03: Task Abstraction

Enrico Puppo

Department of Computer Science, Bioengineering, Robotics and Systems Engineering

University of Genova

90529 Data Visualization

1-5 October 2020

https://2020.aulaweb.unige.it/course/view.php?id=4293

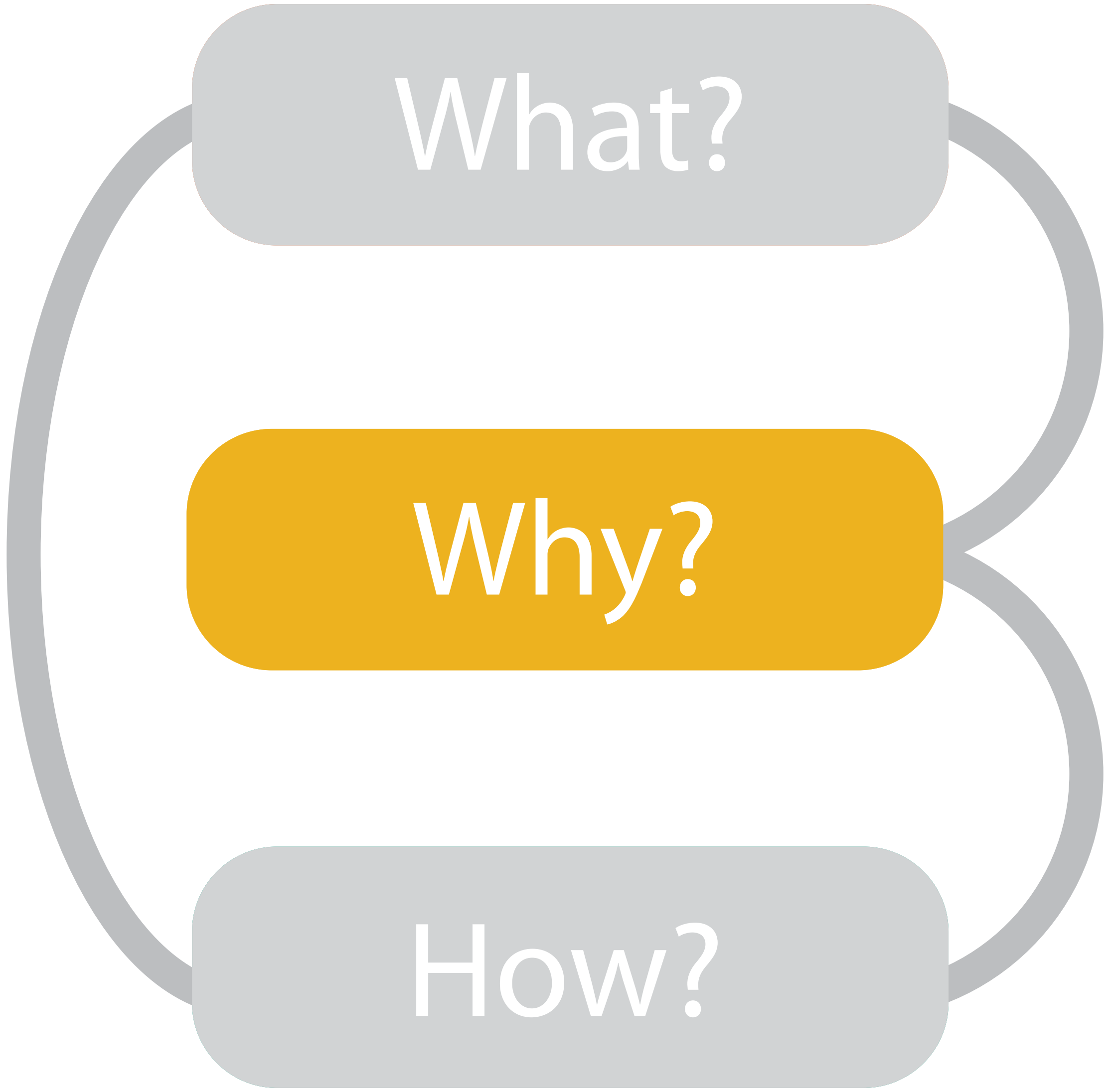
Credits:

material in these slides is partially taken from

- T. Munzner, University of British Columbia
- A. Lex, University of Utah

other credits in the slides

Design cycle

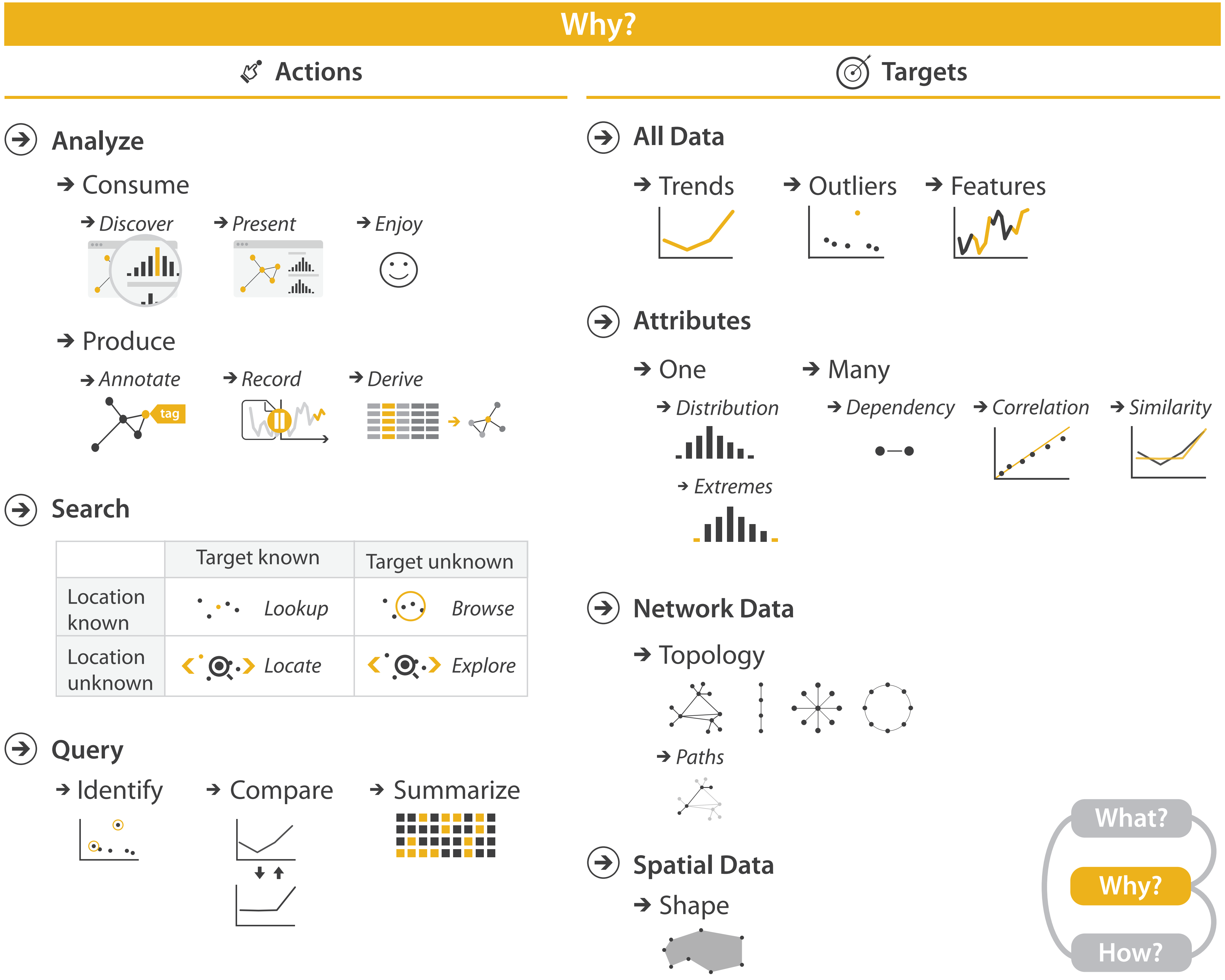


Task abstraction

- Goal:
  - infinite number of domain tasks
  - can be broken down into simpler abstract tasks
  - translate domain-specific terms into generic concepts
  - advantages of abstract tasks:
    - can be addressed in a systematic way
    - capture the purpose of vis app
    - plan user tasks and how they use data
  - complex activities: sometimes chains of tasks, output of one is input to the next
  - tasks may require transforming original data by deriving new data
- Global framework:
  - Actions: what the app & user do
  - Targets: what data are used/affected by actions

