

# Visualizing textual collation

Ronald Haentjens Dekker  
and  
David J. Birnbaum

Presented at Balisage 2024

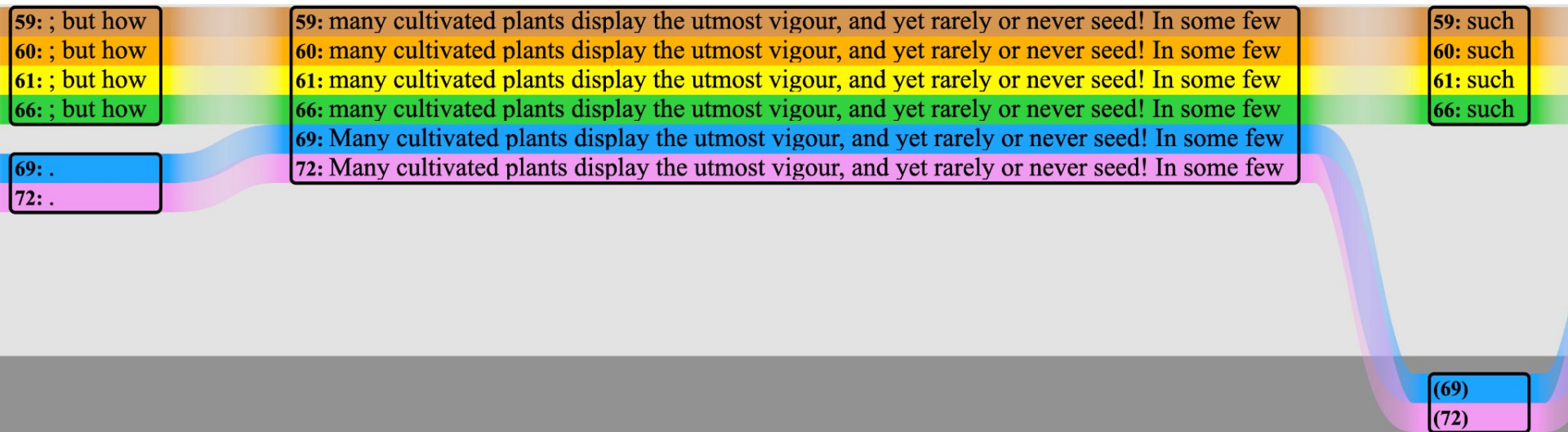
# Presentation overview

**The goal:** What is an alignment ribbon and what story does it tell?

## **The journey**

- Alignment ribbon introduction
- Part 1: About textual collation
- Part 2: Common alignment visualizations, textual and graphic
- (Intermission)
- Part 3: Implementing an alignment ribbon in SVG
- Conclusions
  - About visualization
  - About the alignment ribbon

# Alignment ribbon visualization



## Parts 1 and 2: Why

Imagining an effective alignment visualization

# Part 1: About textual collation

- Why textual scholars care about collation
- Three challenges of collation ... and one more
- The Gothenburg Model of textual collation

# Why textual scholars care about collation

- Coming up with a theory of the text
- Analyzing the transmission
- Reconstruction of the original version
- Study of the creative process (genetic criticism)
- Creation of a reading version

# Three challenges of collation ... and one more

1. Repetition
2. Transposition
3. Witness count and scalability

**Bonus challenge:** order effects and order-independent multiple-witness alignment

# The Gothenburg model of textual collation

1. Tokenization
2. Normalization
3. Alignment
4. Analysis (manual and machine-assisted)
5. Visualization (no obligatory base text)



# Part 2: Modeling and visualizing alignment

## Textual visualizations

- Critical apparatus
- Alignment table

## Graphic visualizations

- Rhine Delta / Variant graph
- TRAViz and other enhancements
- Alluvial flow
- Storyline visualization

# Critical apparatus

δὲ ἐγέννησεν τὸν Ἰωσαφάτ, Ἰωσαφάτ δὲ ἐγέννησεν τὸν  
Ἰωράμ, Ἰωράμ δὲ ἐγέννησεν τὸν Ὀζίαν, 9 Ὀζίας δὲ  
ἐγέννησεν τὸν Ἰωαθάμ, Ἰωαθάμ δὲ ἐγέννησεν τὸν Ἀχάζ,  
Ἀχάζ δὲ ἐγέννησεν τὸν Ἑζεκίαν, 10 Ἑζεκίας δὲ ἐγέν-  
νησεν τὸν Μανασσῆ, Μανασσῆς δὲ ἐγέννησεν τὸν Ἀμώς,  
Ἀμώς<sup>2</sup> δὲ ἐγέννησεν τὸν Ἰωσίαν, 11 Ἰωσίας δὲ ἐγέννη-

