VAID Specification Codebook

Analytical Task Specification

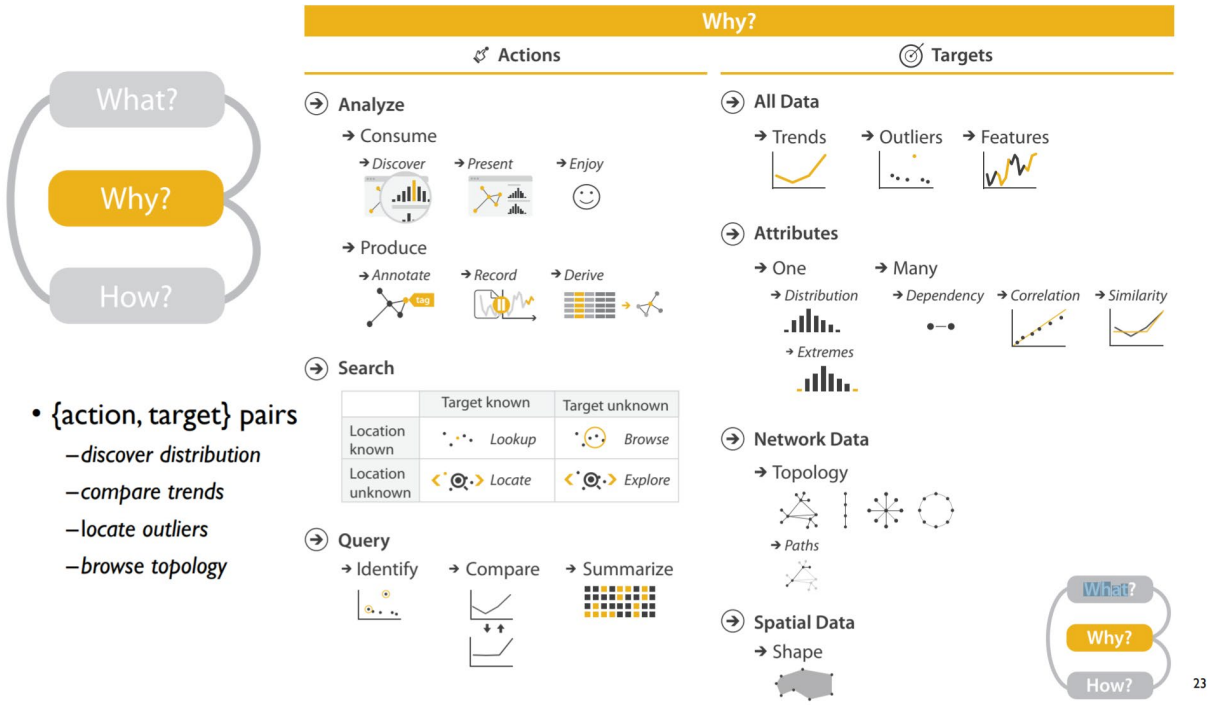


Figure is from *Munzner T. Visualization analysis and design[M]. CRC press, 2014. Page 46 & Page 56*

1. ACTION

The definitions of actions are quoted from *Brehmer M, Munzner T. A multi-level typology of abstract visualization tasks[J]. IEEE transactions on visualization and computer graphics, 2013, 19(12): 2376-2385.*

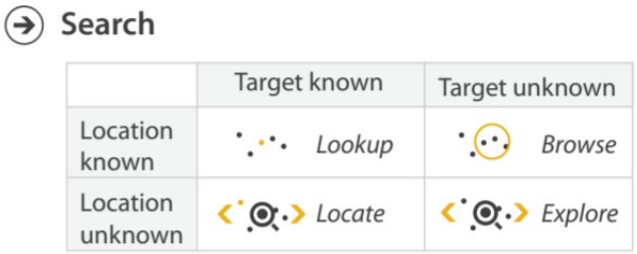
Consume

- **Present** refers to “the use of visualization for the succinct communication of information, for telling a story with data, guiding an audience through a series of cognitive operations”.
- **Discover** is “about the generation and verification of hypotheses, associated with modes of scientific inquiry”.
- **Enjoy** refers to “casual encounters with visualization”.

Produce

Definition: We use “produce in reference to tasks in which the intent is to generate new artifacts, including transformed or derived data, annotations, recorded visualization interactions, or screenshots of static visualizations”.

Search



- **Lookup:** target known, location known
- **Browse:** target unknown, location known
- **Locate:** target known, location unknown
- **Explore:** target unknown, location unknown

Query

- **Identify:** “returns characteristics or reference for a target”
- **Compare:** “returns characteristics or reference for two or multiple targets”
- **Summarize:** “returns characteristics or reference for a whole set of targets”

1.2 TARGET

Tabular Data

Values, extremum, ranges, distributions, anomalies, clusters, correlation, similarities, orders
The taxonomy is based on *Amar R, Eagan J, Stasko J. Low-level components of analytic activity in information visualization[C] IEEE Symposium on Information Visualization, 2005. INFOVIS 2005. IEEE, 2005: 111-117.*

Graph Data

Graphs, nodes, links/paths, topology/structures, group/clusters

The taxonomy is based on *Lee B, Plaisant C, Parr C S, et al. Task taxonomy for graph visualization[C] Proceedings of the AVI workshop on BEyond time and errors: novel evaluation methods for information visualization. 2006: 1-5.*

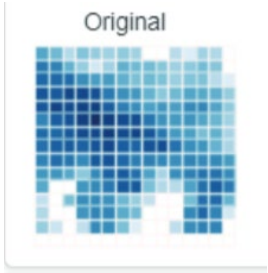
Data & Visualization Specification

We specify the visualization configuration in Visual Analytics designs by referring to declarative programming grammars, such as Vega, Vega-lite, Echarts, which are proven to be efficient to specify the visual encodings and compositions.
Specifically, Echarts supports generation of graph-related data:

- Sankey: Node + Link
- Tree/treemap:
    - data that is organized in a hierarchical structure, {name, children:[{name, children}], collapse:boolen}.
    - Layout: radial,
- Graph:
    - Node + Link

Our primary goal is to specify the visual mapping from data to visual channels/layouts.

