

Lecture 3: JS, Design Rules of Thumb

DS 4200
FALL 2020
Prof. Cody Dunne
NORTHEASTERN UNIVERSITY

Slides and inspiration from Michelle Borkin, Krzysztof Gajos, Hanspeter Pfister, Miriah Meyer, Jonathan Schwabish, and David Sprague

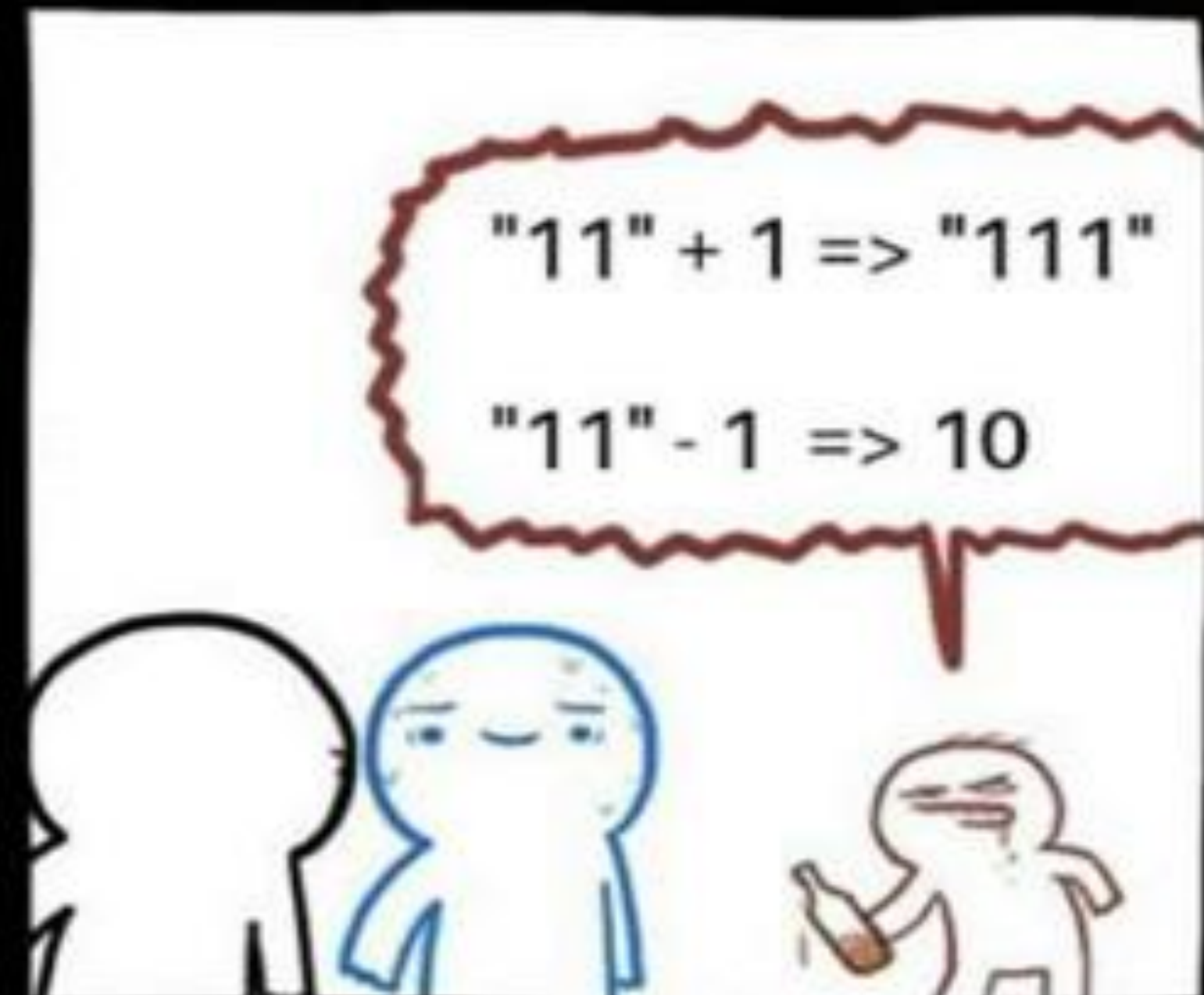
12:01AM DEADLINES?

POLL RESULTS: MOVING TO PREVIOUS DAY @
11:59PM

JS TIPS AND TRICKS



@redcoders

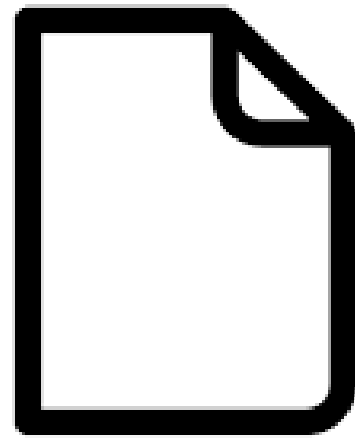


JavaScript is **bad**

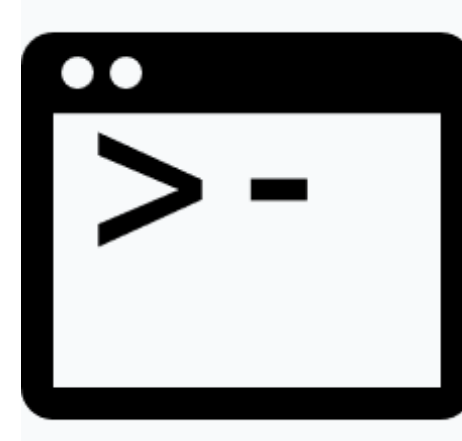
JavaScript is good

- You can change the appearance and behavior of everything that you see in a webpage
- Extremely easy to make other people access your work
- You can write good code if you know how

Starting a Project



index.html



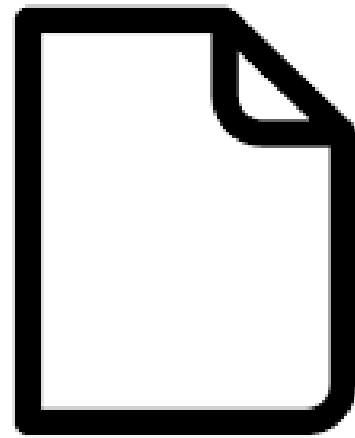
`python3 -m http.server`



Browser open on
127.0.0.1:8000

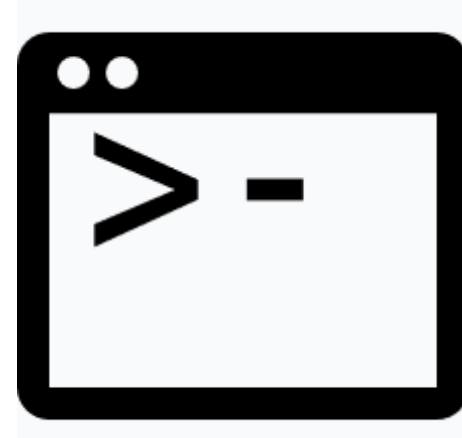
Running your code → loading page in the browser

Starting a Project



index.html

You can open index.html directly from the browser without having a server running, but **you will encounter problems later**



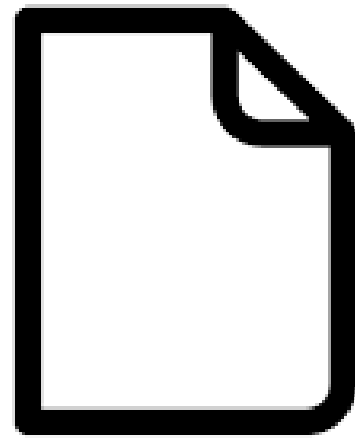
```
python3 -m http.server
```

Run this in the root folder of your project

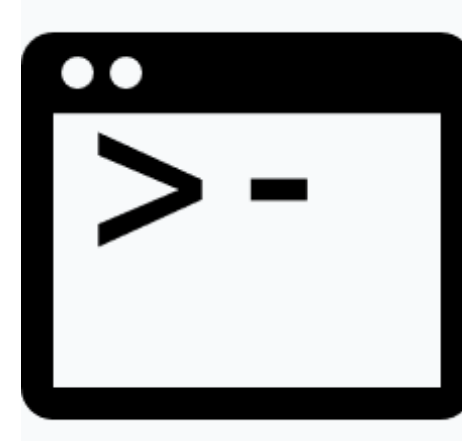


Browser open on
127.0.0.1:8000

Starting a Project



index.html



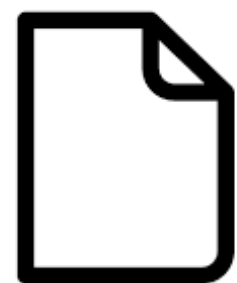
`python3 -m http.server`



Browser open on
127.0.0.1:8000



style.css



script.js

Editor recommendations

All of them are pretty light, very customizable and ready out of the box

Sublime <https://www.sublimetext.com/>

- lightweight but you can obtain everything you need through plugins
- the only one in this list that is not open source

Vscode <https://code.visualstudio.com/> (by Microsoft)

- some additional features like autocompletion are built in
- runs on electron (very customizable but heavier than necessary on resources)

Atom <https://atom.io/> (by Github)

- runs on electron too

Brackets <http://brackets.io/> (by Adobe)

- runs on electron too

Notepad++ <https://notepad-plus-plus.org/>

- Windows on C++

Not ready out of the box:

Vim

- only recommended if you want to spend a good chunk of time configuring it and learning new shortcuts.

Where do I put my script?

Where do I put my script in an HTML page?

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>title</title>
  </head>
  <body>
    <div>content...</div>
    <div>content...</div>
  </body>
</html>
```

Ways to run a script

Inline

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>title</title>
  </head>
  <body>
    <div>content...</div>
    <div>content...</div>
    <script>
      ... your code ...
    </script>
  </body>
</html>
```

- does NOT scale
- will make you very confused when your code becomes longer
- only good for fast prototyping

From another file

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>title</title>
    <script src="./main.js"></script>
  </head>
  <body>
    <div>content...</div>
    <div>content...</div>
  </body>
</html>
```

- much better, can add as many files as you want and divide your code effectively

From another file (better)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>title</title>
  </head>
  <body>
    <div>content...</div>
    <div>content...</div>
    <script src="./main.js"></script>
  </body>
</html>
```

- scripts at the end avoid need for dealing with async, defer, or onload event handlers

```
<!DOCTYPE html>
```

```
<html>
```

```
  <head>  
    <meta charset="UTF-8">  
    <title>title</title>  
  </head>
```

```
  <body>  
    <div>content...</div>  
    <div>content...</div>  
  </body>
```

```
</html>
```



Head (document metadata)

Body (content)

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>title</title>
    <script src="./main1.js"></script>
    <script src="./main2.js"></script>
  </head>
  <body>
    <div>content...</div>
    <script src="./main3.js"></script>
    <div>content...</div>
    <script src="./main4.js"></script>
  </body>
</html>
```

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>title</title>
    <script src="./main1.js"></script>
    <script src="./main2.js"></script>
  </head>
  <body>
    <div>content...</div>
    <script src="./main3.js"></script>
    <div>content...</div>
    <script src="./main4.js"></script>
  </body>
</html>
```

In head:

- Executed before everything else
- Can be used to make sure that some resources are accessible before everything else is loaded
- Can't access DOM objects (because they have not been created yet) unless forced to wait
- Loading of this script is blocking towards the loading of the rest of the resources and scripts

In body:

- Executed after some content and before some other content
- Only useful for very small, localized scripts

End of body:

- Able to access every DOM element created in body
- Executed after everything else, won't block loading of the body


```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>title</title>
    <script src="./main1.js"></script>
    <script src="./main2.js"></script>
  </head>
  <body>
    <div>content...</div>
    <script src="./main3.js"></script>
    <div>content...</div>
    <script src="./main4.js"></script>
  </body>
</html>
```

Workarounds to keep in mind if you have issues with flow control:

Option 1:

```
document.addEventListener(
  'DOMContentLoaded', function() { /*fun code to run*/ }
)
```

Use this as a starting point to wait for all content to have loaded in the DOM regardless of where you position your script

The event **DOMContentLoaded** is automatically dispatched by the browser as soon as all the resources are loaded.

