Precalculate "Visual" Variables

Even though the final visual was made with D3.js through JavaScript, it doesn't mean that I had to calculate all the "visual aspects" at the same time (in JavaScript). When dealing with a fixed dataset, which was the case for this Olympics piece, I prefer to precalculate the more complex aspects of the visual because I personally find it easier to perform these calculations in R. But the same applies to any combination of tools. The tool/language that you end up creating your visualization with isn't necessarily the best or easiest tool to figure out aspects such as placements, rotations, sizes, shapes, etc. that are based on the data.

I like to call these types of additions to my data "visual variables." Visual variables have nothing to do with the original dataset, but only apply to how the data will be laid out on the screen. For example, for this project I precalculated the following "visual variables":

- How far each of the five circles as a whole should be rotated to have the opening pointing center down
- How many degrees each of the inner sport feathers had to be rotated, based on the feathers that came before it
- And how far each medal had to be offset from the center of a feather

The only placement variable that I kept calculating "on the fly" in JavaScript was the edition/year scale that decided how far from the center a medal should be drawn. This made it simpler to increase or decrease the circles based on screen width.

These visual variables can either be attached to your original dataset, such as when you've predefined an x and y location per data point, or loaded as a completely separate file and referenced at the right moments. A personal extra benefit in doing this is that it makes my JavaScript a lot more concise. (⌐■_■)

What I love about this type of extremely structured setup, is that once it works for one (nested) object, it works for all of them. Either all ±5,000 medals are still wrongly placed on the screen, or they are all correct after your next code change. You can certainly design with dummy data in a tool such as Sketch or Adobe Illustrator, but it's just impossible to capture the nuance and complexity of seeing your idea applied to *real* data. In Figure 2.8, I finally had all of the medals in place and sporting disciplines labelled.

With the medals in place, I started working on the feather shape. I had not expected that some disciplines would be so exceptionally wide (athletics took up more than half a circle). So I pulled up my now go-to "SVG path playground" website[4] to wrap my head around the math that I needed to mimic the shape of a feather's tip. Luckily, my work with the *Lord of the Rings* visualization from the previous project meant that custom SVG paths were still fresh in my mind and it surprised me how fast I got the feather tip shape to appear (see Figure 2.9).

To make the wider feathers look better, I created multiple tips if a feather was too wide (a simple idea, but not so easy to actually create). A fair amount of iterations later, and having increased my knowledge of custom SVG paths even further, the circles looked a lot better (Figure 2.10).

Fig.2.8

All the medals are finally in their correct location.

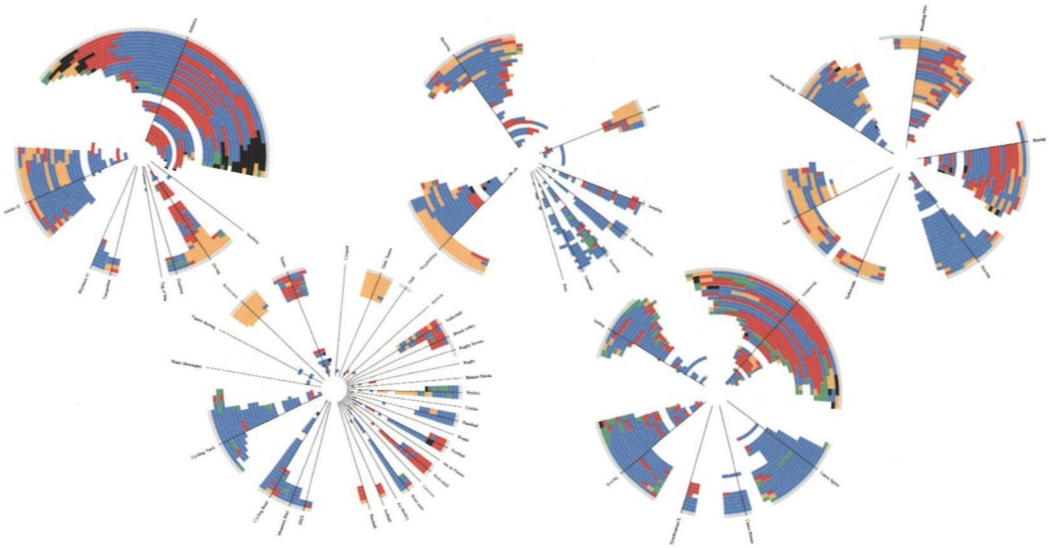


Fig.2.9

Clipping the circles so each sport's feather would end in exactly one "tip."

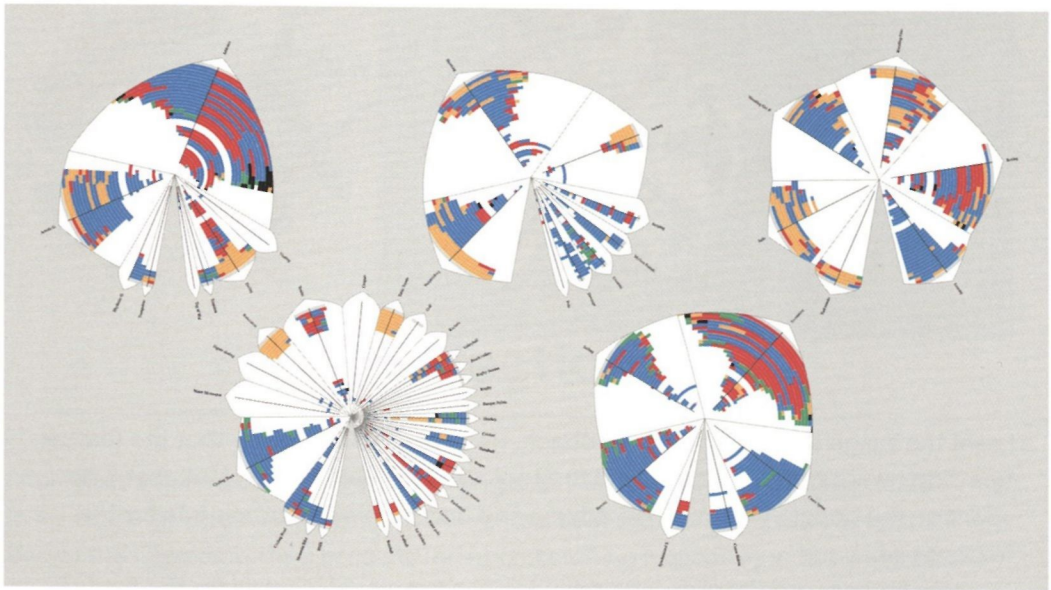
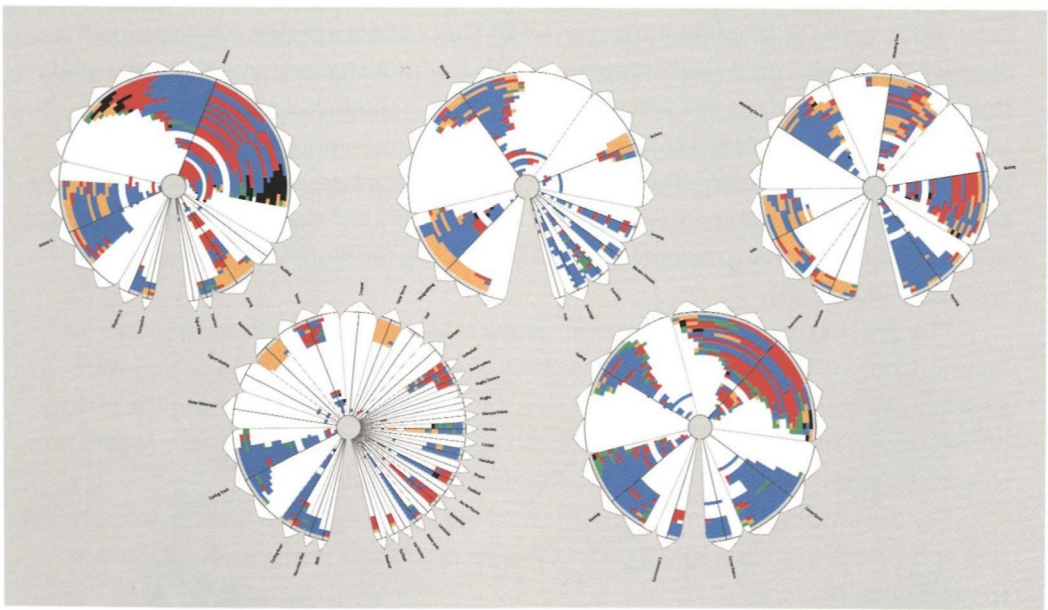


Fig.2.10

If a sporting discipline was too wide, it got several feather tips.



[4] SVG Cubic Bézier Curve Example: http://blogs.sitepointstatic.com/examples/tech/svg-curves/cubic-curve.html