1. Specify the visual mapping between “field”, “type”, and “encoding”




```
{
   "mark": "rect",
   "encoding": {
     "y": {
       "field": "flow_1",
       "type": "nominal"
     },
     "x": {
       "field": "flow_2",
       "type": "nominal"
     },
     "color": {
       "field": "flow_coverage",
       "type": "quantitative"
     }
   }
}
```

- About “field”: using the name in the paper, and check if the two fields are the same column (double encoding).
- About “aggregate”: bin, mean, count, sum, min, max, median

2. Specify the composition type of the visualization

Aggregate Types

- Count
- Sum
- Min/max
- Average (mean)
- Median
- Variance
- Stdev
- q1, q3, ci0, ci1

Composition Types



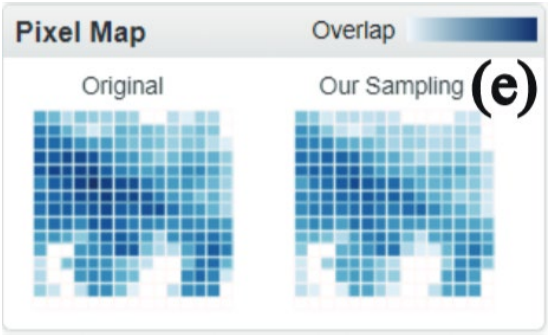

Facet

Specification:

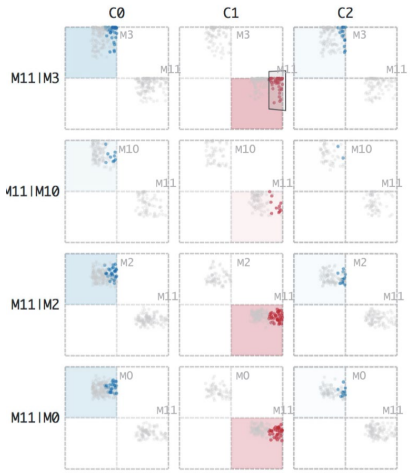
```
facet: {
        layout: “grid/circular/mirrored” (default grid)
        row:{}
        column:{}
},
spec: {}
```

Example:

1. Grid layout

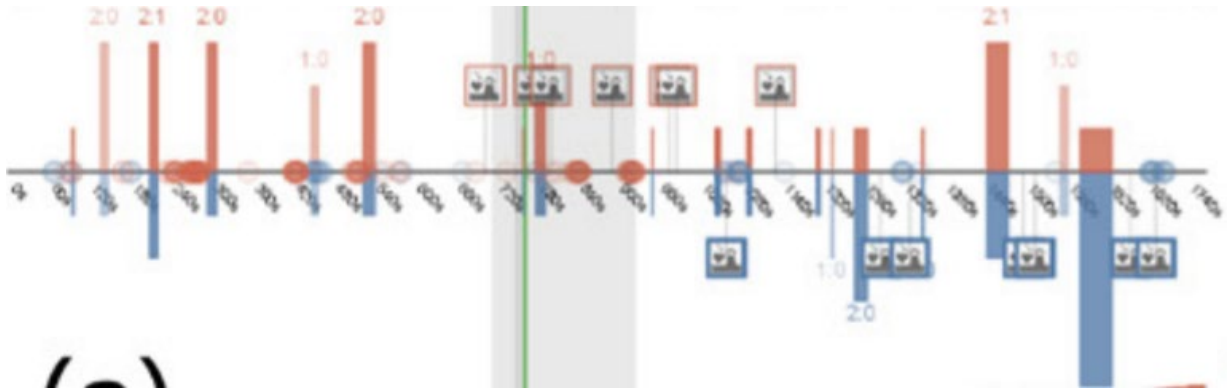



```
{
  "facet": {
    "column": {
      "field": "method",
      "type": "nominal"
    }
  },
  "spec": {
    "mark": "rect",
    "encoding": {
      "y": {
        "field": "flow_1",
        "type": "nominal"
      },
      "x": {
        "field": "flow_2",
        "type": "nominal"
      },
      "color": {
        "field": "flow_coverage",
        "type": "quantitative"
      }
    }
  }
}
```




```
{
  "facet": {
    "column": {
      "field": "class (Ck)",
      "type": "nominal"
    },
    "row": {
      "field": "model_pair (Mi, Mj)",
      "type": "nominal"
    }
  },
  "spec": {
    "mark": "point",
    "encoding": {
      "x": {
        "field": "Mi_prediction_score",
        "type": "quantitative"
      },
      "y": {
        "aggregate": "Mj_prediction_score",
        "type": "quantitative"
      },
      "color": {
        "field": "ground_truth_is_Ck_or_not",
        "type": "nominal"
      }
    }
  }
}
```

