have learned during the activity on unfamiliar situations.

Detailed Activity Description

Specific Learning Outcomes

By completing this activity, students will:

1. Apply knowledge of ggplot layers to create visualizations that match given criteria
2. Explain the reason for the usage of individual ggplot layers.
3. Identify instances of standard evaluation (SE) and non-standard evaluation (NSE) to differentiate between the syntax used between packages/functions.

Resources

1. Computer
    a. Internet connectivity

    b. Zoom or similar conferencing installed (if synchronous)
    c. Web browser

2. Created activity material web app
    a. Deployed at: `https://equil.shinyapps.io/lesson`
    b. GitHub repository (for self-deployment or local running): `https://github.com/equilnite/tidyverse-lesson.`
    c. See Appendix A for steps and notes on deployment or local running

Activity Overview

This activity is intended for students in 11th grade and up for students who are currently or have taken an introduction data science course (or similar course) that uses R. Participants are expected to have proficiency with DS concepts in visualization and data manipulation. The Stat 33A/B material at UC Berkeley (Ulle, 2020) will be a source of inspiration for the material of the activity; otherwise, all material is created by me.

The activity consists of a web app with lecture style notes and interactive R code snippets. Within the activity, students will work through the web app on the topics of layers in ggplot2 and SE and NSE in R. Students will demonstrate their knowledge with blank or incomplete code snippets involving visualization creation. In addition to visualization creation, true-false style questions on SE and NSE will be provided and drawn from examples in both the tidyverse and base packages in R.

ggplot is specifically chosen due to the abstracted processes that IDS courses using the package would normally introduce it. For example, if you ask a typical student in an IDS course how to make a histogram using ggplot, they will normally just demonstrate that something like `ggplot(…) + geom_histogram()` will do it—without being able to explain coherently why there is a `+` connecting the function calls. By abstracting the reason for this behavior, the teaching of ggplot is abstracted. To be taught in a less abstracted way (which is what I will do), student will have to learn how to explain such a reasoning for the ggplot plot code, this means having to learn that it is based on *The Grammar of Graphics* (Wilkinson, 2005), which basically details a comprehensible and standard “grammar” for data visualization. So, students will be taught to understand that ggplot operates on a different grammar compared to the rest of R.

SE and NSE is taught due to the abstracted explanation of tidyverse code. Students in IDS courses will normally not be able to differentiate between syntax usage in tidyverse versus other packages not based in the tidyverse ideology of human readable code. One example is not being able to differentiate between the base `filter` function and the one in dplyr since all they have had to know is using the dplyr one. Since they do not necessarily know when they can unquote or need to quote variable (column) names within data frames, they abstract it. And abstracting this idea is problematic since students who want to utilize other packages might be confused when they must refer to variable names in quotes compared to them being to refer to variables name—unquoted—in tidyverse functions.

This activity may be done asynchronously or synchronously, depending on the need. All sections of the activity may be done independently by a student; however, synchronous work is recommended due to the technical aspects of the content of the activity.

In synchronous work, the facilitator is at least expected to give a walkthrough of the web app and be able to talk more about the pre-work section if required by a student. The student should also be sharing their work with the facilitator over time such that immediate feedback may be given to a student. All technical and conceptual questions should be addressed by the facilitator. Synchronous work is also recommended since many of the elements in the activity are interactive; being able to observe the student gradually giving typing their code or seeing the exact elements of the activity they reference provides insight into their understanding of several parts of the material.

In asynchronous work, this activity may also be administered with Appendices A and B such that technical difficulty is reduced. Since the activity flows in a narrative form and explains all technical aspects that students would require to complete assessment tasks, students should be capable of completing the activity by themselves. Since understanding is not fully demonstrated by completion of the webpage itself, Appendix C may be provided to the student to make up for the missing prompts or observations that would be made during synchronous work.

