

Library Checkouts for the Top 500 “Greatest” Novels

Neel Gupta

2026-02-25

Data Essay

Introduction

This dataset contains library circulation information for books in the [Top 500 “Greatest” Novels](#)—that is, the novels most widely held in libraries according to [OCLC](#), a major library consortium.

The library checkout data comes from the city of Seattle. The Seattle Public Library’s (SPL) [open checkout data](#) is one of the only publicly available sources of book reception data in the country (Gupta, Christensen, et al. 2025; Walsh 2022). The dataset presented here is a combination of both the [Top 500 “Greatest” Novels](#) and a [mirrored version of the SPL’s open checkout data](#), recording checkouts from 2005 until February 2025.

Brief Survey

If you use our materials in your class or another setting, we would love to [hear about it!](#)

Which Books Are Most Popular?

```
//|echo: false
import {viewof tableTopTitles, plot, headingBlock, viewof selectionInput, viewof tableTopTit.
```

```
//|echo: false

viewof search
viewof tableTopTitlesMode
```

```
//|echo: false

headingBlock
plot
viewof selectionInput
```

Dataset

```
//|echo: false
// Using the Post45 Data Viewer
// https://observablehq.com/d/8bb63a6cde9addff
import {viewof dataSummaryView, Tabulator, viewof selectedColumns, viewof dataSet, tableCont
```

```
//|echo: false
//|output: false

// Using the Post45 Data Viewer
// https://observablehq.com/d/8bb63a6cde9addff
generateTabulatorTableFromCSV(
  "#table-container4",
  "https://responsible-datasets-in-context.s3.us-west-2.amazonaws.com/top_500_spl_df.csv",
  {
    displayedColumns: [
      // Top 500 info and checkouts
      "top_500_title",
      "author",
      "materialtype",
      "year-month",
      "checkoutyear",
      "checkoutmonth",
      "checkouts",
      'top_500_rank',
      'pub_year',
      'genre',

      // Specific SPL Record
      "title",
      "creator",
```

```

    // Long description
  ],

  columnPopups: [
    "Title of the matched Top 500 'Greatest' Novel",
    "Corresponding Author from the Top 500 'Greatest' Novels",
    "Material type",
    "Year and month of checkout",
    "Year of Checkout",
    "Month of Checkout",
    "# of Checkouts of the SPL entry in the given month and year",

    "Rank in the Top 500 'Greatest' Novels",
    "Year of Publication",
    "Assigned Genre label from the Top 500 'Greatest' Novels",

    "Title in the SPL's open checkout data",
    "Creator in the SPL's open checkout data",

  ],

  // columnWidths: {
  //   governor: "220px",
  //   party: "140px",
  //   years_in_office: "220px",
  //   school: "220px",
  //   bio_text: "40%"
  // },

  // Treat these as numeric so you get proper sort/filter ranges
  numericColumns: ["checkouts", "top_500_rank", 'pub_year'],

  // Good facets for filtering
  categoryColumns: [
    "top_500_title",
    "author",
    'genre',
    "materialtype"
  ],

  // Default sort: by state, then governor

```

```

sortColumns: ["state_territory", "governor"],
sortOrders: ["asc", "asc"],

// Buttons (kept the same IDs you used)
buttonContainerId: "#button-container1",
rawButtonId: "#download-raw1",
urlCopyButtonId: "#copy-url1",

// Optional: if your helper supports label maps for 1/0 columns, this is handy.
// booleanLabelColumns: {
//   college_attendance: { "1": "Yes", "0": "No" },
//   ivy_attendance: { "1": "Yes", "0": "No" },
//   lawyer: { "1": "Yes", "0": "No" },
//   military_service: { "1": "Yes", "0": "No" },
//   born_in_state_territory: { "1": "Yes", "0": "No" },
//   intl_born: { "1": "Yes", "0": "No" }
// }
}
);

```

[Download Full Data](#)

[Download Full Data](#)

[Copy Full Data URL](#)

[Download Table Data \(including filtered options\)](#)

[Download CSV](#)

[Download JSON](#)

[Download Excel](#)

Creative Commons License

This work is licensed under CC BY 4.0

What's in the data?

From the Seattle Public Library, we inherit the following columns.

- **usageclass:** denotes if the item is “physical” or “digital.”
- **checkouttype:** denotes the vendor tool used to check out the item.

- **materialtype:** describes the type of item checked out (examples: book, song, movie, music, magazine).
- **checkoutyear:** the 4-digit year of checkout for this record.
- **checkoutmonth:** the month of checkout for this record.
- **checkouts:** a count of the number of times the title was checked out within the “checkout month.”
- **isbn:** a comma-separated list of isbns associated with the item record for the checkout.
- **title:** the full title and subtitle of an individual item.
- **creator:** the author or entity responsible for authoring the item according to the spl.
- **subjects:** the subject of the item as it appears in the catalog.
- **publisher:** the publisher of the title.
- **publicationyear:** the year from the catalog record in which the item was published, printed, or copyrighted.

The dataset contains extensive metadata information on the [Top 500 “Greatest” Novels](#).

i Click to view all metadata fields

Basic Info on Novels:

- **top_500_rank:** numeric rank of text in oclc’s original top 500 list.
- **top_500_title:** title of text, as recorded in oclc’s original top 500 list.
- **author:** author of text, as recorded in oclc’s original top 500 list.
- **pub_year:** year of first publication of text, according to wikipedia.
- **orig_lang:** original language of text, according to wikipedia.
- **genre:** genre of text, as recorded in oclc’s original top 500 list (filtered by the ‘choose genre’ dropdown).

Author Demographic Info:

- **author_birth:** author year of birth, according to viaf.
- **author_death:** author year of death, according to viaf.
- **author_gender:** author gender, according to viaf. note: viaf only includes binary gender categories, with an alternate option of “unknown.” although we want to resist binary categorizations of gender, we have used viaf because it provides the most comprehensive and accurate information we could find for authors on this list, and because it can be difficult if historical authors held non-binary identities. if we find evidence that any of the authors on the list identified or identify as non-binary, we will change the gender categories to reflect their identifications.
- **author_primary_lang:** author’s primary language of publication, according to viaf.
- **author_nationality:** author’s nationality according to viaf. viaf includes multiple national associations for many authors, but we have only collected information on

the first country associated with each author. importantly, this does not include information on tribal citizenship or on changes in nationality across an author's lifetime.

- **author_field_of_activity:** author's primary fields of activity, according to viaf. viaf includes data from multiple global partner institutions, but we only collect viaf data associated with the library of congress (loc).
- **author_occupation:** author's primary occupations, according to viaf. viaf includes data from multiple global partner institutions, but we only collect viaf data associated with the library of congress (loc).

Library Holdings Info:

- **oclc_holdings:** total physical library holdings listed in worldcat for an individual work (owi), according to classify.
- **oclc_eholdings:** total digital library holdings listed in worldcat for an individual work (owi), according to oclc.
- **oclc_total_editions:** total editions of an individual work—physical and digital—listed in worldcat according to oclc.
- **oclc_holdings_rank:** numeric rank of text based on total holdings recorded in worldcat.
- **oclc_editions_rank:** numeric rank of text based on total number of editions recorded in worldcat.

Online Popularity Info:

- **gr_avg_rating:** average star rating for a text on goodreads.
- **gr_num_ratings:** total number of ratings for a text on goodreads.
- **gr_num_reviews:** total number of reviews for a text on goodreads.
- **gr_avg_rating_rank:** numeric rank of text based on average goodreads rating.
- **gr_num_ratings_rank:** numeric rank of text based on overall number of ratings on goodreads.

Unique Identifiers and URLs:

- **oclc_owi:** work id on oclc. a work id represents a cluster based on “author and title information from bibliographic and authority records.” a title can be represented by multiple clusters, and therefore multiple owis. more information about oclc work clustering can be found [here](#).
- **author_viaf:** author viaf id.
- **gr_url:** url for text on goodreads.
- **wiki_url:** url for text on wikipedia.
- **pg_eng_url:** url for english-language text on project gutenber.

- **pg_orig_url:** url for original-language text (where applicable) on project guten-berg.
- **full_text:** full text of the novel, if it is in the public domain.

Note that we provide two titles and two different author fields. One is sourced from the Top 500 “Greatest” Novels, and the other from the SPL’s open checkout data. Reconciling book data is difficult, and different versions and editions of the same text can have slightly different title and author variants.

Where Did The Data Come From? Who Collected It?

For more details on the Top 500 “Greatest” Novels, refer to the initial [post](#) written by Anna Preus and Aashna Sheth, which summarizes the criterion and processes used to generate the source list of texts in the first place.

The SPL’s open checkout data was organized by David Christensen, Data Analysis Lead at the Seattle Public Library. The data from 2005 to 2016 is originally from the digital artwork, “Making Visible the Invisible,” by studios of George Legrady.



Figure 1: George Legrady, “Making Visible the Invisible” (2005). Digital art installation at the Seattle Central Library. Credit: [Seattle Public Library](#).

i Fun Fact: The Art Installation Behind the Data

The checkout data powering this dataset has an unusual origin: a digital art installation. In 2005, artist and UC Santa Barbara professor [George Legrady](#) installed “Making Visible the Invisible” on six LCD screens in the Seattle Central Library’s Mixing Chamber. The screens display real-time visualizations of anonymous checkout activity across all 27 library locations, including running totals, floating titles, and a colorful “Keyword Map Attack” organized by Dewey Decimal categories. By 2017, the library realized the installation had quietly accumulated 12 years of historic checkout data, which became the foundation for [Seattle’s open data portal](#) and, eventually, this dataset. Read more on the [SPL blog](#).

