Self-Reflection: Creating a changelog

TOTAL POINTS 1

1. 1 / 1 point

Overview

Now that you have learned about the importance of keeping track of changes in your data analysis, you can pause for a moment and track what you are learning. In this self-reflection, you will consider your thoughts about changelogs and respond to brief questions.

This self-reflection will help you develop insights into your own learning and prepare you to incorporate changelogs into your data cleanings procedures. As you answer questions—and come up with questions of your own—you will consider concepts, practices, and principles to help refine your understanding and reinforce your learning. You've done the hard work, so make sure to get the most out of it: This reflection will help your knowledge stick!

The importance of changelogs

In previous activities, you've reviewed the different types of questions to ask before exploring data, the importance of pre-cleaning data, the basic functions of SQL, how to clean data with spreadsheets, and more. As a junior data analyst, most of your projects will consist of these activities. As you have experienced, each of these tasks follows a complicated process. Therefore, consistent and accurate record-keeping is essential to keeping you on track.

A **changelog** is a document used to record the notable changes made to a project over its lifetime across all of its tasks. It is typically curated so that the changes it records are listed chronologically across all versions of the project.

The major benefit to using changelogs is that contributors and users connected with the project get a specific list of what important alterations have been made, when they were made, and sometimes, what version they were released for. It is an invaluable tool for communicating how the project has evolved over time to coworkers, management, and stakeholders.

Best practices for changelogs

A changelog for a personal project may take any form desired. However, in a professional setting and while collaborating with others, readability is important. These guiding principles help to make a changelog accessible to others:

- Changelogs are for humans, not machines, so write legibly.
- Every version should have its own entry.
- Each change should have its own line.
- Group the same types of changes. For example, *Fixed* should be grouped separately from *Added*.
- Versions should be ordered chronologically starting with the latest.
- The release date of each version should be noted.

All the changes for each category should be grouped together. Types of changes usually fall into one of the following categories:

- Added: new features introduced

- Changed: changes in existing functionality
- Deprecated: features about to be removed
- Removed: features that have been removed
- Fixed: bug fixes
- Security: lowering vulnerabilities

Examine a sample changelog

Examine the figure below for an example of a changelog. Note that the following example is written in Markdown, as it is common to keep changelogs as a readme file in a code repository.

```
# Changelog
This file contains the notable changes to the project

Version 1.0.0 (02-23-2019)
## New
    - Added column classifiers (Date, Time, PerUnitCost, TotalCost, etc. )
    - Added Column “AveCost” to track average item cost

## Changes
    - Changed date format to MM-DD-YYYY
    - Removal of whitespace (cosmetic)

## Fixes
    - Fixed misalignment in Column "TotalCost" where some rows did not match with correct dates
    - Fixed SUM to run over entire column instead of partial

```

What to record in a changelog

Now that you're familiar with the example, consider what changes you need to record in a changelog. To start, you record the various changes, additions, and fixes that were discussed above. Arrange them using bullets or numbering with one change per line. Group similar changes together with a label describing the change immediately above them.

Use different version numbers for each milestone reached in your project. Within each version, place the logged changes that were made since the previous version (milestone). Dates are not generally necessary for each change, but they are recommended for each version.

In an upcoming course, you will have the opportunity to complete a capstone project. This will be a great chance to demonstrate your ability to organize a project like a professional data analyst by keeping your own changelog.

You can do this using a simple text file or spreadsheet and include your changelog with the project write-up. It will help you stay organized and collaborate with others. Keep this in mind when you reach the capstone project in an upcoming course, and don’t be afraid to revisit this lesson if you have questions.

Reflection

Consider what you reviewed about changelogs in this reflection:

- What makes for a good changelog?
- How do you decide if a change is significant enough to include in the changelog?

Now, write 2-3 sentences (40-60 words) in response to each of these questions. Type your response in the text box below.

A good changelog is one that is easy to understand by everyone within and outside the team. Every change need to have its own line and owner and the date the change is made

Correct

Great work reinforcing your learning with a thoughtful self-reflection! A good reflection on this topic would include how an effective changelog indicates the notable changes to a project.

A changelog should capture any of the following changes to the dataset while cleaning:

- Treated missing data
- Changed formatting
- Changed values or cases for data

You have made some of these changes while cleaning data in previous activities. If you had kept a changelog during those activities, you would have described and categorized each change. When in doubt about the significance of a change, you should enter it into the changelog.