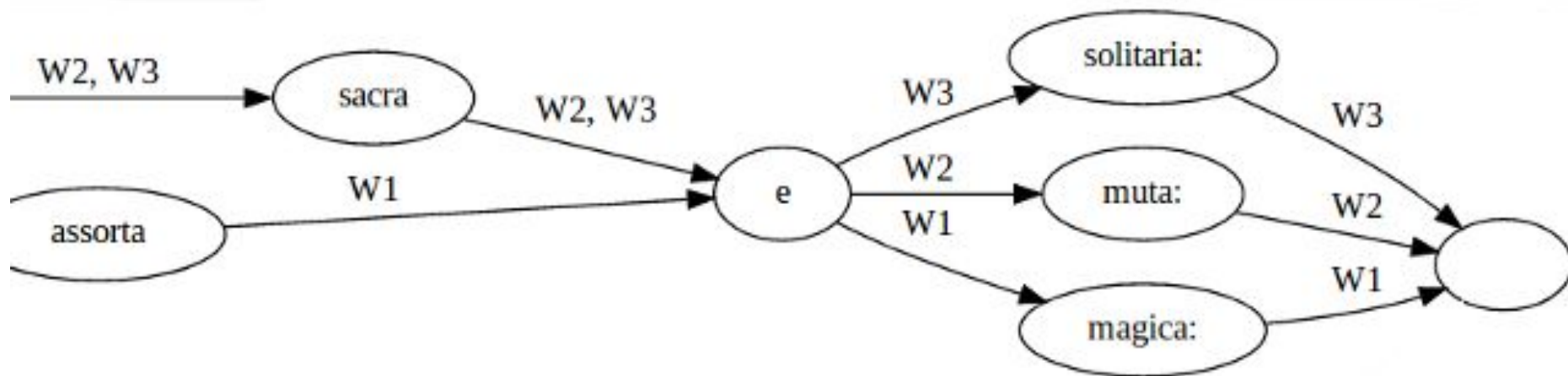
<sup>1</sup> 7-8 {B} Ἀσάφ, Ἀσάφ <sup>p<sup>lvid</sup></sup> Ⲛ B C (D<sup>Luke</sup>) *f*<sup>1</sup> *f*<sup>13</sup> 700 1071 *l*<sup>185m</sup> <sup>pt</sup>  
*it*<sup>aur,c,d</sup> *Luke*<sup>g<sup>1</sup>,k,q</sup> *syr*<sup>hmg</sup> *cop*<sup>sa,bo</sup> *arm* *eth* *geo* (Epiphanius) // Ἀσά, Ἀσά  
K I W Δ Π 28 33 565 892 1009 1010 1079 1195 1216 1230 1241 1242 1365 1546  
(2148 Ἀσσά) *Byz* *Lect*<sup>m</sup> *l*<sup>185m</sup> <sup>pt</sup> *it*<sup>a,f,ff<sup>1</sup></sup> *vg* *syr*<sup>c,s,p,h,pal</sup> Epiphanius Augustine

<sup>2</sup> 10 {B} Ἀμώς, Ἀμώς Ⲛ B C (D<sup>Luke</sup>) Δ Θ Π\* *f*<sup>1</sup> 33 1071 1079 1546  
*l*<sup>1627m</sup> *it*<sup>c,d</sup> *Luke*<sup>ff<sup>1</sup>,g<sup>1</sup>,k,q</sup> *cop*<sup>sa,bo,fay</sup> *arm* *eth* Athanasius Epiphanius // Ἀμών,  
Ἀμών K I W Π<sup>2</sup> *f*<sup>13</sup> 28 565 (700 892 1195 Ἀμμών, Ἀμμών) 1009 1010 1216

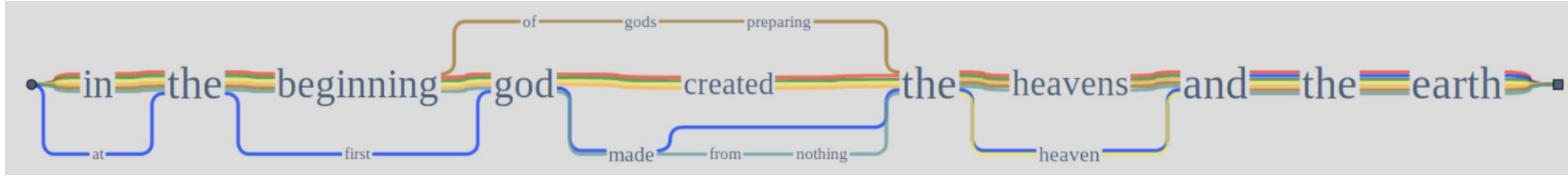
## Alignment table

<b>1859</b>	The	result	of	the	various,	quite	unknown,
<b>1860</b>	The	result	of	the	various,	quite	unknown,
<b>1861</b>	The	result	of	the	various,	quite	unknown,
<b>1866</b>	The	result	of	the	various,	quite	unknown,
<b>1869</b>	The	results	of	the	various,		unknown,
<b>1872</b>	The	results	of	the	various,		unknown,