Olympic Feathers

↳ OlympicFeathers.VisualCinnamon.com

ALL OLYMPIC GOLD MEDAL WINNERS

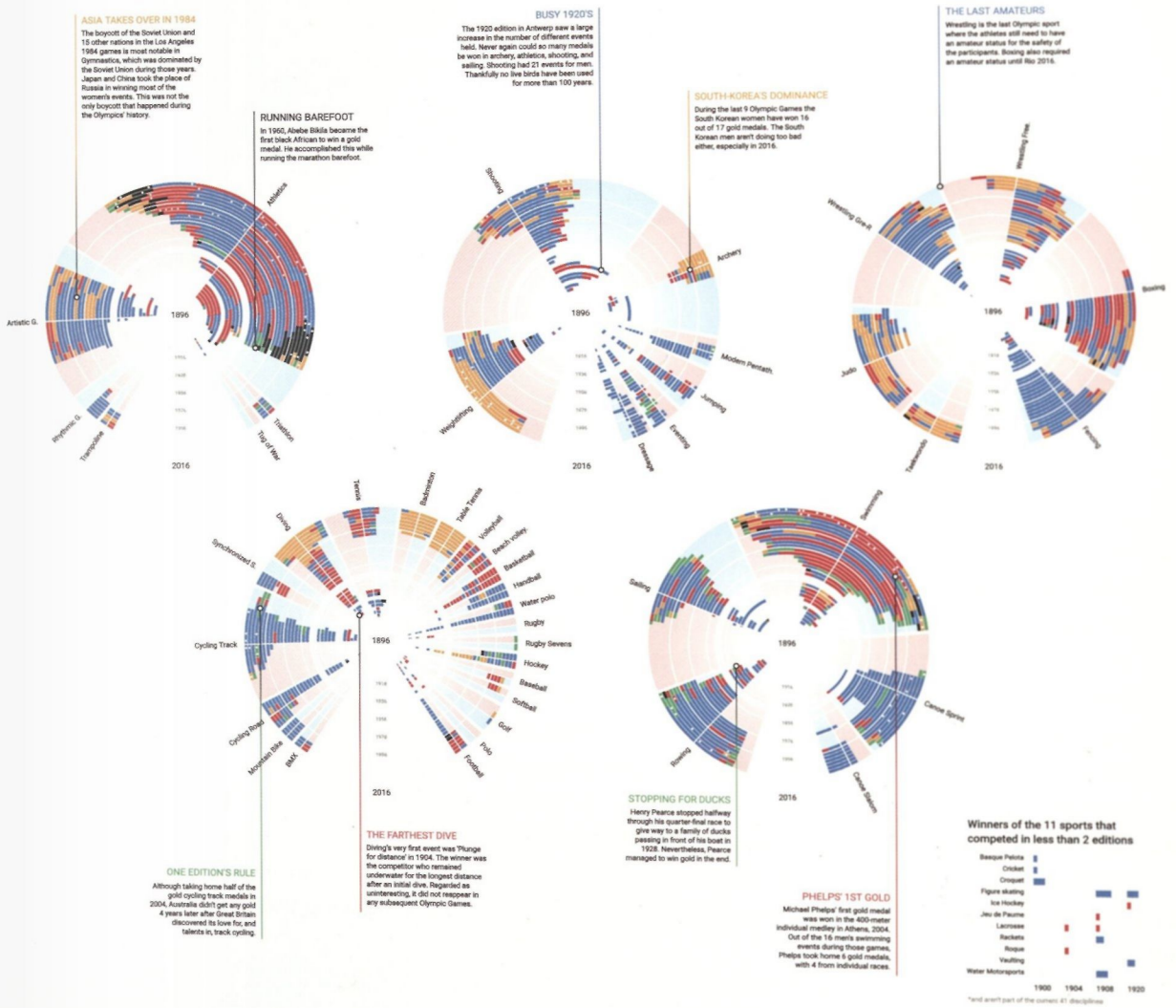
summer editions since 1896

More than 5000 Olympic events have had a winner, rewarded with a gold medal from 1904 onwards, in the Summer Olympics since the first games of 1896. Investigate the visuals below to see how each of these medals has been won in the 56 different sporting disciplines that have competed at the games, of which 41 are still held at Rio 2016.

Most of the Olympic sports started out being a men only event. Thankfully this started to change during the 2nd half of the last century. Even the number of medals that can be won for one discipline is slowly becoming the same for both genders. Today at Rio there are 3 disciplines left in which only one gender can compete; the Greco-Roman wrestling, already at the games since the very first edition, is done solely by men. Rhythmic gymnastics & synchronized swimming on the other hand, both at the Olympics since 1984, are only performed by women.

Although Rio could have been celebrating the 31st Olympic Games, 3 editions have been canceled, due to WW I in 1916 and WW II in 1940 & 1944. And yes, Tug of war has truly been part of 5 Olympic Games, from 1900 to 1920. Hover over the medals to see the winning athlete or team or hover over the time-line in the bottom of each circle to find your own interesting stories.

Instead of a medal being represented by a specific width, in these visuals 1 medal always has the same arc length. This makes sure that the more recent the edition of the games, the more emphasis it gets due to the increasing size of the ring.



HOW TO READ A FEATHER

Each circle represents a grouping of several different (but approximately) similar themed sports, such as water or ball sports. Within a circle we find slices. Let's call each slice a feather to make it easier to distinguish as a whole. Each feather represents one discipline.

A feather is split up into 31 sections, radiating outward. Starting from the first Olympic Games in 1896 at the center to the current Olympic Games in Rio in 2016 at the other end. Each discipline is twice as wide as the maximum number of medals that could ever be won during one edition for a gender (men and women get the same width).

The next split is by gender. For the example feather to the right, the small bars going upward on the light red background are gold medals won by women. The bars going towards the bottom, with the light blue background are gold medals won by men.

All the medals have the same arc length and you can see in the bottom (men) section of the example feather to the right how wide 1 medal is for each edition of the Olympics. For medals won by a men & woman team or two gold medals in the same event each person gets 0.5 medal assigned. The medal bars are colored according to the continent in which the country of the winning athlete or team lies. Furthermore, for each edition and gender, the bars are stacked from the continent that won the most medals to the least.

Finally, some sport disciplines have Olympic records, such as athletics and swimming. As an extra level of detail, the events in which the gold medalist reached a currently standing Olympic record (after Rio 2016) are marked with a white dot. You can see the record when you hover over the medal.

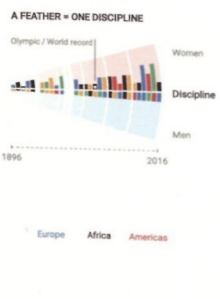


Fig.2

ASIA TAKES OVER IN 1984

The boycott of the Soviet Union and 15 other nations in the Los Angeles 1984 games is most notable in Gymnastics, which was dominated by the Soviet Union during those years. Japan and China took the place of Russia in winning most of the women's events. This was not the only boycott that happened during the Olympics' history.

RUNNING BAREFOOT

In 1960, Abebe Bikila became the first black African to win a gold medal. He accomplished this while running the marathon barefoot.

BUSY 1920'S

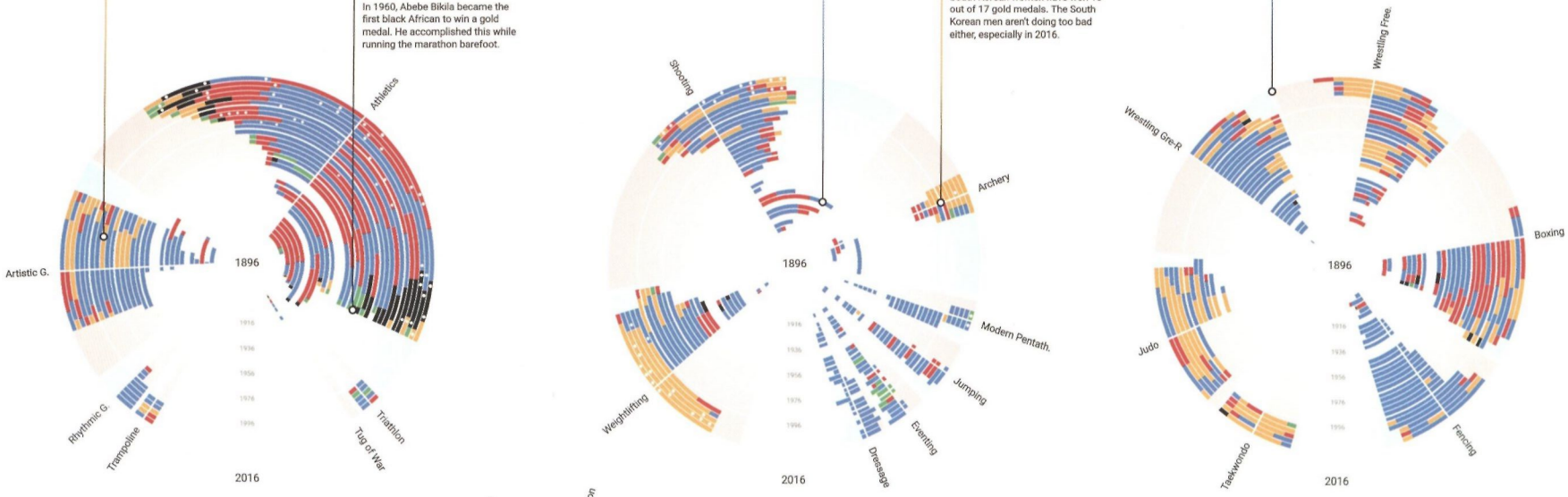
The 1920 edition in Antwerp saw a large increase in the number of different events held. Never again could so many medals be won in archery, athletics, shooting, and sailing. Shooting had 21 events for men. Thankfully no live birds have been used for more than 100 years.

SOUTH-KOREA'S DOMINANCE

During the last 9 Olympic Games the South Korean women have won 16 out of 17 gold medals. The South Korean men aren't doing too bad either, especially in 2016.

THE LAST AMATEURS

Wrestling is the last Olympic sport where the athletes still need to have an amateur status for the safety of the participants. Boxing also required an amateur status until Rio 2016.

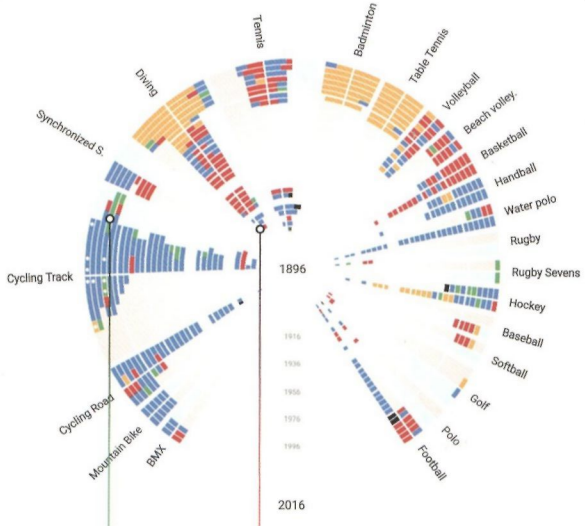


ONE EDITION'S RULE

Although taking home half of the gold cycling track medals in 2004, Australia didn't get any gold 4 years later after Great Britain discovered its love for, and talents in, track cycling.

THE FARTHEST DIVE

Diving's very first event was 'Plunge for distance' in 1904. The winner was the competitor who remained underwater for the longest distance after an initial dive. Regarded as uninteresting, it did not reappear in any subsequent Olympic Games.

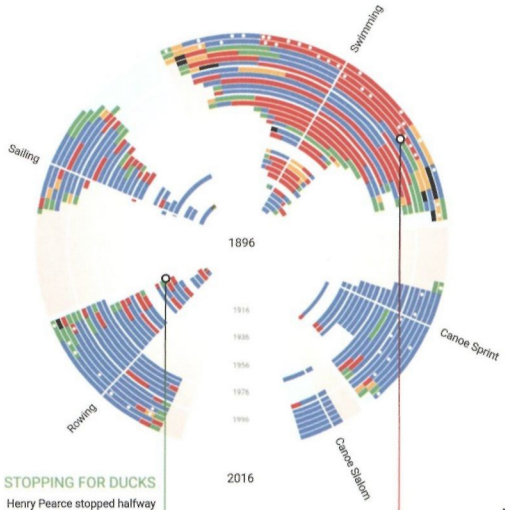


STOPPING FOR DUCKS

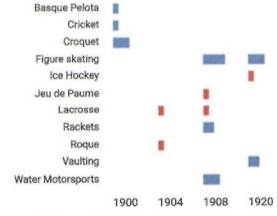
Henry Pearce stopped halfway through his quarter-final race to give way to a family of ducks passing in front of his boat in 1928. Nevertheless, Pearce managed to win gold in the end.

