Towards Exploratory Data Analysis for Pharo

Oleksandr Zaytsev
`olk.zaytsev@gmail.com`

Abstract

Data analysis and visualizations techniques (such as split-apply-combine) make extensive use of associative tabular data-structures that are cumbersome to use with common aggregation APIs (for arrays, lists or dictionaries). In these cases a fluent API for querying associative tabular data (like the ones provided by Pandas, Mathematica or LINQ) is more appropriate for interactive exploration environments. In Smalltalk despite the fact that many important analysis tools are already present (for *e.g.,* in the PolyMath library), we are still missing this essential part of the data science toolkit. These specialized data structures for tabular data sets can provide us with a simple and powerful API for summarizing, cleaning, and manipulating a wealth of data-sources that are currently cumbersome to use. In this paper we introduce the `DataFrame` and `DataSeries` collections - that are specifically designed for working with structured data. We demonstrate how these tools can be used for descriptive statistics and exploratory data analysis - the critical first step of data analysis which allows us to get the summary of a data set, detect mistakes, determine the relations, and select the appropriate model for further confirmatory analysis. We then detail the implementation trade-offs that we are currently facing in our implementation for Pharo and discuss future perspectives.

1 Introduction

The simplicity and power of Smalltalk combined with the live environment of Pharo creates a productive combination for data analysis. Provided the proper tools and open source libraries for machine learning, statistics, and optimization, Pharo can become both a powerful tool for professional data analysts, and a simple environment for everyone who wants to experiment with a simple data set. Many important tools and algorithms are already implemented in libraries such as PolyMath, but we are still missing essential data-structures for tabular data and a fluent API for advanced data-analysis techniques. To overcome this problem we introduce in this paper the `DataFrame` and `DataSeries` collections for working with structured data, and through several examples, we demonstrate how these tools can be used for descriptive statistics and exploratory data analysis allowing us to get the summaries of data sets, detect mistakes, determine relations, and select the appropriate models for further analysis.

All pictures in this paper are the built-in DataFrame visualizations that are created with Roassal2 and can be reproduced by following the steps in Section 3

The rest of this paper is structured as follows: Section 2 provides a brief introduction into explanatory data analysis, answering the following questions: *What is EDA good for and how to do it right ?*. It also provides basic knowledge about statistics and data analysis, such as: statistical variables, types of variables etc. Then Section 3 details the new DataSeries and DataFrame collections for structured data. Section 4 gives a step-by-step example of how to perform EDA on the well-known Iris data set using the new collections and API. Finally Section 5 concludes the paper and discusses implementation trade-offs as well as future perspectives.

2 Exploratory Data Analysis

Exploratory data analysis (EDA) is an approach to analyzing data sets to summarize their main characteristics. It allows us to make some sense of the data by visualizing it and exploring its statistical properties. According to Howard J. Seltmanan, any method of looking at data that does not include formal statistical modeling and inference falls under the term exploratory data analysis [3]. It is an important first step of data analysis which helps us to select the model that we will be fitting to the data during the following steps of confirmatory data analysis.

EDA can be particularly useful for uncovering underlying structure of a dataset, detecting outliers and anomalies, determining relationships among the explanatory variables, assessing the direction and rough size of relationships between explanatory and outcome variables, and selecting applopriate models for further analysis [2].

By the number of variables and the ... the techniques of exploratory data analysis can be cross-classified into four types: univariate graphical, univariate non-graphical, multivariate graphical, and multivariate non-graphical.

3 DataSeries and DataFrame

DataSeries and DataFrame are the high-level data structures that make data analysis in Pharo fast and easy. [1] DataFrame can be loaded into a Pharo image with the following Metacello script:

```
Metacello new
   baseline: 'DataFrame';
   repository: 'github://PolyMathOrg/DataFrame';
   load.
```

DataSeries is an ordered collection that combines the properties of both Dictionary and Array, together with some extra functionality. It has a name and contains an array of data mapped to the corresponding array of keys (index values).
The easiest way of creating a series is by converting it from an array.

```
series := #(a b c) asDataSeries.
```

The keys will be automatically set to the numeric sequence which can be described as an interval (1 to: n), where n is the size of array, and the name will remain empty. Both name and the keys of a DataSeries can be changed later:

```
series name: 'letters'.
series keys: #(a1 a2 a3).
```

DataFrame is a tabular data structure and an ordered collection of columns. It works like a spreadsheet or a relational database with one row per subject and one column for each subject identifier, outcome variable, and explanatory variable. DataFrame has both row and column indices which can be changed if needed. The important feature of a DataFrame is that whenever we ask for a specific row or column, it responds with a DataSeries object that preserves the same indexing.
A simple DataFrame can be created from an array of rows or columns.

```
df := DataFrame rows: #((John 25 true)(Jane 21 false)).
df := DataFrame columns: #((John Jane)(25 21)(true false)).
```