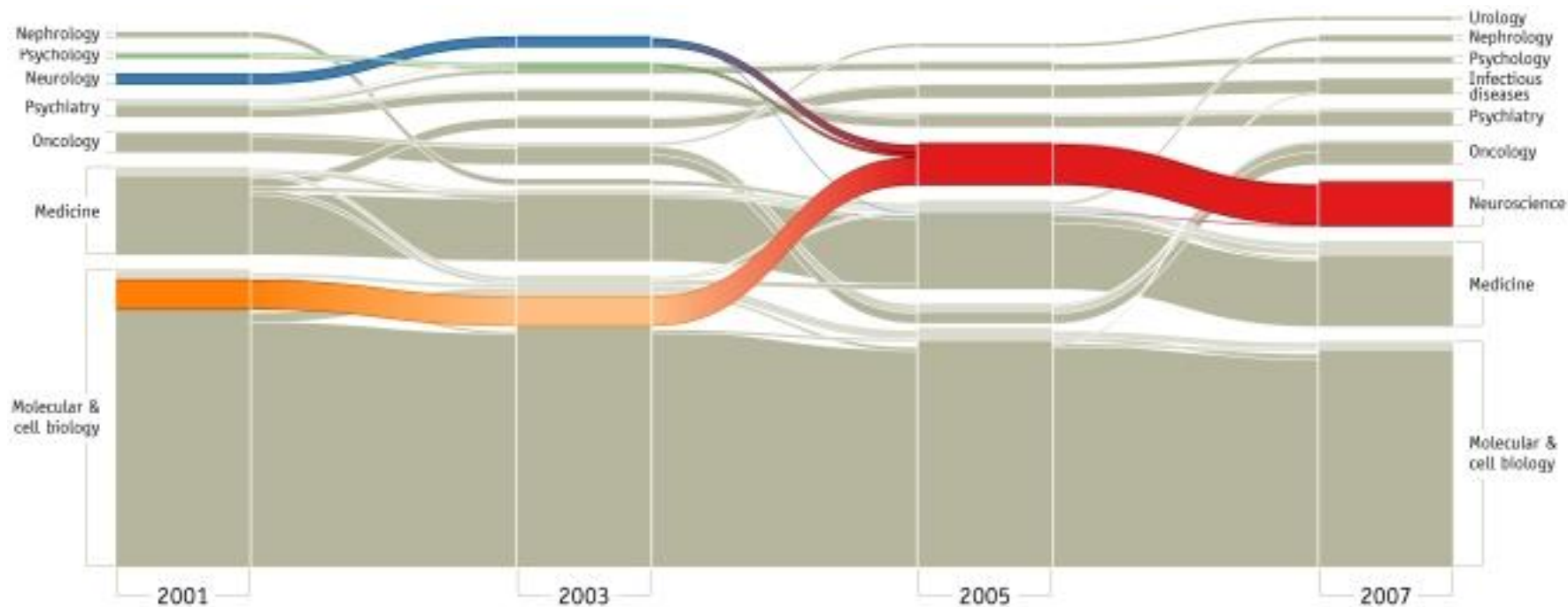
## Rhine Delta / Variant graph



# TRAViz and other enhancements

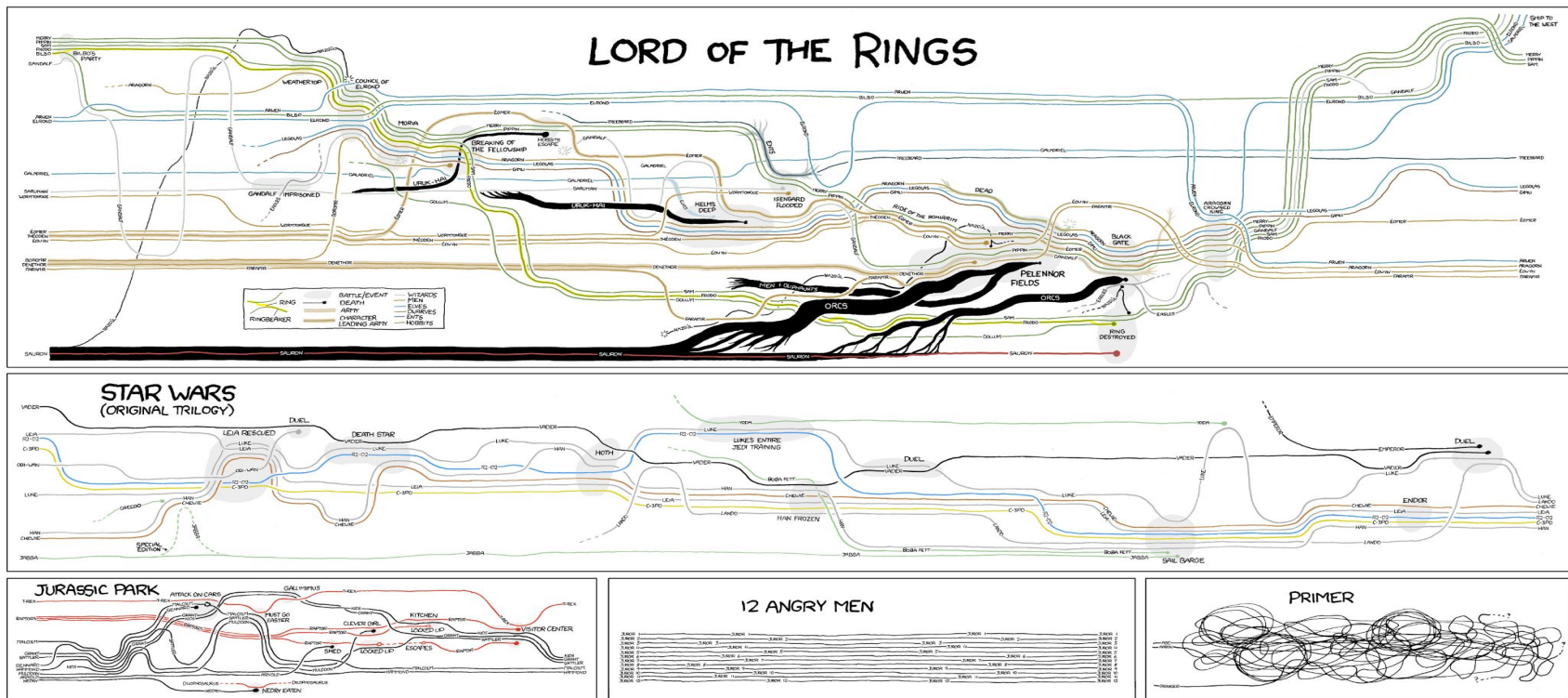


# Alluvial flow



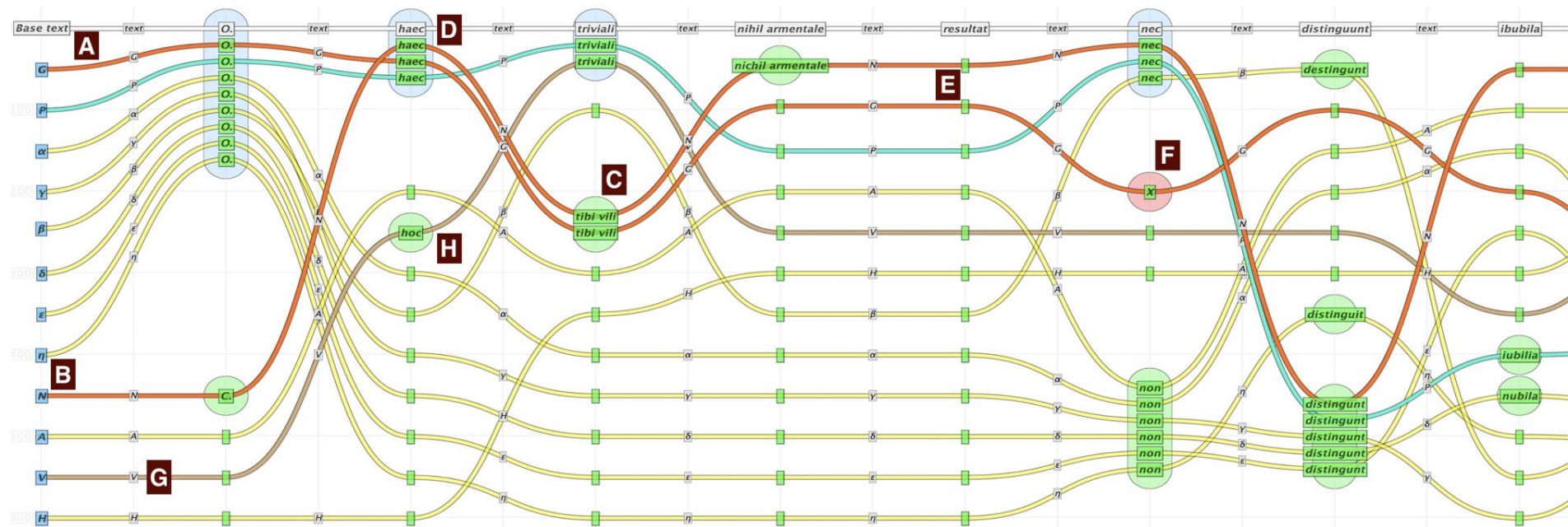
# Storyline and screenplays

THESE CHARTS SHOW MOVIE CHARACTER INTERACTIONS.  
THE HORIZONTAL AXIS IS TIME. THE VERTICAL GROUPING OF THE  
LINES INDICATES WHICH CHARACTERS ARE TOGETHER AT A GIVEN TIME.