To our knowledge, the SPL is the only library to release checkout data by item with this level of temporal detail in the United States. Their published data is a result of the [Seattle Open Data Program](#), which is an initiative designed to increase transparency into city operations for

the public.

Why Was The Data Collected? How Is The Data Used?

While the SPL uses the data internally to help inform acquisition decisions, reading programs, marketing, and more, we as humanities researchers are interested in using checkout data to study cultural trends at scale. Thus, the data collected here is a subset of the SPL's open checkout data, and part of an [ongoing project](#) to track the dynamics of literary popularity through this data.

Though a powerful resource, the SPL's open checkout data, like all book data, struggles with persistent book identifiers. The same underlying "work" can often have different editions, author name variants, and metadata—for instance, Toni Morrison's *Beloved* is listed as "Beloved," "Beloved (unabridged)," "Beloved : a novel / by Toni Morrison." It is difficult to cluster different versions of the same underlying work together at scale, but we have experimented previously with using a combination of semi-automated methods and manual oversight to cluster smaller sized corpora of interest (Gupta, Maor, et al. 2025).

The Top 500 "Greatest" Novels presents a set of works that have persisted in popularity and relevance well past their first publication date. These are exceptionally popular titles that have avoided the fate of the majority of books, which often fall out of circulation soon after publication (Cohen 2018; Sorensen 2007). Studying novels like these is useful to understand how popularity and reception is working for novels that are cultural touchstones.

How Was The Data Collected?

The SPL's internal approach to capturing the checkout data includes an anonymization process that keeps data collection disconnected from identifiable patrons (Gupta, Christensen, et al. 2025). The library counts each checkout but uses a de-identification approach at the point of data capture to keep the data anonymous ensuring that even internal researchers can't access damaging personally identifiable information.

On our side, the most challenging task is merging and reconciling the SPL's open checkout data with the Top 500 "Greatest" Novels. To reconcile the data, we employ a multi-step algorithm to capture all the records in the library data that may match one of the novels in our source list.

We manipulate the title and author fields in the SPL's open checkout data to normalize against common variants. For example, many titles have "(unabridged)" appended to them, and many authors have a last name, first name formatting (e.g Collins, Suzanne). Once we've simplified the SPL's open checkout data to basic versions of title and creator, we group by those fields to reduce our dataset down to about 800,000 unique titles.

We then run a two-stage algorithm to pair novels from the Top 500 “Greatest” Novels to checkout records. We iterate through all 500 of our novels and compare the last name of the author to the creator field in the SPL’s open checkout data, checking if the creator field contains the last name of the author somewhere in its text. We then use the [Python RapidFuzz library](#) to run a fuzzy matching of the Top 500 “Greatest” Novels title against the title field in the SPL’s open checkout data. We use the [Partial Ratio](#) algorithm which identifies the optimal alignment of the shorter string in the longer string. We use a threshold of 85.

For an example of how this algorithm is working, consider the novel *Catching Fire* by Suzanne Collins. A matching SPL record has the title as *Catching Fire: (movie tie in edition)*. The Partial Ratio algorithm returns a perfect 100 match between this title and the true title, *Catching Fire*. Our algorithm is designed to have high *recall*, rather than *precision*. In other words, we would rather identify more matches that could be wrong than miss matches that could be right. Our preference for recall over precision is because there are *tons* of books in the dataset. Once we have a list of matching records, we can manually look at matches to ensure accuracy. It is much more difficult to identify matches in the first place from over 800,000 unique titles!

Still, the recall of our algorithm will not be perfect, and we will miss some editions that should be clustered in. For example, alternate language editions that lack the English title will often fail the fuzzy match. For example, Stendhal’s *The Red and the Black* in French is titled *Le Rouge et le Noir*, which will not pass our fuzzy matching threshold (luckily we’re only missing 11 checkouts from the French edition).

Our algorithm returns matches for every single novel on our list! But because of our high recall approach, we need to filter out mismatches. For example, Khaled Hosseini’s *A Thousand Splendid Suns* got matched onto the record *And the Mountains Echoed by the Bestselling Author of the Kite Runner and a Thousand Splendid Suns*. These types of cases require manual oversight to filter out.

Our approach has a couple of systematic failures besides weird edge cases. Book series where only the first entry is in the list (*Artemis Fowl*, *Diary of a Wimpy Kid*, *The Maze Runner* etc.) often return all the matching entries from the series, because each future title references the original title. These were excluded manually.

An interesting case we often see when matching library data to these popular novels is dealing with checked out titles that contain more than one text. Think of the entire *Lord of The Rings Trilogy* in one large volume. We make the decision to map checkouts for the entire *Lord of the Rings Trilogy* to each of its entries in the Top 500 “Greatest” Novels, meaning that one checkout is being counted three separate times in this new dataset.

Uncertainty in the Data

We do not claim our matching to be perfect. Besides alternate language editions, it is likely that other strange variants have been missed by our method. Additionally, the SPL's open checkout data is not a perfect proxy for book popularity. Library checkout data is affected by regional dynamics, library programming, and the nature of the library system (Gupta, Christensen, et al. 2025; Gupta, Maor, et al. 2025). Perfect matching would require large-scale manual inspection, which is part of the reason why this type of book reconciliation work is difficult and labor-intensive.

We also don't know specifically what a library checkout *means*. Does a library checkout indicate a discrete "read" of a book? Not necessarily. Just because you check out a book doesn't mean you read it. By contrast, you may have checked out a book once and read it 5 times in quick succession.

More broadly, do library checkouts correlate with book purchasing behavior from the same region? Do checkouts from the Seattle system indicate national interest from library patrons or just regional interest? These are difficult questions to answer, and we talk more about how we can use library checkouts as proxies for constructs of interest in prior work (Gupta, Christensen, et al. 2025).

Library checkouts in Seattle have risen over time partly due to increased participation in the library system and the rise of digital books. As a result, the SPL's open checkout data does not represent a stable configuration of patrons engaging in consistent checkout behavior. You should not treat the longitudinal checkout data as representing the same relationship to readership across the time series. In other words, be careful about making claims about change over time. Increases and decreases in number of checkouts often need to be contextualized against changing library policies, the proliferation of different licensing models and digital alternatives, and other confounding variables.

What can we do with the data?

Top 10 Authors and Books in the Dataset

The first question we may ask of the data is which of these widely held library book and authors actually get read. To that end, we'll present the top 10 authors below.

```
# Note on installation: https://statsandr.com/blog/an-efficient-way-to-install-and-load-r-pa
# install.packages("rmarkdown")
# too big
# library(tidyverse)
```

```

from datetime import date
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.dates as mdates
import pandas as pd
import numpy as np

# Load Top 500 SPL Data
top_500_df = pd.read_csv("https://responsible-datasets-in-context.s3.us-west-2.amazonaws.com,

top_authors = (
    top_500_df
    .groupby('author')['checkouts']
    .sum()
    .sort_values(ascending=False)
    .head(10)
    .reset_index()
    .rename(columns={'author': 'Author', 'checkouts': 'Total Checkouts'})
)

top_authors['Rank'] = range(1, 11)
top_authors = top_authors[['Rank', 'Author', 'Total Checkouts']]
top_authors['Total Checkouts'] = top_authors['Total Checkouts'].apply(lambda x: f"{x:,}")

top_authors.style.hide(axis='index').set_table_styles([
    {'selector': 'th', 'props': [('text-align', 'left'), ('font-weight', 'bold')]},
    {'selector': 'td', 'props': [('text-align', 'left')]}
]).set_caption("Top 10 Authors by Total Checkouts")

```

Table 1: Top 10 Authors by Total Checkouts

Rank	Author	Total Checkouts
1	J.K. Rowling	145,161
2	Suzanne Collins	67,500
3	Jane Austen	53,731
4	J.R.R. Tolkien	52,374
5	Stieg Larsson	45,227
6	Stephen King	39,926
7	Charles Dickens	37,749
8	Stephenie Meyer	37,370

Rank	Author	Total Checkouts
9	Gillian Flynn	35,366
10	John Grisham	33,372

Perhaps unsurprisingly, we see J.K. Rowling at the top of the list, in part because of the per-book popularity of the *Harry Potter* series, and in-part because of the raw number of *Harry Potter* books in our database. Let's also take a look at the top-10 books as well.

```
top_books = (
    top_500_df
    .groupby('top_500_title')['checkouts']
    .sum()
    .sort_values(ascending=False)
    .head(10)
    .reset_index()
    .rename(columns={'top_500_title': 'Title', 'checkouts': 'Total Checkouts'})
)

top_books['Rank'] = range(1, 11)
top_books = top_books[['Rank', 'Title', 'Total Checkouts']]
top_books['Total Checkouts'] = top_books['Total Checkouts'].apply(lambda x: f"{x:,}")

top_books.style.hide(axis='index').set_table_styles([
    {'selector': 'th', 'props': [('text-align', 'left'), ('font-weight', 'bold')]},
    {'selector': 'td', 'props': [('text-align', 'left')]}
]).set_caption("Top 10 Books by Total Checkouts")
```

