Data Visualization

Assoc. Prof. Dorien Herremans

50.038 Computational Data Science

SEMMA Methodology

1

Sample

Extract representative datasets for analysis

2

Explore

Visualize & understand patterns in data

3

Modify

Transform features & engineer variables

4

Model

Apply algorithms to fit predictive models

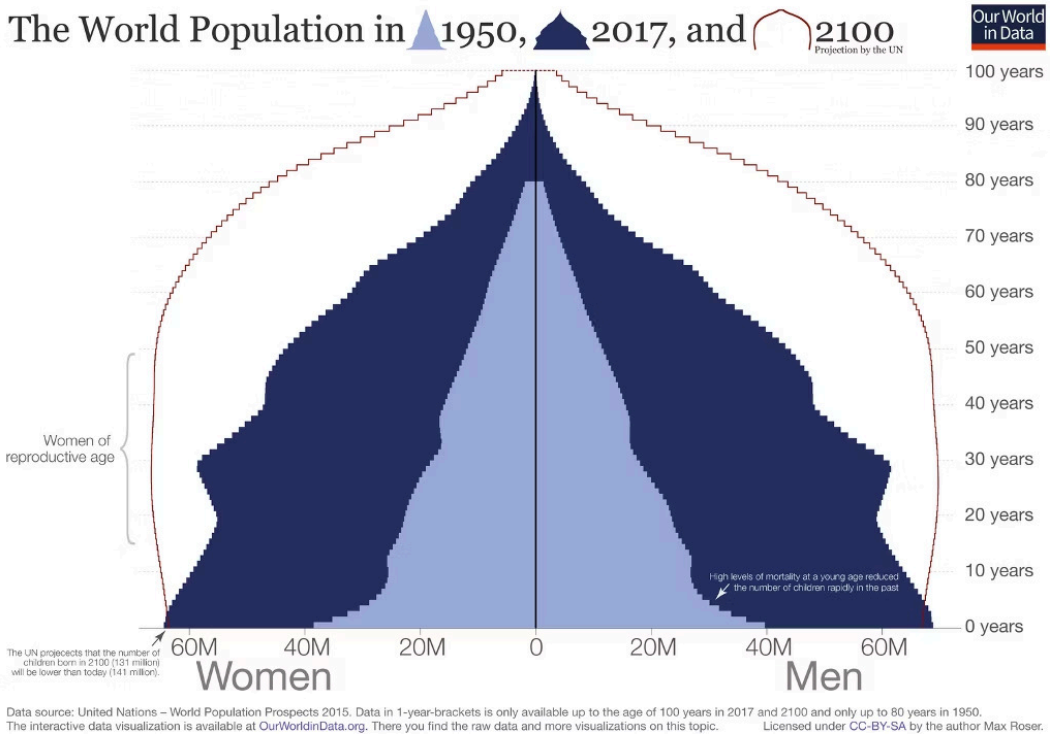
5

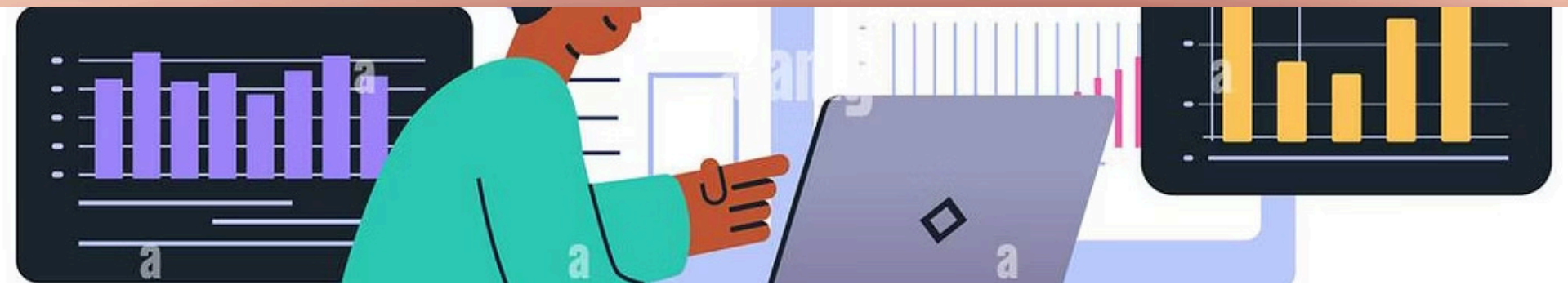
Assess

Evaluate reliability & compare performance

A picture says more...

This goes for graphs as well...





Why Visualize Data?

Data Analysis

Understand patterns, detect anomalies, **derive** insights

Requires comprehensiveness

Communication

Simplify complex information, tell compelling stories

Requires clarity & focus

Watch: The Art of Data Storytelling

Early Data Visualization

Pre-1800s: From Cave Walls to Census Records

Cave Paintings

c. 30,000 BC: Early humans tracked resources through visual marks

Egyptian Records

3000 BC: Hieroglyphic census data recorded population & harvests

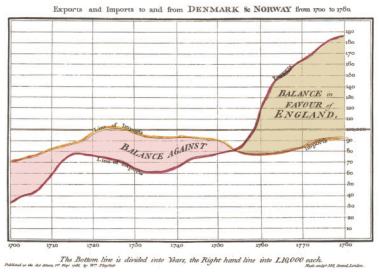
Medieval Maps

1100s-1400s: Symbols showed geographical & economic data

William Playfair: Father of Modern Charts

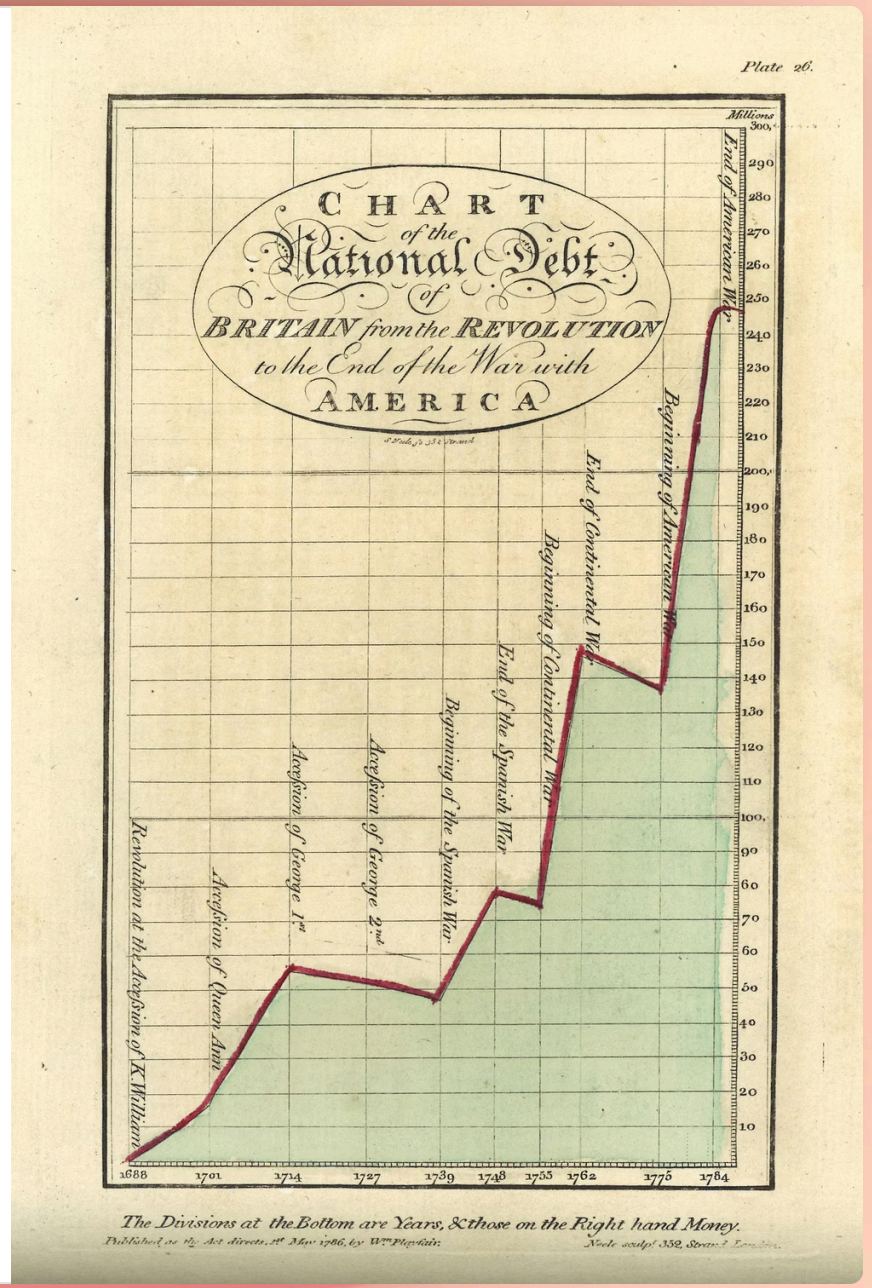
1786-1801

1. **1786: Bar Charts**
   Compare categories visually
2. **1786: Line Charts**
   Show economic trends over time
3. **1801: Pie Charts**
   Visualize proportions of Britain's economy



← Exports to and from Norway & Denmakr from 1700 to 1780 [balance against vs in favour]

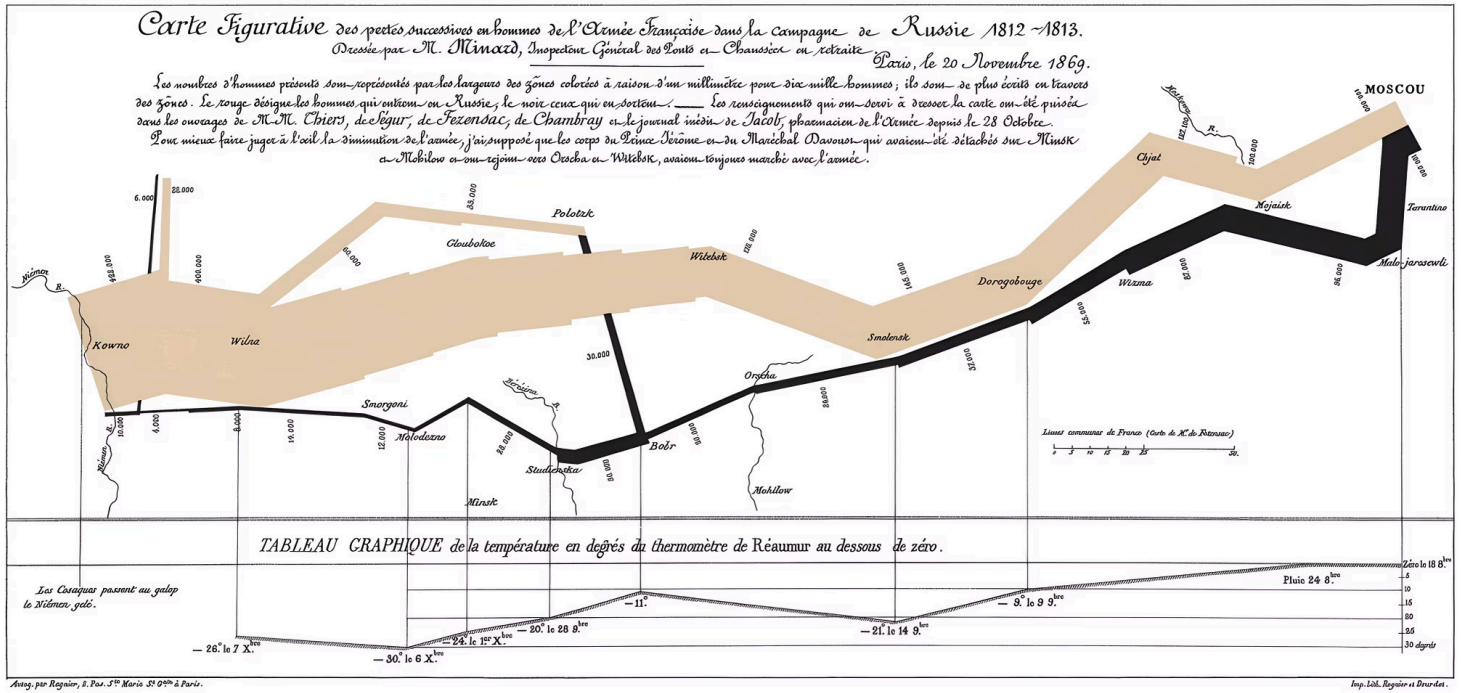
Legacy: Playfair's innovations **remain core tools** in modern data visualization, emphasizing **visual storytelling**



Charles Minard's Masterpiece

Napoleon's 1812 Russian Campaign

"It may well be the best statistical graphic ever drawn." — *Edward Tufte*



5

Variables Encoded

Army size, location, direction, dates, temperature

422K

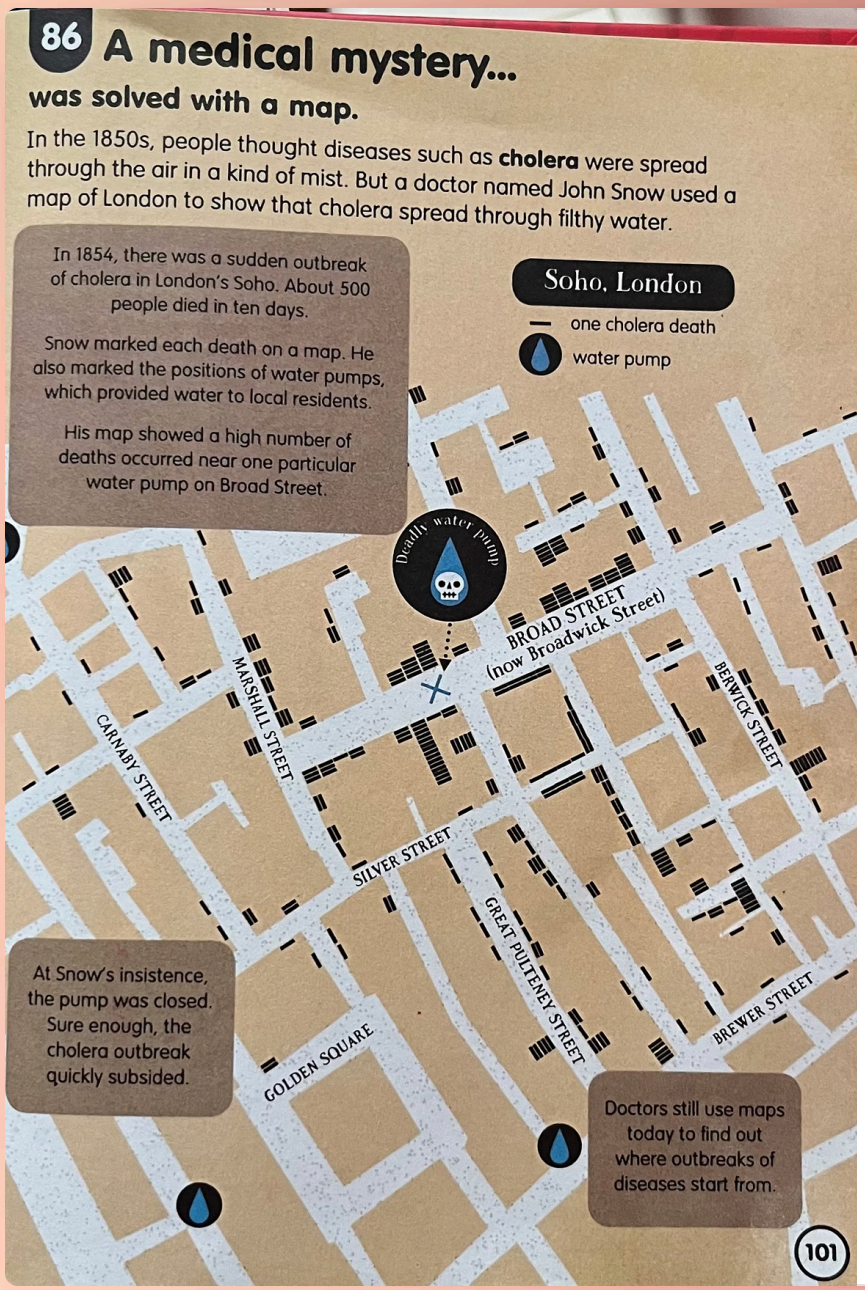
Initial Army

Soldiers who began the march to Moscow

10K

Survivors

Only 10,000 returned from the campaign



John Snow's Cholera Map

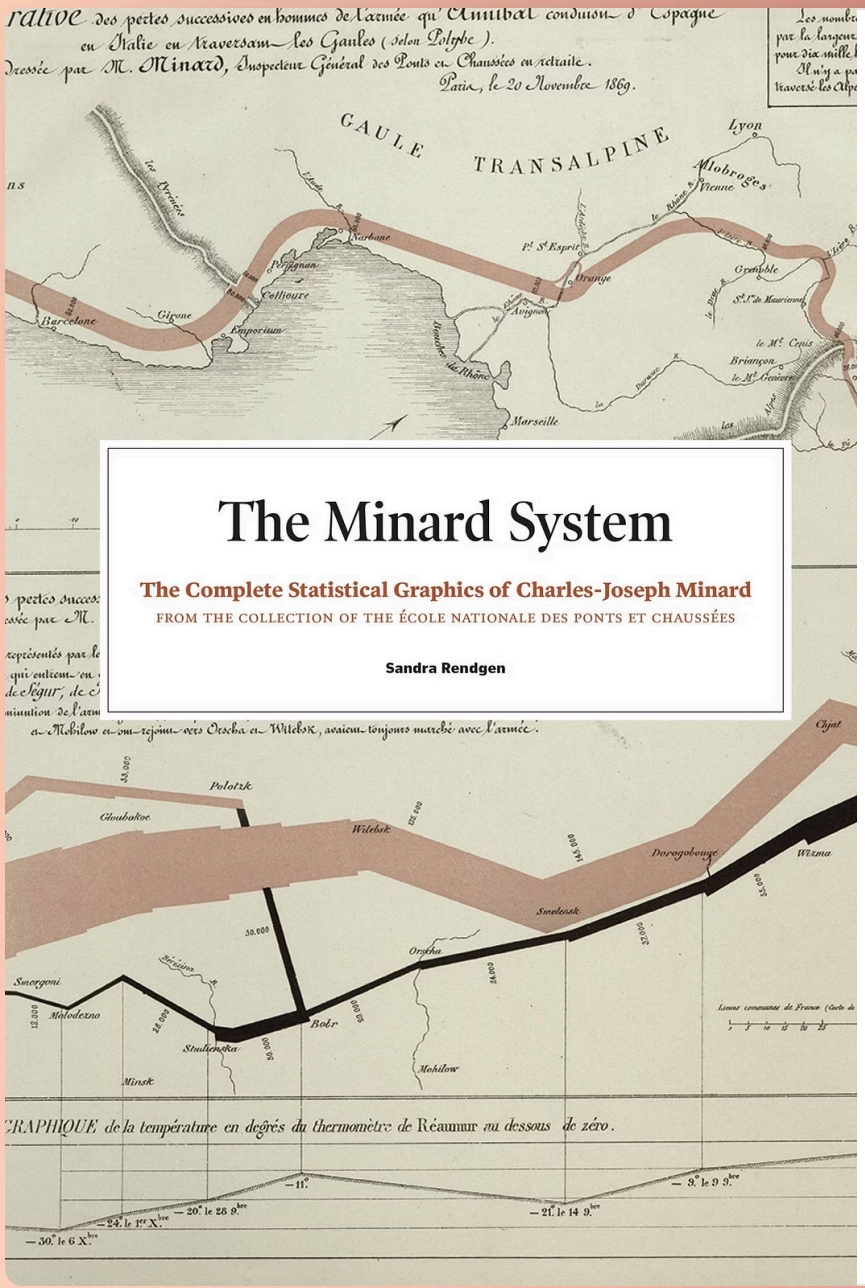
London, 1854

Breakthrough Discovery

Dr. John Snow used a **dot map** to track cholera cases, clustering around contaminated water pumps

1	**Each dot represented a cholera death**
2	**Pattern revealed contaminated water source**
3	**Proved waterborne transmission theory**
4	**Led to removal of Broad Street pump handle**

ⓘ Revolutionary use of spatial data visualization for public health



The Minard System

The Complete Statistical Graphics of Charles-Joseph Minard

FROM THE COLLECTION OF THE ÉCOLE NATIONALE DES PONTS ET CHAUSSÉES

Sandra Rendgen

The Evolution Continues

From Historical Foundations to Modern Innovation

01

Historical Impact

Data visualization became essential **tools for policy-making & science**

02

Complex Storytelling

Multi-variable visualizations like Minard's map proved **data could tell rich stories**

