

AnnoGram

An Annotative Grammar of Graphics Extension

Md Dilshadur Rahman · **Rahat Zaman** · Andrew McNutt · Paul Rosen

SCI Institute · University of Utah

Annotations are essential for understanding visualizations — yet they remain afterthoughts in most tools.

Annotations in Visualization

What is an Annotation?

Process that:

- adds **highlights** or **context**
- to a **complete visualization**
- for **emphasis** or **improvement**

Annotations are essential for understanding visualizations — yet they remain afterthoughts in most tools.

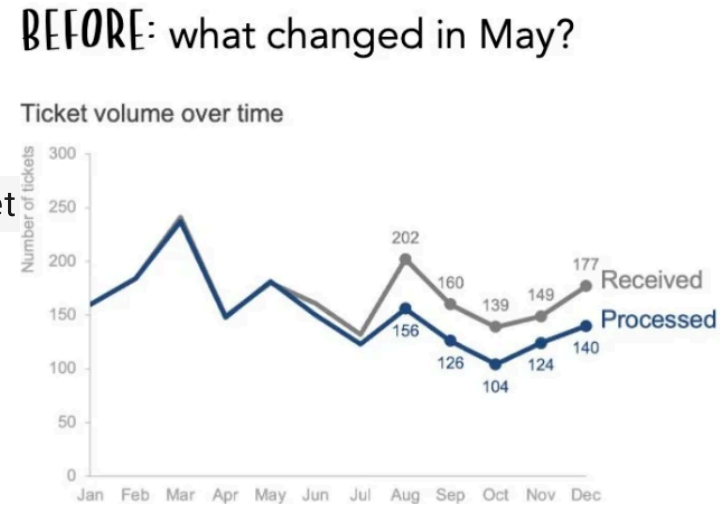
Annotations in Visualization

What is an Annotation?

Process that:

- adds **highlights** or **context**
- to a **complete visualization**
- for **emphasis** or **improvement**

Annotations are essential for understanding visualizations — yet



Annotations in Visualization

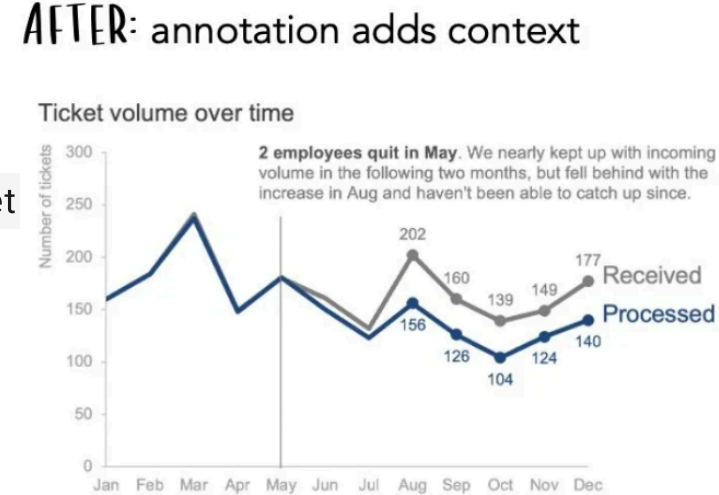
What is an Annotation?

Process that:

- adds **highlights** or **context**
- to a **complete visualization**
- for **emphasis** or **improvement**

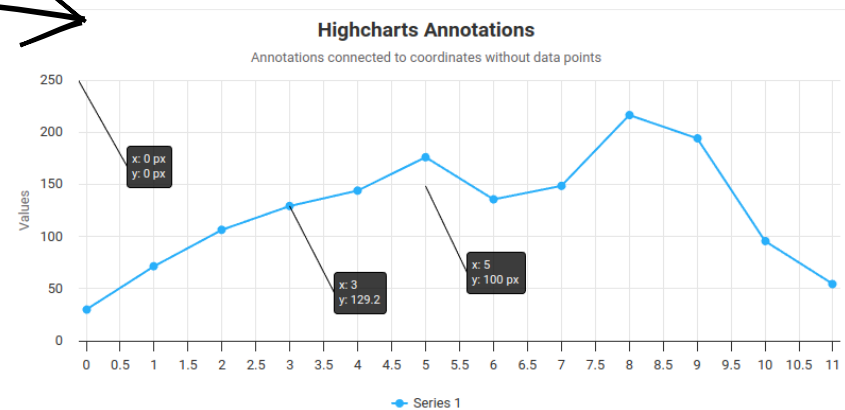
Annotations are essential for understanding visualizations — yet

- Help **explain, contextualize, and guide** visual interpretation
- Improve **memorability** and **comprehension**



HighChart.js

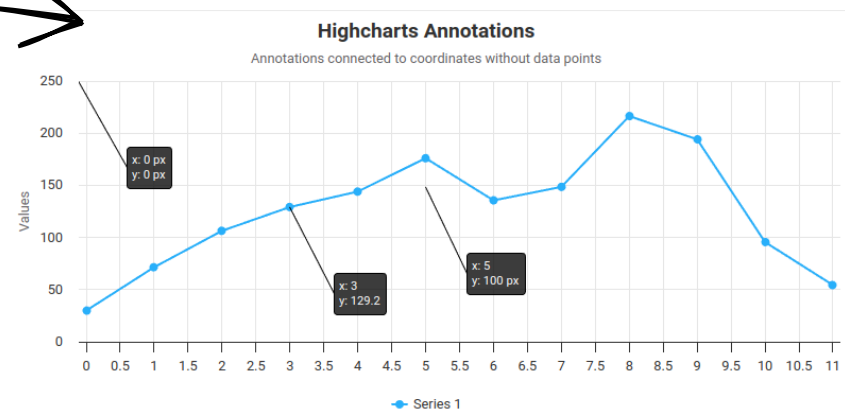
```
chart: { type: 'line' },
series: [{ data: [ ... ] }],
annotations: [{
  labels: [{
    point: { x: 3, y: 129.2 }
  }, {
    point: { x: 0, y: 0 }
  }, {
    point: { x: 5, y: 100 }
  }]
}]
```



Problem 1: existing grammars treat annotations as *secondary* constructs

HighChart.js

```
chart: { type: 'line' },
series: [{ data: [ ... ] }],
annotations: [{
  labels: [{
    point: { x: 3, y: 129.2 }
  }, {
    point: { x: 0, y: 0 }
  }, {
    point: { x: 5, y: 100 }
  }]
}]
```

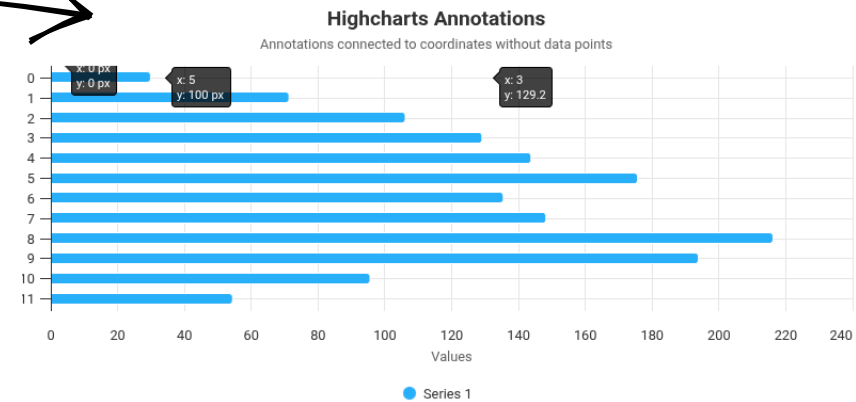


Problem 1: existing grammars treat annotations as *secondary* constructs

HighChart.js

```
chart: { type: 'bar' },
series: [{ data: [ ... ] }],
annotations: [{
  labels: [{
    point: { x: 3, y: 129.2 }
  }, {
    point: { x: 0, y: 0 }
  }, {
    point: { x: 5, y: 100 }
  }]
}]
```

Problem 2: no portability of annotations across visualizations



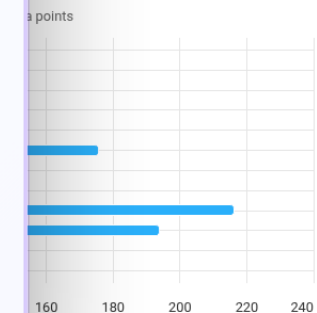
Problem 1
annotatio

ions

```
chart: { type:
series: [{ dat
annotations: [
  labels: [{
    point: { x
  }, {
    point: { x
  }, {
    point: { x
  }]
}]
```

Key Problems of Annotation Support

- Only text support
- No binding with data
- No portability across visualizations



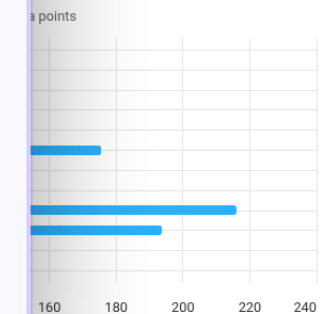
Problem 1
annotatio

ions

```
chart: { type:
series: [{ dat
annotations: [
  labels: [{
    point: { x
  }, {
    point: { x
  }, {
    point: { x
  }]
}]
```

Key Problems of Annotation Support

- Only text support
- No binding with data
- No portability across visualizations



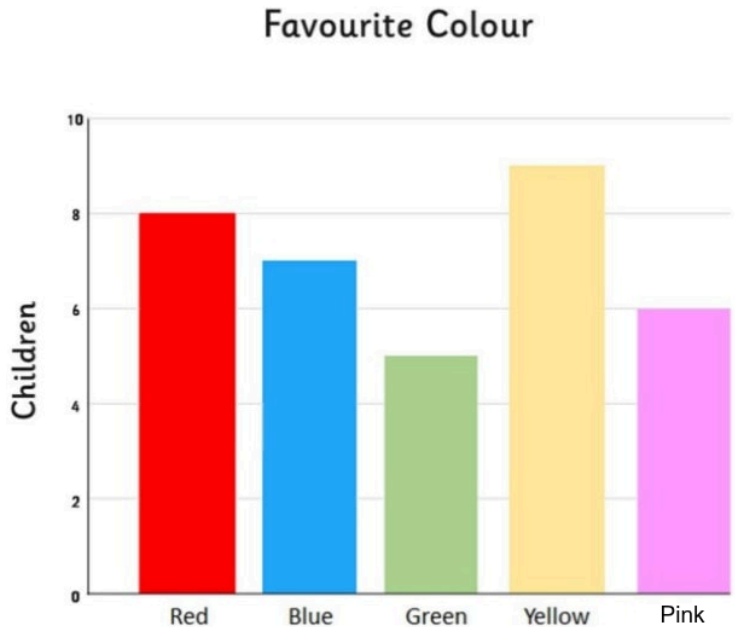
These problems exist across popular visualization libraries

Design a grammar of graphics extension that follows an annotation taxonomy and makes them portable across various chart types.

