Advanced Data Management (CSCI 680/490)

Provenance

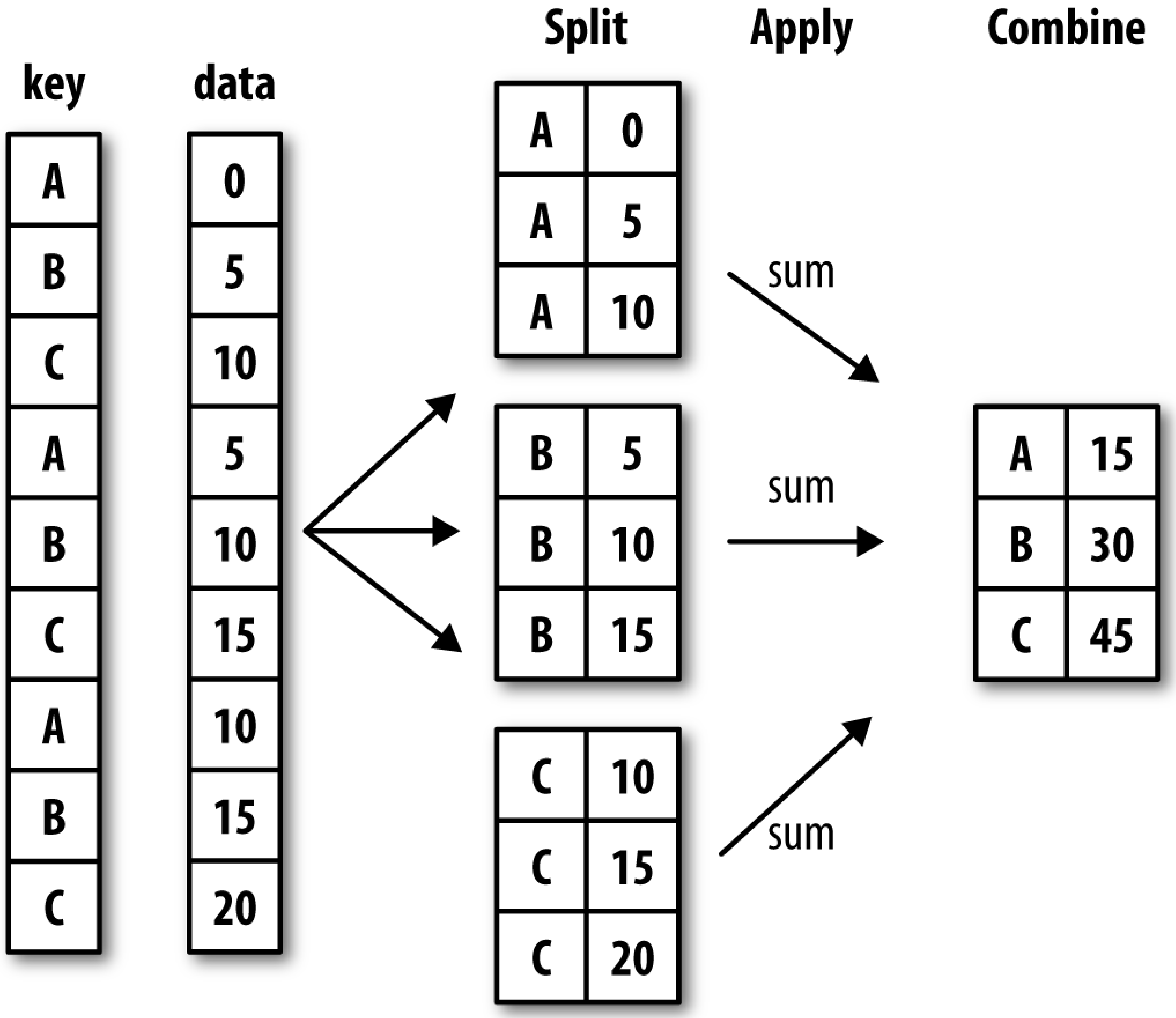
Dr. David Koop

Split-Apply-Combine

- Coined by H. Wickham, 2011
- Similar to Map (split+apply) Reduce (combine) paradigm
- The Pattern:
  1. **Split** the data by some grouping variable
  2. **Apply** some function to each group independently
  3. **Combine** the data into some output dataset
- The apply step is usually one of :
  - Aggregate
  - Transform
  - Filter