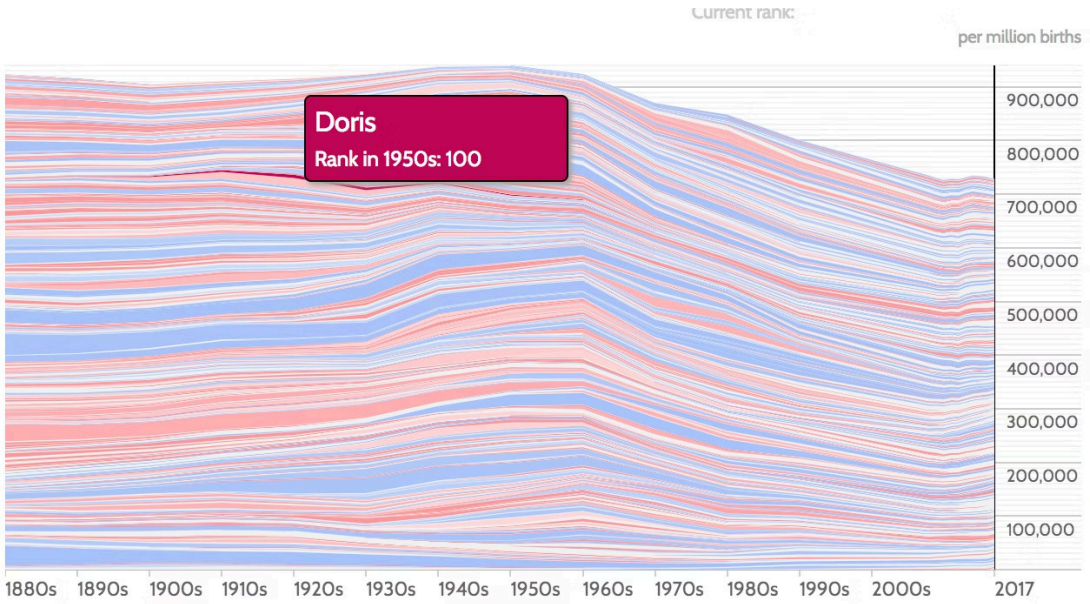
03

Modern Era

- Design principles
- Big data availability
- Live dashboards & real-time updates
- Interactive exploration
- AI-powered insights

Baby Names Voyager

Interactive Data Exploration (Wattenberg et al. 2005)



Explore 100+ years of U.S. baby name trends through dynamic, interactive visualization

Try it: **Baby Names Voyager**

Baby Names Voyager

Interactive Data Exploration (Wattenberg et al. 2005)

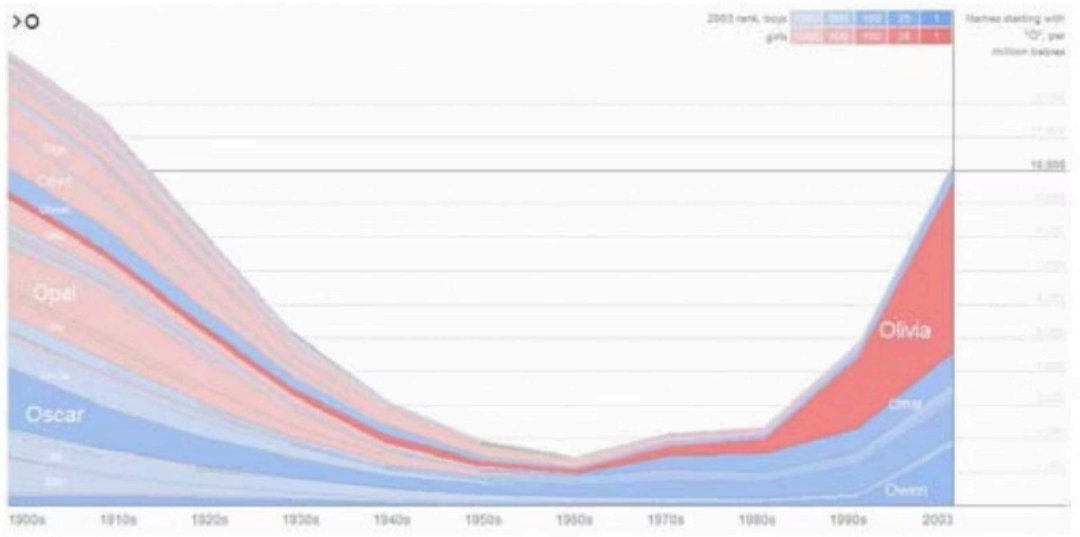


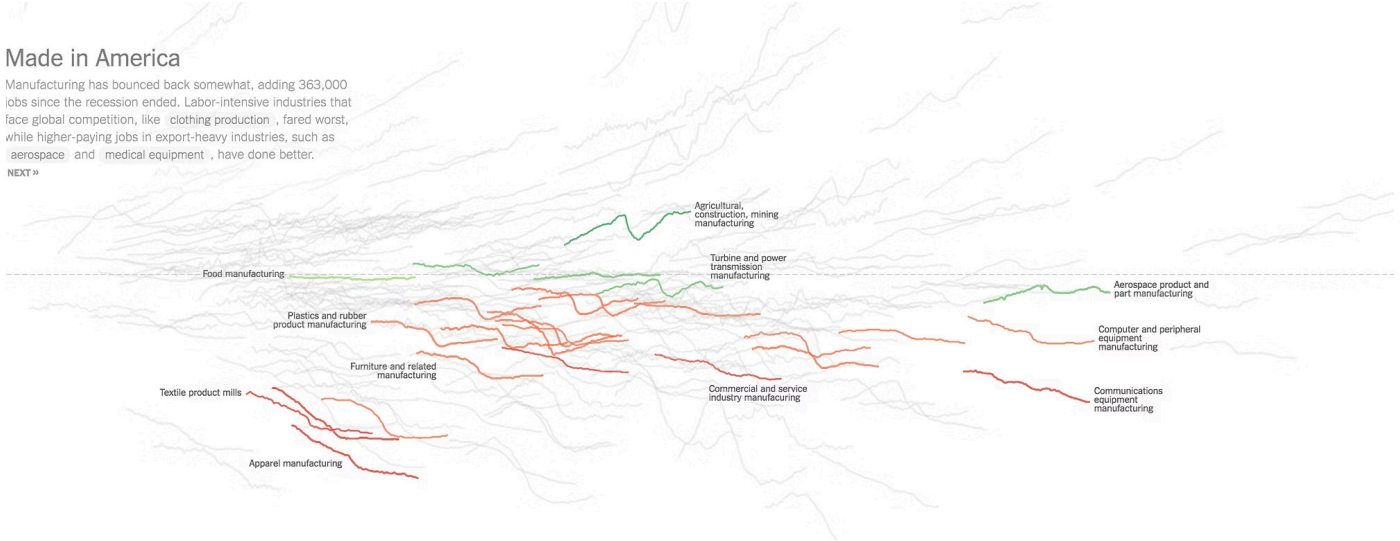
Figure 2. Names beginning with O

Explore 100+ years of U.S. baby name trends through dynamic, interactive visualization

Try it: **Baby Names Voyager**

How the Recession Reshaped the Economy

NY Times Interactive: 255 Industry Charts (2014)



Each line shows job changes for a specific industry over 10 years (2004-2014)

Explore the full interactive

Why Interactive Visualizations?

Enhanced Exploration

Users investigate multiple dimensions without needing separate charts for each view

Improved Engagement

Interactive elements make complex data more intuitive and memorable

Better Decisions

Real-time dashboards enable immediate analysis of live data updates

Handle Complexity

Multidimensional data becomes interpretable through user-controlled interaction

Best Practices

01

Define Your Question

What information are you communicating? What's your goal?

02

Assess Available Data

What datasets do you have? What level of detail exists?

03

Supplement Strategically

How can you enhance your data with external sources?

04

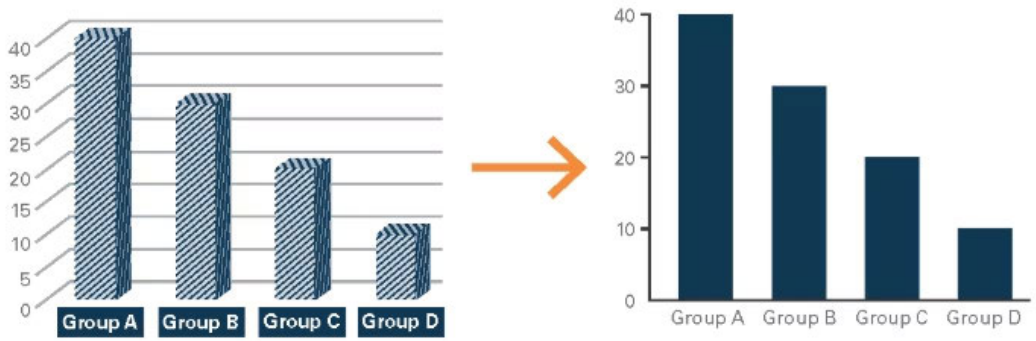
Apply Core Principles

Follow established design guidelines (coming next)

Reduce Clutter

Simplicity Enhances Understanding

> "Perfection is achieved not when there is nothing more to add, but when there is nothing left to take away."



Eliminate gratuitous features

Does each element reveal new information?

Reduce chart junk

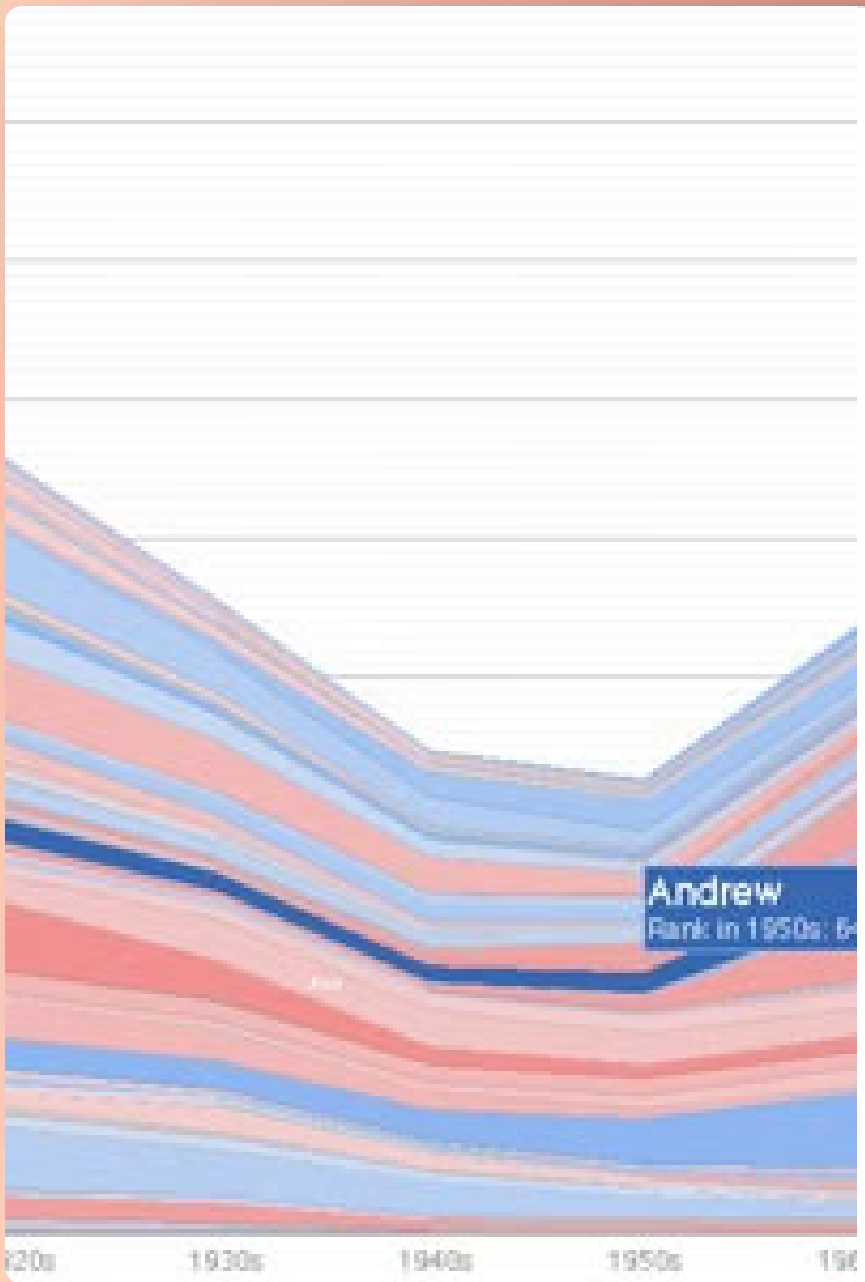
Remove unnecessary decorations, shadows, 3D effects

Minimize table clutter

Use white space strategically, simplify gridlines

Focus attention

Every visual element should serve a purpose



Interactive Chart Design

Progressive Disclosure of Information

Static Charts

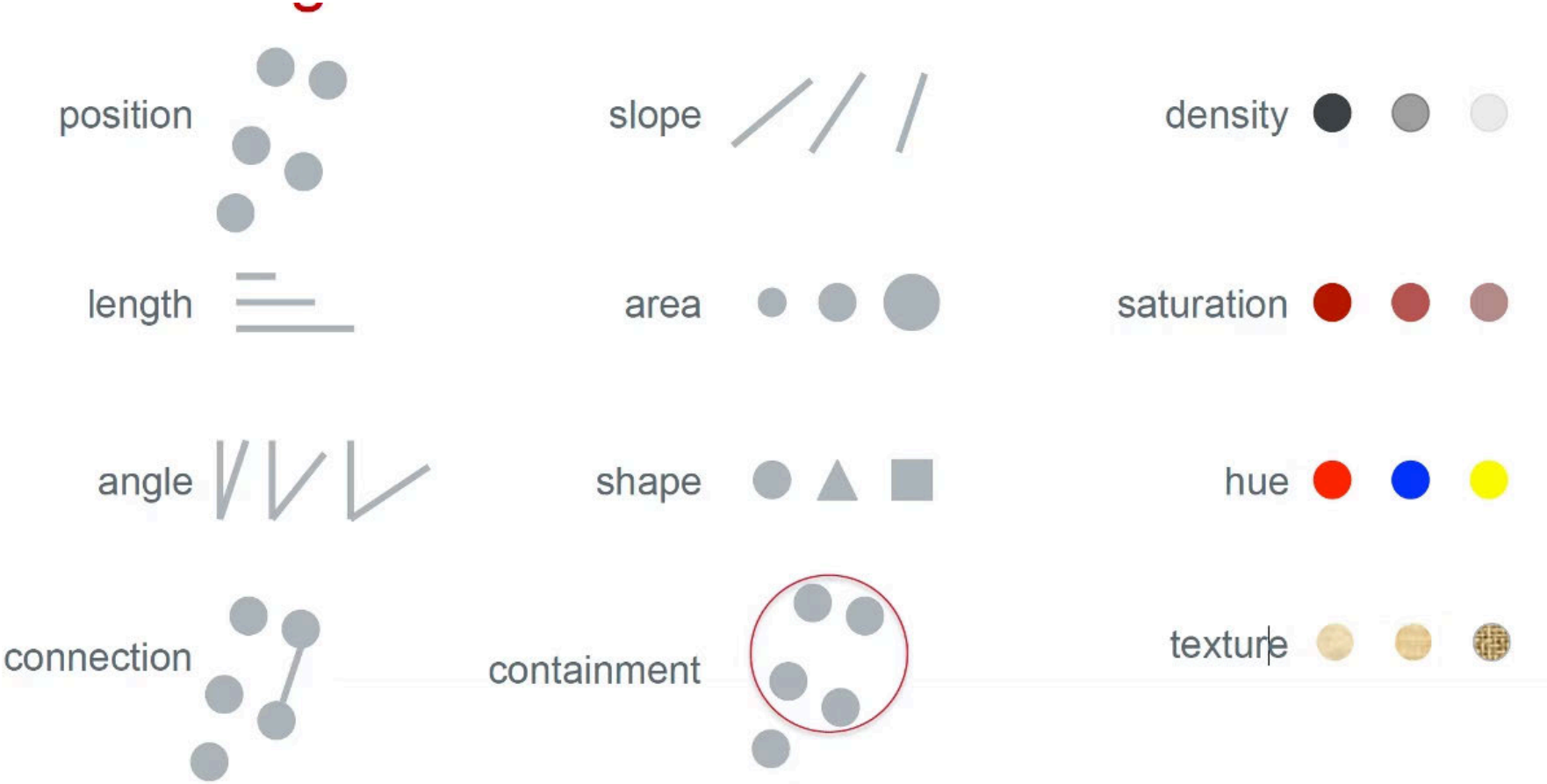
- Show all information upfront
- Risk of overwhelming viewers
- Limited by space constraints

Interactive Charts

- Keep initial view simple
- Dynamically reveal details on demand
- Show navigation-critical info only

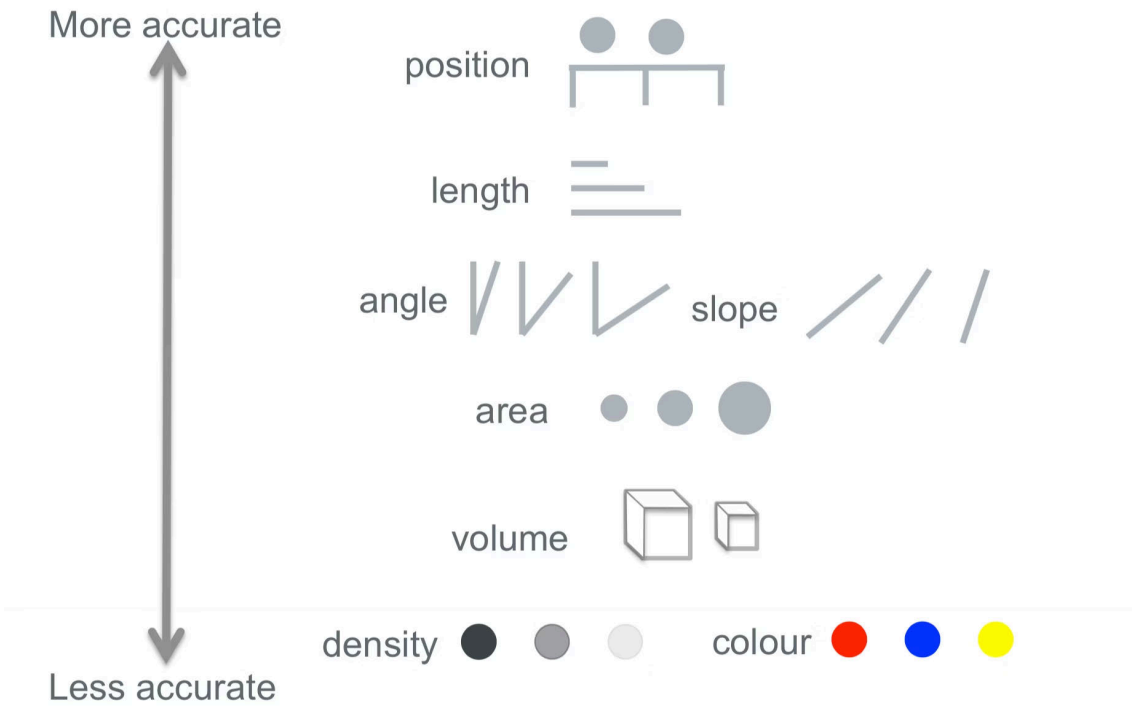
Interactive design allows you to **hide complexity** and reveal it progressively based on user interest

Encoding schemes



Accuracy of Perception

Cleveland & McGill's Hierarchy (1985)