Table 2: Top 10 Books by Total Checkouts

Rank	Title	Total Checkouts
1	Gone Girl	35,366
2	The Handmaid's Tale	33,240
3	Harry Potter and the Sorcerer's Stone	31,386
4	The Book Thief	30,549
5	Twilight	29,289
6	The Hunger Games	27,960
7	The Giver	27,906
8	Pride and Prejudice	27,250
9	The Girl on the Train	26,610
10	A Wrinkle in Time	24,762

Let's take a look at one of the major books in our dataset, Jane Austen's *Pride and Prejudice*

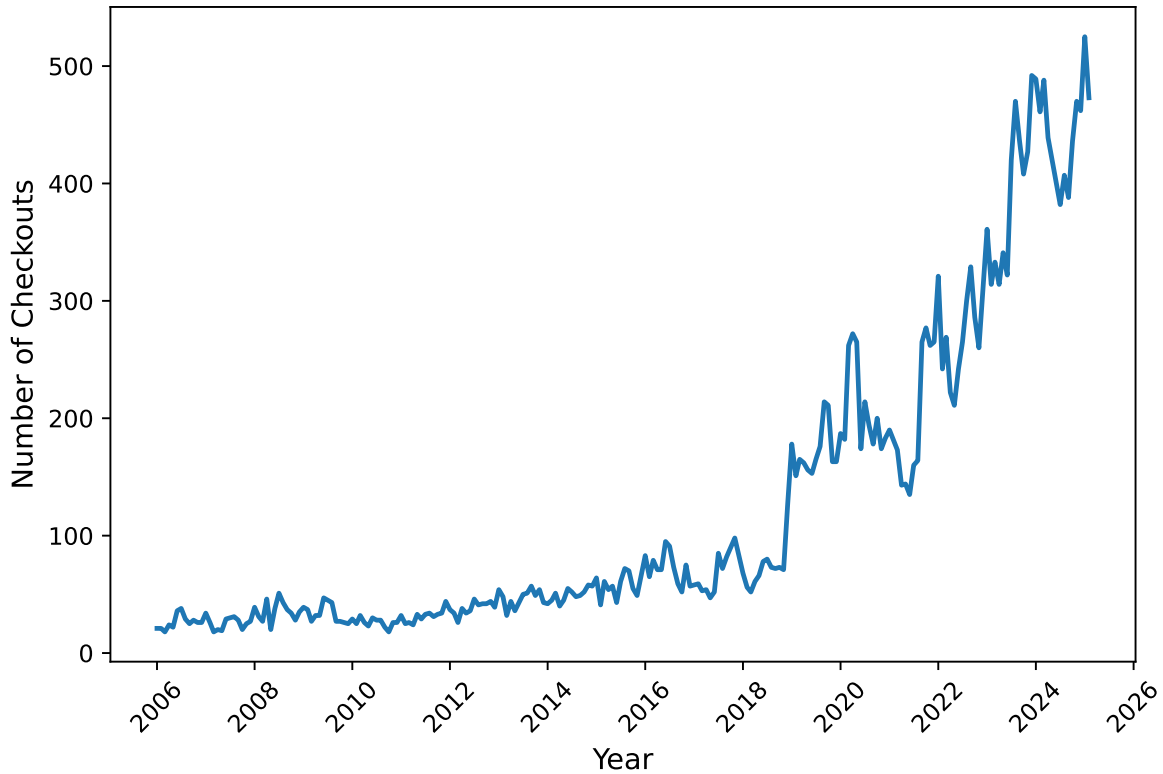
```
pride_prejudice_checkouts = top_500_df[top_500_df['top_500_title'] == 'Pride and Prejudice']
pride_prejudice_timeseries = (
    pride_prejudice_checkouts
    .groupby('year-month')['checkouts']
    .sum()
    .reset_index()
)

# Convert year-month to datetime for better x-axis formatting
pride_prejudice_timeseries['date'] = pd.to_datetime(pride_prejudice_timeseries['year-month'])

plt.figure(figsize=(7, 5))
plt.plot(pride_prejudice_timeseries['date'], pride_prejudice_timeseries['checkouts'], linewidth=2)
plt.title('Pride and Prejudice Checkouts Over Time', fontsize=14, fontweight='bold')
plt.xlabel('Year', fontsize=12)
plt.ylabel('Number of Checkouts', fontsize=12)

# Format x-axis to show years only
plt.gca().xaxis.set_major_locator(mdates.YearLocator(2))
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%Y'))
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

Pride and Prejudice Checkouts Over Time



Austen's novel really grows in the library system in the late 2010s and into the pandemic, with a majority of its reading happening in the last couple of years. Every one of these novels have their own circulation stories, and I encourage readers to explore specific books of interest and think about why their curves look the way that they do.

Authorial Death

In prior work (Gupta, Maor, et al. 2025), we present evidence that an author's death leads to a temporary boost in library checkouts for that author (at least if they're famous).

We verify that relationship by generating time-series plots of authors who died in the last 20 years.

```
# Note on installation: https://statsandr.com/blog/an-efficient-way-to-install-and-load-r-pa
# install.packages("rmarkdown")
# too big
```

```

# library(tidyverse)
from datetime import date
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.dates as mdates
import pandas as pd
import numpy as np

# Load Top 500 SPL Data
top_500_df = pd.read_csv("https://responsible-datasets-in-context.s3.us-west-2.amazonaws.com,

dead_authors = (
    top_500_df
    .groupby(['author', 'checkoutyear', 'checkoutmonth'], as_index=False)
    .agg({'checkouts': 'sum', 'author_death': 'first'})
)

dead_authors['author_death'] = pd.to_numeric(
    dead_authors['author_death'], errors='coerce'
)

dead_authors = dead_authors[dead_authors['author_death'] > 2005]

# author deaths are only by year in the dataframe, so we add specific month and days for our
author_death_dates = {
    "Aleksandr Isaevich Solzhenitsyn": date(2008, 8, 3),
    "Betsy Cromer Byars": date(2020, 2, 26),
    "Beverly Cleary": date(2021, 3, 25),
    "Chinua Achebe": date(2013, 3, 21),
    "Colleen McCullough": date(2015, 1, 29),
    "Cormac McCarthy": date(2023, 6, 13),
    "Daniel Keyes": date(2014, 6, 15),
    "Donald J. Sobol": date(2012, 8, 11),
    "E.L. Konigsburg": date(2013, 4, 19),
    "Gabriel García Márquez": date(2014, 4, 17),
    "Gary Paulsen": date(2021, 10, 13),
    "Harper Lee": date(2016, 2, 19),
    "Herman Wouk": date(2019, 5, 17),
    "J.D. Salinger": date(2010, 1, 27),
    "Jean Craighead George": date(2012, 5, 15),
    "John Le Carré": date(2020, 12, 12),

```

```

    "Kurt Vonnegut": date(2007, 4, 11),
    "Michael Crichton": date(2008, 11, 4),
    "Natalie Babbitt": date(2016, 10, 31),
    "Nicholas Evans": date(2022, 8, 9),
    "Norman Mailer": date(2007, 11, 10),
    "Norton Juster": date(2021, 3, 8),
    "Patricia MacLachlan": date(2022, 3, 31),
    "Paula Fox": date(2017, 3, 1),
    "Ray Bradbury": date(2012, 6, 5),
    "Richard Adams": date(2016, 12, 24),
    "Robert James Waller": date(2017, 3, 10),
    "Sid Fleischman": date(2010, 3, 17),
    "Sue Grafton": date(2017, 12, 28),
    "Toni Morrison": date(2019, 8, 5),
    "Umberto Eco": date(2016, 2, 19),
    "William Styron": date(2006, 11, 1),
}

dead_authors['author_death_date'] = dead_authors['author'].map(author_death_dates)
dead_authors['author_death_ts'] = pd.to_datetime(dead_authors['author_death_date'])
sns.set(style="whitegrid")

# Choose a fixed color palette for all plots
palette = sns.color_palette("tab10", n_colors=10) # 10 distinct colors, cycle if >10 authors

# build checkout datetime
dead_authors['date'] = pd.to_datetime(
    dict(
        year=dead_authors['checkoutyear'],
        month=dead_authors['checkoutmonth'],
        day=1
    )
)

# ensure death timestamp exists and is datetime
dead_authors['author_death_ts'] = pd.to_datetime(
    dead_authors['author_death_date'], format="%Y", errors="coerce"
)

```

```

# Map each author to a fixed color
author_list = sorted(dead_authors['author'].unique())
author_color_map = {author: palette[i % len(palette)] for i, author in enumerate(author_list)}

#filter down to a subset of dead_authors

dead_authors = dead_authors[dead_authors['author'].isin(['Beverly Cleary', 'Chinua Achebe', '

for author, df_a in dead_authors.groupby('author'):
    death_date = df_a['author_death_ts'].iloc[0]

    if pd.isna(death_date):
        print(f"Skipping {author}: no valid death date")
        continue

    plt.figure(figsize=(7, 5))

    # Use standardized color for this author
    color = author_color_map[author]

    # Smooth line without markers
    plt.plot(df_a['date'], df_a['checkouts'], color=color, linestyle='-', linewidth=2, label=

