Lecture 10: Interactivity

Today’s Visualization

Is over at The Pudding

In general, **The Pudding** is a source of very many very inspiring pieces of narrative visualization. Explore!

Why Interaction?

> **Incorporating interactivity and reactivity into visualization…**
>
> > …expands the physical limits of what can be consumed in a given space.
> >
> > …broadens the variety of analyses to serve different curiosities within a project.
> >
> > …facilitates manipulations of data to accommodate varied interrogations.
> >
> > …amplifies the overall control and potential customisation of an experience.
> >
> > …increases the range of techniques for engaging users with dynamic displays.

(Andy Kirk, Handbook of Data Visualization)

Types of Interaction

Yi, Kang, Stasko, Jacko, *Toward a Deeper Understanding of the Role of Interaction in Information Visualization*, InfoVis 2007 establish 7 categories of interactions in visualization:

1. **Select**: mark something as interesting
2. **Explore**: show me something else
3. **Reconfigure**: show me a different arrangement
4. **Encode**: show me a different representation
5. **Abstract/Elaborate**: show me more or less detail
6. **Filter**: show me something conditionally
7. **Connect**: show me related items

Munzner discusses:

1. **Change View over Time** - orthogonal to Yi et al
2. **Select Elements** - ie **Select**
3. **Changing Viewpoint** - ie **Reconfigure**
4. **Reducing Attributes** - ie **Abstract** or **Filter**

Types of Interaction

Wilkinson discusses (Chapter 17):

- UI Methods for constructing graph specifications
- Exploration

1. **Filtering**

   - by category
   - by range

2. **Navigating**

   - **zooming**
   - **panning**
   - **lensing**

3. **Manipulating** - Node dragging - Categorical reordering
4. **Brushing and Linking** - ie selecting in one view and seeing effects in multiple views
5. **Animating**
6. **Rotating**
7. **Transforming**

Specifying Interactions

Modern day GUI tool-kits almost universally work with an **event-driven** paradigm.
The software system runs an infinite loop that manages reactions and keeps a roster of functions that are responsible for reacting to possible user events.
We distinguish between **event** (the user action), **control** (the feature to which the action is applied) and **function** (the operation that is performed).

Mouse	Keyboard	Touchscreen	Window	I/O	Timer
move	key press	finger tap	resize	start	time elapsed
drag	key release	drag	minimize	done	
click		pinch	maximize	error	
double click		rotate	restore		
button press			close		
button release					

Common types of GUI events

Filtering

Example Events and Controls

- Select a button or a link
- Select an item from a menu list
- Select multiple items from a check-box or menu list
- Alter the state of a toggle or radio button
- Alter the position of a handle along a scale slider
- Alter the position of two handles along a scale slider (range selector)
- Enter a value into an input box
- Select a mark within a chart
- Select a mark within a legend

Example Functions

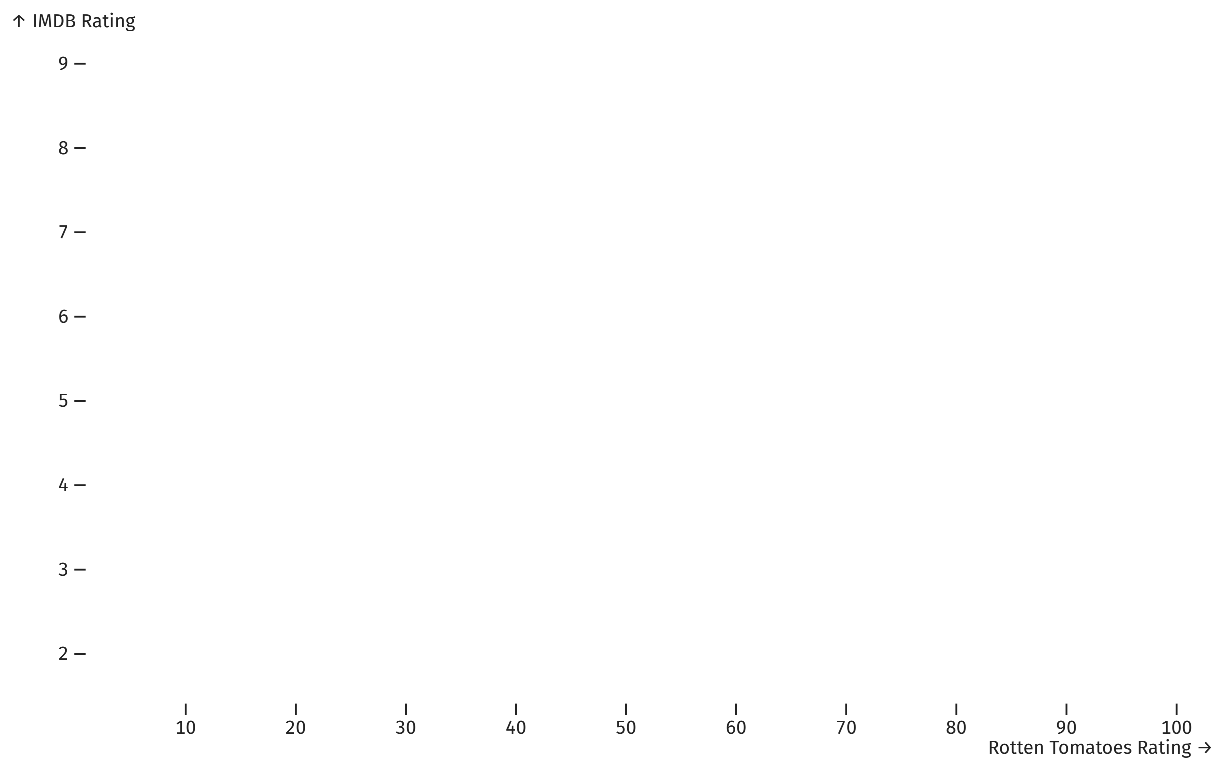
- Apply a categorical data filter (one or several combinations)
- Apply a quantitative data filter (one value or a range)
- Reset all values to their original state