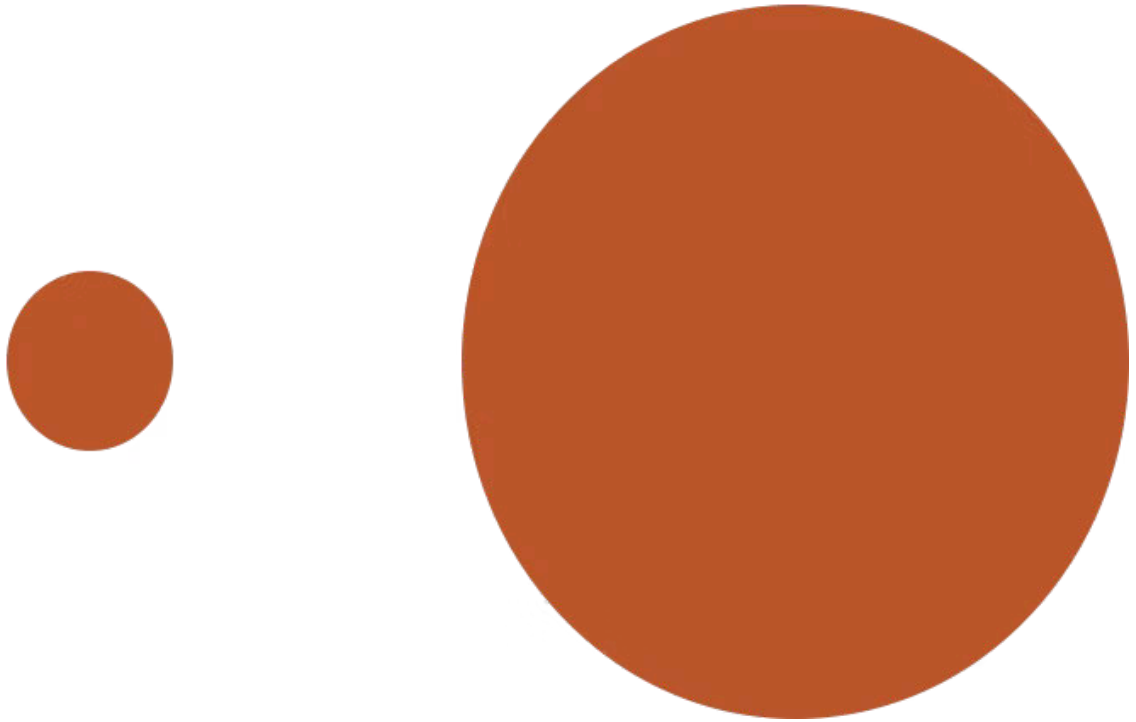


Most Accurate: Position, length, angle, slope

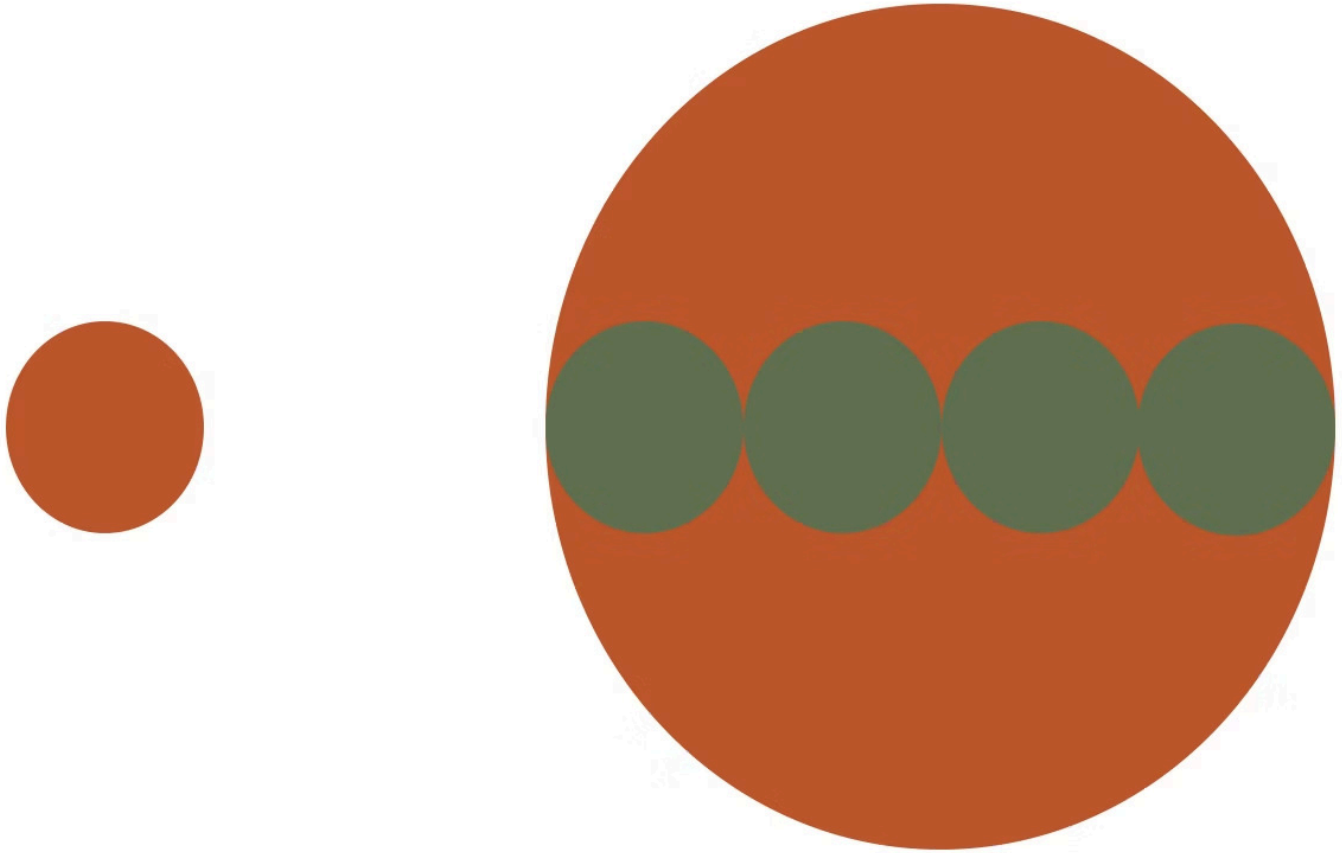
Least Accurate: Color saturation, area, density, volume

Cleveland, W.S. & McGill, R. Science 229, 828-833 (1985)

Compare area of circles



Compare area of circles



Which is brighter?

Which is brighter?

(128, 128, 128) vs (144, 144, 144)

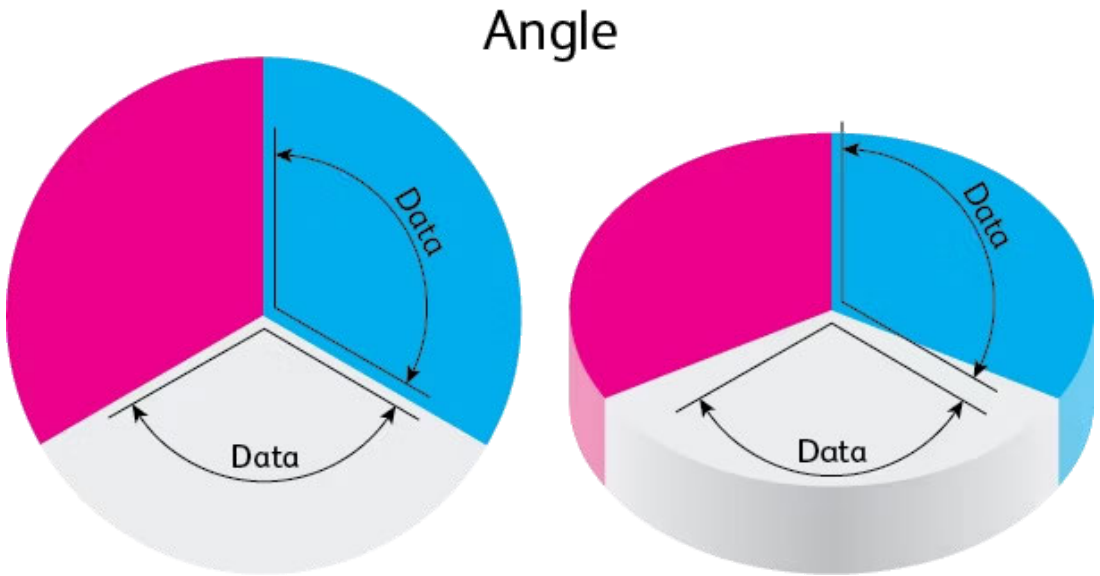
Just Noticeable Difference

Perception Principles

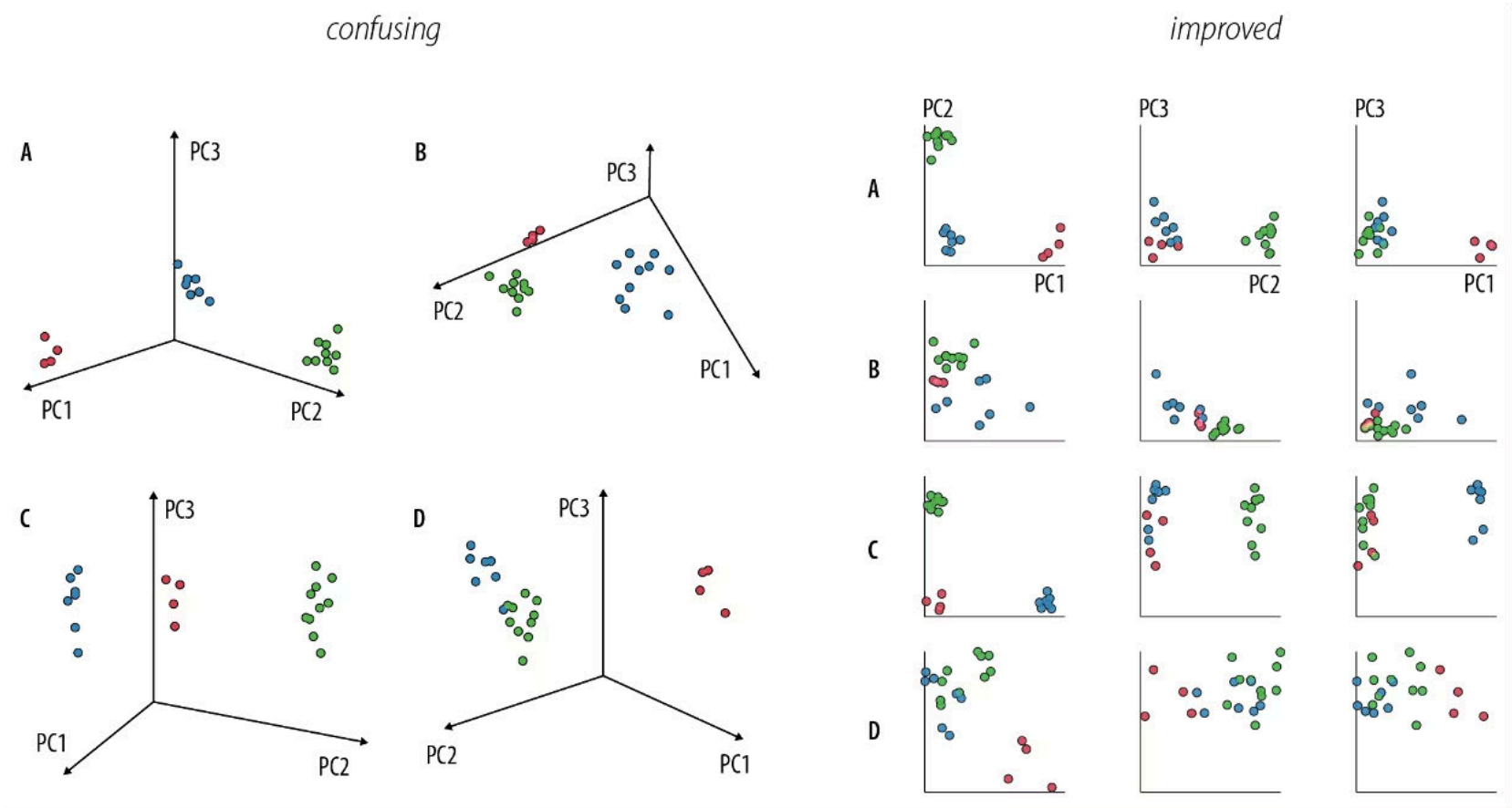
Key Principles

- **Ratios matter more than magnitude** – We perceive relative differences, not absolute values
- **Discrete perception** – Continuous variations appear as discrete steps to human vision
- **Design implication** – Use sufficient contrast for meaningful differentiation

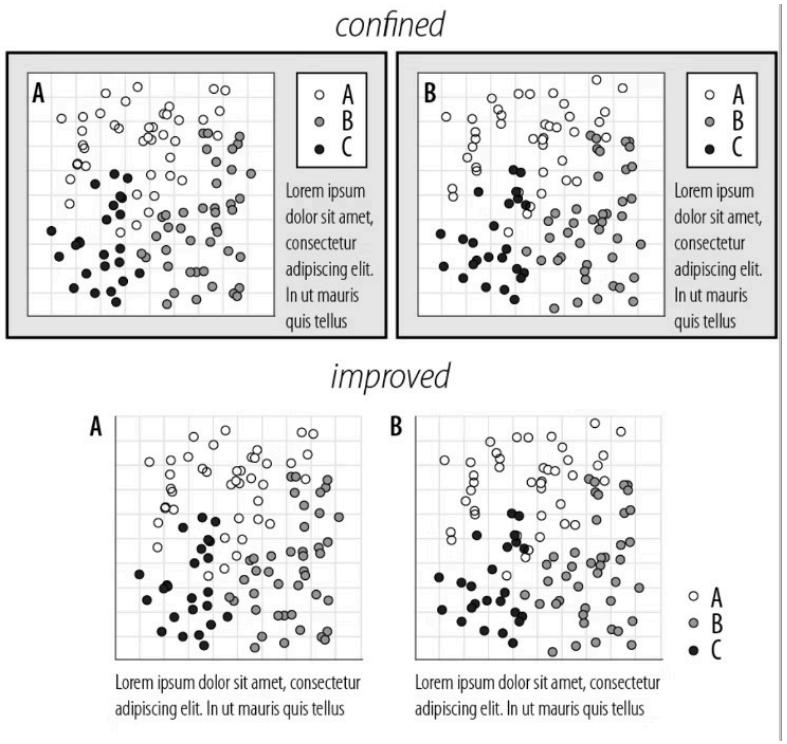
Careful with 3D and pie charts



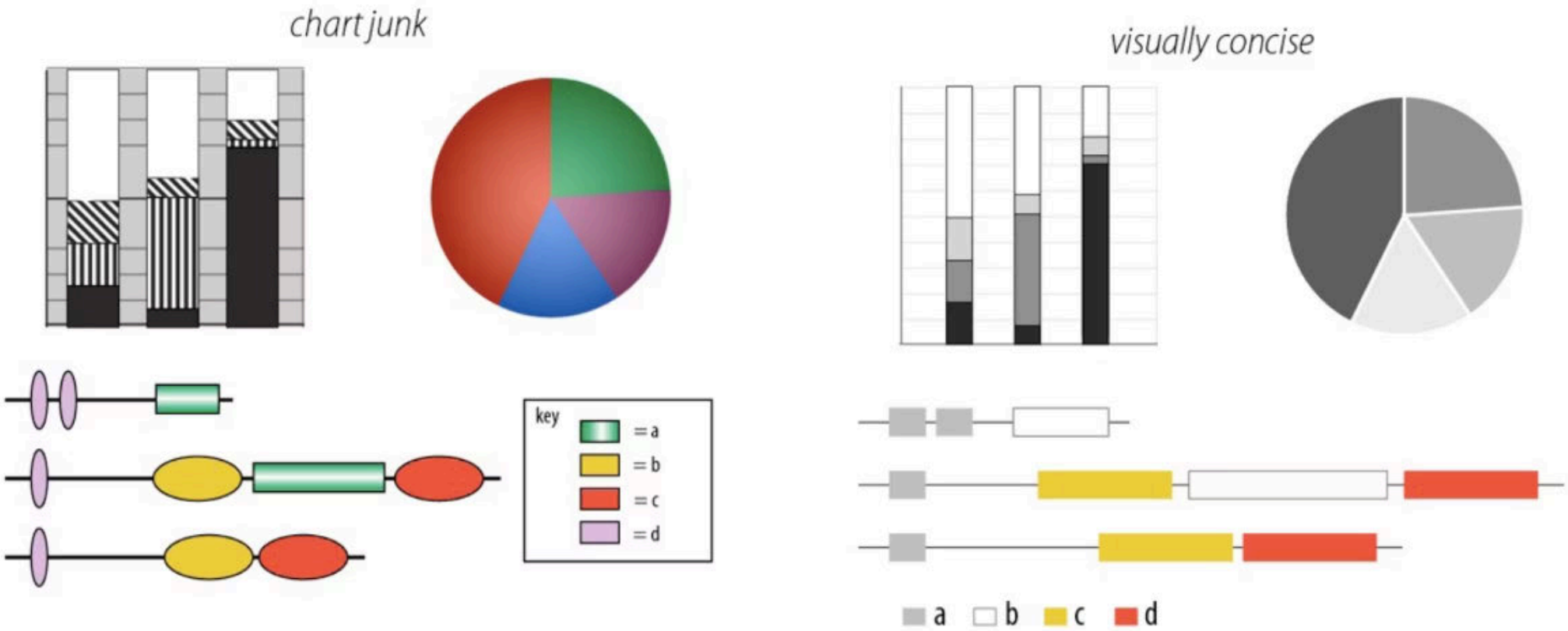
Avoid 3D



Increase data to ink ratio



Avoid chart junk



Brewer Color Palettes

Scientifically Designed for Data Visualization

Qualitative
For categorical distinctions

Sequential
For ordered data from low to high.

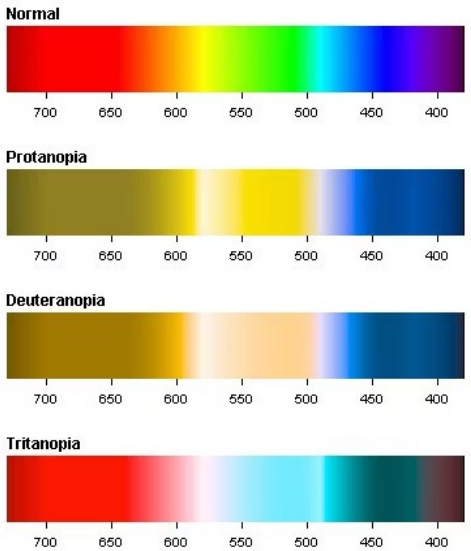
Diverging
For data with meaningful midpoint.

Explore palettes at **colorbrewer2.org** – based on HSV color model for optimal perception

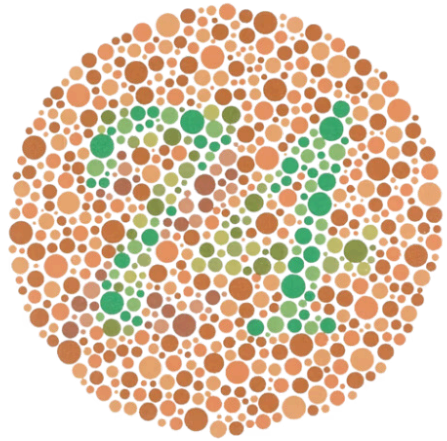
The **HSV (Hue, Saturation, Value) model** is a cylindrical, user-oriented color space that represents colors similarly to human perception, separating color information (Hue) from intensity (Saturation) and brightness (Value).