PHELPS' 1ST GOLD

Michael Phelps' first gold medal was won in the 400-meter individual medley in Athens, 2004. Out of the 16 men's swimming events during those games, Phelps took home 6 gold medals, with 4 from individual races.



Winners of the 11 sports that competed in less than 2 editions



*and aren't part of the current 41 disciplines

Fig. 2.14 b
A zoom in on the five circles.

I wanted to do something silly and light-hearted to commemorate the Obama's eight years in office. And the first thing I could think of was all of their appearances on late-night talk shows I watched on YouTube (that "Slow Jam the News with President Obama" was so good). I started digging around to see if there was a list of talk show appearances for both Barack and Michelle—and it was IMDb to the rescue again! Both of them had their own IMDb pages, so I cross-referenced Barack Obama's 214 credits and Michelle Obama's 123 with a Wikipedia article of late-night American network TV programs. I was able to narrow down the credits to 24 late-night appearances for Barack and 22 for Michelle. I then compiled a list with the date of their appearance, name of the late-night talk show, and the show's YouTube channel if available.

Using that information, I was able to use the YouTube Search API to search for keyword "obama" and filter the results down by `channelId` and a publish date within five days of the interview date. Unfortunately, because I wasn't sure how many videos were published for each interview (if at all), I set the `maxResults` to 15. This meant that I would get back videos within those days that had nothing to do with the Obamas, and I had to manually go through all of the videos to weed out the irrelevant ones (there were 244 videos, and 186 were ultimately filtered out, leaving only 58).

The list was unfortunately incomplete, as there were interviews with past shows and hosts like the *Late Show with David Letterman* and the *Tonight Show with Jay Leno* that weren't on YouTube. I chose to keep them out instead of trying to find the videos on other websites or on unofficial channels, in the hopes that this would make for cleaner and more consistent data.

Around that time, I attended the annual D3.js unconference. I told some friends there about the dataset I had gathered and how I wasn't sure what to do with it. They were enthusiastic: "You should get the captions for each video and do something with the words!"; "Wouldn't it be cool if you could run facial detection on the video and correlate their emotions with what they're saying?" And even though I was quite worried about how long it would take me, the idea took root in my mind.

After some research, I realized there wasn't a financially affordable way for me to pass whole videos into any facial recognition software, but thankfully I was still surrounded by some very resourceful friends at the conference. They suggested an alternative: take screenshots of the video and upload the images to Google Cloud Vision API for analysis. Here's what I did (and the Node.js packages I used):

1. I took the list of all videos, and downloaded them (and their captions, if available) with `youtube-dl`.
2. I used `vtt-to-json` to convert the captions into JSON and got the timestamp every time someone talked, and used that timestamp to screenshot the video with `fluent-ffmpeg`.
3. For each screenshot, I uploaded it to Google Cloud Vision API, which gave back data of any faces it found in the image, the *x*/*y* bounds of the face, and any emotions detected on the face.
4. I saved the videos' captions into one JSON file.
5. I joined the captions with the annotated image data (including emotions) from the Google Cloud Vision API.

This entire process took a few days, and once I finally had all the data cleaned I started thinking about the design. I learned my lesson from "Travel," and made sure to explore the basic shape of the data, as well as all the different types of data I had on hand before starting to sketch.

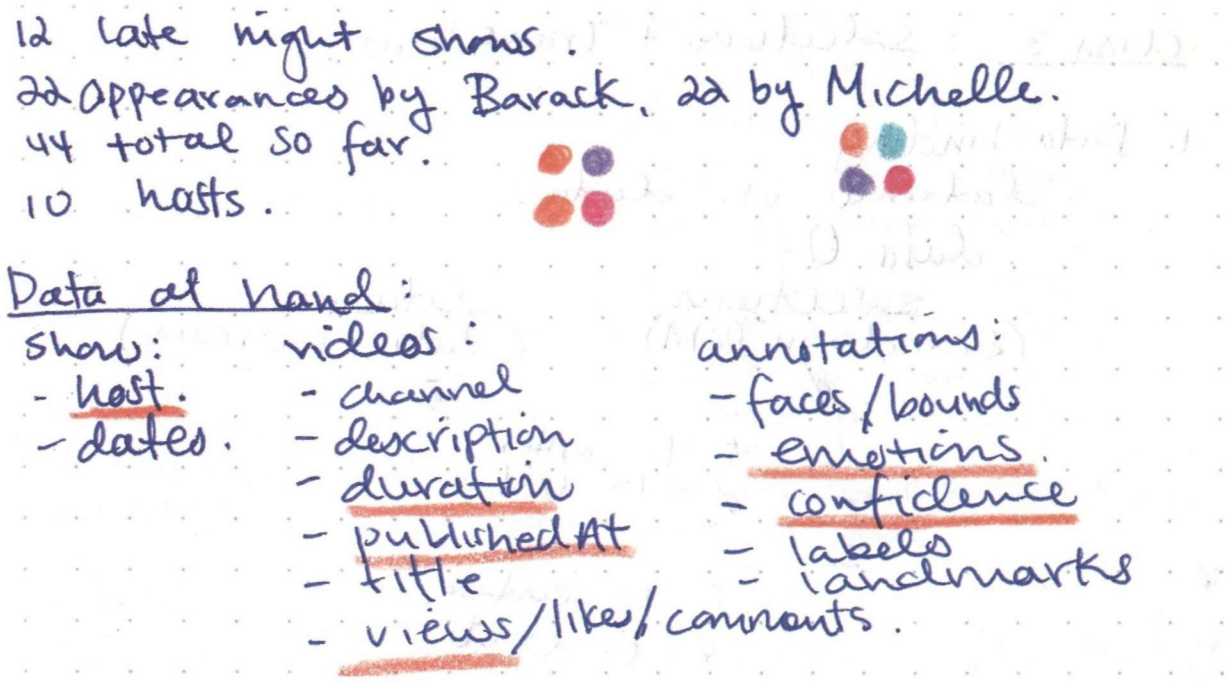


Fig.4.1
List of the data I had.

Data Can Be Found in Many Different Ways

When I start on a data visualization project, I always start with a curiosity, a topic I'm interested in—whether that's the Obamas' late-night talk show appearances, recurring themes in *Hamilton* the musical, *Dance Dance Revolution*, or summer blockbusters. I find that having that curiosity keeps me motivated and gives me direction when looking for datasets.

Once I have a curiosity, I start with a Google search. Sometimes I'm lucky, and I find a dataset that's already downloadable in JSON or CSV format (*very* rare). Oftentimes, I'll find a website where the data is in plain text on the page; in those instances, I can get the data with their API or by scraping the site. For scraping, I like using Node.js modules (mostly `http` to programmatically request the webpage), and for APIs, I've found that there's usually a corresponding Node.js package that does all the hard parts (like OAuth authentication for signing into a service) for me.

If I can't piece the data together from resources online, then I have to manually gather and enter the dataset. It's tedious work, but I've learned (the hard way) that Excel makes the process so much easier. One time, I found the data I wanted in image form, and was able to get the underlying raw data by just ... asking the person who ran the website. It was amazing.

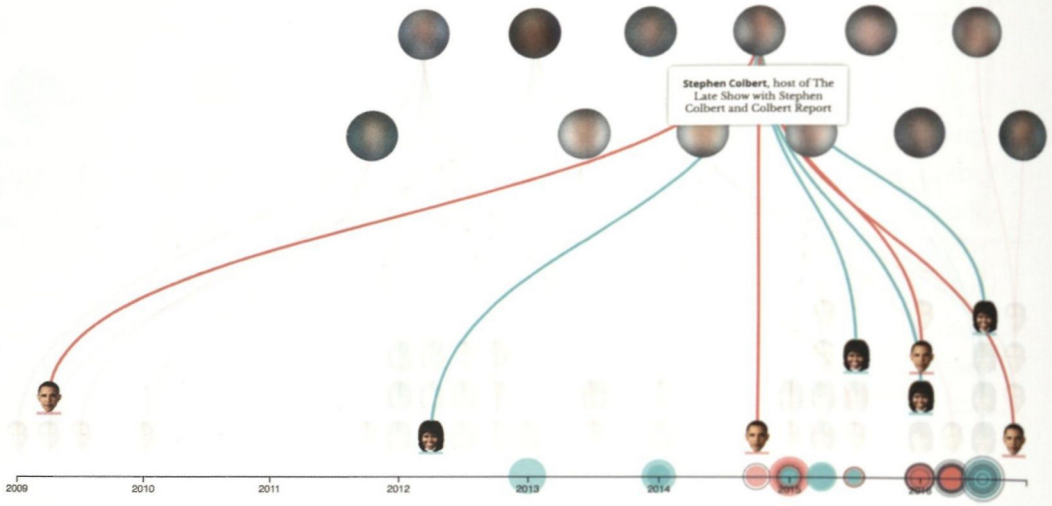
For a short explanation of JSON and CSV, see "Technologies & Tools" at the beginning of the book.

Whenever I think, "Somebody must have done this already" while coding, somebody usually has!

You can read about this story in my "Music" chapter. (*≧▽≦)

Fig.4.7

For section two, hovering a host shows corresponding guest appearances. It was a very simple implementation, but it made me really happy because it helped a lot in making the tangle of links easier to navigate.

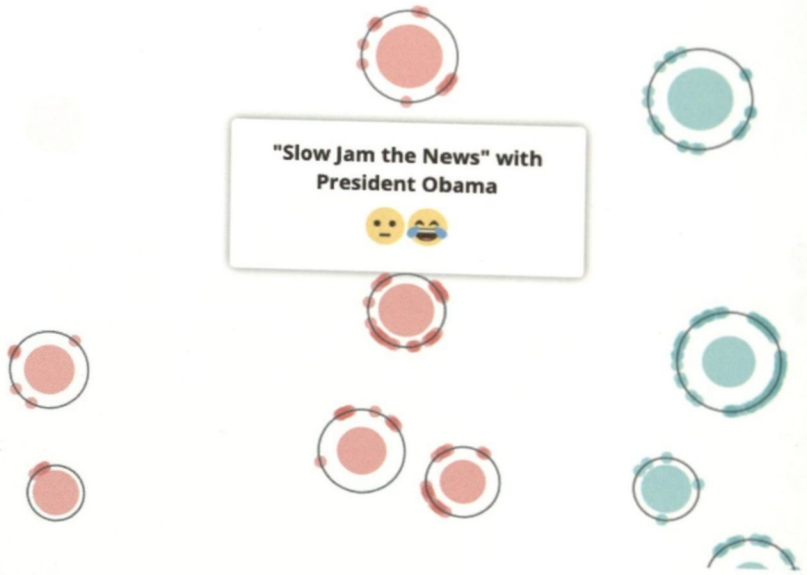


In the third section, I introduced the videos I downloaded from YouTube, with the *x*-axis using the same timeline as the previous section (I love this little detail) and the *y*-axis being the number of views. In the fourth section, I filtered down to just the videos with captions; the filled circle radius is the number of views, and the outer ring radius is the duration of the video. It's fun to see that some of the videos got a lot of views despite being shorter in duration, or vice versa. The small dots on the ring represented the times when Google Cloud Vision API said there were expressions of joy, so we can see whether the laughter was present throughout the video, or if it was concentrated in specific blocks (Figure 4.8). My favorite part about this was actually in the description, where I calculated the number of times the POTUS and FLOTUS laughed and found out that the FLOTUS had significantly more laughter!