[T. Brandt]

Split-Apply-Combine



[W. McKinney, Python for Data Analysis]

Split-Apply-Combine

- `df.groupby('Island')[['Culmen Length (mm)',`
  `'Culmen Depth (mm)']].mean()`
- `df.groupby('Island').agg({'Culmen Length (mm)': 'mean',`
  `'Culmen Depth (mm)': 'mean'})`
-
```
df.groupby('Island').agg(
    cul_length=('Culmen Length (mm)', 'mean'),
    cul_depth=('Culmen Depth (mm)', 'mean'))
```

	cul_length	cul_depth
Island		
Biscoe	45.257485	15.874850
Dream	44.167742	18.344355
Torgersen	38.950980	18.429412

Transform Example

```
In [76]: df
Out[76]:
   key  value
0    a    0.0
1    b    1.0
2    c    2.0
3    a    3.0
4    b    4.0
5    c    5.0
6    a    6.0
7    b    7.0
8    c    8.0
9    a    9.0
10   b   10.0
11   c   11.0
```

```
In [77]: g = df.groupby('key').value

In [78]: g.mean()
Out[78]:
key
a    4.5
b    5.5
c    6.5
Name: value, dtype: float64

In [79]: g.transform(lambda x: x.mean())
Out[79]:
0     4.5
1     5.5
2     6.5
3     4.5
4     5.5
5     6.5
6     4.5
7     5.5
8     6.5
9     4.5
10    5.5
11    6.5
Name: value, dtype: float64
```

[W. McKinney, Python for Data Analysis]

Transform Example

```
In [76]: df
Out[76]:
   key  value
0    a    0.0
1    b    1.0
2    c    2.0
3    a    3.0
4    b    4.0
5    c    5.0
6    a    6.0
7    b    7.0
8    c    8.0
9    a    9.0
10   b   10.0
11   c   11.0
```

```
In [77]: g = df.groupby('key').value

In [78]: g.mean()
Out[78]:
key
a    4.5
b    5.5
c    6.5
Name: value, dtype: float64

In [79]: g.transform(lambda x: x.mean())
Out[79]:
0     4.5
1     5.5
2     6.5
3     4.5
4     5.5
5     6.5
6     4.5
7     5.5
8     6.5
9     4.5
10    5.5
11    6.5
Name: value, dtype: float64
```

or `g.transform('mean')`

[W. McKinney, Python for Data Analysis]

Crosstabs and Pivot Tables

- `pd.crosstab([tips.time, tips.day], tips.smoker, margins=True)`

	smoker	No	Yes	All
time	**day**			
Dinner	**Fri**	3	9	12
	Sat	45	42	87
	Sun	57	19	76
	Thur	1	0	1
Lunch	**Fri**	1	6	7
	Thur	44	17	61
All		151	93	244

