

# Interactive Visualizations (IVI) Bericht

Nicola Rohner

SE FE 2023

**Repository:** <https://github.com/nicolarohner1337/ivi>

## Abstract

In diesem Bericht werden die Lernergebnisse des Moduls IVI zusammengefasst.

Für alle ausser LE1 wurden folgende Daten verwendet:

GroupLens Research hat Bewertungsdatensätze von der MovieLens-Website <https://movielens.org> gesammelt und zur Verfügung gestellt. Die Datensätze wurden über verschiedene Zeiträume hinweg gesammelt, abhängig von der Grösse des Satzes. Das ZIP findet man unter folgendem Link: <https://files.grouplens.org/datasets/movielens/ml-latest.zip> Die Daten umfassen 27.000.000 Bewertungen und 1.100.000 Tag-Anwendungen für 58.000 Filme von 280.000 Nutzern. Beinhaltet Tag-Genom-Daten mit 14 Millionen Relevanz-Scores für 1.100 Tags.

Die Visualisierungen in diesem Bericht wurden alle selbst erstellt. Der Datensatz wurde bereits für die Challenge CDS1 genutzt und für die Erarbeitung von GDV.

## Contents

|          |  |          |
|----------|--|----------|
| <b>1</b> | <b>LE1: Performance</b>                                  | <b>2</b> |
| 1.1      | WebGL . . . . .  | 2        |
| 1.2      | Tiling . . . . .   | 3        |
| <b>2</b> | <b>LE2: Grundsätze für die Gestaltung von Dashboards</b> | <b>4</b> |
| 2.1      | Schneiderman's Mantra . . . . .                          | 4        |
| 2.2      | Verknüpfte Ansichten . . . . .                           | 5        |
| 2.3      | Brushing . . . . .                                       | 5        |
| 2.4      | Filtering . . . . .                                      | 5        |
| 2.5      | Implementation mit Dash . . . . .                        | 5        |
| <b>3</b> | <b>LE3: HCI Grundlagen</b>                               | <b>6</b> |
| 3.1      | Fitts's Law . . . . .                                    | 6        |
| 3.2      | Weber's Law . . . . .                                    | 6        |
| 3.3      | 5 Dimensionen der Interaktionsgestaltung . . . . .       | 7        |
| 3.4      | Implementation mit Dash . . . . .                        | 7        |
| <b>4</b> | <b>Anhang</b>  | <b>9</b> |
| 4.1      | Figure 1 . . . . .                                       | 9        |
| 4.2      | Figure 2 . . . . .                                       | 10       |
| 4.3      | Figure 3 . . . . .                                       | 11       |
| 4.4      | Figure 4 . . . . .                                       | 12       |
| 4.5      | Figure 5 . . . . .                                       | 13       |
| 4.6      | Figure 6 . . . . .                                       | 13       |
| 4.7      | Figure 7 . . . . .                                       | 13       |
| 4.8      | Figure 8 . . . . .                                       | 14       |
| 4.9      | Figure 9 . . . . .                                       | 14       |
| 4.10     | Figure 10 . . . . .                                      | 15       |
| 4.11     | Figure 11 . . . . .                                      | 16       |

# 1 LE1: Performance

In Zeiten von Big Data ist es wichtig, dass Visualisierungen schnell und performant sind. In diesem LE wird untersucht, wie sich die Performance von Visualisierungen auf verschiedene Arten beeinflussen lässt.

## 1.1 WebGL

WebGL ist eine leistungsstarke Technologie, die Berechnungen beschleunigt. Allerdings gibt es einige Einschränkungen. WebGL benötigt eine GPU (Grafikkarte), die nicht immer in allen Browsern verfügbar ist. Die CPU verarbeitet alle Hauptfunktionen des Computers, während die GPU viele kleinere Aufgaben gleichzeitig ausführt.[3]

| CPU  | GPU   |
|--|---|
| Generalistische Komponente - Verarbeitet die wichtigsten Verarbeitungsfunktionen eines Computers | Spezialisierte Komponente - Verarbeitet Grafik- und Video-Rendering       |
| Kernanzahl- 2-64 (die meisten CPUs)  | Kernanzahl- Tausende  |
| Führt Prozesse seriell aus   | Führt Prozesse parallel aus   |
| Besser bei der Verarbeitung einer grossen Aufgabe zur gleichen Zeit                              | Besser bei der Verarbeitung mehrerer kleinerer Aufgaben zur gleichen Zeit |

Table 1: CPU vs GPU

Mit WebGL gerenderte Daten werden als Pixelraster gezeichnet, was in einigen Fällen zu verpixelten oder unscharfen Bildern führen kann. Browser begrenzen die Anzahl der WebGL Kontexte, auf die ein bestimmtes Webdokument zugreifen kann, was die Darstellung von WebGL-Visualisierungen einschränken kann. Browser setzen Grenzen für die Höhe und Breite von Visualisierungen, die WebGL verwenden, was bei extrem grossen Plots zu Problemen führen kann.[12]

Um die Performance Unterschiede zu untersuchen habe ich eine Visualisierung erstellt, welche 100000 zufällige Punkte in einem 2D Koordinatensystem darstellt. Einmal wurde die Visualisierung mit dem Modus SVG und einmal mit WebGL gerendert. Was bereits qualitativ auffällt ist, dass die WebGL Visualisierung deutlich schneller angezeigt wird im Chrome Browser.

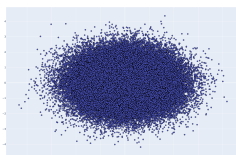


Figure 1: Random dots

Die Chrome Entwickler Tools verfügen über ein Performance Profiler Tool, welches die Performance einer Webseite analysieren kann. Es unterteilt die Laufzeitleistung in das RAIL-Modell[10]: Response, Animation, Idle und Load. Im Kontext zur Visualisierung betrachte man die Scripting, Rendering und Painting Kategorien.[8]

| Kategorie | SVG    | WebGL | %Δ     |
|-----------|--------|-------|--------|
| Scripting | 3778ms | 651ms | -580%  |
| Rendering | 2026ms | 14ms  | -1447% |
| Painting  | 197ms  | 2ms   | -985%  |

Table 2: Performance Unterschiede

Zu beobachten ist ein signifikanter Unterschied in den Rendering und Painting Zeiten der beiden Visualisierungen. Auch die subjektive Wahrnehmung der Performance ist deutlich besser bei der WebGL Visualisierung. Auch die Tradeoffs zwischen Performance und Qualität sind deutlich zu erkennen. Die SVG Visualisierung ist deutlich schärfer und die Punkte sind deutlich besser zu erkennen.