Option 2:

Build system / task runner tool set up to do flow control (out of the scope of this class, Google if you want to know more)

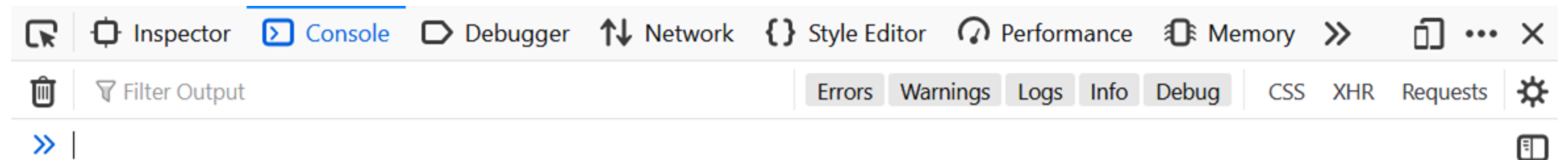
Using the browser console

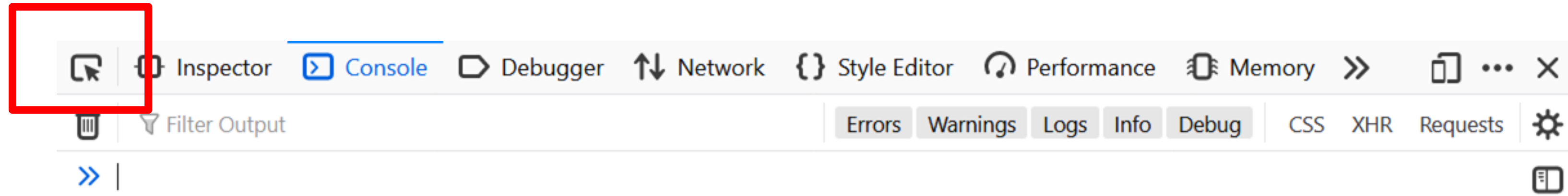
Open the browser console

Ctrl+shift+k on Firefox

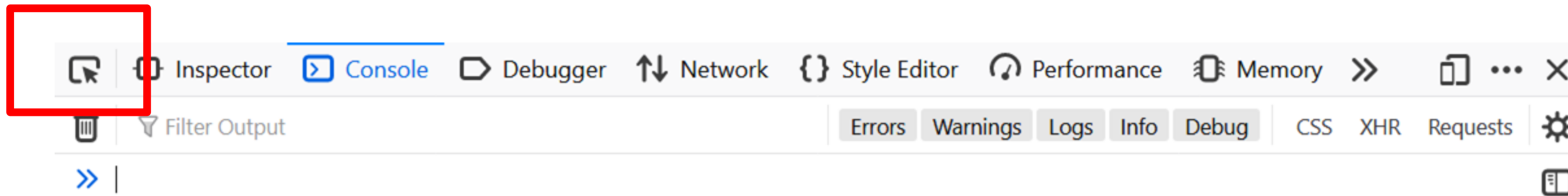
Ctrl+shift+j on Chrome

Or click anywhere on the page with your right click and select “Inspect Element” then click “Console” in the menu





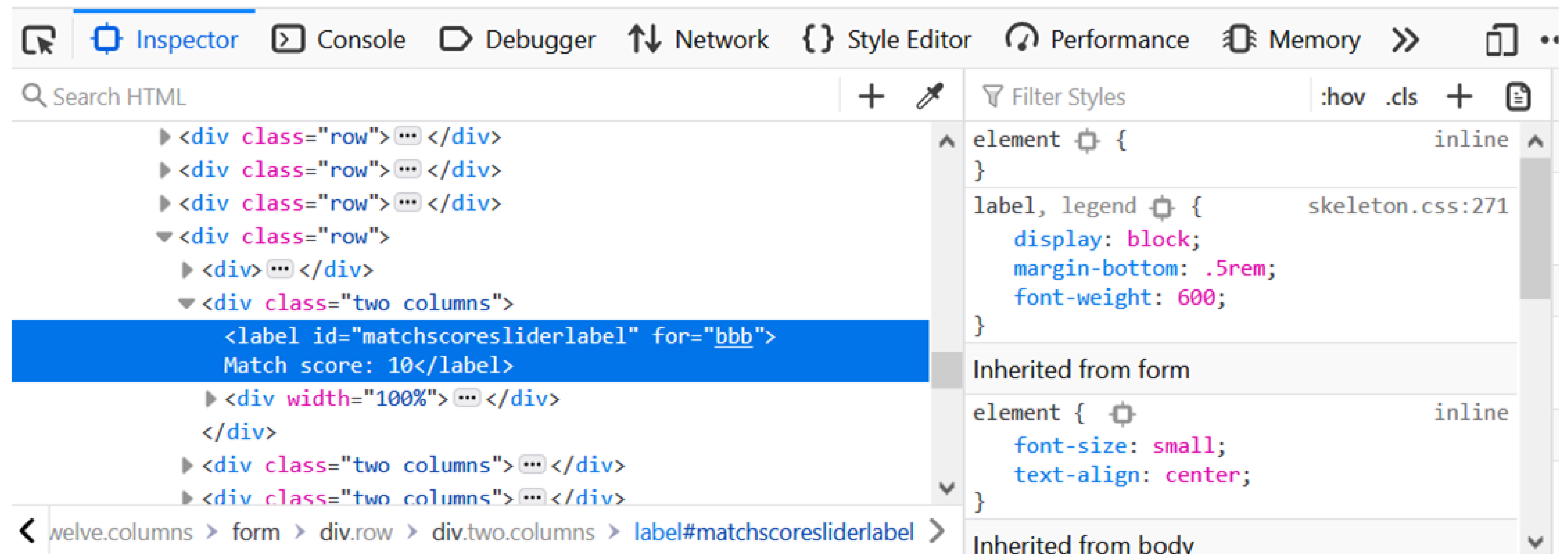
Will allow you to select any element in the page and see its properties, position in the DOM, etc.

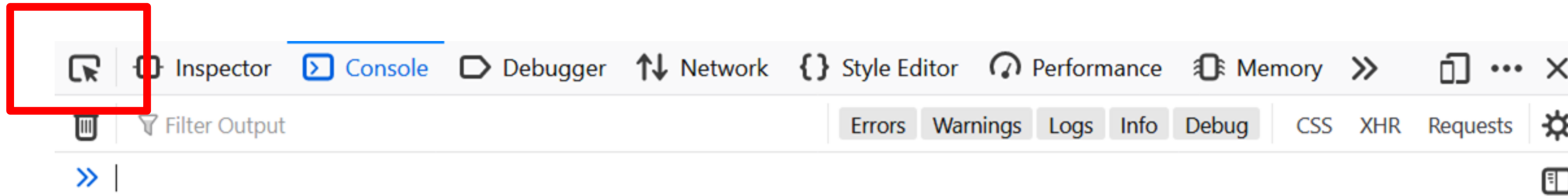


Will allow you to select any element in the page and see its properties, position in the DOM, etc.

CSS associated to selected element

Selected element in the DOM

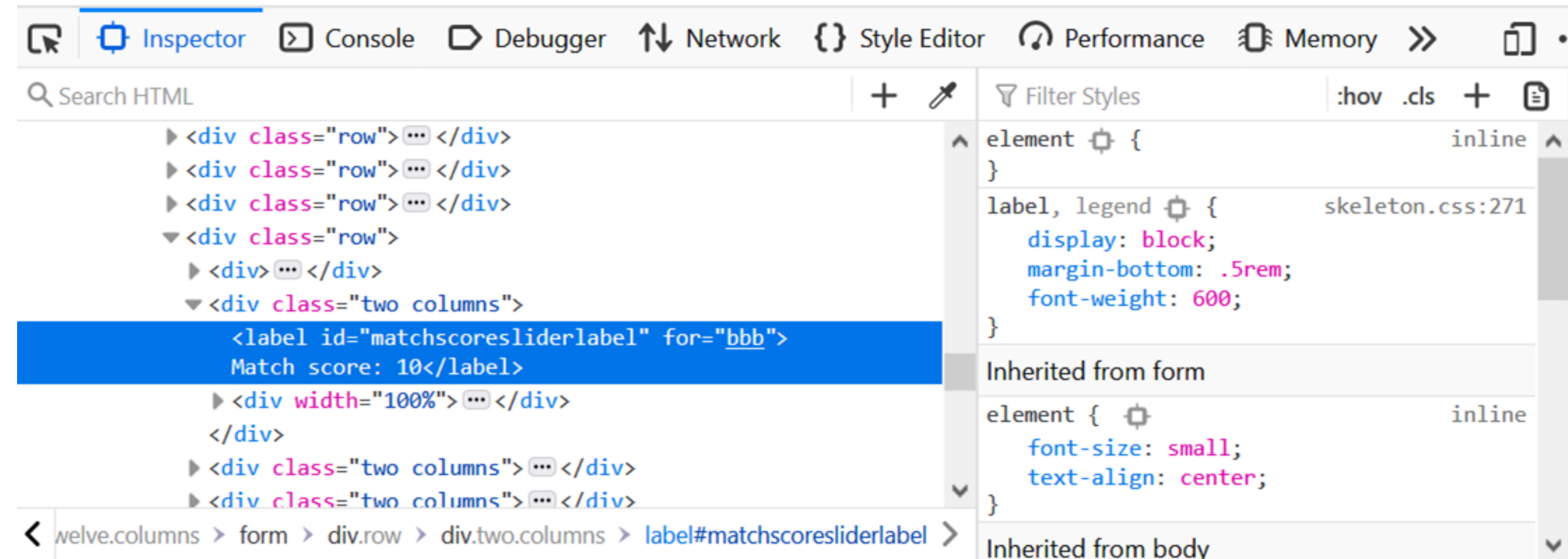


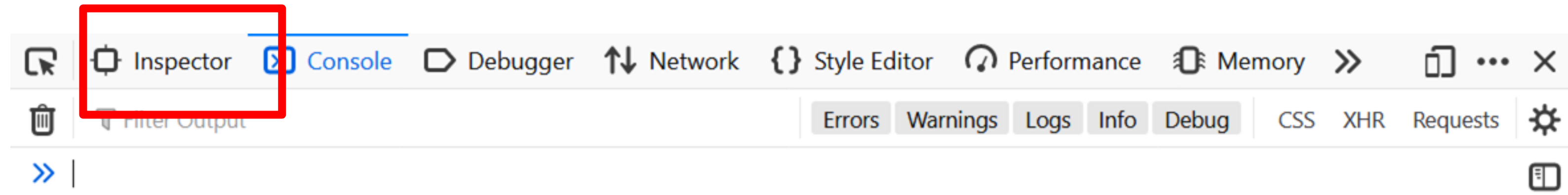


Will allow you to select any element in the page and see its properties, position in the DOM, etc.

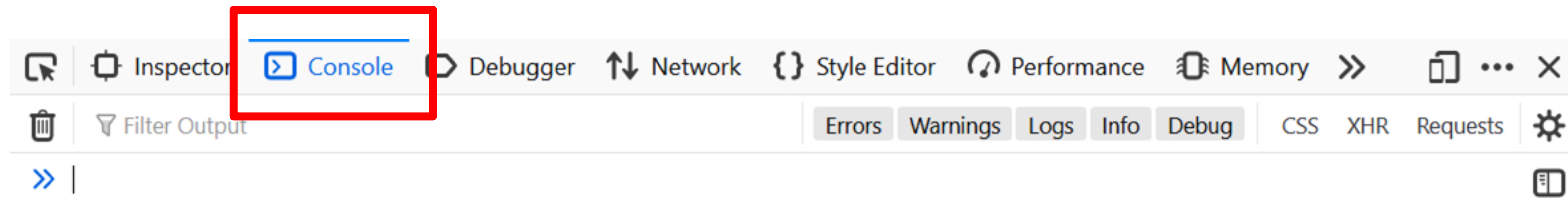
Will allow you to answer questions such as:

- What is the id of this element that I am seeing?
- Is this element in the correct position in the DOM?
- What events are associated to this element?
- How would this element look like if I make it red without having to re-run the whole page?