    # Vertical line for death
    plt.axvline(death_date, color='red', linestyle='--', linewidth=2, alpha=0.7, label='Deat

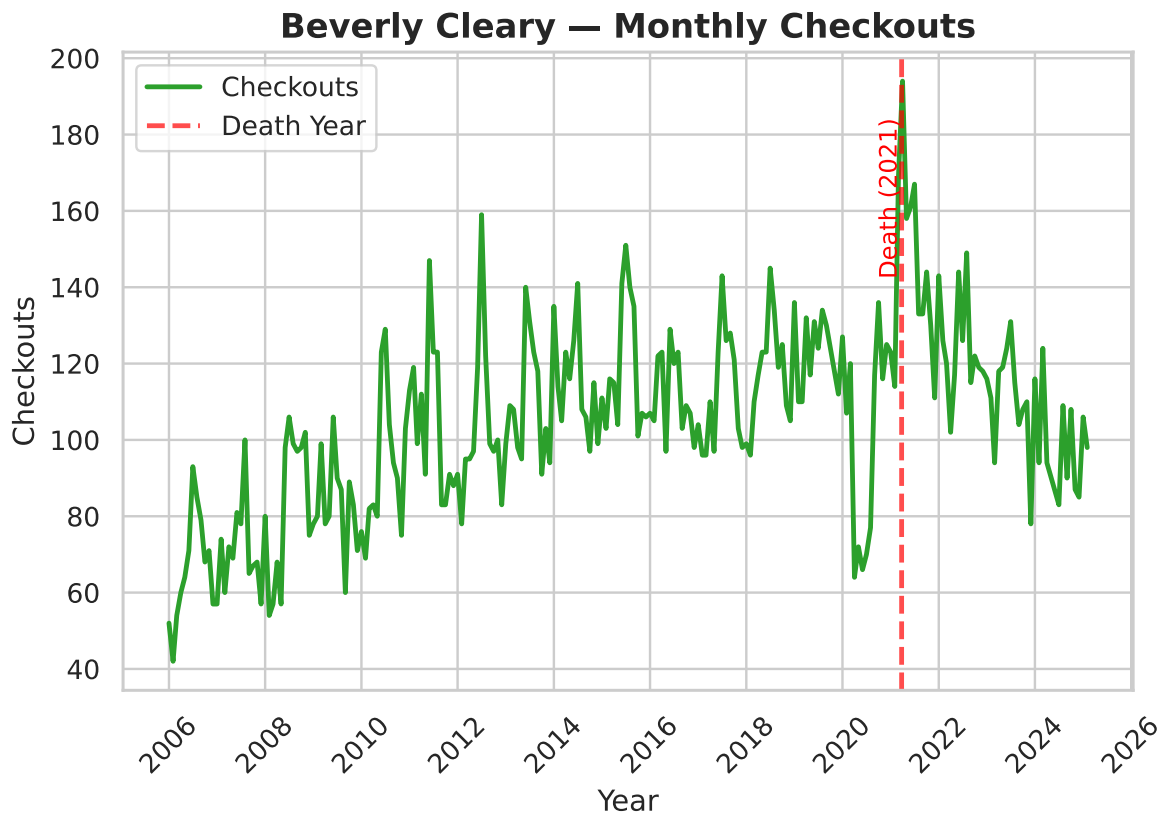
    # Annotate death year
    plt.text(death_date, df_a['checkouts'].max()*0.95, f"Death ({death_date.year})",
             rotation=90, color='red', va='top', ha='right', fontsize=10)

    # Formatting
    plt.title(f"{author} - Monthly Checkouts", fontsize=14, fontweight='bold')
    plt.xlabel("Year", fontsize=12)
    plt.ylabel("Checkouts", fontsize=12)
    plt.legend()