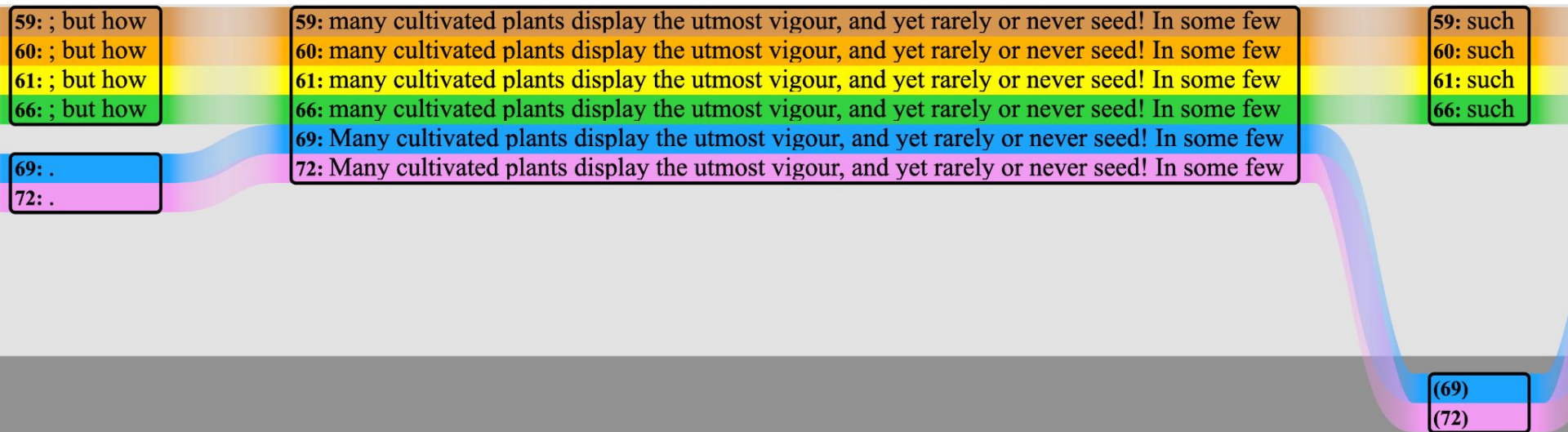


## Storyline and textual alignment





# Alignment ribbon visualization



# Intermission



# Visualizing textual collation

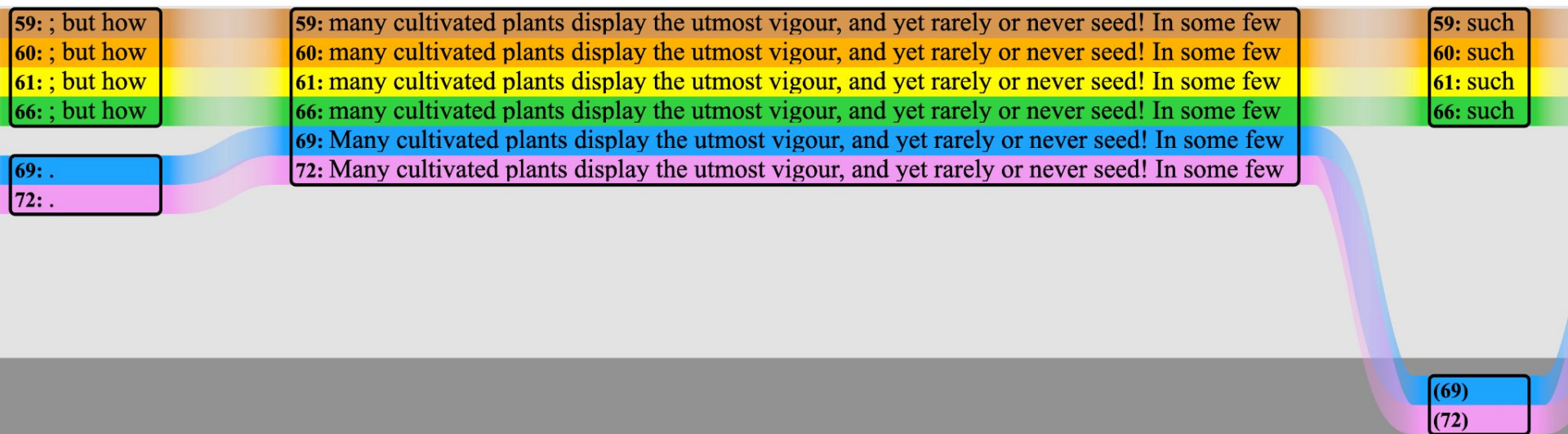
David J. Birnbaum  
and  
Ronald Haentjens Dekker

Presented at Balisage 2024

## Part 3: How

Expressing a dynamic  
alignment ribbon in SVG

## Part 3: Alignment ribbon visualization



# Alignment ribbon roots

- Alignment table
  - Reading view of full text of all witnesses
- Rhine Delta / variant graph
  - Changing patterns of agreement and variation
- Alluvial flow
  - Changing shape of entire witness set
- Storyline
  - Movement of individual witness through changing patterns of agreement

# Implementation overview

- Language is Scala
- Input is plain text
  - ... but could include markup
- Output is HTML5 (CSS, JavaScript) with embedded SVG
  - ... not just a single SVG document

# Implementation challenges and methods: overview

## 1. Text length

- Rectangles must match length of longest reading in alignment point

## 2. Flows

- Witness flow must appear continuous across alignment points

## 3. Z-index

- SVG elements must overlay one another correctly

## 4. Dynamic interactivity

- Must support user-mediated truncation (including ellipsis points)



# Text length

<character	dec="65"	hex="0x41"	name="A"	str="A"	width="11.456"/>
<character	dec="66"	hex="0x42"	name="B"	str="B"	width="10.672"/>
<character	dec="67"	hex="0x43"	name="C"	str="C"	width="10.672"/>
<character	dec="68"	hex="0x44"	name="D"	str="D"	width="11.552"/>
<character	dec="69"	hex="0x45"	name="E"	str="E"	width="9.776"/>
<character	dec="70"	hex="0x46"	name="F"	str="F"	width="8.896"/>
<character	dec="71"	hex="0x47"	name="G"	str="G"	width="11.552"/>
<character	dec="72"	hex="0x48"	name="H"	str="H"	width="11.552"/>
<character	dec="73"	hex="0x49"	name="I"	str="I"	width="5.328"/>

Birnbaum and Taylor 2021, “How long is my SVG <text> element?”

<https://balisage.net/Proceedings/vol26/html/Birnbaum01/BalisageVol26-Birnbaum01.html>

# Compute text length

1. `string-join($tokens, " ") !` (: fuse into one string :)  
`string-to-codepoints(.) !` (: sequence of Unicode codepoints (integers) :)  
`codepoints-to-string(.) !` (: sequence of individual Unicode characters :)  
`local:compute-length(.) =>` (: sequence of lengths of individual characters :)  
`sum()` (: total length :)
2. `$tokens !`  
`local:compute-length(.) =>` (: memoize; sequence of token lengths :)  
`sum()` (: total length :)
3. `fold-left(($tokens, 0, function ($x, $y) {local:compute-length($y) + $x})`