Annotation Grammar

annogram.grammar

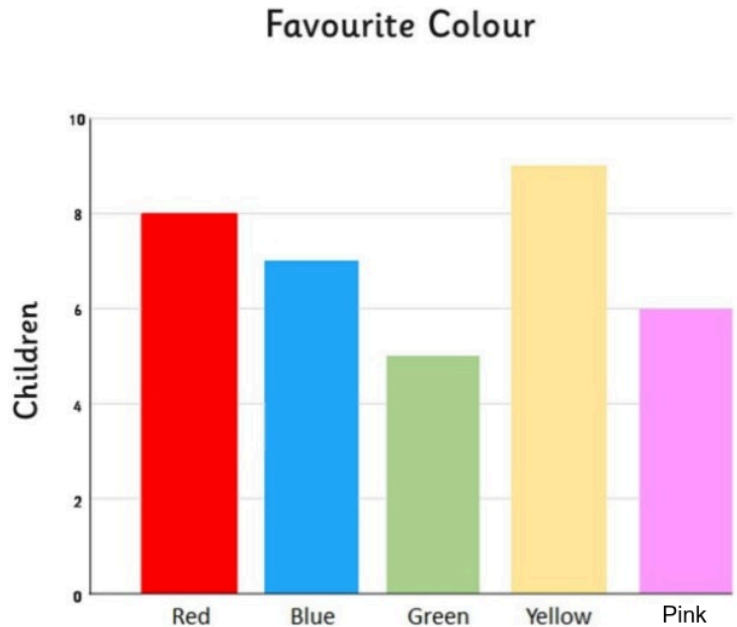
```
annotation := [
  target := (
    |id
    |FixedPos
    |ChartPart
    |DataPoint
    |Axis
    |None
  )[]
]
```



Annotation Grammar

annogram.grammar

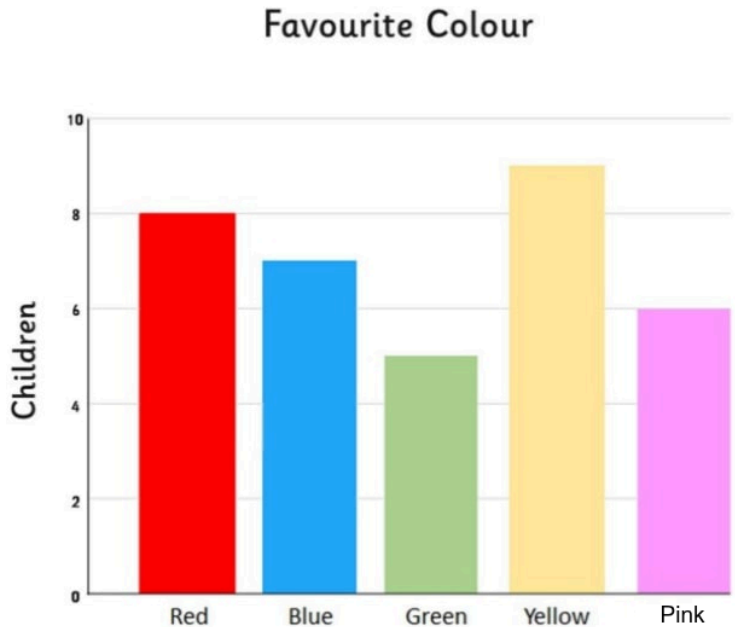
```
annotation := [
  target := (
    |id
    |FixedPos
    |ChartPart
    |DataPoint
    |Axis
    |None
  )[]
]
```



Annotation Grammar

annogram.grammar

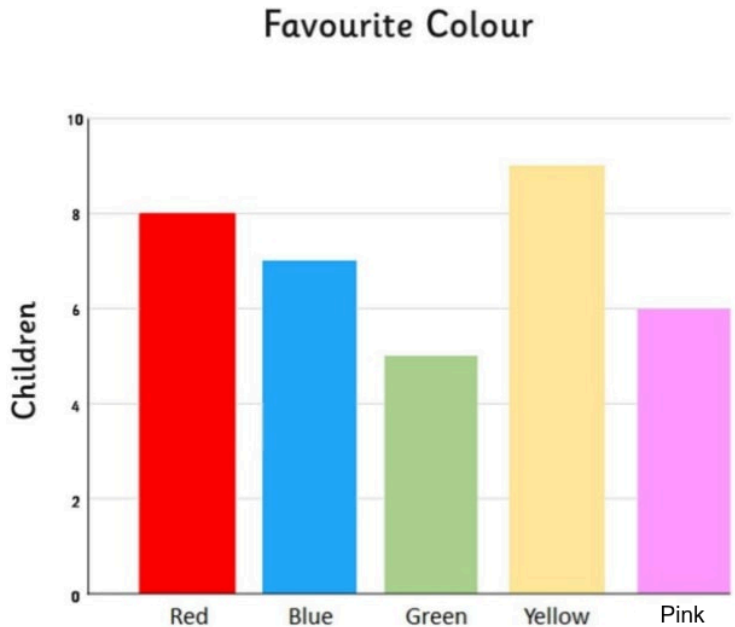
```
annotation := [
  target := (
    |id
    |FixedPos
    |ChartPart
    |DataPoint
    |Axis
    |None
  )[]
]
```



Annotation Grammar

annogram.grammar

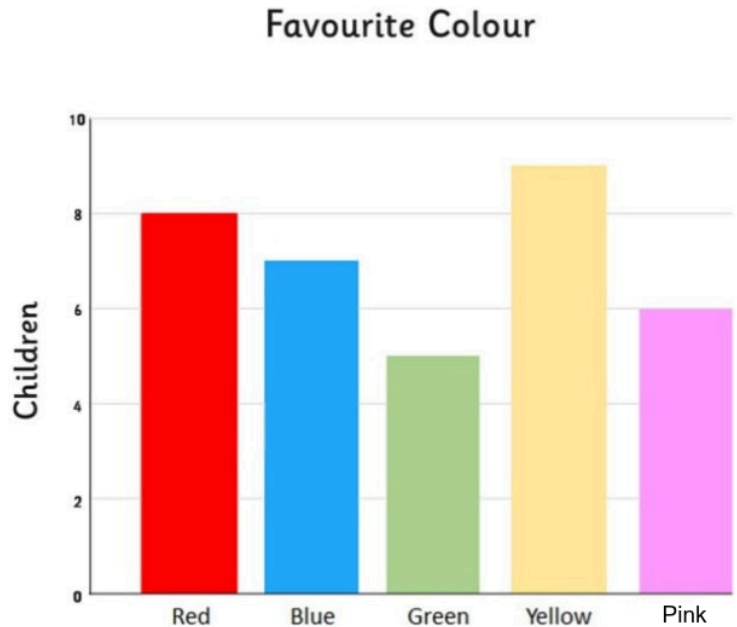
```
annotation := [
  target := (
    |id
    |FixedPos
    |ChartPart
    |DataPoint
    |Axis
    |None
  )[]
]
```



Annotation Grammar

annogram.grammar

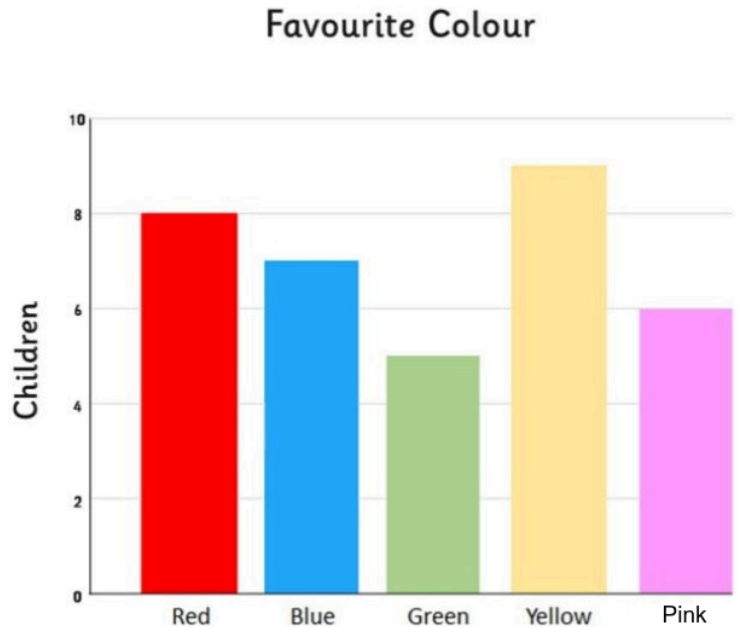
```
annotation := [
  target := (
    |id
    |FixedPos
    |ChartPart
    |DataPoint
    |Axis
    |None
  )[]
]
```



Annotation Grammar

annogram.grammar

```
annotation := [
  target := (
    |id
    |FixedPos
    |ChartPart
    |DataPoint
    |Axis
    |None
  )[]
]
```



Annotation Grammar

annogram.grammar

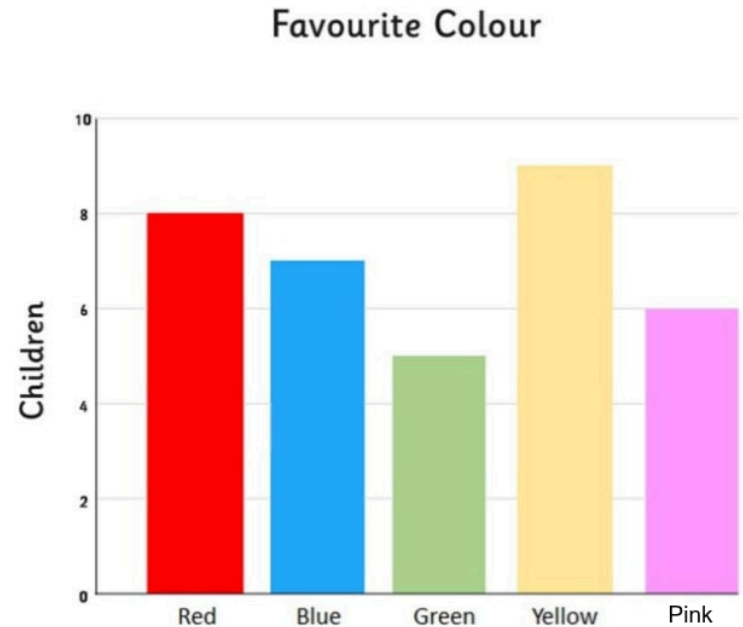
```
annotation := [
  target := (id|FixedPos|ChartPart|DataPoint|Axis|None)[]
]
```

Annotation Targets

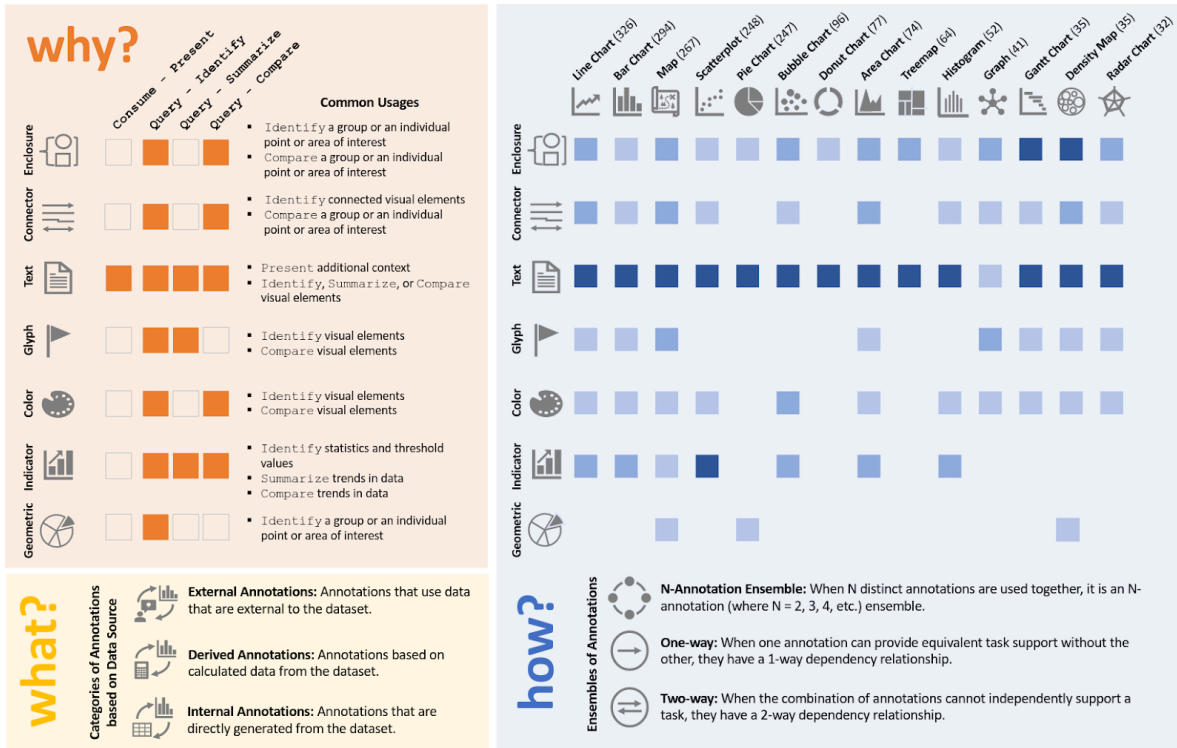
DataPoint := *Expression* | `index[]`

ChartPart := `title` | `legend` | `subtitle` | `caption` | ⋯

Axis := { `axis` := `x` | `y`, `parts` := `label` | `tick` | `grid` | `tick-label`; `range` $\overset{\text{opt}}{:=}$ **Coord**[] | *Expression* }



