Easily Edit DataFrames in JupyterLab and Track Your Changes

Pandas is powerful and very flexible, but it does have some weaknesses. One big point of friction I ran into is the inability to double-click on a cell and change its value. Below I will detail the context of my problem, explain how I customized my JupyterLab notebook to improve my workflow, and share the code I wrote for you to use as you see fit.

# Background

I started learning Python nearly a year ago. The timing was right for me: a friend showed me some really interesting analyses he did with Python for his dissertation on the role of genre in popular music <<https://tom-johnson.net/2017/10/09/genre-part-2-genre-tags/>>, and I started a great new job with a boss who encouraged me to pursue data science and statistics.

I started learning Python and data science by reading some great books <[www.goodreads.com/book/show/31670583-think-like-a-data-scientist](http://www.goodreads.com/book/show/31670583-think-like-a-data-scientist)> <<https://www.goodreads.com/book/show/14744694-python-for-data-analysis>> <[www.goodreads.com/book/show/26457146-python-data-science-handbook](http://www.goodreads.com/book/show/26457146-python-data-science-handbook)> and taking a number of MOOCs, but a number of people have pointed out that the best way to learn is to start working on a project that interests you <<https://www.aitimejournal.com/@admond.lee/3-steps-to-build-a-data-science-portfolio>>. I have been interested in college football since my freshman year of college (my team’s lousy past couple <<https://fbschedules.com/ncaa-2017/team/florida-state>> seasons <<https://fbschedules.com/ncaa-2018/team/florida-state>>), and I decided to use my knowledge of the sport to build a computer poll/predictive model.

I’ve wanted to make a computer poll for a few years, but I didn’t know where to start. I turned to some active communities for ideas: the College Football <<https://www.reddit.com/r/cfb>> and CFBAnalysis <<https://www.reddit.com/r/CFBAnalysis/>> subreddits. r/CFB has a slate of user-created computer polls every season, and r/CFBAnalysis is a smaller, statistically-minded community that serves as a great place to share resources and cross-pollinate ideas. At this point I had a vague idea of what I wanted to do and some very rudimentary Python skills. All I needed was a dataset.

Reddit user BlueSCar maintains a fantastic free database of college football statistics <<https://collegefootballdata.com>>. He gathers a wide range of data from ESPN and other sources, including play-by-play data for each game, drive summaries, matchup information, game location, recruiting rankings, and conference affiliation. It is an invaluable repository for anyone interested in exploring college football data. There is one big issue, however: ESPN’s data sometimes <<http://www.espn.com/college-football/playbyplay?gameId=401012797&wsVar=us~ncf~gamepackage,desktop,en>> have <<http://www.espn.com/college-football/playbyplay?gameId=401019511&wsVar=us~ncf~gamepackage,desktop,en>> problems <<http://www.espn.com/college-football/playbyplay?gameId=401012776&wsVar=us~ncf~gamepackage,desktop,en>>.

I won’t go into great detail about my methodology here, but one of the main measures I want to derive and model is how a team performs while they are actively trying to win with their best players and best tactics. To do this, I need filter out any garbage time <<https://en.wikipedia.org/wiki/Garbage_time>> plays. I am using college football analytics guru Bill Connelly’s definition <<https://www.footballstudyhall.com/2017/10/20/16507348/college-football-analytics-game-states>> of garbage time, defined as point differentials of 36 in the second quarter, 26 in the third quarter, and 20 in the fourth quarter. In order to properly filter out garbage time and get accurate measurements of each team, I need to start with correct statistics.