I like the inten
the dots and
ring, but it's d
some of my m
experimental
I'm not sure if
same thing no
it is quite over
and confusing

Fig.4.8

In section four, hovering on a caption within a video shows the emotions that were detected for that caption.



For the final section (that I spent the whole project building up to), my goal was to create an interface that would encourage exploration of the emoji-filled photos. It centers around a timeline of the captions and emotions detected in the selected video, and hovering the timeline shows the full screenshot for that scene with emojis placed on any detected faces.

How Scrollytelling Works

I use scrollytelling when I want to explain different parts of the data or break down interesting visual insights step-by-step to a more general audience. It's great because I can control exactly how the reader interacts with the visualization, and all the reader has to do is scroll.

Scrollytelling is when a user scrolls past a series of "containers" on the page—usually `divs`—and each container triggers a change in the visualization. At the bare minimum, there are four things that scrollytelling has to do:

1. Add event listener for scroll. I also usually use `lodash.js`'s `debounce` to make sure that the callback function isn't called more than once every 64–200ms, depending on how computationally heavy the visualization update is.

2. Within the scroll callback function, calculate which container we're in given the scroll position and each container's top and bottom *y*-positions.

3. Trigger corresponding changes to the visualization. If there are animations and those animations are tied to scroll (the animation progress is directly mapped to the scroll progress, rather than automatically started upon entering the container), calculate the difference between the scroll position and the top of that container.

4. Divide the difference by the height of the container, and pass that decimal value to the interpolator.

I implemented my first scrollytelling project from scratch because I wanted to understand how it worked, but I've since relied on JavaScript libraries to help me manage the scrolling. My current favorite is `Scrollama.js` from The Pudding, because of its super straightforward API and focus on performance; I highly recommend it. (And let's face it, scrollytelling isn't something you ever want to implement twice!)

A callback function is a common type of function used in web development that is run ("called back") when some other action—in this case, scroll—has occurred.

For how animations and interpolators work, see the lesson "Custom Animations" on page 269 of my "Nature" chapter.

`Scrollama.js` aims to be more performant by offloading the computations in step 2 (which can get expensive if done on every scroll) to the Intersection Observer API, which can detect when an element enters the viewport—I'm such a fan!

As soon as I implemented the timeline, I knew I needed to implement the *fisheye* effect—a way for users to hover and expand specific slices to better preview them. I had seen the *New York Times* implementation of the horizontal fisheye for navigating through fashion week images[4], and I knew Nadieh had adjusted that code for canvas before. It took a bit of digging to find an SVG equivalent for cropping centered images (use `preserveAspectRatio` and set the `viewBox` width the same as the image width), and a few passes through Nadieh's and the original fisheye code to get the subset I needed (mapping an *x*-coordinate to its new position based on the distortion) to get it working in my own code.

I found my
cropping s
on Sara So
blog[5]; she h
explainers.

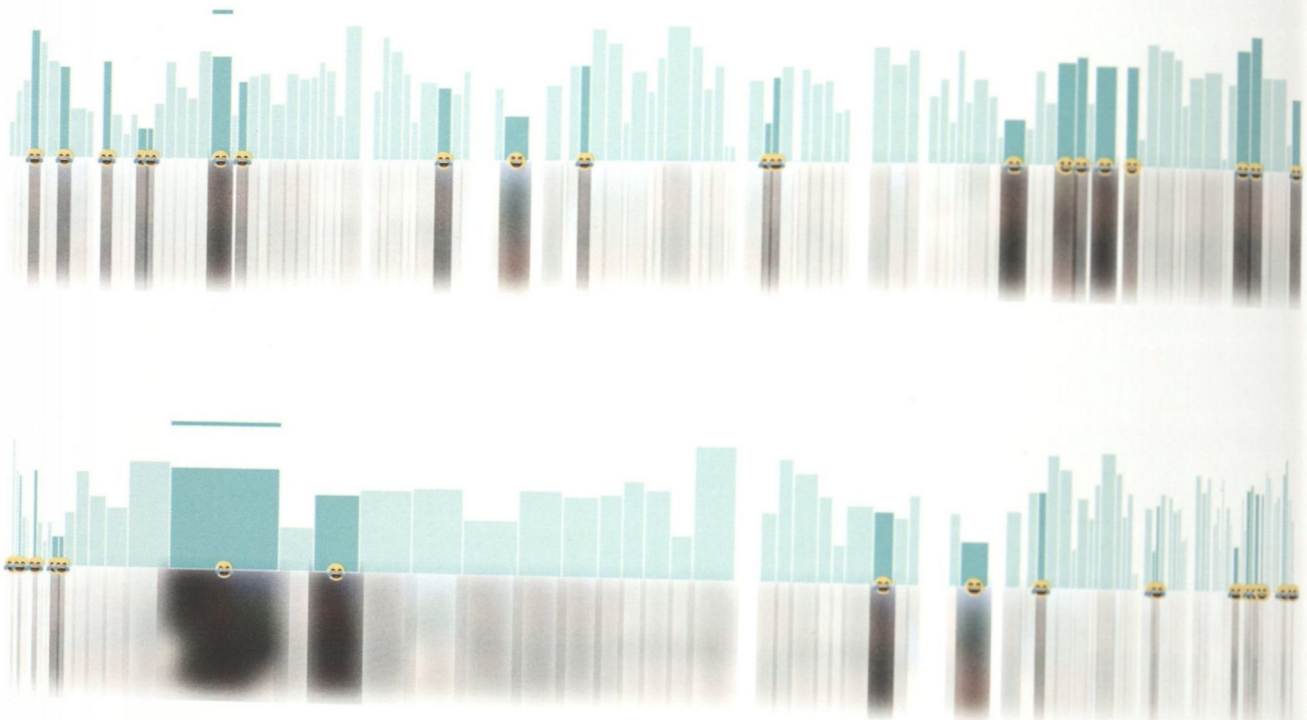


Fig.4.9

The timeline as-is (top) and the timeline with the fisheye effect on hover (bottom).

One of the final things I had to do was make sure that all the visualizations worked in mobile. It was the first time I really tried to be mobile friendly, and after some research, I ended up using `ismobilejs`, a JavaScript package that detects what device the user is on (and I've used this package in all my projects ever since). If the package detected the user was coming from mobile, I passed in a narrower width for all my visualizations, resulting in very squished visualizations (but they did fit the screen!). My happiest mobile moment, though, was getting that fisheye effect working in mobile. When Ilnitially used the browser's built-in `touchmove` event listener, scrubbing was extremely buggy and finicky. After spending an entire day thinking of a different interaction I could make with the timeline on mobile, I realized that D3.js already had a great touch implementation with the *drag* module; the end result was so smooth it was magical butter.

For examples
designing vis
specifically f
see Nadieh's
For Mobile A
Desktop Isn't
About Scalin
on page 290
"Culture" cha

[4] Front Row to Fashion Week, February 2014: http://www.nytimes.com/newsgraphics/2014/02/14/fashion-week-editors-picks/index.html
[5] https://www.sarasoueidan.com/blog/svg-coordinate-systems/

Reflections

This project took much longer than all my previous *Data Sketches* projects, but I'm so happy with myself for overcoming all the technical challenges I faced and figured out. I'm especially happy that I implemented scrollytelling from scratch because I now know exactly how it all works and use that knowledge regularly in my work. One of the best things, though, is that it gave me a sense of technical fearlessness, where I really felt that if there was something I wanted to implement, that I'd be able to figure out how to do it, given enough time and the right Google searches.

Sadly, the response to this project was less enthusiastic than I was hoping for, considering the amount of effort I put into animating the visuals and the overall amount of time I ended up spending. I think this was because my text merely recited the facts in the data—which wasn't very memorable for anyone. (This was in contrast to my "Book" project about *Hamilton* the musical, where I really focused on what I learned in my analysis and wrote an article that really made the *Hamilton* fans *feel* for the musical; that was much more successful.)

But that in turn also taught me a great lesson; how I feel about my own projects shouldn't be dictated by external validation, but rather how much fun I had building it. And I'm really proud of everything I was able to accomplish; I think it really set me up well technically for all my subsequent projects. I like my silly, fun, Obamas project very much. ｡ﾟ✶. \ (*^∀^*)ノ.✶ﾟ｡

I had to get my hands on a whole bunch of fantasy book titles and scraping the web looked like the fastest way to do this. On Amazon I found a section that showed the top 100 fantasy authors from that day. I wrote a small web scraper function in R with the `rvest` package that automatically scanned through the five pages on Amazon (20 authors per page) and saved their names. However, I couldn't find an easy way to get their most popular books and the Amazon API seemed too much of a hassle to figure out.

Luckily, Goodreads has a very nice API. I wrote another small script with help from the `rgoodreads` package to request information about the 10 most popular books per author, along with information about the number of ratings, average rating, and publication date for each of the 100 authors that I had gotten from the Amazon list.

Most big
APIs seem
much has
for that m

↳ Data Can Be Found in Many Different Ways

While I didn't come across a single public dataset with a bunch of fantasy book titles, I knew that there are other ways of finding data than simply looking for structured CSV or JSON files. Instead I figured out what *was* available: the Amazon author list to get popular fantasy authors and the Goodreads API to retrieve information about those authors. Combining those resources, I was able to create the dataset of fantasy book titles.

Next came the trickiest part; I had to do text mining on the titles, which in this case consisted of text cleaning, replacing words by more general terms, and clustering of similar titles. For the text cleaning I made a few choices. For one, I only kept the authors that had a *median* number of ratings per book that was above 20. Furthermore, I wasn't looking for any omnibus—a collection of several books—or books written by many people. For this (although not perfect) I looked at how often the exact same book title appeared in my list and took out those that appeared more than twice. Furthermore, I removed all books with terms such as "box," "set," or "edition," making sure to manually check all the deletions. Finally, I scrapped books with no ratings. This left me with 862 book titles from 97 different authors.