Color Blindness Considerations

Design for Accessibility



Normal Vision



Red-Green Color Blind

Critical: 10-20% of population has red-green color blindness. Always test your color choices or use colorblind-safe palettes

- Use ColorBrewer palettes with colorblind-safe options
- Supplement color with patterns, shapes, or labels
- Test visualizations with colorblind simulators

Integrate Text & Graphics

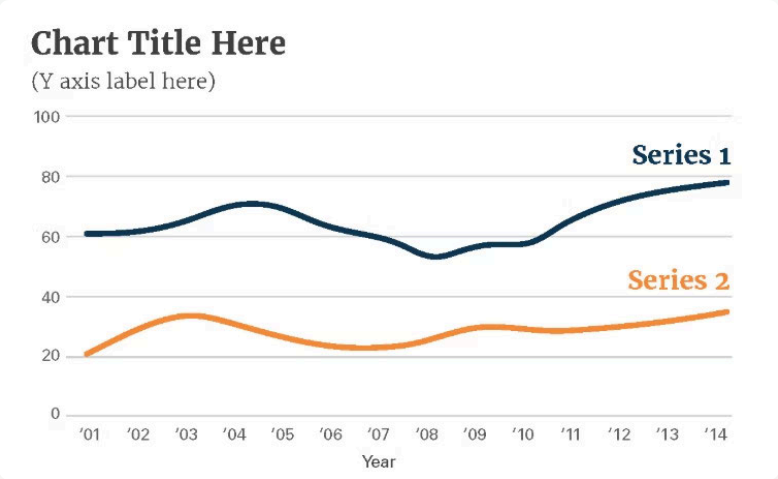
Complementary, Not Redundant

❌ Avoid: Slideshow Effect

- Text simply narrates the visual
- Information presented twice

✓ Better: Complementary Design

- Visuals illustrate key points
- Text provides **context & interpretation**
- Each element adds unique value
- Graphics work **standalone** if needed



Visual Processing Pathway

How Our Brain Sees Data

Light Enters Eye

Processed by retina, detecting basic features: edges, motion, brightness

Signal to Visual Cortex

Occipital lobe receives and processes visual information

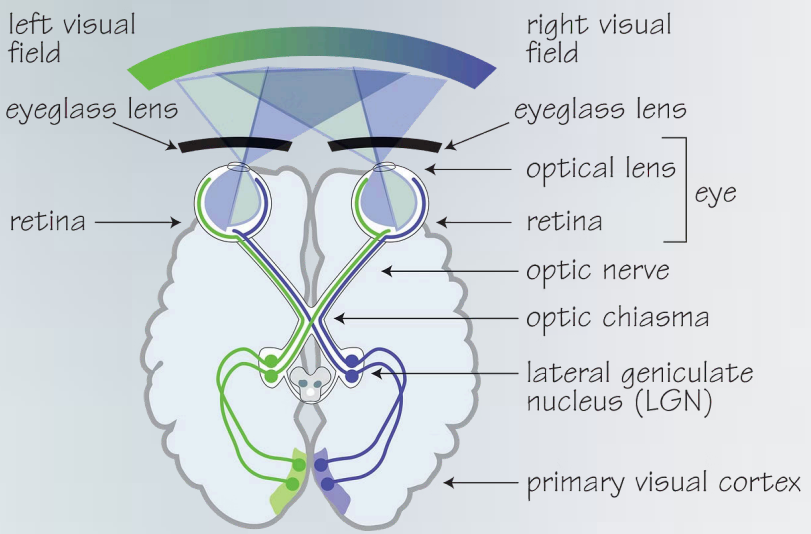
Higher-Order Processing

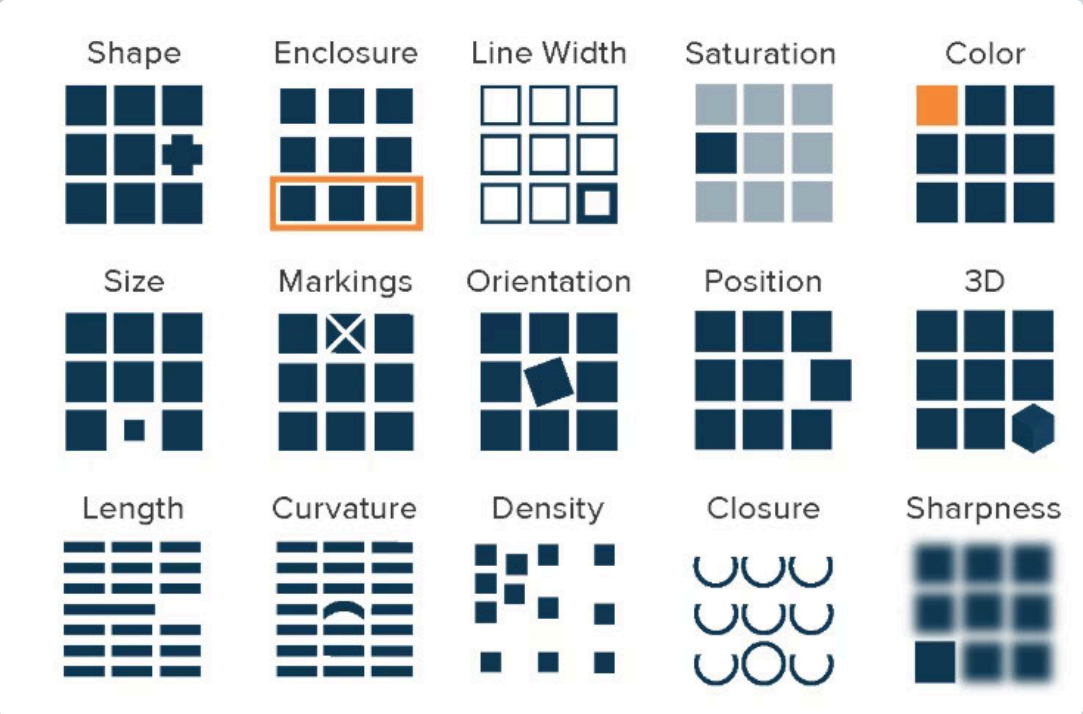
Parietal & temporal lobes assign meaning to visual patterns

Pattern Recognition

Brain rapidly detects patterns and anomalies within milliseconds

This entire process happens in **milliseconds**, enabling rapid data comprehension through visual channels





Preattentive Processing

What We See Before We Think

Before we actively focus, our brain **automatically detects** certain visual properties in less than 250ms

Color
Instantly distinguishes hues

Form
Recognizes shapes automatically

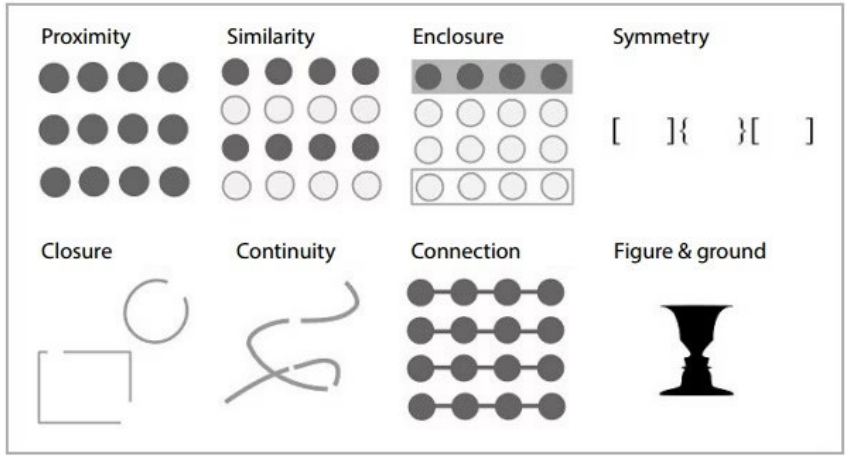
Position
Detects spatial relationships

Motion
Notices movement instantly

Design tip: To draw attention, create clear breaks between groups using preattentive attributes

Gestalt Psychology Principles

How We Group Visual Elements (1912)



*Source: **FusionCharts Blog***

Proximity
Objects near each other appear grouped

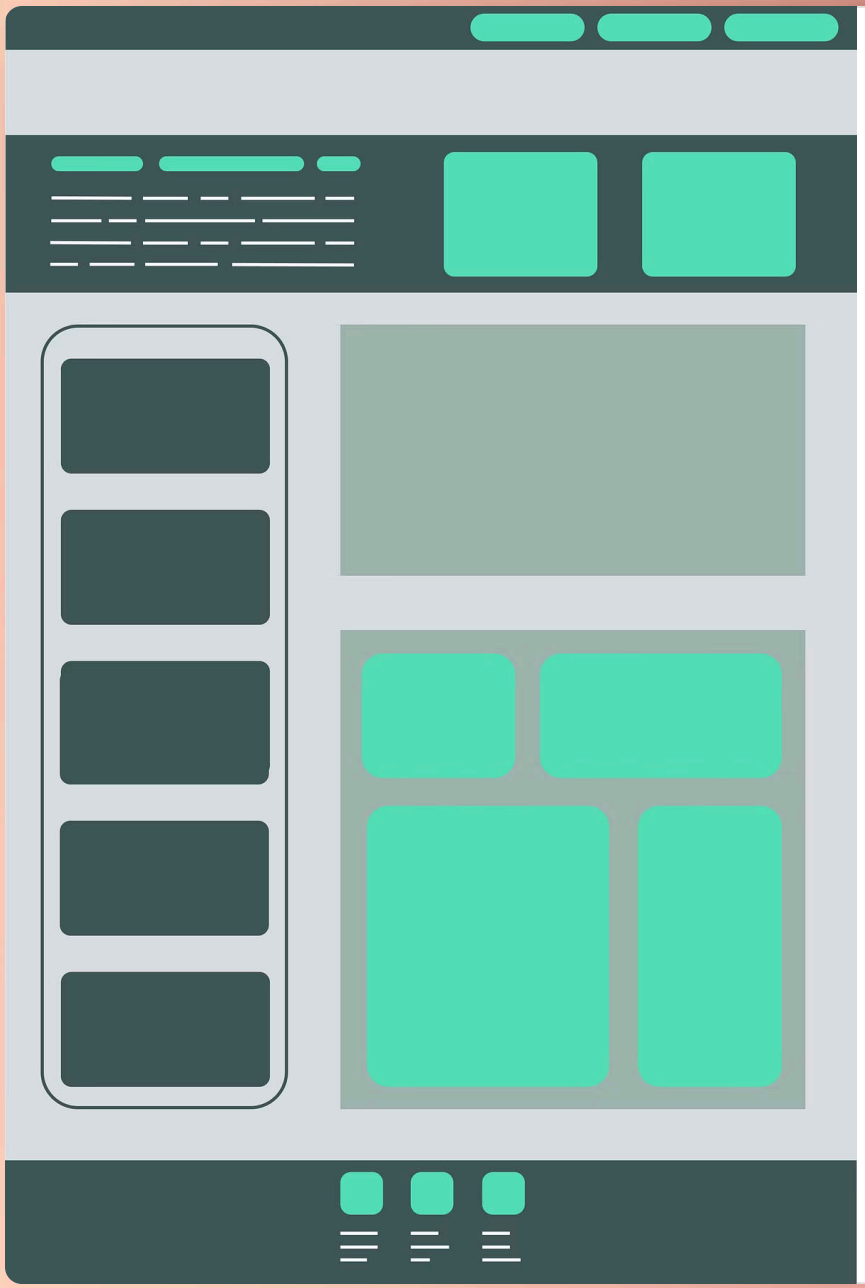
Similarity
Similar elements are perceived as related

Continuity
Eye follows lines and curves naturally

Closure
Mind completes incomplete shapes

Figure-Ground
Distinguishes objects from background

Common Region
Elements in bounded areas feel related



Applying Gestalt to Data Viz

Use Consistent Colors & Shapes

Apply same visual attributes to related categories throughout your visualization

Arrange Elements Logically

Group related data spatially to reduce cognitive effort and guide interpretation

Support Natural Reading Flow

Ensure labels, legends, and annotations follow intuitive left-to-right, top-to-bottom patterns

> Strategic application of Gestalt principles helps viewers unconsciously organize and understand complex data

Working Memory & Cognitive Load

The 4-7 Rule

4–7

Items

Maximum in working memory at once

Reduce Cognitive Load

- **Minimize clutter** – Avoid excessive gridlines, 3D effects, decorations
- **Use chunking** – Group related data visually to reduce perceived complexity
- **Leverage hierarchy** – Bold key insights, subtle secondary information

---

1

Design Implication

Simpler is better – A well-designed bar chart beats an overcomplicated 3D pie chart

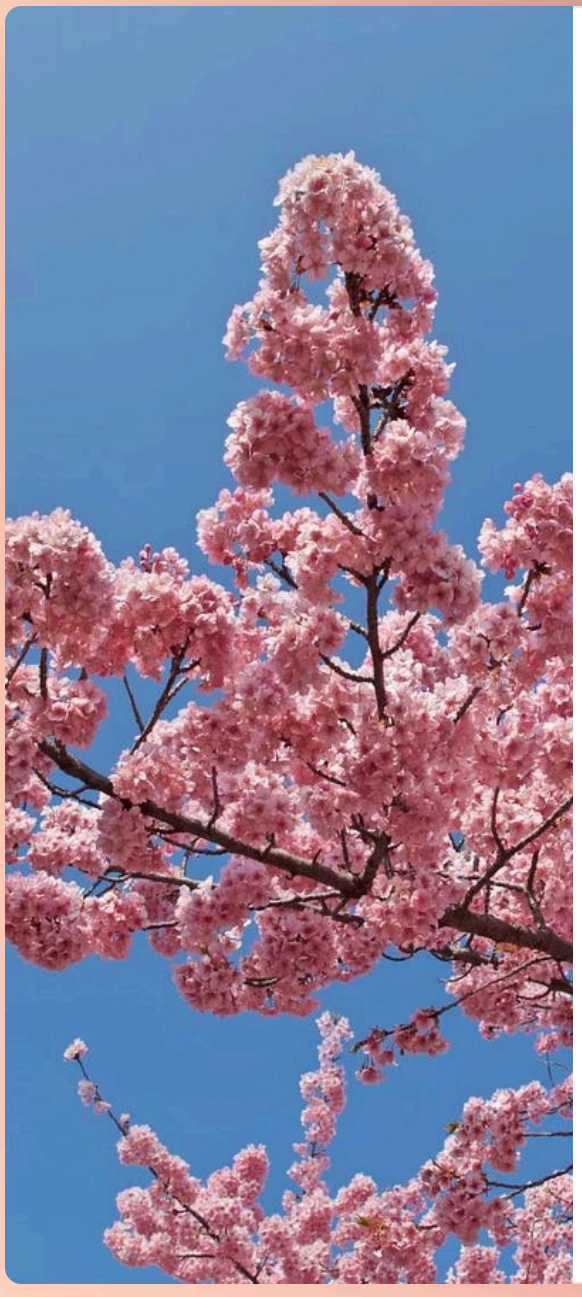
2

Guide the Eye

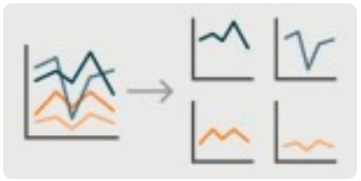
Use annotations, color emphasis, and layout to highlight key messages

Who is your audience?

- How detailed do they want to see the data?
- Do they have a technical background?

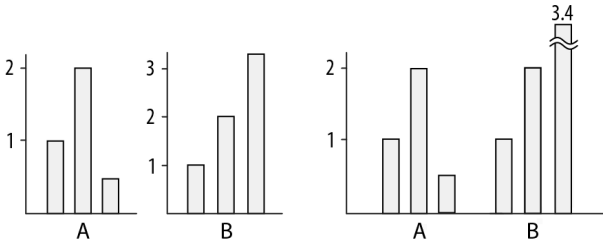


Additional Best Practices



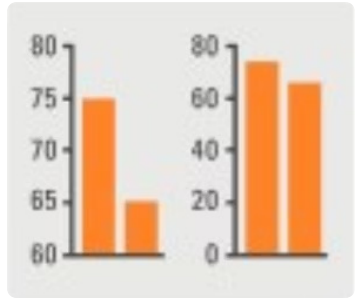
Break Up Complexity

Split complicated graphs into digestible pieces



Readable Labels / not misleading

Ensure all text is legible at intended viewing size



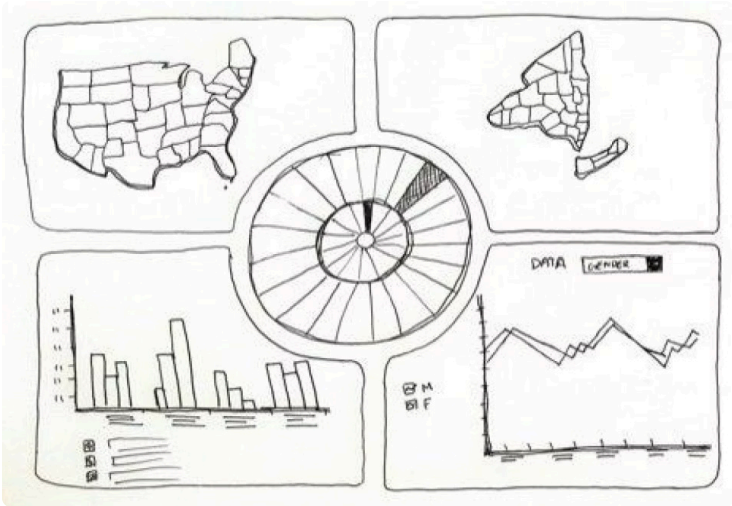
Add Annotations

Guide attention to important patterns or insights

Ethics matter: Don't cherry-pick data to mislead. Show the full picture honestly

Sketch Before You Code

Plan Your Visualization Strategy



> "A few minutes with pencil and paper can save hours of coding time"

Why Sketch?

- Rapid iteration without coding overhead
- Focus on concept before implementation
- Easy to share and get feedback
- Clarifies design decisions early

Key Questions

- How will it be viewed? (desktop, mobile, print)
- What's the primary message?
- Which chart type best fits the data?
- What interactions are needed?

The Future of Data Visualization

AI & Data Storytelling

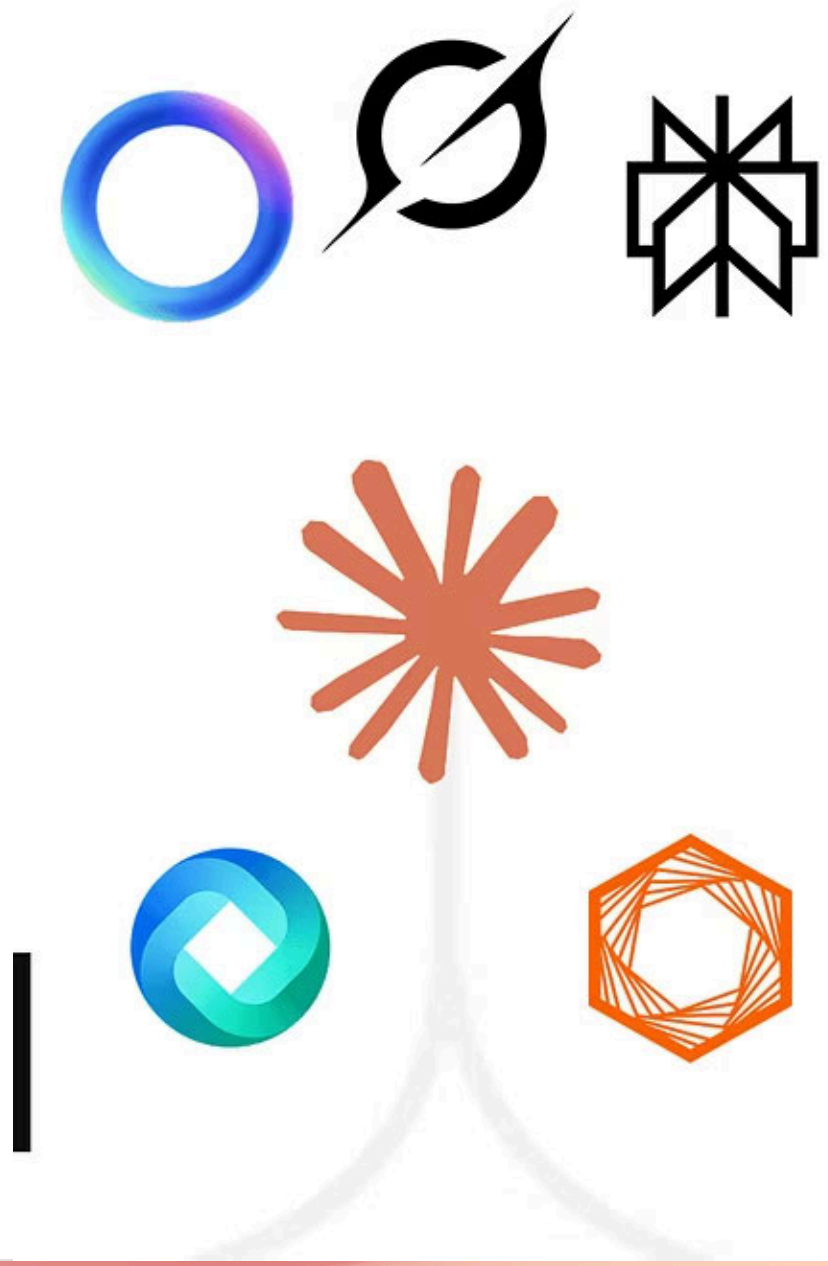
Generative AI and NLP now help **automate insights** from visualizations

Which tools? How does it work?