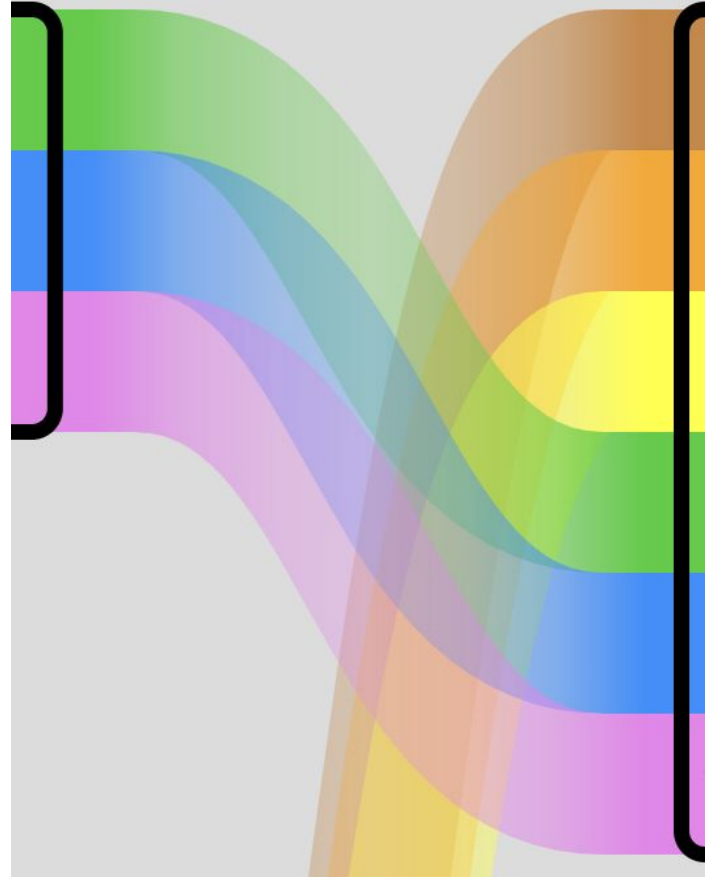
# Flows

1. Shape: <path>
2. Stroke:
  - a. Color gradient
  - b. The straight gradient surprise
3. Fill: The fill surprise
4. Opacity: Tuning the opacity and the background
5. Bonus: vector-effect="non-scaling-stroke"

# Flows: Shape

```
<path d="M 0,234.0  
      L 10,234.0  
      C 40,234.0 40,9.0001 70,9.0001  
      L 80,9.0001"  
      stroke="url(#peruGradient)"  
      stroke-width="18" fill="none"/>
```

- Straight lines (M, L) at both ends
- Cubic Bézier curve (symmetrical)





# Flows: Stroke and gradient

```
<path d="..." stroke="url(#peruGradient)" stroke-width="18" fill="none"/>
```

```
<linearGradient id="peruGradient" x1="0%" x2="100%"  
  y1="0%" y2="0%">
```

```
<stop offset="0%" stop-color="peru" stop-opacity="1"/>
```

```
<stop offset="6%" stop-color="peru" stop-opacity="1"/>
```

```
<stop offset="20%" stop-color="peru" stop-opacity=".6"/>
```

```
<stop offset="35%" stop-color="peru" stop-opacity=".4"/>
```

```
<stop offset="50%" stop-color="peru" stop-opacity=".3"/>
```

```
<stop offset="65%" stop-color="peru" stop-opacity=".4"/>
```

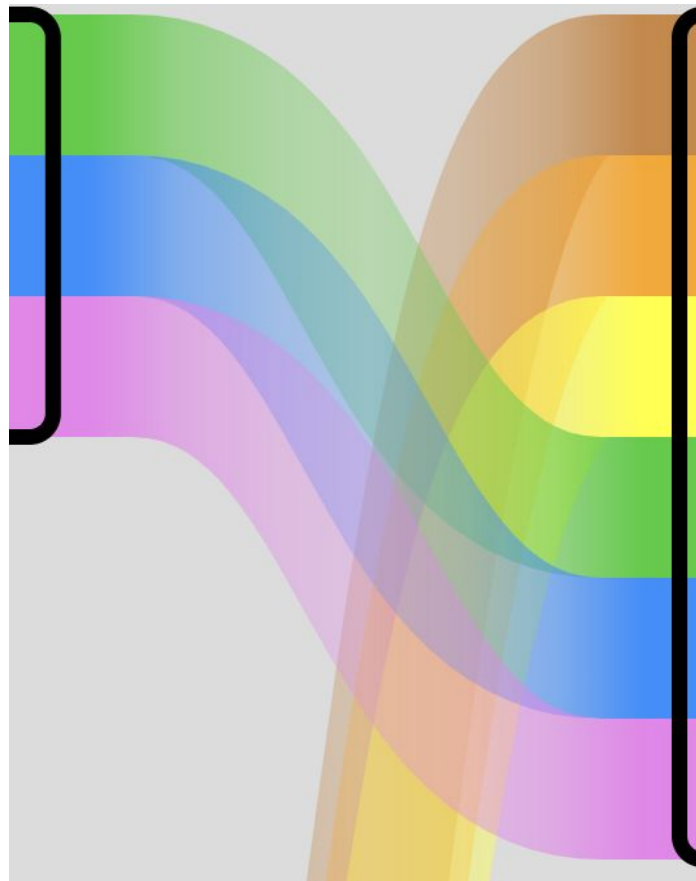
```
<stop offset="80%" stop-color="peru" stop-opacity=".6"/>
```

```
<stop offset="94%" stop-color="peru" stop-opacity="1"/>
```

```
<stop offset="100%" stop-color="peru" stop-opacity="1"/>
```

```
</linearGradient>
```

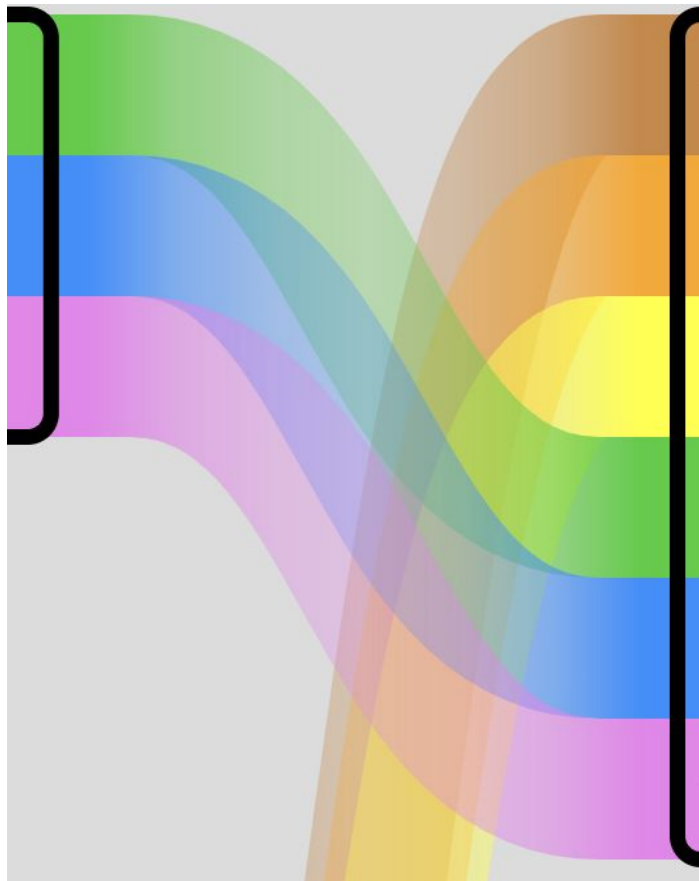
- Solid (100%) at both ends
- Shade into 30% in middle