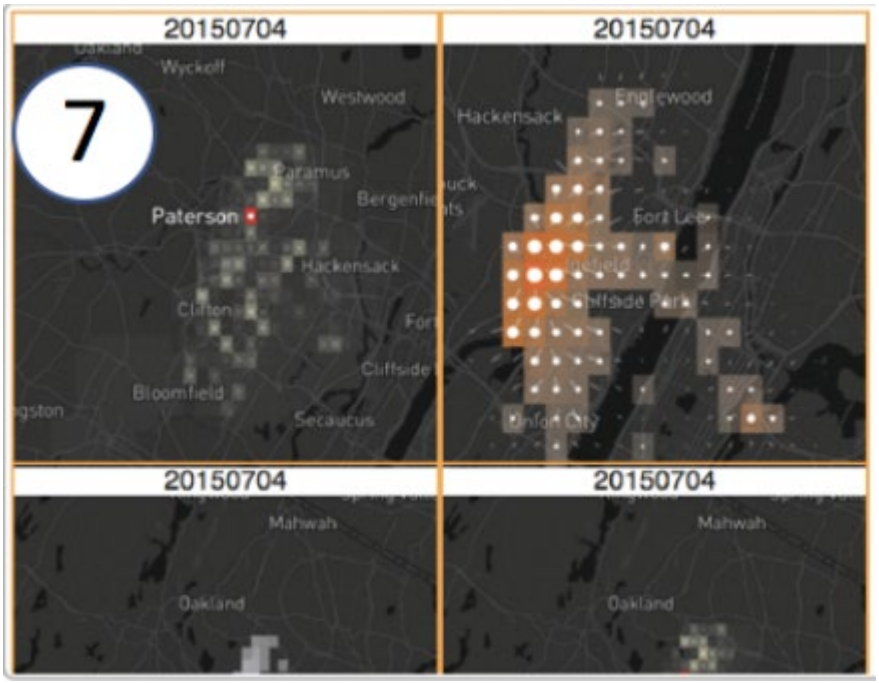
2. Mirrored layout



```
{
  "layer": [
    {
      "facet": {
        "layout": "mirrored",
        "row": {
          "field": "team",
          "type": "nominal"
        },
        "color": {
          "field": "team",
          "type": "nominal"
        }
      },
      "spec": {
        "mark": "bar",
        "encoding": {
          "x": {
            "field": "time",
            "type": "temporal"
          },
          "y": {
            "field": "combat_result",
            "type": "quantitative"
          },
          "width": {
            "field": "duration",
            "type": "quantitative"
          }
        }
      }
    },
    {
      "mark": "point",
      "remark": "attacking towers",
      "encoding": {
        "x": {
          "field": "time",
```

```
        "type": "temporal"
      },
      "color": {
        "field": "team",
        "type": "nominal"
      }
    }
  },
  {
    "mark": "others",
    "remark": "occupying light towers",
    "encoding": {
      "x": {
        "field": "time",
        "type": "temporal"
      },
      "stroke": {
        "field": "team",
        "type": "nominal"
      }
    }
  }
 ]
}
```

3. List layout



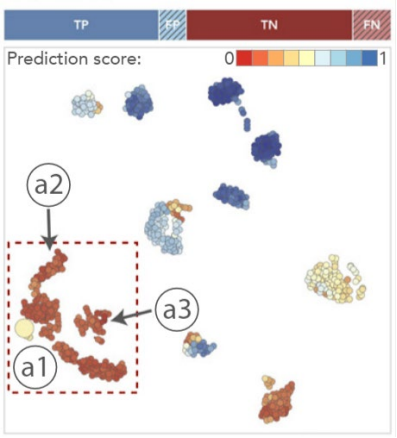

```
{
```

```
  "facet": {
      "field": "date",
      "columns": 2,
      "type": "temporal"
    } },
  "spec": {
  }
}
```

Concat

Vega-lite: vconcat, hconcat. However, there might be multiple concatenation directions, such as crossing. Therefore, we have a concat + layout specification

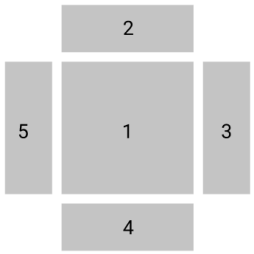
1. Vertical/horizontal




```
{
  concat: {
    layout: 'vertical'
  },
  spec: [
    {
      mark: 'bar',
      encoding: {
        x: {
          aggregate: 'count',
          type: 'quantitative'
        },
        color: {
          field: 'confusion_type',
          type: 'nominal'
        }
      }
    },
    {
      mark: 'point',
```

```
      encoding: {
        x: {
          field: 'dr_1',
          type: 'quantitative'
        },
        y: {
          field: 'dr_2',
          type: 'quantitative'
        },
        color: {
          field: 'prediction_score',
          type: 'quantitative'
        }
      }
    }
  ]
}
```

2. Crossing




```
{
  "concat": {
    "layout": "crossing
  "},
  "spec": [
    {
      "mark": "geoshape",
      "position": 1,
      "encoding": {}
    },
    {
      "mark": "line",
      "position": 2,
      "encoding": {}
    },
    {
      "mark": "line",
      "position": 3,
      "encoding": {}
    }
  ]
```

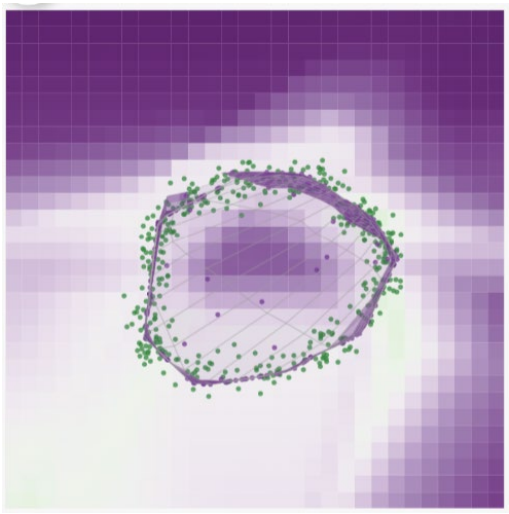
```
}
```

Layer

Specification:

```
layer: []
```

1. Plain layout



```
{
  "layer": [
   {,
    {
      "mark": "point",
      "encoding": {
        "x": {
          "field": "dimension_1",
          "type": "quantitative"
        },
        "y": {
          "field": "dimension_2",
          "type": "quantitative"
        }
      }
    }
  ]
}
```