Filtering

▶ Code

Major Genre
Action
Adventure
Black Comedy
Comedy
Concert/Performance
Documentary
Drama
Horror
Musical
Romantic Comedy
Thriller/Suspense
Western

▶ Code



Highlighting

Example Events and Controls

- Select a button or a link
- Select an item from a menu list
- Select multiple items from a check-box or menu list
- Alter the state of a toggle or radio button
- Alter the position of a handle along a scale slider
- Alter the position of two handles along a scale slider (range selector)
- Enter a value into an input box
- Select a mark within a chart
- Mouseover a mark from within a chart
- Select a range of marks from within a chart (**brushing**)
- Select a mark within a legend

Example Functions

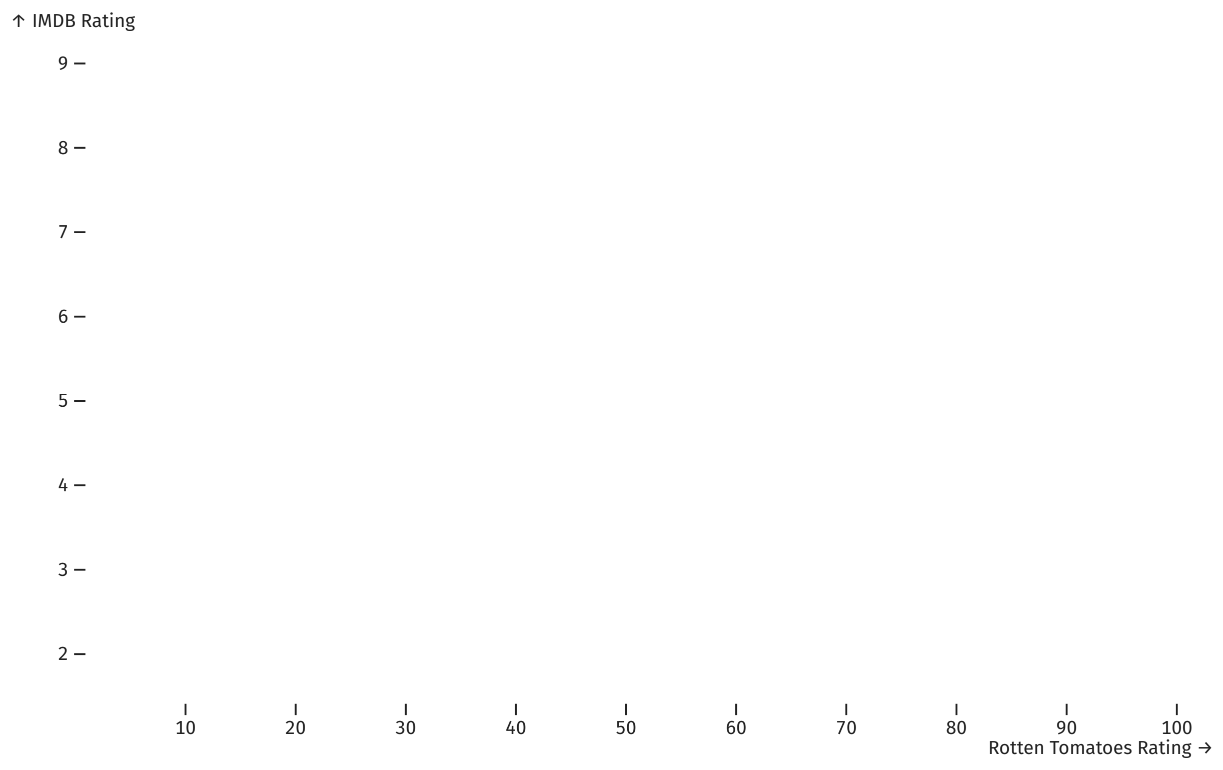
- Highlight selection
- Highlight values based on selection
- Highlight associations between selected values
- Rearrange the order of the data
- Form calculations based on selection