# Flows: Gradient surprise

```
<path d="M 0,9.0  
      L 10,9.0  
      C 40,9.0 40,9.0001 70,9.0001  
      L 80,9.0001"  
      stroke="url(#peruGradient)" stroke-width="18"  
      fill="none"/>
```

- Gradient requires width
- Gradient not applied to straight line (or path) because
  - Straight line has no bounding box and ...
  - ... therefore no width ...
  - ... because stroke-width doesn't count as width!
- To fix: Add .0001 to right edge of all flows so that line won't be straight



# Flows: Fill surprise

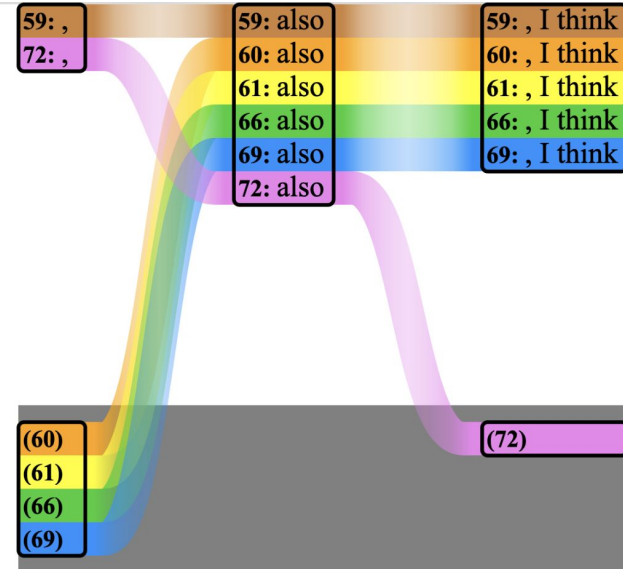
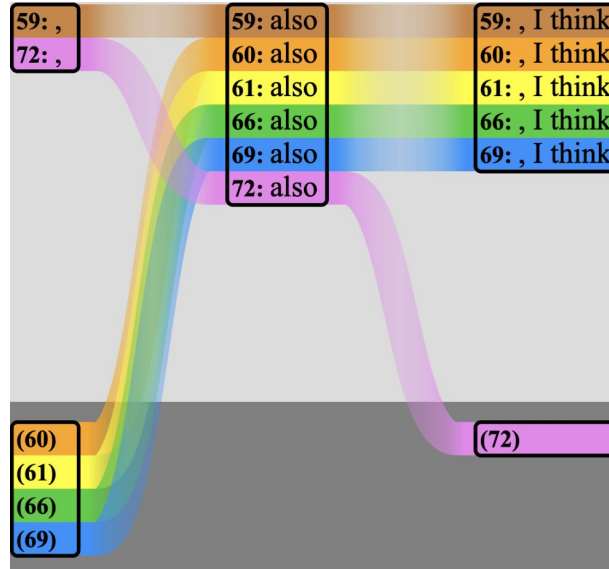
```
<path d="M 0,9.0  
      L 10,9.0  
      C 40,9.0 40,9.0001 70,9.0001  
      L 80,9.0001"  
      stroke="url(#greenGradient)"  
      stroke-width="18"  
      fill="none"/>
```

- <path> elements are implicitly connected
- And therefore automatically filled
- Default @fill value is "black"
- Fix: specify fill="none"