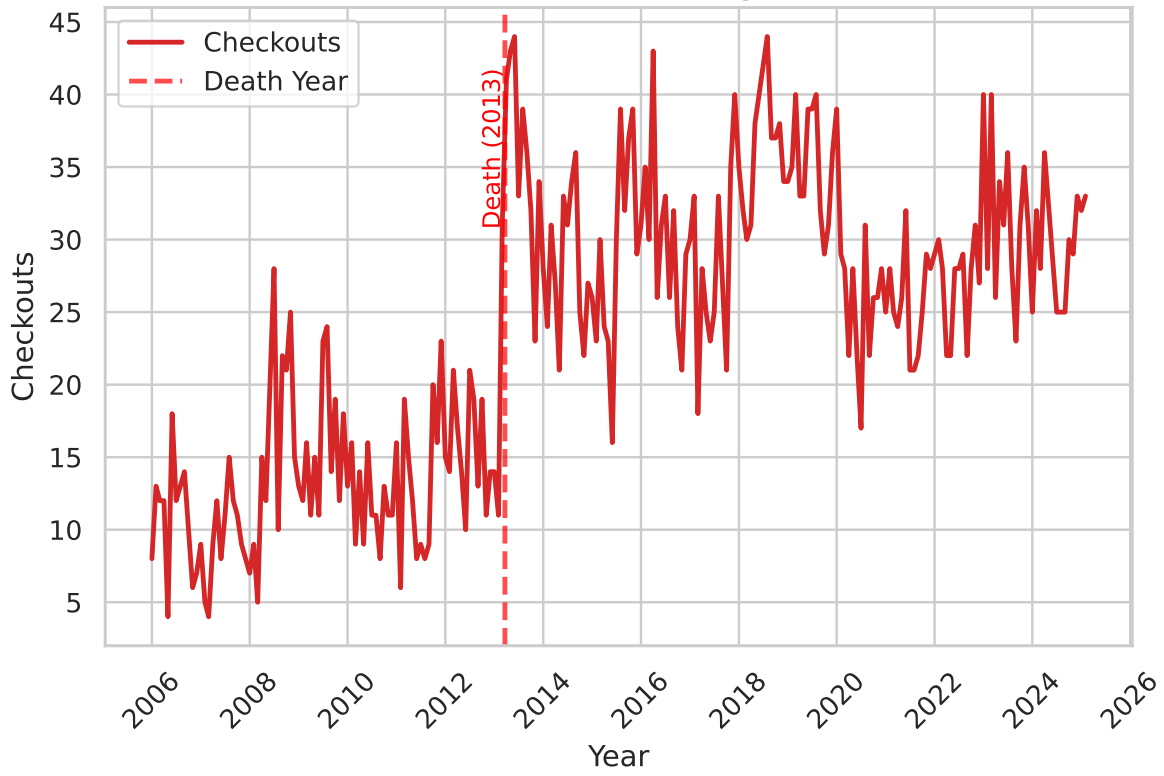
    # Better x-axis formatting
    plt.gca().xaxis.set_major_locator(mdates.YearLocator(2))
    plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%Y'))
    plt.xticks(rotation=45)

```

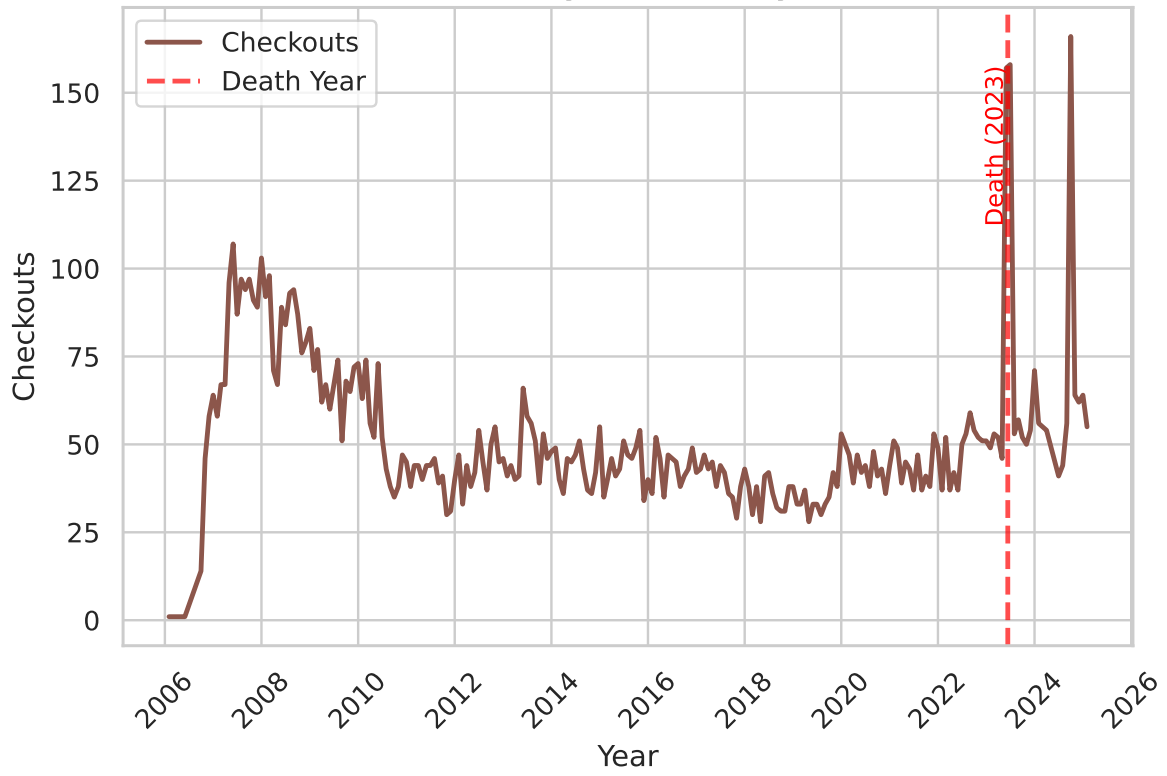
```
plt.tight_layout()
plt.show()
```



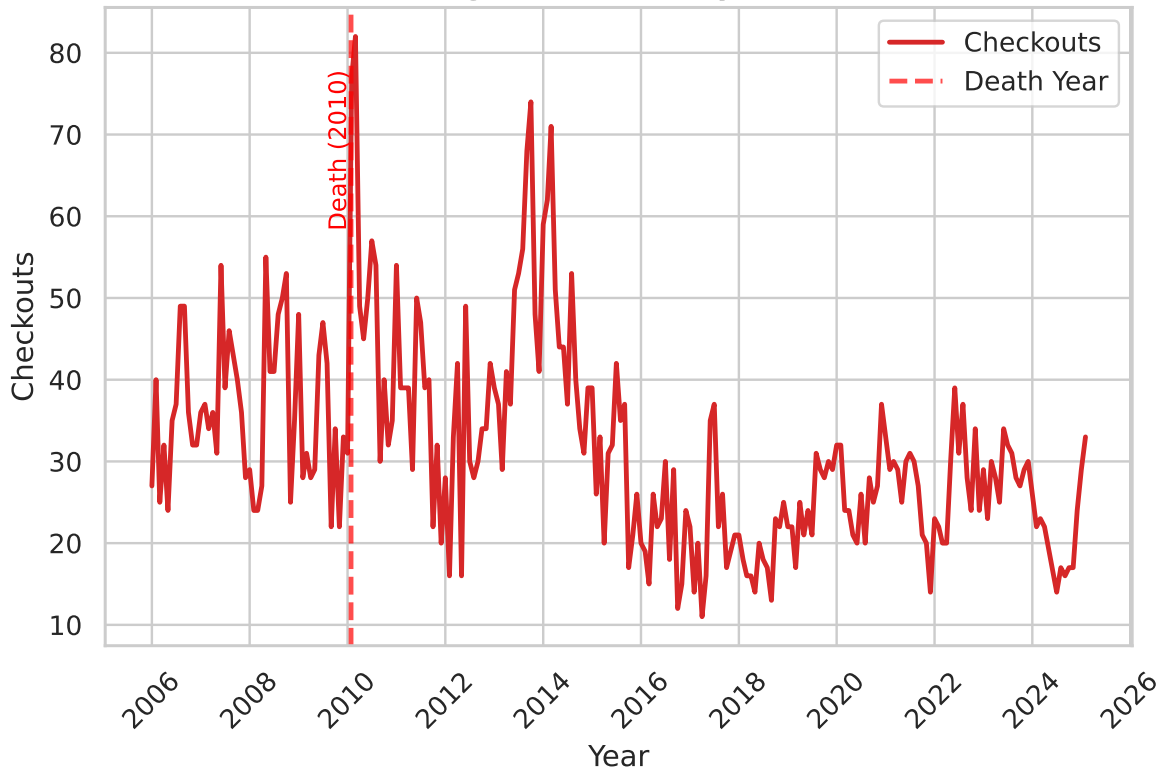
Chinua Achebe — Monthly Checkouts



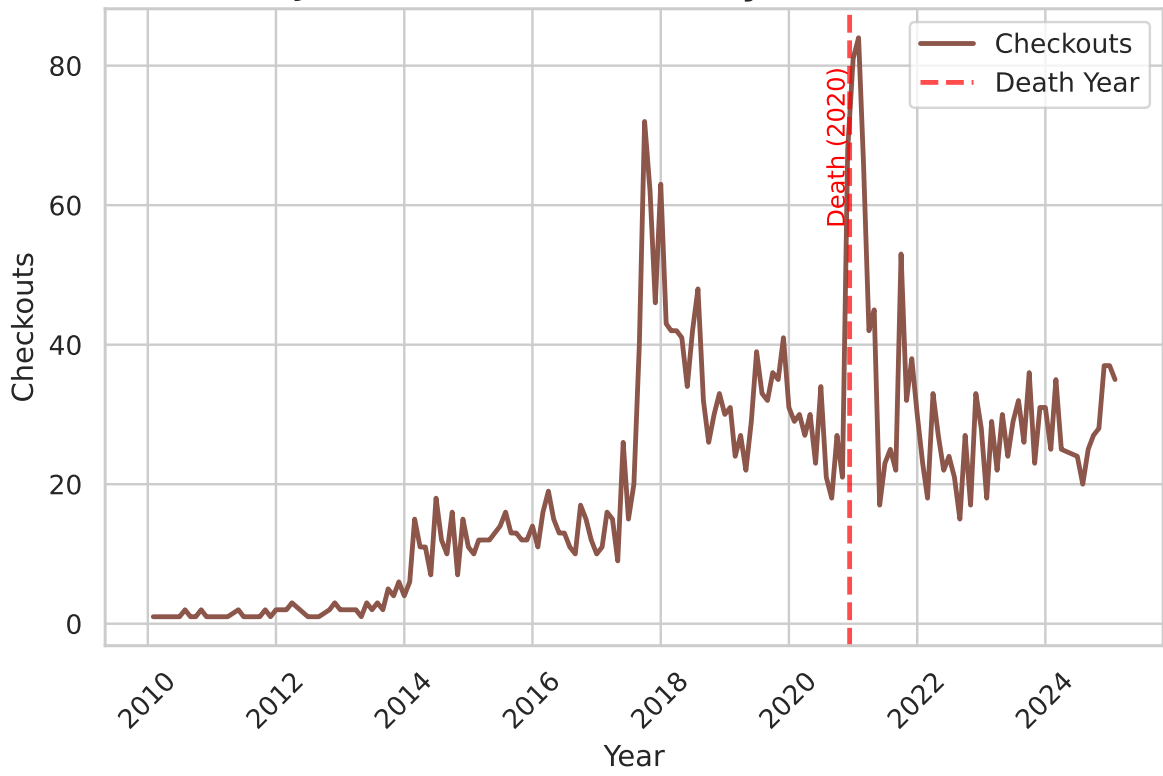
Cormac McCarthy — Monthly Checkouts



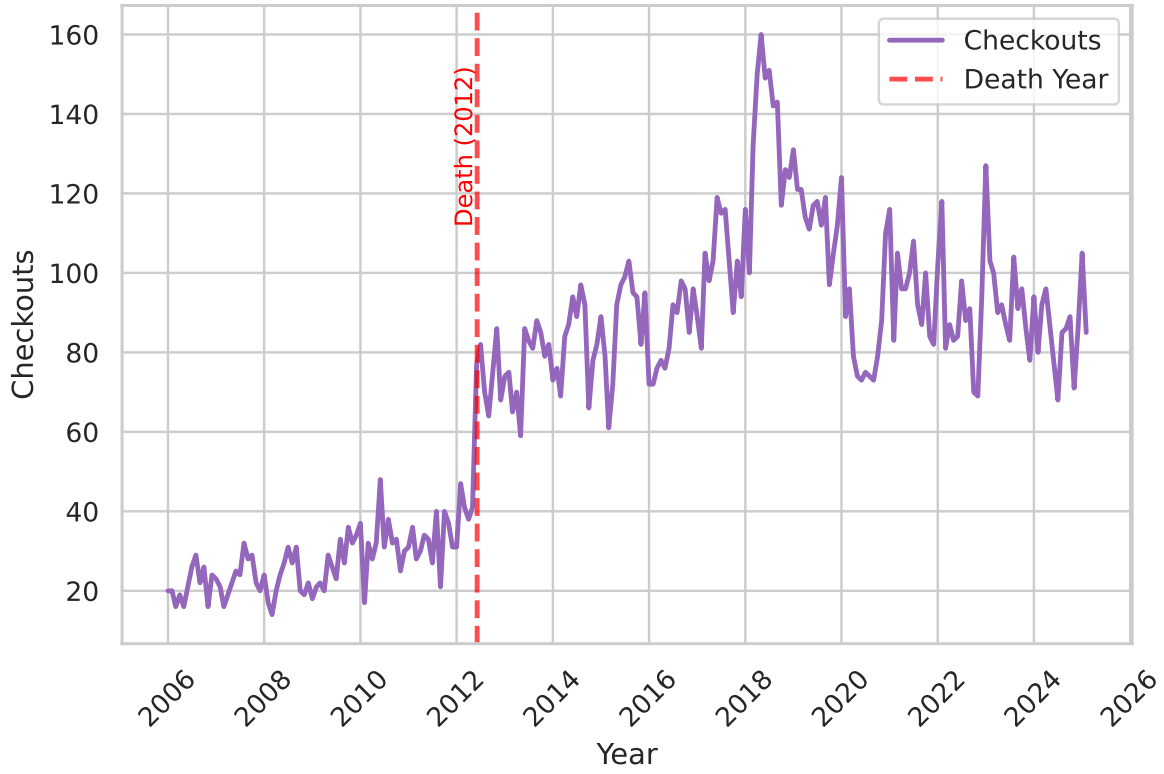
J.D. Salinger — Monthly Checkouts

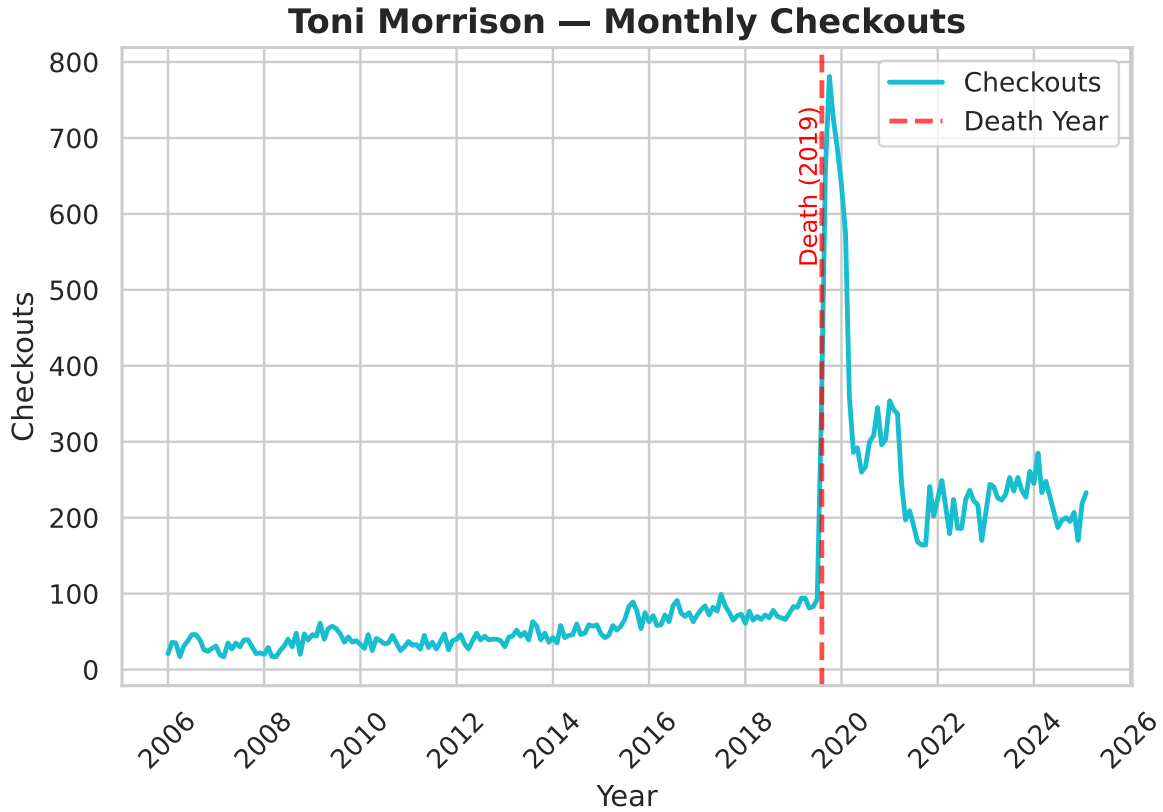


John Le Carré — Monthly Checkouts



Ray Bradbury — Monthly Checkouts





We display seven prominent authors who died in the timeframe, and we encourage readers to explore the rest of the authorial deaths on their own.

Backlist Titles

Berglund and Steiner (2021) present evidence from Scandinavian fiction that an author’s new releases boost interest in their previous titles. Books by the same author affect each other’s success. The Top 500 “Greatest” Novels has 89 different authors with multiple releases, which raises a natural question: do works by the same author tend to have similar patterns of interest over time?

To test this, we looked at whether books by the same author tend to get checked out at similar rates. For example, if Stephen King’s *The Shining* is getting checked out a lot this month, are his other books also getting checked out more than usual? We measured this using a correlation coefficient — a number between 0 and 1 that captures how closely two things move together, where 0 means no relationship and 1 means a perfect one. On average, we found a moderate correlation of 0.42, meaning that knowing one book’s checkout trend explains about 42% of

what's happening with the same author's other books. The remaining 58% depends on factors specific to each book, like whether it was assigned in a class or got a recent news mention.