Figure 2: Qualität Unterschiede links SVG, rechts WebGL

## 1.2 Tiling

Tiling kann die Performance von interaktiven Visualisierungen von grossen Datenmengen verbessern, indem es die Daten in kleinere Kacheln aufteilt und nur die benötigten Teile lädt und darstellt. Dadurch wird die Rechenleistung effektiver genutzt und die Belastung auf den Computer verringert. Die interaktive Natur von Tiling ermöglicht es den Benutzern, schnell und einfach zwischen verschiedenen Ausschnitten der Daten zu navigieren und eine schnellere und reaktionsfähigere Erfahrung zu erleben.[9] Datashader ist ein Grafik-Pipeline-System zur schnellen und flexiblen Erstellung aussagekräftiger Darstellungen grosser Datensätze. Datashader unterteilt die Erstellung von Bildern in eine Reihe expliziter Schritte, die es ermöglichen, Berechnungen an Zwischendarstellungen vorzunehmen.[2]

Wir visualisieren hier die räumliche Verteilung der Taxifahrten in New York City. Die Daten werden von Plotly zu Verfügung gestellt und sind unter folgendem Link zu finden: <https://raw.githubusercontent.com/plotly/datasets/master/uber-rides-data1.csv>

| Date/Time | The date and time of the Uber pickup |
|-----------|--------------------------------------|
| Lat       | The latitude of the Uber pickup      |
| Lon       | The longitude of the Uber pickup     |

Table 3: Newyork Uber Data

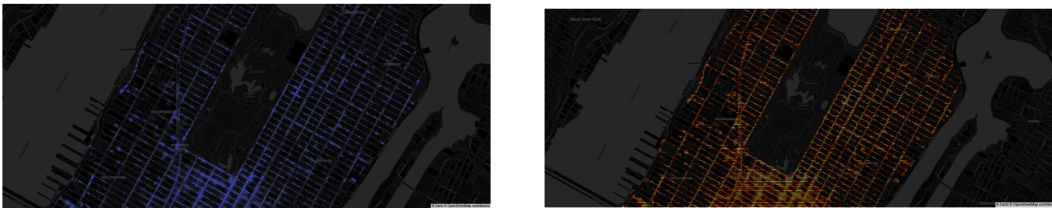


Figure 3: Taxifahrten in Newyork (Plotly links, Datashader Tiles rechts)

Zu beobachten ist das die unprozessierten Daten mit Plotly zwar schöner dargestellt werden. Aber viel länger brauchen um angezeigt zu werden. Die Datashader Visualisierung ist deutlich schneller und die Daten werden deutlich besser dargestellt. Diese Beobachtungen widerspiegeln sich auch in den Performance Daten. Zwar wird nur die Scripting Zeit signifikant verkürzt.

| Kategorie | Plotly | Datashader Tile | % $\Delta$ |
|-----------|--------|-----------------|------------|
| Scripting | 1241ms | 492ms           | -39%       |
| Rendering | 6ms    | 6ms             | $\pm 0\%$  |
| Painting  | 12ms   | 12ms            | $\pm 0\%$  |

Table 4: Performance Unterschiede

Spannender bei diesem Fall sind die Unterschiede des genutzten Speichers. Wenn wir den JavaScript Heap betrachten sehen wir einen signifikanten Unterschied.[11] Plotly braucht fast 130MB um die Daten zu visualisieren, der Datashader Tile braucht nur 23MB.

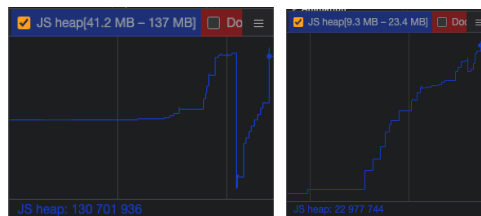


Figure 4: Memory Heap Plotly vs. Datashader Tile

Einen ähnlichen Ansatz verfolgen auch Frameworks wie Deck.gl. Diese Frameworks sind jedoch auf WebGL basiert und können somit nur auf Browsern mit WebGL Unterstützung verwendet werden. Sie prozessieren für verschiedene Ansichten voragregierte Tiles und können so sehr schnell und performant visualisieren.[5]

## 2 LE2: Grundsätze für die Gestaltung von Dashboards

Ein wichtiger Aspekt bei der Entwicklung von Dashboards ist das Konzept. Dabei hat der Benutzer Zugriff auf verschiedene Steuerelemente und mehrere verknüpfte Visualisierungen. Diese Designprinzipien sind entscheidend für die Entwicklung eines erfolgreichen Dashboards.

### 2.1 Schneiderman's Mantra

Schneiderman's Mantra ist ein Organisationsprinzip für die Erstellung von Visualisierungssystemen. Es lautet wie folgt:

**Overview first:** Im Überblicksschritt wird der gesamte Datensatz in einer geeigneten Anzeigemethode dargestellt. Dies bietet einen Überblick auf hoher Ebene und gibt Kontext für die nächsten Schritte in der Visualisierung.

**Zoom and filter:** Zoomen auf einen Abschnitt des Datensatzes ermöglicht das Entfernen von überflüssigen Daten anhand der angezeigten Koordinaten. So erhalten die relevanten Daten mehr Auflösung und Detail. Das Filtern entfernt überflüssige Daten basierend auf gewünschten Attributen, vereinfacht die Anzeige Ihrer Daten und schafft mehr Platz für Details.

**Details on demand:** Details auf Abruf geben dem Benutzer Kontrolle über die Daten und ermöglichen es, ohne Überladen des Bildschirms weiter zu erkunden. Das Tooltip ist die häufigste Implementierung. Eine andere Möglichkeit besteht darin, ein Feld auszuwählen und die Daten hervorzuheben.[7]

Eine mögliche Umsetzung mit diesem Prinzip könnte wie folgt aussehen. Im ersten Schritt wird eine Übersicht über die Anzahl Filme pro Genre gegeben.

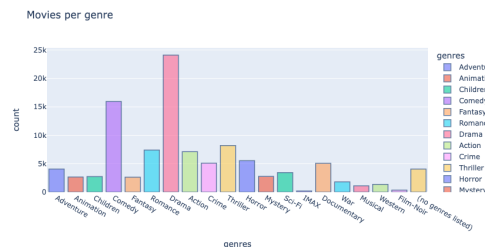


Figure 5: Genre Übersicht

Wenn der Benutzer nun auf ein Genre klickt oder auf ein Genre zoomt wird die Anzahl Filme pro Jahr angezeigt.

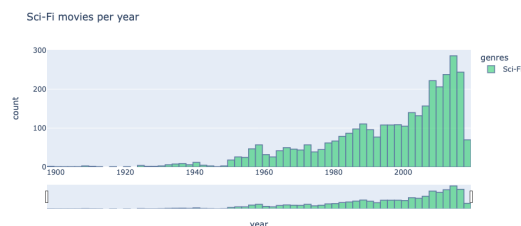


Figure 6: Filtering/Zooming: Anzahl Filme pro Jahr

Mit einem Tooltip kann der Benutzer nun die Details zum einzelnen Film einsehen.

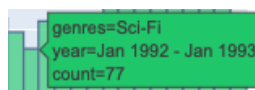


Figure 7: Tooltip: Details on demand

Diese Interaktionen können mit Hilfe Plotly.js Chart Events realisiert werden. Wenn ein solches Event getriggert wird kann die Funktion `Plotly.restyle` aufgerufen werden. Diese Funktion ermöglicht es die Daten eines Charts zu ändern.