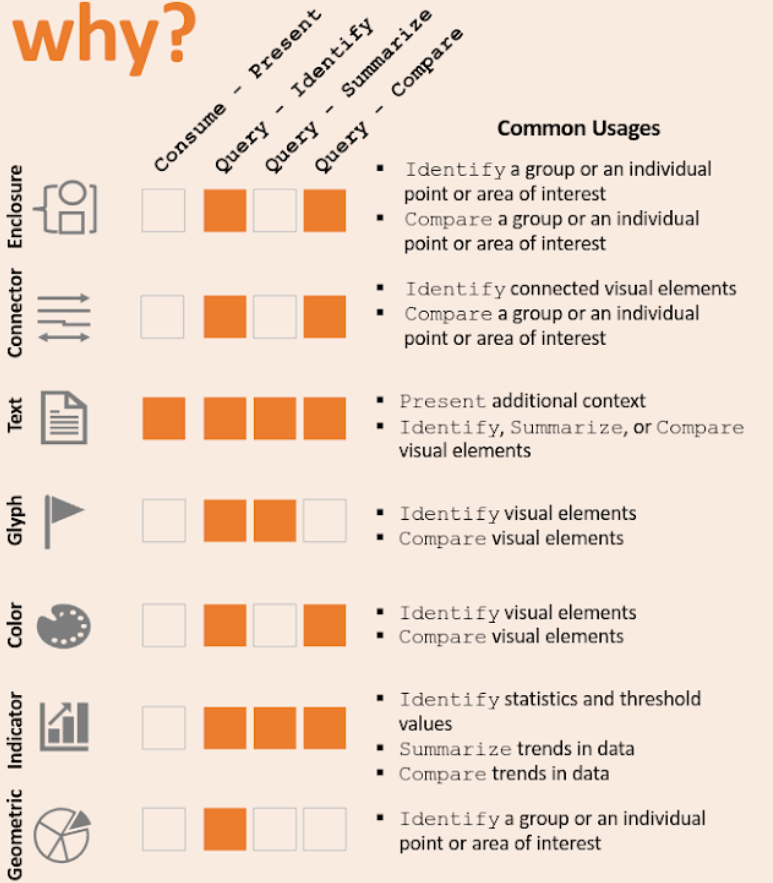
Annotation Taxonomy



M. D. Rahman, B. Doppalapudi, G. J. Quadri, and P. Rosen.

A survey on annotations in information visualization: Empirical insights, applications, and challenges.

why?





hat?

of Annotations
Data Source

External Annotations: Annotations that use data that are external to the dataset.

Derived Annotations: Annotations based on

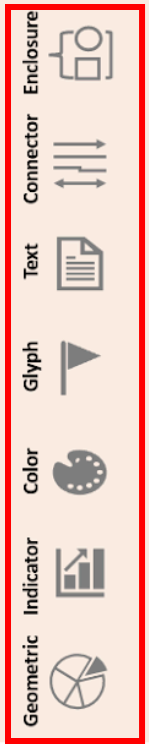
ow?

of Annotations

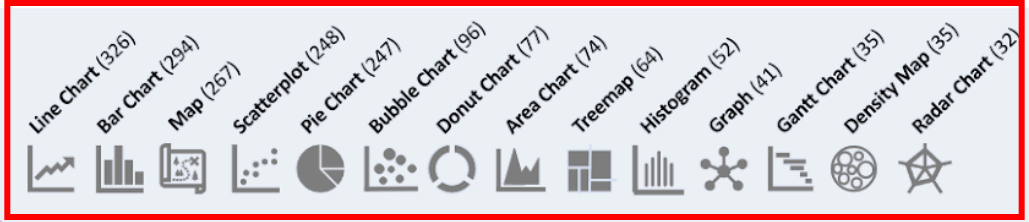
N-Annotation Ensemble: When N distinct annotations are used together, it is an N-annotation (where N = 2, 3, 4, etc.) ensemble.

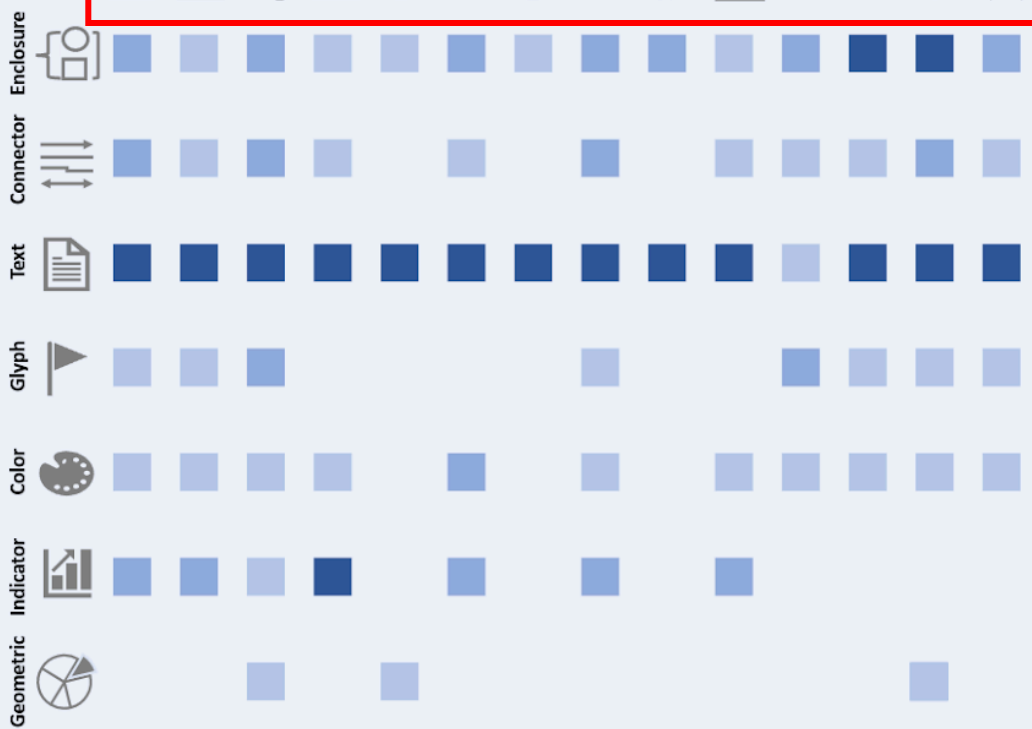
One-way: When one annotation can provide equivalent task support without the other, they have a 1-way dependency relationship.

why?



Consume - Present	Query - Identify	Query - Summarize	Query - Compare	Common Usages
☐	■	☐	■	▪ `Identify` a group or an individual point or area of interest ▪ `Compare` a group or an individual point or area of interest
☐	■	☐	■	▪ `Identify` connected visual elements ▪ `Compare` a group or an individual point or area of interest
■	■	■	■	▪ `Present` additional context ▪ `Identify`, `Summarize`, or `Compare` visual elements
☐	■	■	☐	▪ `Identify` visual elements ▪ `Compare` visual elements
☐	■	☐	■	▪ `Identify` visual elements ▪ `Compare` visual elements
☐	■	■	■	▪ `Identify` statistics and threshold values ▪ `Summarize` trends in data ▪ `Compare` trends in data
☐	■	☐	☐	▪ `Identify` a group or an individual point or area of interest





hat?

of Annotations

Data Source

External Annotations: Annotations that use data that are external to the dataset.

Derived Annotations: Annotations based on

ow?

of Annotations

N-Annotation Ensemble: When N distinct annotations are used together, it is an N-annotation (where N = 2, 3, 4, etc.) ensemble.

One-way: When one annotation can provide equivalent task support without the other, they have a 1-way dependency relationship.

Annotation Grammar

annogram.grammar

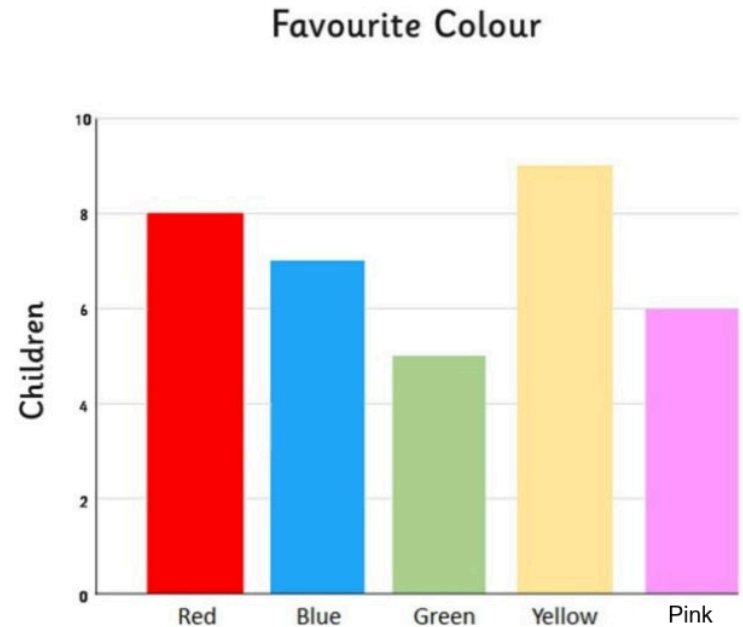
```
annotation := [
  target := (id|FixedPos|ChartPart|DataPoint|Axis|None)[]
]
```

Annotation Targets

DataPoint := *Expression* | `index[]`

ChartPart := `title` | `legend` | `subtitle` | `caption` | ⋯

Axis := { `axis` := `x` | `y`, `parts` := `label` | `tick` | `grid` | `tick-label`; `range` $\overset{\text{opt}}{:=}$ **Coord**[] | *Expression* }



Annotation Grammar

annogram.grammar

```
annotation := [
  target := (id|FixedPos|ChartPart|DataPoint|Axis|None)[]
]
text := Annotation<TextAnn>
enclosure := Annotation<EnclosureAnn>
connector := Annotation<ConnectorAnn>
indicator := Annotation<IndicatorAnn>
```

Annotation Targets

DataPoint := *Expression* | `index[]`

ChartPart := `title` | `legend` | `subtitle` | `caption` | ⋯

Axis := $\begin{cases} \texttt{axis} := \texttt{x} \mid \texttt{y}, \texttt{parts} := \texttt{label} \mid \texttt{tick} \mid \texttt{grid} \mid \texttt{tick-label} \\ \texttt{range} \overset{\text{opt}}{:=} \textbf{Coord}[] \mid \textit{Expression} \end{cases}$