Timeless Principles Remain

Despite technological advances, **clarity, simplicity, and storytelling** remain the foundations of great visualization

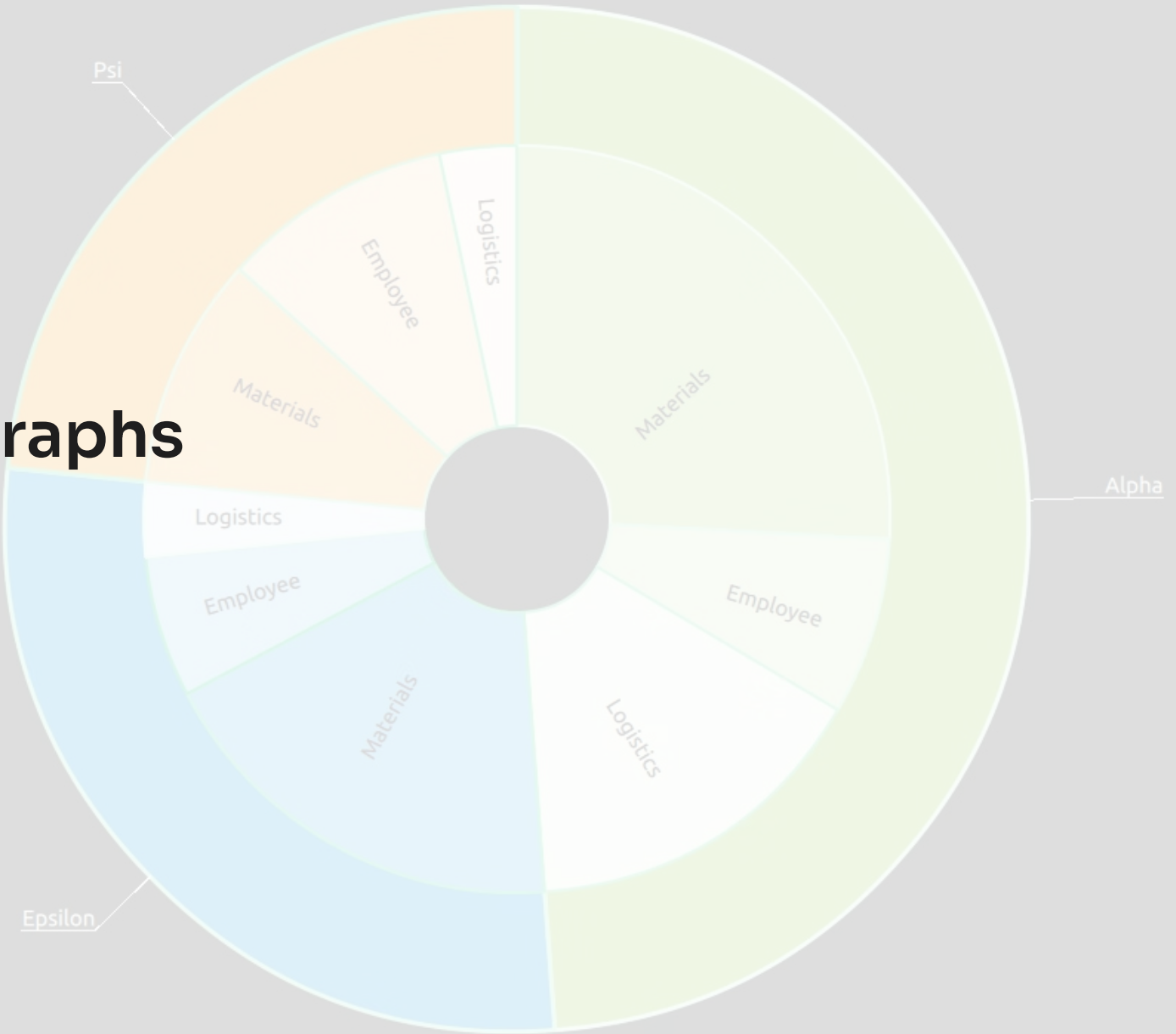
AI can generate charts, but human insight still drives meaningful interpretation



Qt Quick Graphs Testbed

Common graphs

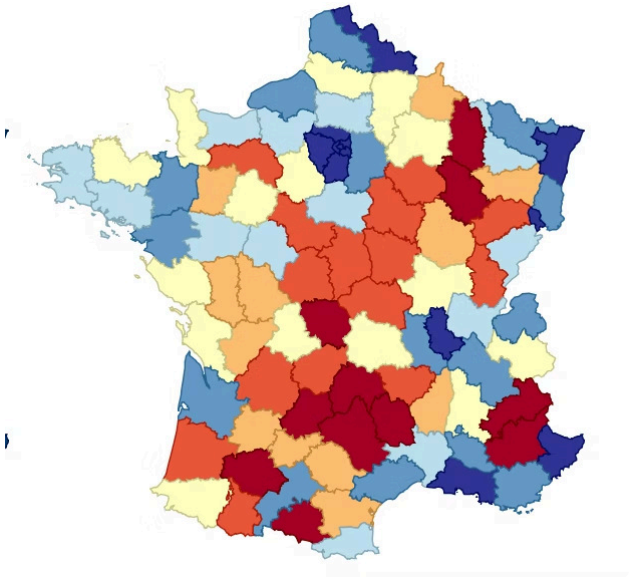
2D Graphs



GEOGRAPHIC DATA

Choropleth Maps

Color-Encoded Geographical Regions



Map source: ***GADM***

Definition

Divided geographical areas colored, shaded or patterned in relation to a data variable

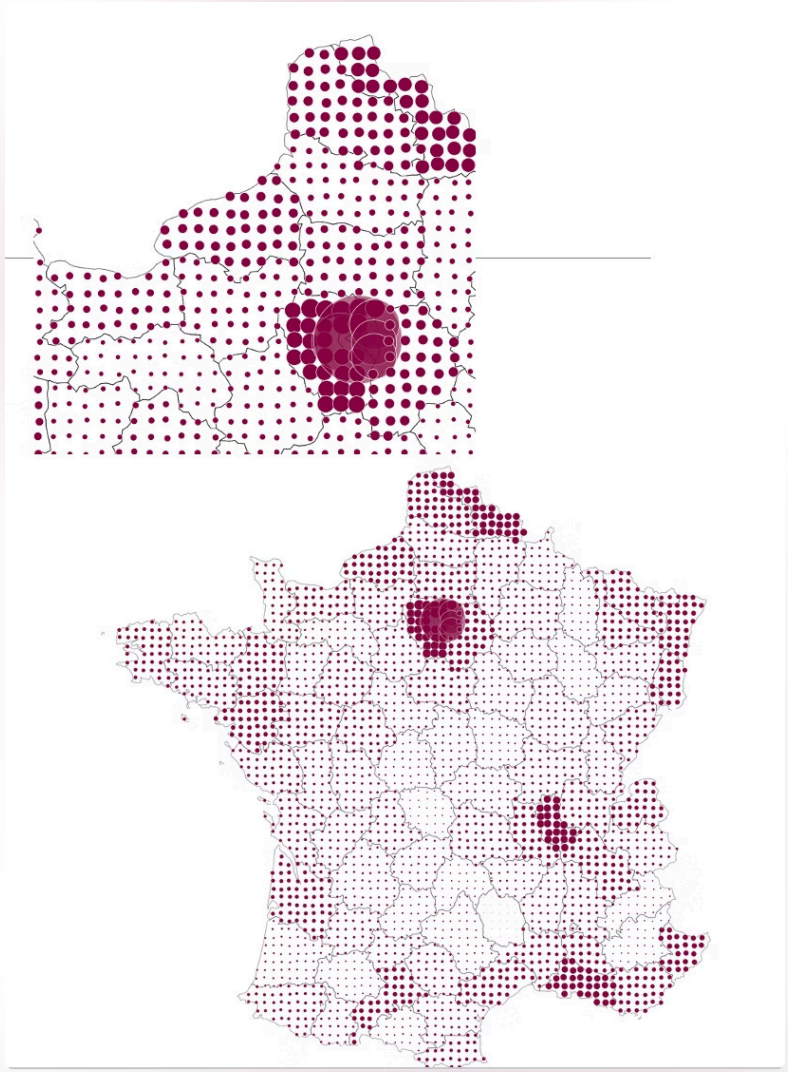
Example

Population data for French departments

Common Pitfalls

- **Exact values hidden** – Consider adding hover tooltips
- **Raw vs. normalized data** – Always use normalized values (e.g., per capita, per sq km)

Critical Error: Encoding raw population instead of population density distorts perception



Dot Grid Maps

Showing Quantity & Density

Key Advantage

Shows both **quantity** and **density** of distribution simultaneously

Example Use

Population distribution across French departments

Compare Density

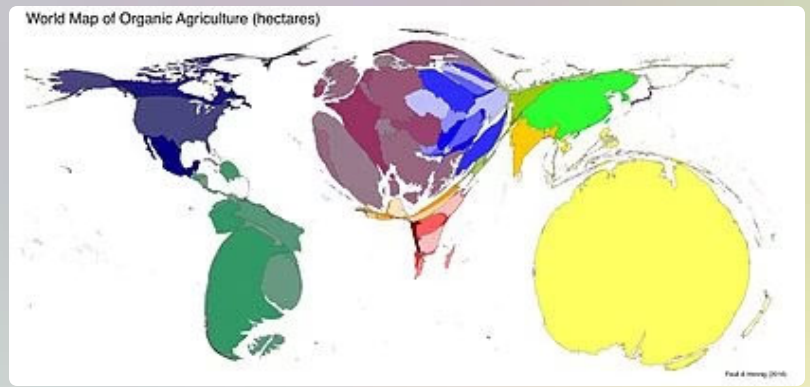
Individual circles show concentration patterns

Assess Totals

Overall dot count reveals total population per region

Visual Intuition

Dense clusters immediately signal high-population areas



Cartograms

Distorting Geography to Show Data

Distorts geometry or space of a map to convey alternative variables (e.g., population, travel time, economic output)

Area Cartograms

Region size proportional to data value

Example: Organic farming area per country [shown on left]

Distance Cartograms

Spatial relationships based on travel time or cost

Example: Transit time from central city

Trade-off: Enhanced data clarity vs. reduced geographic recognizability

TEMPORAL DATA

Time Series Visualizations

Tracking Change Over Time

Connected scatterplot showing multiple variables changing over time

Identify Trends

Spot upward, downward, or cyclical patterns

Compare Series

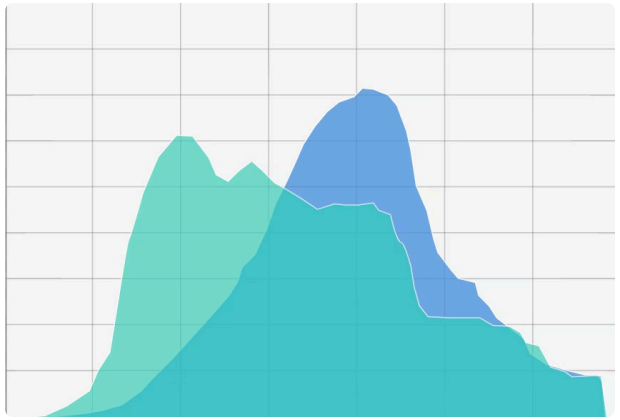
Multiple lines reveal relative performance

Mark Events

Annotate significant moments that influenced data

Area Charts

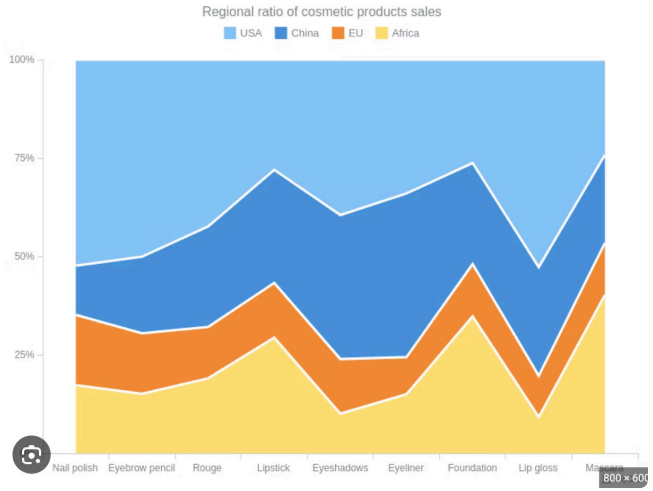
Comparing Quantities Over Time



Overlapping Area Chart

Transparent shading allows all values to remain visible for easy comparison

- **Use overlapping** when individual trends matter most



Stacked Area Chart

Shows cumulative total and individual contributions simultaneously

- **Use stacked** when showing composition of a whole over time

Arc diagrams

Multimodal Charts

COMPOSITION

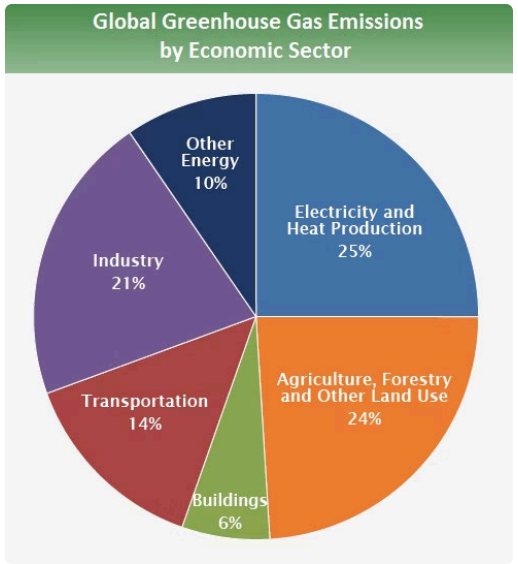
Pie Charts

Visualizing Parts of a Whole

Divided into sectors to illustrate numerical proportion — arc length and angle of each sector is proportional to the quantity it represents

Best Practices

- Limit to 5-7 categories maximum
- Start largest slice at 12 o'clock
- Order slices by size (except for special categories like "Other")
- Consider alternatives for precise comparisons

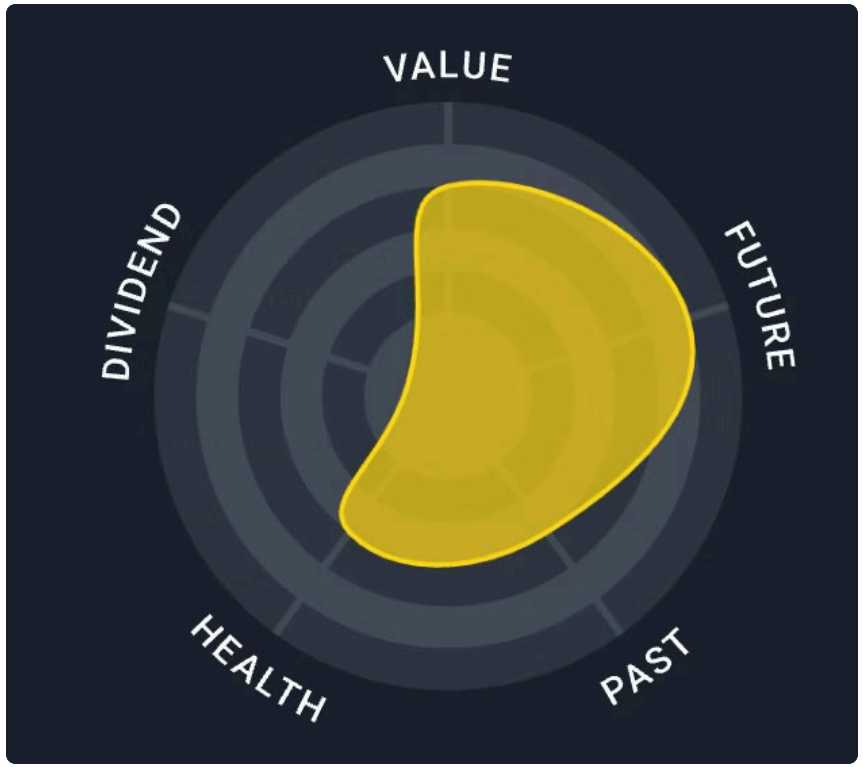


Source: EPA

Pie charts excel at showing **dominant categories** but struggle with precise value comparisons

Radar/Spider Charts

Multi-Variable Comparison



First Quantum Mining on SimplyWallstreet.com

Structure

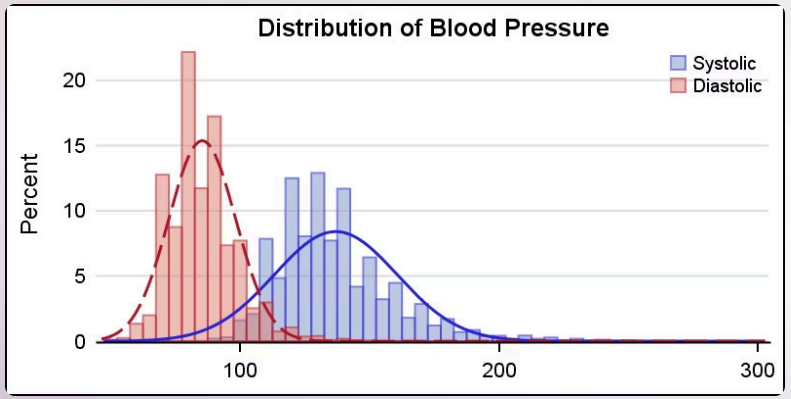
Values plotted along separate axes radiating from center, showing relative performance across dimensions

Common Uses

- Sports statistics (player performance profiles)
- Budget allocations across departments
- Product feature comparisons
- Skill assessments

When to Use

Most effective when displaying **5+ variables** that share a common scale



DISTRIBUTION

Histograms

Understanding Data Distribution

Purpose

Display the **frequency distribution** of continuous data

Structure

Rectangles with heights proportional to count, widths equal to bin size

Key Insight

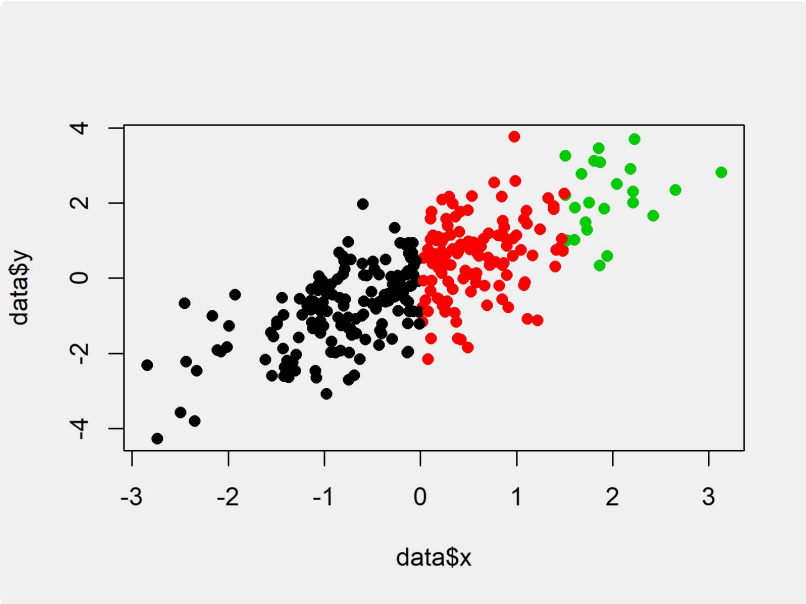
Reveals shape of distribution: normal, skewed, bimodal, uniform

Bin size matters: Too few bins hide patterns; too many bins create noise. Experiment to find optimal granularity