## 2.2 Verknüpfte Ansichten

Das Paradigma der verknüpften Ansichten ist eine Methode, die mehrere einfache Ansichten von Daten verwendet. Wenn Sie mit einer Ansicht interagieren, ändert sich die Anzeige der Daten in allen verknüpften Ansichten. Ein einfaches Beispiel wäre, dass die Auswahl eines Datums in einer Ansicht die Daten in allen anderen Ansichten auch ändert. Kurz gesagt wenn sich eine Ansicht ändert, ändern sich alle anderen Ansichten auch.[17]

## 2.3 Brushing

Brushing ist eine Methode, die es ermöglicht, Daten in einer Ansicht zu selektieren und diese dann in anderen Ansichten zu verwenden. Ein Beispiel wäre, dass Sie eine Ansicht mit einem Scatterplot haben, in der Sie einen Bereich selektieren. Diese Selektion wird dann in einer anderen Ansicht mit einem Histogramm verwendet.[1]

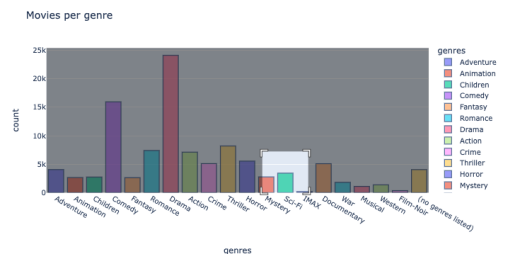


Figure 8: Brushing

## 2.4 Filtering

Dynamische Abfragen sind eine direkte Möglichkeit, um Daten mit Schiebereglern, Knöpfen und Kategorien zu filtern. Sie zeigen globale Eigenschaften auf und helfen bei der Beantwortung spezifischer Fragen ohne Fehlermeldungen. Fortgeschrittene Filter erlauben in der Regel OR-Kombinationen innerhalb eines Attributs oder AND-Kombinationen über Attribute hinweg. So können komplexere Abfragen gestellt werden.[13]

## 2.5 Implementation mit Dash

Mit Dash habe ich ein interaktives Dashboard entwickelt, welches die oben genannten Prinzipien implementiert. Das Dashboard ist in Python geschrieben und basiert auf dem Dash Framework. Dash ist ein Framework, welches auf Plotly.js basiert und es ermöglicht interaktive Webanwendungen zu entwickeln.[4]

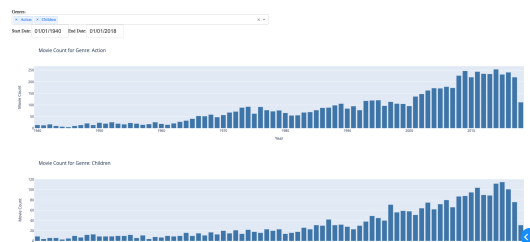


Figure 9: Dashboard

Mit einem Mult-Select können die Genres ausgewählt werden. Es wird eine zusätzliche verknüpfte Ansicht angezeigt, welche die Anzahl Filme pro Jahr anzeigt. Zudem kann mit einem Datepicker ein Zeitraum ausgewählt werden. Dieser wird dann in allen anderen Ansichten verwendet.

### 3 LE3: HCI Grundlagen

Das Ziel von HCI ist es, die Interaktion zwischen Menschen und Computern zu untersuchen und zu verbessern. In diesem Abschnitt werden die Grundlagen von HCI untersucht und die wichtigsten Prinzipien vorgestellt.

#### 3.1 Fitts's Law

Fitts' Gesetz besagt, dass die Zeit, die benötigt wird, um einen Zeiger zu einem Zielbereich zu bewegen, eine Funktion aus Entfernung und Zielgrösse ist. Es wird in der Benutzenerfahrung und -schnittstellengestaltung verwendet, um Schaltflächen gross zu machen und den Abstand zwischen der Aufgabe und der zugehörigen Schaltfläche kurz zu halten. Es ist wichtig zu beachten, dass dieses Gesetz auf schnelle, zeigende Bewegungen anwendbar ist. Durch die Anwendung dieses Gesetzes haben standardisierte Benutzeroberflächenelemente die Zeit reduziert und die Produktivität gesteigert.[15]

$$ID = \log_2 \left( \frac{2D}{W} \right) \quad (1)$$

ID = Index of Difficulty

D = Distance to target

W = Width of target

Wenn dieses Gesetz auf Figure 5 anwenden. Sollte man mit einer möglichst kleinen Bewegung die gewünschte Genre auswählen können. In Plotly kann dies mit dem **hovermode** umsetzen. So kann zum Beispiel das Genre IMAX sehr schnell ausgewählt werden und muss nicht mit einem präzisen Klick das Genre auswählen.

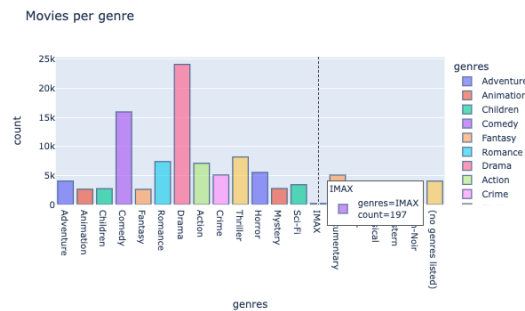


Figure 10: Fitts's Law hover mode

#### 3.2 Weber's Law

Webers Gesetz beschreibt die menschliche Fähigkeit, kleine Unterschiede in Stimuli wie Formen, Farben und Gewichten zu erkennen. Es besagt, dass das Verhältnis des kleinsten Unterschieds  $\Delta I$  den eine Person wahrnehmen kann, zum Anfangsreizwert  $I$  für eine bestimmte Messung konstant  $K$  ist. Die Weber-Fraktion repräsentiert die Steigung dieses Verhältnisses, und die Grösse des gerade wahrnehmbaren Unterschieds ist ein konstanter Anteil des ursprünglichen Stimuluswerts. Dieses Gesetz hat Auswirkungen für Grafikdesigner, die es nutzen können, um effektivere Benutzeroberflächen zu entwerfen, indem sie sicherstellen, dass es ausreichende Unterscheidung zwischen ähnlichen Stimuli wie Linienstärken oder Farbschattierungen gibt.[14]

$$\frac{\Delta I}{I} = K \quad (2)$$

$\Delta I$  = Delta to Initial Stimulus Value

$I$  = Initial Stimulus Value

$K$  = Weber Fraction

### 3.3 5 Dimensionen der Interaktionsgestaltung

Die fünf Dimensionen der Interaktionsgestaltung wurden von Gillian Crampton Smith, Professor am Royal College of Art in London, und Kevin Silver, einem erfahrenen Interaktionsdesigner, definiert. Die Dimensionen (1D) Worte, (2D) visuelle Darstellungen, (3D) physische Objekte/Räume, (4D) Zeit und (5D) Verhalten werden von Interaktionsdesignern genutzt, um die Interaktion zwischen Nutzern und einem Produkt oder einer Dienstleistung in ganzheitlicher Weise zu betrachten.[16]

#### 1D: Worte

Die Verwendung von passenden und leicht verständlichen Wörtern ist im Interaction Design wichtig, um eine reibungslose Interaktion zwischen dem Benutzer und dem Produkt oder der Dienstleistung zu gewährleisten.

#### 2D: Visuelle Darstellungen

Visuelle Elemente wie Bilder, Iconografie und grafische Darstellungen tragen zur Kommunikation zwischen Benutzer und Produkt bei und können genauso wirkungsvoll wie Text sein.