Pre-work (optional)

Given that this activity is intended for current students or past students of an IDS course, an optional section will be included in the activity ("the 'R'efresher" section in the activity). This optional section reviews the very basics of R (i.e., types of data structures, assignment, order of operations, using `ggplot2` to create visualizations, and the usage of dplyr to selectively clean data). By including information about data structures and objects (assignment), the activity can then reference to that knowledge when talking about the object created by ggplot and be used as a basis of understanding what SE and NSE means in terms of our expectation when we run code. Any additional information in this section is purely intended to help students elicit prior knowledge about R syntax and skills in R.

(Synchronous) Activity Timeline

Section	**Direction(s)**
Introduction *3-10 mins*	1. Introduce the specific learning objective to the students. 2. Send the deployed app or have students follow the steps (documented on the GitHub repo and in Appendix A) to run the web app individually. 3. Instruct students on how to navigate the web app (See Appendix B for guidance). 4. If doing pre-work, guide students through the section. a. Primary focus: reviewing R basics presented and interpreting R code examples at the level of previous course work.
Activity *30 mins**	• Students will work through the material. • Answer any questions students may have that are not clearly stated in the material.
ggplot Assessment #1 *5 mins*	• Assessment Description: reverse-engineering a visualization to equivalent code. o Meant to gauge the most basic understanding that the student may have about ggplot. • Have students identify the layers before looking at the code snippet and completing the assessment. o Refer them to the table of layers in the material • Students should already have explicit knowledge of how to use the needed functions.

	o The students may have forgot the arguments of the `labs` function. Encourage them to look at the hint if so. • On completion—ask students to present their code and highlight the types of layers used and the reason for them.
ggplot Assessment #2 *10 mins*	• Assessment Description: criteria for a visualization in text form to equivalent code to create the visualization. o Challenging to students because it requires the usage of a scales type layer which is not used in the Stat 20 curriculum o Meant to observe how well a student can approach and unfamiliar situation by applying knowledge about fundamentals of ggplot layers • Let students determine their own approach. o Recommended: Ask them to identify the types of layers needed and the reason that type(s) are needed. • Students may become lost on the scale criteria. o Refer them to the hint. • On completion—ask students to present their visualization and code to explain the reasoning for the usage of specific layers/functions.
SE and NSE Assessment *5 mins*	• Assessment Description: identifying code as using standard evaluation or non-standard evaluation. • Students may be confused on the distinction between strings and objects in the context of evaluation. • On completion—ask students to explain some select answers, preferably also questions that used strings.

*Assessments are not factored into this time, though assessments are part of the activity themselves.

Assessment

Students will demonstrate their understanding on ggplot material through two tasks to write code that reproduces specified visualizations. One task will be two reverse-engineer a visualization to equivalent code and the other is to write code based on a text description of a desired visualization. On completion of the code, determine how accurate their resulting visualization is according to the tasks' requirements and then have the student explain their reason and usage for the layers they utilized in their code.

SE and NSE understanding will be primarily assessed by the correctness of responses to a true-false style quiz on the activity. On completion of the quiz, prompt the student to further explain their processes in determining evaluation types for the 3rd, 4th, 6th, and 8th questions.

Appendix C may be administered for asynchronous delivery.

Rubrics

ggplot2 assessment

Criteria	Beginning	Developing	Proficient	Strong
Accuracy of visualization recreation	Completes at least one task with major errors in recreating desired visualization(s)	Completes both tasks with major errors in recreating desired visualization(s)	Completes both tasks with minor errors in recreating desired visualization(s)	Completes both tasks—fulfilling all desired aspects of the visualization(s)

Explanation of visualization code	Explanation at least includes some reference to layer types	Explanation includes reference to layer types and relation from desired aspects of the visualization to code	Explanation is clear and concise and can attribute decisions in code written to aspects of the visualization with minor errors.	Explanation demonstrates understanding of every aspect of the code written and their relation to the visual properties of the image(s).
Guiding Questions	Does the visualization resulting from the student's code match up with the desired characteristics? How well is the student able to explain their code in terms of the material shown to them i.e., can they explain each function used and relate them to layers and the ggplot object?			

