Detecting Inconsistent Data Types

Lukas Blass (1950003), Eunice Jun (1623594) & Sam Kaufman (1022827)

Motivation
All empirical fields rely on valid statistical analyses. Unfortunately, conducting valid statistical analyses is challenging. There are two interconnected challenges: picking the correct statistical test to test a hypothesis and applying the test to the appropriate data. Tea [2] aims to address the former challenge by automating test selection. However, Tea requires the data to be in long format, a common denormalized format. Tea only guarantees correct application of statistical tests to data when this assumption is met. Any errors made during data formatting ("wrangling") prior to use in Tea programs goes unnoticed. Tea's current inability to detect semantically incorrect data in long format threatens the validity of its analyses, which ultimately undermines its goal. This is a problem present in other tools, such as R and Python packages (e.g., statsmodels) that assume data to be formatted in specific ways.

As an example, consider an analyst's task of comparing the prices of cars across two different years. Suppose the dataset contains the following columns: Car model, Price in 1990 (as measured in dollars - e.g., 100,000), and Price in 2000 (as measured in a percentage change from the price in 1990- e.g., -10).

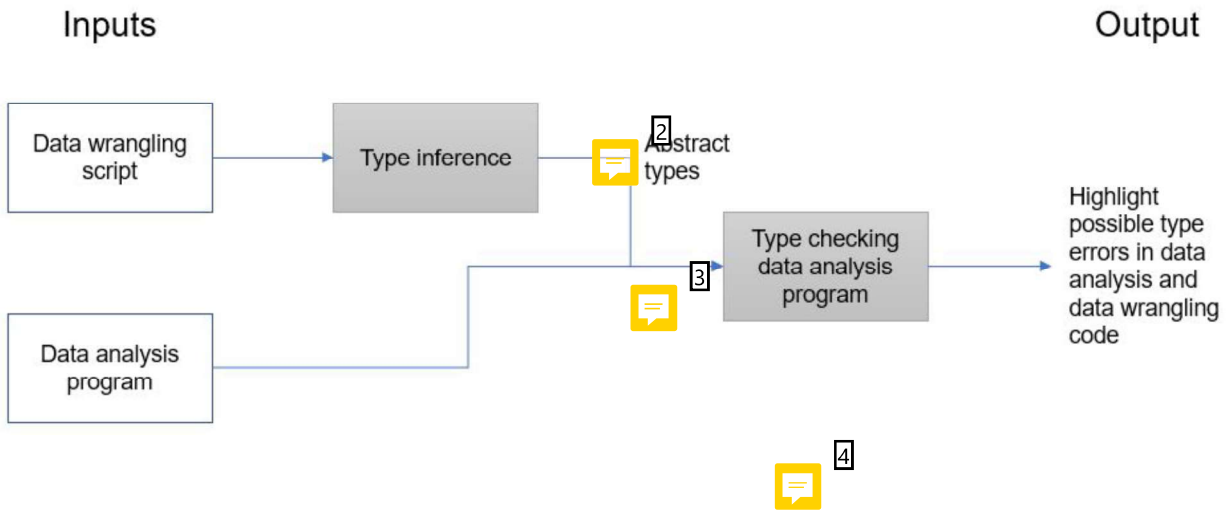
When the analyst reformats the dataset into long format, they collapse the two price columns into one and add a year column. The reformatted dataset now has the following columns: Car model, Price, and Year. However, the analyst forgot to change the units of the "Price in 2000." As a result, the resulting "Price" column has values of mixed units (implicit). All the values are still numbers, but they have semantically different meanings. Some refer to dollar prices. Others refer to percentages. In other words, data in the same column have incompatible *abstract types* [1].

Calling a statistical analysis function in R or Tea using the "Price" and "Year" columns still type checks and even executes, but the resulting analysis is invalid. The results are uninterpretable as the abstract types of data in the Price column are treated as dollar values, when they are really a mixture of dollar and percentage values.

Aim: We aim to surface potential errors in statistical analyses and data wrangling code to prevent inconsistent abstract types in data for the same variable/column.

Approach
We anticipate creating a linter for Python that could later be integrated into Tea. The linter would require two programs as input: 1) data wrangling code and 2) analysis code. As output, the linter would highlight lines of data wrangling and analysis code that are likely the source of error.

It operates in two steps on aforementioned inputs (data wrangling and data analysis programs). In a first step, abstract types of the columns of the input data are inferred using the data wrangling code. Every operation performed on the dataset reveals information about the types of the involved variables. For example, take the sum of two columns that provides us the information that both columns, as well as the result, must be of the same type. This process allows to refine the information about the types with every line of code. In the second step of the algorithm the knowledge about the abstract types is applied to the data analysis program and the user is alerted if inconsistencies with the inferred types are detected. This process is visualized in the diagram below.



A fundamental tension to this system design is found in the choice between the data wrangling or analysis code as the source of truth for the abstract type inference. We choose to infer abstract types from data wrangling code because we assume the wrangling code is more representative of users' intentions.