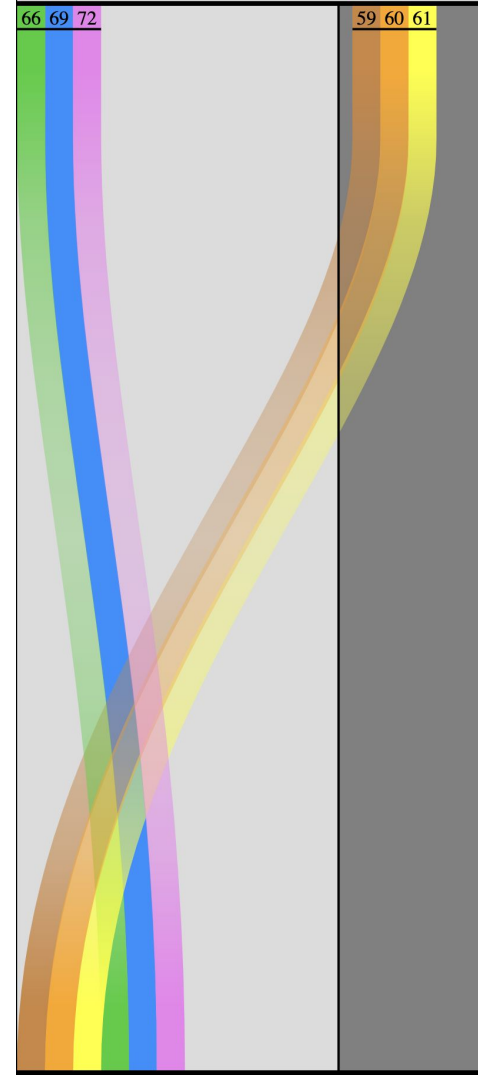
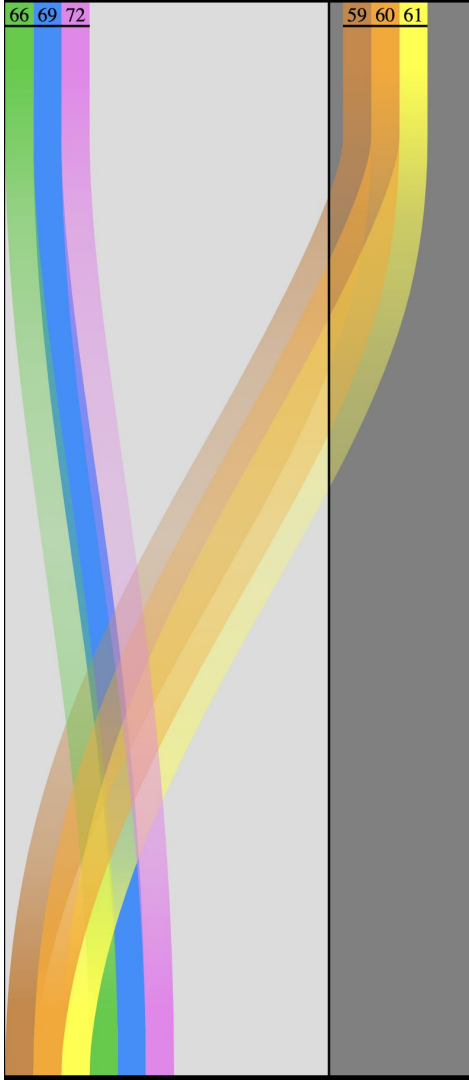


# Flows: Opacity and background

- 30% opacity effective for overlap ...
- ... but bad for non-overlap
- Fix: non-white background



vector-effect=  
"non-scaling-stroke"



z-index

59: WHEN we

60: WHEN we

61: WHEN we

66: WHEN we

69: WHEN we

72: WHEN we

59: WHEN we

60: WHEN we

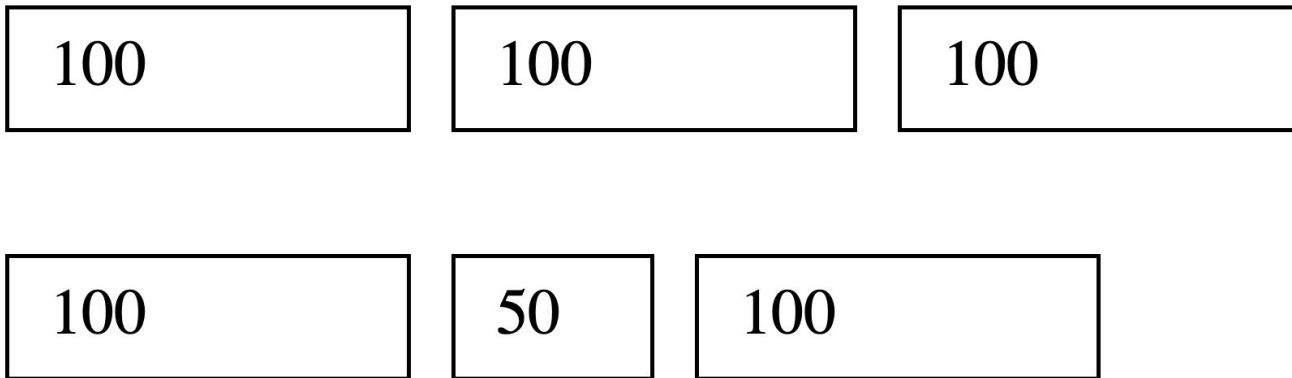
61: WHEN we

66: WHEN we

69: WHEN we

72: WHEN we

# Flexbox



```
<section>
```

```
  <div class="w100">100</div>
```

```
  <div class="w100">100</div>
```

```
  <div class="w100">100</div>
```

```
</section>
```

```
<section>
```

```
  <div class="w100">100</div>
```

```
  <div class="w50">50</div>
```

```
  <div class="w100">100</div>
```

```
</section>
```

```
section {  
  display: flex;  
  flex-direction: row;  
  gap: 10px;  
  padding: 5px 10px;  
  margin-bottom: 20px;  
}
```

```
div {  
  border: 1px solid black;  
  box-sizing: border-box;  
  padding: 5px 10px;  
}
```

```
.w100 {  
  width: 100px;  
}  
.w50 {  
  width: 50px;  
}
```

# Conclusions: about visualization

- Visualizations tell a story about data
- Effectiveness of any visualization results from emphasizing some features and excluding others
- There is no single best visualization



# Conclusions: about the alignment ribbon

- The alignment ribbon shares features with:
  - Other general flow-type visualizations
    - Alluvial fan
    - Storyline
    - (cf Sankey, parallel coordinates)
  - Rhine Delta / variant graph
  - Alignment table
- The alignment ribbon implementation employs:
  - Fragmented SVG
  - Dynamic resizing
  - CSS flexbox

Thank you for your attention!

Need more collation? Join us for a Birds of a Feather session this afternoon!