SE and NSE assessment

Criteria	Beginning	Developing	Proficient	Strong
Total correct	At least 2 correct.	At least 4 correct.	At least 6 correct.	8 (all) correct.
Explanation of answers	Explanation includes at least some sort of substantial relation to the material.	Explanation is at least able to draw from examples.	Explanation includes distinction between strings and objects with minor errors in other aspects.	Explanation includes proficiency description but with no error
Guiding Questions	Is the student able to answer the quiz questions correctly? How well can the student explain their answer? Can the student differentiate between using strings versus non-referenceable object names in the global environment?			

Student Evaluation

In the learner trial for this activity, a student who was well motivated and was strong in the R material taught in the current offering of Stat 20 at UC Berkeley (Fall 2020) was selected. When going through the activity, the student showed strong understanding of the material presented to them in the activity through their assessment answers and work. As all the work is primarily delivered in an interactive format with mostly independent work, it is best to evaluate students based on the interactions that the student has with the material and their explanations and reasoning of our assessment items. Answers to Appendix C can be used in lieu of the verbal communication or observation of a student's progress on the activity if there is no direct interaction between the facilitator and student.

For example, in the second ggplot assessment, my learner was trying to find the layer needed to "make the x-axis show the values in intervals of 10". This mean that she had to find a scales type layer that could do that, and it was not something that was gone over in Stat 20 curriculum. So, what happened was that she was able to refer to the table of ggplot layers in the activity to identify the need for a scales type layer. Then going onto finding the specific function to add as a layer on the ggplot documentation website, she was able to elicit her knowledge of data visualization and statistical terms (e.g., bins, continuous, discrete) to find the correct function among many other options (i.e., `scale_x_continuous`). She further showed use the same prior knowledge to correctly identify the range the data spanned when specifying the required intervals by referring to a previous scatterplot of the same data provided in the activity. An important thing to note is that also our learner had directly gone to attempting to `+` a scale layer to her visualization, showing how understood that there was a layered grammar and went directly to that step rather than say looking at the example of a specific function usage and seeing that they must `+` it to the rest of the visualization code.

The main defining point of her demonstration of a strong understanding on ggplot was the usage of an unfamiliar function (scale type layers are not used in Stat 20) and that it could be simply "added" to the visualization by using the `+` operator. As this activity is a build on of prior knowledge, my student successfully referring to their prior knowledge and ease of able to utilize an unfamiliar function displayed evidence of the student being able to apply their new technical knowledge about ggplot layers to perform a task outside the scope of their current knowledge. Although this is not specifically defined as an item on our rubric, this is an essential goal of what our explanation items in the rubric aims to do; we primarily want to look at how the student understands the code they write and viewing the little details that we can infer in lieu of our explanation criteria.

An explicit way of of the student being evaluated on the rubric for ggplot would for example, be the student explaining their usage of + and identification of layers in the context of the material. If the student can relate the layers to the table and have justification for why they chose that type of layer based on the criteria for a visualization, they can receive full marks on that rubric part.

Our rubric on the assessment of SE and NSE is also nothing substantial since the assessment is in a quiz form that is hard to pinpoint the exact thought process. But we will be looking for how well a student and identify between instances of each through correctness on the problems. More vaguely is assessing the student's method of differentiation between SE/NSE. The quiz covers a multitude of examples that make use of using strings versus object names. For the explanation component, we look for understanding of which things can be referenced by object name and string. For example, on this problem:

```
y <- rep(c(T,F), 25)
filter(cars, y)
```

We expect for the student to be able to understand that y can be directly referenced and assessed in the scope of our global environment and a simple explanation such as "we know what y" is would suffice. And another:

```
select(cars, 'dist')
```

We expect for the student to be able to differentiate between just using the string `‘dist’` and the object name dist. A simple explanation such as "`‘dist’` is quoted, so we don't need to know what it is" would suffice in this case.

Reflection

Based on what I have seen in my process of creating this facilitation guide and running a learner trial, I feel one of my motivations towards K12 DS education would be trying to increase programming skills among students who want to aim past their IDS course. But this obviously conflicts in the conceptual data science focus most IDS courses have, so this also brings up also the question of how much programming they should be learning or how I should deliver such instruction.

