Software Carpentry Workshop Evaluation

Caitlyn Pickens

March 11, 2013

1 Introduction

A pre-workshop evaluation of the Software Carpentry Boot Camp was conducted on January 12, 2013; the matched post-workshop evaluation occurred January 13, 2012. 33 participants completed the pre-assessment; 29 participants completed the post-assessment. Of the responses, 9 instances can be matched as pre- and post-data for participants. This report includes a description of each part of the pre- and post-surveys, analysis of the data collected for these surveys, and a list of recommendations for the University of Chicago and for Software Carpentry for future workshops and assessment.

2 Workshop Evaluation, Scales and Results

Boot camp participants completed a pre-workshop survey containing three scales: Perception of Computational Ability, Computational Understanding, and Python Coding Ability. Students also completed several demographic questions and responded to an open-ended question about workshop expectations. The post-workshop survey contained two scales, Computational Understanding and Python Coding Ability, as well as questions about overall perceptions of the workshop.

Due to difficulty with distributing the surveys, many individuals did not receive the first page of the pre-assessment (which contained the Perception of Computational Ability and the Computational Understanding scales). Further, the Computational Understanding scale was largely modified from the pre- to the post- survey, making comparison between the two surveys inconclusive. As a result, the analysis of these scales has been excluded. If the University of Chicago or Software Carpentry require this analysis, it can be completed at a later date (although interpreting the data is not recommended).

For future workshops, it is recommended that all surveys are distributed via online methods to avoid data loss due to paper-related complications. Further, it is recommended that the scales be standardized across pre- and post-assessment.

The Python Coding Ability scale was used from Libarkin 2012 for both the pre- and post-surveys. The scale was tested to ensure it was appropriate for this population using the Kaiser-Meyer-Olkin measure of sampling adequacy and Bartlett's Test of sphericity.

For the purposes of the analysis, the items were forced onto a single factor (in line with Libarkin 2012). When not forced, it is evident that there are multiple distinct constructs being assessed. For example, in the case of the Python Coding Ability scale, 2-3 constructs exist, possibly measuring actual Python coding ability, problem-solving confidence, need for outside assistance when programming, or motivation/anxiety related to programming. There are also several items that seem to lack face validity in the survey, which is evident in low commonalities, to be discussed later.

2.1 Demographics

Demographic data are reported for pre-workshop respondents, with n=33 participants responding. Participants ranged in age from 21-63, with an average age of 29.2 years, a median of 26 years, and a mode of 29 years. Participants were predominantly female (n=18), with the remainder male (n=15) and no transgendered participants. Participants were 48% Caucasian (n=16), 42% Asian (n=14), 6% biracial Caucasian/Latino(a) (n=2), and 3% Latino(a) (n=1). The academic status of participants included undergraduates (n=1), graduate students (n=24), M.S.-holding professionals (n=1), Ph.D.-holding professionals (n=6), and unspecified "other" (n=1). Of the respondents, 6% indicated no previous programming experience (n=2), 21% indicated less than 6 months of programming experience (n=7), 6% indicated less than 1 year programming experience (n=2), 18% indicated less than 2 years programming experience (n=6), and 48% indicated more than 2 years experience (n=16).

2.2 Python Coding Ability

Items from Libarkin 2012 corresponded to Python coding ability. Participants were asked to indicate the extent to which they agreed with statements about their Python coding ability for 22 Likert-type items, with ratings of Strongly Disagree, Disagree, Agree, and Strongly Agree. A score of 1 implies low ability (Strong Disagreement), while a score of 4 implies high ability (Strong Agreement).

Factor Analysis. In the pre-workshop data, a single scale explains 37.07% of the variance and all items yielded factor loadings over 0.20, suggesting the presence of a single scale (Table 1). In the post-workshop data, a single scale explains 41.61% of the variance and all items yielded factor loadings over 0.33 (Table 2).

Reliability Analysis. Cronbach's alpha was calculated to establish internal consistency of items. The pre-survey had a calculated alpha of 0.67, the post-survey 0.70. The pre-survey alpha is slightly less than the preferred 0.70, but the low value may be due to sample size constraints.

Results. Due to the missing data and the problems with matching data, paired-samples tests could not be conducted. Independent-samples tests could also not be conducted because the pre- and post-survey did not have independent samples. Consequently, only item-level analyses were conducted. Table 1 and Table 2 show the communalities, factor landings, response average, and response meaning for each individual item on the pre- and post- surveys respectively.