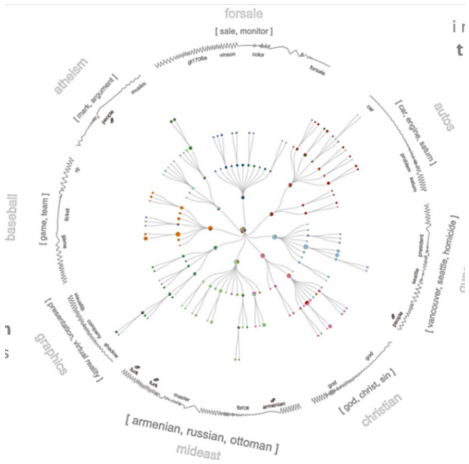
2. Circular layout with content inside

a. ring/donut
mark: “bar”, layout: “circular”


```
{
  "layer": [
    {
      "mark": "geoshape"
    },
    {
      "mark": "point",
      "encoding": {
        "x": {
          "field": "geo_x",
          "type": "quantitative"
        },
        "y": {
          "field": "geo_y",
          "type": "quantitative"
        }
      }
    },
    {
      "mark": "bar",
      "layout": "circular",
      "encoding": {
        "x": {
          "field": "hour",
          "type": "nominal"
        },
        "y": {
          "field": "flow_magnitude",
          "type": "quantitative",
          "aggregate": "sum"
        },
```

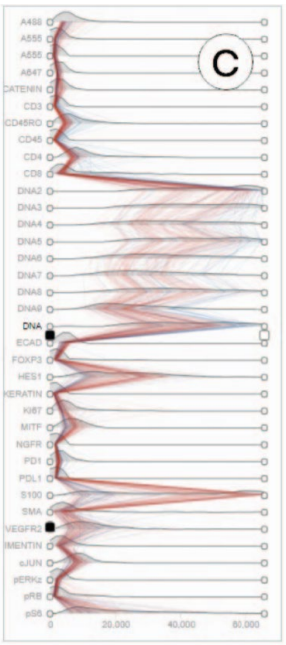
```
      "color": {
        "field": "flow_magnitude",
        "type": "quantitative",
        "aggregate": "mean"
      },
      "ring": {
        "field": "focus_interval",
        "type": "quantitative"
      }
    }
  },
  {
    "mark": "graph",
    "encoding": {
      "node_x": {
        "field": "OD_x",
        "type": "quantitative"
      },
      "node_y": {
        "field": "OD_y",
        "type": "quantitative"
      },
      "link": {
        "field": "OD_path",
        "type": "relation"
      },
      "link_color": {
        "field": "time",
        "type": "temporal"
      }
    }
  }
 ]
}
```

b. segments
facet:{layout: circular}




```
{
  "layer": [
    {
      "facet": {
        "layout": "circular",
        "sector": {
          "field": "topic",
          "type": "nominal"
        }
      },
      "spec": {
        “concat”: {
              “layout”: 'vertical'
            },
            spec:: [
          {
            "mark": "text",
            "encoding": {
              "text": "topic"
            }
          },
          {
            "mark": "text",
            "encoding": {
              "text": "sub-topic"
            }
          },
          {
            "mark": "line",
            "encoding": {
              "frequency": "noiseness"
            }
          }
        ]
      }
    },
```

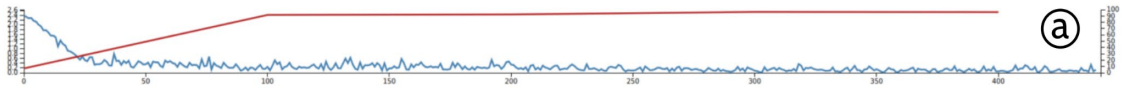
```
  {
    "nested": {
      "parent": {
        "mark": "tree",
        "encoding": {
          "node": {
            "field": "topic",
            "type": "nominal"
          },
          "link": {
            "field": "tree_hierarchy",
            "type": "relation"
          }
        }
      },
      "child": {
        "child_type": "configured",
        "canvas": "node",
        "configuration": {
          "mark": "arc",
          "encoding": {
            "theta": {
              "aggregate": "count",
              "type": "quantitative"
            },
            "color": {
              "field": "document-authors",
              "type": "nominal"
            },
            "size": {
              "field": "text_amount",
              "type": "quantitative"
            }
          }
        }
      }
    }
  }
]
}
```




```
{
  nested: {
    parent: {
      mark: 'line',
      encoding: {
        x: {
          field: 'feature_value',
          type: 'quantitative'
        },
        y: {
          field: 'feature',
          type: 'nominal'
        },
        color: {
          field: 'subset',
          type: 'nominal'
        }
      }
    },
    child: {
      child_type: 'configured',
      canvas: 'axis',
      configuration: {
        mark: 'area',
        encoding: {
          x: {
            field: 'feature_value',
            bin: true,
            type: 'quantitative',
            remark: 'square root scaling'
          },
```

```
        y: {
          aggregate: 'density',
          type: 'quantitative'
        }
      }
    }
  }
 }
}
```