```
# grabbing the subset of the df with authors that show up multiple times
top_500_novels = pd.read_csv("../../../datasets/top-500-novels/final_merged_dataset_no_full_t
top_500_novels.groupby('author').count()
author_counts = top_500_novels['author'].value_counts()
multi_book_authors = author_counts[author_counts > 1].index.tolist()
multi_book_authors
multi_book_df = top_500_df[top_500_df['author'].isin(multi_book_authors)]

# to do correlations, need to fill in missing values with 0s
def fill_after_first_value(series):
    filled = series.copy()
    has_seen_value = False
    for i in range(len(series)):
        if not pd.isna(series.iloc[i]):
            has_seen_value = True
        elif has_seen_value:
            filled.iloc[i] = 0
    return filled

author_pivot = multi_book_df.pivot_table(
    index='year-month',
    columns='top_500_title',
    values='checkouts',
    aggfunc='sum'
).fillna(0)

author_pivot = author_pivot.apply(fill_after_first_value)

all_correlations = []

for author, df_a in multi_book_df.groupby('author'):
    titles = df_a['top_500_title'].unique()
    if len(titles) < 2:
        continue # Need at least two books to compare

    author_data = author_pivot[titles]

# Calculate correlation matrix
```

```

corr_matrix = author_data.corr()
all_correlations.append((author, corr_matrix))

mean_correlations = {}
for author, corr_matrix in all_correlations:
    # Exclude diagonal by masking it
    mask = np.triu(np.ones(corr_matrix.shape), k=1).astype(bool)
    mean_corr = corr_matrix.where(mask).stack().mean()
    mean_correlations[author] = mean_corr
mean_correlations
mean_of_means = np.mean(list(mean_correlations.values()))
print(mean_of_means)

```

0.42244261897179364

It's hard to interpret that number without context. If we look at just the correlations on average between every book in the Top 500 "Greatest" Novels, we only observe a coefficient of 0.16. So books by the same author are almost 3x more correlated than what we would expect on average.

```

wide_df = top_500_df.pivot_table(
    values="checkouts",
    index="year-month",
    columns=["top_500_title"],
    aggfunc="sum", # Sum in case of duplicates
    fill_value=np.nan, # Fill missing values with 0
)

wide_df = wide_df.apply(fill_after_first_value)

correlation_matrix = wide_df.corr()

mask = np.triu(np.ones(correlation_matrix.shape), k=1).astype(bool)
mean_correlation = correlation_matrix.where(mask).stack().mean()
mean_correlation

mean_correlation = correlation_matrix.where(mask).stack().mean()
print(mean_correlation)

```

0.15918878442212858

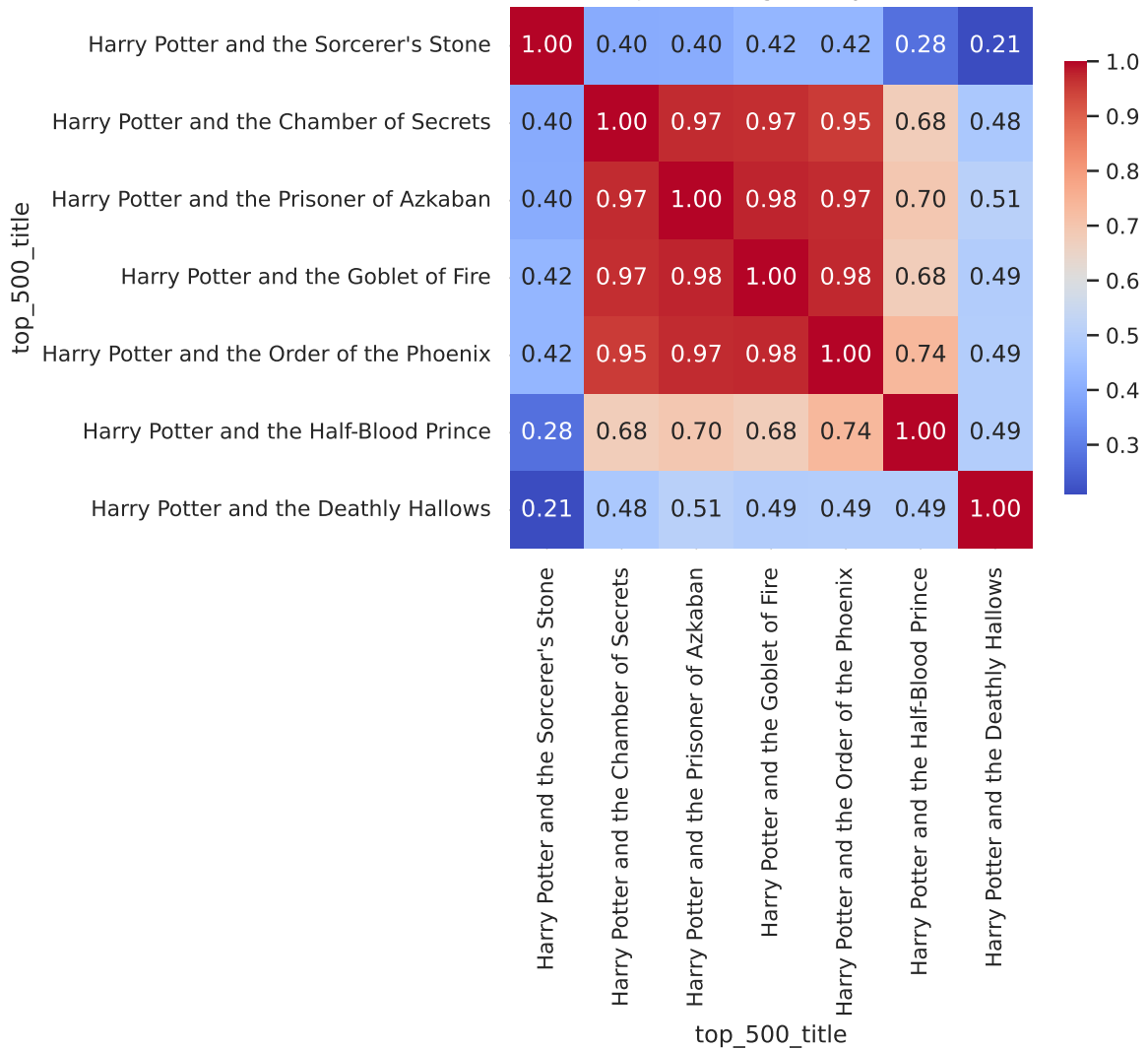
Let's look at a few specific examples. A lot of books by the same author belong to a book series. For example, all of J.K. Rowling's *Harry Potter* books are in the Top 500 "Greatest" Novels, so we can generate a correlation heatmap for her books.