Annotation Types

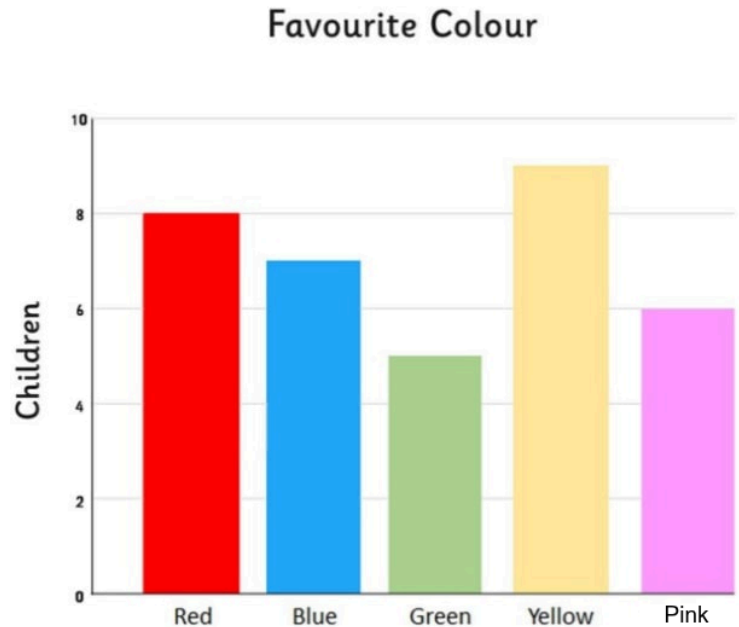
Annotation$\langle X \rangle$:= (`id`, **Style**, X), **Style** = ⋯

TextAnn := (*string*, **Position** | *None*)

EnclosureAnn := (**Rect** | **Ellipse** | `SVGShapePath` | ⋯, **Position** | *None*)

ConnectorAnn := (`Markers` | *None*, `SVGShapePath`, `linear` | `catmull-rom` | ⋯)

IndicatorAnn := (`line` | `area` | `arrow` | ⋯, *Expression*)



Annotation Grammar

annogram.grammar

```
annotation := [
  target := (id|FixedPos|ChartPart|DataPoint|Axis|None)[]
]
text := Annotation<TextAnn>
enclosure := Annotation<EnclosureAnn>
connector := Annotation<ConnectorAnn>
indicator := Annotation<IndicatorAnn>
```

Annotation Targets

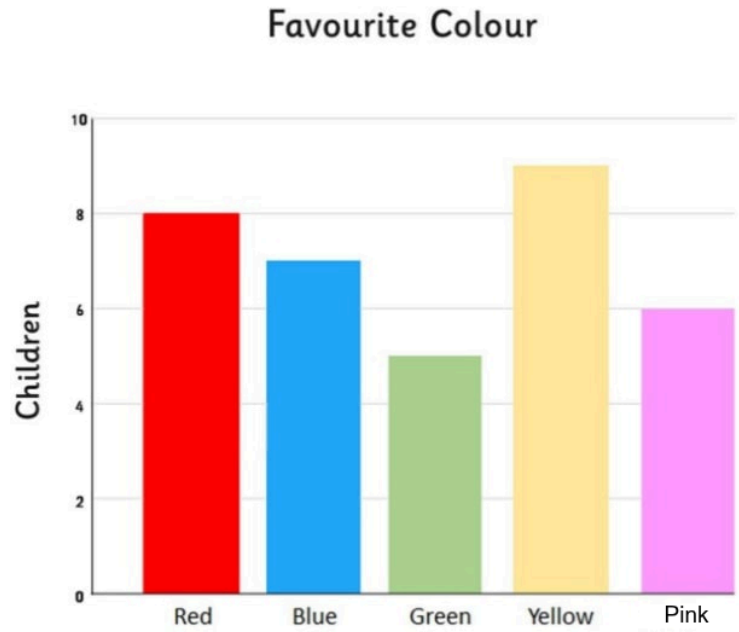
DataPoint := *Expression* | `index[]`
ChartPart := `title` | `legend` | `subtitle` | `caption` | ⋯
Axis := { `axis` := `x` | `y`, `parts` := `label` | `tick` | `grid` | `tick-label`; `range` :=$^{\text{opt}}$ **Coord**[] | *Expression* }

Annotation Types

Annotation⟨X⟩ := (`id`, **Style**, X), **Style** = ⋯
TextAnn := (*string*, **Position** | *None*)
EnclosureAnn := (**Rect** | **Ellipse** | `SVGShapePath` | ⋯, **Position** | *None*)
ConnectorAnn := (`Markers` | *None*, `SVGShapePath`, `linear` | `catmull-rom` | ⋯)
IndicatorAnn := (`line` | `area` | `arrow` | ⋯, *Expression*)

Placement

Position := **FixedPos** | **Anchor1D** | **Anchor2D**
Anchor1D := `auto` | `start` | `mid` | `end`, **Anchor2D** := `auto` | `upLeft` | `midRight` | ⋯
FixedPos := {`type` := `data` | `pixel`, `x` := **Coord**, `y` := **Coord**}
Coord := *string* | *number*



Expressivity of Annotations

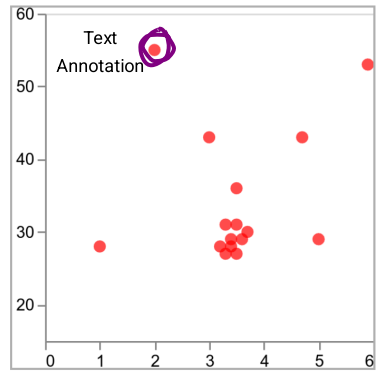
Support identifying the target and distance from annotation, and adds additional annotations if needed.

```
{
"$schema": ...,
...
"annotations": [{
  "target": ...,
  "text": {
    "text": "Text Annotation",
  },
}]
}
```

Expressivity of Annotations

Support identifying the target and distance from annotation, and adds additional annotations if needed.

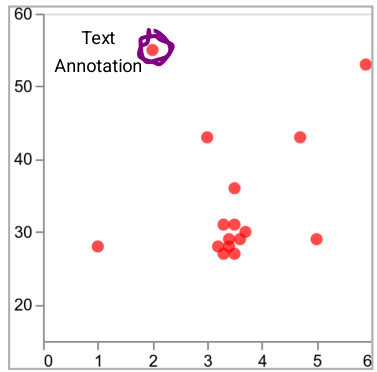
```
{
"$schema": ...,
...
"annotations": [{
  "target": ...,
  "text": {
    "text": "Text Annotation",
  },
}]
}
```

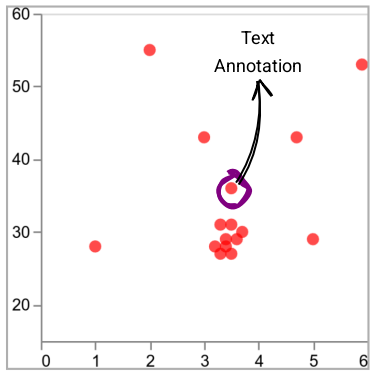


Expressivity of Annotations

Support identifying the target and distance from annotation, and adds additional annotations if needed.

```
{
"$schema": ...,
...
"annotations": [{
  "target": ...,
  "text": {
    "text": "Text Annotation",
  },
}]
}
```

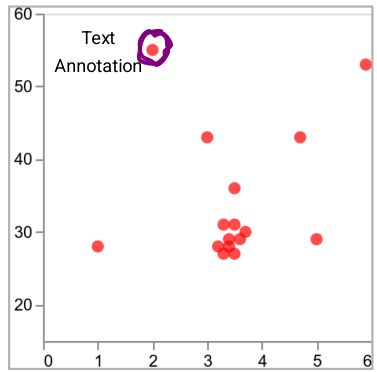


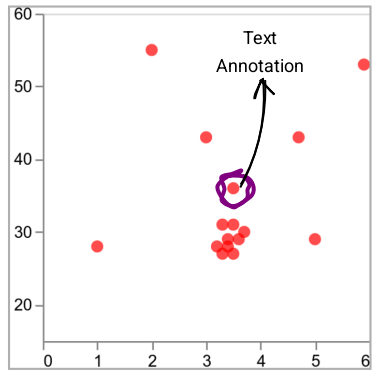


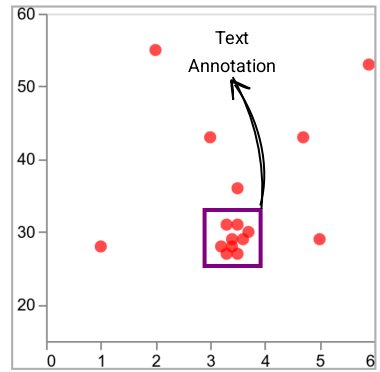
Expressivity of Annotations

Support identifying the target and distance from annotation, and adds additional annotations if needed.

```
{
"$schema": ...,
...
"annotations": [{
  "target": ...,
  "text": {
    "text": "Text Annotation",
  },
}]
}
```



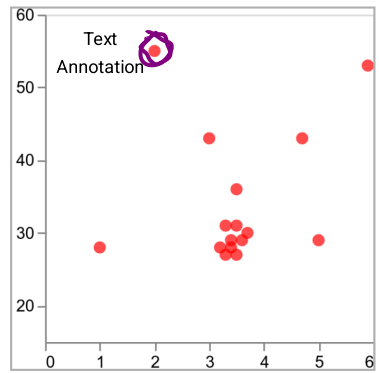


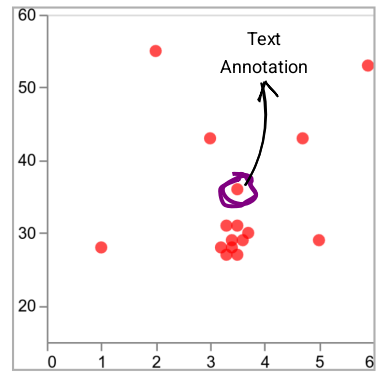


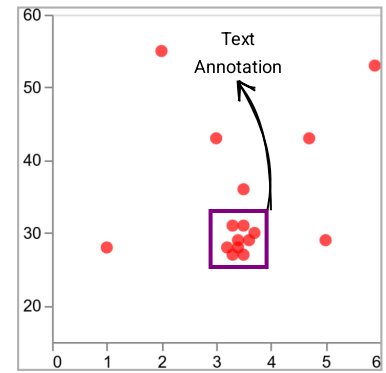
Expressivity of Annotations

Support identifying the target and distance from annotation, and adds additional annotations if needed.

```
{
"$schema": ...,
...
"annotations": [{
  "target": ...,
  "text": {
    "text": "Text Annotation",
  },
}]
}
```



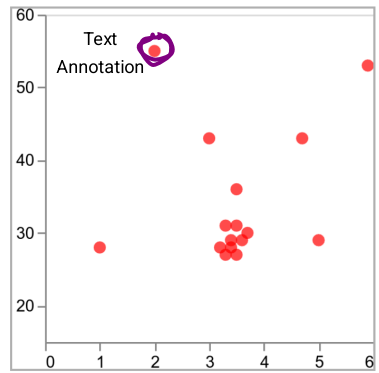


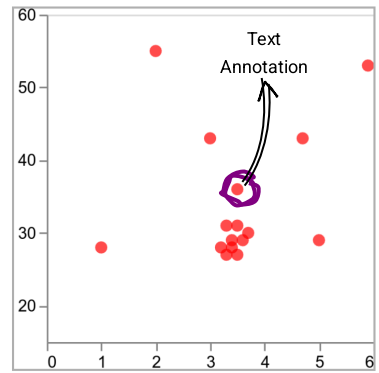


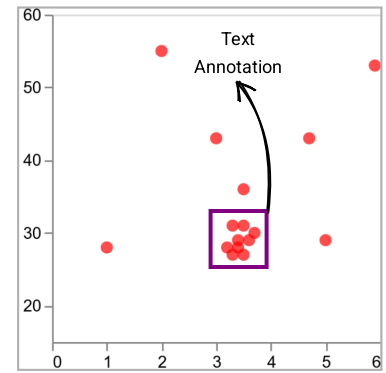
Expressivity of Annotations

Support identifying the target and distance from annotation, and adds additional annotations if needed.

```
{
"mark": { "Bar Chart " },
"annotations": [{
    "target": ...,
    "text": {
      "text": "Text Annotation"
    }
}]}
}
```