#### 3D: Physische Objekte/Räume

Das Medium, über das der Benutzer mit dem Produkt interagiert, wie zum Beispiel ein Mobil- oder Tablet-Bildschirm oder eine Computermouse, muss bei der Gestaltung für eine einfache Benutzerfreundlichkeit berücksichtigt werden.

#### 4D: Zeit

Elemente, die sich im Laufe der Zeit verändern, wie Animationen und Videos, können Benutzer einbinden und ihr Erlebnis mit dem Produkt verbessern. Fortschrittsbalken-Animationen können auch hilfreich sein, um den Fortschritt eines bestimmten Prozesses oder einer Operation anzuzeigen.

#### 5D: Verhalten

Das tatsächliche Verhalten der Anwendung, einschliesslich Aktionen, Reaktionen und Präsentationen, muss so gestaltet werden, dass es für Benutzer leicht anpassbar und verständlich ist. Zum Beispiel das Anzeigen einer Erfolgsmeldung mit einer Zusammenfassung, wenn eine Aufgabe abgeschlossen ist oder das Einbinden von Wischaktionen.[6]

### 3.4 Implementation mit Dash

Rückblickend auf mein Dashboard aus LE2, kann ich sagen, dass ich die meisten dieser Prinzipien umgesetzt habe. Aber es gibt noch einige Punkte, welche ich verbessern könnte.

Zum Beispiel bracht es mehr Zeit das Jahr auszuwählen in einem "Date Picker" als in einem Range Slider.

Zudem habe ich die verschiedenen Genres mit Hilfe von Farben mehr differenziert und den Hovermode auf der X-Achse aktiviert

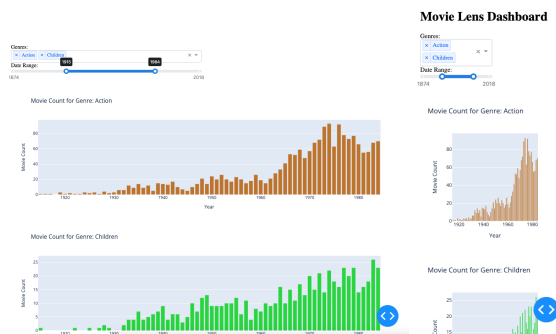


Figure 11: Dashboard mit Verbesserungen / Mobile

Zudem habe ich die dritte Dimension der Interaktionsgestaltung untersucht. Wie sieht das Dashboard auf einem Smartphone aus?

Können die verschiedenen Elemente noch gut ausgewählt werden?

Kann man die Daten gleich gut interpretieren?