Now the data was ready for some text cleaning by removing digits, punctuation, and stop words (which are some of the most common words in the language, such as "a," "is," "of," and carry no meaning to interpreting the text). I did a quick word count after the title cleaning to get a sense of what words occur most often in book titles. As these are words, I couldn't resist visualizing the results as a word cloud (see Figure 5.1). The bigger the size of the word, the more often it appears in titles (the location and angle have no meaning). I was very happy to see how often the word "magic" occured!

I also remove
very specific
to this partic
dataset of bo
such as "Part

Fig.5.1

The words occurring most often in the 862 fantasy book titles. The bigger a word's size, the more often it occurs.

I wanted to look for trends in these words. However, for a standard text mining algorithm, the words "wizard" and "witch" are as different as "wizard" and "radio," even though we humans understand the relationship between these words. I first tried to automatically get hypernyms of each noun in the titles, but that sadly didn't give me good enough results, the terms weren't general enough or already overgeneralized. I therefore set about doing it manually and replaced all ±800 unique words across all titles by more general terms, such as "name," "magic," "location," and so on.

A hypernym is a word that lies higher in the hierarchy of concepts. Like "fruit," which is a hypernym of a "banana."

↳ Manually Add New Variables to Your Data

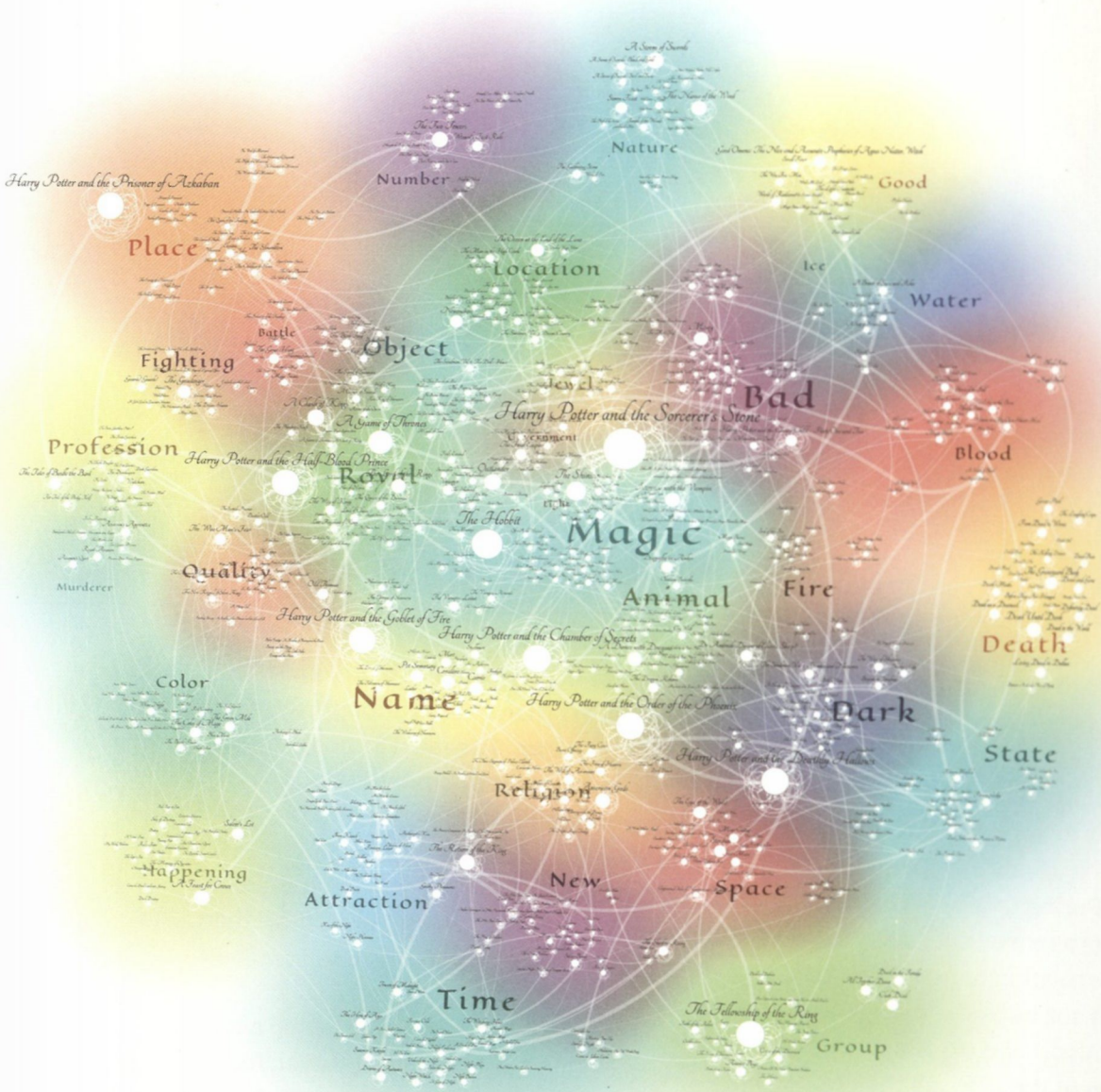
Manually enriching your data, because either doing it perfectly with the computer isn't possible or takes too long, or because the extra data is unstructured, is something that you need to embrace when doing data analysis and creating data visualizations.

In this case, after trying an automated route, I *manually* converted each unique word from all the titles into a more general term. This variable in turn became the main aspect that defined the location of the books, thus it became quite important and worth the time investment!

I loaded this curated list back into R and replaced all the specific title words with their general ones. The final data preparation step was to turn the set of fantasy book titles into a numerical matrix, which could then be used in clustering analyses. I won't go into the details of how this was done, but if you're interested, you can google for "Document Term Matrix."

Magic is Everywhere

↳ MagicIsEverywhere.VisualCinnamon.com



Magic is everywhere

Investigating patterns in Fantasy book titles

The titles from the top 10 books of the top 100 best-selling fantasy authors on Amazon were collected. Using text-mining the titles were analyzed for general subjects or terms, such as **fire**, **royal**, **time**, & more. Finally, these titles were clustered in a 2-dimensional plane, which placed books with similar themed titles together.

Created by Nadieh Bremer | Visual Cinnamon

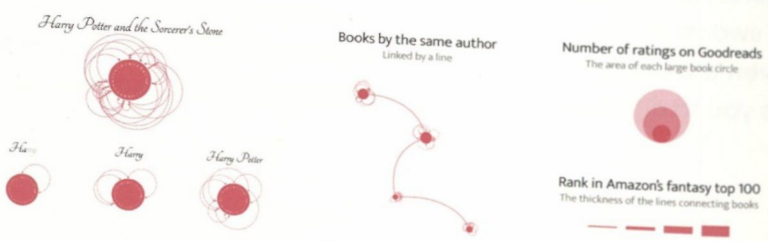
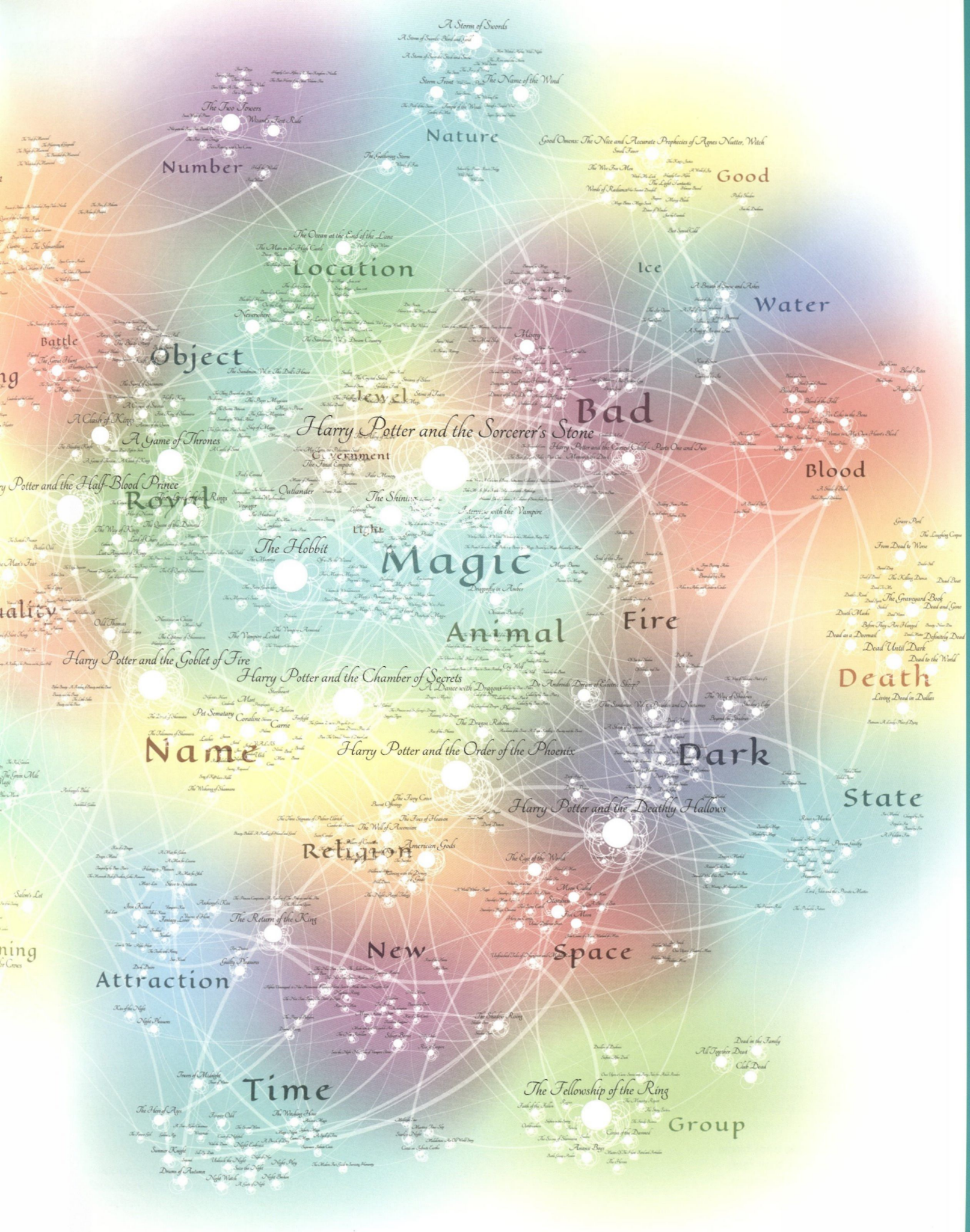


Fig.5.15

The final pos
print based
"Magic is Ev


A Storm of Swords
The Name of the Wind
Nature
The Two Towers
Number
Good Omens: The Nice and Accurate Prophecies of Agnes Nutter, Witch
Good
The Ocean at the End of the Lane
Location
Ice
Water
Battle
Object
Bad
Harry Potter and the Sorcerer's Stone
A Clash of Kings
A Game of Thrones
Government
Blood
Harry Potter and the Half-Blood Prince
Royal
Outlander
The Hobbit
Magic
Light
Animal
Fire
Harry Potter and the Goblet of Fire
Harry Potter and the Chamber of Secrets
Death
Name
Harry Potter and the Order of the Phoenix
Dark
State
Harry Potter and the Deathly Hallows
Religion
American Gods
The Return of the King
New
Space
Attraction
Time
The Fellowship of the Ring
Group

At first, I filtered by the main characters "Alexander Hamilton" and "Aaron Burr" because I was curious about how their relationship evolved throughout the musical. But Taia, my Hamilton expert, convinced me that there were enough Hamilton-Burr analyses out there already, and that the characters Eliza Hamilton (née Schuyler, Alexander Hamilton's wife) and Angelica Schuyler (Eliza's sister) would be much more interesting to explore instead. I wholeheartedly agreed.