```
jk_corr = next((corr for author, corr in all_correlations if author == 'J.K. Rowling'), None)

if jk_corr is not None:
    # Focus on main Harry Potter series
    hp_order = [
        "Harry Potter and the Sorcerer's Stone",
        "Harry Potter and the Chamber of Secrets",
        "Harry Potter and the Prisoner of Azkaban",
        "Harry Potter and the Goblet of Fire",
        "Harry Potter and the Order of the Phoenix",
        "Harry Potter and the Half-Blood Prince",
        "Harry Potter and the Deathly Hallows"
    ]
    jk_corr_hp = jk_corr.reindex(index=hp_order, columns=hp_order)

    # Plot heatmap
    plt.figure(figsize=(6, 5))
    sns.heatmap(jk_corr_hp, annot=True, fmt=".2f", cmap='coolwarm', cbar_kws={"shrink": 0.8})
    plt.title("Correlation Matrix of J.K. Rowling's Harry Potter Books Checkouts")
    plt.show()
```

Correlation Matrix of J.K. Rowling's Harry Potter Books Checkouts

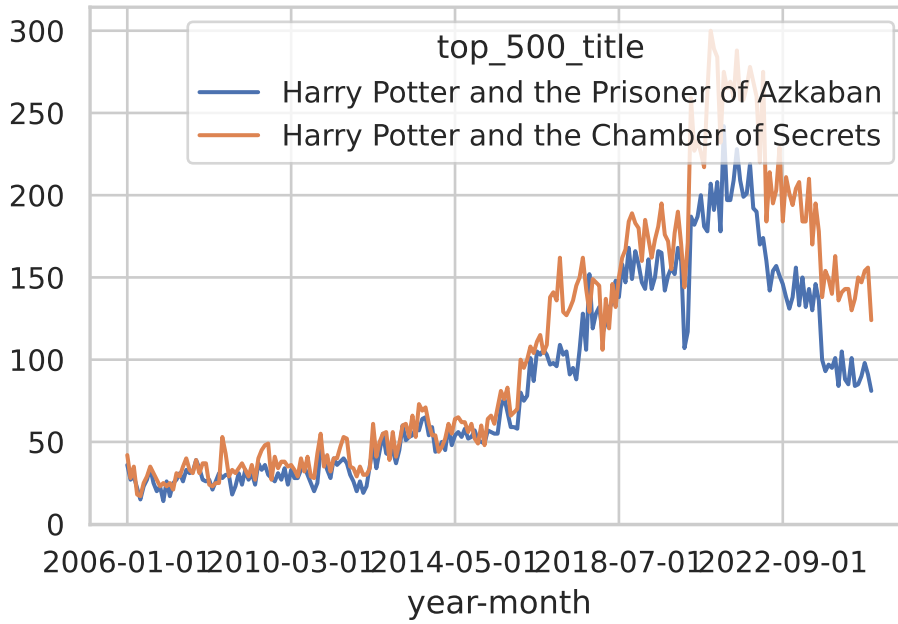


The heatmap is really striking. The middle four books of the series are highly correlated with each other, while the first book and the final two seem to have their own dynamics.

Let's look at some time-series plots to figure out what's going on with the edge books.

First, *The Prisoner of Azkaban* vs. *The Chamber of Secrets*

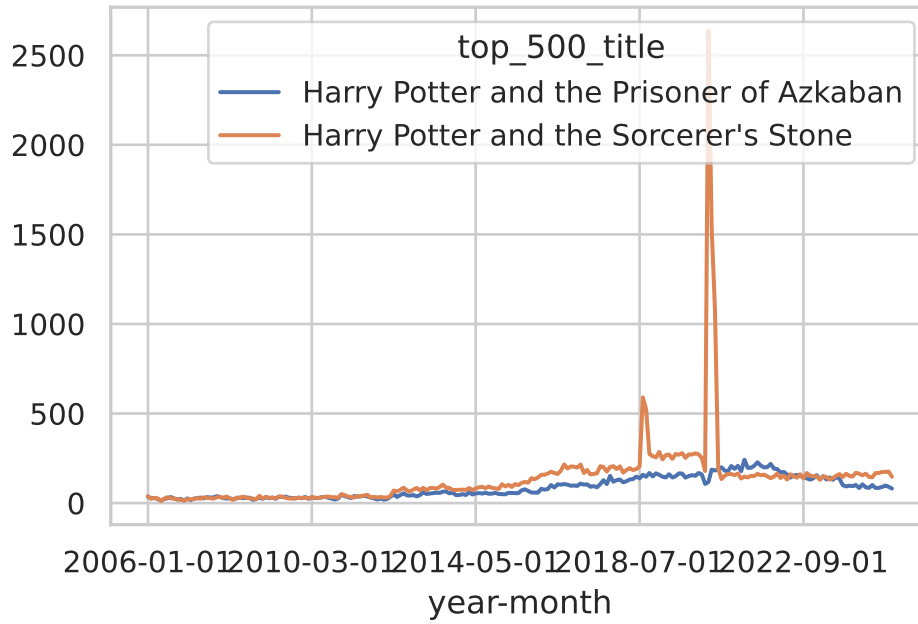
```
wide_df[["Harry Potter and the Prisoner of Azkaban", "Harry Potter and the Chamber of Secrets"]]
```



These have a correlation coefficient of 0.97, and that lines up with what we're seeing in the time-series plot.

How about if we compare *The Prisoner of Azkaban* to *The Sorcerer's Stone*?

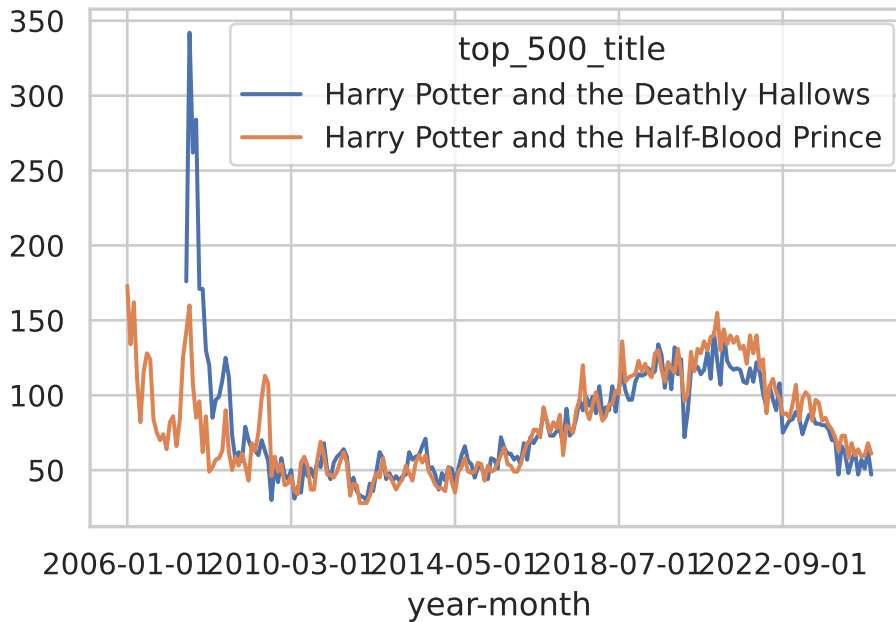
```
wide_df[["Harry Potter and the Prisoner of Azkaban", "Harry Potter and the Sorcerer's Stone"]]
```



They actually track really well together for most of the time frame, but we see a huge spike for *The Sorcerer's Stone* in 2020, which is going to really confound the correlation coefficient.

How about *The Deathly Hallows* and *The Half-Blood Prince*?

```
wide_df[["Harry Potter and the Deathly Hallows", "Harry Potter and the Half-Blood Prince"]].
```



These have a more obvious explanation. Both these books were released in the last 20 years, so they have large numbers of checkouts following their actual release. This follows what we know about how books function commercially, where a large share of their sales come immediately around their release (Sorensen 2007).

Outside of their release date behavior, we see that these books still tend to follow the same dynamics. In fact, if we trim out the first few years of our timeframe and only look at our data after 2007, the correlation coefficients go straight back up!