3. Dual Axes



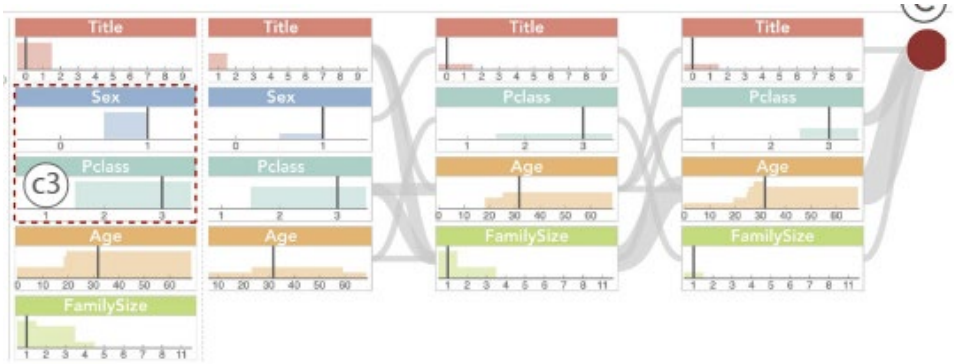
```
{
 layer: [
  {
   mark: 'line',
   encoding: {
    x: {
     field: 'step',
     type: 'nominal'
    },
    y: {
     field: 'accuracy',
     type: 'quantitative'
    }
   }
  },
  {
   mark: 'line',
   encoding: {
    x: {
     field: 'step',
     type: 'nominal'
    },
    y: {
     field: 'error',
     type: 'quantitative'
    }
   }
  }
 ],
 resolve: {
  scale: {
   y: 'independent'
  }
```

```
  }
}
```

Nested

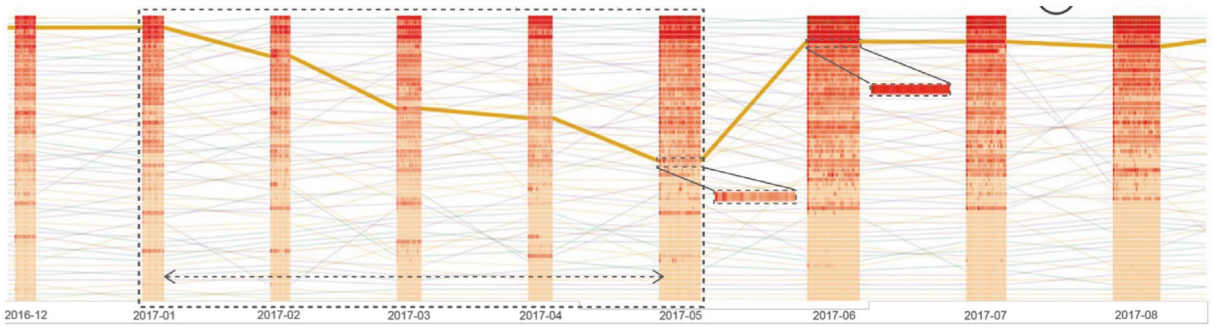
Specification:

```
nested: {
        parent:{},
        child:{
                “canvas”:”inner_circular_area/node”,
                "child_type": "configured/exemplified",
        }
}
```




```
{
  "nested": {
    "parent": {
      "mark": "sankey",
      "encoding": {
        "node": {
          "field": "attribute_type",
          "type": "nominal"
        },
        "link": {
          "field": "decision_path",
          "type": "relation"
        }
      }
    },
    "child": {
      "canvas": "node",
      "configuration": {
        "mark": "bar",
        "encoding": {
```

```
        "x": {
          "field": "attribute_value",
          "type": "quantitative"
        },
        "y": {
          "aggregate": "count",
          "type": "quantitative"
        }
      }
    }
  }
}
```

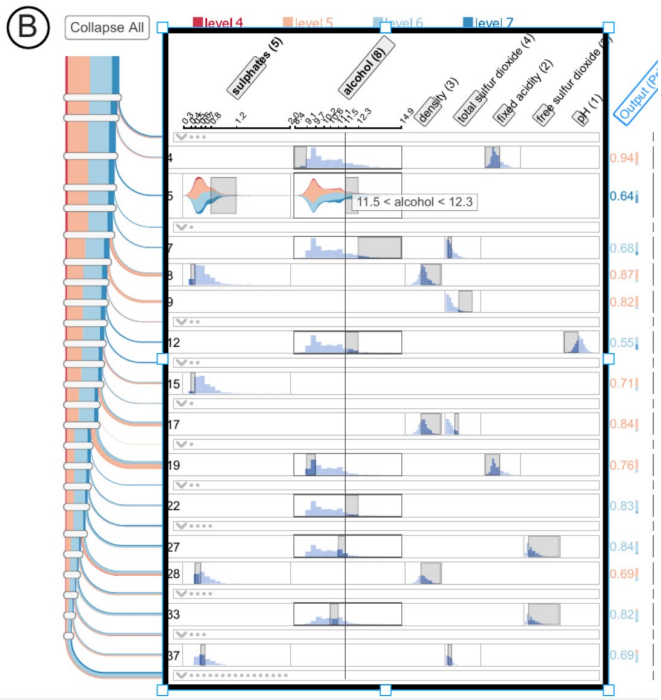



```
{
  nested: {
    parent: {
      mark: 'line',
      encoding: {
        x: {
          field: 'year_month',
          type: 'temporal'
        },
        y: {
          field: 'exchange',
          type: 'nominal'
        },
        color: {
          field: 'continent',
          type: 'nominal'
        }
      }
    },
    child: {
      canvas: 'axis',
      configuration: {
        mark: 'rect',
        encoding: {
          x: {
            field: 'year_month',
```

```
          type: 'temporal'
        },
        xOffset: {
          field: 'day',
          type: 'temporal'
        },
        y: {
          field: 'exchange',
          type: 'nominal'
        },
        width: {
          field: 'client',
          aggeragate: 'count',
          type: 'quantitative'
        },
        color: {
          field: 'transaction_volume',
          type: 'quantitative'
        }
      }
    }
  }
}
```