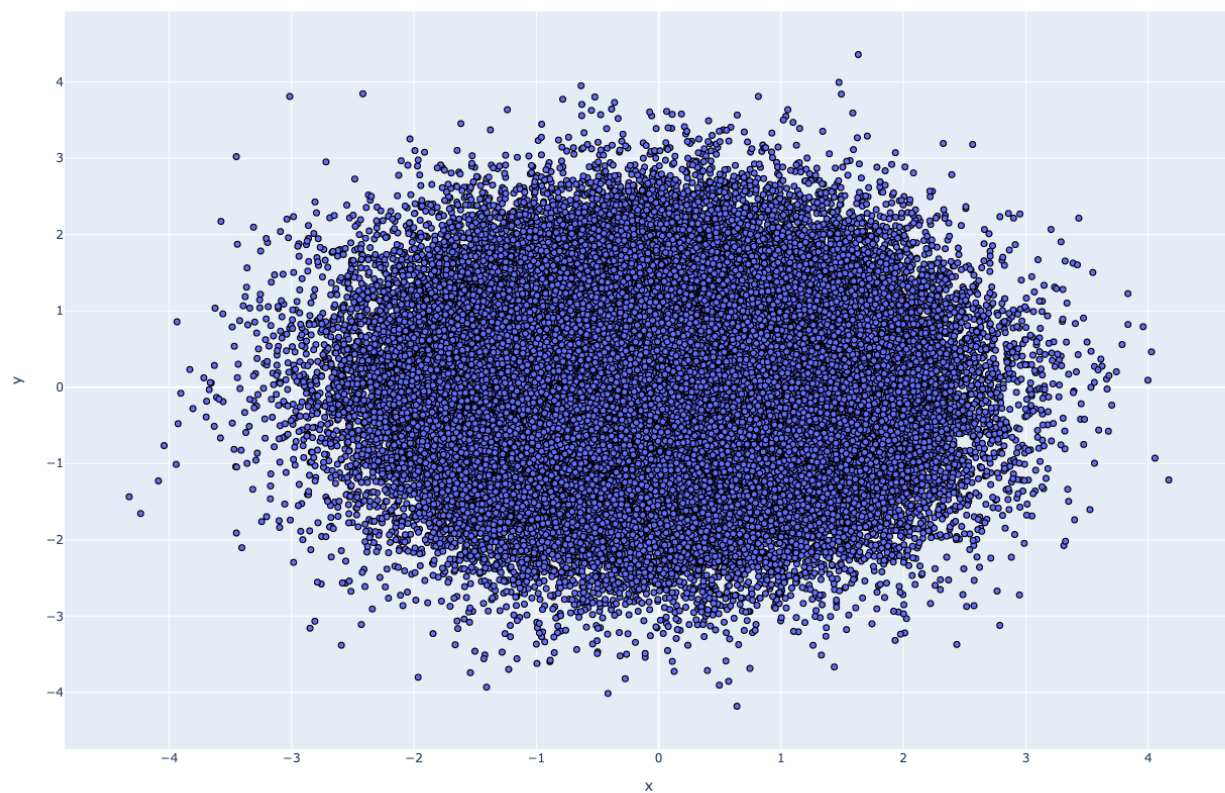
## References

- [1] Richard A. Becker and William S. Cleveland. “Brushing Scatterplots”. In: *Technometrics* 29.2 (May 1987), pp. 127–142. ISSN: 0040-1706, 1537-2723. DOI: 10.1080/00401706.1987.10488204. URL: <http://www.tandfonline.com/doi/abs/10.1080/00401706.1987.10488204> (visited on 04/13/2023).
- [2] Joseph A. Cottam, Andrew Lumsdaine, and Peter Wang. “Abstract rendering: out-of-core rendering for information visualization”. In: IS&T/SPIE Electronic Imaging. Ed. by Pak Chung Wong et al. San Francisco, California, USA, Dec. 23, 2013, 90170K. DOI: 10.1117/12.2041200. URL: <http://proceedings.spiedigitallibrary.org/proceeding.aspx?doi=10.1117/12.2041200> (visited on 03/18/2023).
- [3] *CPU vs. GPU: What’s the Difference?* URL: <https://www.cdw.com/content/cdw/en/articles/hardware/cpu-vs-gpu.html> (visited on 04/13/2023).
- [4] *Dash Overview*. URL: <https://plotly.com/dash> (visited on 04/13/2023).
- [5] *deck.gl. Home — deck.gl*. URL: <https://deck.gl/> (visited on 03/19/2023).
- [6] Akshay Devazya. *Interaction design and its 5 dimensions*. Medium. Dec. 10, 2022. URL: <https://uxdesign.cc/interaction-design-and-its-dimensions-39ca7e1d09f0> (visited on 05/07/2023).
- [7] HAMPDATAVISUALIZATION. *Schneiderman’s Mantra*. Data Visualization. Feb. 26, 2016. URL: <https://hampdatavisualization.wordpress.com/2016/02/26/schneidermans-mantra/> (visited on 04/13/2023).
- [8] Kayce Basques. *Analyze runtime performance*. Chrome Developers. URL: <https://developer.chrome.com/docs/devtools/performance/> (visited on 03/18/2023).
- [9] *Map Tiles: Everything You Need To Know*. URL: <https://carto.com/blog/map-tiles-guide> (visited on 04/13/2023).
- [10] *Measure performance with the RAIL model*. web.dev. URL: <https://web.dev/rail/> (visited on 03/19/2023).
- [11] *Memory management - JavaScript — MDN*. Feb. 23, 2023. URL: [https://developer.mozilla.org/en-US/docs/Web/JavaScript/Memory\\_Management](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Memory_Management) (visited on 03/19/2023).
- [12] Plotly. *Webgl vs svg in Python*. WebGL vs SVG in Python. URL: <https://plotly.com/python/webgl-vs-svg/> (visited on 03/18/2023).
- [13] B. Shneiderman. “The eyes have it: a task by data type taxonomy for information visualizations”. In: *Proceedings 1996 IEEE Symposium on Visual Languages*. Proceedings 1996 IEEE Symposium on Visual Languages. ISSN: 1049-2615. Sept. 1996, pp. 336–343. DOI: 10.1109/VL.1996.545307.
- [14] *Virtual lab*. URL: [https://hci-iitg.vlabs.ac.in/webers\\_low.html](https://hci-iitg.vlabs.ac.in/webers_low.html) (visited on 05/07/2023).
- [15] *What is Fitts’ Law?* The Interaction Design Foundation. URL: <https://www.interaction-design.org/literature/topics/fitts-law> (visited on 05/06/2023).
- [16] *What is Interaction Design?* The Interaction Design Foundation. URL: <https://www.interaction-design.org/literature/topics/interaction-design> (visited on 05/07/2023).
- [17] Graham Wills. “Linked Data Views”. In: *Handbook of Data Visualization*. Ed. by Chun-houh Chen, Wolfgang Härdle, and Antony Unwin. Springer Handbooks Comp.Statistics. Berlin, Heidelberg: Springer, 2008, pp. 217–241. ISBN: 978-3-540-33037-0. DOI: 10.1007/978-3-540-33037-0\_10. URL: [https://doi.org/10.1007/978-3-540-33037-0\\_10](https://doi.org/10.1007/978-3-540-33037-0_10) (visited on 04/13/2023).

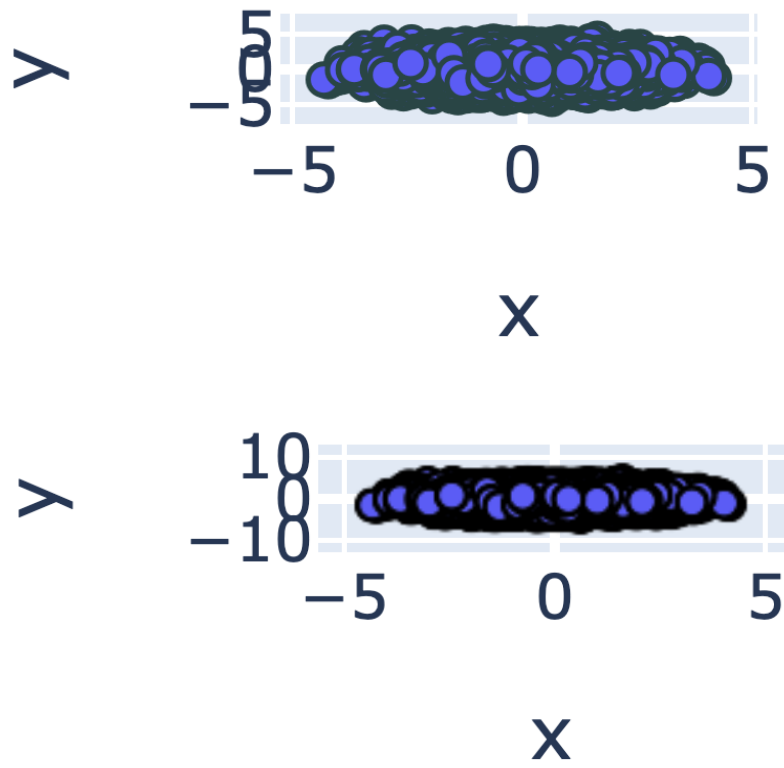


## 4 Anhang

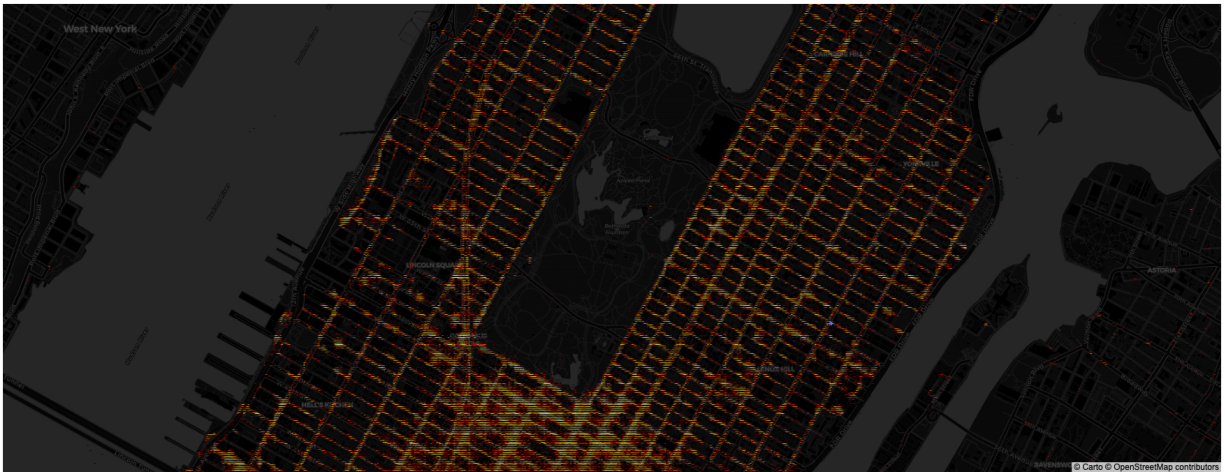
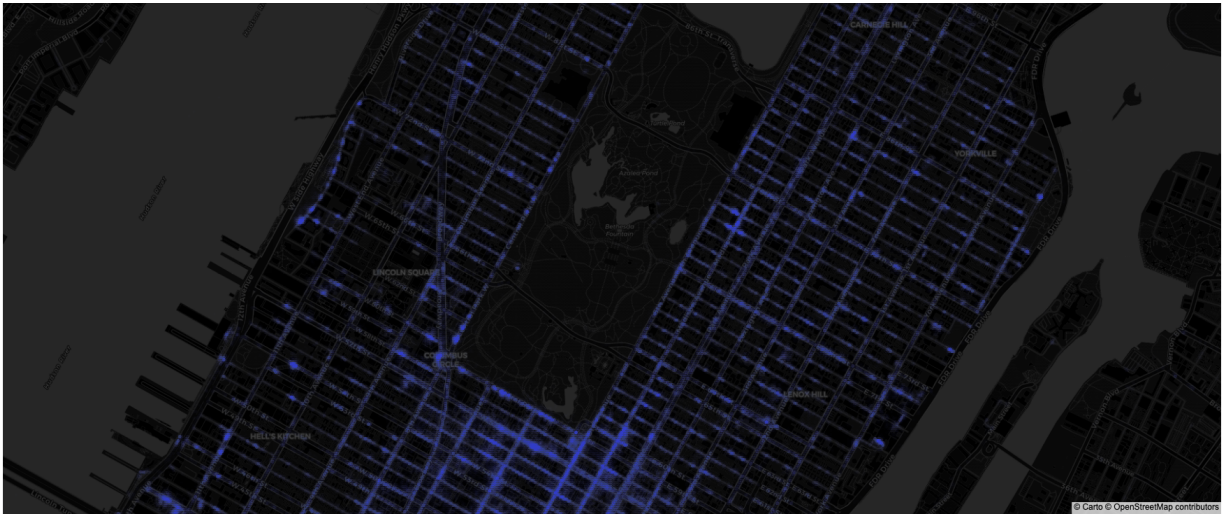
4.1 Figure 1