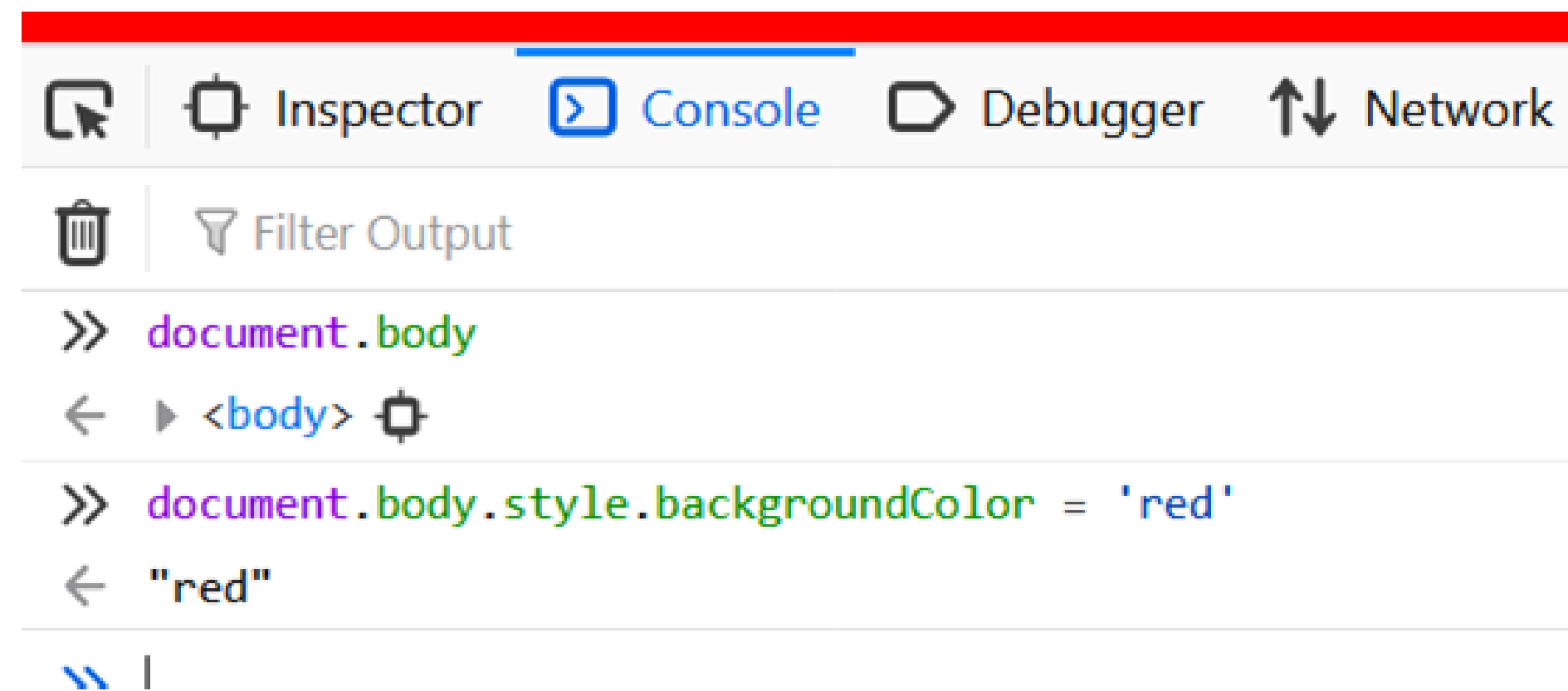


Shows the structure of the page plus CSS style associated with it



Shows print output and **errors**
Can run scripts after page is loaded

example:



Everything is an object

And everything can be printed in the console

If you **print an object in the browser console**, you can **navigate the fields of the object** and the functions associated with it

```
>> graph_sandbox
<- [-]
  animate: true
  ▶ data: Array(25) [ (4) [-], (3) [-], (4) [-], - ]
  ▶ grid: Array(7) [ undefined, (39) [-], (39) [-], - ]
  horizontal_spacing: 172.8
  init_padding: 43.2
  left_padding: -43.2
  ▶ levels: Array(5) [ "very_high", "high", "normal", - ]
  link_opacity: 1
  link_stroke_width: 4
  ▶ links: Array(150) [ {-}, {-}, {-}, - ]
  max_iterations: 20
  max_rank: 27
  node_width: 34.56
  ▶ nodes: Array(150) [ {-}, {-}, {-}, - ]
  ▶ opt: Object { numDays: 25, animate: true, minEventPerColThreshold: 3, - }
  ▶ path: Array(7) [ "source", "Breakfast", "Lunch", - ]
  ▶ sink: Object { depth: 5, color: "gray", fake_out: true, - }
  ▶ source: Object { depth: 0, color: "#fff", fake_in: true, - }
  ▶ start_heights: Object { very_high: 0, high: 7, normal: 14, - }
  ▶ svg: Object { _groups: (1) [-], _parents: (1) [-] }
  svg_index: 2
  svgname: "braids-container-sandbox"
  top_padding: 80
  vertical_spacing: 8
  ▶ <prototype>: Object { - }
```

Note: you can access any DOM element too as JavaScript objects

Callbacks and events

Callbacks and events

“Event-driven architecture”: the flow of a program is defined by **events**.

Events can be generated by the user or by the browser. Examples of events that you will want to use a callback for:

- . user interacts with an element
- . loading of a resource is completed
- . browser window is resized
- . request to some API is returned
- ...

Callbacks and events

Most of the events that you will use are already defined by the browser.

Examples:

- **mouseover**: cursor enters the bounding box of a specified element
- **mouseout**: cursor exits the bounding box of a specified element
- **onClick**: user clicks on specified element
- **onWindowResize**: browser window is resized
- **onDocumentReady**: all resources in document are loaded

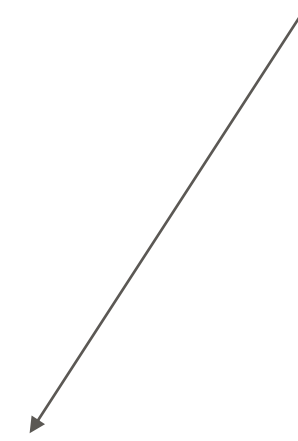
You can also define and dispatch your own events

Callbacks and events

Adding an event listener to an item:

```
item.on('mouseover', function(){  
    console.log('hello');  
})
```

a callback



Events are usually managed using callbacks.

Callbacks are nameless functions that are executed after a condition is verified.

Callbacks and events

Adding an event listener to an item:

```
item.on('mouseover', function(){  
    console.log('hello');  
})
```

a callback

≈

```
item.on('mouseover', () => {  
    console.log('hello');  
})
```

Events are usually managed using callbacks.

Callbacks are nameless functions that are executed after a condition is verified.

Callbacks and events

Callbacks are not only for events:

```
myArray = [1, 2, 3, 4, 5, 6]
result = myArray.filter( function(a) => {
    return a%2==0
})
// returns [2, 4, 6]
```

In this case, we use a callback to filter an array, keeping only even numbers

Callbacks and events

Similar to **lambdas** in python

JS

```
myArray = [1, 2, 3, 4, 5, 6]
result = myArray.filter(function(a) => {
    return a%2==0
})
// returns [2, 4, 6]
```

Python

```
myArray = [1, 2, 3, 4, 5, 6]
result = list(filter(lambda a: (a%2 == 0),
myArray))
// returns [2, 4, 6]
```


Ways to declare a variable

- `x = 5;`

- `var x = 5, y = 6, z = 7;` Global

- `let x = 5;` Scope of the variable is constrained to the scope in which it has been declared.

- `const x = 5;` Scope limited, x has to be constant.

Recommended to generally use **let**
and **const** instead of **var**

```
if (true) {  
    var foo = 5;  
}
```

```
console.log(foo); // 5
```

```
if (true) {  
    let foo = 5;  
}
```

```
console.log(foo); // undefined
```

Always be aware of the data type that you are dealing with



 **ShadowCheetah**
@shadowcheets

Javascript is weird.

```
> ('b' + 'a' + + 'a' + 'a').toLowerCase()  
< "banana"
```

1:30 PM · Aug 12, 2019 · [TweetDeck](#)

65 Retweets **206** Likes

<https://github.com/denysdovhan/wtfjs>

Ways to declare a function

Function declaration

```
function name (params) {  
    ...  
}
```

Function expression

```
let name = function (params) {  
    ...  
}
```

Arrow function

```
let name = (params) => {  
    ...  
}
```

All of these will have almost the same effect

Hoisting: a function will be positioned at the top of the scope and made available at any point of its own scope even before its own declaration

Arrow functions will let you write a lot of fun oneliners:

```
// custom sorting function  
[3, 1, 2, 4].sort((a, b) => a < b)  
→ [1, 2, 3, 4]
```

```
// custom filtering function  
[1, 2, 3, 4].filter(a => a%2 == 0)  
→ [2, 4]
```

```
// sum of all elements in an array  
[1, 2, 3, 4].reduce((a, b) => a + b, 0)  
→ 10
```

```
// sort then filter then sum  
[3, 1, 2, 4].sort((a, b) => a < b).filter(a => a%2 == 0).reduce((a, b) => a + b, 0)  
→ 6
```

Style guides

Google style guide: <https://google.github.io/styleguide/javascriptguide.xml>

Airbnb: <https://github.com/airbnb/javascript>

Standardjs: <https://standardjs.com/#the-rules>

Idiomatic: <https://github.com/rwaldron/idiomatic.js>

Linting

Linters force you to write code following some pre-established policies.

Jshint: <http://www.jshint.com/>

jshint: <https://jshint.com/> started as a fork of jshint, customizable

prettier: <https://prettier.io/> customizable

Automated code review

one of many tools to check issues in your code:

<https://www.codacy.com/>

(example)