Conclusions. As noted in Libarkin 2012 and Pickens 2013, certain items exhibited low communalities and therefore may need to be considered for alteration or removal from the Python Coding Ability scale. These items were:

- "I can find a way to concentrate on my program, even when there are many distractions around me."
- "I can find ways of motivating myself to program, even if the problem area is of no interest to me."

Other items (not identified by previous reports) exhibited similarly low communalities, and therefore should be observed in future analyses for low communalities (and then possibly considered for alteration or removal). They include:

- "I cannot complete a programming project unless I can call someone for help if I get stuck."
- "I can complete a programming project if I have just the built-in help facility for assistance."
- "I cannot complete a programming project unless I have the language reference manual."
- "I can mentally trace through the execution of a complex multi-file program developed by someone else."
- "I can write a complex Python program to solve any given problem as long as the specifications are clearly defined."

It is also recommended that Software Carpentry consider creating separate scales for Python coding ability and Python confidence/anxiety, since the Python

Coding Ability scale as-is appears to measure multiple constructs. The items identified as having low communalities may better fit into a scale concerning confidence/anxiety.

3 Overall Workshop Impression

Participants were asked to respond to several close-ended questions related to overall impressions of the workshop. Participants also responded to open-ended questions about their expectations for the workshop (pre-survey) and their perceptions of the workshop (post-survey).

3.1 Close-Ended Questions

Eight questions asked participants to rate components of the workshop, as well as the overall workshop on a 5-point Likert scale of Very Poor-Poor-Adequate-Good-Very Good. A response of 1 corresponds to a Very Poor rating and a response of 5 corresponds to a Very Good rating. Participants were also asked if the workshop met their needs on a 4-point scale (4=very well), if they learned as expected from the workshop, if understanding of computational science changed, and if they would recommend the workshop. In total, 17 participants responded to the close-ended questions.

Results. Post-workshop responses to questions about the efficacy of the workshop indicate that participants were generally satisfied with the workshop (Table 3). On average, participants rated the workshop components as Adequate-Good.

Participants generally felt the workshop met their needs and would recommend it to others. Participants gave the workshop a rating of 3.47 out of 4, indicating the workshop met their needs well to very well. 82% of responding participants felt their computational understanding changed, and 100% felt they learned what they hoped to learn. Finally, 100% of respondents indicated that they would recommend the workshop to others.

It is important to note here that the means and percentages cannot reflect the massive data loss in this survey study. Approximately half of participants did not return after lunch during the first day of the workshop and did not complete either the pre- or post-surveys. Then, another half of the remaining participants did not complete the post-survey because it was not allotted specific time during the workshop.

Table 1. Python Coding Ability (Pre)

Items	Communalities	Factor Loadings	Response Average	Response Meaning
I can write syntactically correct Python statements.	0.61	0.78	2.15 ±1.00	Disagree
I understand the language structure of Python.	0.44	0.67	2.22 ±0.98	Disagree
I can write logically correct blocks of code using Python.	0.43	0.66	2.19 ±0.97	Disagree
I can write a Python program that displays a greeting message.	0.55	0.74	2.48 ±1.15	Disagree
I can write a Python program that computes the average of three numbers.	0.64	0.8	2.45 ±1.12	Disagree
I can write a Python program that computes the average of any set of numbers.	0.64	0.8	2.21 ±1.05	Disagree
I can write a small Python program to solve a problem that is familiar to me.	0.76	0.87	2.30 ±1.08	Disagree
I can write a complex Python program to solve any given problem as long as the specifications are clearly defined.	0.23	0.48	1.55 ±0.62	Strongly Disagree
I can make use of a pre-written function if given a clearly labeled declaration of the function.	0.04	-0.2	2.76 ±0.94	Disagree
I cannot complete a programming project unless someone else helps me get started.	0.32	0.56	2.27 ±0.91	Disagree
I cannot complete a programming project unless someone shows me how to solve the problem first.	0.44	0.66	2.13 ±0.72	Disagree
I can complete a programming project if I have a lot of time to complete the program.	0.45	-0.67	3.03 ±0.53	Agree
I can complete a programming project if I have just the built-in help facility for assistance.	0.3	-0.55	2.71 ±0.59	Disagree
I can find ways of overcoming the problem if I get stuck at a point on a programming project.	0.25	-0.5	2.9 ±0.70	Disagree
I can come up with a suitable strategy for a given programming project in a short time.	0.32	-0.56	2.47 ±0.72	Disagree
I cannot complete a programming project unless I can call someone for help if I get stuck.	0.24	0.49	2.22 ±0.75	Disagree
I can mentally trace through the execution of a complex multi-file program developed by someone else.	0.14	-0.37	2.33 ±0.78	Disagree
I can rewrite confusing portions of code to be more readable.	0.41	-0.64	2.42 ±0.97	Disagree
I can find a way to concentrate on my program, even when there are many distractions around me.	0.28	-0.53	2.88 ±0.55	Disagree
I can find ways of motivating myself to program, even if the problem area is of no interest to me.	0.15	-0.39	2.72 ±0.63	Disagree
I can write a program that someone else can comprehend.	0.29	-0.54	2.94 ±0.61	Disagree

