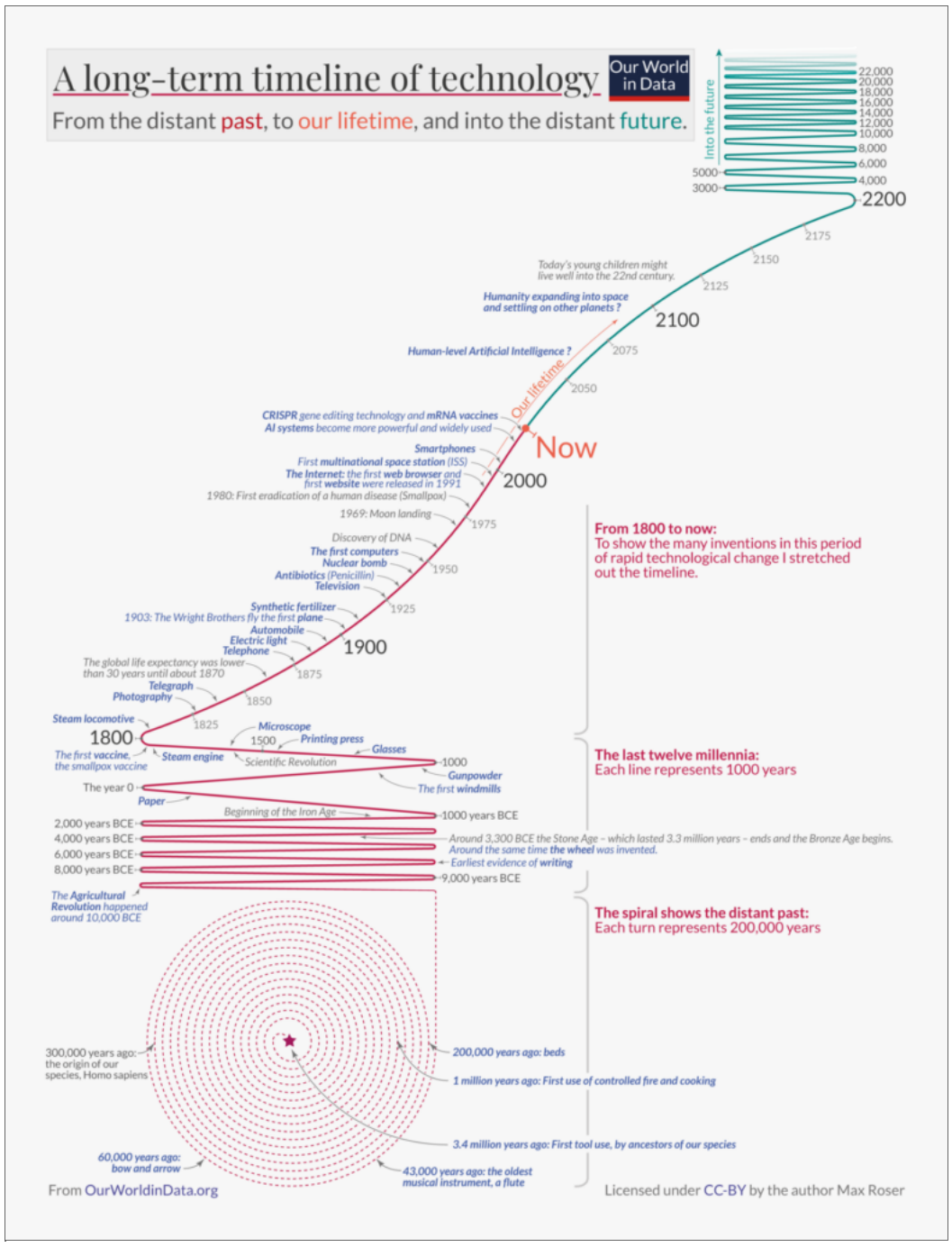
Lecture 7: Tabular Data, Network Data

Today's Visualization



Assignment

Ooooooops, I forgot to put a submission point up on Blackboard.
Fixed now, redesign assignment extended until next week.

Tabular Data

Taxonomy of tabular visualizations

Munzner classifies visualizations of tabular data by 4 core design choices to be made:

1. How values are expressed
2. How marks are separated, ordered, aligned
3. Which axis orientation is used
4. Whether the layout is dense or space-filling

Key to Munzner's analysis is the key/value semantics introduced in Chapter 2. Depending on data types of keys, different choices are needed.

Express: 1 quantitative value

A single quantitative value (for instance activation times for a neural spike train, arrival times of cars or race competitors, …)

- Code

Express: 2 quantitative values

The **scatterplot** is a widely spread idiom for expressing 2 primary quantitative variables, possibly adding any number of additional attributes in secondary visual channels.