Those two line produce exactly the same DataFrame. Once again, the names (key values) of both rows and columns were not explicitly specified, so by default they will set to numerical indices: #(1 2) for rows and #(1 2 3) for columns. These names can be changed later:

```
df columnNames: #(Name Age IsMarried).
```

4 Exploring Iris Data Set

We can start by loading iris data set from a CSV file.

```
data := DataFrame fromCsv: '/path/to/iris.csv'.
```

DataFrame comes with a built-in collection of data sets that are widely used as examples for data analysis and machine learning problems. Iris is among them, so an alternative way of loading it would be simply

```
data := DataFrame loadIris.
```

Now let's take a look at the first and the last 5 entries in our table. These slices are called *head* and *tail* of a data frame.

```
data head.
data tail.
```

The output will be the following table

```
1  (5.1 3.5 1.4 0.2 #setosa)
2  (4.9 3 1.4 0.2 #setosa)
3  (4.7 3.2 1.3 0.2 #setosa)
4  (4.6 3.1 1.5 0.2 #setosa)
5  (5 3.6 1.4 0.2 #setosa)
```

4.1 Univariate non-graphical EDA

To access a single variable we ask a data frame for a specific column, using its name or number. The result will be a DataSeries object.

```
series := data column: #sepal_width.
series := data columnAt: 1.
```

What we can do with a column depends on a type of statistical variable it represents. The best univariate non-graphical EDA for categorical data is a simple tabulation of the frequency of each category [3]. If the data is quantitative, we can ... `min, max, range, average, median, mode, stdev, variance`

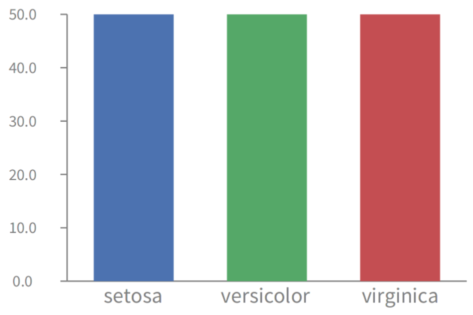
```
series average.
series stdev.
```

4.2 Univariate graphical EDA

4.2.1 Categorical

Histogram is the only graphical technique that can be used for a categorical variable.

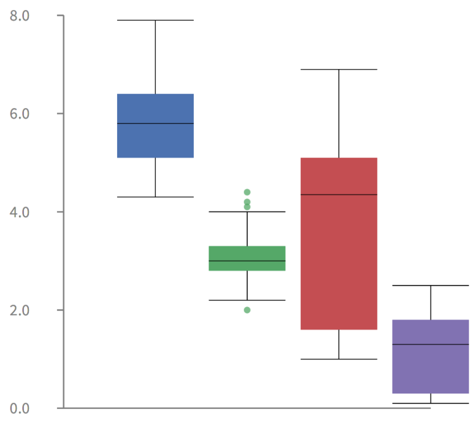
```
var := data column: #species.
var histogram.
```



4.3 Multivariate non-graphical EDA

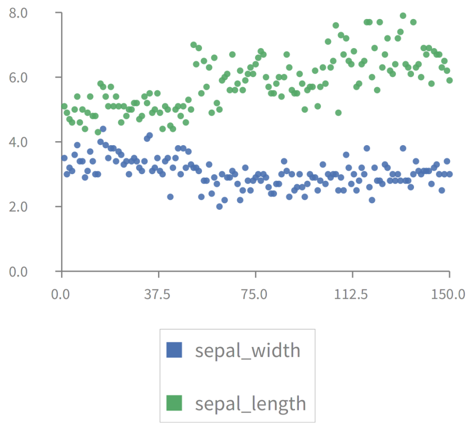
Multivariate non-graphical EDA shows the relationship between two variables in form of either cross-tabulation (categorical data) or statistics (quantitative data).

4.4 Multivariate graphical EDA



Let's look at the scatterplot of two statistical variables representing the width and length of a sepal. To do that we ask our DataFrame to give us specific columns, in our case, `sepal_width` and `sepal_length`, then we ask these columns (the result will be another DataFrame) to visualize themselves.

```
vars := data columns: #(sepal_width sepal_length).
vars scatterplot.
```



5 Conclusion & Future work

At the time of writing this paper DataFrame is capable of... However, a lot of functionality is still missing. For example, we need tools for

- data wrangling
- data aggregation and grouping

References

[1] Wes McKinney. *Python for Data Analysis*. O'Reilly Media Inc., 2012.

[2] NIST/SEMATECH. *e-Handbook of Statistical Methods*. http://www.itl.nist.gov/div898/handbook/. Accessed: 2017-06-20.

[3] Howard J. Seltman. *Experimental Design and Analysis*. 2015.