I filtered by Eliza and Angelica, then by Eliza and Alexander, Angelica and Alexander, and finally by their conversations. I looked at what phrases they sang the most and was happily surprised to find that for Eliza, it wasn't "helpless" (the title of her main song), but "look around at how lucky we are to be alive right now" and "that would be enough/what would be enough." That was the point at which I really fell in love with her character, with her optimism, and with how much she matured throughout the story. I knew then that I had to center the story around her.

With my main story figured out, I decided to start outlining and working on my rough draft (Figure 5.8). From the beginning, I wanted to appeal to a wider audience that might not be familiar with the visualizations that I'm used to. I wanted to ease them in slowly and get them used to all the different layouts available in the filter tool. And because I knew that it would be a lengthy article, I wanted to create a delightful enough experience to keep my readers scrolling. I used `D3.js`'s new *force simulation* module to position the dots and have them explode out, dance around, and zoom back together on scroll (Figure 5.9). It was a really fun effect.

But after the introduction, I didn't know how to include all of the interesting insights I found through my analysis. I had a long stretch of writer's block and went through three rounds of rough drafts, none of which I was satisfied with (and none of which anybody will ever see. (; ๑´Ħ`๑)). I knew before I started that I would struggle with writing the most (I'm a horribly slow writer), but I reassured myself that I had the visuals covered, and even though I was slow, I wasn't a *bad* writer. How hard would it be to write and make visuals at the same time? Turns out, very, *very* hard. While I could do both tasks separately, I had never given thought to how I'd weave both the visuals and the words together.

I learne
importa
from To
"Animati
and Exp

This who
gave me
bigger re
journalist
do both.

Design to Maximize for Delight

When I first started creating visualizations, I thought that every visual element had to have a purpose; there shouldn't be flashiness for flashiness's sake. But after watching Tony Chu's "Animation, Pacing, and Exposition" talk I decided to give it a try for my Hamilton project. On scroll, I made the dots dance around the screen—a frivolous addition, but it really delighted my friends when I showed them and, more importantly, kept them scrolling.

I've been a firm believer ever since that adding subtle—and sometimes flashy—touches to my visualizations can give readers a much more enjoyable experience, even if they add nothing to the visualizations' readability and understandability.

[1] "Animation, Pacing, and Exposition" by Tony Chu: https://www.youtube.com/watch?v=Z4tB6qyxHJA.

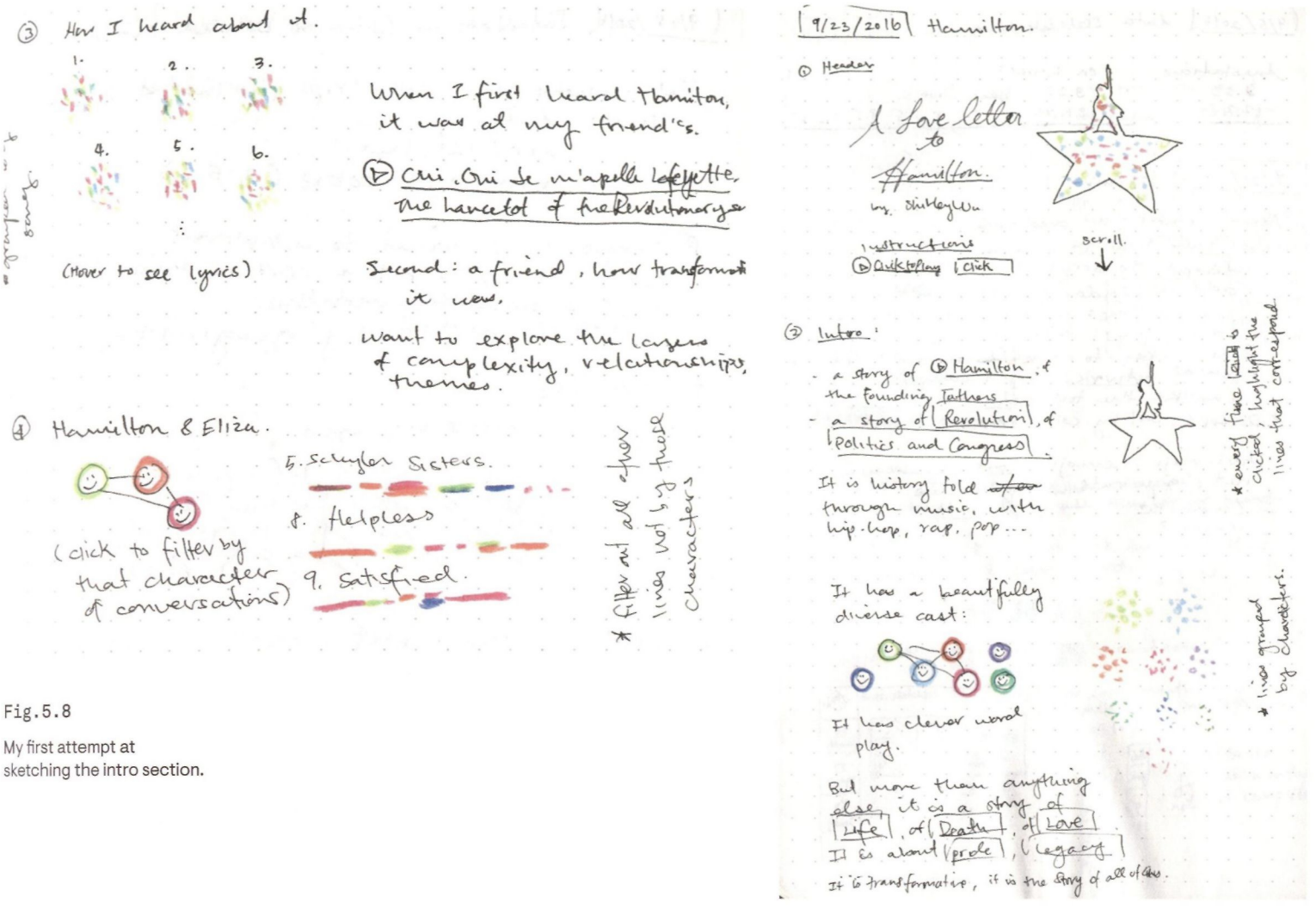


Fig.5.8

My first attempt at sketching the intro section.