Towards students, I feel like my activity would help with their perspective of viewing computing as a much more flexible thing, where you can generalize rules to do more specific things (unlike them memorizing these specific things to do tasks). For a facilitator who is not used to these concepts, it would help them understand a new way of helping students understand the type of concepts I showed in the activity. Our result of also being able to assess a learner based on their interactions with the material might also encourage facilitators to take a dive into how the student thinks rather than waiting for the student to give an explanation. In this way, I think it can help enrich facilitators' view of learning computing in a way that they can now try to rely less on assessment through post-thought problem reflections (such as Appendix C; students fill it out after already thinking it through) and try to dissect the thought process themselves to see how the student(s) utilized their computing concepts to do something.

For my next steps, I think I would want to extend this lesson into more parts so that more programming concepts that are introduced in Stat 20 are explained in a more comprehensible way for students to understand. On the topic of Stat 20, a textbook is being developed just for the class, so I would also want to take a part in that textbook and maybe help in developing an appendix which can detail R like how I did in this activity. If I had to start over, I think I would want to observe a wider variety of responses from different students. Due to the relatively high skill-level of my specific learner, my perspective on how well it went is probably biased, so gaining a much wider range of responses would help me revise my activity accordingly.

References

The datascience Package. (n.d.). Retrieved December 01, 2020, from http://data8.org/zero-to-data-8/datascience.html

Gould, R., Machado, S., Johnson, T. A., Molyneux, J., Casillas, M., Estevez, H., . . . Zanontian, L. (n.d.). Introduction to Data Science Curriculum. Retrieved December 01, 2020, from https://curriculum.idsucla.org/

Kafai, Y, Proctor, C., and Lui, D. (2019). From Theory Bias to Theory Dialogue: Embracing Cognitive, Situated, and Critical Framings of Computational Thinking in K-12 CS Education. International Computing Education Research Conference (ICER '19), August 12–14, 2019, Toronto, ON, Canada. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3291279. [40 min]

Ulle, N. (2020). Introduction to (Advanced) Programming with R. GitHub repository. Retrieved December 01, 2020, from https://github.com/IntroToProgrammingWithR/2020-spring-stat33ab

Wilkinson, L. (2005). *The Grammar of Graphics* (Statistics and Computing). New York, NY: Springer New York. https://doi.org/10.1007/0-387-28695-0.

Appendices

Appendix A: Running/Deploying the Activity from Source

The source code for the web app containing the activity material can be found at: `https://github.com/equilnite/tidyverse-lesson`.

A deployed version of the web app can be observed at: `https://equil.shinyapps.io/lesson/`. This deployment may expire at any time or have downtime.

The steps for to run the web may be found in the `README.md` file in located in the GitHub repo. Here is an abridged version of the file content:

REQUIREMENTS

To run this app locally, R and RStudio need to be installed on your local machine. This app was additionally deployed with R version 3.6.3 (2020-02-29); compatibility issues may arise if used with another R version

The only file required to run and/or deploy the web app is the RMarkdown file `lesson.Rmd`.

To properly run the RMarkdown file, make sure that the `tidyverse`, `learnr`, `shiny`, `rmarkdown`, and `knitr` packages are installed; these packages can be installed by running `install.packages(c('tidyverse', 'learnr', 'shiny', 'rmarkdown', 'knitr'))` in a R session.

HOW TO RUN

RStudio

If using R studio, simply click on the "Run Document" button on the top left of the coding pane.

Terminal

If using the terminal, run `rmarkdown::run("lesson.Rmd")`.

HOW TO DEPLOY

There are several services that you may choose to deploy this web app with. An option is using shinyapps.io on `https://shinyapps.io`. The steps to using shinyapps.io as your hosting service and deploying your app can be found at: `https://shiny.rstudio.com/articles/shinyapps.html`.

Appendix B: Navigating the Web App

GENERAL USAGE

The student may scroll up and down through the page to read the contents of the activity. There are also several parts in the activity where there are questions prompted for the student to answer. These are mostly all assessments and the tasks include multiple choice and select all that apply questions that students may submit to check their answers. Interactable code snippets for students to work on visualization code are provided but they give no indication of correction besides being able to see a visualization created by the code. After completing a section ("Topic"), students may use the buttons on the bottom of the page to sequentially navigate between sections.