```
multi_book_df_post_2009 = multi_book_df[multi_book_df['checkoutyear'] >= 2009]

author_pivot_post_2009 = multi_book_df_post_2009.pivot_table(
    index='year-month',
    columns='top_500_title',
    values='checkouts',
    aggfunc='sum'
).fillna(0)

author_pivot_post_2009 = author_pivot_post_2009.apply(fill_after_first_value)

all_correlations_post_2009 = []
```

```

for author, df_a in multi_book_df_post_2009.groupby('author'):
    titles = df_a['top_500_title'].unique()
    if len(titles) < 2:
        continue # Need at least two books to compare

    author_data = author_pivot_post_2009[titles]

    # Calculate correlation matrix
    corr_matrix = author_data.corr()
    all_correlations_post_2009.append((author, corr_matrix))

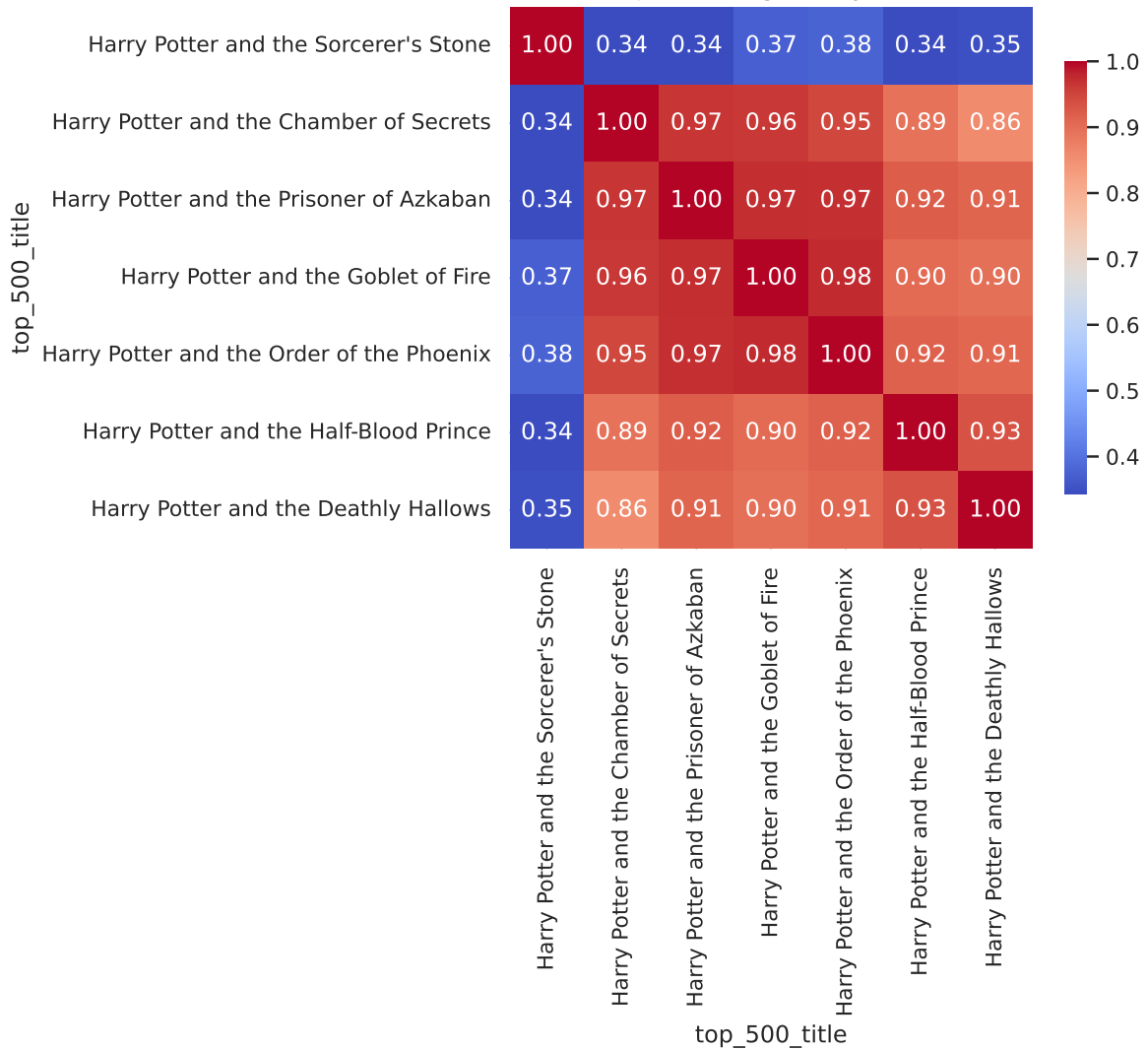
jk_corr_post_2009 = next((corr for author, corr in all_correlations_post_2009 if author == 'J.K. Rowling'), None)

if jk_corr_post_2009 is not None:
    # Focus on main Harry Potter series
    hp_order = [
        "Harry Potter and the Sorcerer's Stone",
        "Harry Potter and the Chamber of Secrets",
        "Harry Potter and the Prisoner of Azkaban",
        "Harry Potter and the Goblet of Fire",
        "Harry Potter and the Order of the Phoenix",
        "Harry Potter and the Half-Blood Prince",
        "Harry Potter and the Deathly Hallows"
    ]
    jk_corr_hp_post_2009 = jk_corr_post_2009.reindex(index=hp_order, columns=hp_order)

    # Plot heatmap
    plt.figure(figsize=(6, 5))
    sns.heatmap(jk_corr_hp_post_2009, annot=True, fmt=".2f", cmap='coolwarm', cbar_kws={"shrink": 0})
    plt.title("Correlation Matrix of J.K. Rowling's Harry Potter Books Checkouts")
    plt.show()

```

Correlation Matrix of J.K. Rowling's Harry Potter Books Checkouts



The high correlation between the books in the series suggest that these books are likely complementary goods. The appeal for one of these books increases with the appeal for the other. Of course this makes sense: readers who finish *The Prisoner of Azkaban* are likely to move straight onto *The Goblet of Fire* in quick succession, and the value of any one of these books to a reader is intrinsically tied to the value of the other books in the series, and the series as a whole.

Genres

Another affordance of the Top 500 “Greatest” Novels is its genre information. Although only about half of the entries are filled out, the genre metadata allows us to compare books within the same genre to see if there are any relationships between their reception.

We can conduct the same correlation test as we did with authors to inspect which genres are the most internally correlated.

```
genre_correlations = {}
for genre, df_g in top_500_df.groupby('genre'):
    if genre == 'na':
        continue

    genre_pivot = df_g.pivot_table(
        index='year-month',
        columns='top_500_title',
        values='checkouts',
        aggfunc='sum'
    ).fillna(0)

    genre_pivot = genre_pivot.apply(fill_after_first_value)

    corr_matrix = genre_pivot.corr()

    # Exclude diagonal by masking it
    mask = np.triu(np.ones(corr_matrix.shape), k=1).astype(bool)
    mean_corr = corr_matrix.where(mask).stack().mean()
    genre_correlations[genre] = mean_corr

# Create a nice-looking DataFrame table
genre_corr_df = pd.DataFrame(
    list(genre_correlations.items()),
    columns=['Genre', 'Mean Correlation']
).sort_values('Mean Correlation', ascending=False).reset_index(drop=True)

genre_corr_df['Mean Correlation'] = genre_corr_df['Mean Correlation'].round(3)

print(genre_corr_df.to_string(index=False))
```

Genre	Mean Correlation
horror	0.571

scifi	0.512
autobio	0.474
war	0.374
fantasy	0.286
allegories	0.268
action	0.226
history	0.208
thrillers	0.164
mystery	0.160
political	0.140
bildung	0.118
romance	0.111

Sci-fi and horror are the most internally correlated genres, while political, bildung, and mystery are less internally correlated.

Conclusion

This essays demonstrates some of the cool research directions you can explore with the SPL’s open checkout data. The Top 500 “Greatest” Novels represents a particularly canonical and popular set of books, featuring both contemporary and historical authors, and a diverse range of genres. Our work here points to some interesting dynamics that you can analyze within book series, between and within genres, and around prominent external events that may drive readership.

Finally, if you want to explore more library circulation data, you can also checkout the [“What’s Seattle Reading?”](#) project.

References

- Berglund, Karl, and Ann Steiner. 2021. “Is Backlist the New Frontlist?: Large-Scale Data Analysis of Bestseller Book Consumption in Streaming Services.” *Logos* 32 (1): 7–24. <https://doi.org/10.1163/18784712-03104006>.
- Cohen, Margaret. 2018. *The Sentimental Education of the Novel*. Princeton University Press. <https://muse.jhu.edu/book/61065>.
- Gupta, Neel, David Christensen, and Melanie Walsh. 2025. “Seattle Public Library’s Open Checkout Data: What Can It Tell Us about Readers and Book Popularity More Broadly?” *Journal of Open Humanities Data* 11 (August): 46. <https://doi.org/10.5334/johd.332>.

Gupta, Neel, Daniella Maor, Karalee Harris, Emily Backstrom, Hongyuan Dong, and Melanie Walsh. 2025. “The Canon in Circulation: Tracking the Reception of _Norton Anthology_ Authors in Library Checkout Data.” *Anthology of Computers and the Humanities* 3: 1510–22. <https://doi.org/10.63744/P6qPH135jhY2>.

Sorensen, Alan T. 2007. “Bestseller Lists and Product Variety.” *The Journal of Industrial Economics* 55 (4): 715–38. <https://doi.org/10.1111/j.1467-6451.2007.00327.x>.

Walsh, Melanie. 2022. “Where Is All the Book Data?” *Public Books*. <https://www.publicbooks.org/where-is-all-the-book-data/>.

Explore the Data

Explore the Data

Exercises

Programming Exercises

Python

R