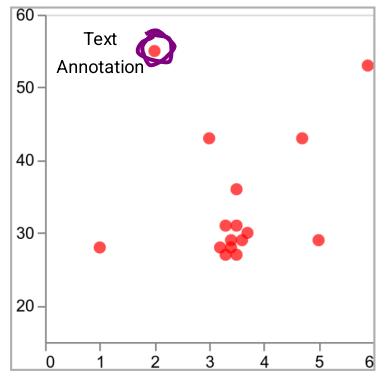


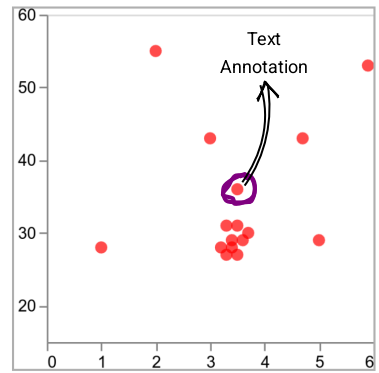


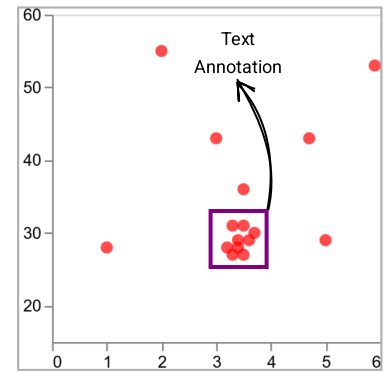
Expressivity of Annotations

Support identifying the target and distance from annotation, and adds additional annotations if needed.

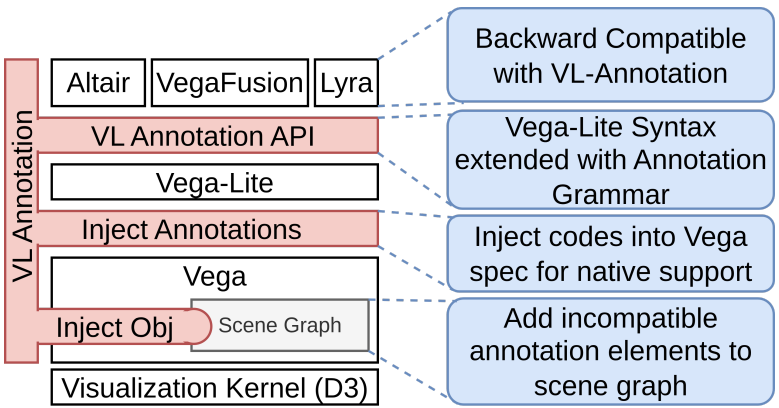
```
{
"mark": { "Bar Chart " },
"annotations": [{
    "target": ... ,
    "text": {
      "text": "Text Annotation"
    }
}]}
}
```

