Luc Guillemot Follow
Data visualization & Social Sciences
Oct 15 · 9 min read

How Does This Data Sound?

Make your data (visualization) accessible, sound up.

Using the two dimensions of a sheet of paper to convey meaning from an opaque dataset is about making data accessible. And it involves visual components: making sure color schemes are colorblind-friendly, using enough contrast between text and their background, labeling data directly instead of using complex keys, etc. But when the visual language is not accessible to a user, data visualization alone is not enough. Is it possible to make data not only visual but also talkative? Here are two propositions, one using the Web standard ARIA, and one using a sound system instead of a visual system.

Affordance and Sequentiality

Most users of assistive technologies navigate web content and interactive elements using the keyboard (people with a motor impairment have difficulty with the precise movements required for using a mouse; blind users rely on assistive technologies such as screen readers).

Affordance and sequentiality are two things impacted by this specific way of interacting with a webpage. As the design of a bench hints potential users into thinking they could sit on it, the design of elements in a webpage nudges users towards what the architect of the webpage wants them to do. Hover, focus or active states of a button make it clear that clicking on it will trigger some form of interaction. We need to reproduce that level of affordance for users of screen readers.

Furthermore, when users navigate a webpage with the keyboard, access to DOM elements is sequential: users must tab through all the interactive elements one by one before reaching an element of interest. Such a linear access also changes the way data visualizations are reached: a sighted user will first glance at the visualization in order to “scan” the main pieces of information (axis, legends, the global shape of the data, correlations, groups, all the structural visual content that

sighted users can analyze through cognition). Non-sighted users will have to go through every elements of the data visualization to wield the same overview and keep everything in memory.

Web Accessibility

W3C standards have paved the way towards a more accessible Web, especially for persons with visual impairments. Web standards such as WAI-ARIA allow users of screen readers to have an experience equivalent to persons who can see fully.

Most screen readers have three interaction modes that can be used on a web page:

1) The “Browse” mode reads every item in the DOM order, using arrow keys or a dedicated combination of keys.

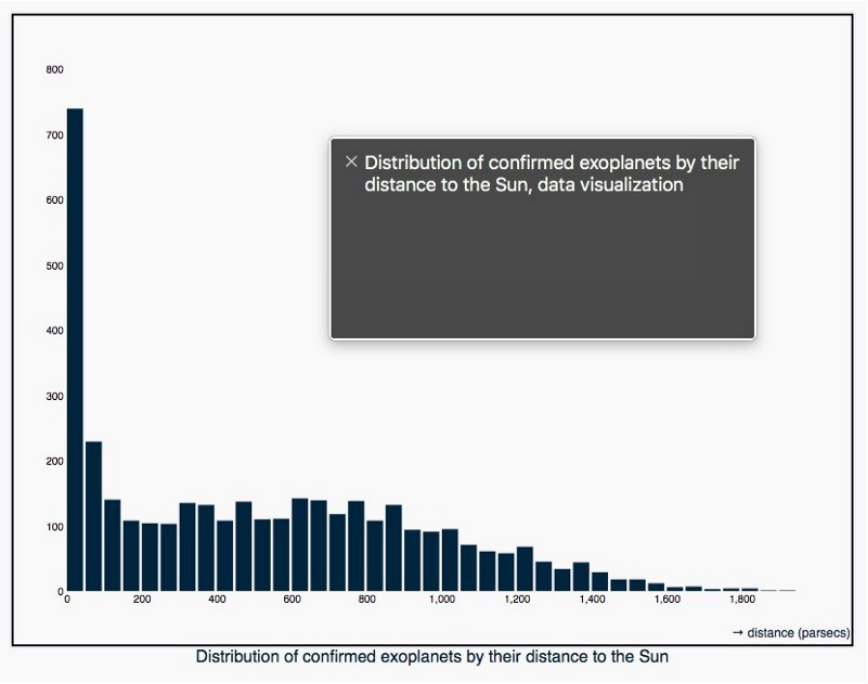
2) The “Form” mode is triggered on interactive elements such as forms or menus and allow the user to interact with web content using a standard set of keys. This mode is applied on HTML semantic markup elements like select, or when an element has the ARIA role “widget”. The user can expect to navigate a menu with arrow keys to activate a button with the space bar, etc.

3) The “application” mode is entered when an element has the ARIA role “application”. This mode is used for interactive components that don’t follow a standard pattern of interaction.

Data visualizations fall into the latter category of interactive elements that don’t follow a standard or constant interaction pattern. In these cases, DOM focus and keyboard navigation have to be specifically defined.