Highlighting

▶ Code

Major Genre
Action
Adventure
Black Comedy
Comedy
Concert/Performance
Documentary
Drama
Horror
Musical
Romantic Comedy
Thriller/Suspense
Western

▶ Code



Navigating

Example Events and Controls

- Select a button (eg zoom level)
- Select tab elements (eg dot stepper)
- Scroll in or out
- Pinch
- Select region from map or menu
- Select, hold, and draw a region of interest
- Select, hold, and move
- Alter the position of a single handle along a scale slider
- Sideward scroll

Example Functions

- Zoom in and out
- Navigate (“Pan”) around a detailed display
- Navigate through a sequence of discrete pages (“scrollytelling”)
- Navigate through a sequence of displays (within the page)
- Navigate through a gradual unveiling of a visualization

Factors Influencing Design

Constraints

- Technical skills
- Development time
- Capabilities of platforms
- Expected user devices

Purpose

- Interactivity, especially combined with gradual revealing of features, may serve to combine for instance an explanatory design with an exploratory interface
- ie, first show what you want to say, then let the user continue to tweak and explore the visualization

Data Representation

- Access to filters, highlights, zooming, panning empowers a user to better understand a complex visualization.

Factors Influencing Design

Trustworthy Design

Interactivity expands the number of ways in which a user can lose trust in your design.

Does your design do what it promises? Can the user trust the functions that it performs?

Will the data grow or change, and is the visualization code robust to expected changes?

Elegant Design

Just because you **can** design with interactivity does not necessarily mean that you **should**.

Strike a balance between overwhelming a user or restricting a user.

Consider how you communicate the affordances of your design.

Factors Influencing Design

Accessible Design

Enabling interactivity also expands the scope of potential access problems.

Example: Tableau - visualizations are interactive by default, with good default choices - but for a smooth experience, users need to go through the Tableau platform itself.

Example: Animation - useability depends a lot on what you want to show, what tasks you want to enable. Sensitive to speed (rapid sequences mean the user might miss changes; slow sequences mean the user might get bored and lose attention)

Can you do the same thing statically? (small multiples is a commonly used option)

Python / Matplotlib

Default interaction set: pan, zoom, toggle guide displays, undo stack.
For functionality in Jupyter Notebooks / JupyterLab, make sure the `ipympl` package is installed and include the cell-magic `%matplotlib widget` to select and enable the corresponding backend.
For more than this, users need to place widgets (available in `ipywidgets`) and bind them to handlers with the `interact`

Python / Altair

Vega, Vega-Lite and Altair have an extensive abstraction for interactivity (…which changed with the shift to Altair v. 5, that is currently a release candidate)

1. Good defaults (zoom, pan) come with merely calling `.interactive()` on the `Chart` object.
2. Fundamental building block is the **parameter**, which comes in two versions: **variables** or **selections**.
3. Variables are created with `altair.param()` and bind to either input contols or selection methods.
4. Selections implement click-to-select, mutating selections, brushing through `selection_point` and `selection_interval`.
5. Influence the Chart by using `altair.condition([condition], [if true], [if false])` to create values that change depending on the interaction.

Python / Altair

Using the `Tooltip` encoding, an input slider for highlighting, and `.interactive()`

▶ Code

Minimal US gross ● 50

Python / Altair

With clickable genre legend and brushing that controls a linked gross earnings histogram.

