Date Recency Assessment Technical Guide

# Introduction

This technical guide outlines the functionality and integration of the Date Recency Assessment Tool within a data quality framework for RDF datasets. The function automates the assessment of observation dates to classify them as 'recent' or 'outdated', contributing to the overall quality and reliability of the data.

# Requirements

Before you begin, ensure that the following prerequisites are met:

- Python environment (Python 3.x recommended)

- Understanding of RDF data structure and the Resource Description Framework

# Implementation

The core functionality is encapsulated within two primary methods:

`assess\_observation\_date\_recency` for assessing date recency and `is\_date\_recent` for determining if a date falls within the recent threshold. The assessment process involves iterating through RDF triples, extracting observation dates, and applying the recency check.

## Code Snippets

Assess Observation Date Recency Method:

```python  
def assess\_observation\_date\_recency(self):

total\_assessments = 0  
 namespace = self.label\_manager.get\_namespace('date\_recency')  
 result\_counts = {'recent': 0, 'outdated': 0}  
  
 for s, \_, o in self.g.triples((None, SOSA.phenomenonTime, None)):  
 if BaseUris.check\_base\_uri(str(s), BaseUris.OBSERVATION):  
 date\_within\_range = False  
 for \_, \_, date\_literal in self.g.triples((o, TIME.inXSDDate, None)):  
 if DateTimeDataQuality.is\_date\_recent(date\_literal):  
 result\_counts['recent'] += 1  
 date\_within\_range = True  
 else:  
 result\_counts['outdated'] += 1  
 break  
  
 total\_assessments += 1  
 result\_label = namespace['recent'] if date\_within\_range else namespace['outdated']  
  
 self.\_add\_assessment\_result(s, AssessmentTypes.OBSERVATION\_DATE\_RECENCY, result\_label)  
  
 self.add\_to\_report('Date Recency Assessments', total\_assessments, result\_counts)  
```

Is Date Recent Method:  
```python  
def is\_date\_recent(date\_literal, years\_back=20):  
 if date\_literal.datatype == XSD.date:  
 current\_year = datetime.now().year  
 date\_year = date\_literal.toPython().year  
 return (current\_year - years\_back) <= date\_year <= current\_year  
 return False  
```

# Conclusion

The Date Recency Assessment function provides a robust solution for enhancing the temporal accuracy of RDF datasets. By following this guide, developers and data scientists can efficiently integrate date recency assessments into their data quality workflows, ensuring their datasets remain relevant and reliable.