▶ Code

Separate/Sort/Align: Bi-stacked bar charts

- Code

Separate / Sort / Align

Sorted bar charts

- Code

Dot-charts, line-charts, bar-charts

▶ Code

Dot-charts, line-charts and bar-charts communicate very similar data types in very similar ways. Crucial distinction in how the viewer perceives the chart:
Lines imply connectivity between the individual observations - leads the eye to look for trends, even if there is no inherent connectivity between the corresponding key attributes.
Bars communicate quantity primarily with an area mark - leads the eye to lend attention proportional to the visual impact of the bar. This is one case where for instance truncated axes lead to misleading charts.

Dot-charts, line-charts, bar-charts

Resulting recommendation:

- Dot-charts for quantitative vs. nominal, truncated axis not inherently dishonest
- Line-charts for quantitative vs. ordinal
- Bar-charts for quantitative vs. nominal, truncated axis inherently dishonest

Perceptually optimal line-charts

▶ Code

Cleveland and McGill (1987) study perceptual accuracy of slope judgements, approaching the question of which slopes are most accurately read for the shape of a curve.
Historically, Karsten (1923) suggested an aspect ratio for graphs of 2:3, American Standards Association (1938) suggested 1:$\sqrt{2}$, and American Standards Institute (1979) suggested 3:4.
Even if we decide to let the data control the choice of aspect ratio, different writers have different suggestion for which angles might be optimal: Von Huhn (1931) “somewhere between 30^o and 45^o”, Weld (1947) 35^o to 45^o, Hall (1958) 30^o to 60^o, Bertin (1967, 1983) 70^o.

Perceptually optimal line-charts

- Code

In perceptural accuracy experiments on judging similarity between adjacent slanted lines, Cleveland and McGill established the highest accuracy (within a span of 0^o - 60^o) to be right near slopes of 45^o.
As a result, one recommendation for line plots is to make either the median absolute slope or the average absolute slope of the line segments in the line plot be close to 45^o: to **bank to 45^o**.
R has support for automatically computing the resulting aspect ratios in the **`ggthemes::bank_slopes`** function, while in Python you may have to compute by hand or *eyeball* the aspect ratio.

More on line plots

The need to accurately judge shapes and slopes of line graphs also contributes to recommendations on axis truncation.

- Code

More important than including a (possibly arbitrary) 0 on the axis is to **avoid non-varying ink** in the line chart.

Many Attributes: SPLOM and Parallel Coordinates

SPLOM - grid of pairwise scatterplots
Parallel Coordinates - each attribute on an axis of its own, lines connect attributes that belong to the same unit.

- Code

Radial Layouts

The `R` help files has the following to say in their description of the `pie` command to draw pie-charts:

> Pie charts are a very bad way of displaying information. The eye is good at judging linear measures and bad at judging relative areas. A bar chart or dot chart is a preferable way of displaying this type of data.
> Cleveland (1985), page 264: “Data that can be shown by pie charts always can be shown by a dot chart. This means that judgements of position along a common scale can be made instead of the less accurate angle judgements.” This statement is based on the empirical investigations of Cleveland and McGill as well as investigations by perceptual psychologists.

Radial Layouts

More compact representation than a pie chart, as well as more accurate visual channels, comes with normalized stacked bar charts.

- Code

Radial Layouts

Radial Layouts **excel** at showing periodic phenomena.

Florence Nightingale, Public domain, via Wikimedia Commons
Note: **There are issues with this graph**

Facets

Facets

Facets are arrangements of frames containing graphics, often so that the arrangement itself carries some information about the data or its structure.
We have seen some examples already.

- Code

Algebra of Facets

Wilkinson develops an entire algebra of facet specifications. `ggplot2` can comfortably handle Wilkinson's `*` and `+` operators, and can be coaxed into handling `/` with added packages and some hands-on work.

- Code

Algebra of Facets

Vega and Altair can currently primarily handle the * operator

- Code

Algebra of Facets

Vega and Altair can currently primarily handle the * operator

- Code

Network Data

Data Representation

Networks are graphs (possibly trees), and there are several choices for data structures to hold graph data:

Adjacency List

Vertices are objects, with each vertex containing a list of its adjacent vertices.

Edges are implicitly encoded in these adjacency lists.

Incidence List

Vertices are objects, with each vertex containing a list of its incident edges.

Edges are objects, with each edge containing a list of its incident vertices.

Adjacency Matrix

2-dimensional matrix with rows representing source vertices, columns representing target vertices, and entries non-zero if an edge is present.