<https://app.codacy.com/app/picorana/sparqling/files?bid=7480002>

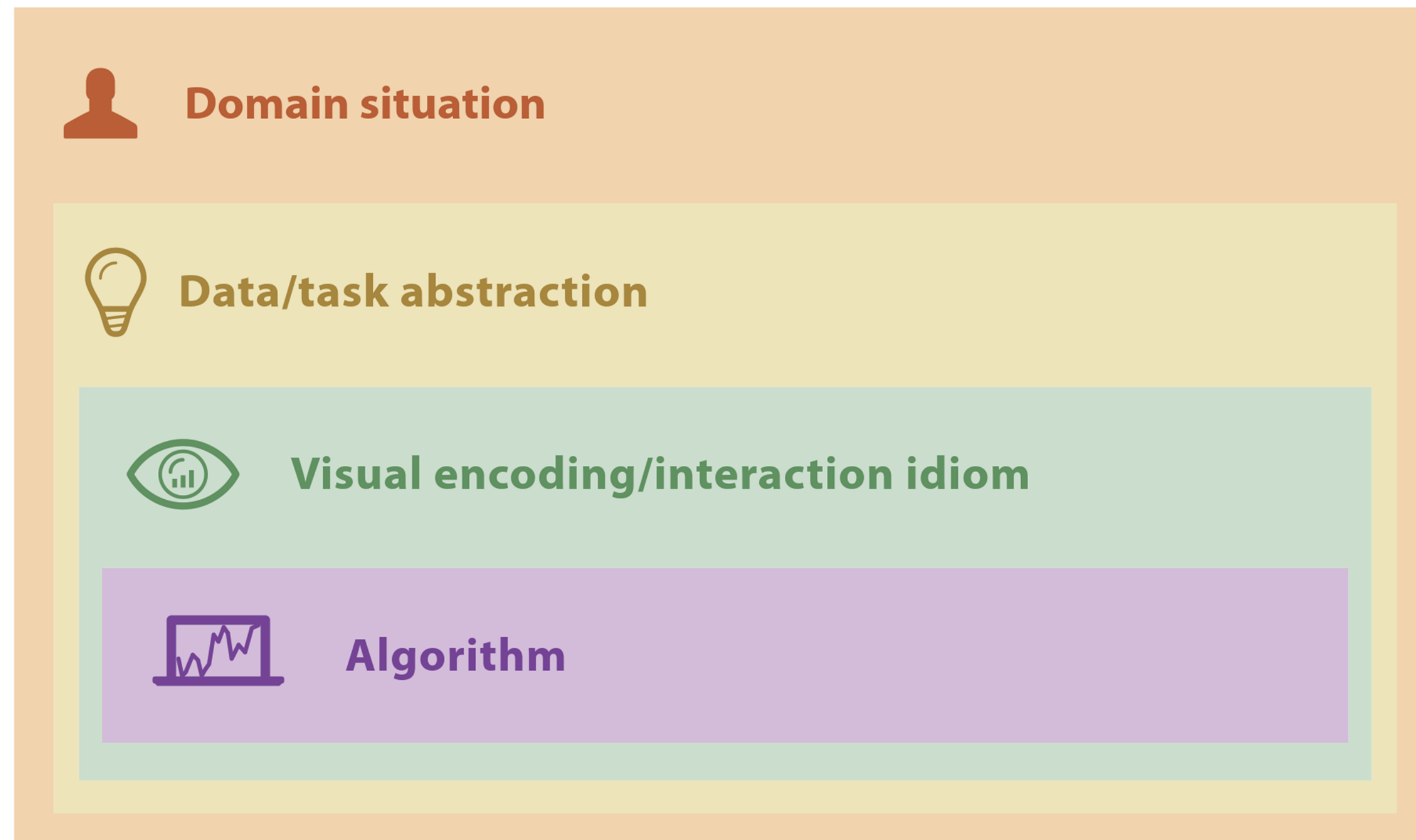
IN-CLASS PROGRAMMING —

JAVASCRIPT

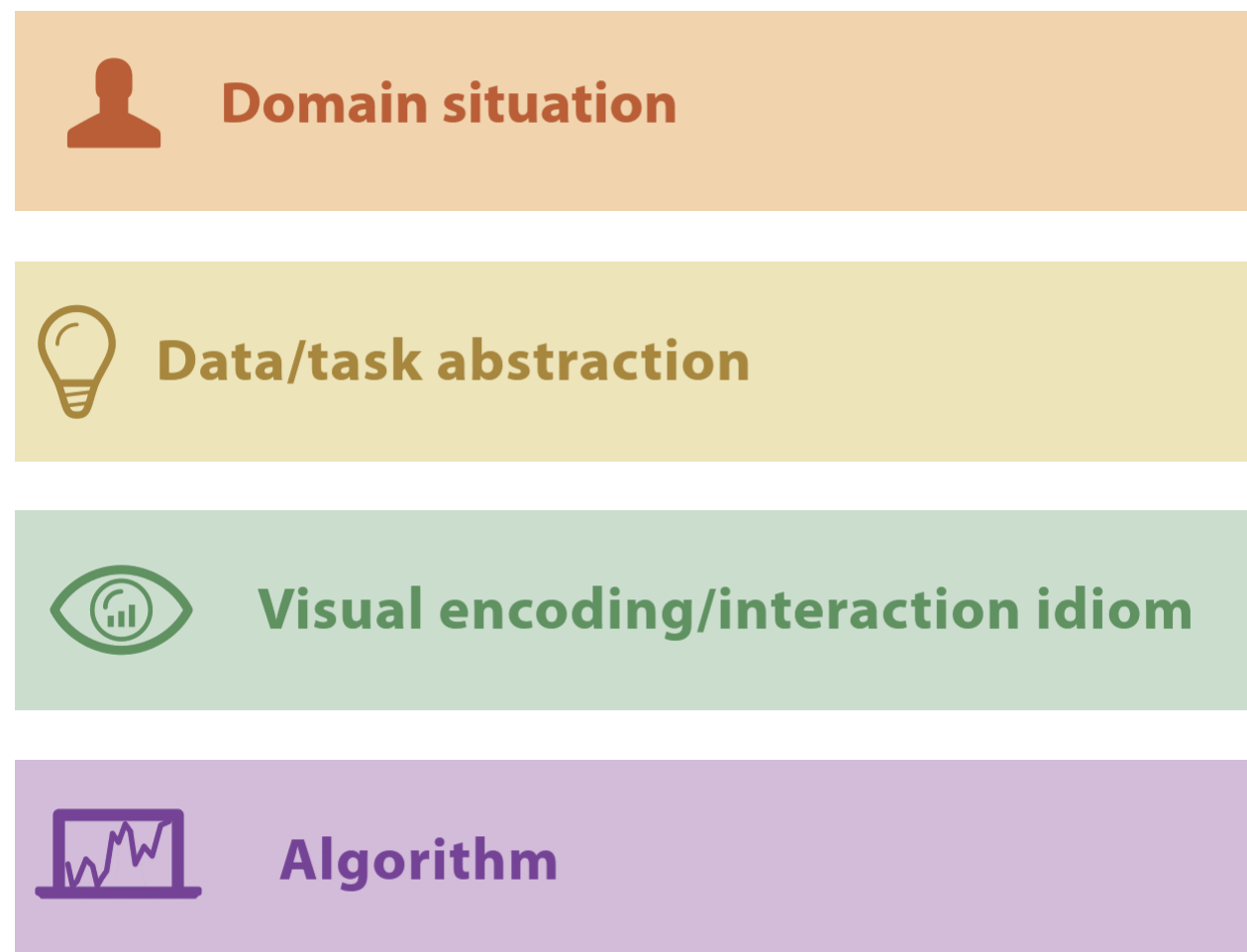
~25 min total

PREVIOUSLY, ON DS 4200...

Nested Model



Threats to Validity *✓ Final Project validation*



- Threat** Wrong problem ✓
 - Validate** Observe and interview target users
- Threat** Wrong task/data abstraction
 - Threat** Ineffective encoding/interaction idiom ✓
 - Validate** Justify encoding/interaction design
 - Threat** Slow algorithm
 - Validate** Analyze computational complexity
 - Validate** Measure system time/memory
- Validate** Qualitative/quantitative result image analysis
Test on any users, informal usability study ✓
- Validate** Lab study, measure human time/errors for task

- Validate** Test on target users, collect anecdotal evidence of utility
- Validate** Field study, document human usage of deployed system
- Validate** Observe adoption rates

Final
project
follow-up

PROJECTS

(Using the nested model via *design study “lite” methodology*)

<https://northeastern.instructure.com/courses/18721/pages/project-overview>

EXPERIENTIAL LEARNING PROJECTS

Why are we doing experiential learning?

Design Study “Lite” Methodology ([Borkin et al. 2017](#), [Syeda et al. \(2020\)](#))

- Design studies are a growing and valuable research area.
- Real-world data visualization experience.
- Visualization for exploration and communication.
- A more realistic experience of creating visualizations, and doing work in general.
- Teaches design, interview, evaluation, communication, and feedback techniques difficult to replicate in a classroom.
- Higher-stakes deliverables.
- Professional development.
- Make a positive impact in the community.
- Publication?

EXPERIENTIAL LEARNING PROJECTS

What are the challenges?

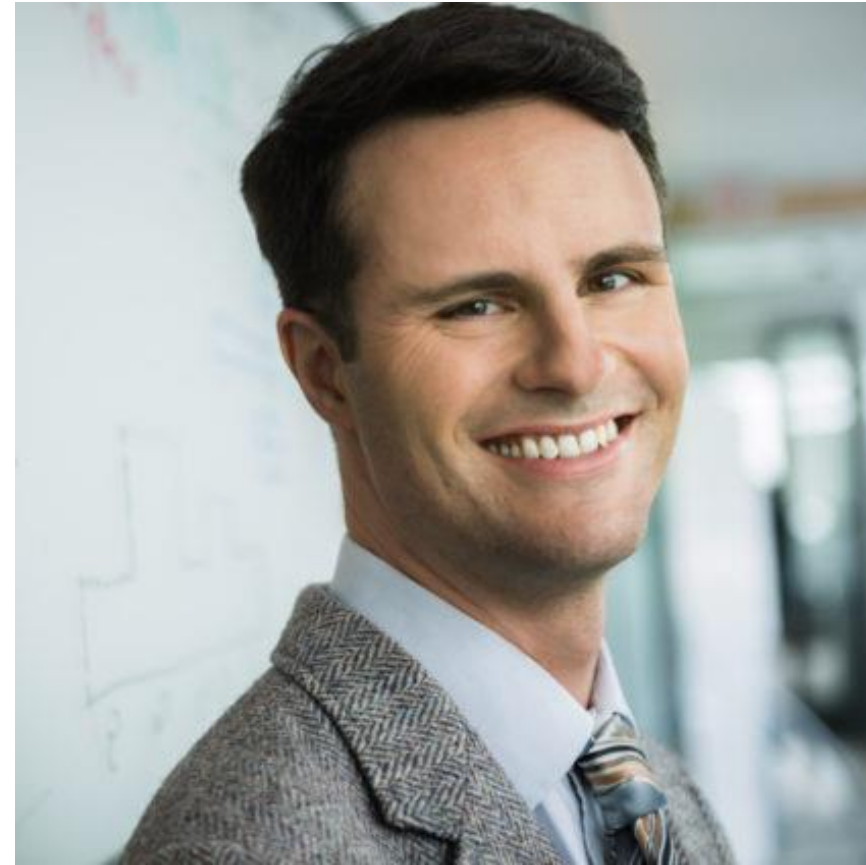
- Real-world data is messy and difficult to gather and process.
- Partners may not have clear goals and expectations.
- There is communication and scheduling overhead, inc. for teaching staff to differentiate assignment grading if necessary.
- Project areas may be too predefined.
- Project areas may be too ambiguous.
- May not actually make a meaningful impact.
- Reduces time for white-room technical education.
- More ambiguous expectations and grading challenges.
- Possible variation in student workload.
- Students may not know they are signing up for Service-Learning in advance (common problem with our registrar).

EXPERIENTIAL LEARNING PROJECTS

Who to blame for getting you into this?



Michelle Borkin



Cody Dunne



EXAMPLES OF SUCCESSFUL COURSE PROJECTS

(Albeit with different requirements per course)

PROJECT EXAMPLE — WWOVIS

Close and Distant Reading via Named Entity Network Visualization: A Case Study of Women Writers Online

Sarah Campbell ^{*} Zheng-yan Yu [†] Sarah Connell [‡] Cody Dunne [§]
Northeastern University

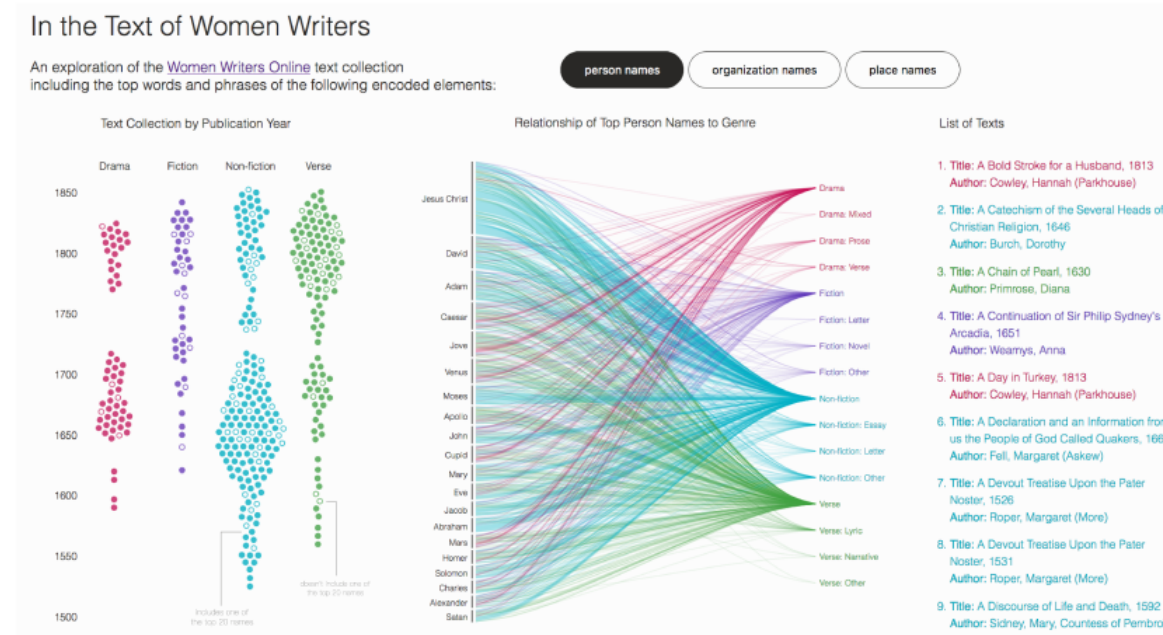


Figure 1: Three linked visualizations showing a named entity network queried from the Women Writers Online textbase. Left: a beeswarm visualization of the genre of each publication by year. Middle: a bipartite node-link visualization of the top 20 named entities connected to the genres of texts they reside in. Right: a list of texts that include at least one top 20 named entity, ordered alphabetically and linked to the full text. Marks are colored categorically by genre: drama is pink ●, fiction is purple ●, non-fiction is blue ●, and verse is green ●. Empty circles show texts that do not include any of the top 20 named entities, e.g. ○.

ABSTRACT

Close reading and distant reading are widely used in digital humanities and can benefit from information visualizations. Digital humanities scholars have curated numerous TEI-encoded textual collections which provide the data necessary for blending both close and distant reading – however we do not have tools to support general users in conducting these blended analyses. In this paper we focus on one such collection: Women Writers Online (WVO). We contribute the design and implementation of a multiple coordinated view network visualization to facilitate close and distant reading in WVO and a transparent view into our iterative design process to help guide future designers and humanists in applying our approach to other textual collections.