References. — Scale developed by Libarkin 2012.

Table 2. Python Coding Ability (Post)

Items	Communalities	Factor Loadings	Response Average	Response Meaning
I can write syntactically correct Python statements.	0.62	0.79	3.10 ±0.72	Agree
I understand the language structure of Python.	0.62	0.79	3.17 ±0.60	Agree
I can write logically correct blocks of code using Python.	0.72	0.849	3.17 ±0.60	Agree
I can write a Python program that displays a greeting message.	0.44	0.67	3.72 ±0.46	Agree
I can write a Python program that computes the average of three numbers.	0.28	0.52	3.79 ±0.41	Agree
I can write a Python program that computes the average of any set of numbers.	0.38	0.62	3.59 ±0.50	Agree
I can write a small Python program to solve a problem that is familiar to me.	0.51	0.71	3.34 ±0.55	Agree
I can write a complex Python program to solve any given problem as long as the specifications are clearly defined.	0.52	0.72	2.69 ±0.71	Disagree
I can make use of a pre-written function if given a clearly labeled declaration of the function.	0.38	0.61	3.52 ±0.51	Agree
I cannot complete a programming project unless someone else helps me get started.	0.29	-0.54	1.97 ±0.68	Strongly Disagree
I cannot complete a programming project unless someone shows me how to solve the problem first.	0.44	-0.66	1.93 ±0.65	Strongly Disagree
I can complete a programming project if I have a lot of time to complete the program.	0.61	0.78	2.69 ±0.66	Disagree
I can complete a programming project if I have just the built-in help facility for assistance.	0.39	0.62	3.28 ±0.65	Agree
I can find ways of overcoming the problem if I get stuck at a point on a programming project.	0.58	0.76	2.79 ±0.56	Disagree
I can come up with a suitable strategy for a given programming project in a short time.	0.29	0.54	3.14 ±0.52	Agree
I cannot complete a programming project unless I can call someone for help if I get stuck.	0.36	-0.60	2.76 ±0.58	Disagree
I can mentally trace through the execution of a complex multi-file program developed by someone else.	0.18	0.43	2.1 ±0.62	Disagree
I can rewrite confusing portions of code to be more readable.	0.42	0.65	2.79 ±0.56	Disagree
I can find a way to concentrate on my program, even when there are many distractions around me.	0.11	0.33	3.00 ±0.60	Agree
I can find ways of motivating myself to program, even if the problem area is of no interest to me.	0.12	0.35	2.93 ±0.53	Disagree
I can write a program that someone else can comprehend.	0.49	0.70	2.72 ±0.75	Disagree

References. — Scale developed by Libarkin 2012.

Table 3. Overall Workshop Impressions

	Minimum	Maximum	Response Average	Response Meaning	
Shell Training	3	5		3.76 ±0.56	Adequate
Python Training	3	5		4.35 ±0.61	Good
GitHub Training	2	5		3.71 ±0.85	Adequate
Debugging Training	3	5		3.88 ±0.70	Adequate
Testing Training	3	5		4.00 ±0.79	Good
Version Control Training	2	5		3.65 ±0.86	Adequate
Useful Unix Tools Training	1	5		3.53 ±1.18	Adequate
Documentation Training	3	5		4.12 ±0.70	Good
Overall rating of the workshop	4	5		4.47 ±0.51	Good
How well did this workshop meet your needs?	3	4		3.47 ±0.51	Well

3.2 Open-Ended Questions

The pre- and post-surveys asked participants to provide comments about their expectations for and opinions of the Software Carpentry workshop. The response rate was very low for the open-ended comments.