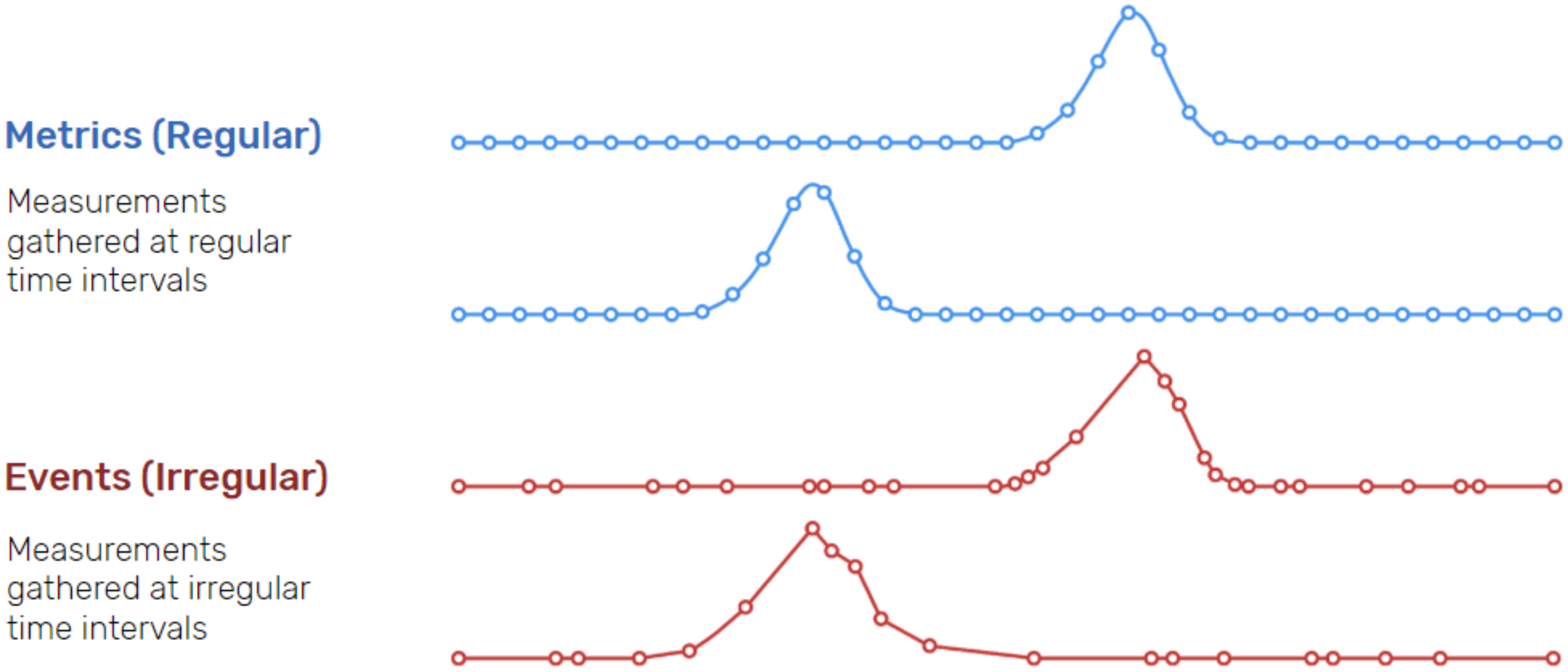
- or… `tips.pivot_table('total_bill',index=['time', 'day'], columns=['smoker'], aggfunc='count', margins=True, fill_value=0)`

What is time series data?

- Technically, it's normal tabular data with a timestamp attached
- But… we have observations of the same values over time, usually **in order**
- This allows more analysis
- Example: Web site database that tracks the last time a user logged in
  - 1: Keep an attribute `lastLogin` that is **overwritten** every time user logs in
  - 2: **Add a new row** with login information every time the user logs in
  - Option 2 takes more storage, but we can also do a lot more analysis!