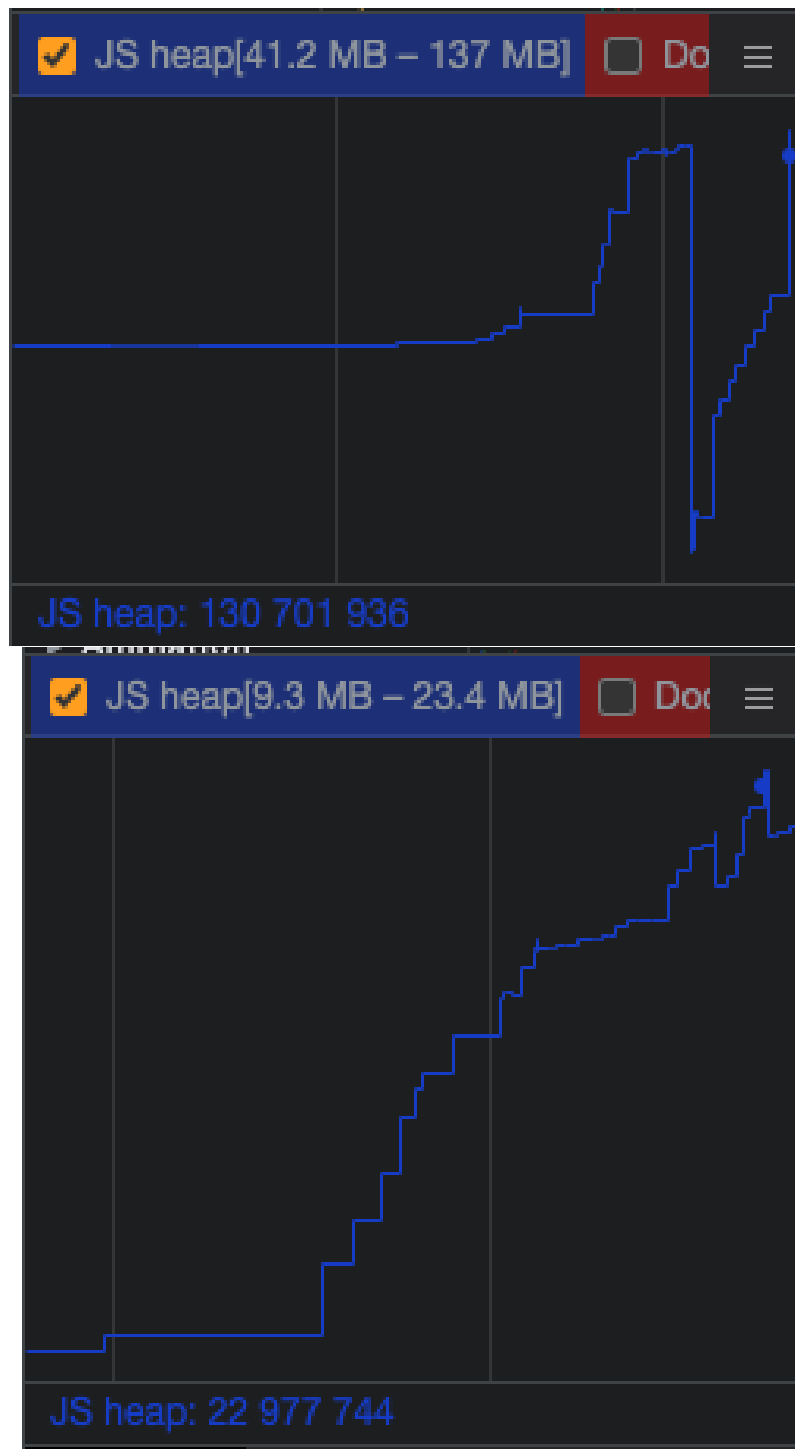
4.2 Figure 2



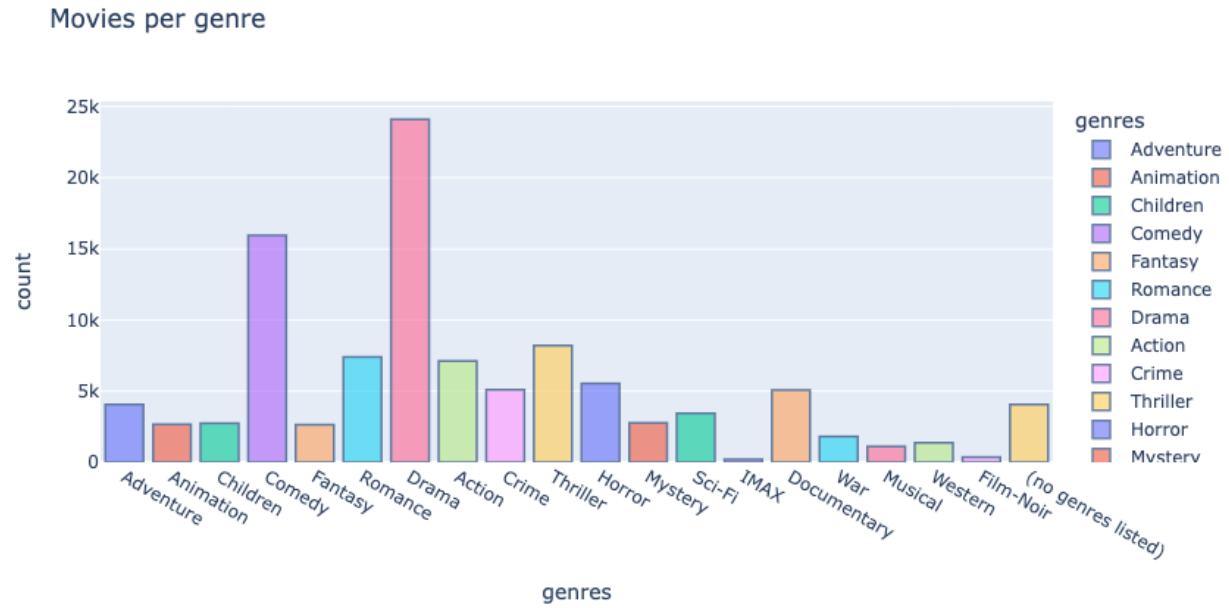
### 4.3 Figure 3



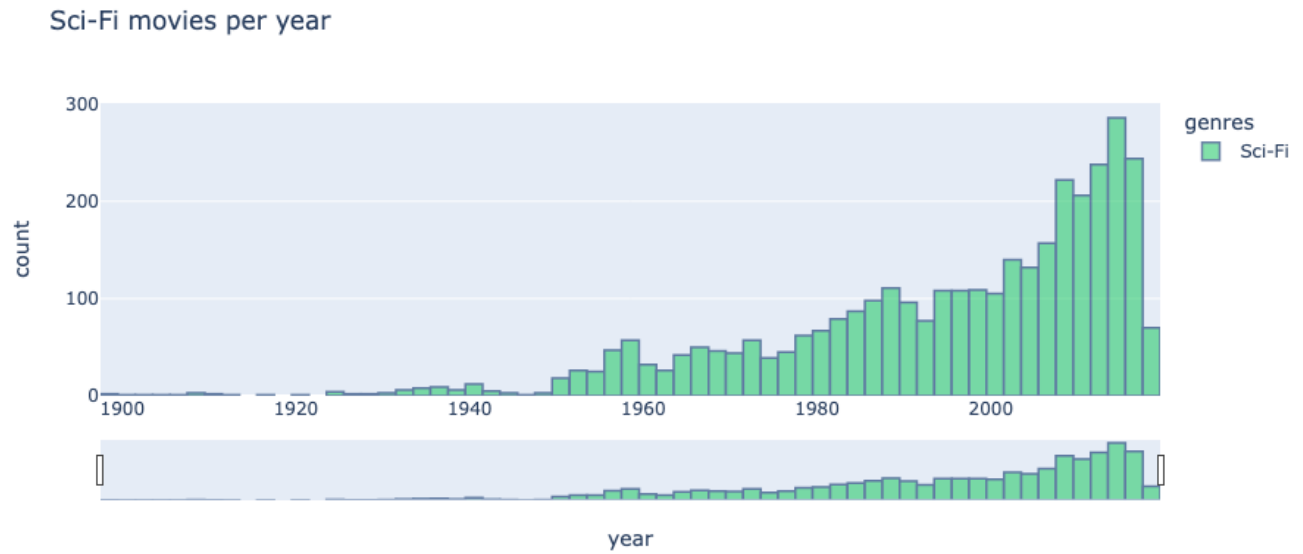
4.4 Figure 4



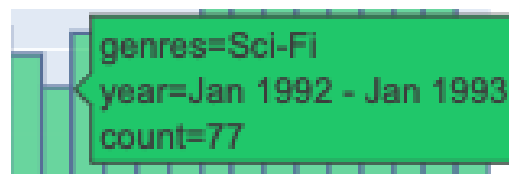
#### 4.5 Figure 5



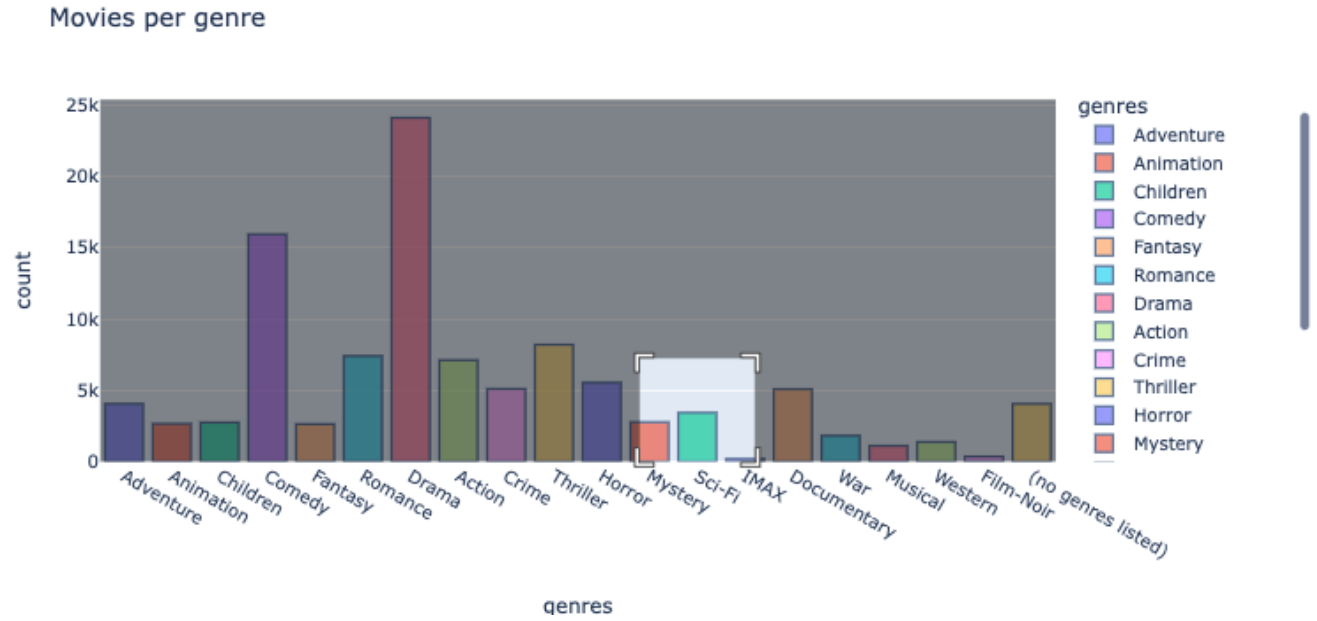