Actions and Targets

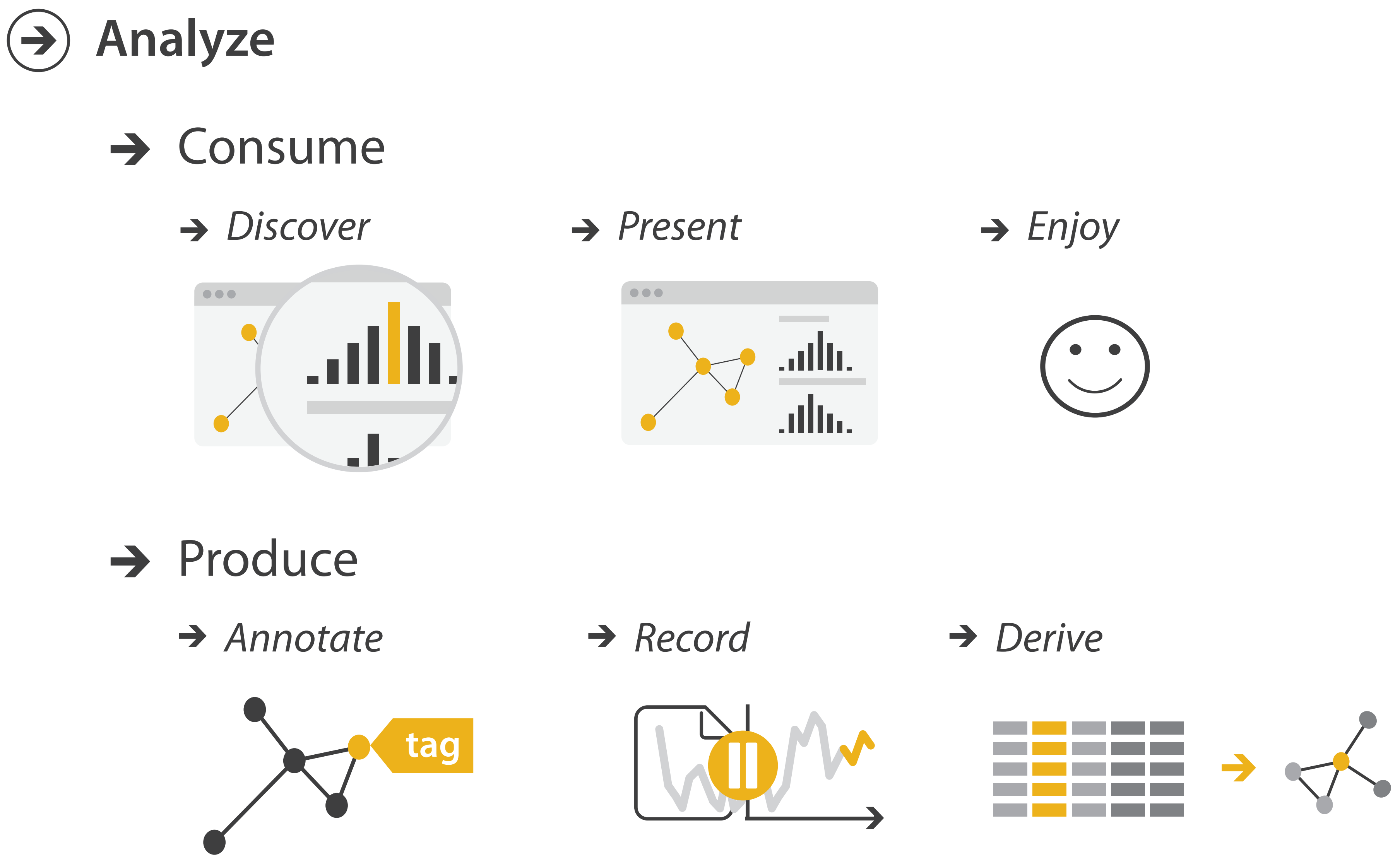
- {action, target} pairs
  - –*discover dependency*
  - –*present distribution*
  - –*annotate features*
  - –*record trends*
  - –*derive correlation*
  - –*lookup path*
  - –*locate outliers*
  - –*browse shape*
  - –*explore similarity*
  - –*identify extremes*
  - –*compare trends*
  - –*summarize topology*



	Target known	Target unknown
Location known	*Lookup*	*Browse*
Location unknown	*Locate*	*Explore*

High-level actions: Analyze

- consume
  - discover vs present
      - classic split
    - aka explore vs explain
  - enjoy
      - newcomer
      - aka casual, social
- produce
  - annotate, record
  - derive
    - crucial design choice



Analyze

→ Consume

→ *Discover* → *Present* → *Enjoy*

→ Produce

→ *Annotate* → *Record* → *Derive*

Consume

- Assumption: data format is already suitable for computation

- Explore/Discover: find new knowledge in data
  - User-driven process
  - Generate hypothesis → Verify conjecture
  - Vis designer doesn’t know in advance what to show
    - user must be able to explore all aspects of data
    - need for sophisticated interaction
    - exploration may need producing new data (Derive) as a sub-task
    - most difficult scenario for the designer

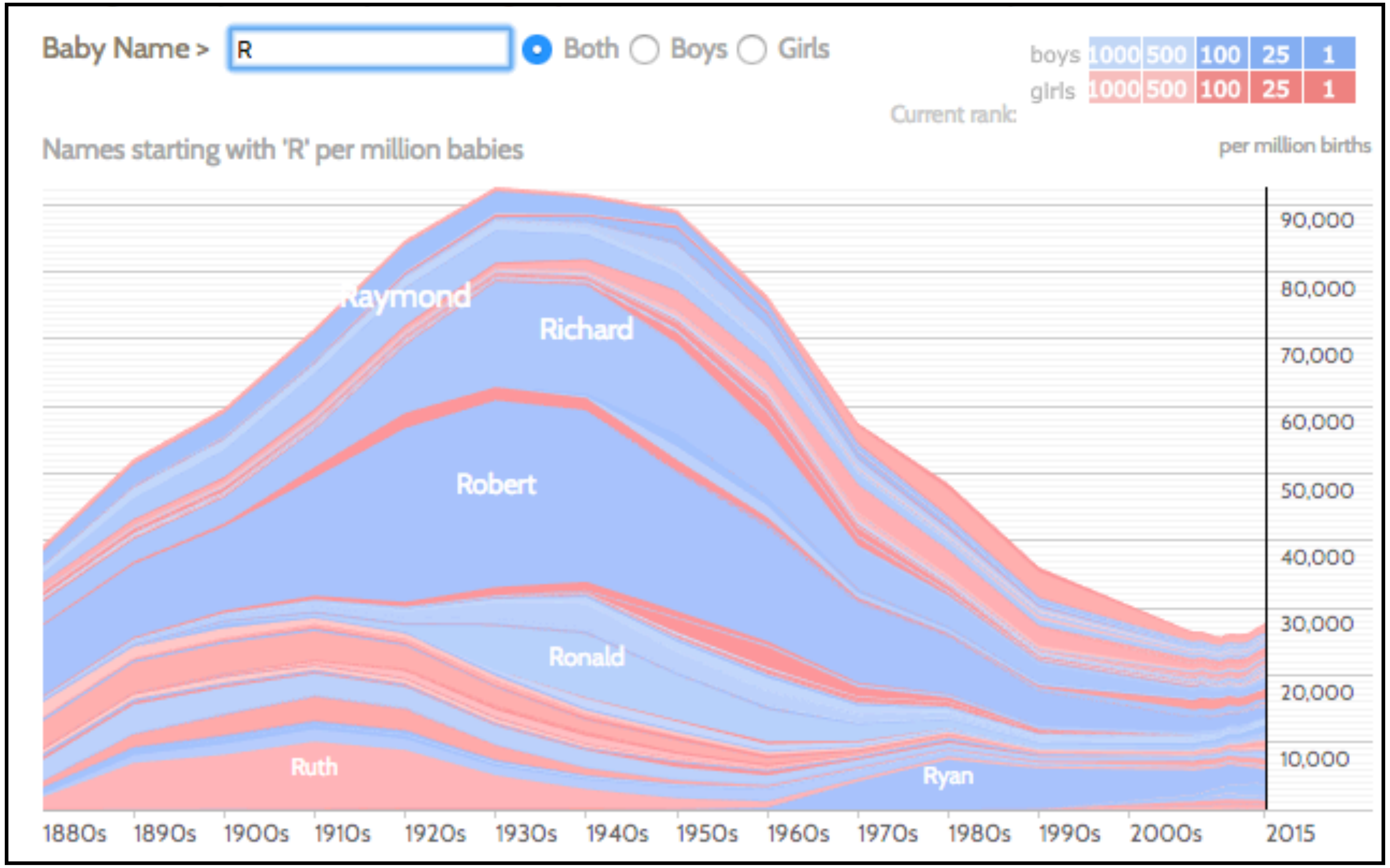
Consume

- Present: communicate information
  - ex.: decision making, forecasting, planning, instruction
  - presenter knows facts she/he wants to communicate to an audience
  - Discover session may generate input to a Present session (see Record)
  - moderate or no interaction
  - storytelling