Time Series Data

- Metrics: measurements at regular intervals
- Events: measurements that are not gathered at regular intervals

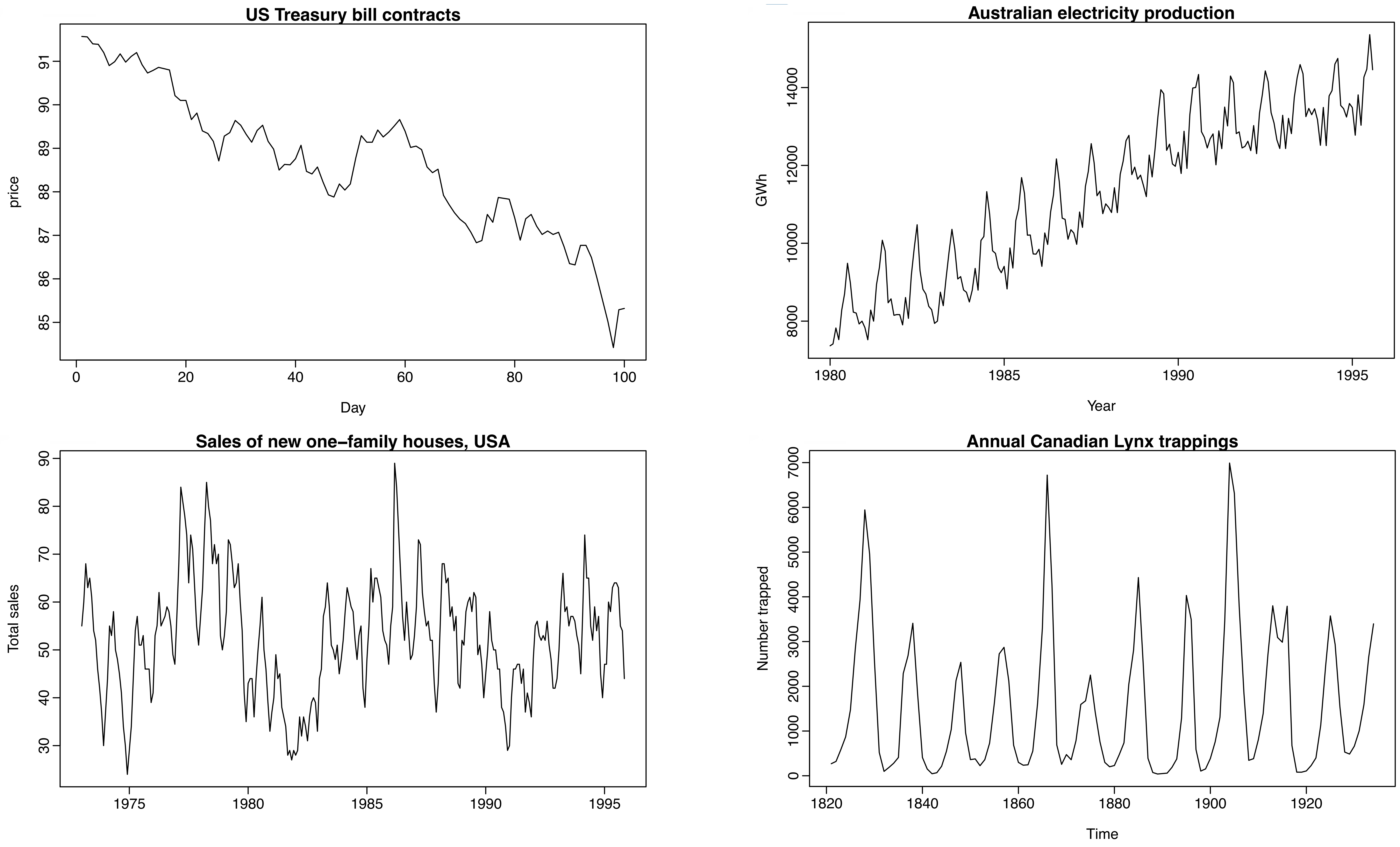


[InfluxDB]

Time Series Databases

- Most time series data is heavy **inserts**, few updates
- Also analysis tends to be on ordered data with trends, prediction, etc.
- Can also consider **stream** processing
- Focus on time series allows databases to specialize
- Examples:
  - InfluxDB (noSQL)
  - TimescaleDB (SQL-based)