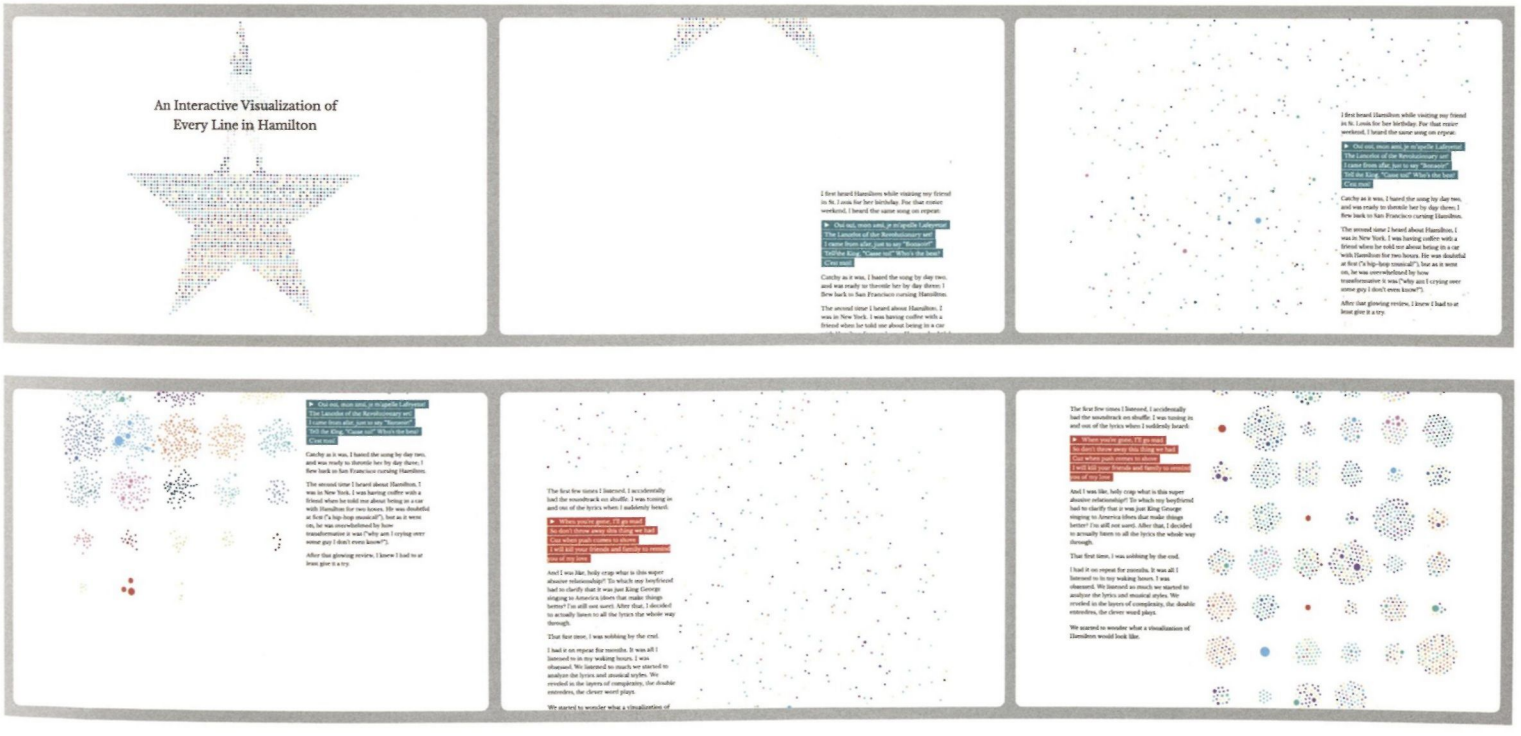
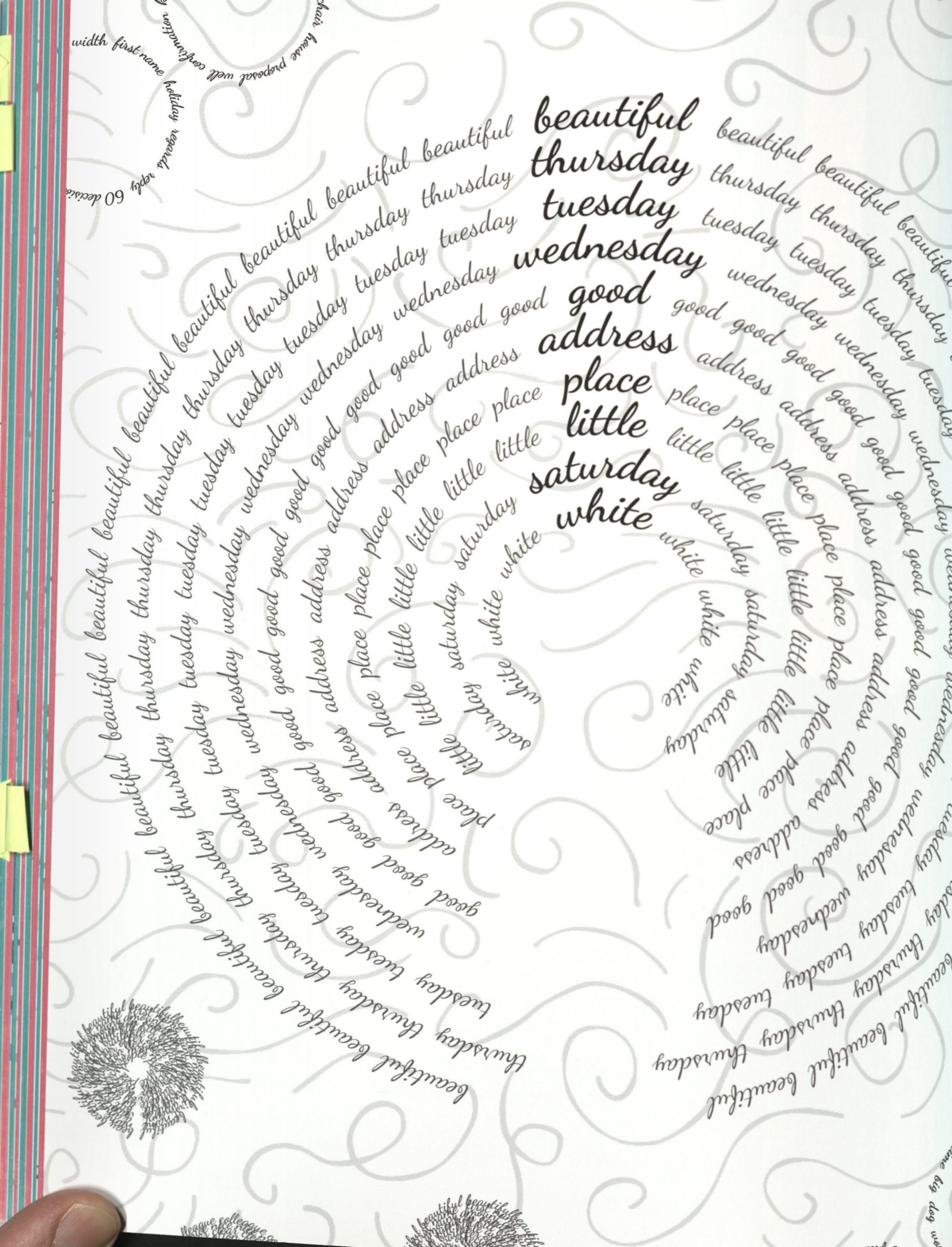


Fig.5.9

Dots exploding out and coming back together on scroll.


beautiful
thursday
tuesday
wednesday
good
address
place
little
saturday
white

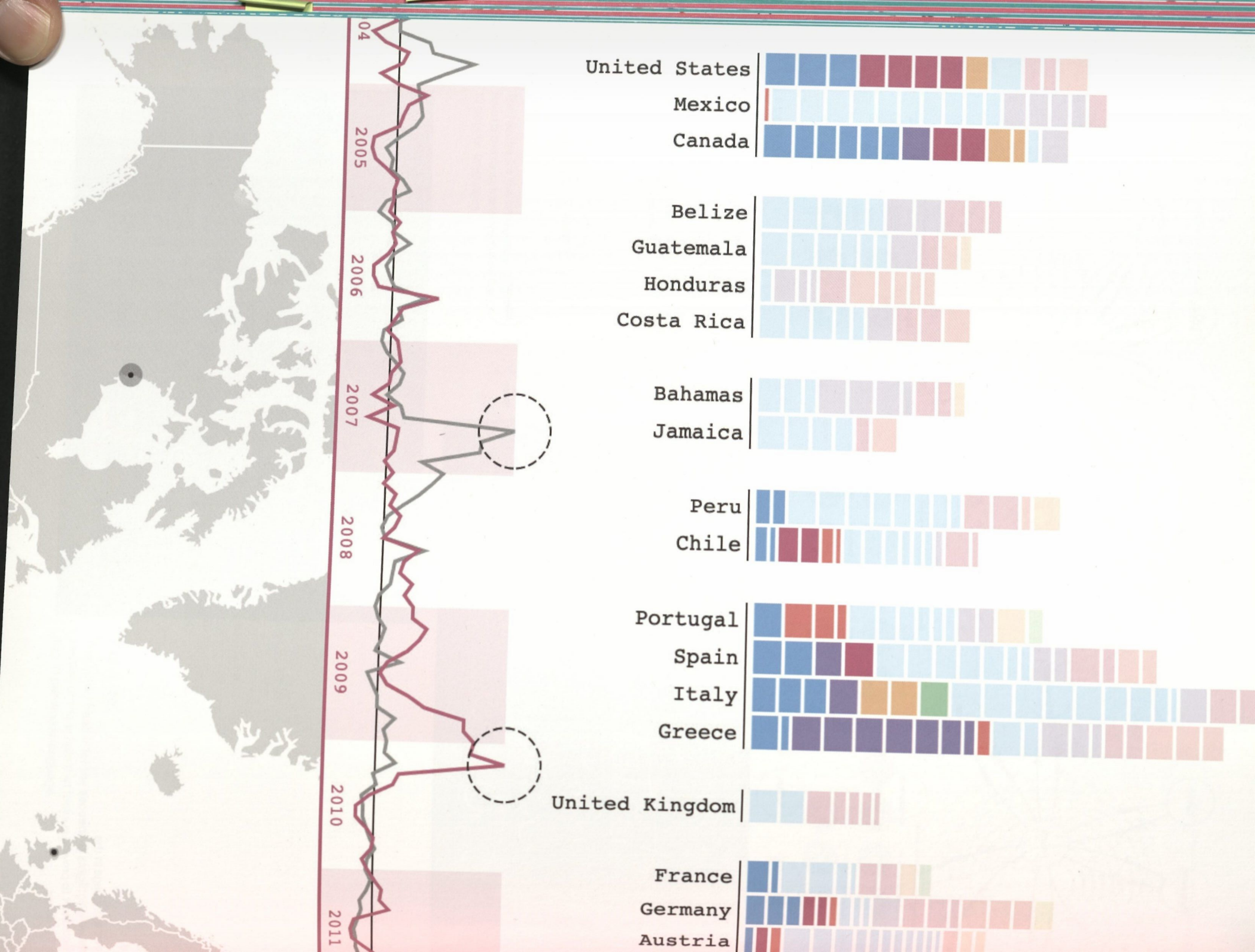
Beautiful in English

NADIEH

I still remember Shirley and me getting an email from Alberto Cairo in October 2016 that asked if we'd be interested in doing a *Data Sketches* style project for Google News Lab. Our answer: yes, of course!

We were completely free to supply our own ideas for topics as long as it revolved around Google data. So Shirley and I made a list with four topics, and each of us put our own spin on it as usual. Simon Rogers, our main point of contact at Google, picked the topic of "Culture," something a bit light and fun. Alberto oversaw the general progress of the designs, akin to an art director.

My angle for "Culture" came from something I experience with Google quite often. Being a Dutch native speaker who communicates in English about 95% of my working day, I sometimes need to translate a word into English. I either type "<<Dutch word>> in het Engels" into Google itself or go to Google Translate ("in het Engels" means "into English" in Dutch). I was really interested to know what people from other languages translate into English the most, especially if it's just one word. Do German speakers search for the same kinds of word translations as Spanish speakers? Would it reveal something about their cultures?


04
2005
2006
2007
2008
2009
2010
2011
United States
Mexico
Canada
Belize
Guatemala
Honduras
Costa Rica
Bahamas
Jamaica
Peru
Chile
Portugal
Spain
Italy
Greece
United Kingdom
France
Germany
Austria

Explore Adventure

SHIRLEY

When Nadieh and I got the email from Alberto Cairo and Simon Rogers to work with Google News Lab, I was ecstatic and beyond intimidated. After all, it was Google, it was Simon and Alberto, and they had search data going back to 2004. They had already published projects in collaboration with Accurat and Truth & Beauty (two data visualization studios whose works I find really inspiring), and I wasn't sure if I could live up to that—but I was determined to try my best.

Nadieh and I explored Google Trends and presented a few ideas, and Simon and Alberto chose our "Culture" proposal. Nadieh decided to look into language and explored the most common words people in other countries searched for to translate into English. I chose to focus on travel and dug into the travel destinations that people in one country searched for in another.

And because Alberto was a fan of our process write-ups, we agreed to include our "Culture" projects and their corresponding write-ups as part of *Data Sketches*.