# Friction in Data Cleaning

I am using CSV files of play-by-play data downloaded from [collegefootballdata.com](http://collegefootballdata.com) as my raw data source. I noticed pretty quickly that there are a number of impossible, mislabeled, and flat-out incorrect plays that I need to fix in order to be satisfied with the results of this project. Additionally, I want to keep a record of which plays I changed to send back to the website so the community as a whole has cleaner data for more accurate analysis. My original workflow for fixing the data was:

* identify and isolate rows that met specific criteria with Pandas
* export the rows to a CSV file and drop them from the DataFrame
* open the CSV in Excel
* find play-by-play data from a different source (usually the school’s athletics website)
* fix the incorrect information
* import the CSV
* merge the two DataFrames

There’s a lot of friction here. Pandas lets you filter data in pretty much any way imaginable, but you need to type all of your criteria. I longed for an interactive way to sort, filter, and edit data directly inside a JupyterLab notebook. I found exactly that when I stumbled across Parul Pandey’s terrific Medium post about enhancing Jupyter notebooks <<https://towardsdatascience.com/bringing-the-best-out-of-jupyter-notebooks-for-data-science-f0871519ca29>>.

# New Workflow

I dove into Qgrid’s documentation and GitHub page as soon as I saw this gif <[https://cdn-images-1.medium.com/max/800/1\*xGvLi1RhdIdSbCA9cPsS2A.gif](https://cdn-images-1.medium.com/max/800/1*xGvLi1RhdIdSbCA9cPsS2A.gif)>. Qgrid’s GitHub page shows off additional functionality you can get with the Qgrid JupyterLab extension and Output widgets from ipywidgets <<https://ipywidgets.readthedocs.io/en/stable/>>.

Qgrid showed me something I didn’t even know I wanted: the ability to display the entirety of a selected row in a separate output view. The DataFrame I’m working with is 24 columns, one of which contains long strings describing each play. Here’s an example:

Kellen Mond pass complete to Quartney Davis for 17 yds for a TD Timeout TEXAS A&M, clock 00:00 LSU Penalty, Defensive Pass Interference (2 Yards) to the LSU 1 LSU Penalty, Unsportsmanlike Conduct ( Yards) to the LSU 1 TEXAS A&M Penalty, False Start (-4 Yards) to the LSU 5 (Kellen Mond pass to Kendrick Rogers for Two-Point Conversion)

A separate view that shows an entire row would save me a lot of scrolling and keep relevant information in a consistent location in the notebook.

At this point I am extremely interested, and pretty confused by the new APIs I have to learn. I spend around 6 hours reading the documentation for both packages and exploring the Qgrid binder demos while hacking together some code that doesn’t work the way I want it to. After taking a break to clear my head, I’m finally able to create the interface below.

I’d like to point out some key features. The right output view displays the row as it exists when you first select it. I changed the yards\_gained and the play\_text columns in row 530, but the changes are not yet reflected in the output view. They do appear in the output view if I select a different row and reselect row 530.

The bottom output view shows the coordinates of the edited cell (name of the DataFrame you edited followed by [row name, column name]), the old value, and the new value. This widget keeps track of changes you make in any Qgrid DataFrame in the notebook and reflects this in the coordinates.

The edits are stored in a DataFrame. You can export them to a CSV file by typing a name for the file in the text box and clicking the ‘Export to CSV’ button. The file will export to the directory the notebook is in. Exporting does not automatically clear edits; click the ‘Clear edits’ button to do that.

# Code

The general steps to building these interactive widgets are:

* 1. Instantiate the output widgets you want to use from ipywidgets package
  2. Write functions telling Qgrid what to do when certain events happen
  3. Write Qgrid event handlers that activate the functions when events occur

I’ll walk through the specific implementation of these steps in the current row and edits widgets.

First, start with the proper imports. Pandas display options are used by the current row widget. These settings are appropriate for the data I’m working on; customize as necessary.

<https://gist.github.com/likethebourbon/f6932d45b16bdb4604098089c9b53081#file-editing-workflow-1-py>

Next, create the output widget that will display the currently selected row. It is an Output widget with some layout options specified. The `overflow` options allow for scrolling within the widget if necessary. I am not sure if all three options (`overflow\_x`, `overflow\_y`, and `overflow`) are all necessary, but after some tooling around this is what gave me the behavior I wanted.

That output widget is just a pane, so it is simple to create. The