Given this problem, we pose the following research question and hypothesis:

Research question: How can using abstract types help analysts detect errors in statistical analysis and wrangling?
Hypothesis: Inferring abstract data types from data wrangling code will help analysts identify errors in statistical analysis and data wrangling more accurately and quickly than examining the wrangling and analysis code alone.

To scope our project, we propose to focus on data wrangling code that uses Pandas data frames to format/represent the data. Pandas is a commonly used library for data analysis [3]. While abstract type inference is a technique that already exists, we are unaware of any applications of the technique to the data science domain. We also focus on data analysis code that utilizes common Null Hypothesis Significance Testing.

Evaluation

Summary of Comments on juneunice_3650064_55752948_503 Project Proposal-commented.pdf

Page: 2

Number: 1 Author: mernst Subject: Sticky Note Date: 4/17/2019 12:49:12 PM

Start out by saying that you will compare the abstract types inferred for the two programs. That is, lead with the most important information rather than giving algorithmic details and only later summarizing the high-level goal.

Actually, that's not what you do. I'm not sure what "apply to the program" means, though. Use direct, clear language rather than vague terms. I am having trouble understanding your proposal.

Also recall that the only way to find errors is to compare two things. In your case, are you comparing some facts about the two programs against one another? Or are you comparing a programmer's expectations with facts about the program? If the former, you don't have to talk so much about the end user -- just say that the tool outputs potential errors. In any event, explain exactly what is being compared.

Number: 2 Author: mernst Subject: Sticky Note Date: 4/17/2019 12:51:23 PM

This inference will produce facts about the input and the output types (of columns in Tea or variables in Python). However, only the output types are relevant to the data analysis program. Therefore, it seems that you can only compare facts about the output of the data wrangling script against uses in the data analysis script. But the inconsistencies are about misuses of columns in the wrangling script input. If you only examine the output variables of the wrangling script, I don't see how you will find the sort of errors you gave in your example.

Number: 3 Author: mernst Subject: Sticky Note Date: 4/17/2019 12:41:34 PM

The dataflow does not make sense to me. I think you mean "abstract type inference" rather than "type inference". Why is that combined into a single input for the "type checking data analysis program" box rather than giving that box two inputs? Also, if they are combined, that implies they are the same kind of data, but a "data analysis" program is not the same as "abstract types".
I think you may mean that you add the inferred types to the program as one step, and then pass that annotated program to a type-checker. Its output is type errors. They might achieve the result of highlighting possible type errors, but that is not what the output is.

Number: 4 Author: mernst Subject: Sticky Note Date: 4/17/2019 12:52:29 PM

I suggest that you run your technique by hand on a couple of programs to better understand it and so you can make it more concrete. I'm not asking for that in the proposal, but you could mention the results.

Number: 5 Author: mernst Subject: Highlight Date: 4/17/2019 12:37:49 PM

Why do you need a source of truth? You are comparing two things, and any difference between them indicates a problem that a human must be called in to solve. I don't understand what problem you are getting at here.

How would your analysis results be different if you used the other data as the source of ground truth?

Number: 6 Author: mernst Subject: Highlight Date: 4/17/2019 12:42:54 PM

This is not a research question. You have a specific proposed approach and you want to evaluate whether it helps.

Number: 7 Author: mernst Subject: Highlight Date: 4/17/2019 12:43:19 PM

Do the analysts get to run the code and sanity-check the results?

We aim to evaluate our approach by conducting a user study. Our user study will involve a between-subjects study where half the participants try to detect errors in wrangling and analysis in Python scripts (from two corpora that Prof. Tim Althoff has) without using our linter and the other half use our linter. We will compare accuracy and time.

Integration with Tea

Our present focus is on Python scripts which wrangle data in Python and perform statistical tests using scipy, statsmodels, and other common libraries. We hope to explore integration with Tea if our approach proves promising.

References

[1] Guo,P., Perkins, J., McCamant, S., and Ernst, M. 2006. Dynamic inference of abstract types. In Proceedings of the 2006 international symposium on Software testing and analysis (ISSTA '06). ACM, New York, NY, USA, 255-265.

[2] Jun, E., Daum, M., Roesch, J., Chasins, S., Berger, E., Just, R., and Reinecke, K. (Under review) Tea: A High-level Language and Runtime System for Automating Statistical Analysis.

[3] Bobriakov, I. Top 15 Python Libraries for Data Science in 2017. https://medium.com/activewizards-machine-learning-company/top-15-python-libraries-for-data-science-in-in-2017-ab61b4f9b4a7

Page: 3

Number: 1 Author: mernst Subject: Highlight Date: 4/17/2019 12:45:03 PM

He has scripts, or he has scripts with errors?

Number: 2 Author: mernst Subject: Sticky Note Date: 4/17/2019 12:10:43 PM

You should cite "ExceLint: automatically finding spreadsheet formula errors" and possibly other work on detecting errors in spreadsheets.