Index Terms: Applied computing—Education—Digital libraries and archives; Human-centered computing—Visualization—Visualization application domains—Information visualization; Human-centered computing—Visualization—Visualization techniques—Graph Drawings;

^{*}e-mail: campbell.sar@husky.neu.edu

[†]e-mail: yu.zheng@husky.neu.edu

[‡]e-mail: sa.connell@northeastern.edu

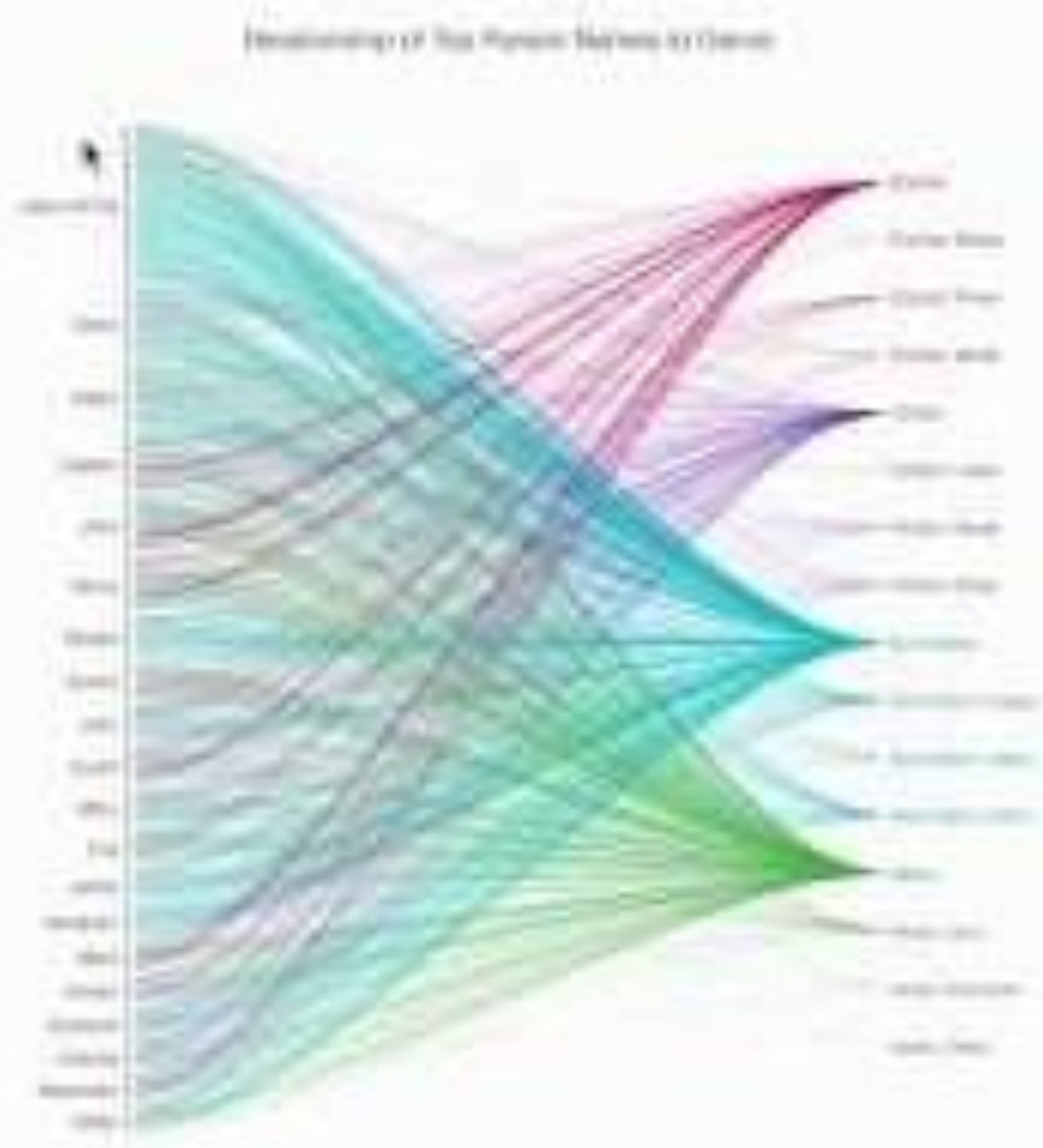
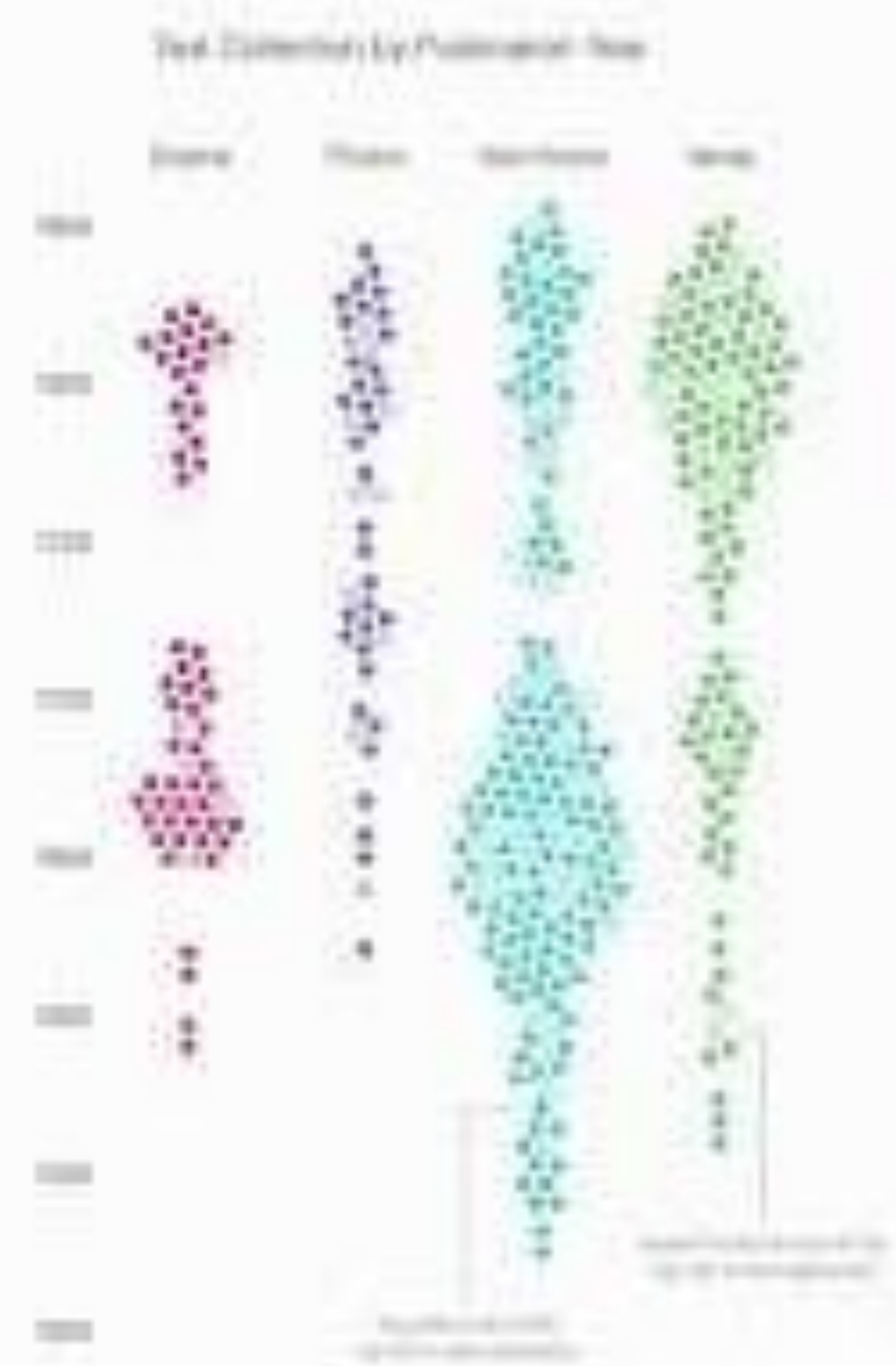
[§]e-mail: c.dunne@northeastern.edu

1 INTRODUCTION

Close and distant reading are two important tools in the digital humanities toolbox which can both benefit from information visualization [6]. **Close reading** is the traditional method for literary criticism. Several visualizations have been developed to support close reading, but existing approaches can benefit from adding supplementary **named entity** information, especially acting persons and their relationships [6]. **Distant reading**, introduced by Moretti [11], alternatively focuses on an abstract view of global features of one or more texts. Network/graph visualizations can be particularly useful for examining relationships between these features and texts for corpus analysis [6]. We designed an interactive visualization to support a blend of close and distant reading – both explorations at scale and text-level investigation.

This paper focuses on the application of our visualization approach in service to the Women Writers Project (WWP). The WWP is a long-term digital humanities research project at Northeastern University, devoted to early modern women’s writing and electronic text encoding. The goal of the project is to bring texts from pre-Victorian women writers out of the archive and make them more accessible to a wide audience of teachers, students, scholars, and the general user. We focus on the WWP’s major textual collection, Women Writers Online (WVO). WVO is a full-text collection of early women’s writing in English. It currently includes full transcriptions, encoded following the standards of the Text Encoding Initiative (TEI), of 407 texts published between 1526 and 1850. In addition to the collection’s broad chronological framing, the texts in WVO also represent a very diverse set of genres, ranging from prophecies, religious meditations, petitions, and recipe books to






CS 7260 FALL 2017:
VISUALIZATION FOR
NETWORK SCIENCE



- ### List of Items
1. Item 1: First Milestone (1990)
 2. Item 2: Second Milestone (1995)
 3. Item 3: Third Milestone (2000)
 4. Item 4: Fourth Milestone (2005)
 5. Item 5: Fifth Milestone (2010)
 6. Item 6: Sixth Milestone (2015)
 7. Item 7: Seventh Milestone (2020)
 8. Item 8: Eighth Milestone (2020-Present)

PROJECT EXAMPLE — JUST TYPEICAL

Just TYPEical: Visualizing Common Function Type Signatures in R

Cameron Moy * Julia Belyakova  Alexi Turcotte  Sara Di Bartolomeo  Cody Dunne 

Northeastern University

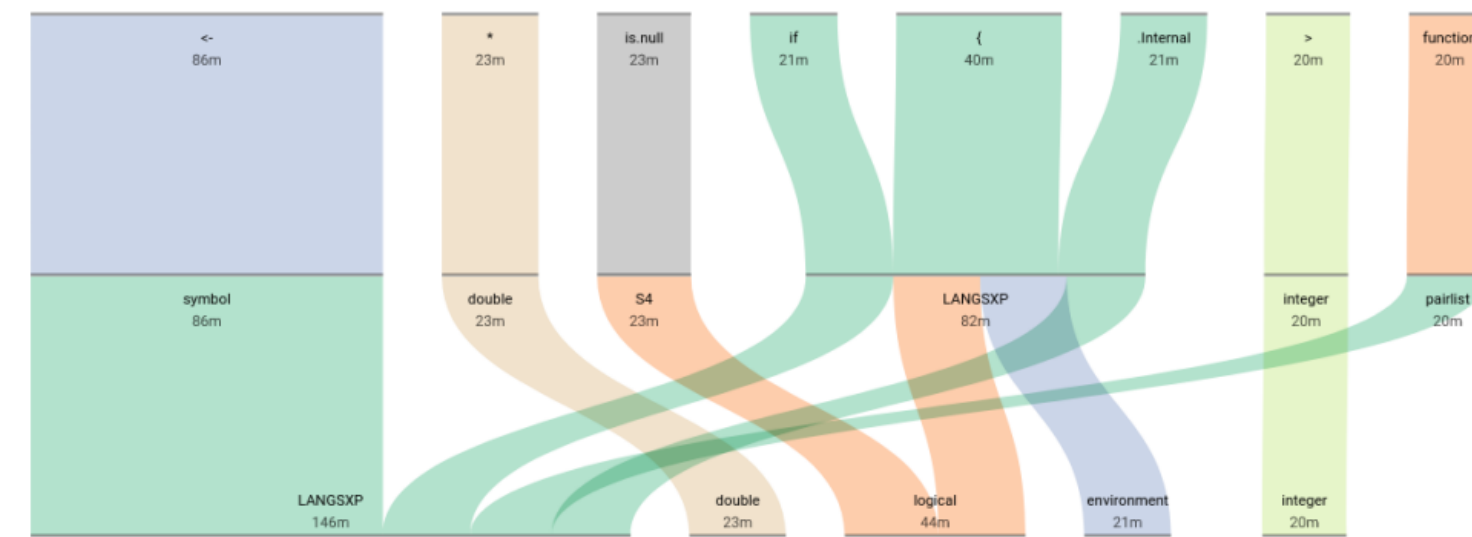


Figure 1: Our type flow visualization showing type signatures for a subset of R's base package functions. Function names are listed at the top followed by the first two argument types. Complete signatures are shown in the full visualization (Fig. 2).

ABSTRACT

Data-driven approaches to programming language design are uncommon. Despite the availability of large code repositories, distilling semantically-rich information from programs remains difficult. Important dimensions, like run-time type data, are inscrutable without the appropriate tools. We contribute a task abstraction and interactive visualization, TYPEICAL, for programming language designers who are exploring and analyzing type information from execution traces. Our approach aids user understanding of function type signatures across many executions. Insights derived from our visualization are aimed at informing language design decisions — specifically of a new gradual type system being developed for the R programming language. A copy of this paper, along with all the supplemental material, is available at osf.io/mc6zt

Index Terms: Human-centered computing—Visualization

1 INTRODUCTION

Programming languages commonly evolve by decree. Often, the language designer decides that a new feature is necessary, or that a past feature was ill-conceived. Thus, the language moves forward — forcing its users to adapt to the changes. However, rarely is language design informed by empirical data on how programmers *actually write* software in practice [6].

Thanks to the prevalence of open source code, it is feasible to collect data on the use of popular programming languages. Vast quantities of code are publicly available on language-specific package servers. To inform programming language design, this collected data needs to be analyzed and interpreted. Programs are complex and highly structured, so researchers often employ static and dynamic

*E-mails: [[camoy](mailto:camoy@northeastern.edu) | [belyakovay](mailto:belyakovay@northeastern.edu) | [alex_i](mailto:alex_i@northeastern.edu)]@ccs.neu.edu, [[dibartolomeo](mailto:dibartolomeo@northeastern.edu) | [c.dunne](mailto:c.dunne@northeastern.edu)]@northeastern.edu

analyses to gather information about specific aspects of programs. Even then, it may be difficult to make sense of the results of these analyses, especially if the data set is large.