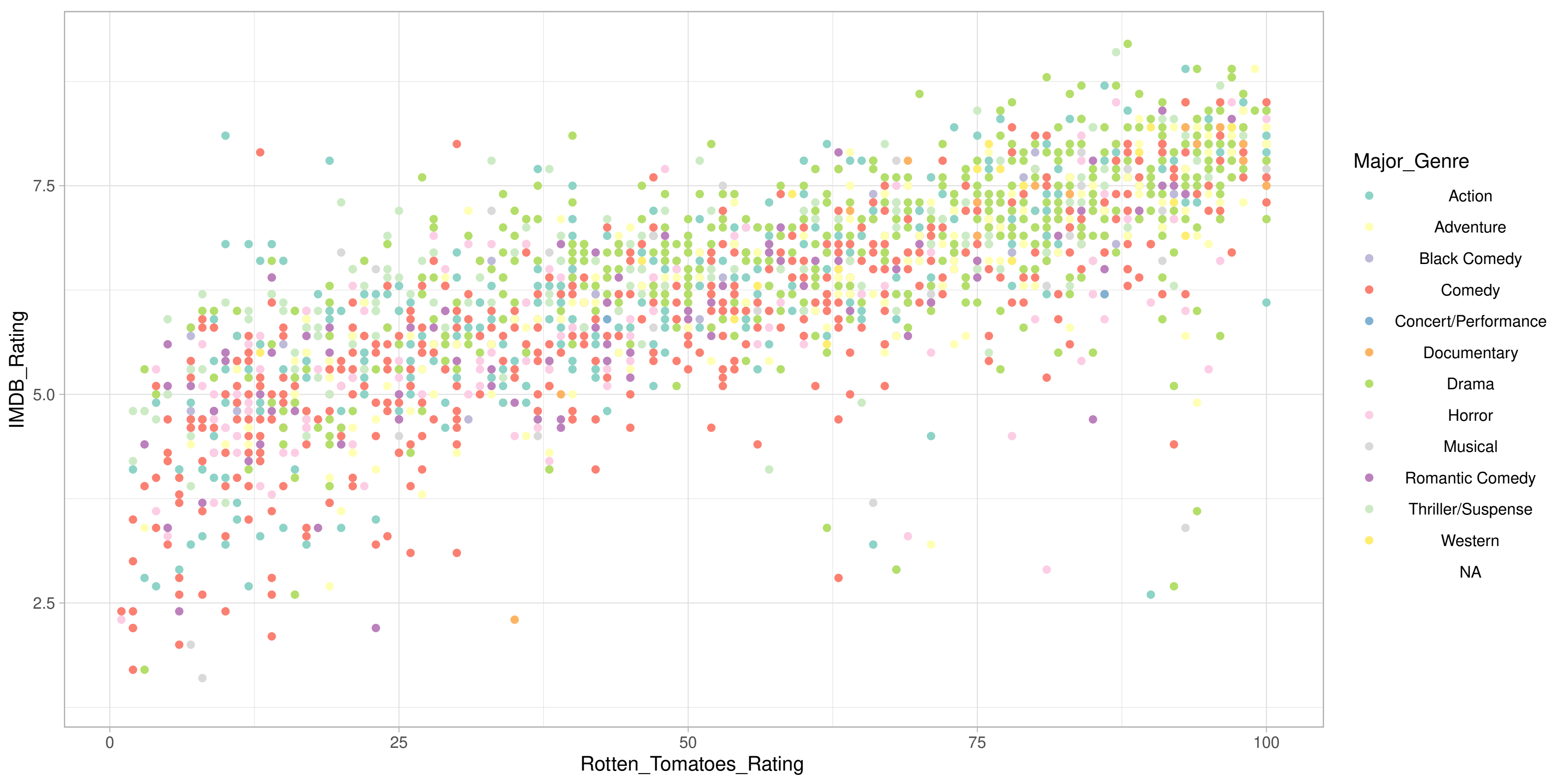
▶ Code

R / ggplot2

The `ggplotly` package lets you convert a `ggplot2` chart to a `plotly` chart that can be embedded in webpages and interactive. Comes with decent default interactions, but getting anything except these interactions is difficult.
The `ggiraph` package lets you switch to *interactive geoms*, and *interactive aesthetics*.