We will create an ARIA-compatible histogram of confirmed exoplanets by their distance to the sun, using the dataset provided by the Planetary Habitability Laboratory. The histogram will have tooltips to give detailed information on each data point. Find here the interactive implementation of this chart, and the full code using React and d3.

1. Make the data visualization accessible by keyboard navigation



When in focus (black outline), VoiceOver announces the title of the data visualization and a custom description of the type of content.

The first step consists in making the data visualization accessible by keyboard navigation and making it clear that the element is interactive, though not standard.

```
<svg
  // Make the element focusable through tabbing, and thus
actionable
  tabindex="0"

  // Make the component interactive
  role="application"

  // Give a more specific role to announce
  aria-roledescription="data visualization"

  // Create a relationship to the title
  aria-labelledby="histogram-title"

>
  ...bars

  <g id="axis" aria-hidden="true" />

</svg>

<figcaption
```

```
  id="histogram-title"
>
  Distribution of confirmed exoplanets by their distance to
the Sun
</figcaption>
```

`tabindex="0"` This attribute enables focus on the svg element through keyboard navigation and inserts it in the DOM tab sequence (by default, only links and form elements can receive focus through tabbing).

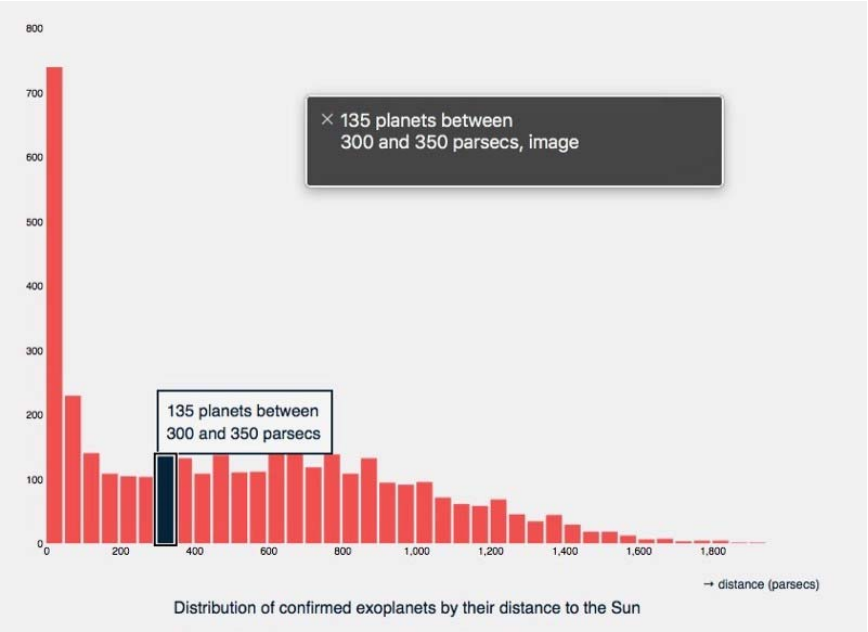
`role="application"` This attribute indicates to screen readers that focus is on an interactive element whose interaction pattern is not standard. This role definition will make some screen readers switch from “browse” mode to “application” mode, hence pass defined keystrokes (such as arrows) on to the browser and web application instead of using them for reading. The same behavior is applied when switching to “Form” mode, except the interaction pattern does not necessarily follow standards like pressing “space” to reproduce a button click. Furthermore, macOS’s built-in screen reader, VoiceOver, doesn’t have single-key navigation (but always uses custom key combinations to navigate a page), it only has the “Browse” mode. For VoiceOver users, defining the application role will be a hint, but won’t change the behavior of VoiceOver dedicated keystrokes.

`aria-roledescription="data visualization"` This attribute is a mechanism to override the default role name announced by screen readers. It gives the opportunity to use a more meaningful role description.

`aria-labelledby="histogram-title"` This attribute is set to the id of the element labeling the application. It is more commonly used for form elements having a `label` but can be used as a title. It creates a relationship between the SVG element and the data visualization title, which is visible to sighted users, enabling screen readers to announce the title when the SVG comes into focus.

`aria-hidden="true"` This attribute on the SVG group containing the axis makes screen readers ignore this element that is a purely graphic way of conveying information. It also prevents a too long tab sequence.

2. Implement keyboard interaction



A user can move focus from data point to data point using arrow keys. When a user navigates to a data point, VoiceOver announces its values. The values are also visible in a tooltip.