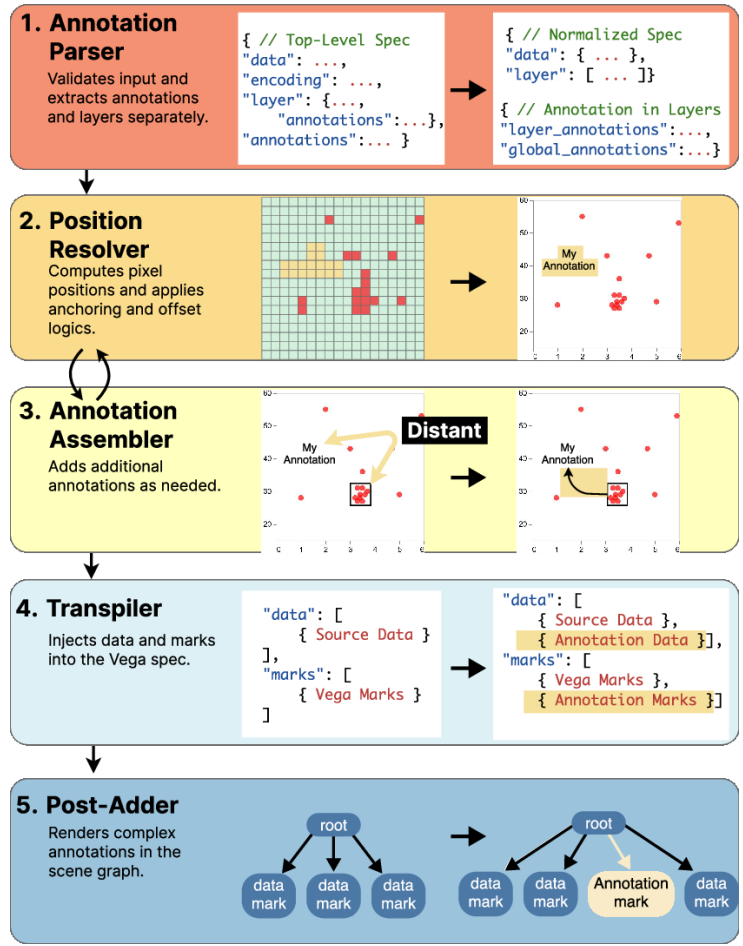




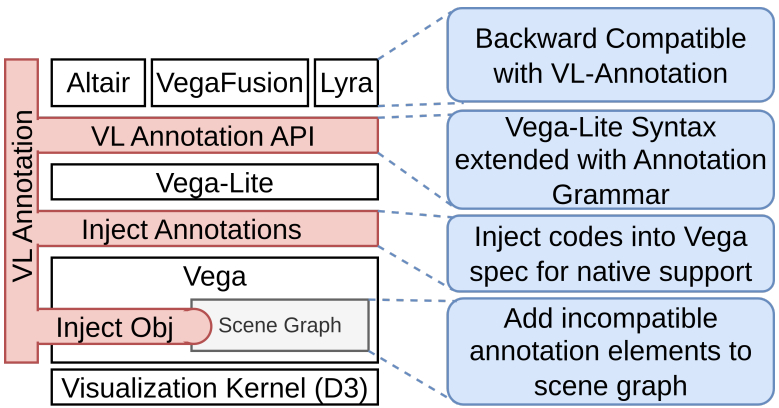


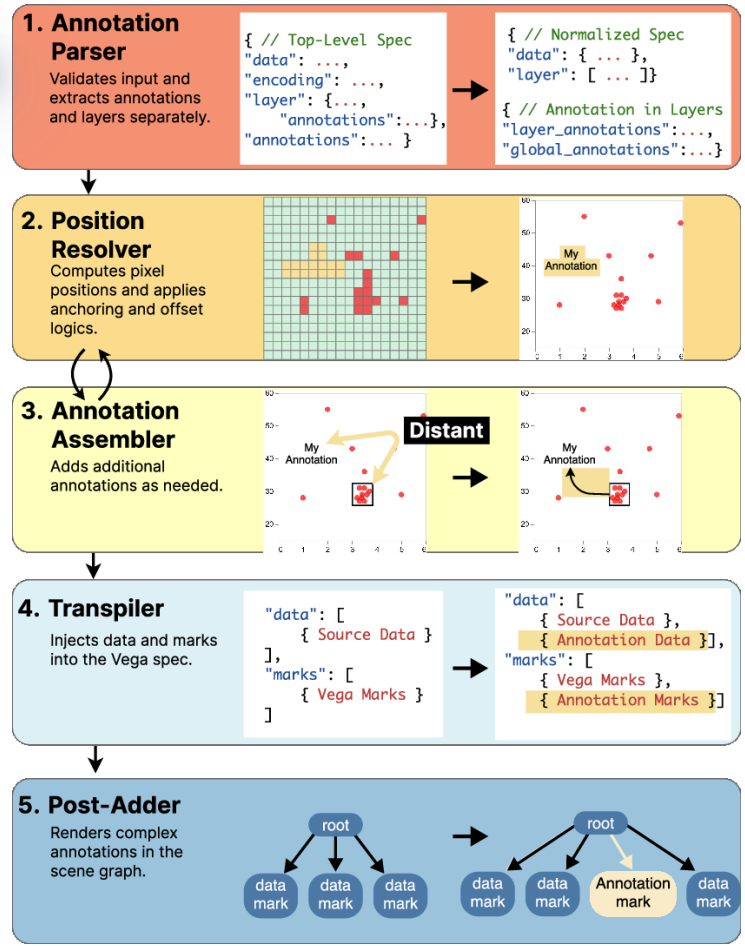
Implementation of AnnoGram in Vega-Lite



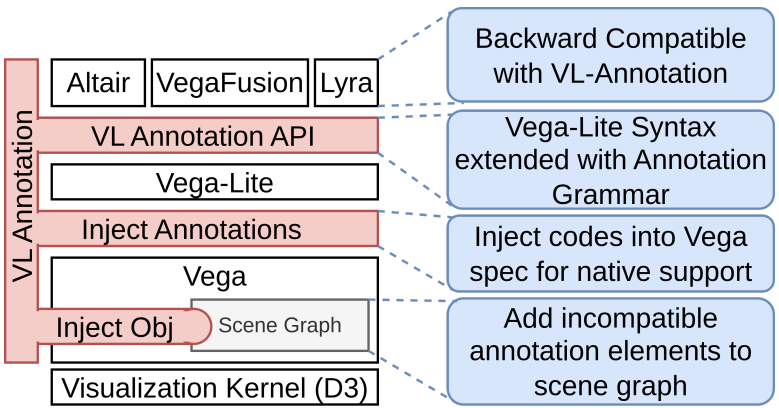


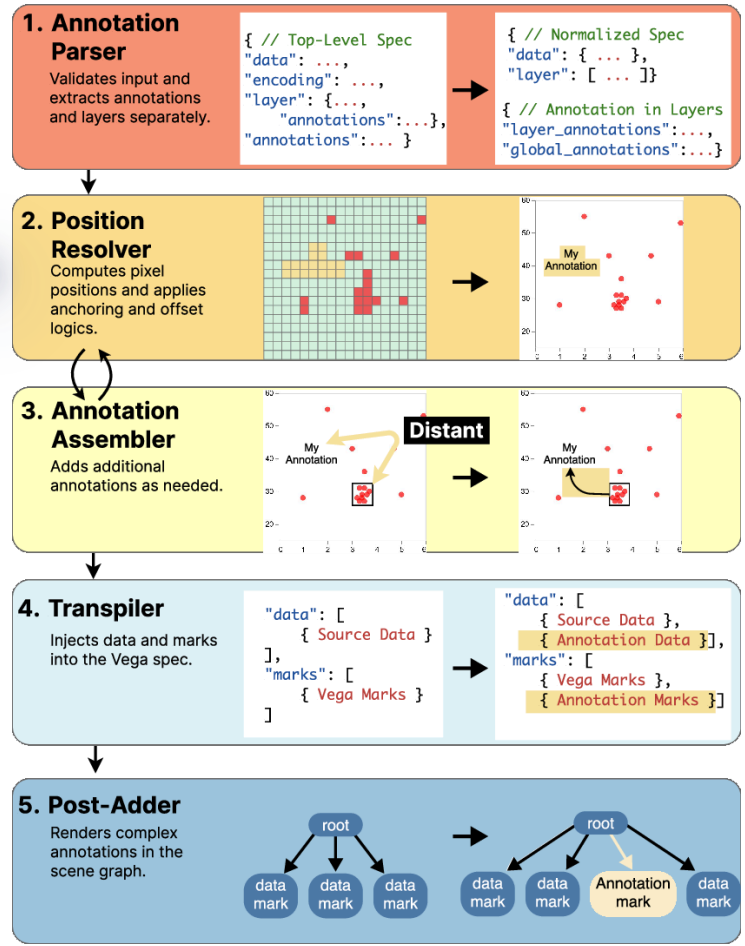
Implementation of AnnoGram in Vega-Lite



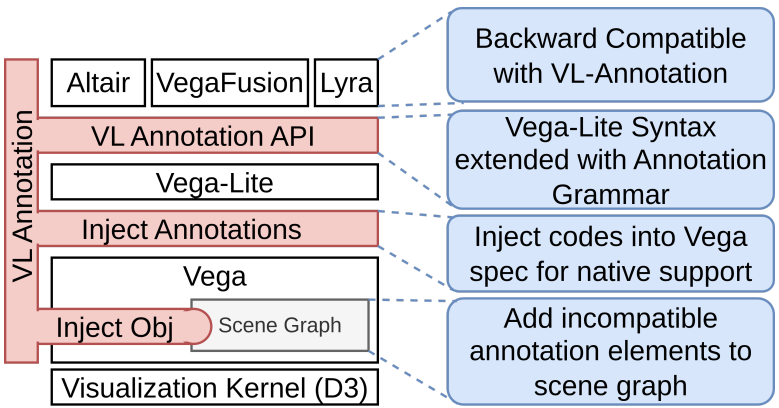


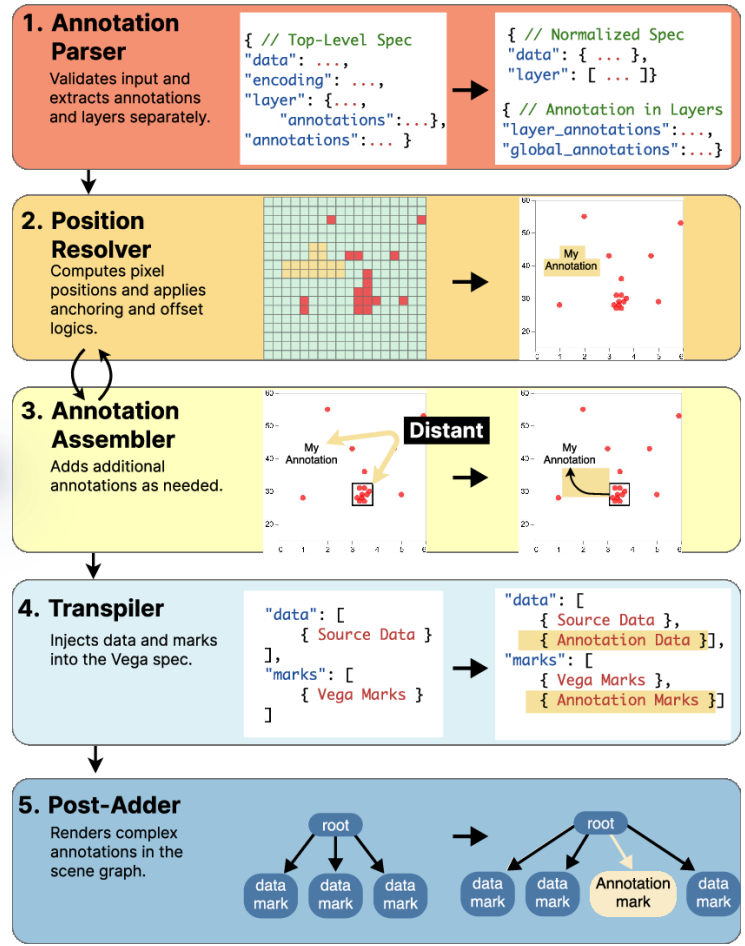
Implementation of AnnoGram in Vega-Lite



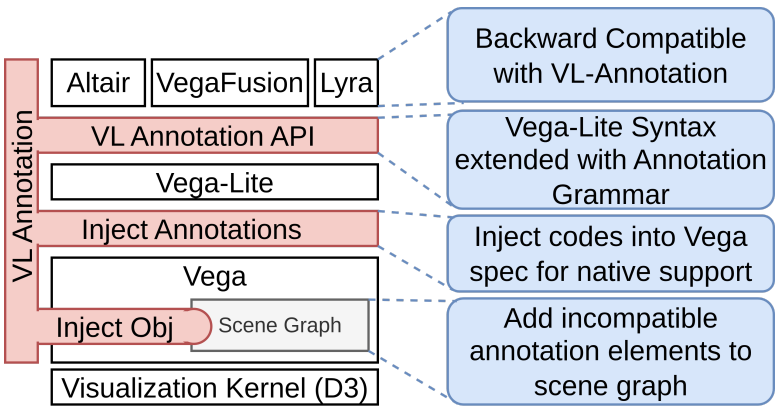


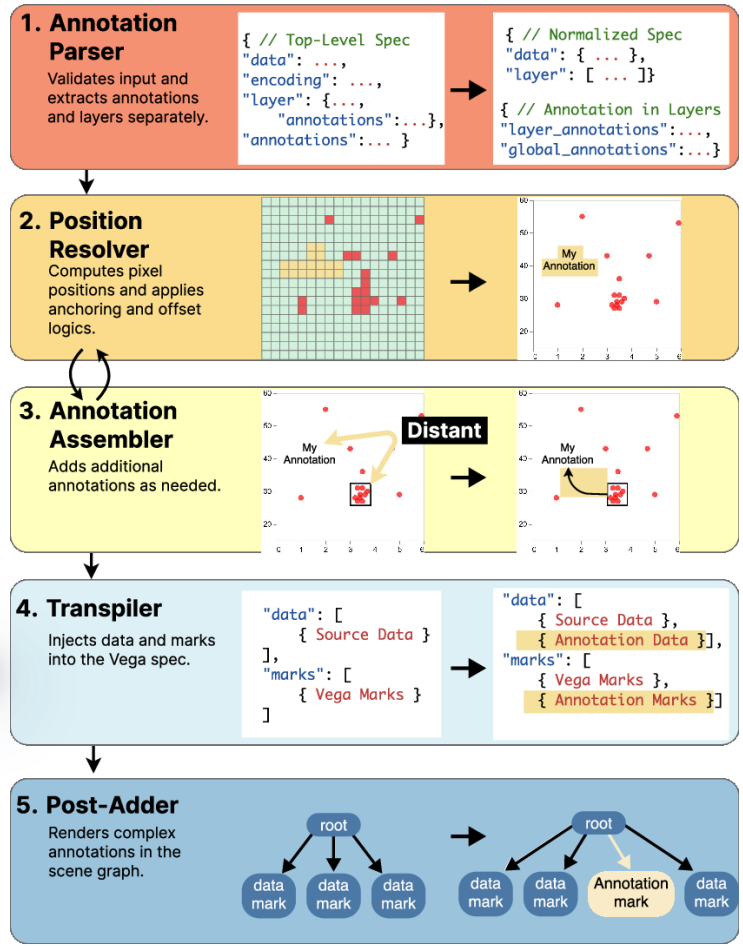
Implementation of AnnoGram in Vega-Lite



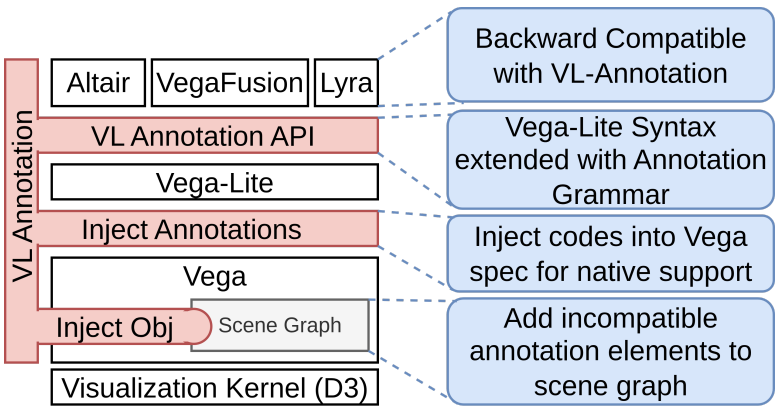


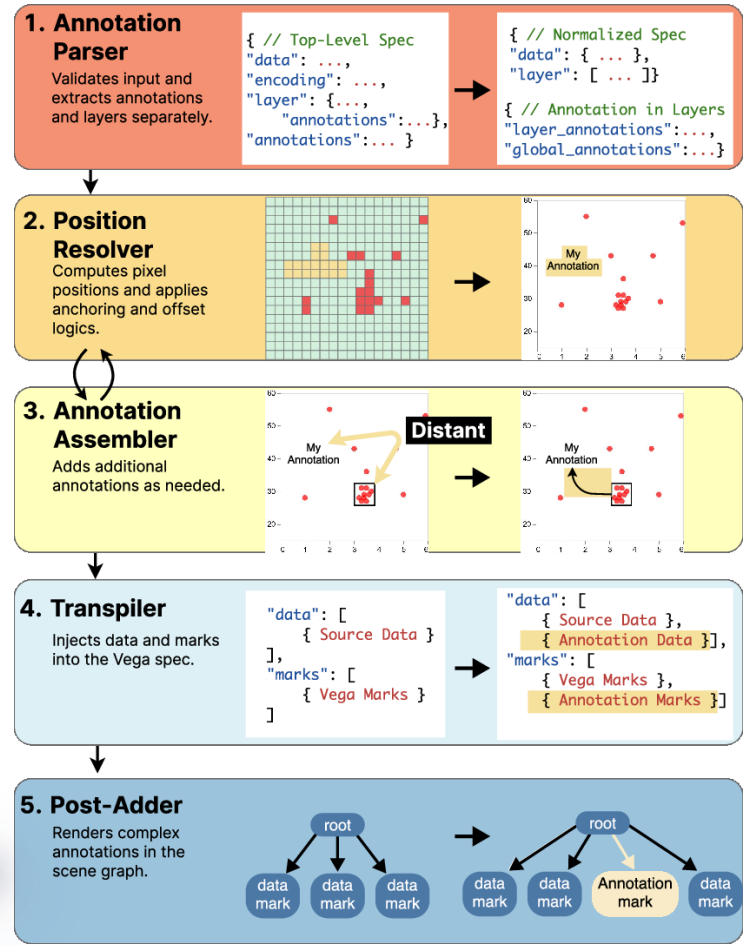
Implementation of AnnoGram in Vega-Lite





Implementation of AnnoGram in Vega-Lite





Final Discussion







Final Discussion







First-Class Annotations

Annotations should be treated as first-class citizens in the visualization grammar.

Final Discussion



First-Class Annotations

Annotations should be treated as first-class citizens in the visualization grammar.



Annotation Grammar

We designed a visualization grammar extension that uses annotation as a first-class construct and dynamically targets data points, chart parts, or coordinate spaces.



Final Discussion



First-Class Annotations

Annotations should be treated as first-class citizens in the visualization grammar.