Conditions (OR)

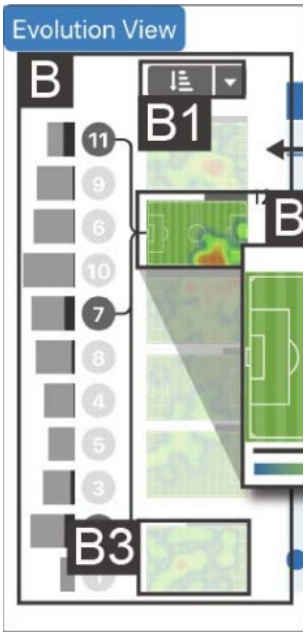
We introduce condition specifications to handle the situations that different visualizations are rendered by conditions.




```
{
  facet: {
    row: {
      field: 'feature_name',
      type: 'categorical',
      sort: 'importance_score'
    },
    column: {
      field: 'rule',
      type: 'ordinal'
    }
  },
  spec: {
    mark: 'bar',
    encoding: {
      x: {
        field: 'feature_value',
        type: 'quantitative',
        bin: true
      },
      y: {
        aggregate: 'count',
        type: 'quantitative'
      }
    }
  },
  alternative: {
    facet: {
      row: {
```

```
    field: 'feature_name',
    type: 'categorical',
    sort: 'importance_score'
  },
  column: {
    field: 'rule',
    type: 'ordinal'
  }
},
spec: {
  condition_1: {
    test: 'feature type is continuous',
    value: {
      mark: 'area',
      encoding: {
        x: {
          field: 'feature_value',
          type: 'quantitative',
          bin: true
        },
        y: {
          aggregate: 'count',
          type: 'quantitative'
        }
      }
    }
  },
  condition_2: {
    test: 'feature type is discrete',
    value: {
      mark: 'bar',
      encoding: {
        x: {
          field: 'feature_value',
          type: 'quantitative',
          bin: true
        },
        y: {
          aggregate: 'count',
          type: 'quantitative'
        }
      }
    }
  },
  condition_3: {
    test: 'a clause is using feature j',
    value: {
      mark: 'rect',
      encoding: {
        x: {
```

```
            field: 'start interval in the clause',
            type: 'quantitative'
          },
          x2: {
            field: 'end interval in the clause',
            type: 'quantitative'
          }
        }
      }
    }
  }
}
```



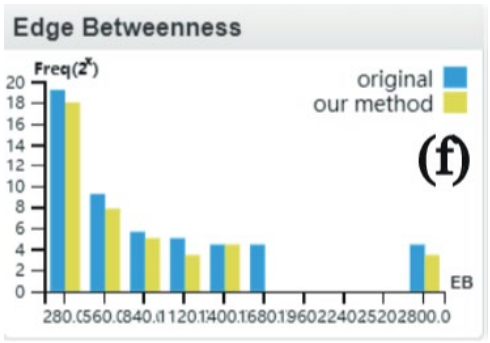

```
nested: {
  parent: {
    mark: 'graph',
    encoding: {
      node_left: {
        field: 'player',
        type: 'nominal'
      },
      node_right: {
        field: 'passing_pattern',
        type: 'nominal'
      },
      link: {
        field: 'player_involved_in_passing_pattern',
        type: 'relation'
      }
```

```
    }
  },
  child: {
    canvas: 'node',
    configuration: {
      condition_1: {
        test: 'node_left',
        value: {
          mark: 'bar',
          encoding: {
            x: {
              field: 'total_pass',
              type: 'quantitative'
            },
            y: {
              field: 'player',
              type: 'nominal'
            },
            color: {
              field: 'pass_in_passing_pattern',
              type: 'nominal',
              remark: 'When hovering on a pattern, a dark bar (Fig. 4 (B)) is presented to show
the number of a player’s passes in that passing pattern'
            }
          }
        }
      },
      condition_2: {
        test: 'node_right',
        value: {
          concat: {
            layout: 'vertical'
          },
          spec: [
            {
              mark: 'bar',
              encoding: {
                x: {
                  field: 'pass_in_passing_pattern',
                  aggregate: 'count',
                  type: 'quantitative'
                }
              }
            },
            {
              mark: 'surface',
              encoding: {
                x: {
                  field: 'soccer_pitch_dim_1',
                  type: 'quantitative'
```

```
            },
            y: {
              field: 'soccer_pitch_dim_1',
              type: 'quantitative'
            },
            surface: {
              field: 'start_or_end_position',
              aggregate: 'count',
              type: 'quantitative'
            }
          }
        }
      ]
    }
  }
}
```

Visualization Types

Type	Remarks	Examples

Bar chart	x, y, color, width, xOffset	
		{
		mark: 'bar',
		encoding: {
		x: {
		field: 'EB',
		type: 'quantitative',
		bin: 'true'
		},
		y: {
		field: 'frequency',
		type: 'quantitative'
		},
		xOffset: {
		field: 'method_type',
		type: 'nominal'
		},
		color: {
		field: 'method_type',
		type: 'nominal'
		}
		}
		}

Bar chart (circular)		
		{
		"mark": "bar",
		"layout": "circular",
		"encoding": {
		"x": {
		"field": "hour",
		"type": "nominal"
		},
		"y": {
		"field": "flow_magnitude",
		"type": "quantitative",
		"aggregate": "sum"
		},
		"color": {
		"field": "flow_magnitude",
		"type": "quantitative",
		"aggregate": "mean"
		},
		"ring": {
		"field": "focus_interval",
		"type": "quantitative"
		}
		}
		}