Programming language design, and type system development in particular, can make use of run-time type signature information. A *type signature* describes the argument and return types a particular function is called with at run time. A *type system* provides a conservative approximation of run-time types. Understanding the frequency of type signatures in the wild is key for the development of new gradual type systems, whose adoption depends on integrating well with existing code. Without data-driven tools, type system designers are left to guess how their language is used in practice.

Our aim is to eliminate such guesswork by assisting designers during multiple phases of development. For example, exploratory analysis can identify unexpected edge cases or weed out language designs incompatible with existing code. We followed the Design Study “Lite” methodology [14] over 7 months to help the developers of a new gradual type system for the R programming language.

The contributions of this ongoing design study are:

- A *task abstraction* for programming language designers analyzing run-time type signatures for type system development.
- *The design and implementation of TYPEICAL*, an interactive visualization of run-time type signatures that supports: filtering data down to interpretable subsets; understanding argument and return types; and comparing type signatures.
- *Initial validation* of our system design with a usability study.

TYPEICAL builds on a data set of run-time type information recorded during the execution of test and example code from the most widely used libraries in the R ecosystem. Our visual design links two well-established visualizations, parallel sets [7] and Treemaps [4] [11], to view and navigate these type traces. While our design study focuses on R, TYPEICAL should be useful for analyzing any language where similar data are available.

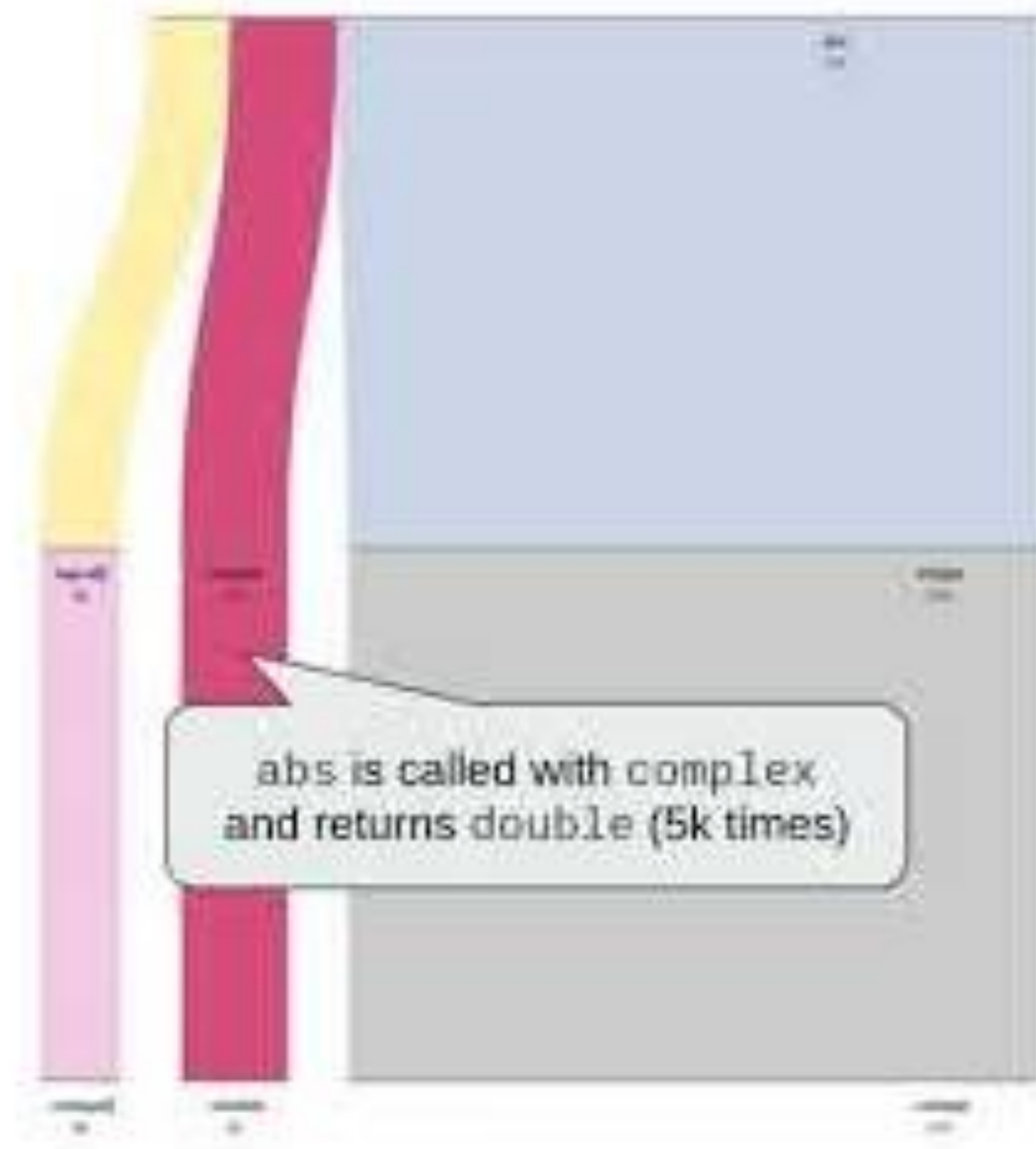
A copy of this paper, source code, and data are available at osf.io/mc6zt, and a demo is online at typeical.github.io

CS 7250 SPRING 2020:
INFORMATION VISUALIZATION:
THEORY AND APPLICATIONS

[Website](#)

[Moy et al. VIS 2020](#)

TYPEical:
A tool for
programming
language designers



PROJECT EXAMPLE — LOCH PROSPECTOR

Loch Prospector: Metadata Visualization for Lakes of Open Data

Neha Makhija ^{ID}* Mansi Jain ^{ID} Nikolaos Tziavelis ^{ID} Laura Di Rocco ^{ID}
Sara Di Bartolomeo ^{ID} Cody Dunne ^{ID}
Northeastern University

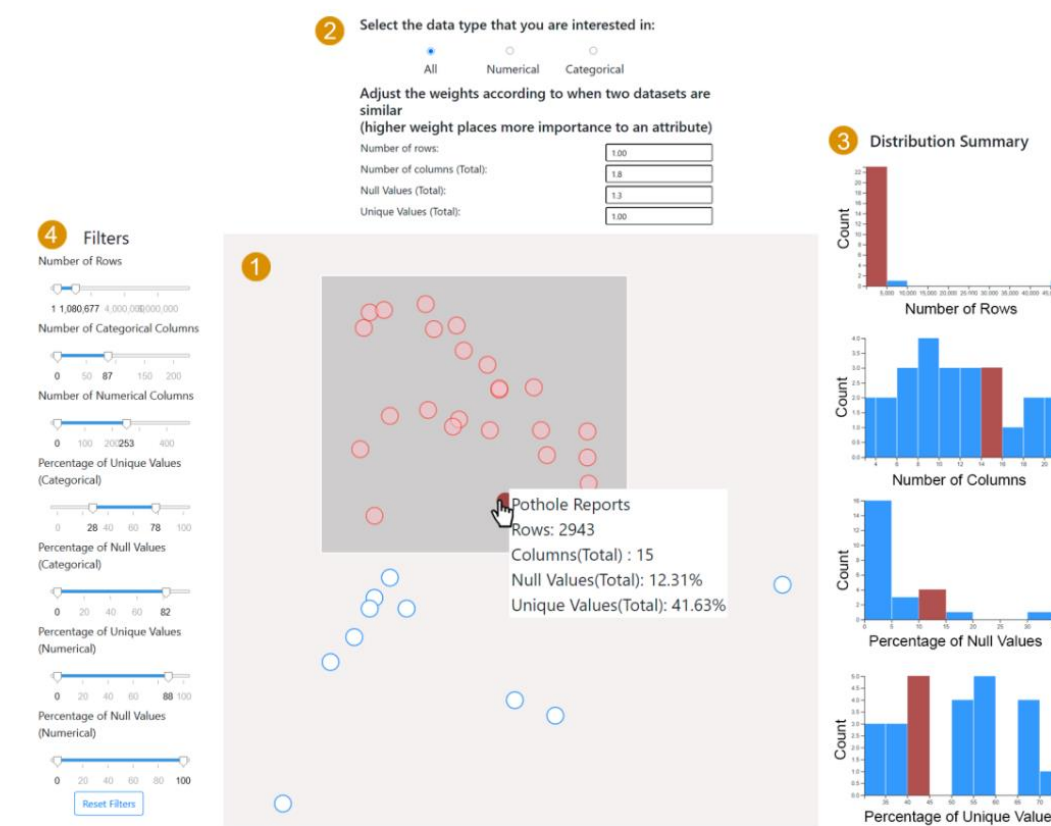


Figure 1: LOCH PROSPECTOR visualizes available datasets in Open Data lakes using four linked components. A multidimensional scaling (MDS) [16] plot 1 shows a point for each dataset, organized spatially by similar metadata characteristics. Weights for the MDS algorithm can be tuned for particular types of metadata using the Visualization Configuration Box 2. Dynamic Filters [2] 4 can be used to explore datasets of interest, with Summary Statistics 3 shown for the currently selected datasets.

ABSTRACT

Data lakes are an emerging storage paradigm that promotes data availability over integration. A prime example are repositories of Open Data which show great promise for transparent data science. Due to the lack of proper integration, Data Lakes may not have a common consistent schema and traditional data management techniques fall short with these repositories. Much recent research has tried to address the new challenges associated with these data lakes. Researchers in this area are mainly interested in the *structural* properties of the data for developing new algorithms, yet typical Open Data portals offer limited functionality in that respect and instead focus on data *semantics*. We propose LOCH PROSPECTOR, a visualization to assist data management researchers in exploring and understanding the most crucial structural aspects of Open Data — in particular,

*Corresponding author. E-mails: [makhija.n | jain.man | tziavelis.n | la.dirocco | dibartolomeo.s | c.dunne]@northeastern.edu

metadata attributes — and the associated task abstraction for their work. Our visualization enables researchers to navigate the contents of data lakes effectively and easily accomplish what were previously laborious tasks. A copy of this paper with all supplemental material is available at osf.io/zkxv9

Index Terms: Human-centered computing—Visualization

1 INTRODUCTION

Recently, the database community has shifted its attention to the data management challenges introduced by data lakes (e.g., [21] [26] [36]). In this paper, we focus on lakes of Open Data [11] [32] due to their prevalent use in data science [19] and by governments and organizations embracing data transparency. Data in these lakes is usually stored in a tabular format but is mainly semi-structured — often as CSV files — due to the dynamic nature of the dataset. Therefore, data in Open Data lakes may lack important *structural information* typically found in a traditional database management system such as column names, data types, and functional dependencies.

Before a researcher is able to develop, optimize, or test algorithms that operate on a lake of Open Data, they must first (1) gain insight into the variation in structural properties and (2) filter to an appropriate subset of the data lake. Understanding the structural properties of data in the lake is key for algorithm design, as these properties directly affect algorithmic operations and performance. E.g., the recommended algorithms for searching, cleaning, and pro-

CS 7250 SPRING 2020:
INFORMATION VISUALIZATION:
THEORY AND APPLICATIONS



PROJECT EXAMPLE — CEREBROVIS

CerebroVis: Designing an Abstract yet Spatially Contextualized Cerebral Artery Network Visualization

Aditeya Pandey, Harsh Shukla, Geoffrey S. Young, Lei Qin, Amir A. Zamani, Liangge Hsu, Raymond Huang, Cody Dunne, and Michelle A. Borkin

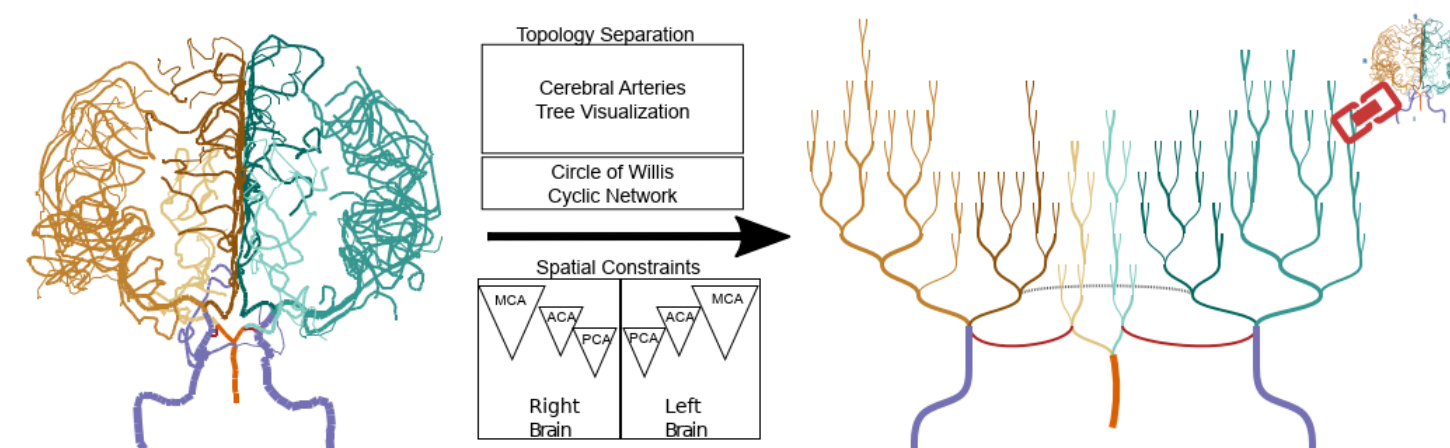


Fig. 1: CerebroVis is a novel network visualization for cerebral arteries. CerebroVis uses an abstract topology-preserving visual design which is put in spatial context by enforcing constraints on the network layout. Here we show the conversion of an almost symmetrical healthy human brain cerebral artery network from a 2D isosurface visualization (left) to CerebroVis (right). Each artery has the same categorical color in both views (see Sec. 3 for a legend).

