Property-Based Testing of Data Analysis Scripts
A Focus on Hypothesis for Python

Jean-Sebastian de Wet, Jan-Philipp Kiel, and Pascal Mager

University of Cologne, Cologne, Germany

Abstract. This paper explores property-based testing as a method to ensure data analysis scripts' reliability, especially in DLR research using Python, Pandas, and Matplotlib. It outlines challenges with traditional testing in scenarios with diverse data values, emphasizing the need for innovative testing strategies. The paper thoroughly covers property-based testing, including its history, key principles, use cases, and integration in the test pyramid. It then focuses on Hypothesis for Python, a powerful tool for property-based testing, discussing its use, integration with pytest, and unique features. Real-world application is demonstrated with code examples, highlighting how property-based testing, especially with Hypothesis, strengthens data analysis scripts' reliability. The paper concludes by summarizing key findings and emphasizing the crucial role property-based testing, like Hypothesis, plays in boosting researchers' confidence with unknown data.

Keywords: Property-Based Testing · Data Analysis Scripts · Hypothesis · Python · pytest · Reliability · Test Pyramid · Code Examples · DLR Research.

1 Introduction

- Briefly introduce the context of data analysis in DLR and the challenges associated with traditional testing approaches.
- Highlight the need for property-based testing in scenarios with large possible value ranges.[?]

2 Background

- Discuss the common data analysis tools used by DLR researchers (Python, Pandas, Matplotlib).
- Explain the challenges of writing tests for data analysis scripts.
- Emphasize the importance of confidence in the face of unknown data.

3 Overview of Property-Based Testing

3.1 History of Property-Based Testing

- Trace the historical development of property-based testing.

3.2 How Property-Based Testing Works

- Explain the core concept of property-based testing.
- Discuss the idea of defining invariants for functions.

3.3 Main Use Cases

- Explore various scenarios where property-based testing is particularly useful.
- Discuss how it complements traditional testing approaches.

3.4 Position in the Test Pyramid

- Describe where property-based testing fits in the test pyramid.

3.5 Tools and Programming Languages

- Provide an overview of existing property-based testing tools.
- Highlight tools for different programming languages.

4 Introduction to Hypothesis for Python

4.1 Overview of Hypothesis

- Provide a brief introduction to Hypothesis for Python.
- Mention its key features and advantages.

4.2 How to Use Hypothesis

- Write a mini how-to guide on using Hypothesis, including integration with pytest.
- Include code snippets for better understanding.

5 Main Concepts and Features of Hypothesis

5.1 Strategies and Data Generation

- Explain the concept of strategies in Hypothesis for generating test data.

5.2 Property-Based Testing with pytest

- Detail how Hypothesis integrates with pytest.
- Provide examples of test functions using Hypothesis.

5.3 Data Analysis Applications and Benefits

- Discuss how data analysis applications can benefit from Hypothesis.
- Reference specific features, such as the support for NumPy.

6 Application of Hypothesis in Data Analysis

6.1 Code Examples

- Provide practical code examples demonstrating the use of Hypothesis in data analysis scripts.
- Showcase scenarios where property-based testing adds value.

```
import numpy as np

def incmatrix(genl1,genl2):
    m = len(genl1)
    n = len(genl2)
    M = None #to become the incidence matrix
    VT = np.zeros((n*m,1), int)  #dummy variable

    #compute the bitwise xor matrix
    M1 = bitxormatrix(genl1)
    M2 = np.triu(bitxormatrix(genl2),1)

    for i in range(m-1):
        for j in range(i+1, m):
            [r,c] = np.where(M2 == M1[i,j])
            for k in range(len(r)):
                VT[(i)*n + r[k]] = 1;
                VT[(i)*n + c[k]] = 1;
                VT[(j)*n + r[k]] = 1;
                VT[(j)*n + c[k]] = 1;

                if M is None:
                    M = np.copy(VT)
                else:
                    M = np.concatenate((M, VT), 1)

                VT = np.zeros((n*m,1), int)

    return M
```

6.2 Illustrative Cases

- Present specific cases where Hypothesis helped discover issues in data analysis scripts.

7 Conclusion

- Summarize the key points discussed in the paper.

- Emphasize the importance of property-based testing, particularly with tools like Hypothesis, in enhancing the reliability of data analysis scripts.

References