Heatmap		{
		"mark": "rect",
		"encoding": {
		"y": {
		"field": "flow_1",
		"type": "nominal"
		},
		"x": {
		"field": "flow_2",
		"type": "nominal"
		},
		"color": {
		"field": "flow_coverage",
		"type": "quantitative"
		}
		}
		}

Surface graph		{
		"mark": "surface",
		"encoding": {
		"x": {
		"field": "dimension_1",
		"type": "quantitative"
		},
		"y": {
		"field": "dimension_2",
		"type": "quantitative"
		},
		"surface": {
		"field": "probability_density",
		"type": "quantitative"
		}
		}
		}

Contour graph	A contour line of a function of two variables is a curve along which the function has a constant value, so that the curve joins points of equal value. https://en.wikipedia.org/wiki/Contour_line	{
		"mark": "contour",
		"encoding": {
		"x": {
		"field": "dimension_1",
		"type": "quantitative"
		},
		"y": {
		"field": "dimension_2",
		"type": "quantitative"
		},
		"contour": {
		"field": "value",
		"type": "quantitative"
		}
		}
		}

Scatter plot		{
		"mark": "point",
		"encoding": {
		"x": {
		"field": "dimension_1",
		"type": "quantitative"
		},
		"y": {
		"field": "dimension_2",
		"type": "quantitative"
		}
		}
		}

Scatter (circular)	In some cases, scatterplots are organized by radius and angle in a circular layout.	{
		mark: 'point',
		layout: 'circular',
		encoding: {
		radius: {
		field: 'jacard index',
		type: 'quantitative'
		},
		theta: {
		field: 'category',
		type: 'nominal'
		}
		}
		}

Line chart		{
		"mark": "line",
		"encoding": {
		"x": {
		"field": "epoch",
		"type": "quantitative"
		},
		"y": {
		"field": "loss",
		"type": "quantitative"
		},
		"color": {
		"field": "loss_type",
		"type": "nominal"
		}
		}
		}

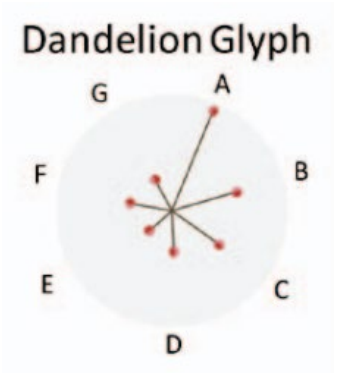
PCP		{
		"mark": "line",
		"encoding": {
		"x": {
		"field": "feature type",
		"type": "nominal"
		},
		"y": {
		"aggregate": "feature value",
		"type": "quantitative"
		}
		}
		}

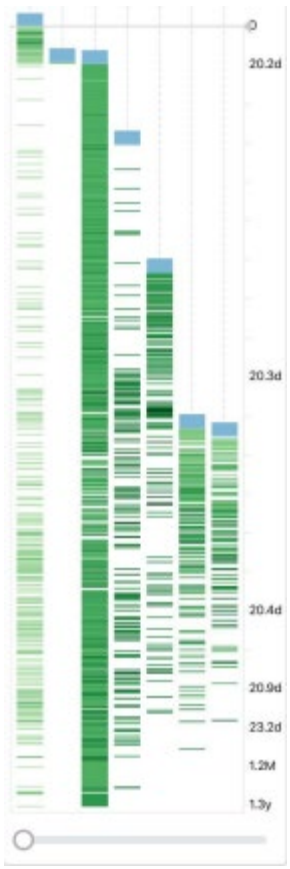
Matrix		{
		"mark": "rect",
		"encoding": {
		"y": {
		"field": "flow_1",
		"type": "nominal"
		},
		"x": {
		"field": "flow_2",
		"type": "nominal"
		},
		"color": {
		"field": "flow_coverage",
		"type": "quantitative"
		}
		}
		}

Radar chart		{
		"mark": "radar",
		"encoding": {
		"theta": {
		"field": "metric_type",
		"type": "nominal"
		},
		"radius": {
		"field": "metric_value",
		"type": "quantitative"
		},
		"color": {
		"field": "method",
		"type": "nominal"
		}
		}
		}

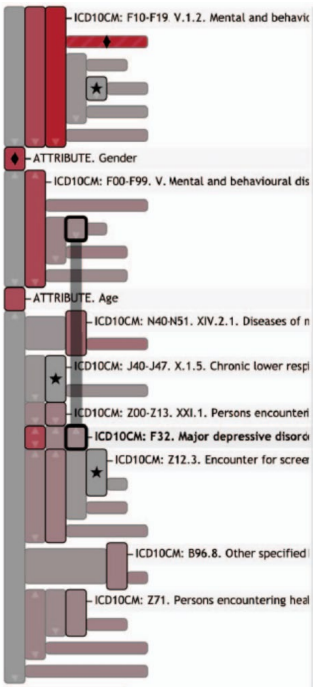
Area chart		{
		"mark": "area",
		"encoding": {
		"x": {
		"field": "epoch",
		"type": "quantitative"
		},
		"y": {
		"aggregate": "sum",
		"field": "count"
		},
		"color": {
		"field": "action/reward_type",
		"type": "nominal"
		}
		}
		}

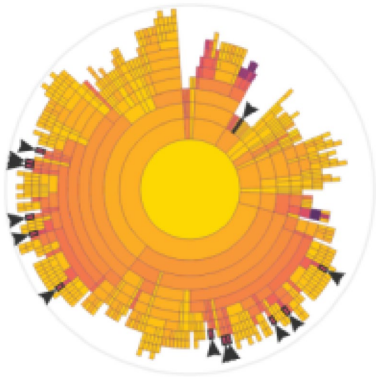
Box plot		{
		"mark": "boxplot",
		"encoding": {
		"y": {
		"field": "word",
		"type": "nominal"
		},
		"x": {
		"field": "activation_distribution",
		"type": "quantitative"
		}
		}