Abstract—Blood circulation in the human brain is supplied through a network of cerebral arteries. If a clinician suspects a patient has a stroke or other cerebrovascular condition they order imaging tests. Neuroradiologists visually search the resulting scans for abnormalities. Their visual search tasks correspond to the abstract network analysis tasks of browsing and path following. To assist neuroradiologists in identifying cerebral artery abnormalities we designed CerebroVis, a novel abstract—yet spatially contextualized—cerebral artery network visualization. In this design study, we contribute a novel framing and definition of the cerebral artery system in terms of network theory and characterize neuroradiologist domain goals as abstract visualization and network analysis tasks. Through an iterative, user-centered design process we developed an abstract network layout technique which incorporates cerebral artery spatial context. The abstract visualization enables increased domain task performance over 3D geometry representations, while including spatial context helps preserve the user’s mental map of the underlying geometry. We provide open source implementations of our network layout technique and prototype cerebral artery visualization tool. We demonstrate the robustness of our technique by successfully laying out 61 open source brain scans. We evaluate the effectiveness of our layout through a mixed methods study with three neuroradiologists. In a controlled experiment our study participants used CerebroVis and a conventional 3D visualization to examine real cerebral artery imaging data and to identify a simulated intracranial artery stenosis. Participants were more accurate at identifying stenoses using CerebroVis (odds ratio 2.5, absolute risk difference 13%). More broadly, we discuss the applications of our design approach to a general design paradigm we call *Abstraction with Context*. A free copy of this paper, the evaluation stimuli and data, and source code are available at osf.io/e5sxt

Index Terms—Network Visualization, Spatial Context, Abstract Design, Flow Network, Medical Imaging, Cerebral Arteries.

1 INTRODUCTION

Arteries in the human brain form a network of blood flow, and a blockage or leakage in this network can lead to life-threatening cerebrovascular conditions such as a stroke or aneurysm. Strokes alone are the fifth leading cause of death as well as a leading cause of serious long-term disability in the United States, and is globally the second leading cause of death after heart disease [32]. Early detection and diagnosis of these conditions is essential for effective life-saving treatment. Conventional diagnostics

rely on an expert neuroradiologist identifying vascular abnormalities through examination of medical images (e.g., CTA, MRA). This data is commonly rendered in 3D in order to assist the doctor with identification of the abnormalities. However, prior research indicates that existing representations of the 3D cerebral arteries—e.g., isosurface, volume rendering, and Maximum Intensity Projection (MIPS)—introduce visual artifacts and task performance challenges such as overplotting/occlusion [19], false impression of geometry [19], and excessive artery bends.

In this design study, we present a novel 2D visualization of the cerebral artery system designed to assist doctors in the identification of cerebrovascular abnormalities. Inspired by existing visualization research which has demonstrated the effectiveness of 2D representations for spatial search tasks in other medical imaging cases, e.g., cardiovascular arteries [6] and connectomics [33], we present a novel 2D abstract representation of the cerebral arteries. To our knowledge, this is the first attempt to approach the cerebrovascular diagnostics tasks faced by neuroradiologists from the perspective of network science and using an abstract 2D visual encoding.

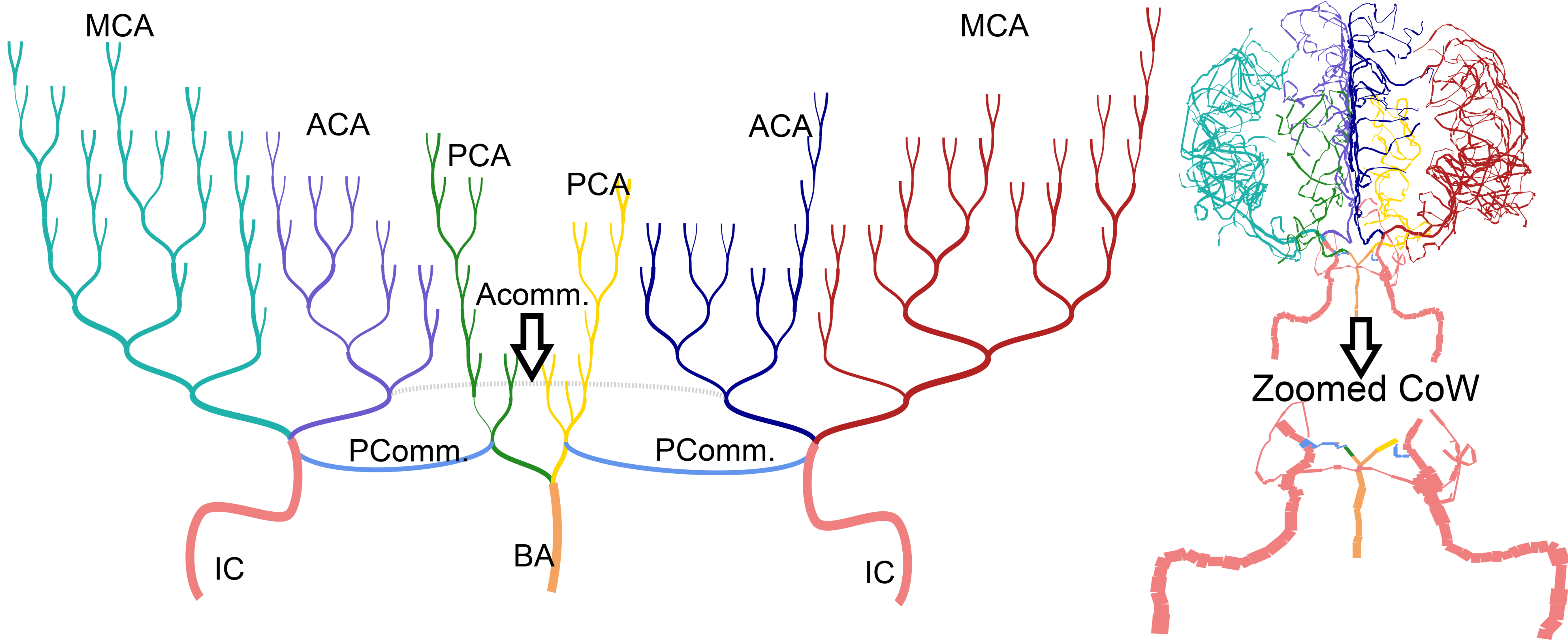
In this paper we first offer a novel framing of cerebral arteries using network theory. Next, we characterize the domain goals and present them as network analysis tasks. In an iterative user-centered design with

- Aditeya Pandey, Harsh Shukla, Cody Dunne, and Michelle Borkin are with Northeastern University. E-mail: {pandey.ad@husky.neu.edu, shukla.h@husky.neu.edu, c.dunne@northeastern.edu, m.borkin@northeastern.edu}
- Geoffrey S. Young, Amir A. Zamani, Liangge Hsu, and Raymond Huang are with Brigham and Women’s Hospital. E-mail: {gyoung@bwh.harvard.edu, azamani@bwh.harvard.edu, lhsu1@bwh.harvard.edu, ryhuang@bwh.harvard.edu}
- Lei Qin is with the Dana-Farber Cancer Institute. E-mail: lqin2@partners.org

Manuscript received xx.xxx. 201x; accepted xx.xxx. 201x. Date of Publication xx.xxx. 201x; date of current version xx.xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxx

CS 7260 FALL 2017:
VISUALIZATION FOR
NETWORK SCIENCE

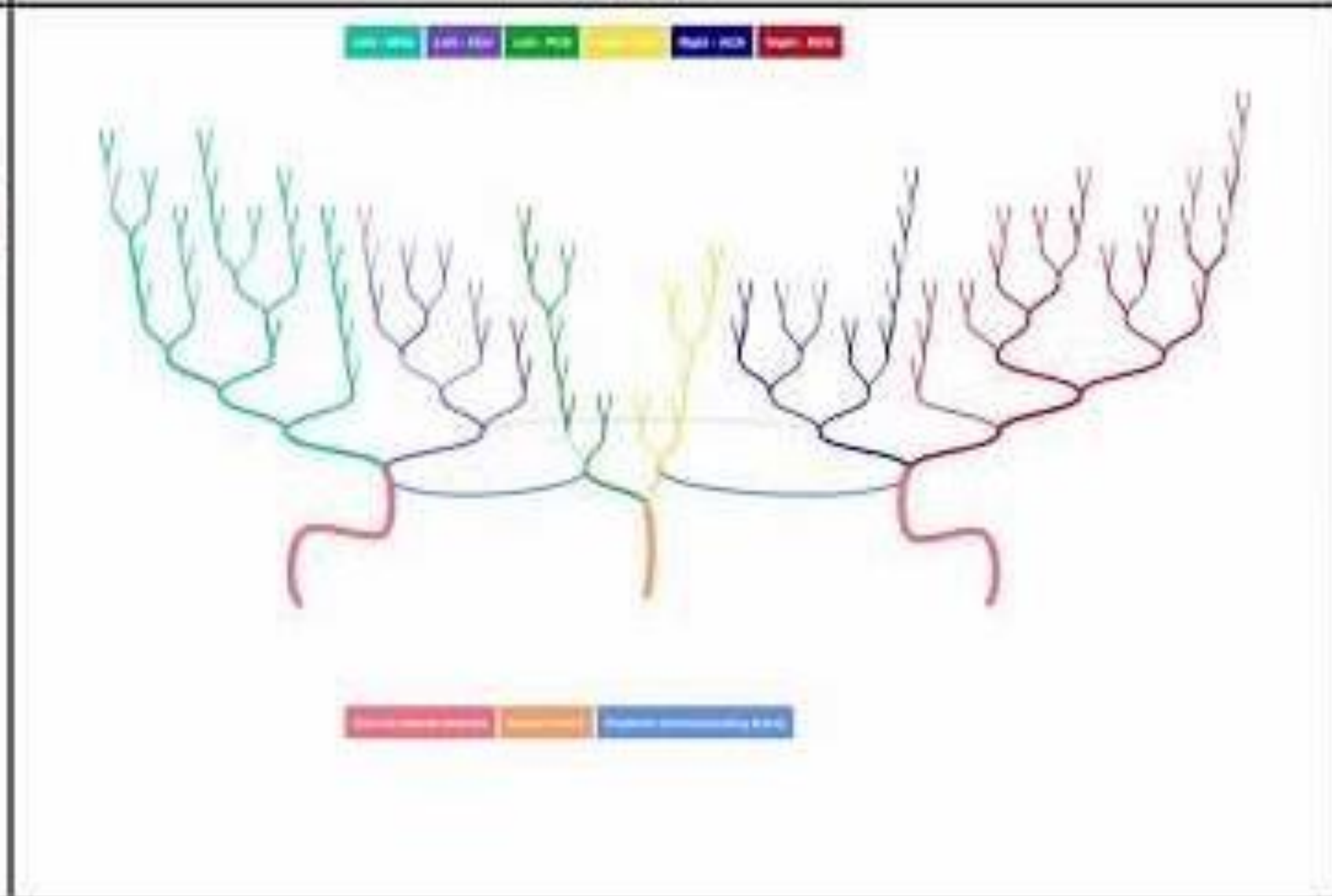
PROJECT EXAMPLE — CEREBROVIS



Brain Surface Projection



CerebroVis




EXAMPLE OF A SUCCESSFUL *DIFFERENTIATED* COURSE PROJECT

(Requires prior instructor approval to waive / alter requirements)

PROJECT EXAMPLE — DIVERSIFORM TIMELINES

Evaluating the Effect of Timeline Shape on Visualization Task Performance

Sara Di Bartolomeo  Aditeya Pandey  Aristotelis Leventidis 
David Saffo  , Uzma Haque Syeda  , Elin Carstensdottir 
Magy Seif El-Nasr  , Michelle Borkin  , Cody Dunne 