RELATIONSHIPS

Scatter Plots

Discovering Correlations



Correlation

Initial insights into correlation / relation between groups

Clusters

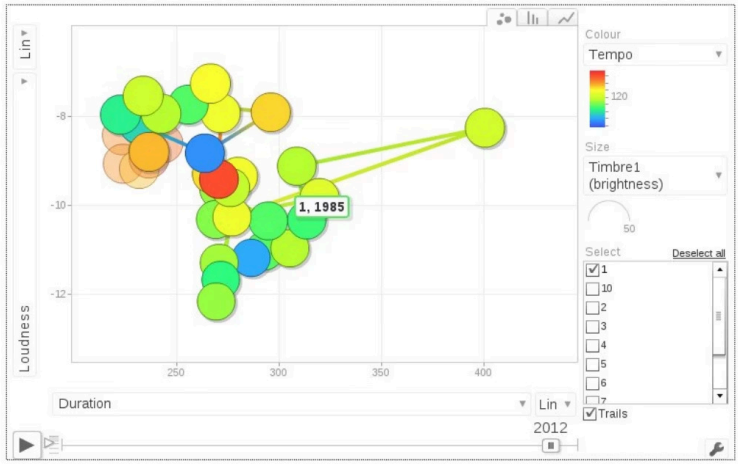
Distinct groupings suggest categories

Outliers

Unusual points warrant investigation

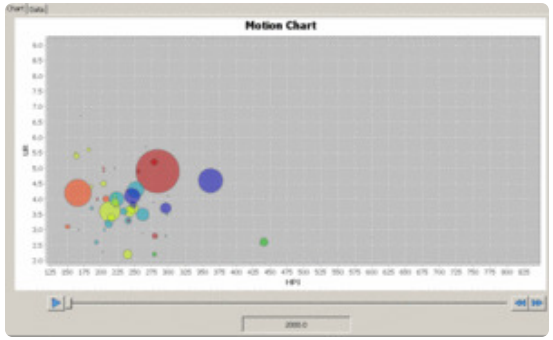
Bubble Charts

Adding a Third Dimension



Standard Bubble Chart

Size encodes third variable



Motion Chart

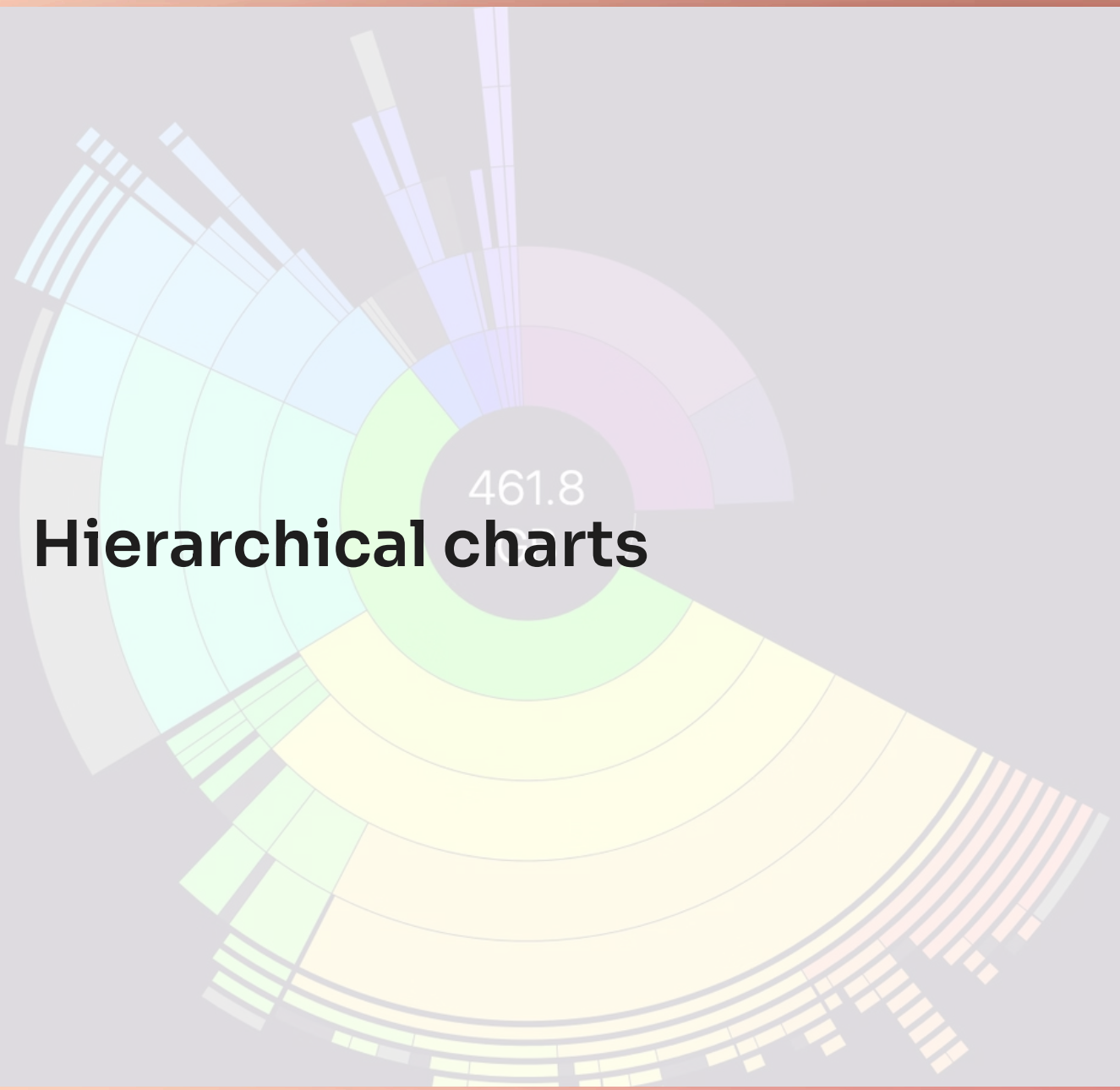
Animation adds time dimension ***dorienherremans.com/dance***

Hans Rosling popularized animated bubble charts to show development trends over time

4+

Dimensions

X, Y, size, color, and optionally animation

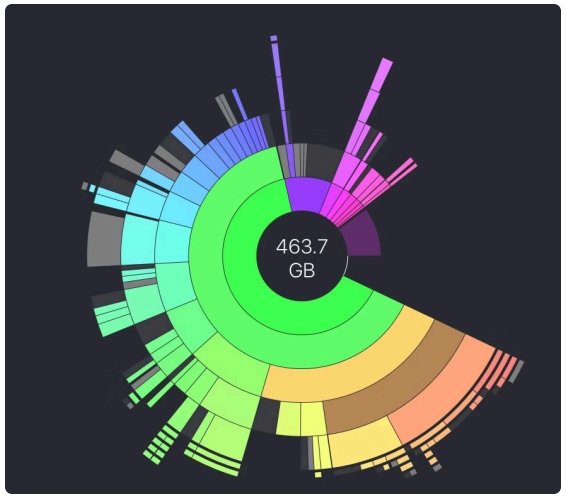


Hierarchical charts

HIERARCHICAL DATA

Ring Diagrams (Sunburst)

Multi-Level Hierarchies



Source: DaisyDisk

Definition

Multi-level pie chart visualizing hierarchical data with concentric circles

Structure

- **Inner ring** – Top-level categories
- **Outer rings** – Nested subcategories
- **Arc size** – Proportional to value

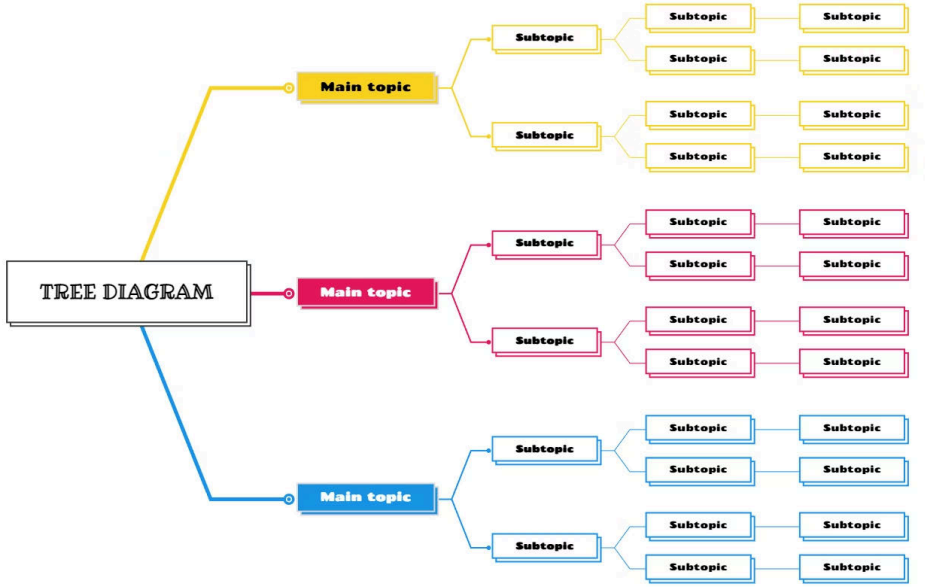
Common Uses

- Disk space analyzers (like DaisyDisk)
- Budget breakdowns by department/project
- Organizational hierarchies

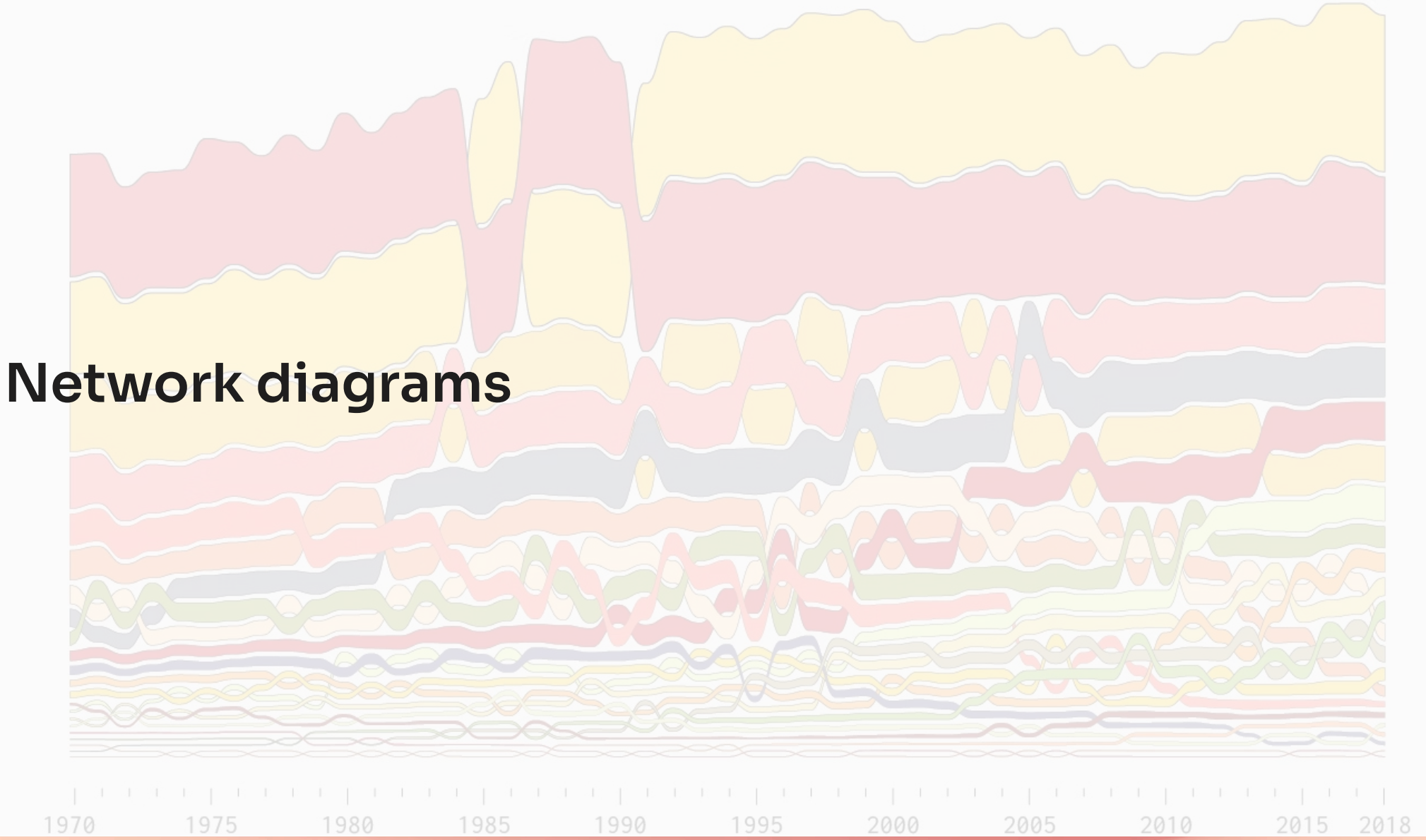
Tree diagram

Represents the hierarchical nature of a structure in graph form. It can be visually represented from top to bottom or left to right.

Also called dendrogram when representing clusters



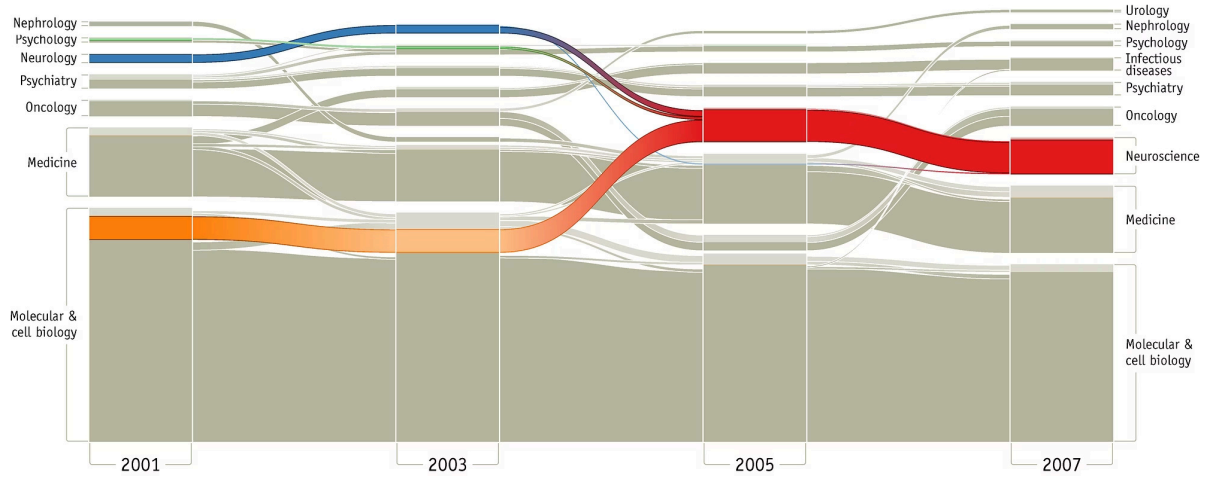
Network diagrams



FLOW & RELATIONSHIPS

Alluvial Diagrams

Visualizing Flow Between States



Flow diagram showing how neuroscience coalesced from related disciplines

Rosvall, M., & Bergstrom, C. T. (2010). Mapping change in large networks. PLoS ONE, 5(1), e8694. CC BY 2.5

Track Changes

See how categories evolve and merge over time

Show Magnitude

Flow width indicates volume of transition

Reveal Patterns

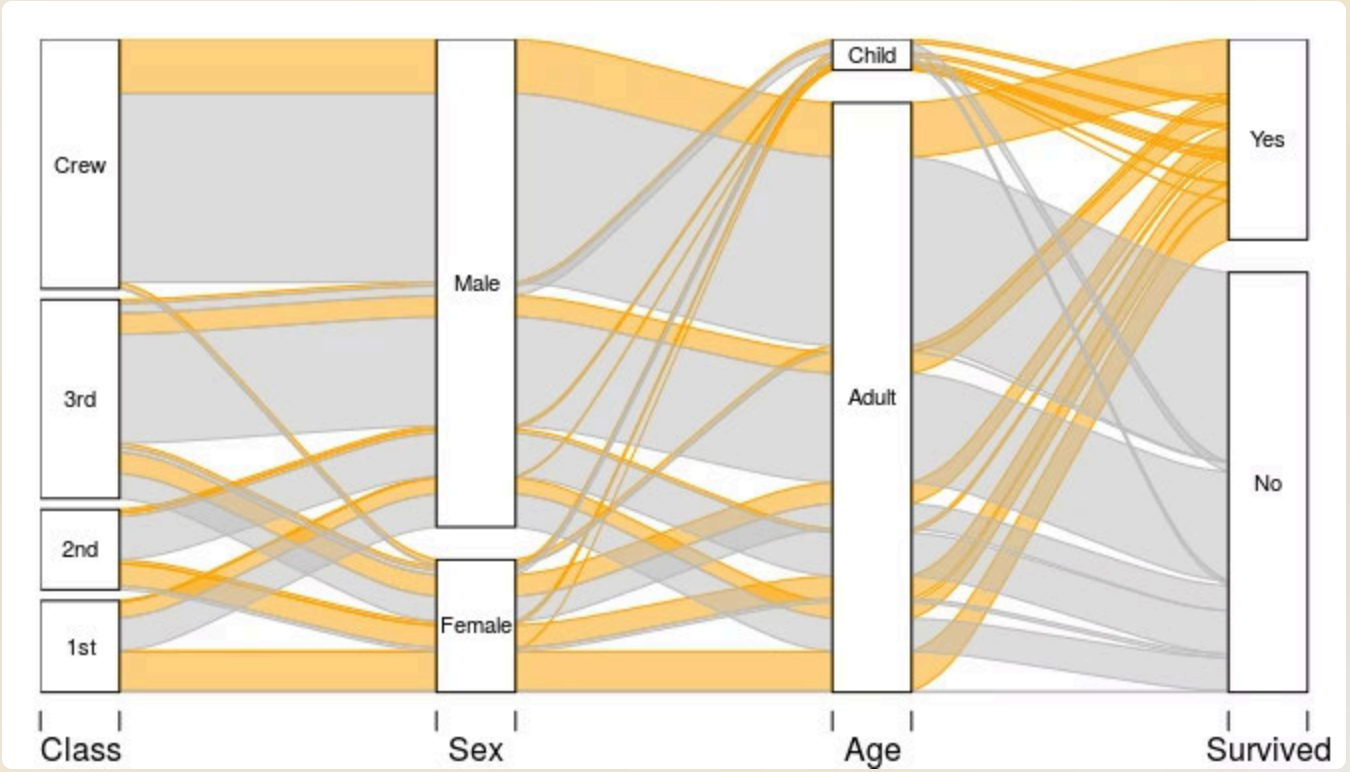
Identify dominant pathways and trends

Alluvial Example: Titanic Survivors

Flow showing relationship between passenger class, gender, age, and survival

Source: ***R alluvial package***

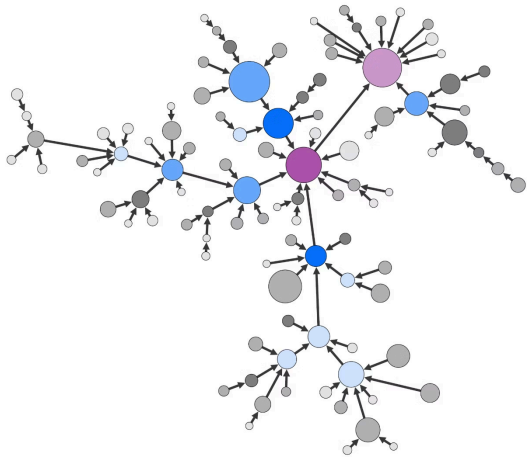
- Reveals patterns: First-class women and children had highest survival rates



NETWORK DATA

Node-Link Diagrams

Visualizing Connections



Example: Company sales network

*Source: **Linkurious***

Nodes
Represent entities (people, companies, concepts)

Links
Show relationships or connections between nodes

Common Uses
Social networks, org charts, citation networks, knowledge graphs

3D: hypergraph

Matrix Visualizations

Showing Multi-Group Relationships

Matrix charts show relationships between **2-4 groups** of information simultaneously

Correlation Matrices

Show strength of relationships between variables

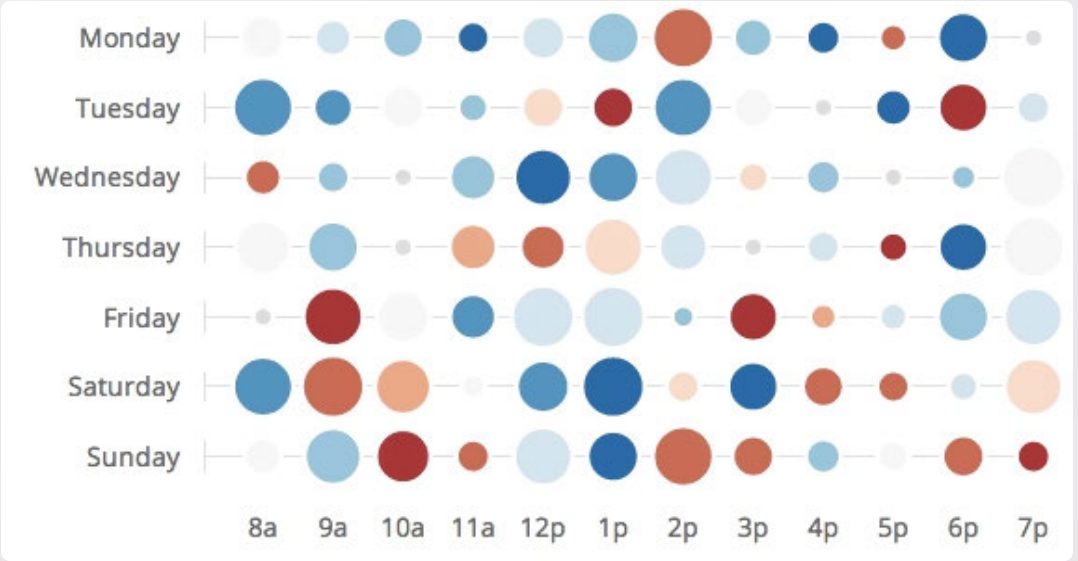
Heatmaps

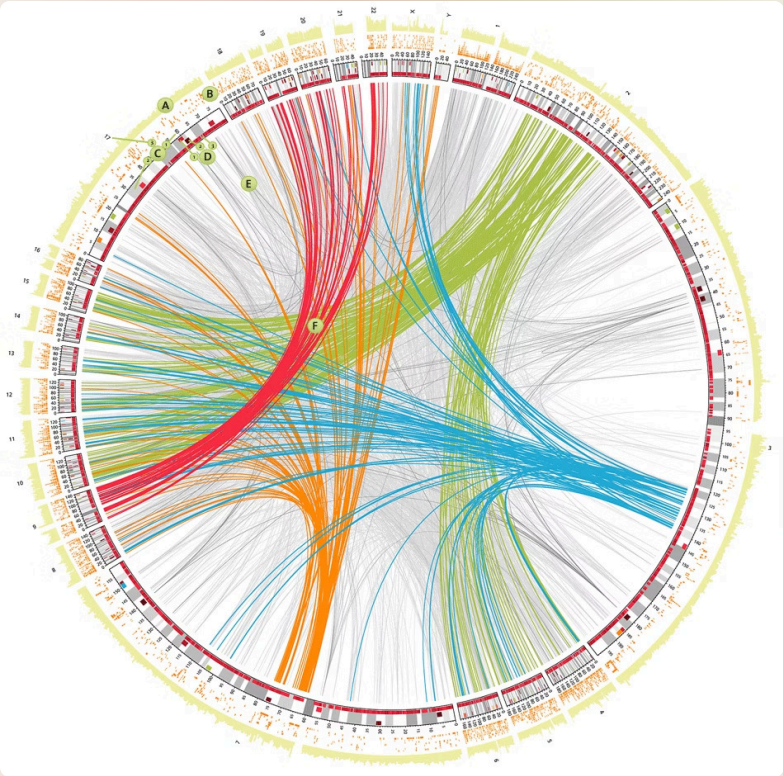
Color intensity indicates magnitude

Adjacency Matrices

Alternative to node-link for dense networks

Matrix format scales better than node-link diagrams for networks with many connections