As a data visualization doesn't follow a standard interaction pattern, we need to implement it. The goal is to make every data point reachable and make screen readers announce information about them. Let's first give the SVG additional attributes to ensure a functional keyboard interaction:

```
<svg
  aria-activedescendant="histogram-bar-1"
  onkeydown="moveFocusToDataPoint()"
>
```

`aria-activedescendant="id"` . This attribute hints screen readers towards which child of the application's main component is currently in focus.

`onkeydown="moveFocusToDataPoint()"` . For screen readers to actually read the label of the currently focused child element, it needs to actually receive focus, which must be done programmatically by a callback function, like:

```
function moveFocusToDataPoint() {
  document.getElementById("histogram-bar-1").focus();
}
```

Second, let's add interactivity to the bars of our histogram, which are SVG `rect` elements:

```
<rect
  id="histogram-bar-1"

  // Allow element to receive focus
  tabindex="-1"

  // id of the tooltip for this data point.
  aria-labelledby="histogram-tooltip-1"

  onkeydown="moveFocusToNextDataPoint()"
  // ...Other attributes
/>

<div
  // Make sure the tooltip is read by aria-labelledby on
rect.
  role="tooltip"

  id="histogram-tooltip-1"
>
  176 planets between 20 and 40 parsecs
</div>
```

`id="histogram-bar-1"` Each bar receives a unique id that is used to set focus.

`tabindex="-1"` A negative value for tabindex allows the element to receive focus programmatically, but doesn't insert it in the tab sequence. It prevents keyboard users from having to tab through too many elements when browsing a webpage. Another solution to prevent a tab sequence that is too long is to add an invisible skip link.

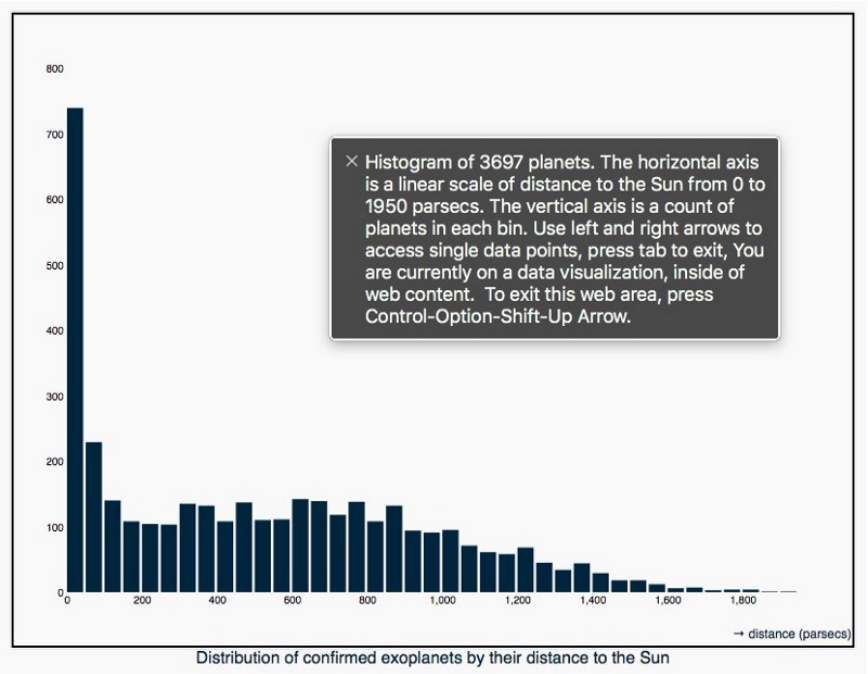
`aria-labelledby="histogram-tooltip-1"` A relationship is created between the aria-labelledby and id attributes sharing the same value. When in focus, screen readers will announce the content of the element linked by this id. Here, we use the id of a tooltip that

is visible to sighted users.

`role="tooltip"` This explicit role is not necessary as the attribute aria-labelledby on the `rect` element should be enough, it may nevertheless add additional support for some screen reader softwares, as mentioned by Heydon Pickering, to ensure that the content of the tooltip is properly announced.

`onkeydown="moveFocusToNextDataPoint()"` Similar to the "key down" event used on the application main element, each bar of the histogram needs to listen to keyboard events in order to move focus to the next-to-be-focused bar.

3. Describe the interaction pattern



Distribution of confirmed exoplanets by their distance to the Sun

VoiceOver announces the content provided by the attribute aria-describedby. Along with a description of the content, it announces how to interact with the data visualization.