THE SIDE BAR

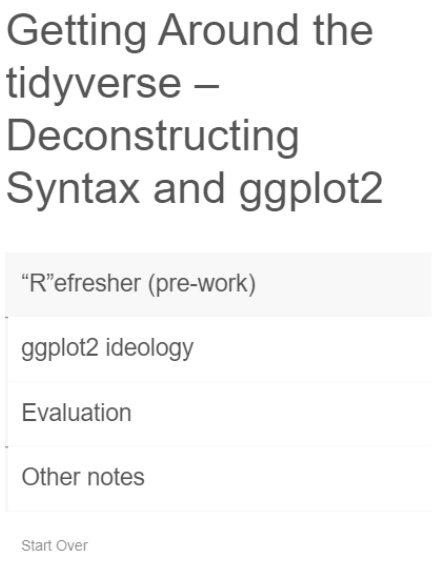
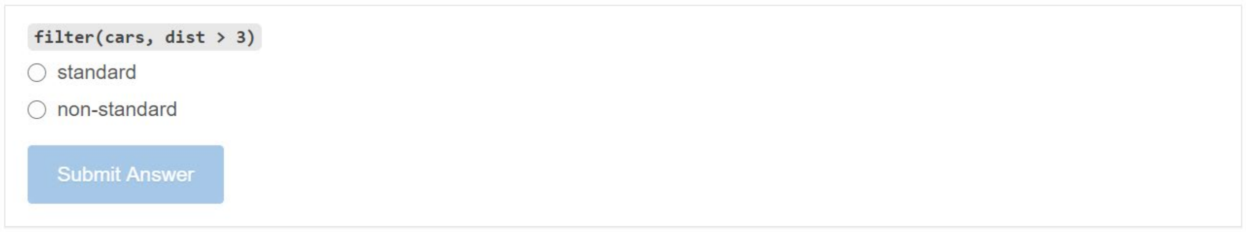


Figure B1: The side bar. A screenshot of the mentioned side bar in the web app.

The side bar contains shortcuts to all the sections for available for the students to read through and do their assessment activities. They may want to use the side bar to quickly navigate between sections.

There is also a `Start Over` button on the bottom of the side bar that can be used to clear all work done on the page and reset the page to the initial appearance.

MULTIPLE CHOICE QUESTIONS

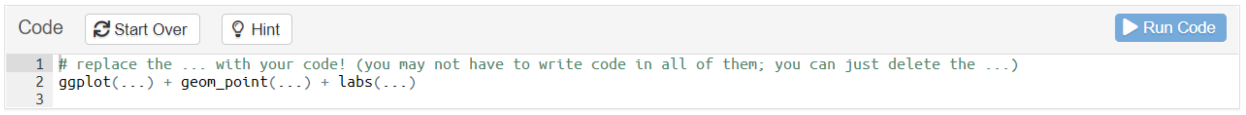


```
filter(cars, dist > 3)
```

Figure B6: A multiple choice question. A screenshot of the second question in the assessment of the section covering standard and non-standard evaluation.

Students may select one of the choices. The answers may be submitted with the Submit Answer button and may not be done again without the Start Over button on the side bar.

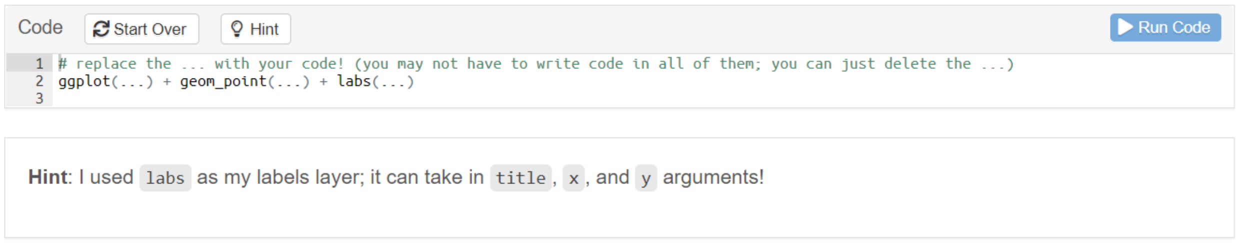
INTERACTIVE CODE SNIPPETS



```
# replace the ... with your code! (you may not have to write code in all of them; you can just delete the ...)
ggplot(...) + geom_point(...) + labs(...)