Circos: Human Genome Visualization

What It Shows

- Location of disease-implicated genes
- **Regions of self-similarity:**
  - Structural variations within populations
  - Chromosomal relationships

Design Elements

- Uses:
  - Links between genomic regions
  - Heat maps for expression data
  - Tiles and histograms for sequence info
- Strategic use of color, transparency, length

> Exemplifies sophisticated multi-layered visualization combining continuity, color theory, and hierarchical information

Tools for visualisation

TOOLS

Tableau

Industry-Leading Business Intelligence

Universal Connectivity

Connect to SQL, MongoDB, Hadoop, CSV, and more

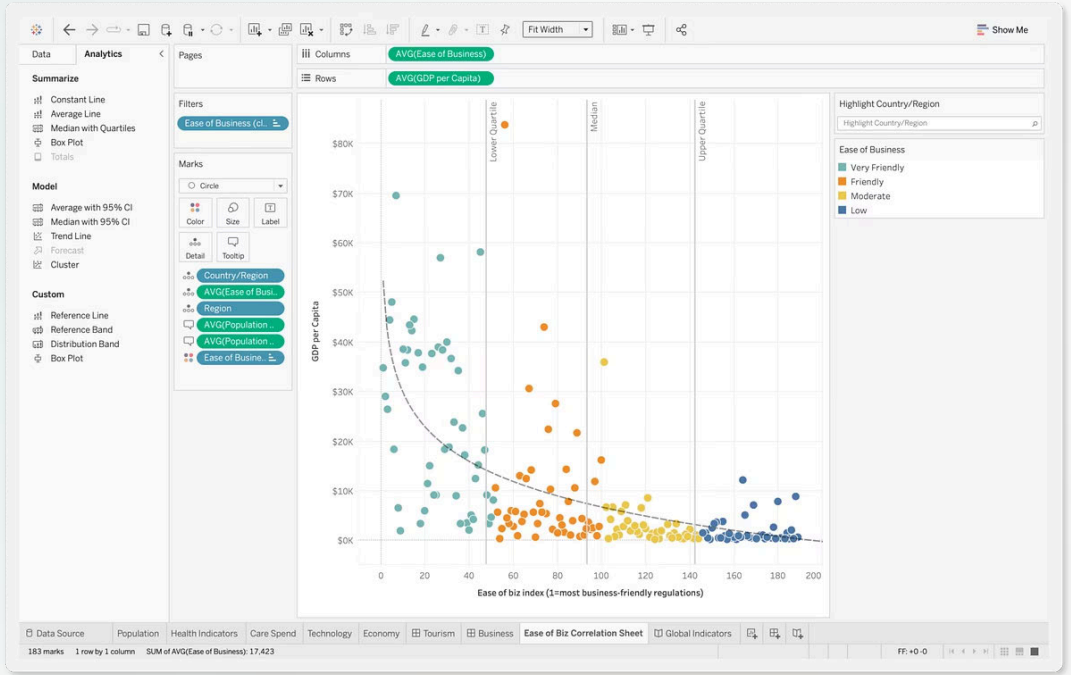
Drag-and-Drop Interface

Rapidly visualize any dataset without coding

Academic Access

Free 1-year student license available

Origin: Commercialized Stanford University research on visual analytics



Apache Superset

Open-Source Alternative

Installation

```
pip install apache-superset
```

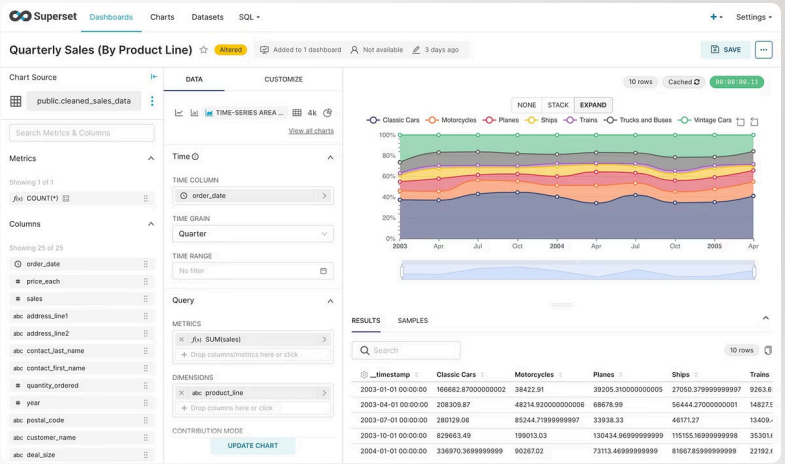
Free, open-source, enterprise-ready

Features

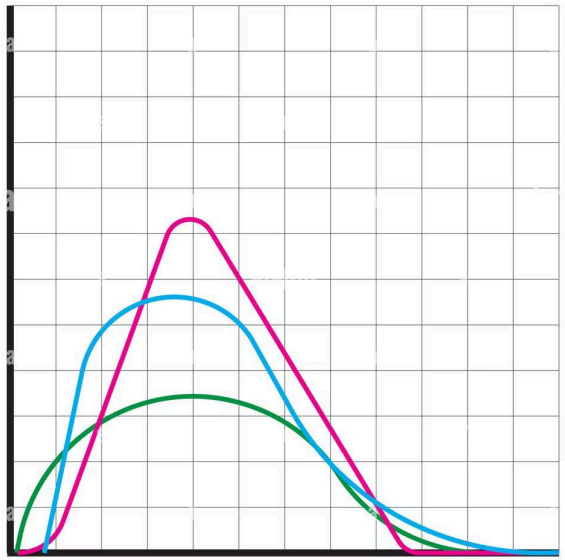
- Interactive dashboards
- SQL Lab for exploration
- Rich visualization library
- Extensible architecture

Also consider: Google Looker Studio (formerly Data Studio) as another free alternative

Learn more: superset.apache.org



Python libraries – matplotlib

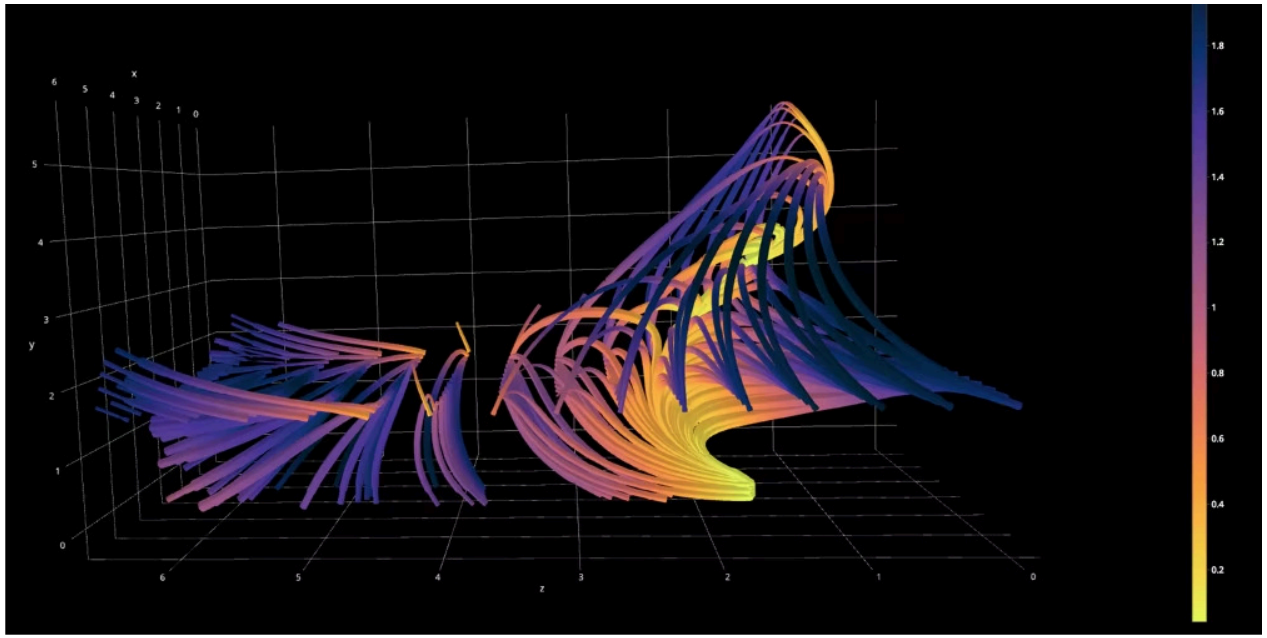
1	2
Matplotlib Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms	**Lab exercises** 

Python libraries – plotly

A Python framework for building analytics web apps.

Also offers maps.

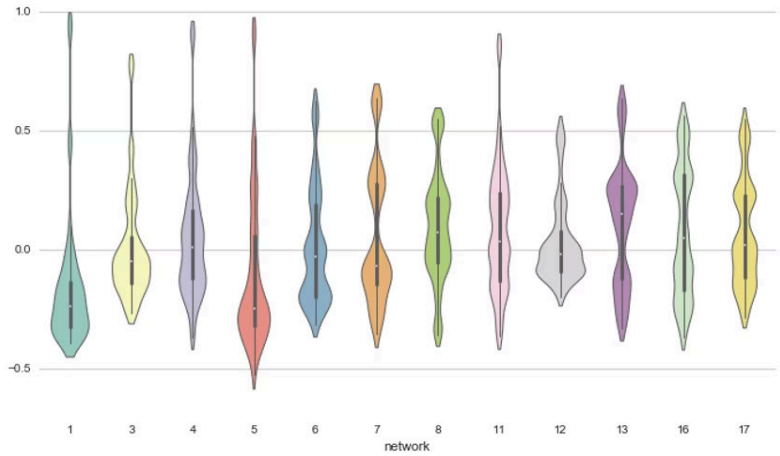
Interactive.

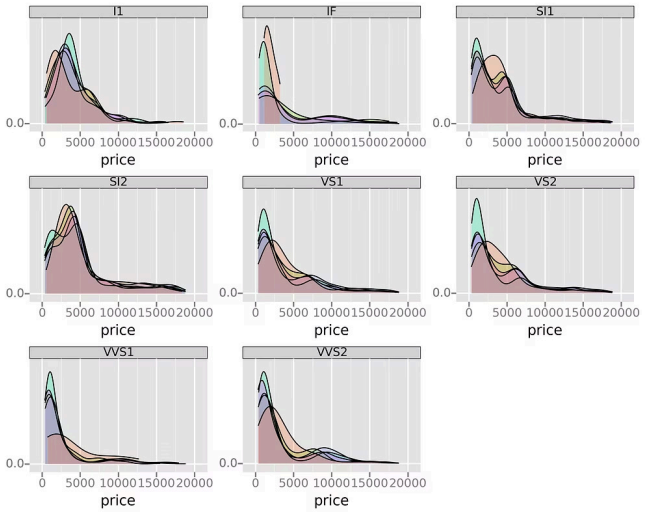


Python lib – seaborn

> "Seaborn is a Python visualization library for **statistical plotting**. It comes equipped with preset styles and color palettes so you can create complex, aesthetically pleasing charts with a few lines of code."

Seaborn is built on top of Python's core visualization library matplotlib.





PYTHON LIBRARIES

ggplot for Python

Grammar of Graphics Approach

1

Based on ggplot2 (R)

Implements Grammar of Graphics concepts in Python

2

Layer Components

Build visualizations incrementally: axes → points → lines → trendlines

Traditional Approach

matplotlib: Configure everything upfront, limited composability

ggplot Approach

Add layers progressively, create complex plots from simple components

geoplotlib

Python Mapping Toolkit

Purpose-Built

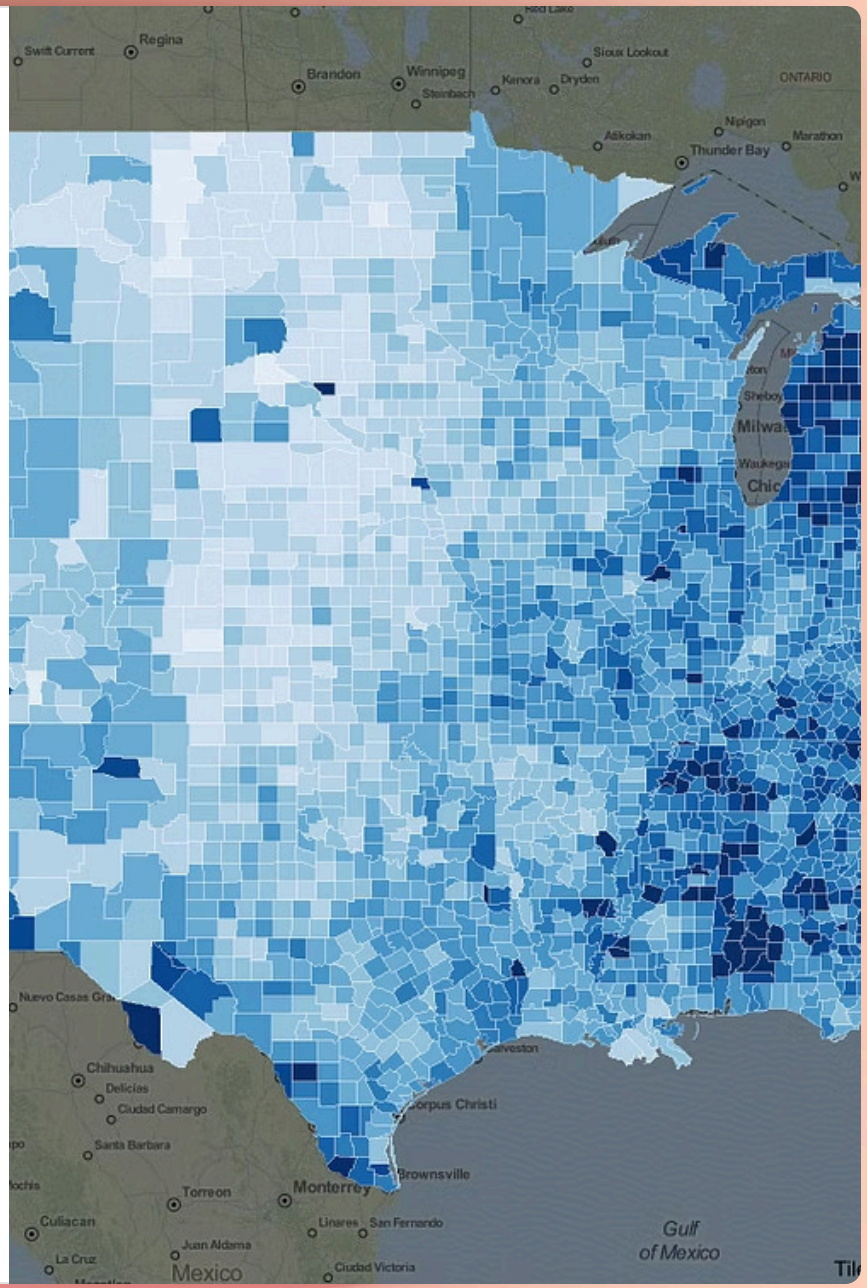
Specialized toolbox for creating maps and plotting geographical data

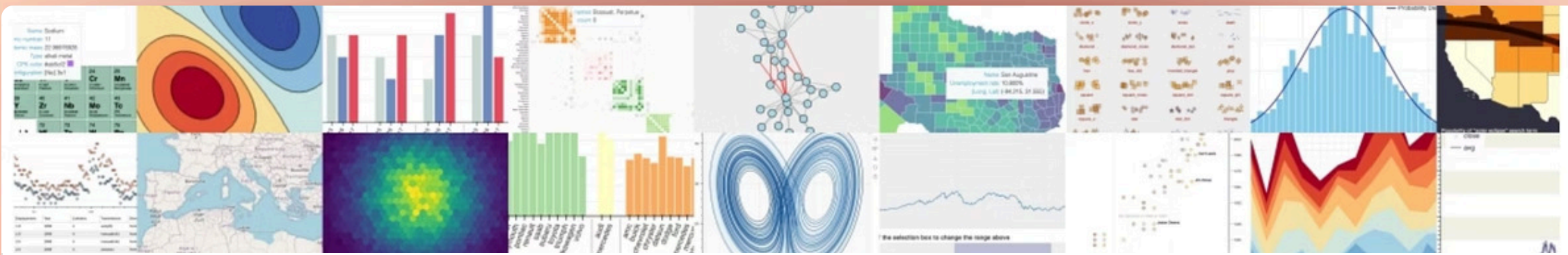
Requirement

Pyglet must be installed (object-oriented OpenGL interface)

Capabilities

Dot maps, heatmaps, choropleths, spatial networks, custom tile providers





Bokeh

Interactive Visualizations for the Web

Python library for creating **interactive** visualizations that render in modern web browsers

Rich Interactivity

Built-in tools for pan, zoom, hover, select

Complex Dashboards

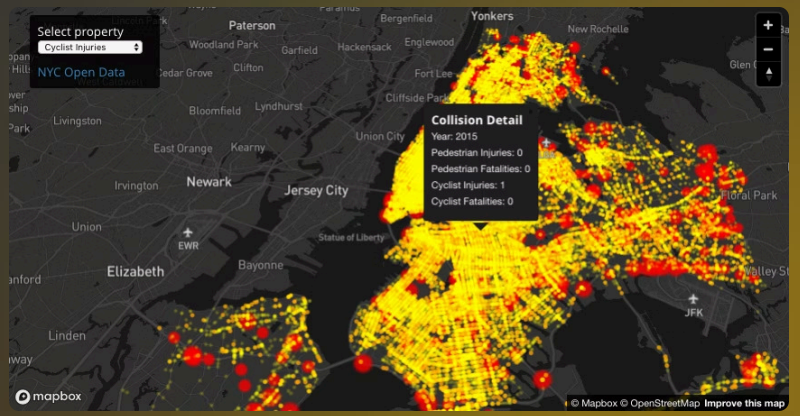
Combine multiple plots with widgets and controls

Streaming Data

Handle real-time data updates seamlessly

Easy Embedding

Export to HTML or embed in web applications



Mapbox

Professional Web Mapping Platform

Industry Adoption

- Foursquare
- Lonely Planet
- Facebook
- The Financial Times
- The Weather Channel
- Snapchat