#### 4.6 Figure 6



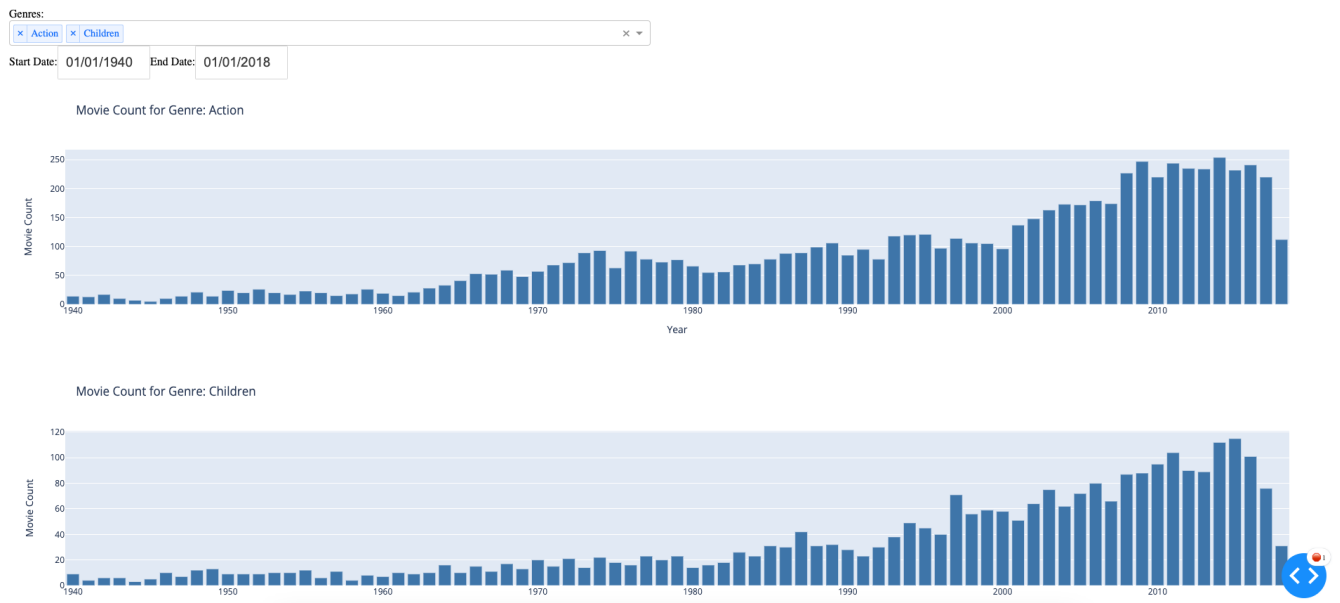
#### 4.7 Figure 7



4.8 Figure 8

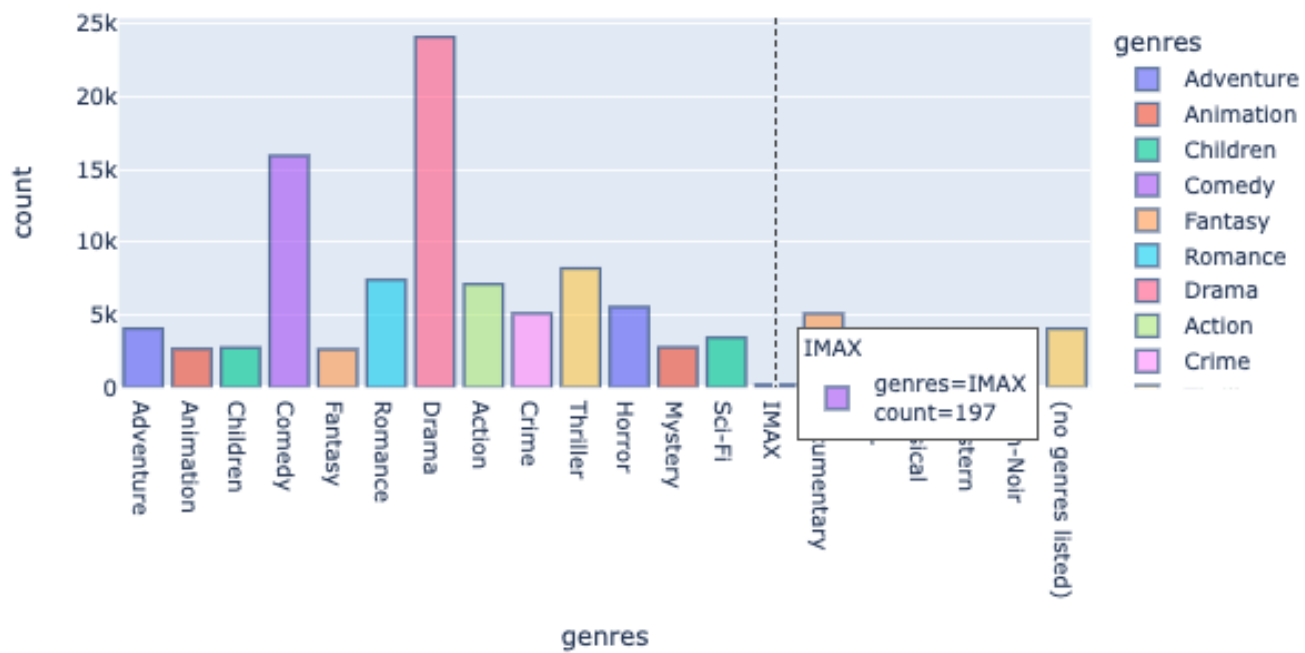


4.9 Figure 9



4.10 Figure 10

Movies per genre

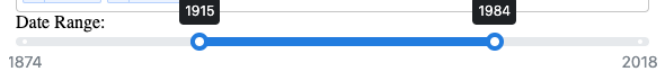


## 4.11 Figure 11

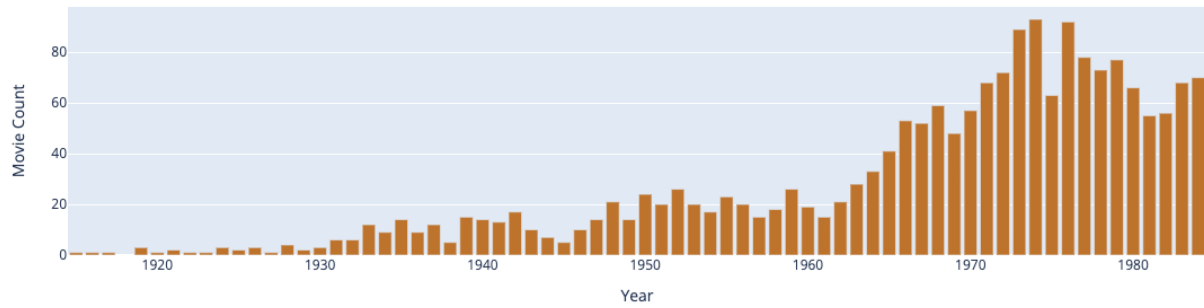
Genres:

☒ Action ☒ Children ☐