```

Figure B3: An interactive code snippet with an additional hint button. A screenshot of the first of the assessments covering `ggplot2` layers.



```
# replace the ... with your code! (you may not have to write code in all of them; you can just delete the ...)
ggplot(...) + geom_point(...) + labs(...)
```

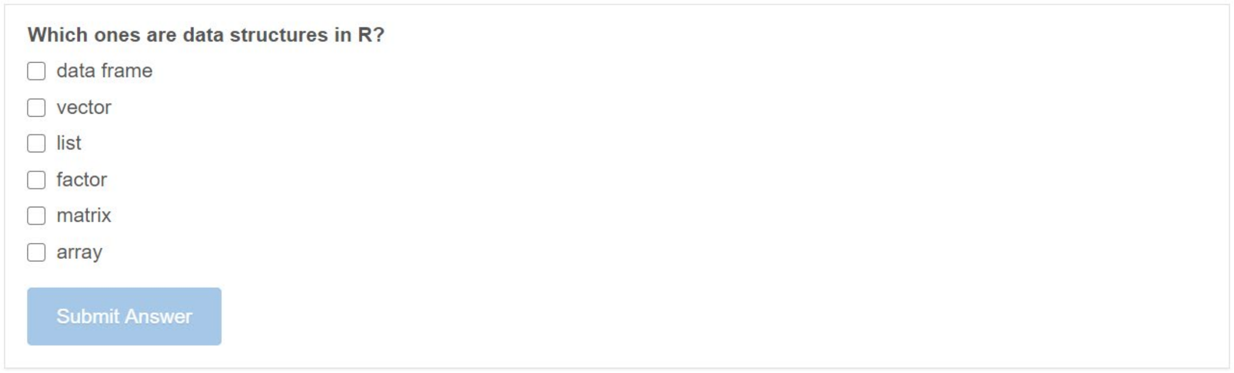
Hint: I used `labs` as my labels layer; it can take in `title`, `x`, and `y` arguments!

Figure B4: The same snippet from Figure B3 but with the hint revealed.

Students may type their own code into these snippets. There are several buttons to use:

- `Start Over`:
  - Reverts the code snippet to its original state
- `Hint`:
  - Reveals a hint that can be read below the code snippet. See Figure B4 for the hint given in the snippet provided in Figure B3.
  - *Note: This is not present in every interactive code snippet.*
- `Run Code`:
  - Runs the code present in the code snippet.
  - *Technical Note: Each "click" of the button runs in its own environment that is a child of the global environment. i.e., Running code does not carry over into consecutive runs of the same or other snippets.*

<u>SELECT ALL THAT APPLY QUESTIONS</u>



Which ones are data structures in R?

- [ ] data frame
- [ ] vector
- [ ] list
- [ ] factor
- [ ] matrix
- [ ] array

Figure B2: A select all the apply question. A screenshot of the only select all the apply question in the lesson material.

Students may click on each box to "check" or "un-check" a selection for the answers to submit to check for correctness. The answers may be submitted with the Submit Answer button and may not be done again without the Start Over button on the side bar.

Appendix C: Asynchronous Worksheet FIXME

Code from an Image

How did you decide which layers to use when trying to recreate a visualization from an image?

Explain how you chose your aesthetics.

Visualization from text

How did you decide which layers to use when trying to create your visualization from requirements?

Explain each function that you used in your code and your reason for choosing that function.

Explain your choice for each of the following:

```
read.csv('somedataset.csv')
```

```
x <- 1:10
ggplot() + geom_histogram(aes(x=x))
```

```
select(cars, 'dist')
```

```
library('ggplot2')
```