Now that the data visualization is keyboard accessible and actionable, the next step consists in giving an overview of the data visualization (what a sighted user gets from glancing at the visualization) and giving indications on how to interact with it. This is what the attribute aria-describedby is for.

```
<svg aria-describedby="histogram-description" />
```

```
<div id="histogram-description" class="histogram-
description">

Histogram of 3697 planets. The horizontal axis is a linear
scale of distance to the Sun from 0 to 1950 parsecs. The
vertical axis is a count of planets in each bin. Use left
and right arrows to access single data points, press tab to
exit"

</div>
```

4. Give visual feedback

```
.histogram-description {
  display: none;
}

svg:focus > rect {
  fill: black
}

rect:focus,
rect:hover {
  fill: black;
}
```

The element containing the description is not needed for sighted users, we can hide it with CSS. On the contrary, the focused elements need to be styled properly to give visual feedback to users not using a screen reader.

5. Reproduce keyboard navigation for mouse navigation.

Don’t forget to add support for mouse users by doubling keyboard events with mouse events!

6. Add a data table

As mentioned by Sergei Kriger, sometimes, the best way of making a chart accessible is to combine the visualization with a (visually hidden) data table, be it with the raw data or with the processed data used for the data visualization. Heydon Pickering offers a very good guide on how to create accessible data tables.

Sonification

A data visualization is often created with an intention, and uses the user's perception to tell a story. And the user still has room for their own interpretation. There is a two-way relationship between the designer of the image and its reader, with the data visualization serving as a transitional object. Can we reproduce this complex relationship when words are much more explicit than images? In addition to the description with words that ARIA allows, would it be possible to give a perceptual experience, to offer a sensorial access to data without using vision?

Sonification is the use of non-speech audio to convey information or perceptualize data. Common examples of sonification are Geiger counters, morse code or sonars. (Audification is the direct transformation of an actual signal into audible sound, the detection of gravitational waves can, for instance, be converted into sound!).

Here is the same histogram implementing a sound feedback.

A sound scale

One of the most important decisions to make when building a data visualization is which scale to use to match the data domain to the visual field of a canvas. In the same way geometric scales match data to visible values, data can be scaled to a sound gamut. For instance, an equal-temperament chromatic scale from A3 to A7 gives 60 semitone intervals that can be matched to a data domain.

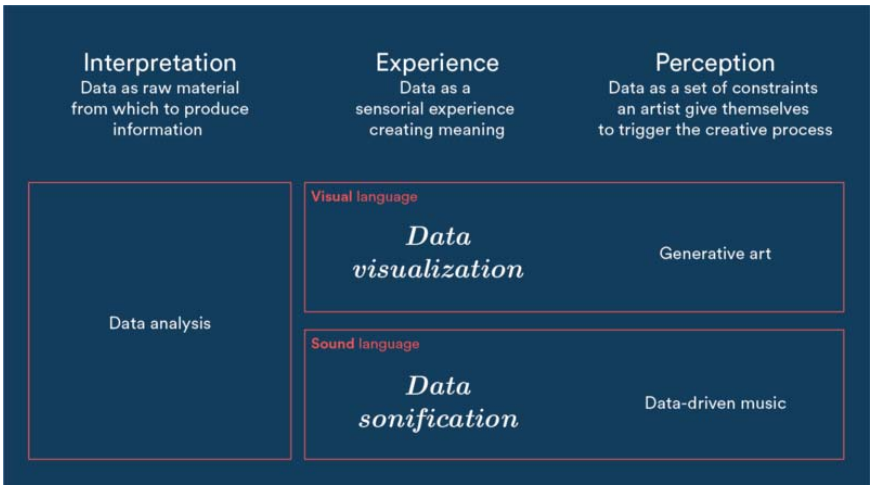
Auditory feedback

Screen readers do have a sound feedback when moving focus from element to element, or when no action is possible. In the same way, we can implement an auditory feedback whose height in the tone scale matches the data value.

Data pattern as a melody

Given the sound scale, the data shape creates a melody that can be listened to in the same way that a data visualization can be glanced at.

A sound language?



When working with data, inclusivity is not necessarily about making charts accessible, but about making data itself accessible, which means that the step of going through a visualization first—hard to pass for a visual designer!—is maybe superfluous. There is a research field in defining a sound language, an intersection between data science and music that artists such as Brian Foo have already followed by using data as creative material to compose music and that the experiments described here attempt to address from the perspective of data visualization. In the same way Jacques Bertin defined visual variables, could it be possible to define a sound system where different dimensions of sound (pitch, intensity, duration, timbre) can be used to describe different dimensions of a dataset?

. . .

References:

- Interactive Demo
- Source Code
- Exoplanets Data Set

. . .

Thanks to everyone at Interactive Things for their feedback on this experiment!