R / ggplot2

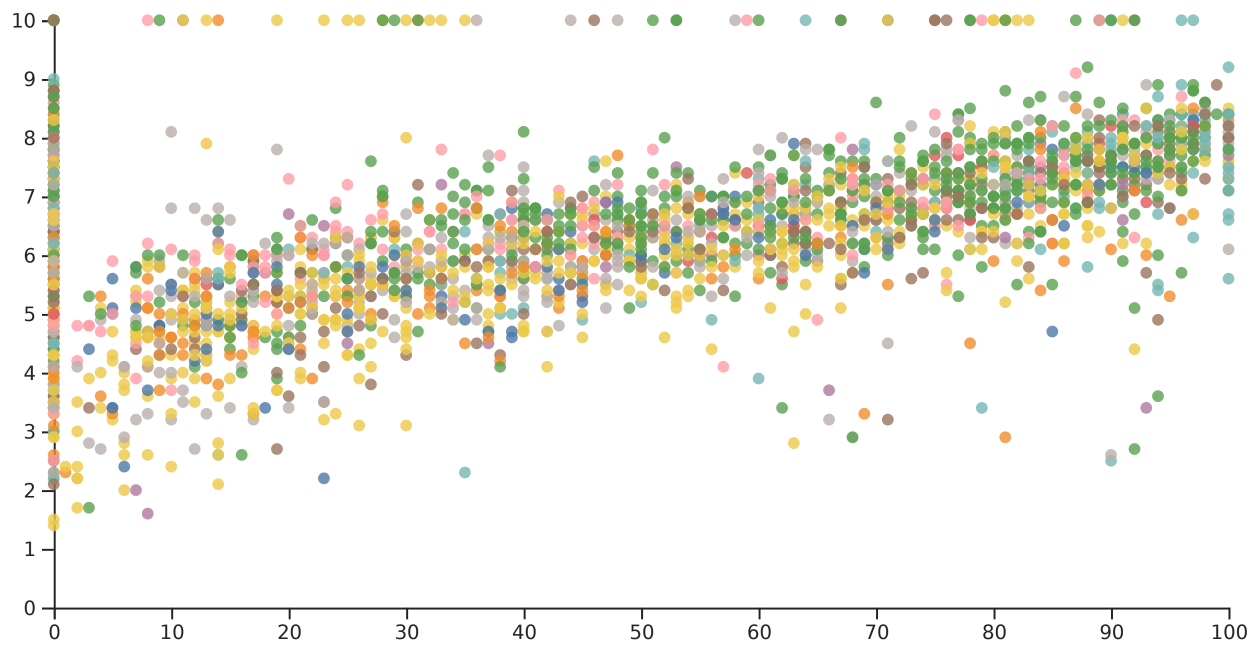
▶ Code



Javascript / Observable.js / D3.js

Here is where interactivity can really shine - Javascript is a frontend language, dedicated to representing interactions.
Interactivity in Observable.js primarily rests on **Inputs** - any widgets you might want, as well as clickable or searchable tables. Unfortunately, brushing is still a work in progress (as of Spring 2023), so direct interactivity in Observable.js is limited to what we can do with these input widgets.
In pure D3.js the situation is better, in that D3.js makes it easy to write and assign event handlers. The price is that we'll need to write more of the graph code ourselves.

Javascript / Observable.js / D3.js



Genres: null Action Adventure Black Comedy Comedy Concert/Performance Documentary Drama Horror Musical Romantic Comedy Thriller/Suspense Western

▼ Code

```
width=600
height=300
epsilon = 50

xScale = d3.scaleLinear([0,100], [0,width]);
yScale = d3.scaleLinear([0,10], [height,0]);
genreScaled3 = d3.scaleOrdinal(d3.schemeTableau10)
  .domain(genres);

legend = d3.select("#d3svg")
  .append("p")
  .attr("id", "legend")
  .append("b")
  .text("Genres: ");
legend.selectAll("span")
  .data(genres)
  .join("span")
    .style("color", m => genreScaled3(m["Major Genre"]))
    .text(m => `${m["Major Genre"]} `);
```

▼ Code

```
stats = d3.select("#d3svg")
  .append("p")
  .attr("id", "stats");

svg = d3.select("#d3svg").append('svg')
  .attr("width", width+epsilon)
  .attr("height", height+epsilon);

container = svg.append("g")
  .attr("class", "container")
  .attr("id", "pointscontainer")
  .attr("transform", `translate(${epsilon/2}, ${epsilon/2})`);

points = container.selectAll("circle.point")
  .data(movies)
  .join("circle")
    .attr("class", "point")
    .attr("cx", m => xScale(m["Rotten Tomatoes Rating"]))
    .attr("cy", m => yScale(m["IMDB Rating"]))
    .attr("r", 3)
    .style("fill", m => genreScaled3(m["Major Genre"]))
    .style("opacity", 0.8);

xAxis = d3.axisBottom(xScale);
xAxisPlacement = container.append("g")
```