Annotation Grammar

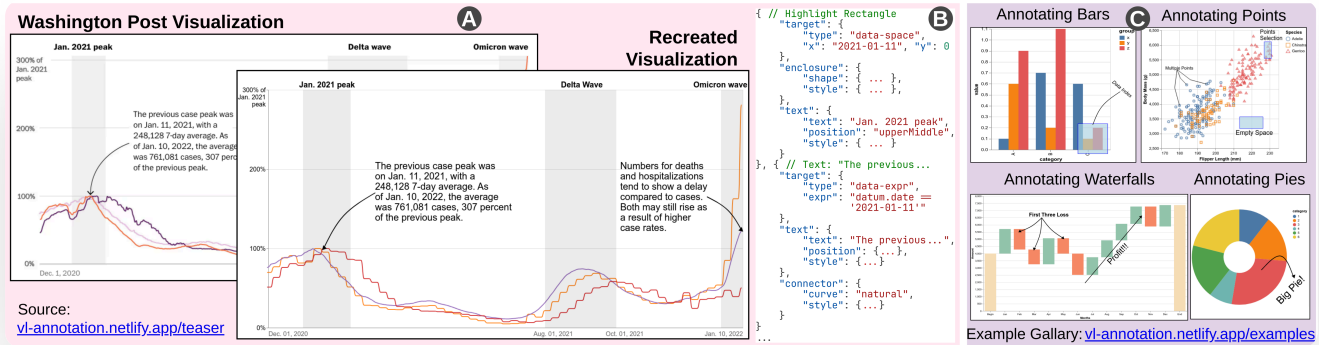
We designed a visualization grammar extension that uses annotation as a first-class construct and dynamically targets data points, chart parts, or coordinate spaces.



Vega-Lite-Annotation Library

We implemented this Annotation Grammar as a Vega-Lite extension library.

Thank You!



AnnoGram: An Annotative Grammar of Graphics Extension

Authors

Dilshadur Rahman

Rahat Zaman

Andrew McNutt

Paul Rosen

Resources

arxiv.org/abs/2507.04236

rahatzamancse/vega-lite-annotation

vl-annotation.netlify.app

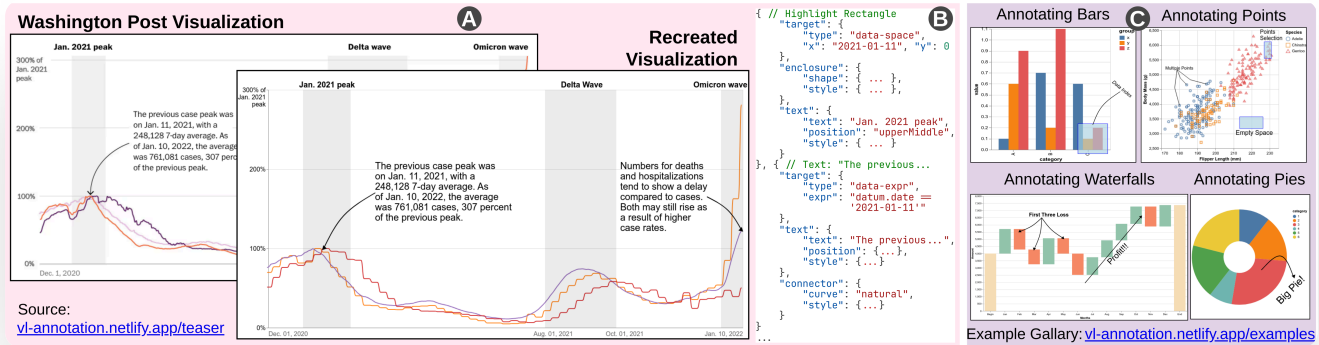
Presenter

Rahat Zaman

rahat.zaman@utah.edu

rahatzaman.me

Thank You!



AnnoGram: An Annotative Grammar of Graphics Extension

Play with Demo!

Authors

Dilshadur Rahman

Rahat Zaman

Andrew McNutt

Paul Rosen

Resources

arxiv.org/abs/2507.04236

rahatzamancse/vega-lite-annotation

vl-annotation.netlify.app

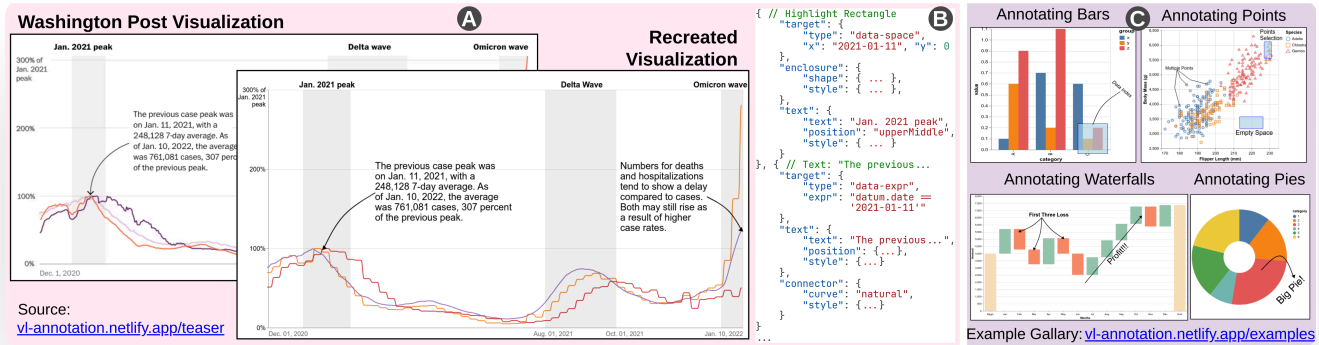
Presenter

Rahat Zaman

rahat.zaman@utah.edu

rahatzaman.me

Thank You!



AnnoGram: An Annotative Grammar of Graphics Extension

Play with Demo!

Looking for Internship!

Authors

Dilshadur Rahman

Rahat Zaman

Andrew McNutt

Paul Rosen

Resources

arxiv.org/abs/2507.04236

rahatzamancse/vega-lite-annotation

vl-annotation.netlify.app

Presenter

Rahat Zaman

rahat.zaman@utah.edu

rahatzaman.me

Other Reference Slides

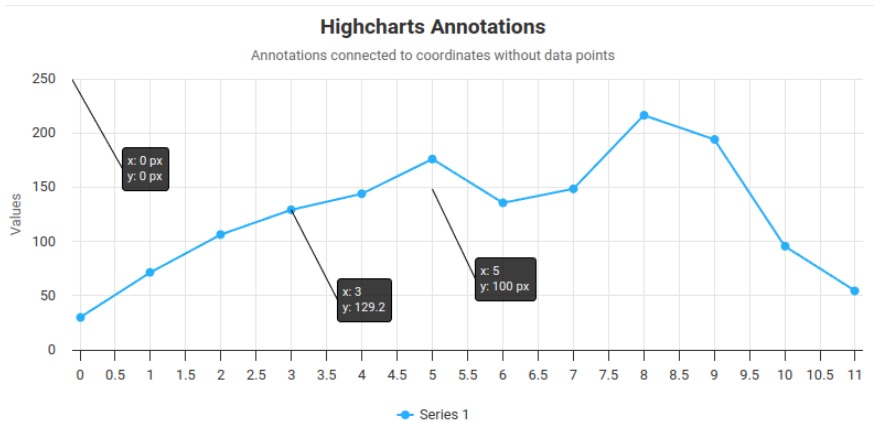
Problem 1: existing grammars treat annotations as *secondary*

```
annotations: [{
  labels: [{
    point: {
      x: 3, y: 129.2,
      xAxis: 0, yAxis: 0,
    }
  }, {
    point: {
      x: 0, y: 0
    }
  }, {
    point: {
      x: 5, y: 100,
      xAxis: 0
    }
  }]
}]
```

Problem 2: no support for established taxonomy

HighChart

- Only a text with a line connector
- No binding with data



Support of Annotations for Popular Libraries



D3

- Low-level library: no support for annotations
- Has annotation extensions (e.g., d3-annotation)
- Annotations are reusable with programmatic control



Plotly

- Built-in text annotation support
- Add annotations by hack with custom HTML
- No reusable annotations



Matplotlib, Seaborn, ggplot2

- Primitive shapes for annotation (e.g., box, line, text)
- Text annotation supported
- No reusable annotations



Declarative Visualization Grammar

🧩 Declarative Visualization Grammar

- A **declarative grammar** specifies **what** to show in a visualization, not **how** to draw it.

🧩 Declarative Visualization Grammar

- A **declarative grammar** specifies **what** to show in a visualization, not **how** to draw it.
- It breaks down charts into **data**, **encodings**, and other components for reusable and composable specs.

🧩 Declarative Visualization Grammar

- A **declarative grammar** specifies **what** to show in a visualization, not **how** to draw it.
- It breaks down charts into **data**, **encodings**, and other components for reusable and composable specs.
- Visualizations are described as **structured specifications** instead of step-by-step drawing code.

🧩 Declarative Visualization Grammar

- A **declarative grammar** specifies **what** to show in a visualization, not **how** to draw it.
- It breaks down charts into **data**, **encodings**, and other components for reusable and composable specs.
- Visualizations are described as **structured specifications** instead of step-by-step drawing code.
- Annotations should be defined and be reusable within the grammar itself.

🧩 Declarative Visualization Grammar

- A **declarative grammar** specifies **what** to show in a visualization, not **how** to draw it.
- It breaks down charts into **data**, **encodings**, and other components for reusable and composable specs.
- Visualizations are described as **structured specifications** instead of step-by-step drawing code.
- Annotations should be defined and be reusable within the grammar itself.

Example libraries



Vega-Lite



ggplot2



Altair



plotnine



Chart.js

AnnoGram Implementation

Annotation Parser

Validates input and extracts annotations in layers separately.

```
{ // Top-Level Spec
  "data": ...,
  "encoding": ...,
  "layer": {
    "annotations": ...,
    ...
  },

  "annotations": ...
}
```



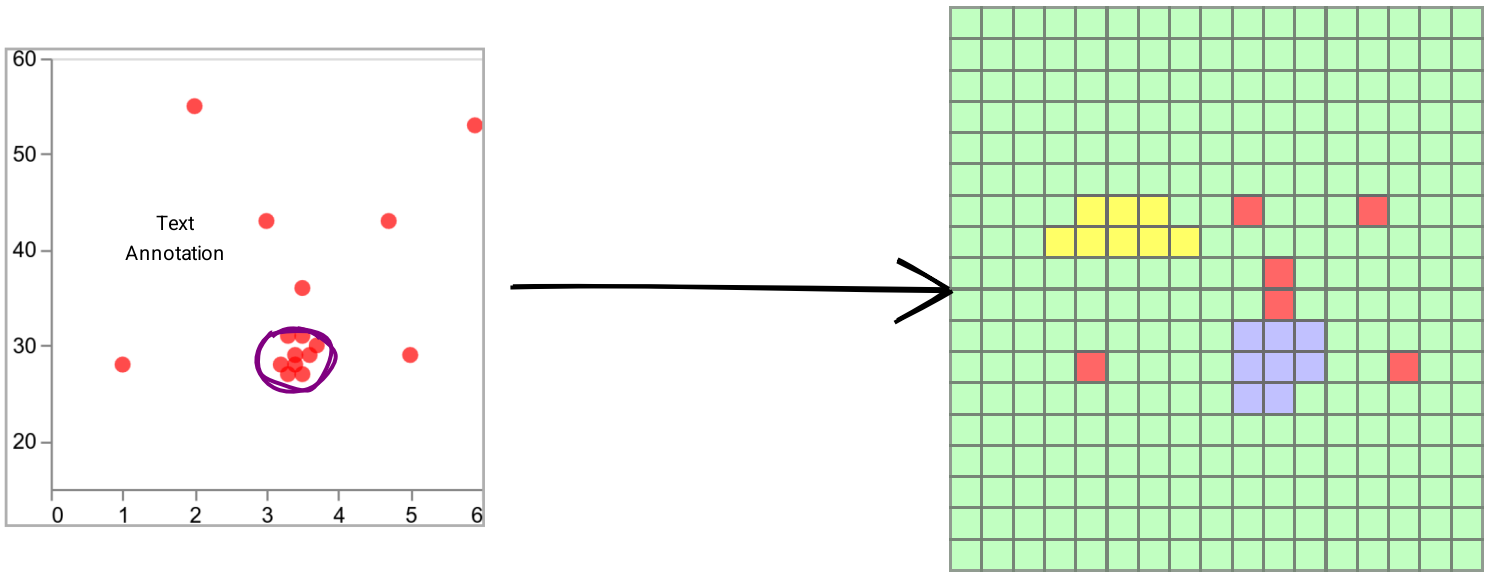
```
{ // Normalized Spec
  "data": { ... },
  "layer": [ ... ]
}


{ // Annotation in Layers
  "layer_annotations": ...,
  "global_annotations": ...
}
```

AnnoGram Implementation

Position Resolver

Computes pixel positions and applies anchoring and offset logics.