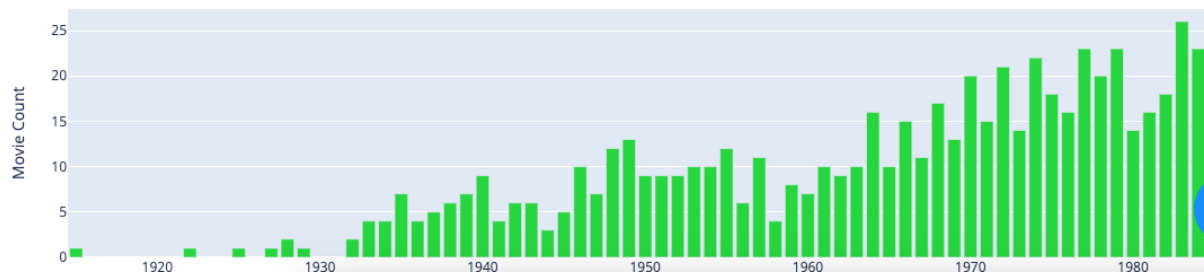
Date Range:



Movie Count for Genre: Action



Movie Count for Genre: Children





# Movie Lens Dashboard

Genres:

×

Action

×

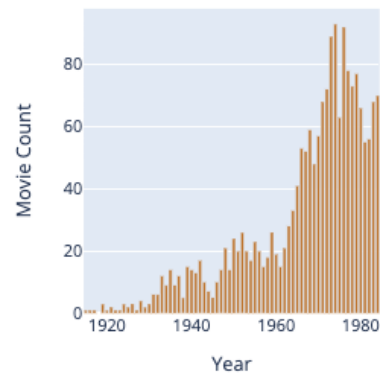
Children

×

Date Range:



Movie Count for Genre: Action



Movie Count for Genre: Children