- Enjoy: casual user, curiosity driven
  - assumption: designer knows the goal of users (not always true)
  - ease of use: clear and explicit semantics, intuitive interaction
  - vis app must draw user's attention to important actions and targets for the goal

Example: enjoy

- NameVoyager:
  - explore baby names and trends



Baby Name > R

● Both ○ Boys ○ Girls

Names starting with 'R' per million babies

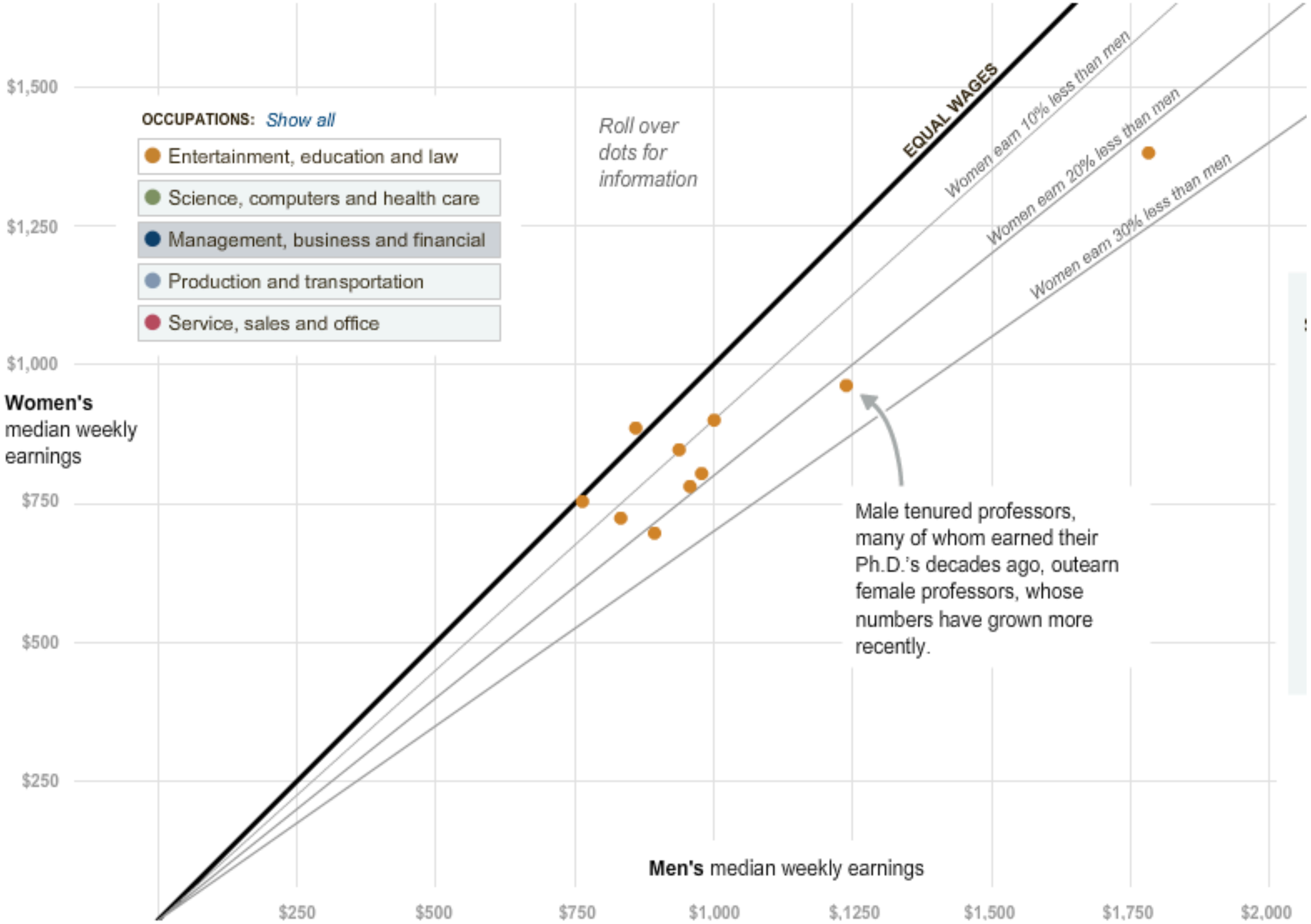
Produce

- Assumption: new data or a modified format are needed for computation
- Typically: output of Produce used as input for the next task

- Annotate:
  - adding annotations to data or groups of data (e.g., as new attributes)
  - properties not apparent in original format, derived/discovered by analyzing data
  - requires heavy user interaction

- Record:
  - save vis elements as persistent artifacts (typically for subsequent Present session)
  - Graphical history: record output of subsequent tasks that transform data

Example: Annotate



Male tenured professors, many of whom earned their Ph.D.'s decades ago, outearn female professors, whose numbers have grown more recently.

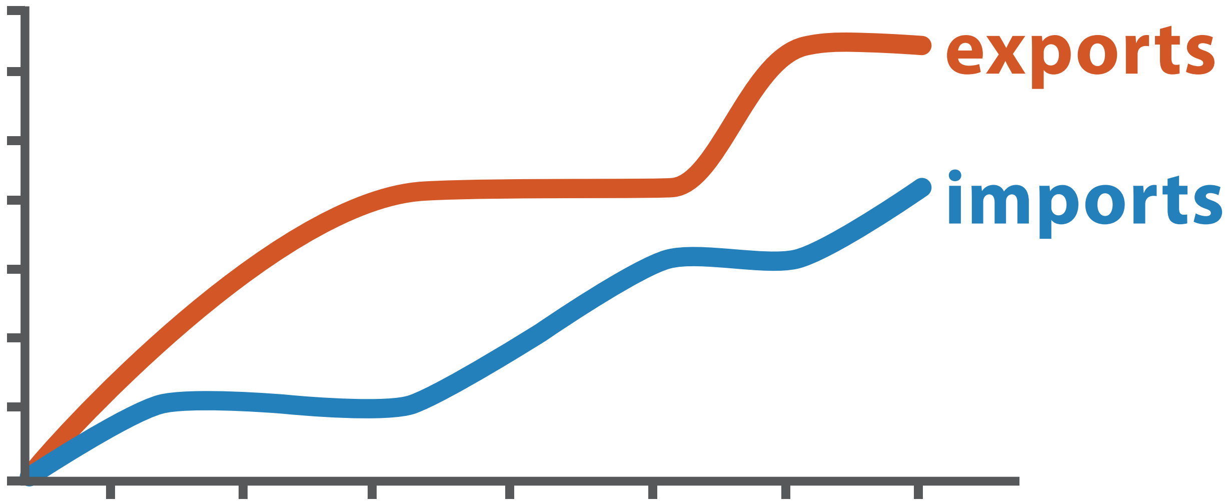
Produce

- Derive/Transform:
  - produce additional data elements/attributes on the basis of existing ones
  - most common task planned by vis designers
  - sophisticated apps may allow user to derive
  - derived attributes extend the dataset
  - how to derive:
    - query external database (new information)
    - combine existing attributes with arithmetic, logic, statistics

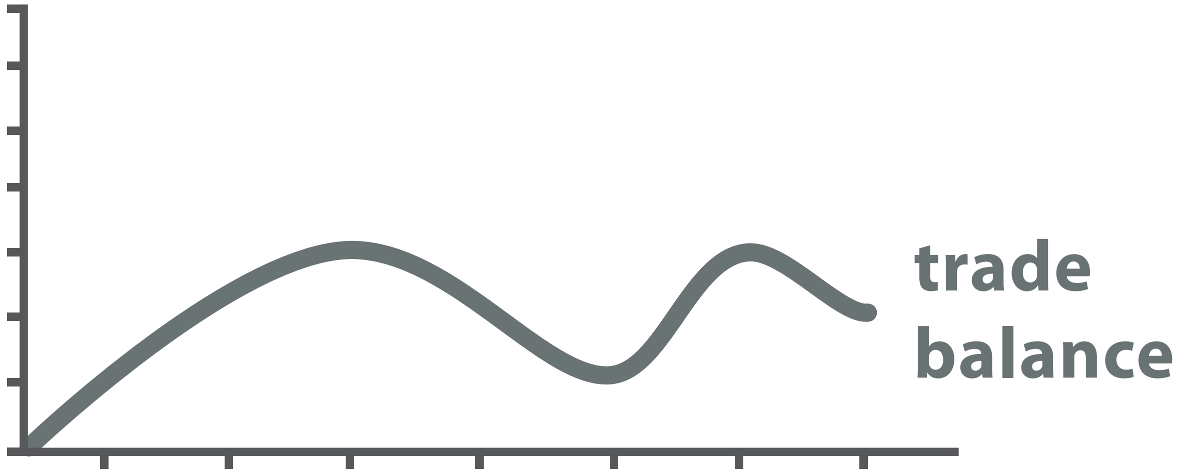
- Derive operations are crucial in designing a vis app

Derive

- don’t just draw what you’re given!
  - decide what the right thing to show is
  - create it with a series of transformations from the original dataset
  - draw that
- one of the major strategies for handling complexity



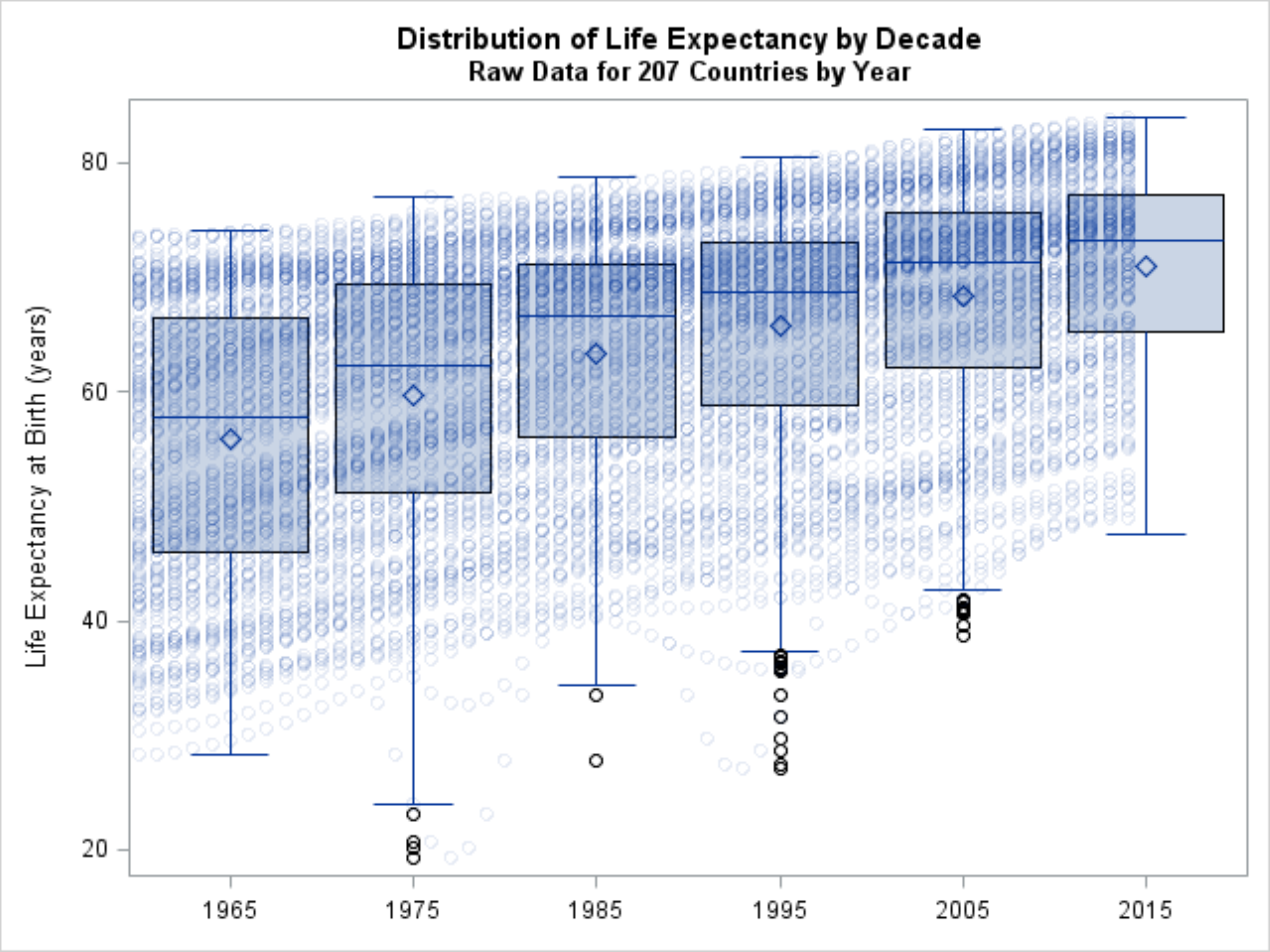
Original Data



trade balance = exports − imports

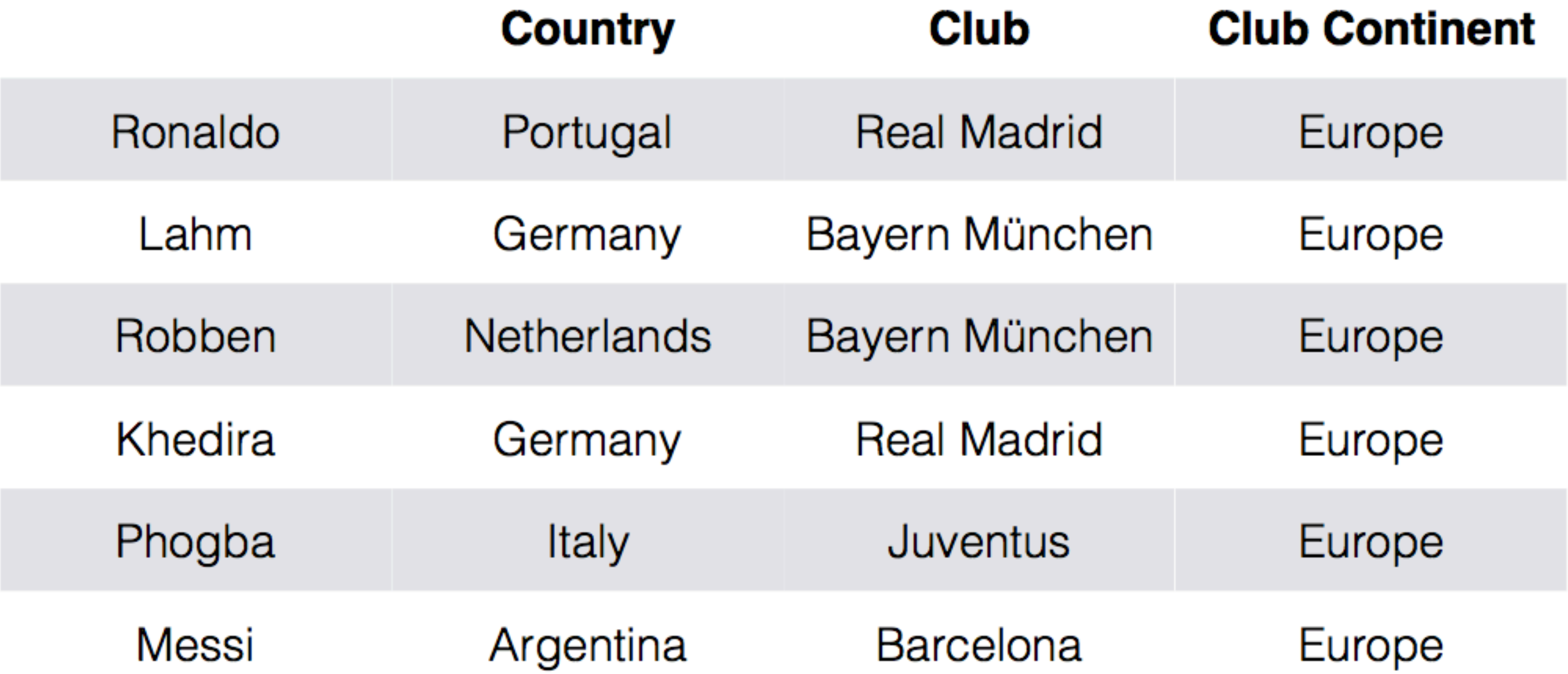
Derived Data

Example: Derive (with statistics)



Example: Derive (with aggregation)

Input table (players in World Cup 2014):

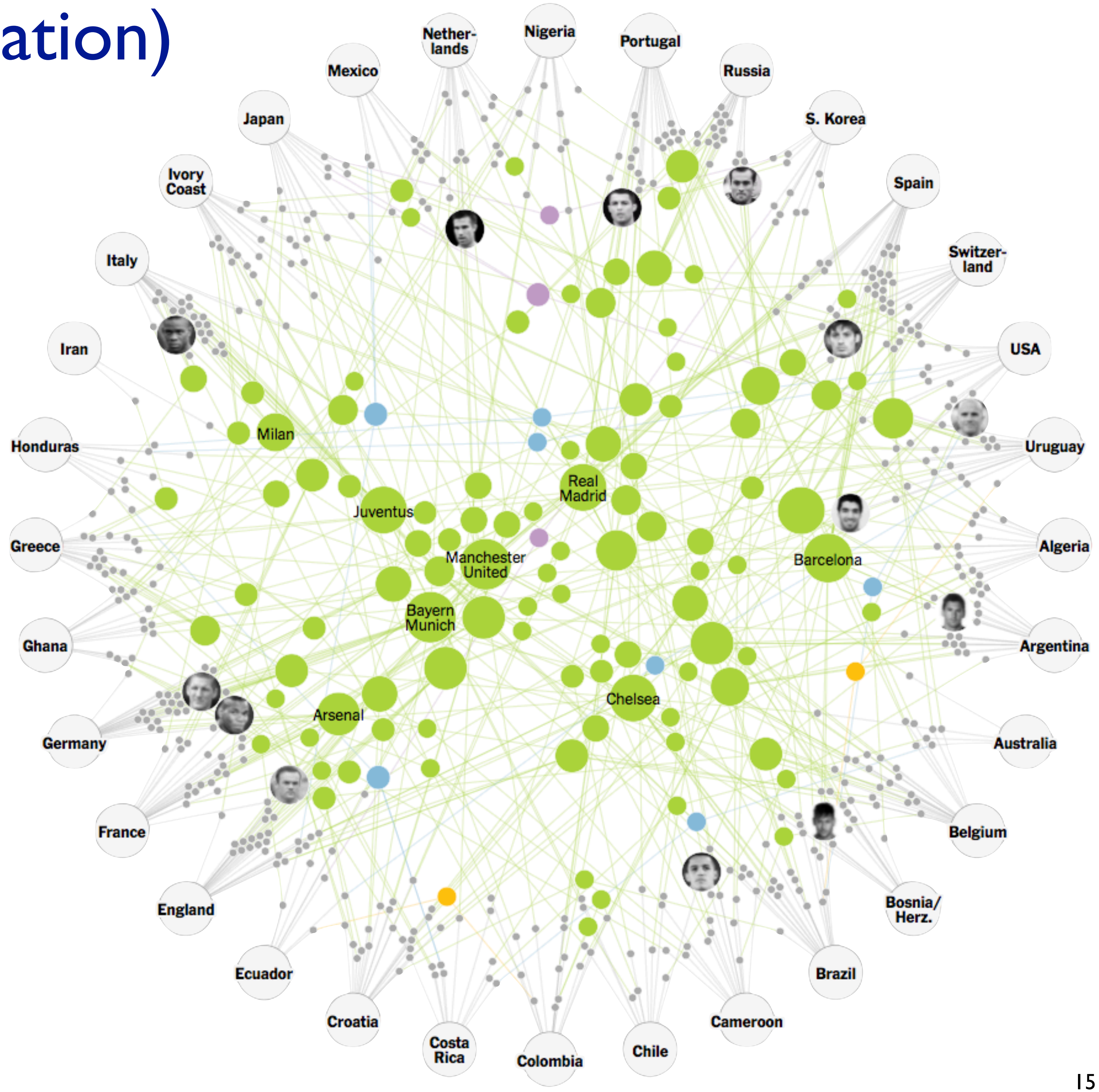


	Country	Club	Club Continent
Ronaldo	Portugal	Real Madrid	Europe
Lahm	Germany	Bayern München	Europe
Robben	Netherlands	Bayern München	Europe
Khedira	Germany	Real Madrid	Europe
Phogba	Italy	Juventus	Europe
Messi	Argentina	Barcelona	Europe

Example: Derive (with aggregation)

Derived network:

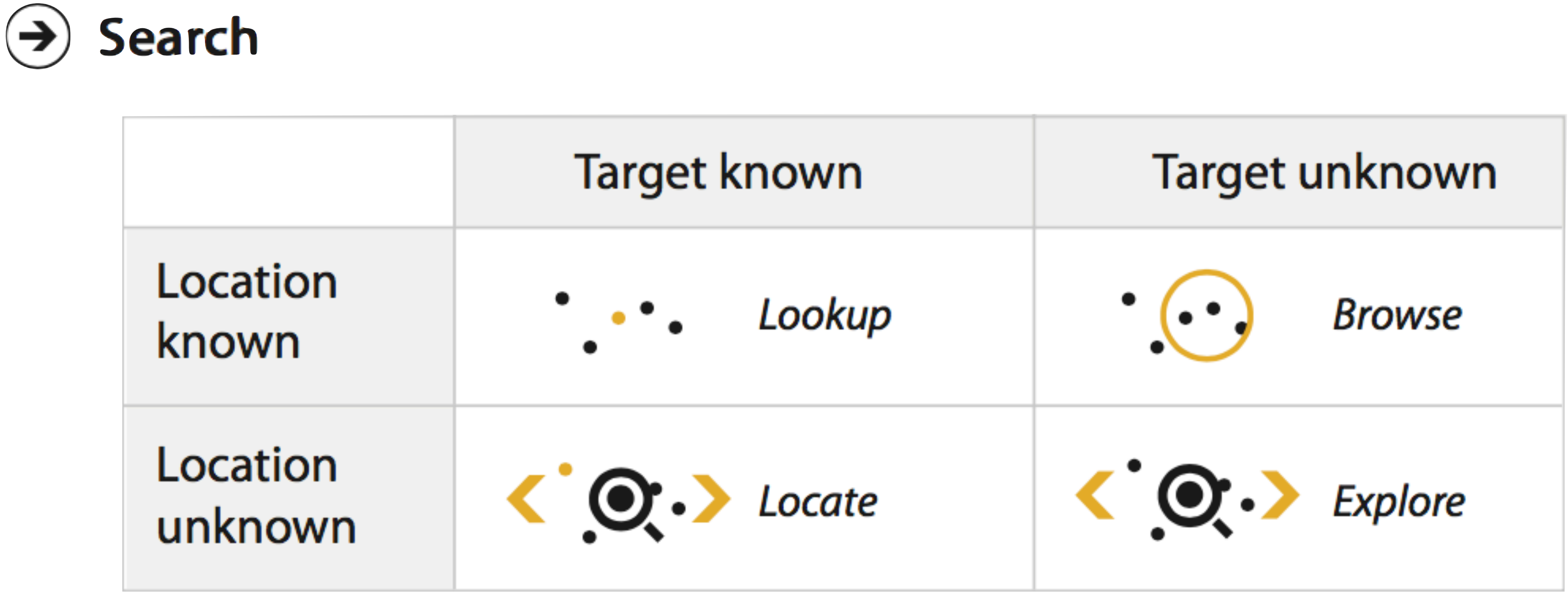
- Clubs with players on at least two national teams



Mid-level actions: search

- Lookup:
  - find info, find related items
- Locate:
  - find relation with context
- Browse:
  - find items with certain characteristics
- Explore:
  - build context and relationships, find relevant items

- what does user know?
  - target, location



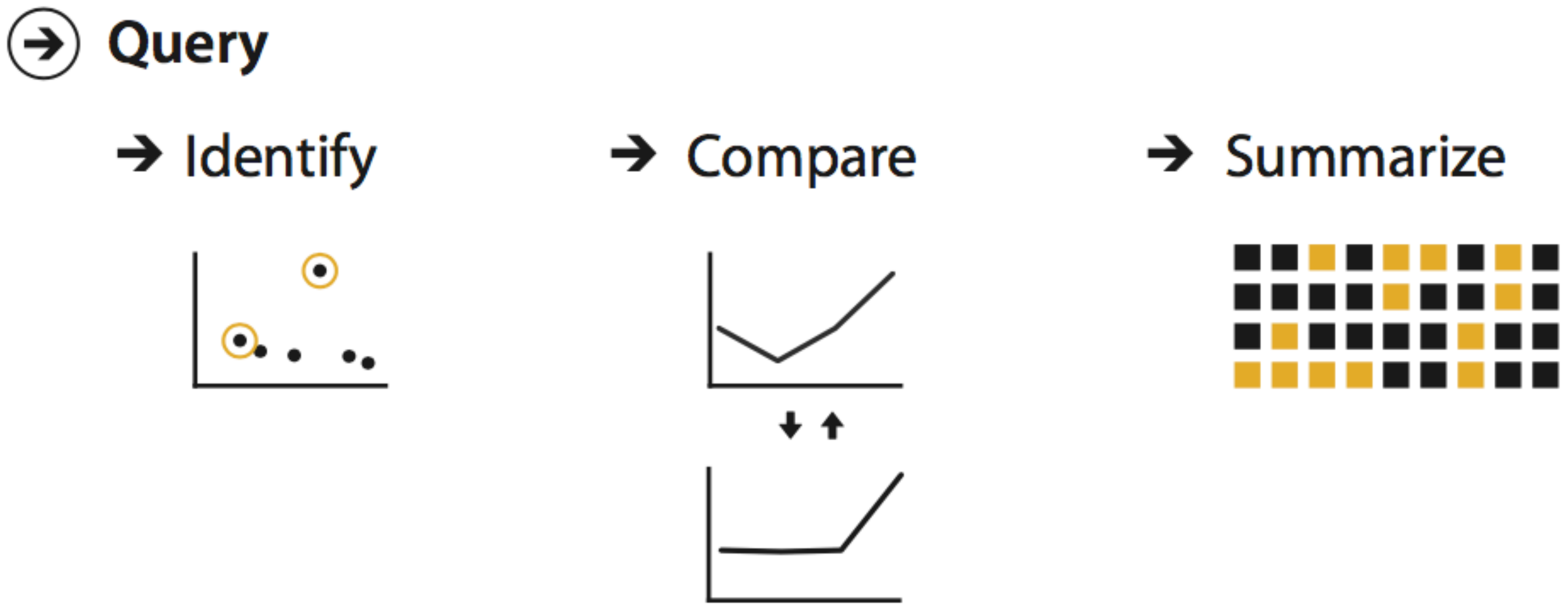
➔ Search

	Target known	Target unknown
Location known	*Lookup*	*Browse*
Location unknown	*Locate*	*Explore*

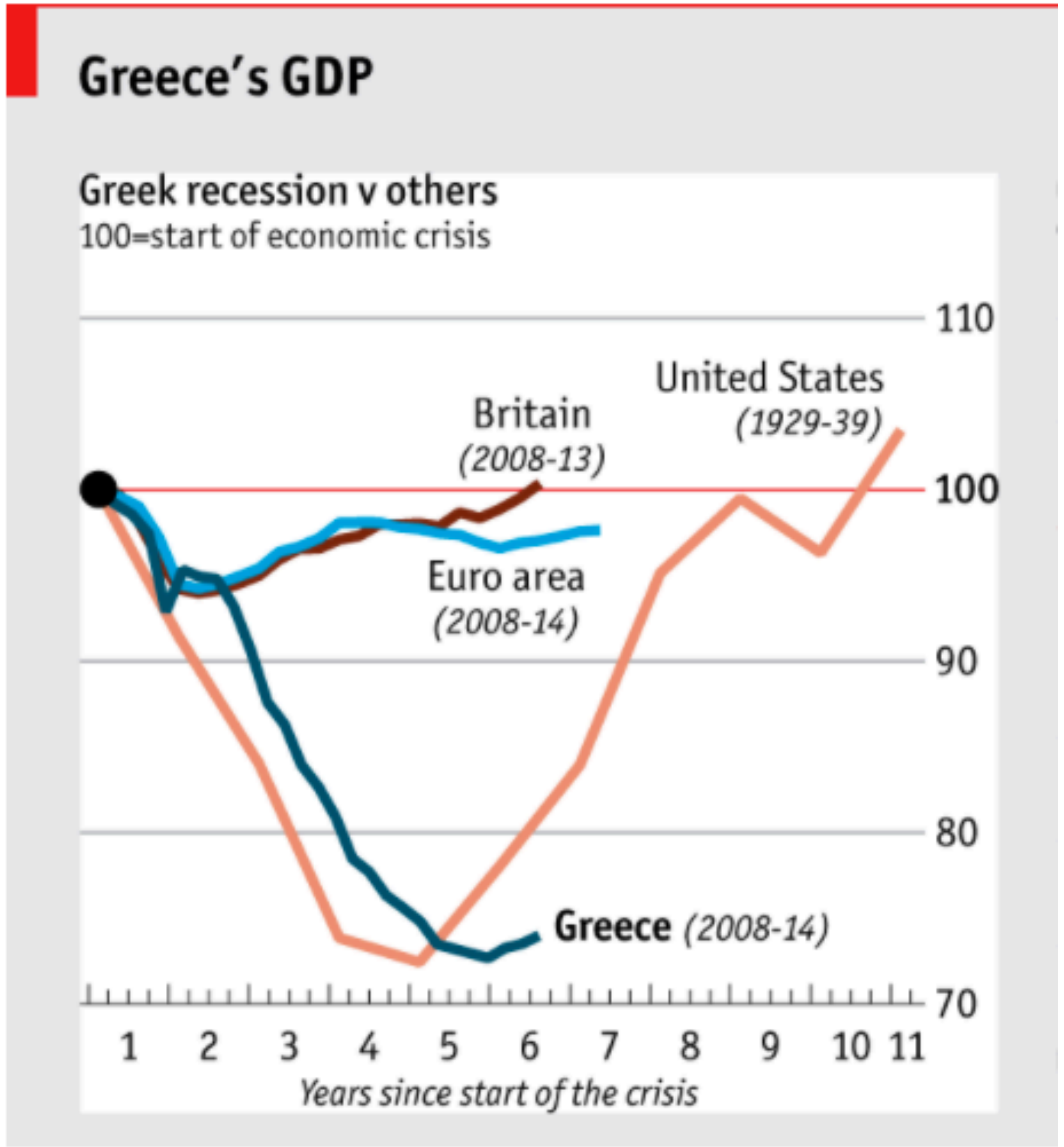
Low-level actions: query

- Identify:
  - single target
- Compare:
  - multiple targets
  - comparison according to one or more attributes
  - show similarity and distances
- Summarize:
  - all targets
  - provide and overview (summary, statistics)

- obtain info about item / group
- and/or relation with context
- categorized on #targets



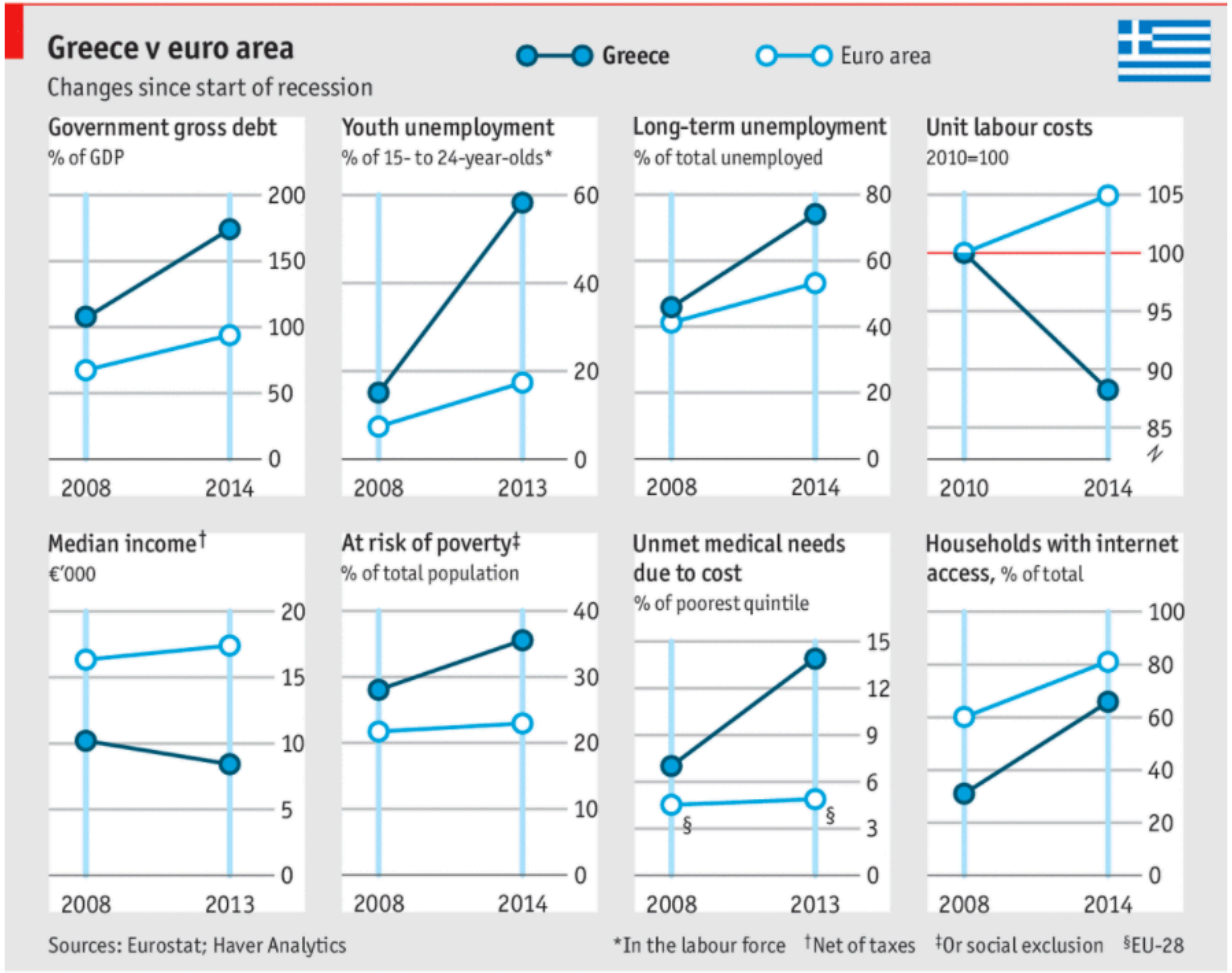
Example: Compare



Greece's GDP

Greek recession v others

100=start of economic crisis



Greece v euro area

Changes since start of recession

Government gross debt
% of GDP

Youth unemployment
% of 15- to 24-year-olds*

Long-term unemployment
% of total unemployed

Unit labour costs
2010=100

Median income†
€'000

At risk of poverty‡
% of total population

Unmet medical needs due to cost
% of poorest quintile

Households with internet access, % of total

Sources: Eurostat; Haver Analytics

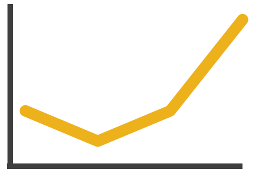
*In the labour force †Net of taxes ‡Or social exclusion §EU-28

Economist.com

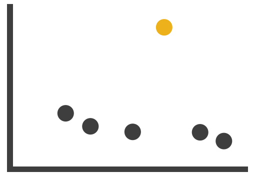
Targets

➔ All Data

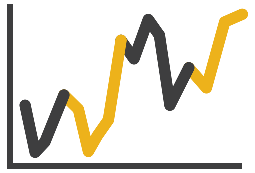
➔ Trends



➔ Outliers



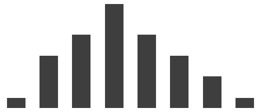
➔ Features



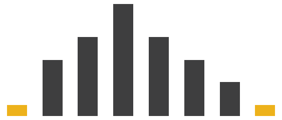
➔ Attributes

➔ One

➔ *Distribution*



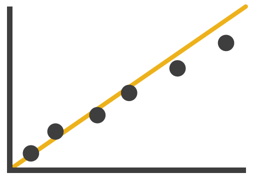
➔ *Extremes*



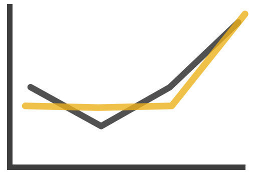
➔ Many

➔ *Dependency*

➔ *Correlation*

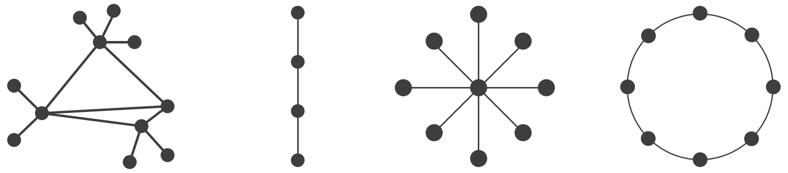


➔ *Similarity*

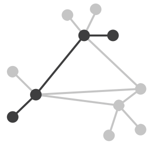


➔ Network Data

➔ Topology

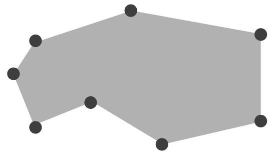


➔ *Paths*



➔ Spatial Data

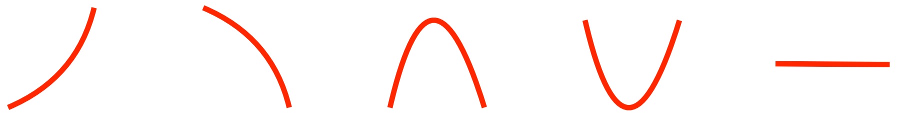
➔ Shape



Targets

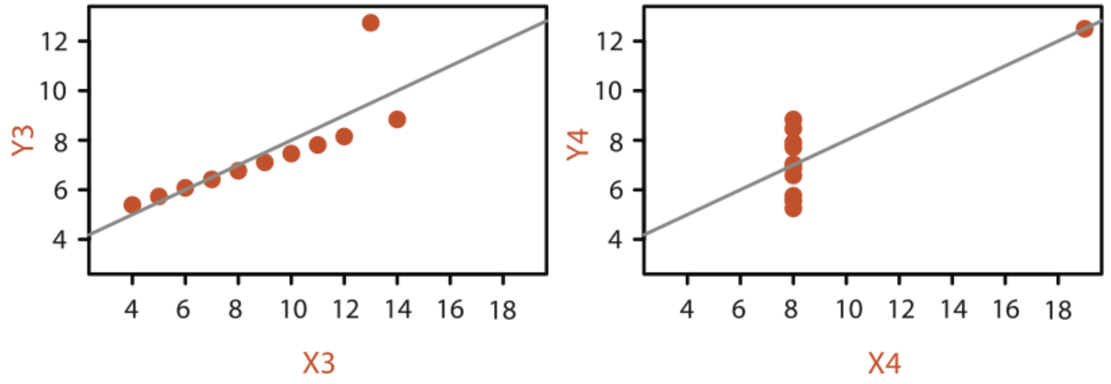
All data

- Trends:
  - high level characterization of a pattern in the data
  - increase, decrease, peak, troughs, plateaus



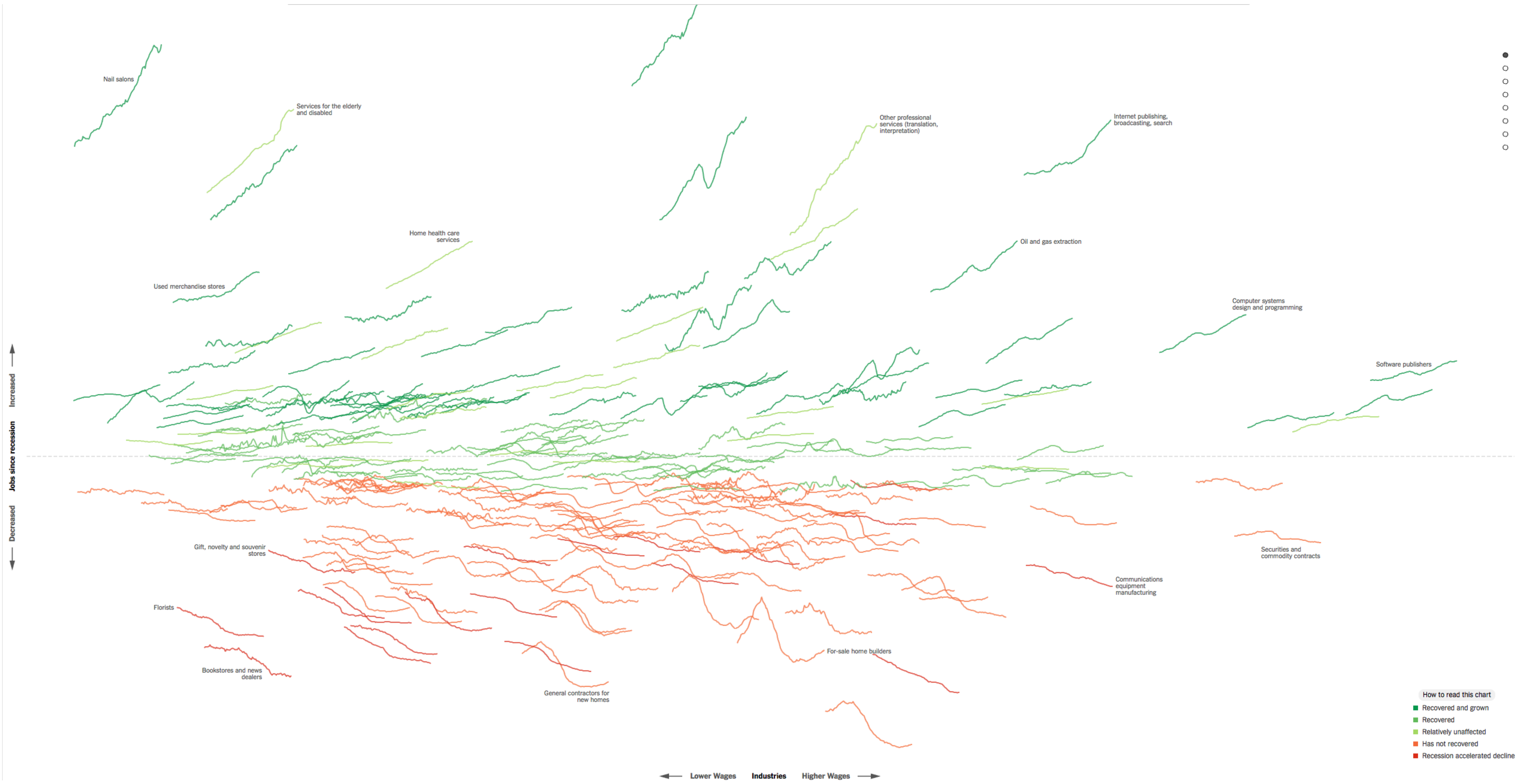
- Outliers:
  - data that don't fit the distribution / trend of the others
  - outliers perturb statistics if they are not detected and removed
  - outliers may bring out either anomalies or important novelties

Remember the Ancombe's quartet?



Example: Trends & Outliers

- Trends: how did job market develop since recession overall?
- Outliers: look at real estate related jobs

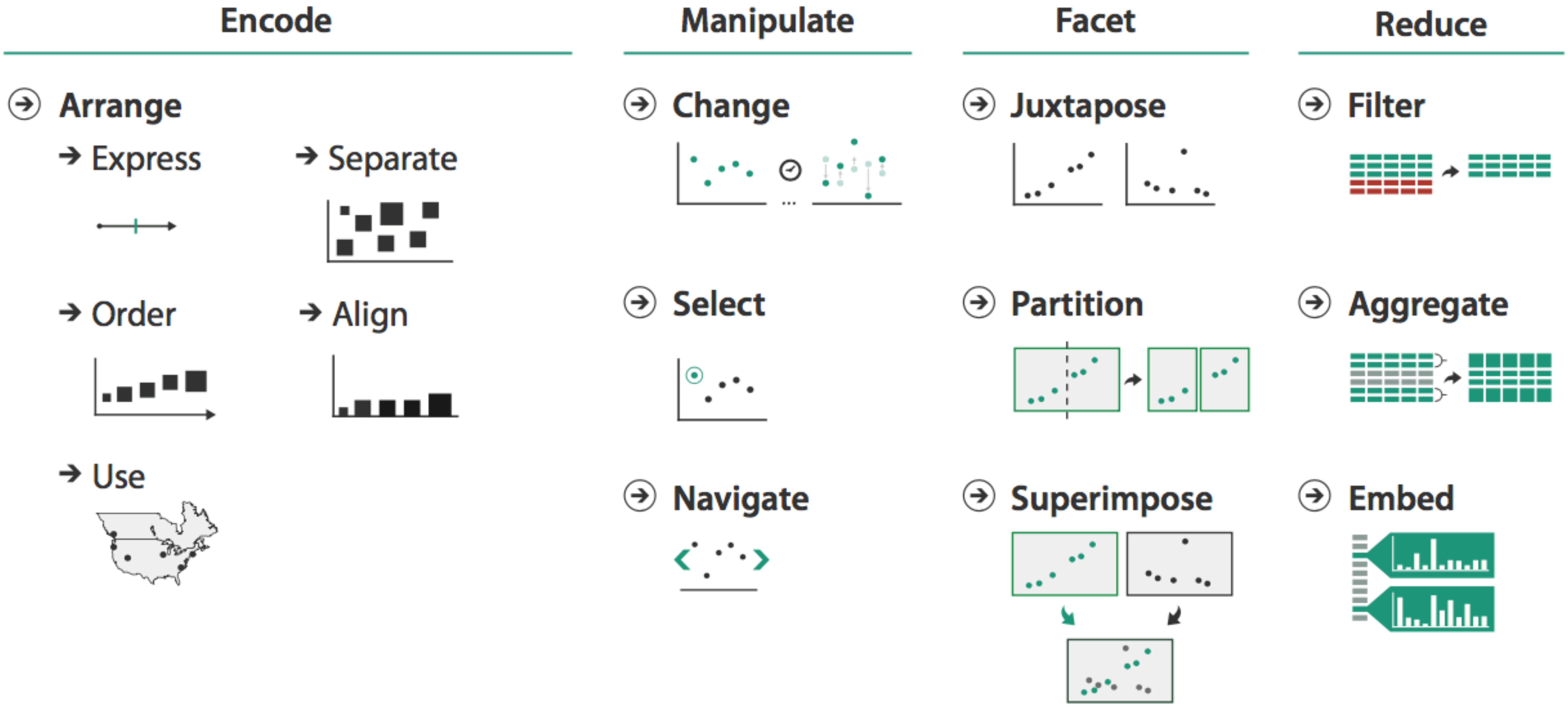


Targets

All data

- Features:
  - structures of interest
  - task dependent, may be related with
    - showing a given pattern
    - having a certain attribute within a given range
    - having a certain combination / correlation of attributes
    - forming a cluster
    - forming a homogeneous region in spatial data
    - …

How: a preview



Encode

Arrange

→ Express

→ Separate

→ Order

→ Align

→ Use

Manipulate

Change

Select

Navigate

Facet

Juxtapose

Partition

Superimpose

Reduce

Filter

Aggregate

Embed

How: a preview

- Encode
  - main class of abstract methods to arrange data into vis
  - three categories depending on data:
    - tables
    - networks & trees
    - spatial
- Strategies for handling complexity:
  - Derive new data (seen in Why, because it doesn't involve vis directly)
  - Manipulate view over time
  - Facet data into multiple views
  - Reduce items and attributes
  - Embed focus & context

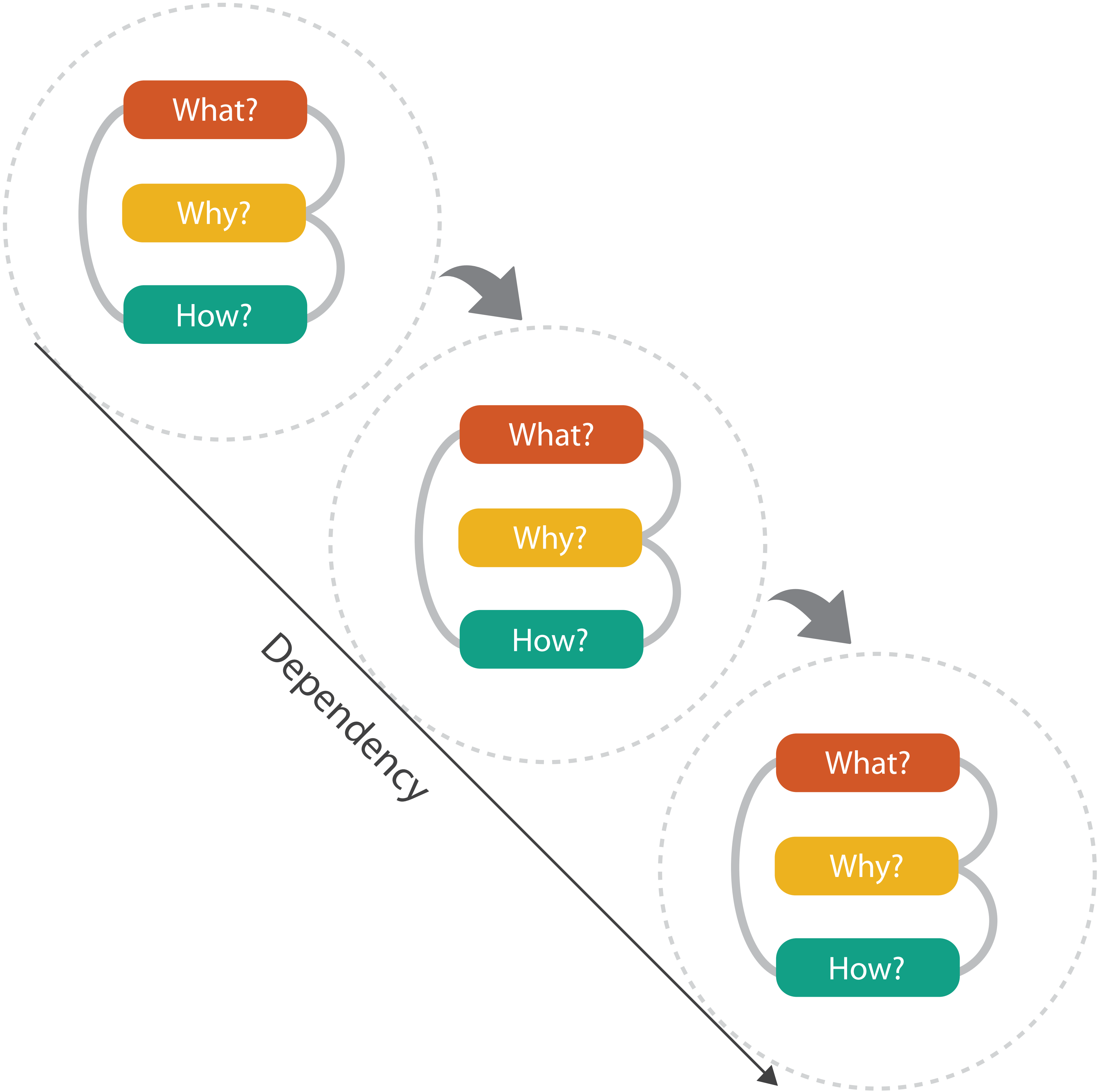
will see them in detail

How: a preview

- The “How” methods are treated in the rest of the course
- Vis “Idioms” are the practical means to implement “How” methods
  - we have already met a few of them
    - bar/line/area charts
    - node-link diagrams
    - boxplots
  - we will investigate them in detail as we go on
  - we will learn how to implement some of them

Chained sequences

- output of one is input to next
  - express dependencies
  - separate means from ends



Next Time

- we will be starting the technical subjects:
  - examples during lecture
  - no slides!
  - be present!

- to read
  - IDV Ch. 3: Technology fundamentals (except Javascript)
  - HTML/CSS/SVG tutorial:

    https://cscheid.net/courses/spr15/cs444/lectures/week2.html