Northeastern University

(dibartolomeo.s | pandey.ad | saffo.d | syeda.u)@husky.neu.edu, elin@ccs.neu.edu,
(magy | m.borkin | c.dunne)@northeastern.edu

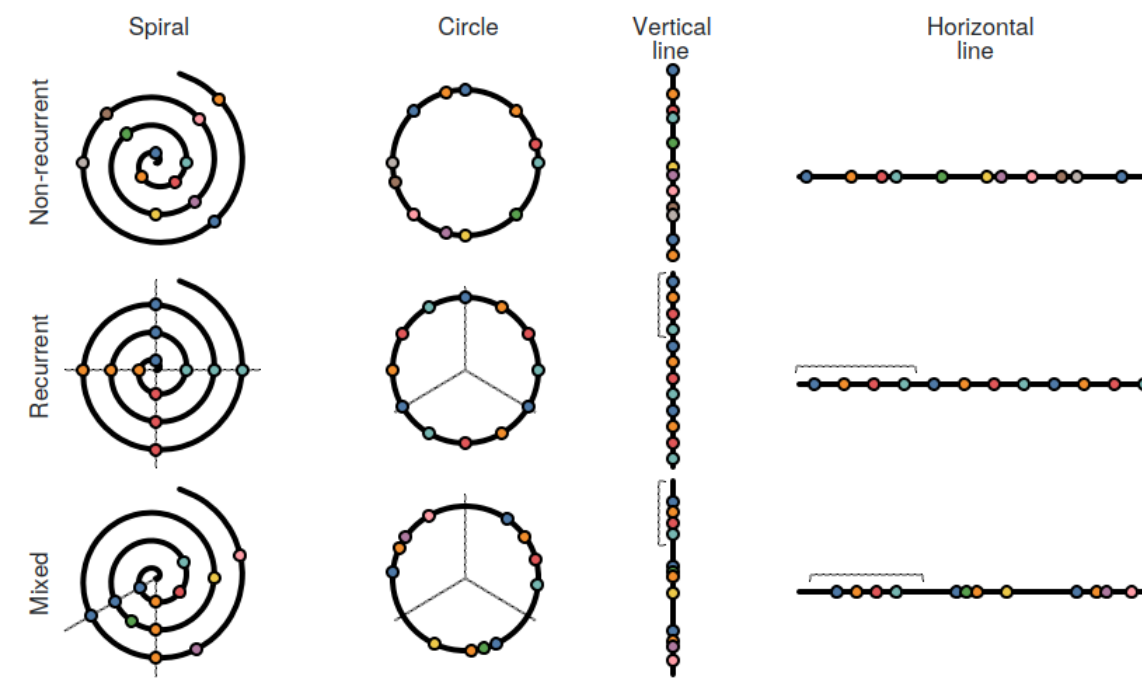


Figure 1. We evaluate the effect on task performance of four timeline shapes (left to right) across three types of temporal event sequence data (top to bottom). The images are simplified versions of the stimuli that we used in our experiment. Each dot on a timeline represents an event and has a specific categorical color to highlight where the dataset has recurrent events. Dashed lines highlight the recurrent intervals or a set of recurrent events.

ABSTRACT

Timelines are commonly represented on a horizontal line, which is not necessarily the most effective way to visualize temporal event sequences. However, few experiments have evaluated how timeline shape influences task performance. We present the design and results of a controlled experiment run on Amazon Mechanical Turk ($n = 192$) in which we evaluate how timeline shape affects task completion time, correctness, and user preference. We tested 12 combinations of four shapes — horizontal line, vertical line, circle, and spiral — and three data types — recurrent, non-recurrent, and mixed event sequences. We found good evidence that timeline shape meaningfully affects user task completion time but not correctness and that

users have a strong shape preference. Building on our results, we present design guidelines for creating effective timeline visualizations based on user task and data types. A free copy of this paper, the evaluation stimuli and data, and code are available at <https://osf.io/qr5yu/>

Author Keywords

Timelines; Temporal Event Sequences; Information Visualization; Controlled Experiments

CCS Concepts

•Human-centered computing → Human computer interaction (HCI); Visualization design and evaluation; Information visualization;

INTRODUCTION

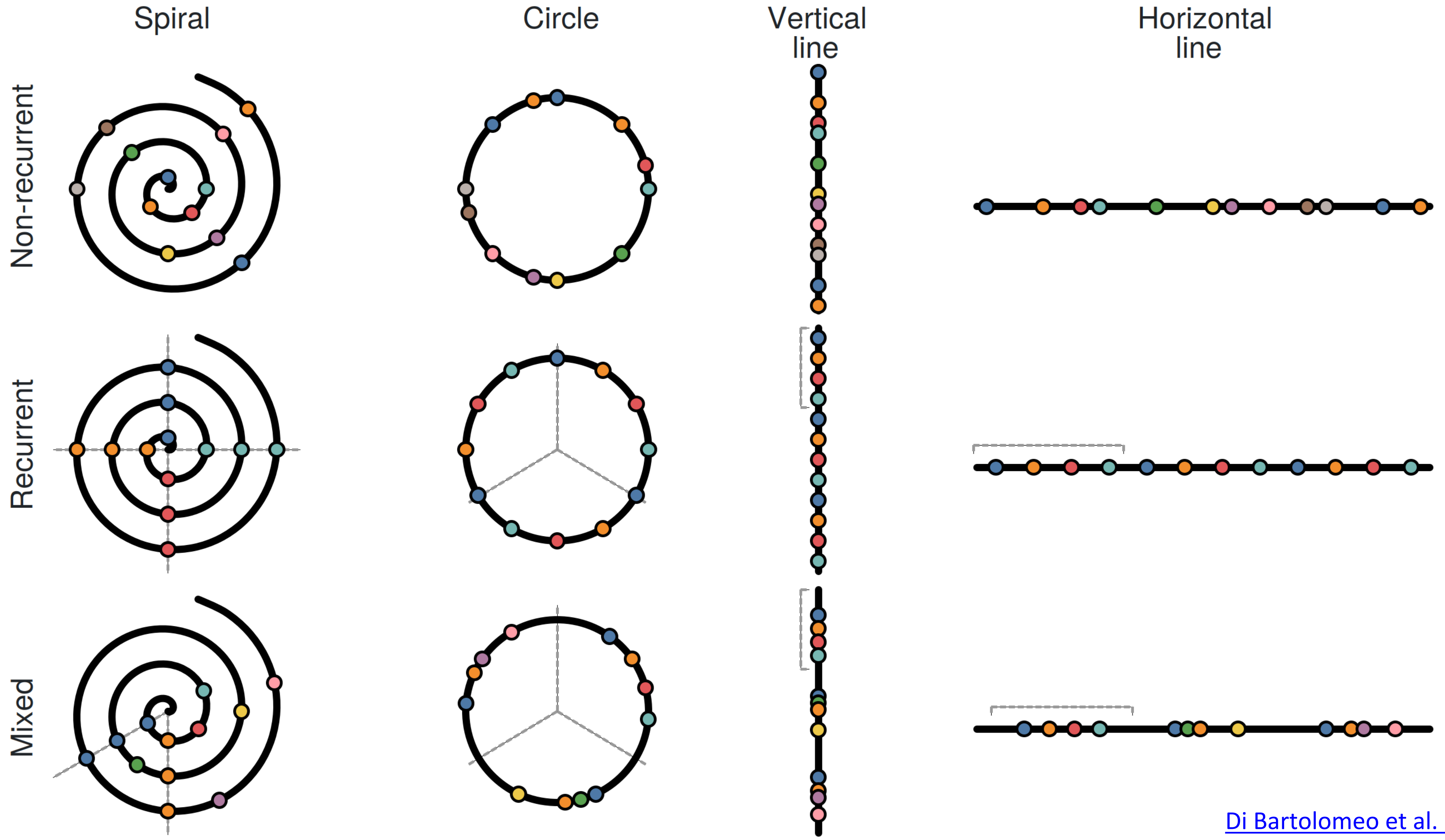
A timeline is a visual representation of a series of events in time. The use of timelines dates back to 17th century [32] when Joseph Priestly designed a visualization that showed the rise and fall of empires in Europe's history. In the modern era, timelines have become prevalent in our daily lives as the de facto representation to show financial trends, weather

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
CHI '20, April 25–30, 2020, Honolulu, HI, USA.
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-6708-0/20/04 ...\$15.00.
<http://dx.doi.org/10.1145/3313831.3376237>

CS 7340 FALL 2018:
THEORY AND METHODS
IN HUMAN COMPUTER
INTERACTION

[Di Bartolomeo et al. CHI 2020](#)

PROJECT EXAMPLE — DIVERSIFORM TIMELINES





PROJECT IDEAS:

VIS + X

Where X = a CS subfield (ML | SEC | NLP | HCC | GAM | NS | SYS | ...)

OR

Where X = a domain application (health, energy, transportation, astronomy, crime...)

POTENTIAL VENUE: IEEE VIS 2021 SHORT PAPERS

Deadline ~2021-06-13

Short Paper Call for Participation

IEEE Vis 2020 solicits submissions in a short paper format. Short papers draw from the same paper types and topics as full papers of VAST, Infovis and Scavis, ranging from theoretical to applied research contributions.

Short papers welcome submissions describing original work with focused and concise research contributions, incremental work such as follow-up extensions or evaluations of existing methods, or exploratory work.

The short paper submission deadline is June 13, 2020 **July 13, 2020**, creating an opportunity to showcase late-breaking research results.

Submission Guidelines:

Short papers can be up to four pages plus up to one page of references. Submissions will be in the VGTC conference two-column format, in line with the [IEEE VIS formatting guidelines](#).

To submit a short paper, go to the [PCS system](#) and select "VGTC > VIS 2020 > VIS 2020 Short Papers" to set up a submission.

Publication Details:

Accepted short papers will be presented at the VIS conference. They will also be included in the conference proceedings and published in IEEE Xplore and assigned a DOI.

Authors of accepted short papers may subsequently submit to TVCG a full paper on the same topic, consistent with the [TVCG standard](#) requiring that papers previously published in conference proceedings make additional substantial scientific contributions that build upon the material of the conference publication or include this material as a part of a larger research effort. The authors are encouraged to make mention of this in their cover letter, provide a copy of the previous conference submission and an outline of the added contributions. Providing these materials will help TVCG reviewers to better gauge the scientific value of the work and defuse any possible misconceptions.

Concurrent submissions of short papers and posters are not possible.

Important Dates (deadline extension):

- Submission Deadline: June 13, 2020 **July 13, 2020** (no separate abstract deadline)
- Notification: July 18, 2020 **August 18, 2020**
- Submission of Camera-Ready Final Version: August 1, 2020 **August 25, 2020**

All deadlines are at **5:00pm Pacific Time (PDT)**.

Vis Short Paper Co-Chairs:

- Enrico Bertini, *NYU Tandon School of Engineering, USA*
- Roxana Bujack, *Los Alamos National Laboratory, USA*
- Christopher Collins, *Ontario Tech University, Canada*
- Wenwen Dou, *University of North Carolina at Charlotte, USA*
- Alexander Lex, *University of Utah, USA*
- Timo Ropinski, *Ulm University, Germany*

Email: vis_shortpapers@ieeewis.org

SUBMIT YOUR WORK

[Papers](#)

[Short Papers](#)

[Posters](#)

[Tutorials](#)

[Workshops](#)

[Panels](#)

[Doctoral Colloquium](#)

[Application Spotlights](#)

SUBMISSION PROCESS

[Submission Information](#)

[Submission Keywords](#)

[Review Process](#)

[Open Practices](#)

PARTICIPATING VIRTUALLY

[Speakers and Presenters](#)

[Video Previews](#)

[Recording your Talk](#)

[Guidance for Virtual Events](#)

Problems with this webpage? Contact vis_shortpapers@ieeewis.org, [file a bug](#), or [suggest a fix](#).



THE PREMIER FORUM FOR ADVANCES IN VISUALIZATION AND VISUAL ANALYTICS

PAST YEARS ARCHIVE

2019 · 2018 · 2017 · 2016 · 2015 · 2014 · 2013
2012 · 2011 · 2010 · 2009 · 2008 · 2007 · 2006
2005 · 2004 · 2003 · 2002 · 2001 · 2000 · 1999
1998 · 1997 · 1996

SOCIAL MEDIA

[Facebook](#)

[Twitter](#)

PROJECTS

In-class project pitches: M 2020-09-30
What questions do you have for me?

Upcoming Assignments & Communication

<https://c.dunne.dev/ds4200f20>

Look at the upcoming assignments and deadlines (**12:01am, moving to 11:59pm previous day**)!

- Textbook, Readings & Reading Quizzes
- Due W 2020-09-23: Assignment **2a,b**
- Due W 2020-09-30: Project **1**

Everyday Required Supplies:

- 5+ colors of pen/pencil
- White paper
- Laptop and charger

Use **Canvas Discussions** for general questions, **email** the **instructor & TAs** for questions specific to you.