In the pre-survey, participants were asked the following: "Please provide any additional comments about your expectations for the workshop below." 11 participants responded to the question, but 2 of the responses were recommendations for future workshops (and therefore have been included in the post-data summaries). Four participants indicated interest in learning more about Python. One specifically expressed a desire to "get excited" about Python. Another student indicated interest in learning to analyze and plot data with programs; one person wanted to learn about how to use programming in research. One participant expressed interest in learning the difference between "scripting" and "programming." One student had previous experience in Java, with experience building complex multi-class programs. One student expressed interest in learning about version control. One student expressed interest in "filling in gaps" in self-taught programming.

In the post-survey, participants were asked two open-ended questions. The first was, "Did your understanding of computational science change because of your participation in this workshop?" 3 participants answered this question. Two students indicated increased understanding of how to use computational tools in research (specifically bash and unix). Two students expressed better understanding of git or version control. One student expressed a desire to learn more about the mathematics and physics underlying computational science.

The second question asked, "Please provide any additional suggestions or comments about the workshop below." 12 participants responded to this question (for a total of 14 analyzed here with the addition of 2 from the pre-survey).

Comments were generally positive, with five participants explicitly thanking the instructors or saying that the workshop was good. Three students indicated difficulty following instruction, one particularly expressed a feeling of "knowledge overload." Two students commented about the location of the workshop being unsatisfactory due to insufficient power plugs and the lack of tabletops for working. Two students expressed a desire for a higher level workshop; one specified a desire to learn more about modular programs while the other recommended having novices to homework prior to the workshop to allow for more complex exercises during the boot camp itself. One participant commented on the difficulty of setting up the virtual machine. Another participant suggested that new instructors be given more training before they are permitted to teach a boot camp. One student suggested that the boot camp have more advertisement. Another requested more shell and scripting lessons in future workshops. One student explicitly requested a session about how all of the topics in the basic skill set provided by Software Carpentry tie together and why they are the most necessary skills for computational scientists. Lastly, one participant requested coffee.

Results. In the qualitative data, participants expressed interest on the pre-survey in learning about or improving ability in Python coding, version control, and manipulating and plotting data. In the post-survey, participants expressed a variety of requests for improving University of Chicago venue-specific events, such as including tables and power outlets in the classroom, advertising the event more, and providing coffee for participants. One participant noted that the workshop moved very quickly and might be better suited to be spread out over four days. Another participant commented on the teaching ability of certain instructors and suggested that Software Carpentry instructors have some sort of training prior to teaching a workshop.

4 Recommendations

Recommendations for Software Carpentry and for the University of Chicago have been identified by this analysis.

First, students at the University of Chicago have indicated interest in the following:

1. Hosting similar workshops in rooms more conducive to laptop work
2. Vetting the instructors more closely before allowing them to teach a workshop
3. Requiring pre-workshop homework for beginners to allow a faster pace during the workshop itself

4. Hosting a more advanced workshop with a focus on modular programming practices

5. Hosting a longer workshop to allow more in-depth lessons

Recommendations for Software Carpentry Boot Camp assessment and evaluation are as follows:

1. Only 9 out of 29 surveys could be matched by participant for pre- and post-data. As a result, the paired-samples tests could not be conducted. This data loss can be partially avoided in the future by migrating the surveys to an online venue and requiring participants to include an email address for matching purposes in order to submit their feedback.

2. The Python Coding Ability scale appears to measure several constructs. The items may need to be filtered and refined to ensure that they target the desired construct.

3. No time was set aside during the workshop for assessment purposes, which may have impacted the low response rate. It is recommended that 15-20 minutes be set aside at the beginning and end of a workshop for the specific purpose of assessment.

4. The post-survey online link was not easily distributable to the audience. It is recommended that the links to both pre- and post-survey be posted on the workshop main website for accessibility.

5. The Computational Understanding scale has been modified from pre- to post-surveys, limiting the types of analyses that can be done on the data. It is necessary to standardize the questions across pre- and post-assessments to get the most useful information from participants.

6. The Perception of Computational Ability scale should be included on both the pre- and post-surveys.

5 Resources

This report cites the following sources:

1. Libarkin 2012 report on Software Carpentry at Michigan State University

2. Pickens 2013 report on Software Carpentry at Scripps Research Institute