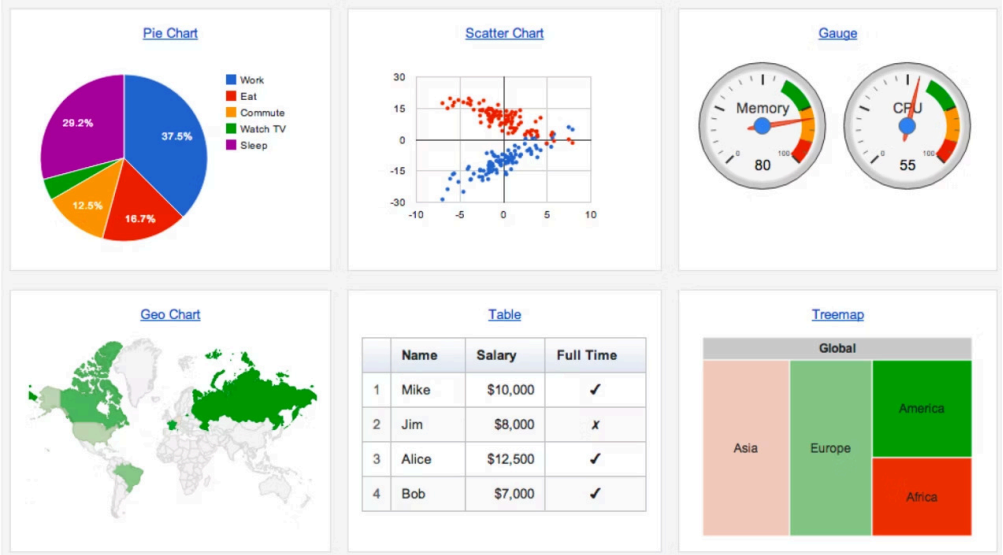
Developer Tools

- RESTful APIs
- JavaScript SDK
- iOS & Android SDKs
- Custom map styling
- Real-time data layers

Powers online maps for websites and mobile applications with customizable, high-performance rendering

Google Chart Tools

- Free, powerful charting library from Google
- Wide variety of chart types (pie, scatter, gauge, geo, table, treemap, etc.)
- Interactive and customizable visualizations
- Easy integration with web applications
- Works with JavaScript and can pull data from various sources



Streamlit

Streamlit

FEATURED

Streamlit

Transform Data Scripts into Web Apps

Turn Scripts into Apps

Convert Python data scripts into interactive web applications

Interactive Frontend

Built-in widgets for user input and interaction

Easy Deployment

Deploy to Streamlit Cloud or your own infrastructure

01

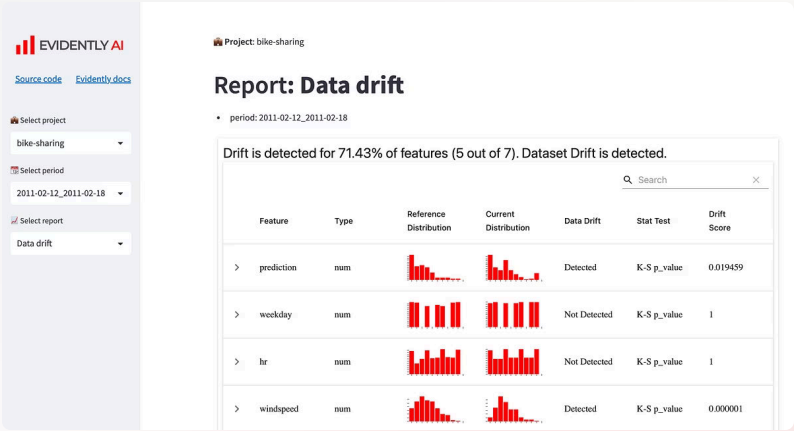
ML Model Interfaces

Create intuitive frontends for machine learning models

02

Data Visualization

Build interactive dashboards for data exploration



Traditional vs. Streamlit Workflow

❌ Traditional ML Deployment

1. Build machine learning model
2. Create API wrapper (Flask/FastAPI)
3. Write HTML/CSS/JavaScript frontend
4. Connect frontend to backend
5. Deploy infrastructure

Time-consuming, requires full-stack skills

✓ Streamlit Approach

1. Build machine learning model
2. Add Streamlit components

Done! Deploy in minutes

Massive time savings by eliminating frontend complexity

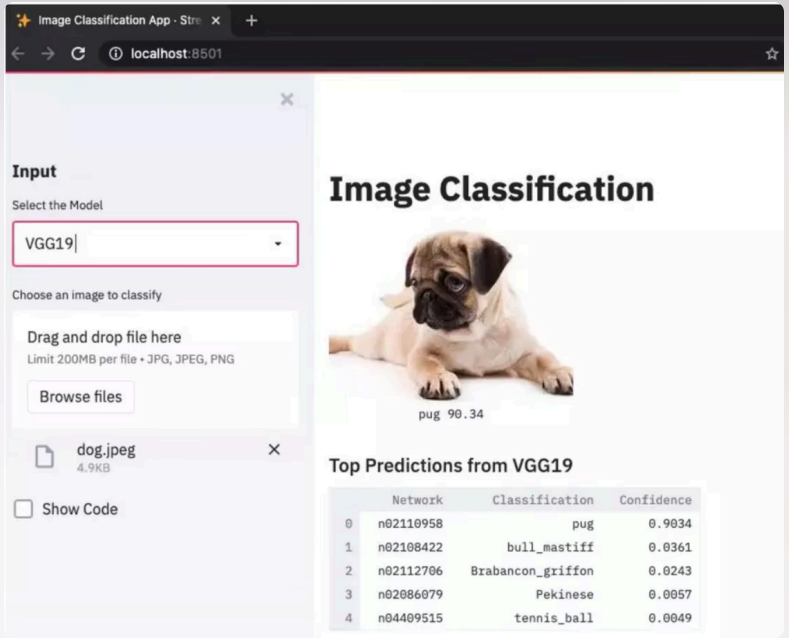
Streamlit handles all the web development complexity behind the scenes

Streamlit Example: Image Classification

ImageNet Classifier in 4 Lines

```
import streamlit as st
uploaded_file = st.file_uploader("Choose an image")
st.image(uploaded_file)
st.write(prediction)
```

Just 4 lines of Streamlit code creates a fully functional image classification interface!



Tutorial - Getting Started with Streamlit

01

Install Streamlit

```
pip install streamlit
```

02

Create Your App

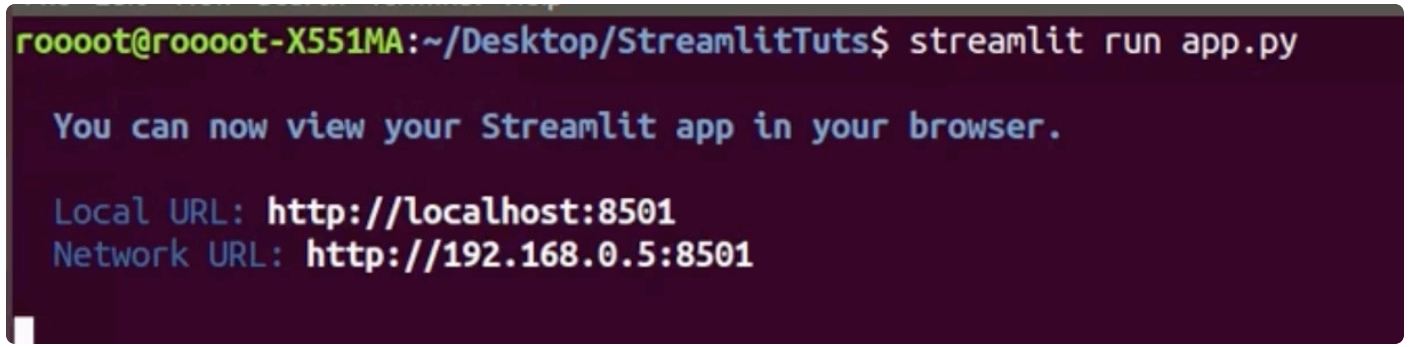
```
touch app.py
```

Add your code to app.py

03

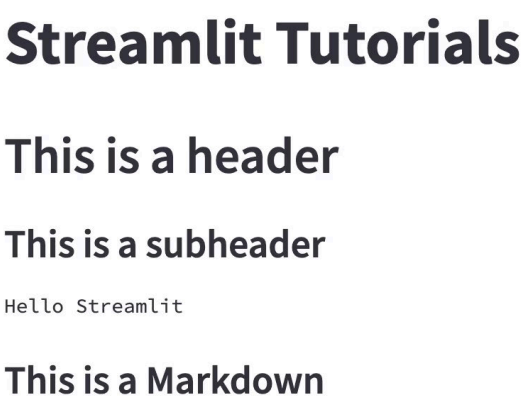
Run Your App

```
streamlit run app.py
```



Your app automatically opens in the browser at `localhost:8501`

Basic Streamlit Components




```python
import streamlit as st
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

# Text/Title
st.title("Streamlit Tutorials")

# Header/Subheader
st.header("This is a header")
st.subheader("This is a subheader")

# Text
st.text("Hello Streamlit")

# Markdown
st.markdown("### This is a Markdown")
```

Rendered Output

Streamlit automatically creates beautiful, responsive layouts

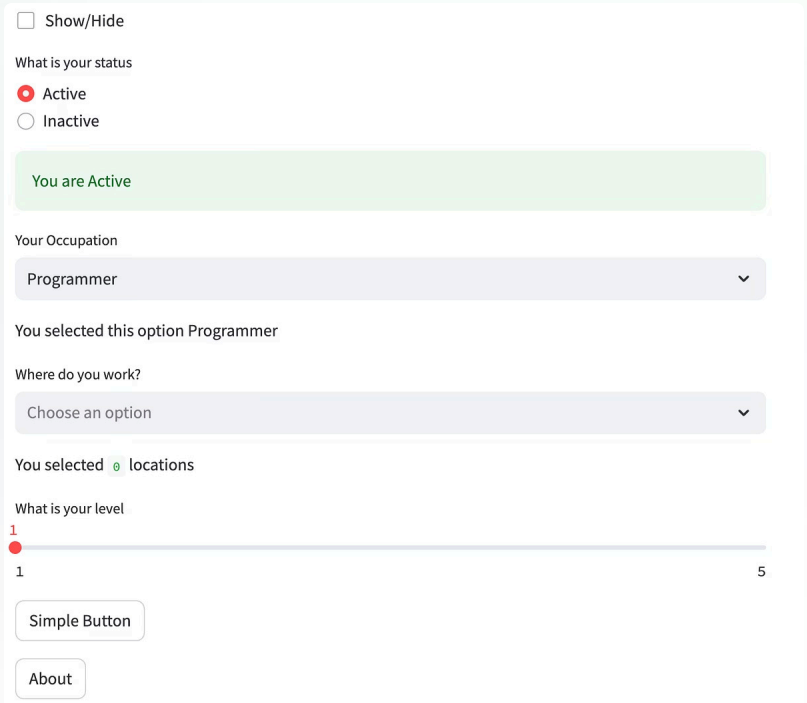
Colorful Text & Error Handling


```
st.success("Successful")

st.info("Information!")

st.warning("This is a warning")

st.error("This is an error Danger")

st.exception("NameError('name three not defined')")
```

Built-in Styling

Streamlit provides colored message boxes for different states

More Visual Components




```python
# Checkbox
if st.checkbox("Show/Hide"):
    st.text("Showing or Hiding Widget")


# Radio Buttons
status = st.radio("What is your status",("Active","Inactive"))

if status == 'Active':
    st.success("You are Active")
else:
    st.warning("Inactive, Activate")

# SelectBox
occupation = st.selectbox("Your Occupation",["Programmer","Data
st.write("You selected this option ",occupation)


# MultiSelect
location = st.multiselect("Where do you work?",("London","New Y
st.write("You selected",len(location),"locations")

# Slider
level = st.slider("What is your level",1,5)


# Buttons
st.button("Simple Button")

if st.button("About"):
    st.text("Streamlit is Cool")
```

User Input Widgets

```python
# Receiving User Text Input
firstname = st.text_input("Enter Your Firstname","Type Here..")
if st.button("Submit"):
    result = firstname.title()
    st.success(result)


# Text Area
message = st.text_area("Enter Your message","Type Here..")
if st.button("Submit2"):
    result = message.title()
    st.success(result)

# Date Input
import datetime
today = st.date_input("Today is",datetime.datetime.now())

# Time
the_time = st.time_input("The time is",datetime.time())
```

Rendered Widgets

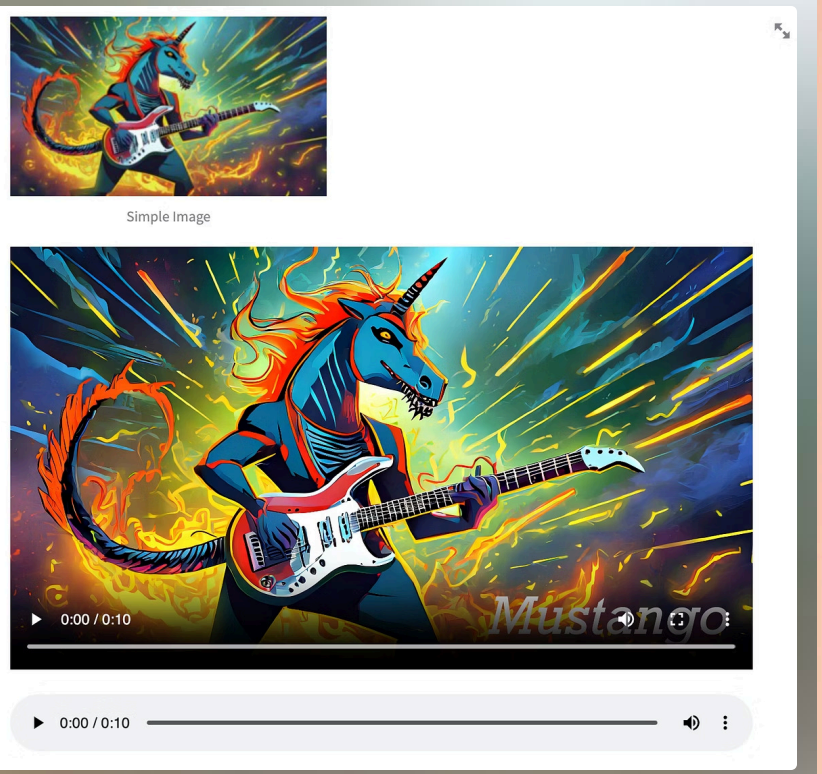
Interactive controls update automatically

All input widgets return their values directly — no callbacks needed!

Multimedia Support

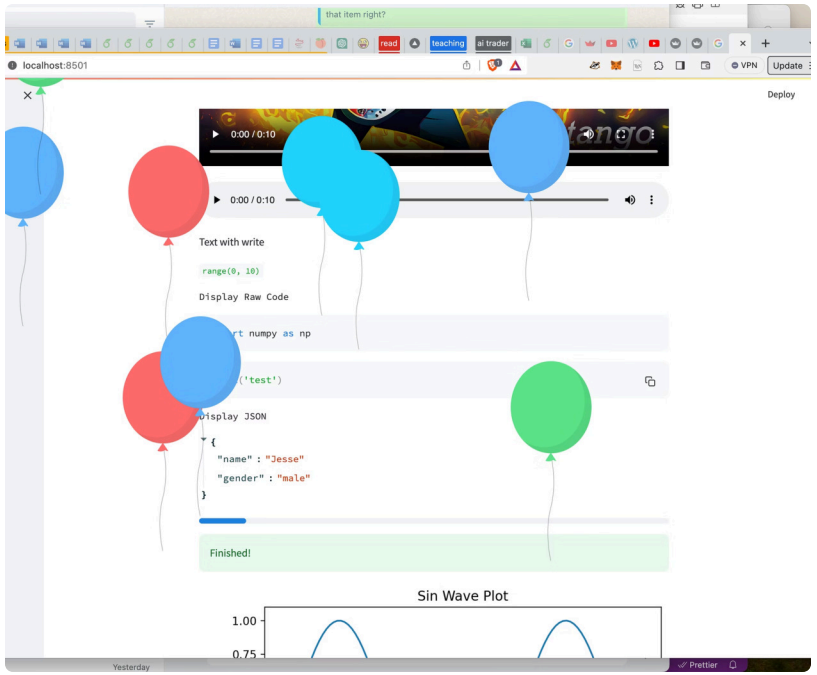
```python
# Images
from PIL import Image
img = Image.open("mustango.jpg")
st.image(img,width=300,caption="Simple Image")


# Videos
vid_file = open("output.mov","rb").read()
# vid_bytes = vid_file.read()
st.video(vid_file)

# Audio
audio_file = open("techno.wav","rb").read()
st.audio(audio_file,format='audio/wav')


# Writing Text/Super Fxn
st.write("Text with write")

st.write(range(10))

# Displaying Raw Code
st.text("Display Raw Code")
st.code("import numpy as np")

# Display Raw Code
with st.echo():
    # This will also show as a comment
    print('test')
```

Display images, audio, and video with simple one-line commands

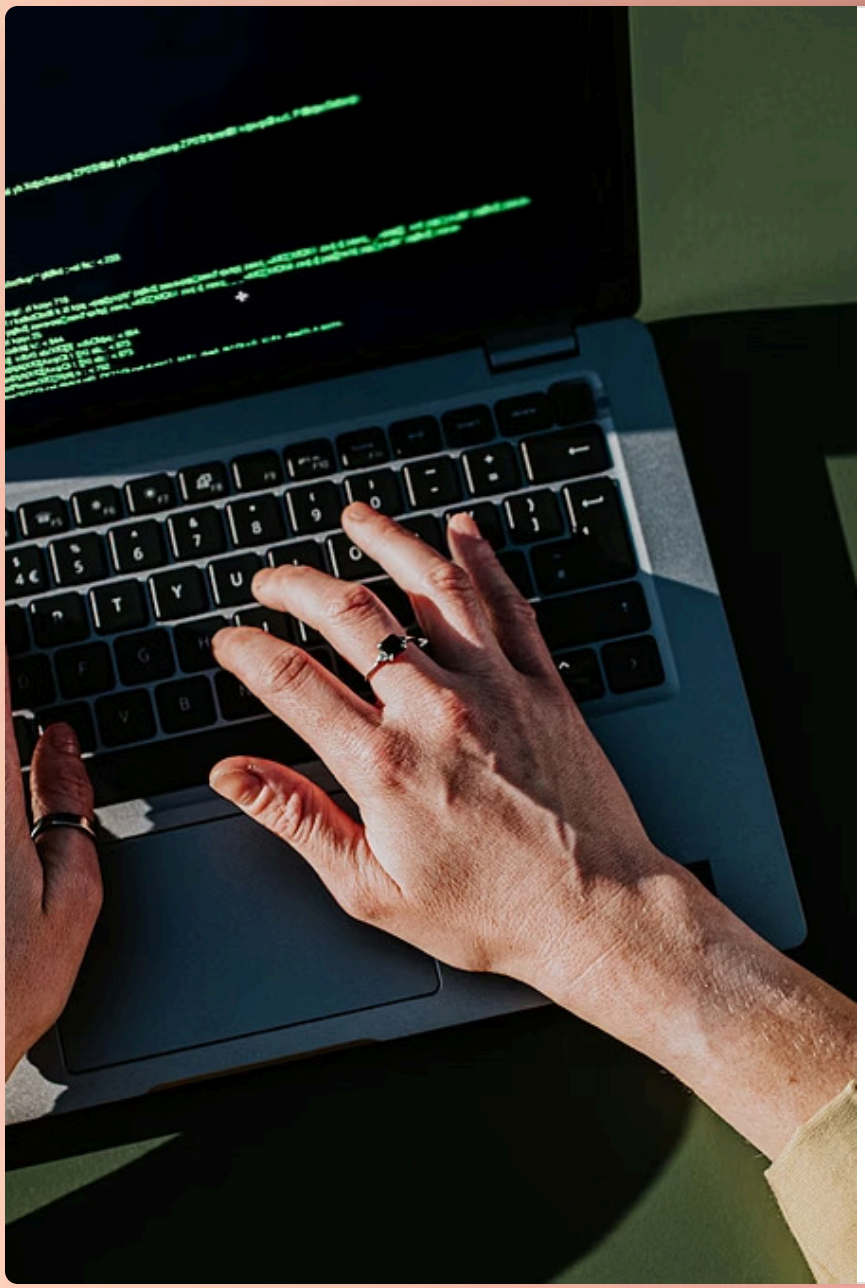


Progress & Waiting States

```python
# Progress Bar
import time
my_bar = st.progress(0)
for p in range(10):
    my_bar.progress(p + 1)


# Spinner
with st.spinner("Waiting .."):
    time.sleep(5)
    st.success("Finished!")


# Balloons
st.balloons()
```



Provide visual feedback during long-running operations



Your Turn: Hands-On Practice

Download Example

Get the example code from eDimension course materials

Experiment

Try the dashboard example and modify it to understand how it works

Useful for project demo!

Quick to create a simple data visualization app using Streamlit

```
# Normal Function
def run_fxn():
    return range(100)

st.write(run_fxn())
```

> "The best way to learn is by doing — start experimenting with the examples and make them your own!"