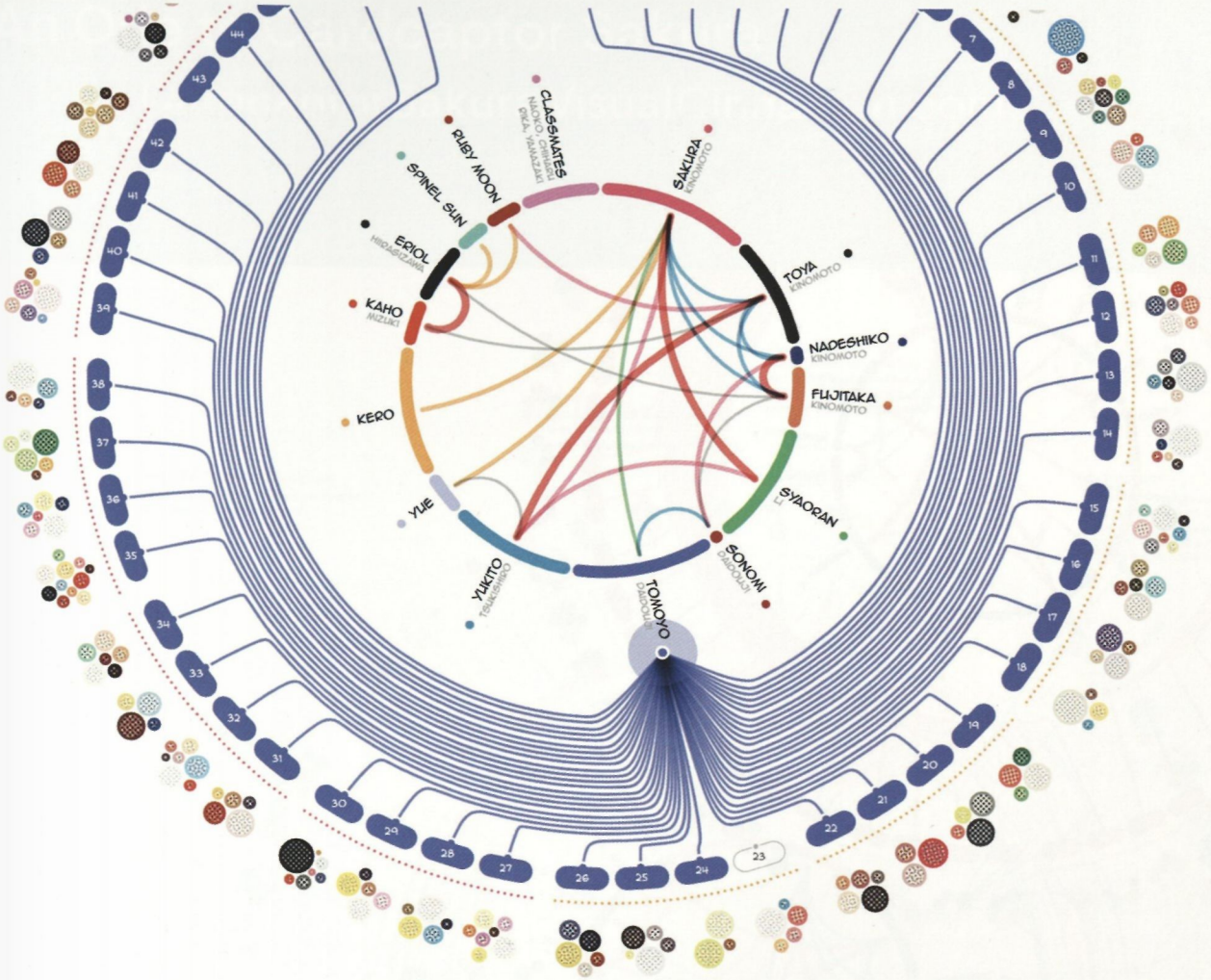


Fig.12.22

Being Sakura's bes
friend (and classm
and dress creator
and maker of videc
Tomoyo also appea
almost every chap

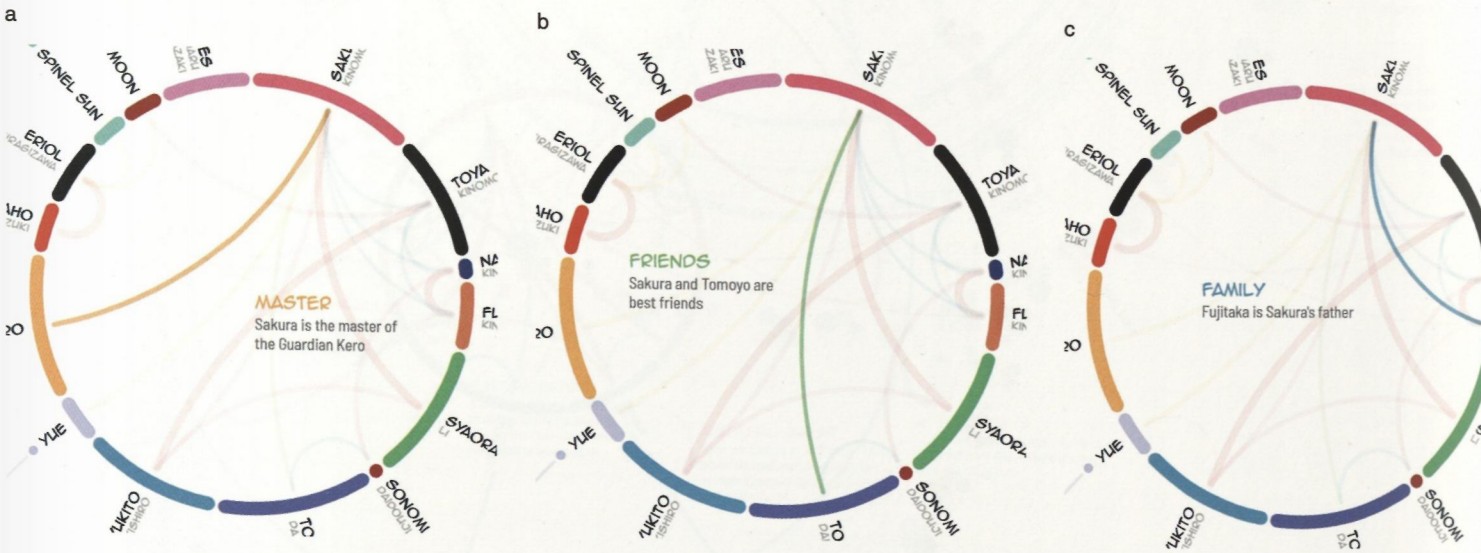


Fig.12.23
(a,b & c)

When hovering over any of the "relationship lines," a small annotation provides more detail about it.

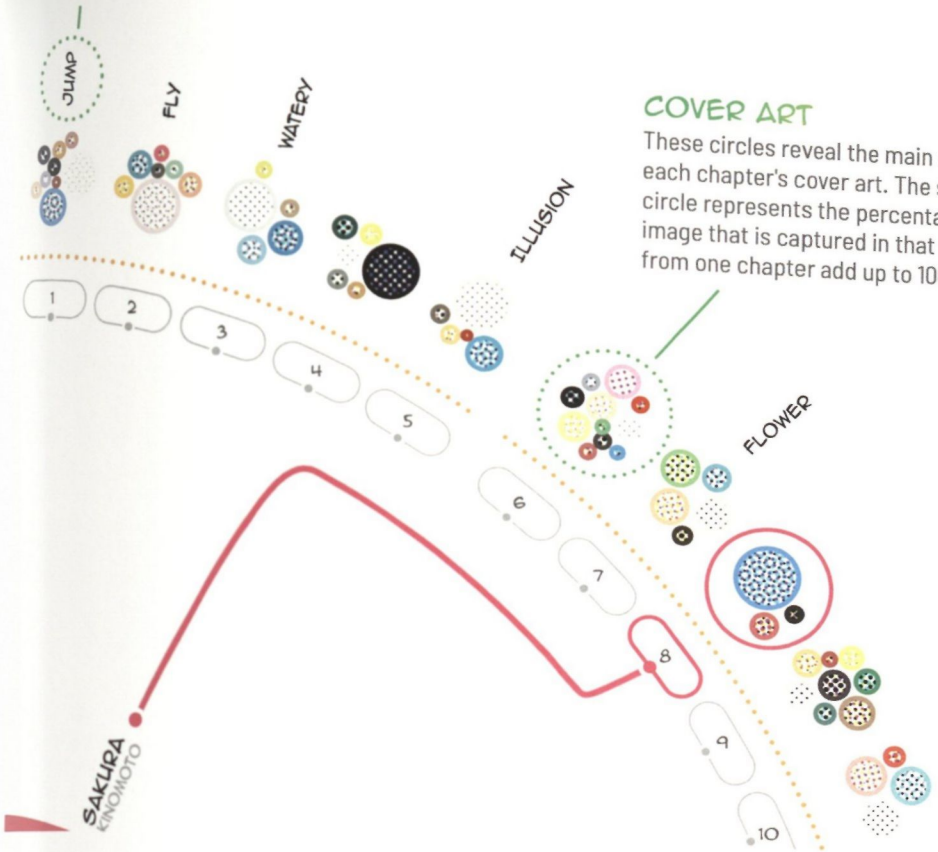


Fig.12.24

A close-up of the outer CMYK dotted circles, where each cluster of circles gives an idea of the colors used on the (gorgeous) cover art of each chapter. Hovering over any cluster will reveal a line for the characters appearing on the cover (here just Sakura).

Fig.12.26

A zoom-in on several CMYK dotted circles.

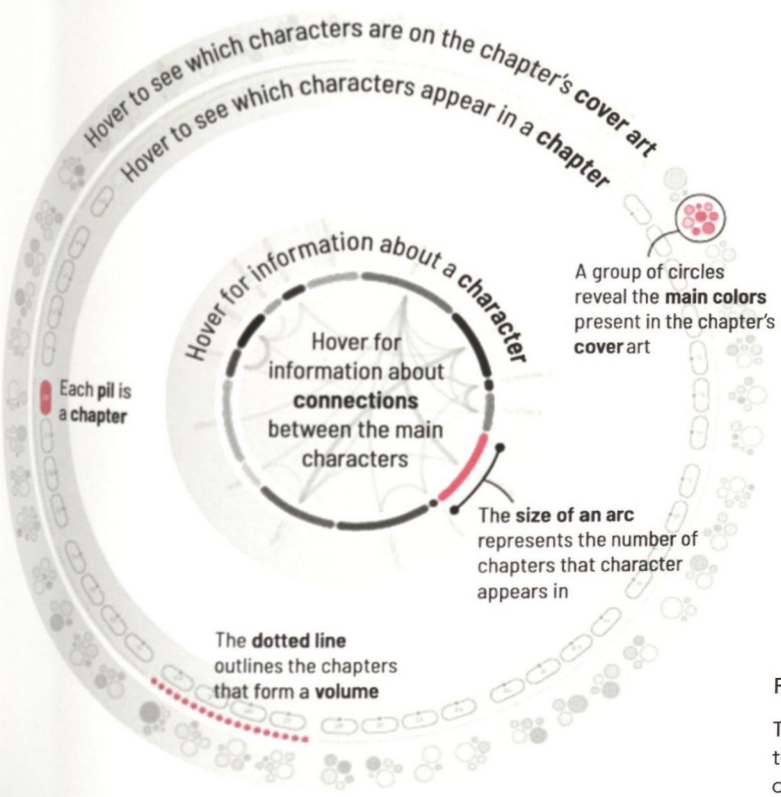


Fig.12.25

The final legend using the visual itself to overlay explanations on what each of the different rings represent.