HW2 Write-Up

Zayn Khan

2025-02-24

Amazon Products Analysis Shiny App Write-Up [HW2]

Script

Link to R script: https://github.com/Zeke07/STAT436/blob/main/hw2/hw2.R (https://github.com/Zeke07/STAT436/blob/main/hw2/hw2.R)

shiny.io application (not recommended since it takes too long to upload): https://zakhan5.shinyapps.io/amazon_app_hw2/ (https://zakhan5.shinyapps.io/amazon_app_hw2/)

The Dataset

For this application, I uncovered a data-set for a collection of Amazon Products. The intent was to simulate the Amazon shopping experience, with added insights. It took 20-30 seconds to read the data from an endpoint in order to launch the app whilst connected to my partner's internet (I will cut myself some slack, as her WiFi is not particularly fast, but just keep that between us, fellow reader!). The canonical attributes were present. Namely, `stars`, `reviews`, `price`, `listPrice`, `isBestSeller`, and `boughtInLastMonth`, which give us a sense for the rating, number of reviews, the current price, the original price (in the dataset, this is 0 if the product is currently not on sale), a boolean specifying if the product is a best-seller, and the number of products sold in the last month. All of these together give a sense for how the product is received in the consumer market, and provides some statistical bearing for whether an item will fare better or worse in general.

Key Metrics

In my mind, there are a few things that I look at when I cross-reference multiple products of the same variety when shopping on Amazon: the ratio between the number of reviews to the rating/stars of a product, the number of items sold in the last month, as well the magnitutde of the price markdown if that product happens to be on sale. I wanted to devise a plot that summarized these three criteria to fast-track the process of product selection.

Plot and Shiny UI

For this plot, I have introduced a scoring metric to handle the collation of all the characteristics I mentioned earlier. I refer to this as the `Product Score`, which is a simple formula:

$$\text{Rating} * \text{Reviews} + \text{Rating}$$

Product rating does not really make sense unless people have provided feedback on the product. I gravitate toward products with high ratings coupled with more reviews. This “score” will be the y-axis of the plot.

In the plot, I overlay two scatterplots:

- **Red points** denote the current (discounted) price.
- **Black points** denote the original price.
- **Purple-dashed lines** connect these points to convey the markdown (if present).

Note that the affect of no discount will merge these two points

Point size represents the number of products sold in the last month, with dynamic breaks and labels ensuring the sizes are appropriate even when queries return different ranges.

This collation of metrics enables users to quickly compare products within their search criteria.Coalescing these metrics, alongside comprehensive filters, will help us digest this enormous dataset. On that note, The universal settings button provides several basic filtering options on the dataset itself: ranges for attributes such as pricing, booleans, a search bar. An interesting **Discovery** which assisted me in condensing the Shiny code was off-loading the setting popover definition to a function. A hover function is provided to display product information for particular points (the `page_sidebar` was a near replacement for `fluidPage`). I reserved a collapsable sidebar that displayed product info (such as title, dynamically loaded image, discounts, ratings, etc) for any single point on the plot. For readability, the user can switch tabs between the plot as well as a data table, and a brushing feature is provided, also reflecting on the data table when activated. **Also**, I did not realize the reactive function would revert the filtered data if you cleared the brush selection by clicking away from the plot. Finally, the data table can be queried as shown in class, with hyperlinks added to re-direct to the official product page.

Data Preparation

Each product in `amazon_products.csv` comes with a `category_id` that is linked to a category name via `amazon_categories.csv` (using an inner join). This join is essential for filtering by category. Additionally, I wrapped the product URL in HTML to display it as a hyperlink, and I adjusted the **listPrice**: if the product wasn't on sale, listPrice was set to 0 and then reassigned to the current price to avoid confusing the UI. A new **sale** column was also added based on these conditions.

Reactive Graph Structure of the Shiny App

The following diagram represents the reactivity in our Shiny application, starting from the initial inputs. The **user inputs** trigger filtering operations, which then update either the the **markdown_df()** or **hover_data()** reactive datasets. This dataset is used to render both the **plot** and **table**, while hover interactions modify the **UI siderbar** of the selected product: