# From R to Python to OCaml:
## Set-Theoretic Analysis for the (Social) Sciences

Claude Rubinson
University of Houston—Downtown
cjr@grundrisse.org rubinsonc@uhd.edu
http://grundrisse.org/QCA/ http://gator.uhd.edu/~rubinsonc/

Houston Functional Programmers
Houston, Texas
May 17, 2017

# Overview

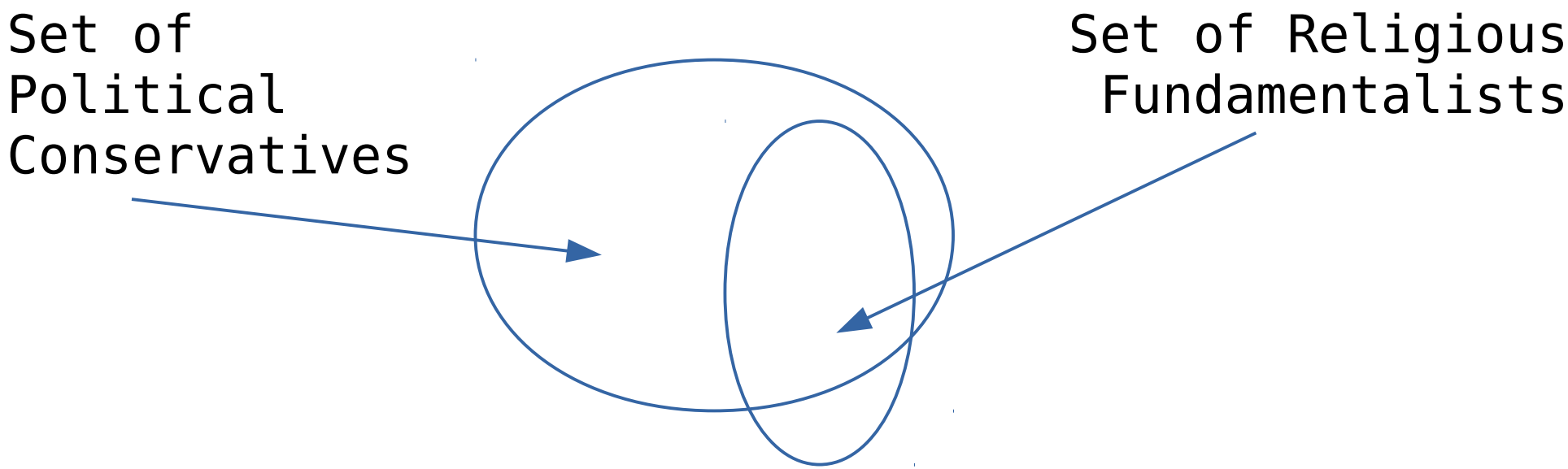
- Introduction to QCA
- History of QCA software
- First (mis-)steps in developing acq & Kirq: the fsQCA package for R
- Second implementation: Python
  - Benefits and limitations of Python
- Third implementation: OCaml
  - Why OCaml?
  - Initial benefits of OCaml and functional programming
  - Software architecture
  - Unresolved issues

# What is Qualitative Comparative Analysis?

- A method of conducting social research by analyzing subset relationships, using Boolean algebra

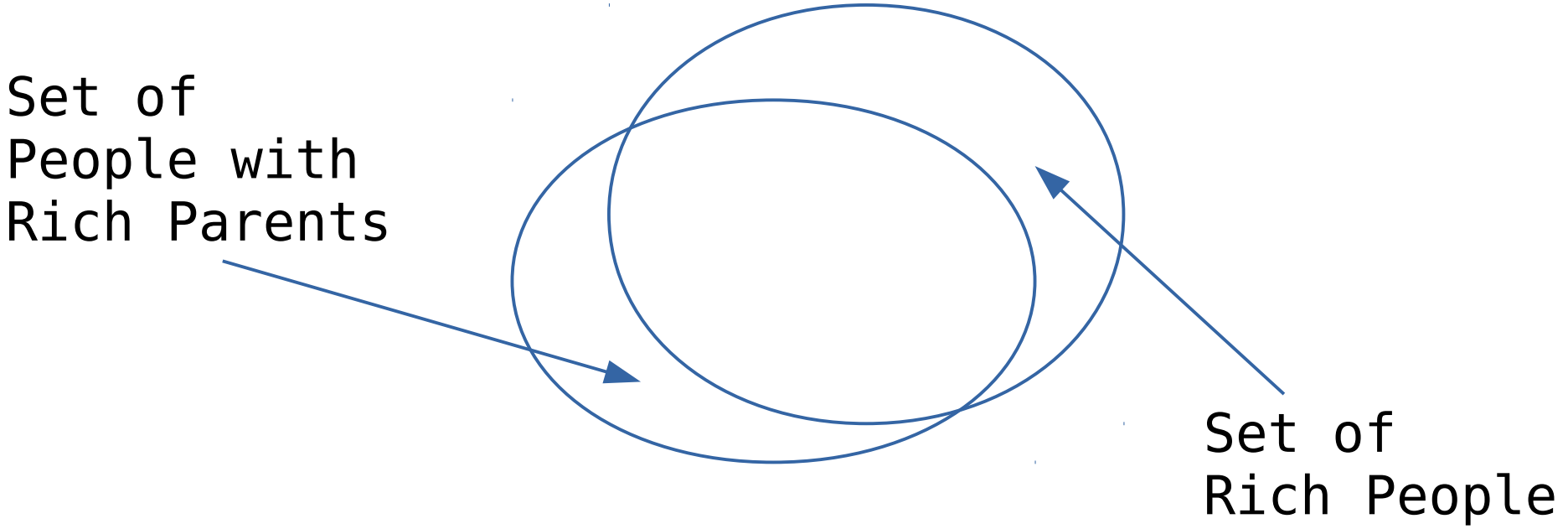
# What is Qualitative Comparative Analysis?

- A method of conducting social research by analyzing subset relationships, using Boolean algebra
- Example: Religious fundamentalists tend to be politically conservative.



# What is Qualitative Comparative Analysis?

- A method of conducting social research by analyzing subset relationships, using Boolean algebra
- Example: Wealthy individuals tend to come from privileged families.

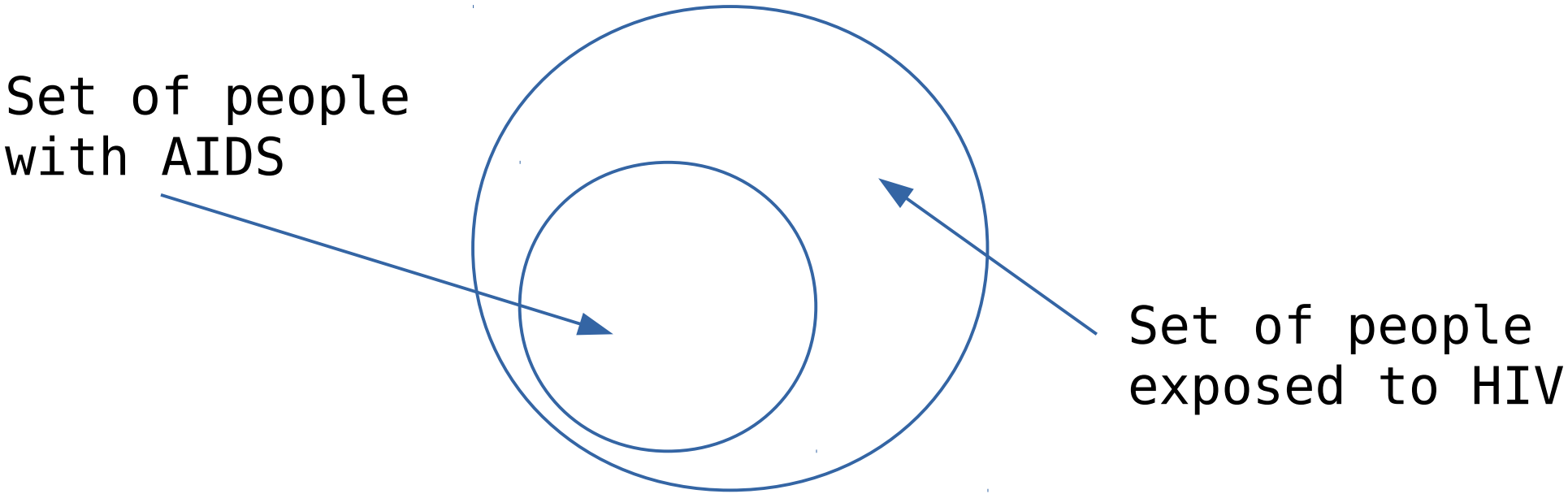


# What is Qualitative Comparative Analysis?

- Particularly concerned with two types of causal relationships: necessary conditions and sufficient conditions
  - Absence of necessary condition means that outcome will (probably) not occur
  - Presence of sufficient condition means that outcome will occur, all or most of the time
  - Necessary and sufficient conditions may be complex (“multiple conjunctural causation”)

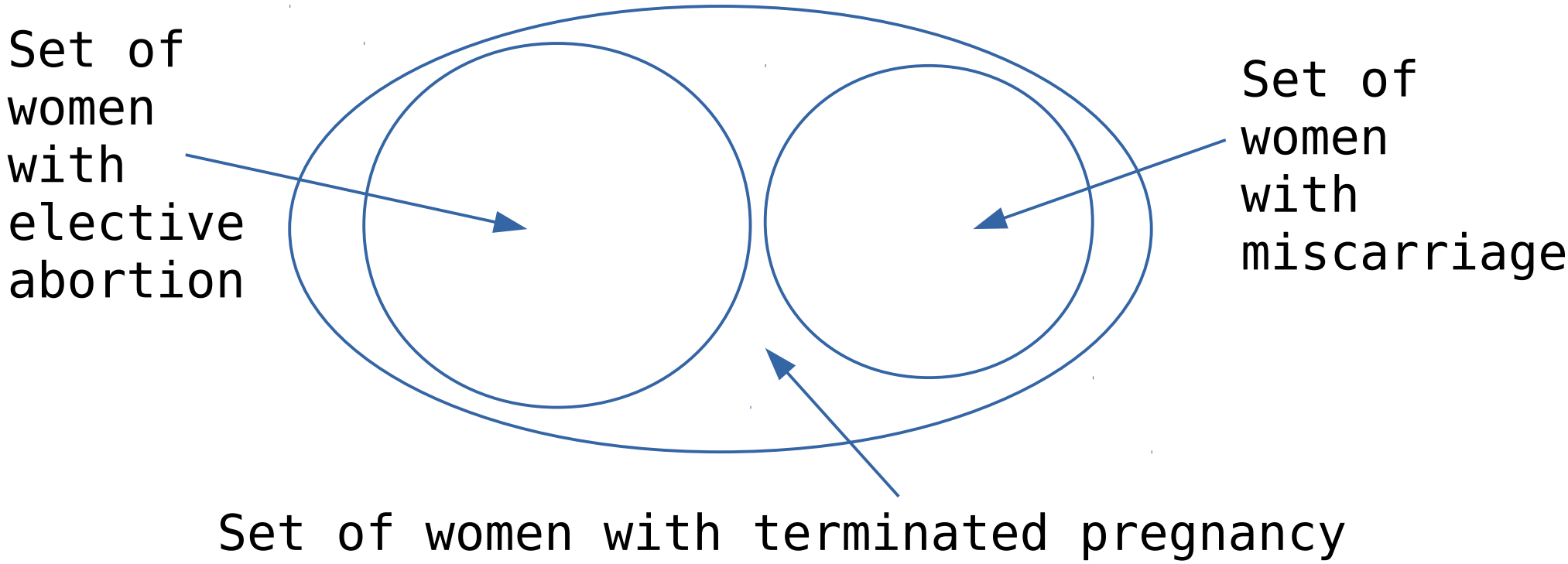
# What is Qualitative Comparative Analysis?

- Necessary condition: cause must be present for outcome to occur
- Example: Must be exposed to HIV to contract AIDS



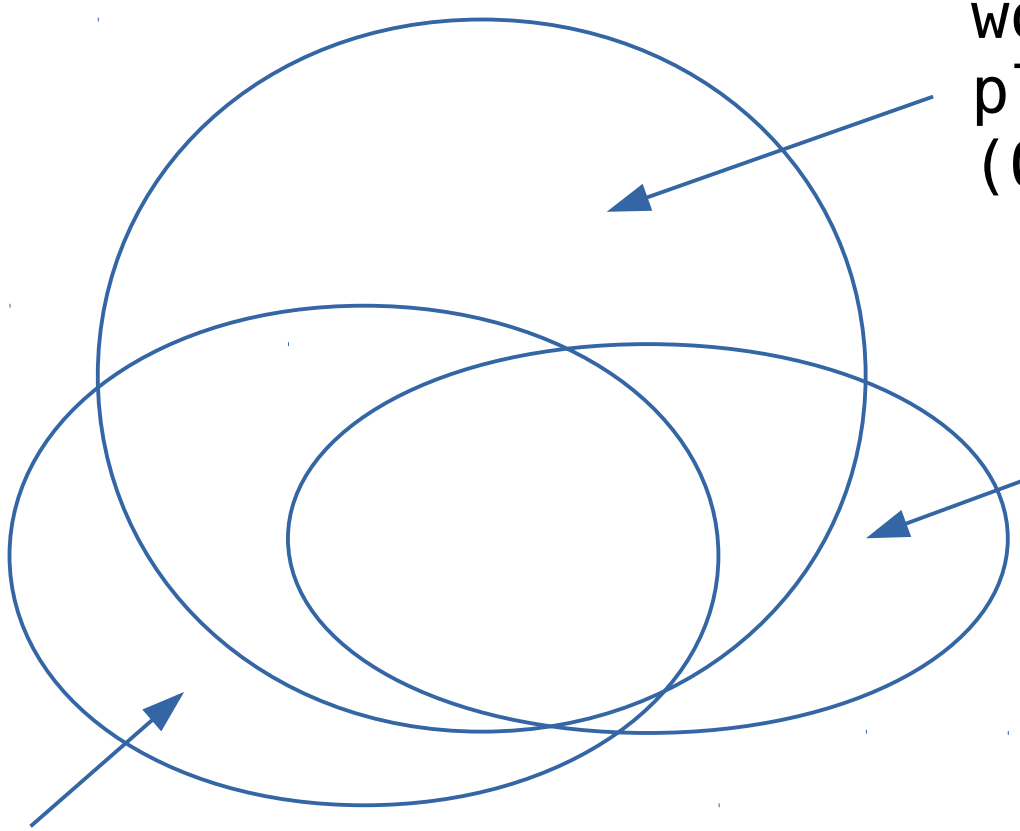
# What is Qualitative Comparative Analysis?

- Sufficient condition: if cause occurs, outcome will occur
- Example: Elective abortion *or* miscarriage will terminate pregnancy



# What is Qualitative Comparative Analysis?

- Sufficient condition: if cause occurs, outcome will occur



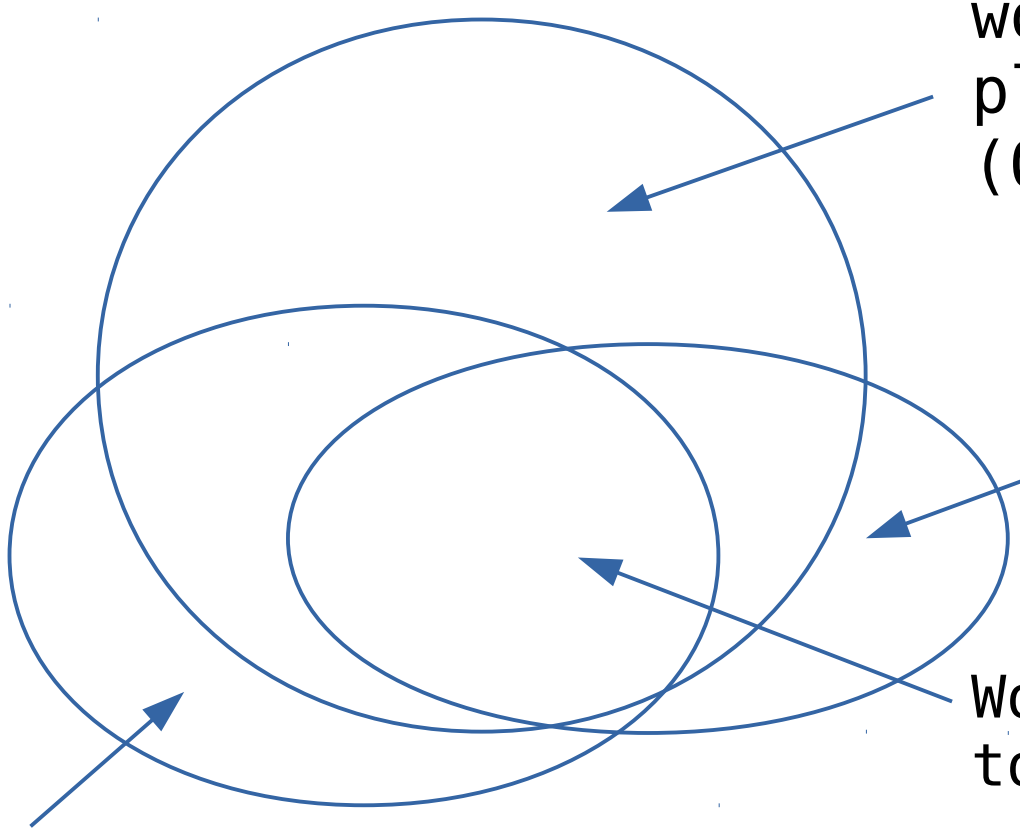
Recently deported women who do not plan to cross again (Outcome)

High SES women who haven't lived in the U.S. and aren't traveling with family ($X_1$)

High SES women who haven't lived in the U.S., have only attempted cross a few times and felt that their last crossing experience was very dangerous ($X_2$)

# What is Qualitative Comparative Analysis?

- Sufficient condition: if cause occurs, outcome will occur



Recently deported women who do not plan to cross again (Outcome)

High SES women who haven't lived in the U.S. and aren't traveling with family ($X_1$)

Women belonging to sets $X_1$ and $X_2$

High SES women who haven't lived in the U.S., have only attempted cross a few times and felt that their last crossing experience was very dangerous ($X_2$)

# What is Qualitative Comparative Analysis?

- Challenges conventional statistical analysis, which is based upon linear-additive model
- Complements other set-theoretic research methods (e.g., SNA and QNA)
- Does not depend upon degrees of freedom, so is useful for small-, medium-, and large-N studies
- Encourages a research process that is “retroductive” and “case-oriented”

# What is Qualitative Comparative Analysis?

## Example: Brown and Boswell (1995)

Truth Table with Contradiction (from Table 4 of Brown and Boswell 1995)

| Recent Black Migrants | Weak Union | Black Strikebreaking | Observations |
|---|---|---|---|
| T | T | T | East Chicago, Pittsburgh, Youngstown |
| T | F | Con | Buffalo, Chicago, Gary, Johnstown, [Cleveland] |
| F | T | F | Bethlehem, Joliet, McKeesport, Milwaukee, New Castle, Reading |
| F | F | F | Decatur, Wheeling |

# What is Qualitative Comparative Analysis?

## Example: Brown and Boswell (1995)

Revised Truth Table without Contradiction (from Table 5 of Brown and Boswell 1995)

| Recent Black Migration | Weak Union | Local Govt Repression | Black Strikebreaking | Observations |
|---|---|---|---|---|
| T | T | T | T | East Chicago, Pittsburgh, Youngstown |
| T | T | F | — | |
| T | F | T | T | Buffalo, Chicago, Gary, Johnstown |
| T | F | F | F | Cleveland |
| F | T | T | F | Bethlehem, Joliet, McKeesport, New Castle, Reading |
| F | T | F | F | Milwaukee |
| F | F | T | F | Decatur |
| F | F | F | F | Wheeling |

# What is Qualitative Comparative Analysis?

## Example: Brown and Boswell (1995)

Revised Truth Table without Contradiction (from Table 5 of Brown and Boswell 1995)

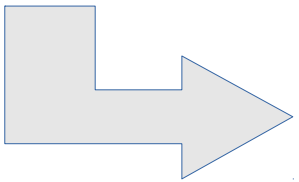
| Recent Black Migration | Weak Union | Local Govt Repression | Black Strikebreaking | Observations |
|---|---|---|---|---|
| T | T | T | T | East Chicago, Pittsburgh, Youngstown |
| T | T | F | — | |
| T | F | T | T | Buffalo, Chicago, Gary, Johnstown |
| T | F | F | F | Cleveland |
| F | T | T | F | Bethlehem, Joliet, McKeesport, New Castle, Reading |
| F | T | F | F | Milwaukee |
| F | F | T | F | Decatur |
| F | F | F | F | Wheeling |

# What is Qualitative Comparative Analysis?

## Example: Brown and Boswell (1995)

Revised Truth Table without Contradiction (from Table 5 of Brown and Boswell 1995)

| Recent Black Migration | Weak Union | Local Govt Repression | Black Strikebreaking | Observations |
|---|---|---|---|---|
| T | T | T | T | East Chicago, Pittsburgh, Youngstown |
| T | F | T | T | Buffalo, Chicago, Gary, Johnstown |



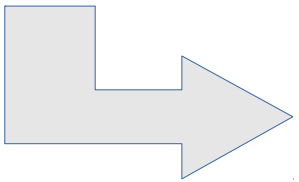
```
RBM *  WU * LGR +
RBM * ~WU * LGR
          = Black Strikebreaking
```

# What is Qualitative Comparative Analysis?

## Example: Brown and Boswell (1995)

Revised Truth Table without Contradiction (from Table 5 of Brown and Boswell 1995)

| Recent Black Migration | Weak Union | Local Govt Repression | Black Strikebreaking | Observations |
|---|---|---|---|---|
| T | T | T | T | East Chicago, Pittsburgh, Youngstown |
| T | F | T | T | Buffalo, Chicago, Gary, Johnstown |

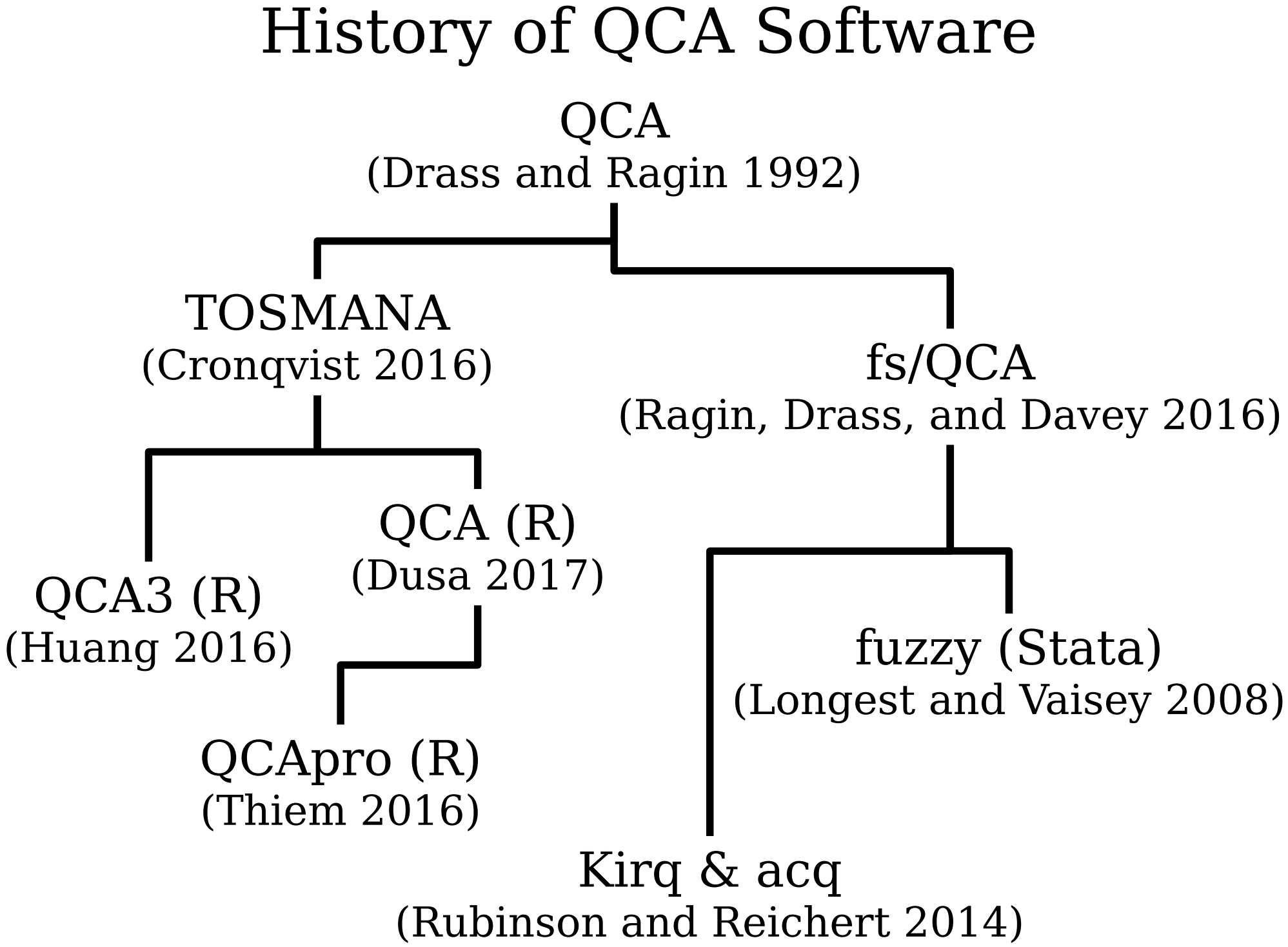


```
RBM *  WU * LGR +
RBM * ~WU * LGR
                = Black Strikebreaking

RBM * LGR       = Black Strikebreaking
```

# Technical and Usability Challenges

- QCA algorithms are:
  - NP-hard (no exact algebraic solution)
  - $O(2^N)$ complexity, where $N$ is number of variables (not observations) in the data set
  - but, because data sets tend to be small and matrix algebra isn't used, no need for high performance numerical computation library
- How to maintain and encourage retroductive, case-oriented research process?
- How to make software that's efficient, useful, and usable?


History of QCA Software
QCA
(Drass and Ragin 1992)
TOSMANA
(Cronqvist 2016)
fs/QCA
(Ragin, Drass, and Davey 2016)
QCA (R)
(Dusa 2017)
QCA3 (R)
(Huang 2016)
fuzzy (Stata)
(Longest and Vaisey 2008)
QCApro (R)
(Thiem 2016)
Kirq & acq
(Rubinson and Reichert 2014)

# Design Goals

- Software that is efficient, useful, and usable:
  - Follow the Unix philosophy
  - Crossplatform and user-friendly
  - Support and encourage good QCA research practices; facilitate close enagement with cases and retroduction
  - Support case-oriented and qualitative research
- Also important:
  - Good performance
  - Avoid sucking up all of my time
  - Fun!

# fsQCA module for R

"Plan to throw one away; you will, anyhow"

- Cross-platform, but requires R
- Not user-friendly
- Too slow
- R programming "considered harmful"
- But: allowed me to realize that the user interface should be task-oriented

# Second Implementation: Python

- acq: QCA at the Unix commandline
  - a “scratch my own itch” project
- Kirq: QCA for everybody else
  - a user-friendly, crossplatform GUI program

## Why Python?

- The surrounding ecosystem
  - Ability to hire others
  - Confidence that the supporting environment is stable and will continue to be maintained
  - Python is *lingua franca* in academia
  - Rich environment for GUI toolkits, installers, etc.
  - Chose Qt for GUI toolkit; PyInstaller for distribution

# Lessons Learned

- acq is fewer lines of code than previous R module, and faster
  - compare to QCA module for R
- Less concern for performance means more attention to functionality and user-interface issues
- Writing acq as Unix shell scripts helped me streamline the QCA analysis; acq and Kirq make it easy to modify and rerun analyses
- Easy to add new functionality to acq
- Have designed Kirq to facilitate interrogation and comparisons of solutions
- Lots of GUI niceties, such as tooltips and pop-out windows
- Session history is Kirq's killer feature
- Importance of “eating your own dogfood”

# Using Python for Academic Projects

- Advantages:
  - Core language is relatively compact, with excellent documentation
  - Relatively easy to find developers
  - Strong, well-developed environment of GUI toolkits, installers, etc.
  - Decent performance out of the box, with some ability to optimize

- Disadvantages:
  - Out-of-box performance often too slow; optimization often difficult
  - Bit rot of external libraries (Windows 10 is especially problematic)
  - Package distribution is a mess, as is associated documentation
  - Standard library not compact; churn is too rapid to keep up with for part-time developer
  - Introductory and intermediate dead-tree documentation is lousy
  - Online signal-to-noise ratio is low
  - Community too insular; overly concerned with “idiomatic Python”

# Violating Spolsky's Rule #1: "Never rewrite code from scratch"

- Issues with Python:
  - Performance
  - Maintenance and distribution
  - Rapidly expanding ecosystem; falling signal-to-noise ratio
  - Not fun!
- Why OCaml? (and why not Haskell?)
  - Compact, stable language with clear syntax
  - Small, highly-curated standard library
  - Small community; very high signal-to-noise ratio
  - Practical multiparadigmatic language
  - http://roscidus.com/blog/blog/2014/06/06/python-to-ocaml-retrospective/
  - Fun!

# Initial Benefits of Functional Programming and OCaml

- Easier to reason about code
  - Higher level of abstraction
  - State changes eliminated/restricted to individual functions
  - Eliminates many mundane errors; reduces need for testing
  - Compact language
  - Small ecosystem: relatively few resources and documentation, but high quality
  - Tuareg mode and Merlin

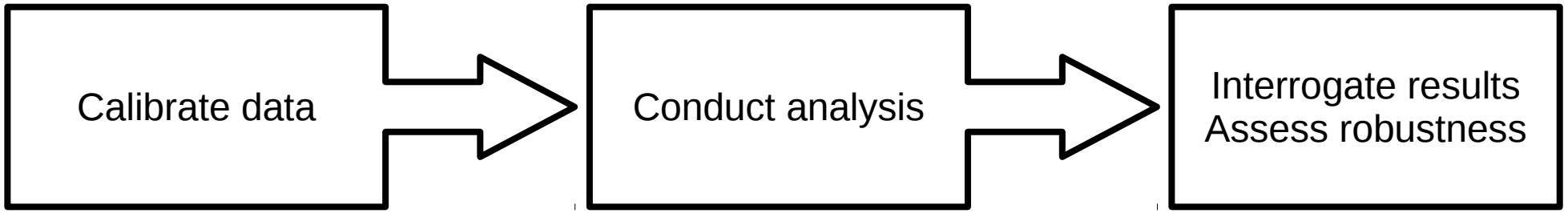
- OCaml's type system
  - Eliminates many classes of errors
  - Encourages explicit design; discourages programming-by-coincidence
  - Pattern matching forces one to think about and address entire domain

- Thinking in terms of types is revelatory
  - Clarifies dependencies within the program
  - Clarifies how different aspects of the program interact with one another
  - QCA algorithms are fundamentally recursive
  - Have solved a number of outstanding problems in QCA

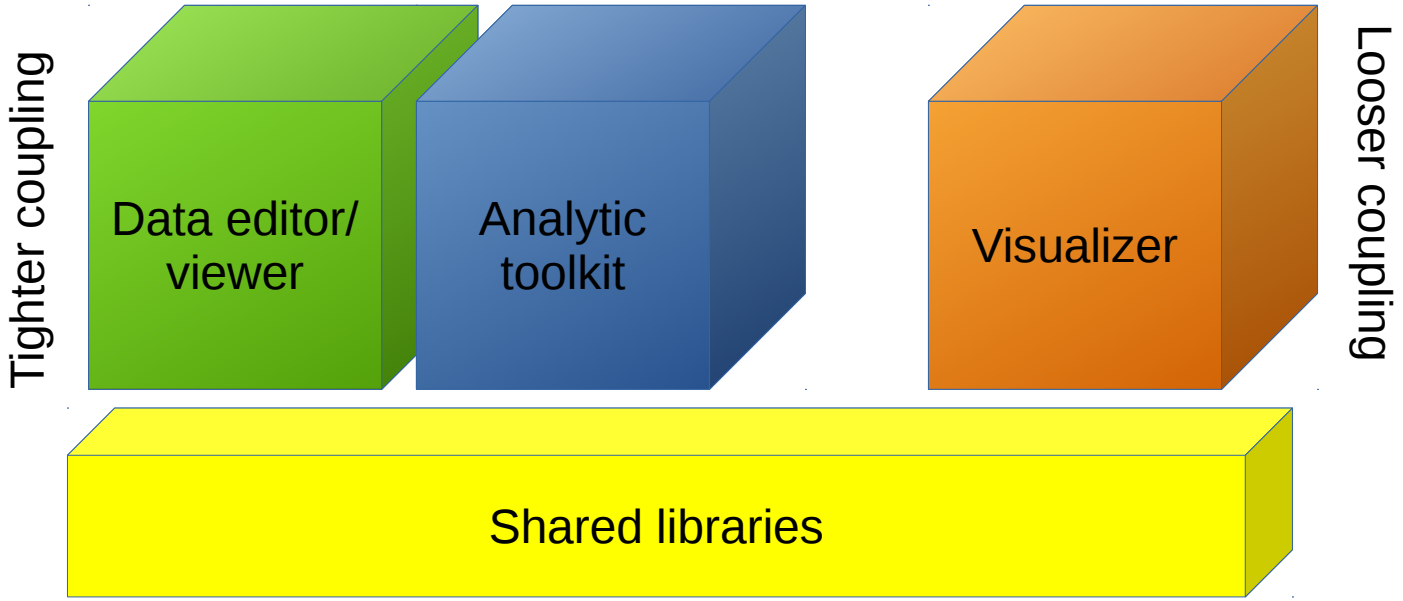
# Software Architecture

"The general tendency is to over-design the second system"

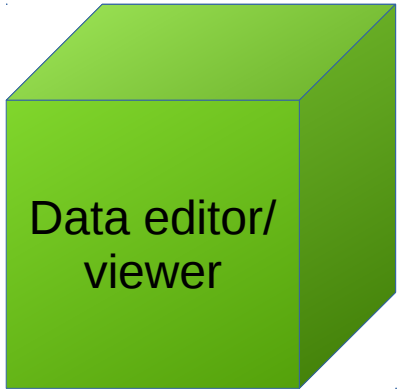
## *Basic QCA software workflow*




## *Three discrete-but-interconnected components*



# Software Architecture




GUI spreadsheet-like interface

- Import, create, edit, and calibrate data
  - "Be liberal in what you accept, and conservative in what you send."
  - Fuzzy set calculations
  - Doman-specific operations
- Select data for analysis
  - Selection of conditions
  - Subset data via Boolean expression
- Visualize data

# Software Architecture

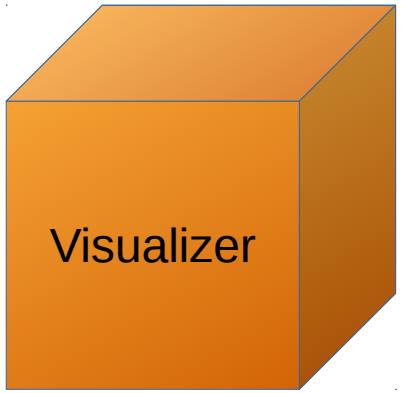

## GUI and CLI Interfaces

- Is essentially Kirq (GUI) and acq (CLI) integrated into a single application
  - Keeps GUI simple and user-friendly
  - Easier to add new features to CLI
- GUI provides core analytic functions, plus session history
- CLI interface provides advanced analytic functions, including results interrogation and robustness tests
- CLI interface also embedded as CUI into GUI to permit advanced analysis and provide access to new features

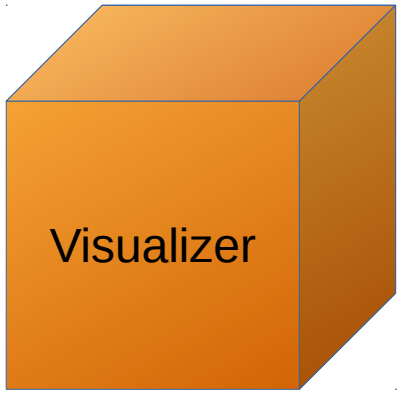
# Software Architecture

GUI and CLI, plus web-frontend

- Focus on small & medium-N data sets
- Standardize inputs; automatically convert between various QCA objects
- Invoke various backends (e.g., TikZ or GnuPlot) as needed; invisible to user
- Plug-in architecture
  - Relatively easy to add/update visualizations
  - Register available visualizations on startup
  - Specify input, output, and required parameters
  - Generate backend code



# Software Architecture




## QCAViz Workflow

*Input*
Serialized representations of QCA object(s) (Calibrated data, truth table, and/or consistency/coverage solutions)

↓

*Pre-processing*
Validation, convert to appropriate type, collect user-specified parameters, etc

↓

*Generate backend code*
Hand off to plug-in script for code generation (GnuPlot, GraphViz, TikZ, etc)

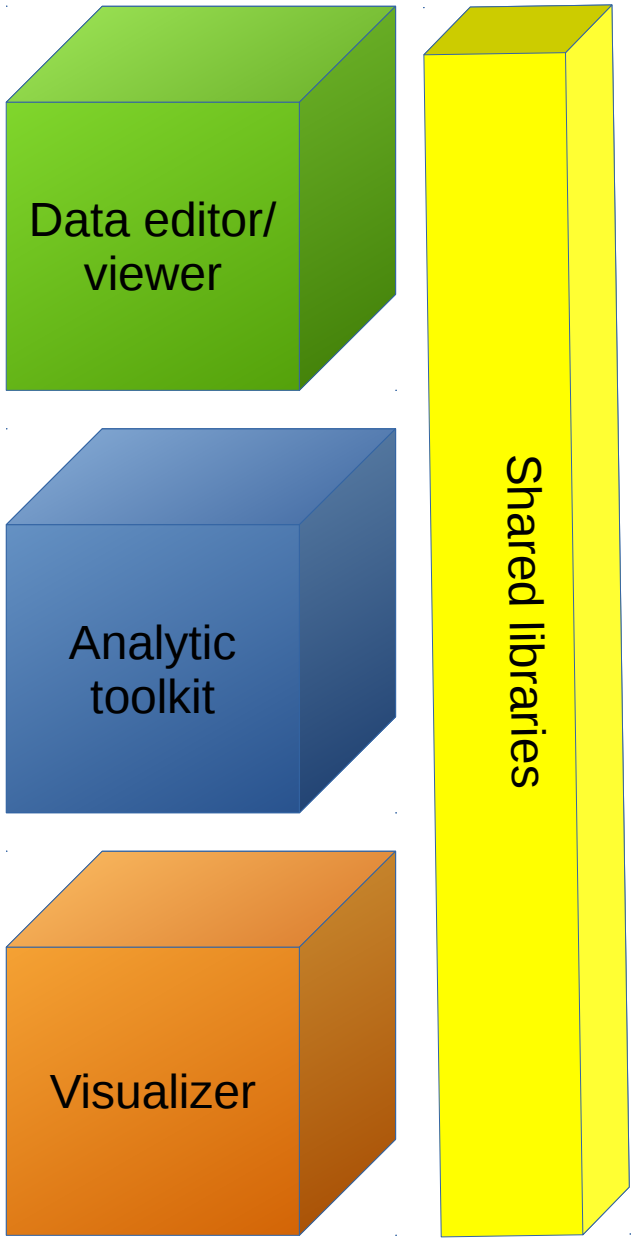
↓

*Post-processing*

↓

*Output:*
- Render image, or
- Convert and save to SVG, EPS, etc, or
- Output raw code for producing image

# Unresolved Issues




- Choice of GUI toolkit
- Communication among data editor, analytic toolkit, and visualizer
- For analytic toolkit, interface (if any) between GUI and CUI
- Do I really need to build a data editor?
- Should the editor be developed as a standalone application?
- Is this a second system?