Can encode weights or multiplicities of edges, but all additional attributes of either edges or vertices has to be stored externally.

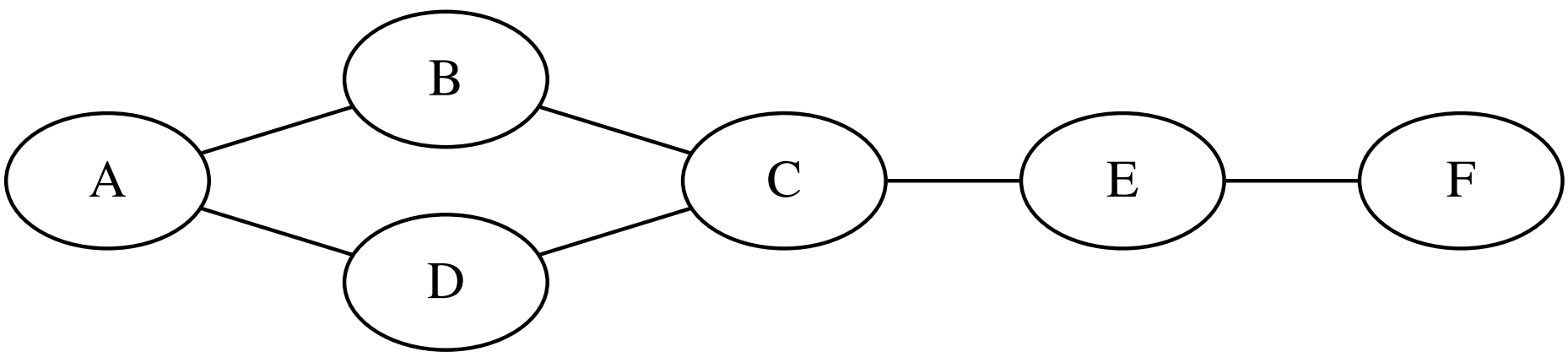
Incidence Matrix

2-dimensional matrix with rows representing vertices and columns representing edges.

Entries are non-zero if a vertex connects to an edge.

Data Representation

▶ Code



▼ Code

```
# Adjacency List
graph = {
  "A": ["B", "D"],      "B": ["A", "C"],
  "C": ["B", "D", "E"], "D": ["A", "C"],
  "E": ["C", "F"],      "F": ["E"]
}
```

▼ Code

```
# Adjacency Matrix
# A B C D E F
  0 1 0 1 0 0 # A
  1 0 1 0 0 0 # B
  0 1 0 1 1 0 # C
  1 0 1 0 0 0 # D
  0 0 1 0 0 1 # E
  0 0 0 0 1 0 # F
```

▼ Code

```
# Incidence List
graph = {
  "vertices": {
    "A": ["AB", "AD"], "B": ["AB", "BC"],
    "C": ["BC", "CD", "CE"], "D": ["AD", "CD"],
    "E": ["CE", "EF"], "F": ["EF"]
  },
  "edges": {
    "AB": ["A", "B"], "AD": ["A", "D"],
    "BC": ["B", "C"], "CD": ["C", "D"],
    "CE": ["C", "E"], "EF": ["E", "F"]
  }
}
```

▼ Code

```
# Incidence Matrix
# AB AD BC CD CE EF
   1  1  0  0  0  0  # A
   1  0  1  0  0  0  # B
   1  0  1  1  1  0  # C
   1  1  0  1  0  0  # D
   1  0  0  0  1  1  # E
   1  0  0  0  0  1  # F
```

Data Representation

For a graph with V the number of vertices and E the number of edges:

Task	Adjacency List	Adjacency Matrix	Incidence Matrix
Store Graph (space)	$O(V+E)$	$O(V^2)$	$O(V \cdot E)$
Add Vertex (time)	$O(1)$	$O(V^2)$	$O(V \cdot E)$
Add Edge (time)	$O(1)$	$O(1)$	$O(V \cdot E)$
Remove Vertex (time)	$O(E)$	$O(V^2)$	$O(V \cdot E)$
Remove Edge (time)	$O(V)$	$O(1)$	$O(V \cdot E)$
Adjacency query (time)	$O(V)$	$O(1)$	$O(E)$

Visual Idioms

7 encoding idioms

a. Vertical Node-Link

b. Icicle

c. Radial Node-Link

d. Concentric Circles

e. Nested Circles

f. Treemap

g. Indented Outline

Important software packages

- Graphviz - several good layout algorithms, decent file format for specifying graph structures
- D3.js - excellent and easy to use force-directed placement layouts
- Gephi - graph visualization workbench
- networkx - graph computation (and some layout) in Python