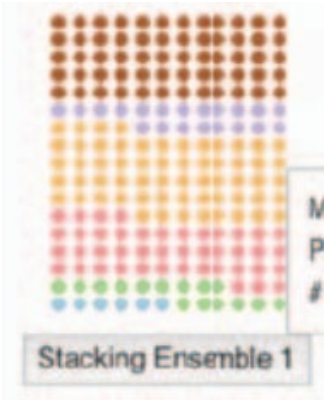
Dandelion graph		
		{
		"mark": "dandelion",
		"layout": "circular",
		"encoding": {
		"theta": {
		"field": "attribute",
		"type": "nominal"
		},
		"radius": {
		"field": "group",
		"type": "nominal"
		}
		}
		}

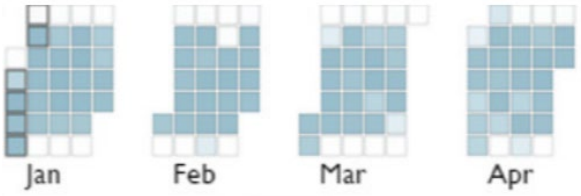
Strip graph		
		{
		mark: 'tick',
		encoding: {
		y: {
		field: 'delay time of reposting',
		type: 'temporal'
		},
		x: {
		field: 'player',
		type: 'nominal'
		},
		color: {
		field: 'post type',
		type: 'nominal'
		}
		}
		}

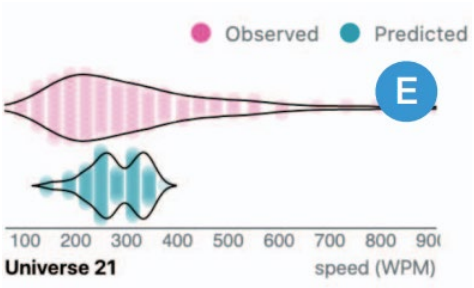
Word cloud		{
		"mark": "word_cloud",
		"encoding": {
		"word": {
		"field": "word",
		"type": "nominal"
		},
		"size": {
		"field": "TF-IDF value",
		"type": "quantitative"
		}
		}
		}

Icicle graph	Special cases of node-link diagrams	
		{
		mark: 'icicle',
		encoding: {
		node: {
		field: 'shifts',
		type: 'node',
		color: {
		field: 'weighted distance',
		type: 'quantitative'
		}
		},
		link: {
		field: 'hierarchy',
		type: 'relation'
		}
		}
		}

sunburst		
		{
		mark: 'sunburst',
		encoding: {
		node: {
		field: 'node',
		type: 'others',
		encoding: {
		color: {
		field: 'metric',
		type: 'quantitative',
		remark: 'Darker colors indicate stronger metrics'
		},
		width: {
		field: 'count',
		type: 'quantitative',
		remark: 'The width of segments communicates the number of leaf nodes, with each leaf having equal weight, and all charts are sorted identically to support comparisons.'
		}
		}
		},
		link: {
		field: 'global pattern hierarchy',
		type: 'relation'
		}
		}
		}

Unit visualization		
		{
		mark: 'unit',
		encoding: {
		unit: {
		field: 'model',
		type: 'node'
		},
		color: {
		field: 'model type',
		type: 'quantitative'
		}
		}
		}

Calendar		
		{
		mark: 'calendar',
		encoding: {
		x: {
		field: 'month',
		type: 'temporal'
		},
		unit: {
		field: 'day',
		type: 'node'
		},
		color: {
		field: 'number of products',
		type: 'quantitative'
		}
		}
		}

Violin plot	A violin plot is a method of plotting numeric data. It is similar to a box plot, with the addition of a rotated kernel density plot on each side. https://en.wikipedia.org/wiki/Violin_plot	
		{
		mark: 'boxplot',
		encoding: {
		yoffset: {
		field: 'observed/predicted',
		type: 'nominal'
		},
		color: {
		field: 'observed/predicted',
		type: 'nominal'
		},
		x: {
		field: 'speed',
		type: 'quantitative'
		}
		}
		}

Venn		{
		mark: 'venn',
		encoding: {
		set: {
		field: 'element-to-group',
		type: 'relation'
		}
		}
		}

Tree		{
		  mark: 'tree',
		  encoding: {
		  node: {
		  field: 'shifts',
		  type: 'node',
		  encoding: {
		  color: {
		  field: 'weighted distance',
		  type: 'quantitative'
		  }
		  }
		  },
		  link: {
		  field: 'hierarchy',
		  type: 'relation'
		  }
		  }
		}

Treemap		{
		  mark: 'treemap',
		  encoding: {
		  node: {
		  field: 'shifts',
		  type: 'node',
		  encoding: {
		  color: {
		  field: 'weighted distance',
		  type: 'quantitative'
		  }
		  }
		  },
		  link: {
		  field: 'hierarchy',
		  type: 'relation'
		  }
		  }
		}

graph		{
		"mark": "graph",
		"encoding": {
		"node": {
		"field": "plan",
		"type": "node"
		},
		"link": {
		"field": "difference between plans",
		"type": "relation"
		}
		}
		}

sankey		{
		"mark": "sankey",
		"encoding": {
		"node": {
		"field": "attribute_type",
		"type": "node"
		},
		"link": {
		"field": "decision_path",
		"type": "relation"
		}
		}
		}