AnnoGram Implementation

Annotation Assembler

Identifies the target and distance from annotation, and then adds additional annotations if needed.

```
{
"$schema": ... ,
...
"annotations": [{
  "target": ... ,
  "text": {
    "text": "Text Annotation",
  },
}]
}
```

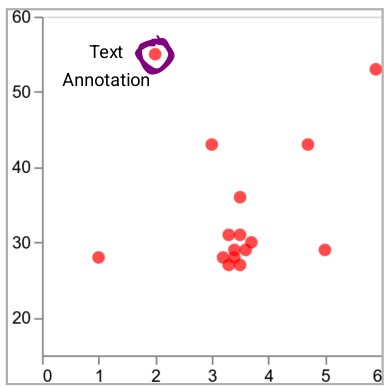


AnnoGram Implementation

Annotation Assembler

Identifies the target and distance from annotation, and then adds additional annotations if needed.

```
{
"$schema": ... ,
...
"annotations": [{
  "target": ... ,
  "text": {
    "text": "Text Annotation",
  },
}]
}
```

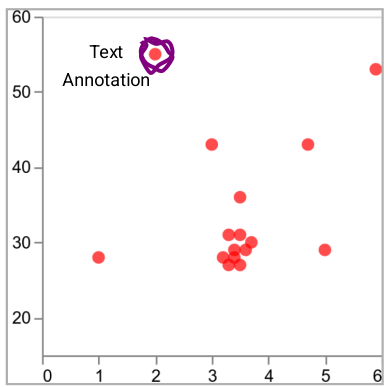


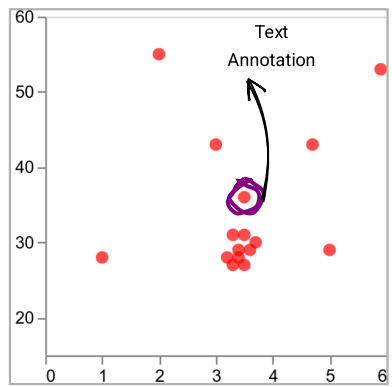
AnnoGram Implementation

Annotation Assembler

Identifies the target and distance from annotation, and then adds additional annotations if needed.

```
{
"$schema": ... ,
...
"annotations": [{
  "target": ... ,
  "text": {
    "text": "Text Annotation",
  },
}]
}
```



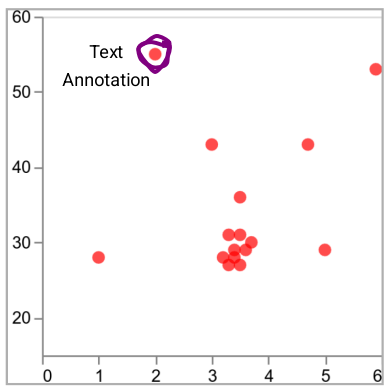


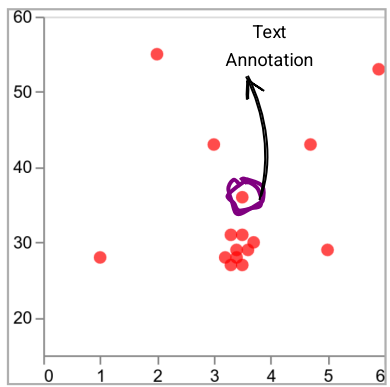
AnnoGram Implementation

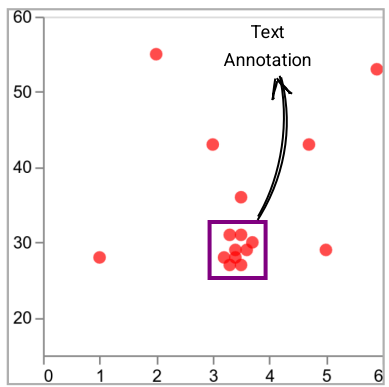
Annotation Assembler

Identifies the target and distance from annotation, and then adds additional annotations if needed.

```
{
"$schema": ...,
...
"annotations": [{
  "target": ...,
  "text": {
    "text": "Text Annotation",
  },
}]
}
```







AnnoGram Implementation

Transpiler

Injects data and marks into the Vega spec.

```
"data": [
  { Source Data }
],
"marks": [
  { Vega Marks }
]
```



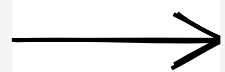
```
"data": [
  { Source Data },
  { Annotation Data }
],
"marks": [
  { Vega Marks },
  { Annotation Marks }
]
```

AnnoGram Implementation

Transpiler

Injects data and marks into the Vega spec.

```
{ // VLA Specification
"data": { ... },
"mark": "bar",
"encoding": {
  "x": { "field": "category" },
  "y": { "field": "value" },
},
"annotations": [{
  "target": {
    "type": "data-expr",
    "expr": "datum.value < 0.25\
      && datum.category == 'C'"
  },
  "text": {
    "text": "Data Index",
    "dy": -4,
    "style": { "angle": 45 }
  }
}]}
```



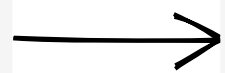
```
"data": [
  { Source Data },
  { Annotation Data }
],
"marks": [
  { Vega Marks },
  { Annotation Marks }
]
```

AnnoGram Implementation

Transpiler

Injects data and marks into the Vega spec.

```
{ // VLA Specification
"data": { ... },
"mark": "bar",
"encoding": {
  "x": { "field": "category" },
  "y": { "field": "value" },
},
"annotations": [{
  "target": {
    "type": "data-expr",
    "expr": "datum.value < 0.25\
      && datum.category == 'C'"
  },
  "text": {
    "text": "Data Index",
    "dy": -4,
    "style": { "angle": 45 }
  }
}]}
```



```
{ // Vega Specification
  "data": [ ... , {
    "name": "data_0",
    "source": "source_0",
    "transform": [ ...  ]
  }],
  "marks": [{
    "type": "rect",
    "from": { "data": "data_0" },
    "encode": { ... }
  }, {
    "name": "AnnotationEnclosuren4pwxqkxn0e",
    "type": "rect",
    "encode": { ... }
  }, ... ],
}
```

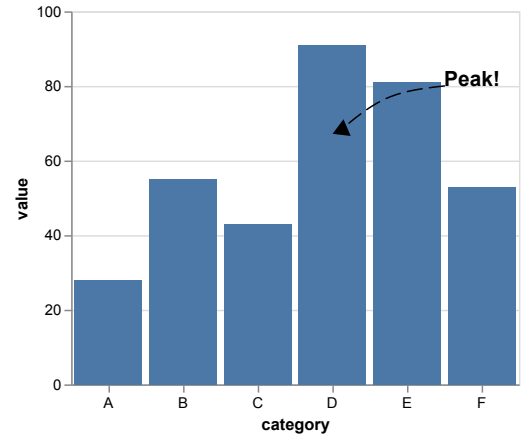
Evaluation of AnnoGram Implementation

We implemented a trio of example charts with annotations across various types of tools.

Programmability	Tool	Bar Ann. LOC	Bar LOC	Line Ann. LOC	Line LOC	Scatter Ann. LOC	Scatter LOC	1st-class Support	Intuitiveness	Error Proneness	Portability
High	D3	151	81	175	76	170	104	✗	Hard	High	Hard
	D3-annotate	161	81	174	77	177	99	✓	Med	Mid	Mid
	ggplot2	120	22	128	43	208	101	✓	Med	High	Mid
	ggplot2-annotate	**73**	20	117	42	178	101	✓	Med	High	Mid
Low	HighCharts	82	32	231	82	169	79	✓	Med	Low	Hard
	Vega	515	161	491	219	387	186	✗	Hard	High	Hard
	Vega-Lite	259	23	253	29	177	34	✗	Hard	High	Hard
	VL Annotation	101	**19**	**95**	**25**	**79**	**31**	✓	Easy	Low	Easy
⋮	Visual Editors (PowerPoint, Figma, etc.)	--						--	Easy	Low	Hard

Annotation Grammar Implementation

```
{
  "$schema": "https://vega.github.io/schema/vega-lite/v6.json",
  "width": 300,
  "height": 250,
  "data": {
    "values": [
      { "category": "A", "value": 28 },
      { "category": "B", "value": 55 },
      { "category": "C", "value": 43 },
      { "category": "D", "value": 91 },
      { "category": "E", "value": 81 },
      { "category": "F", "value": 53 }
    ]
  },
  "mark": "bar",
  "encoding": {
    "x": { "field": "category", "type": "nominal", "axis": { "labelAngle": 0 } },
    "y": { "field": "value", "type": "quantitative" }
  },
  "annotations": [
    {
      "target": {
        "type": "pixel-space",
        "x": 250,
```



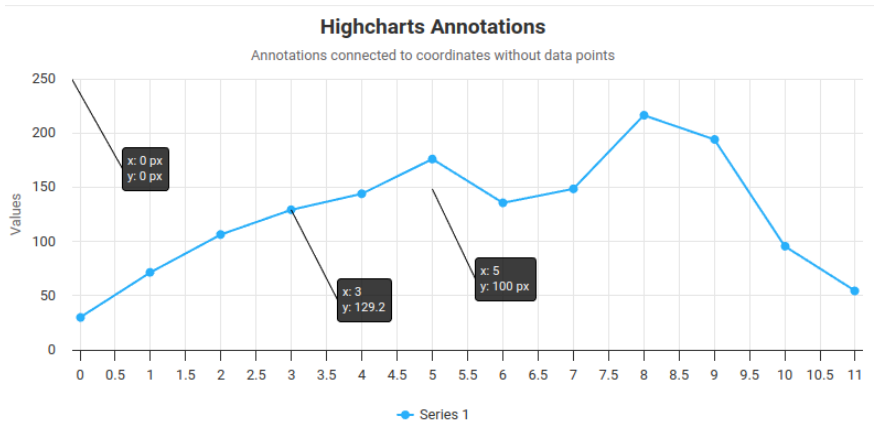
Problem 1: existing grammars treat annotations as *secondary*

```
annotations: [{
  labels: [{
    point: {
      x: 3, y: 129.2,
      xAxis: 0, yAxis: 0,
    }
  }, {
    point: {
      x: 0, y: 0
    }
  }, {
    point: {
      x: 5, y: 100,
      xAxis: 0
    }
  }]
}]
```

Problem 2: no support for established taxonomy

HighChart

- Only a text with a line connector
- No binding with data



Support of Annotations for Popular Libraries



D3

- Low-level library: no support for annotations
- Has annotation extensions (e.g., d3-annotation)
- Annotations are reusable with programmatic control



Plotly

- Built-in text annotation support
- Add annotations by hack with custom HTML
- No reusable annotations



Matplotlib, Seaborn, ggplot2

- Primitive shapes for annotation (e.g., box, line, text)
- Text annotation supported
- No reusable annotations