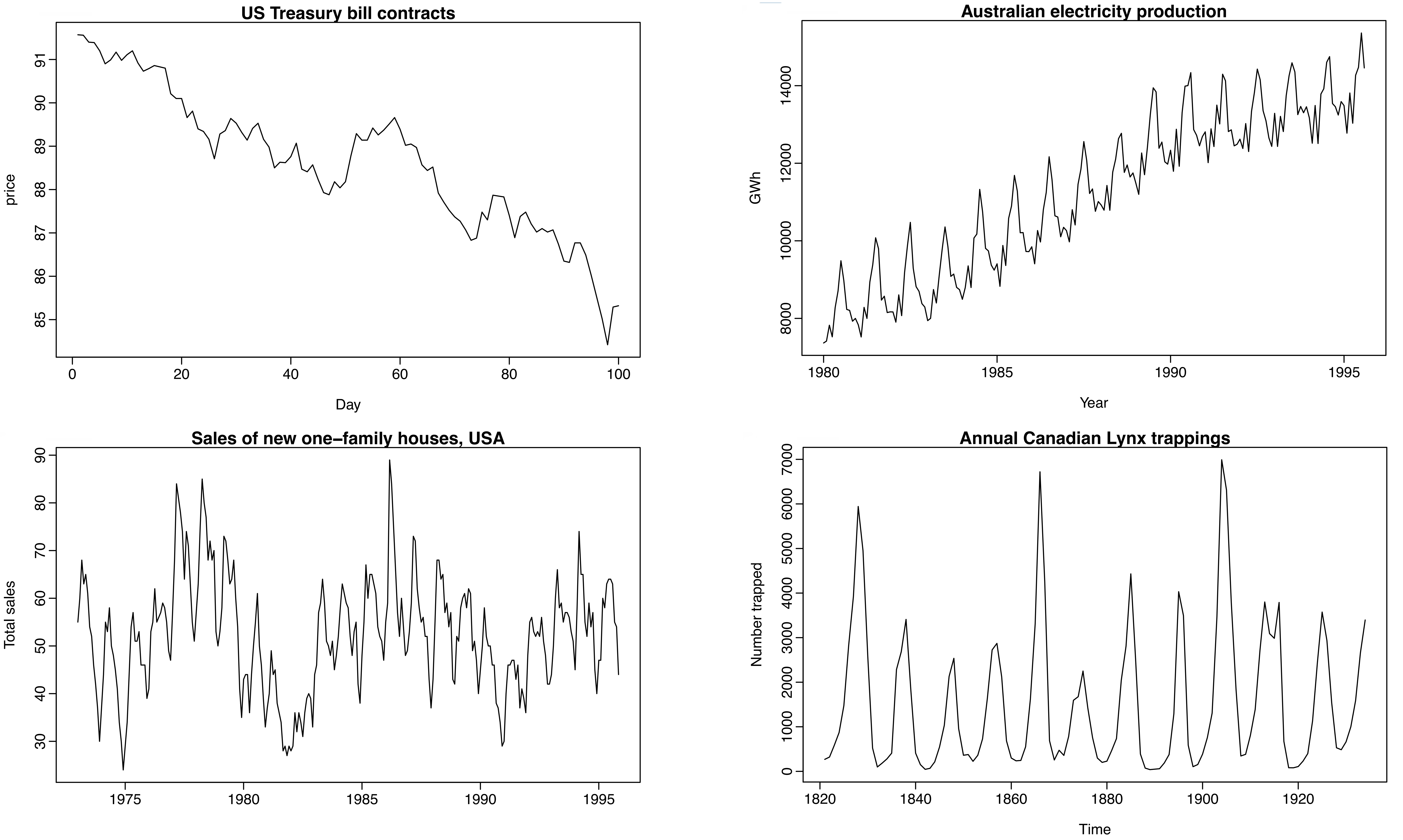
Time Series Patterns



[R. J. Hyndman]

Time Series Patterns

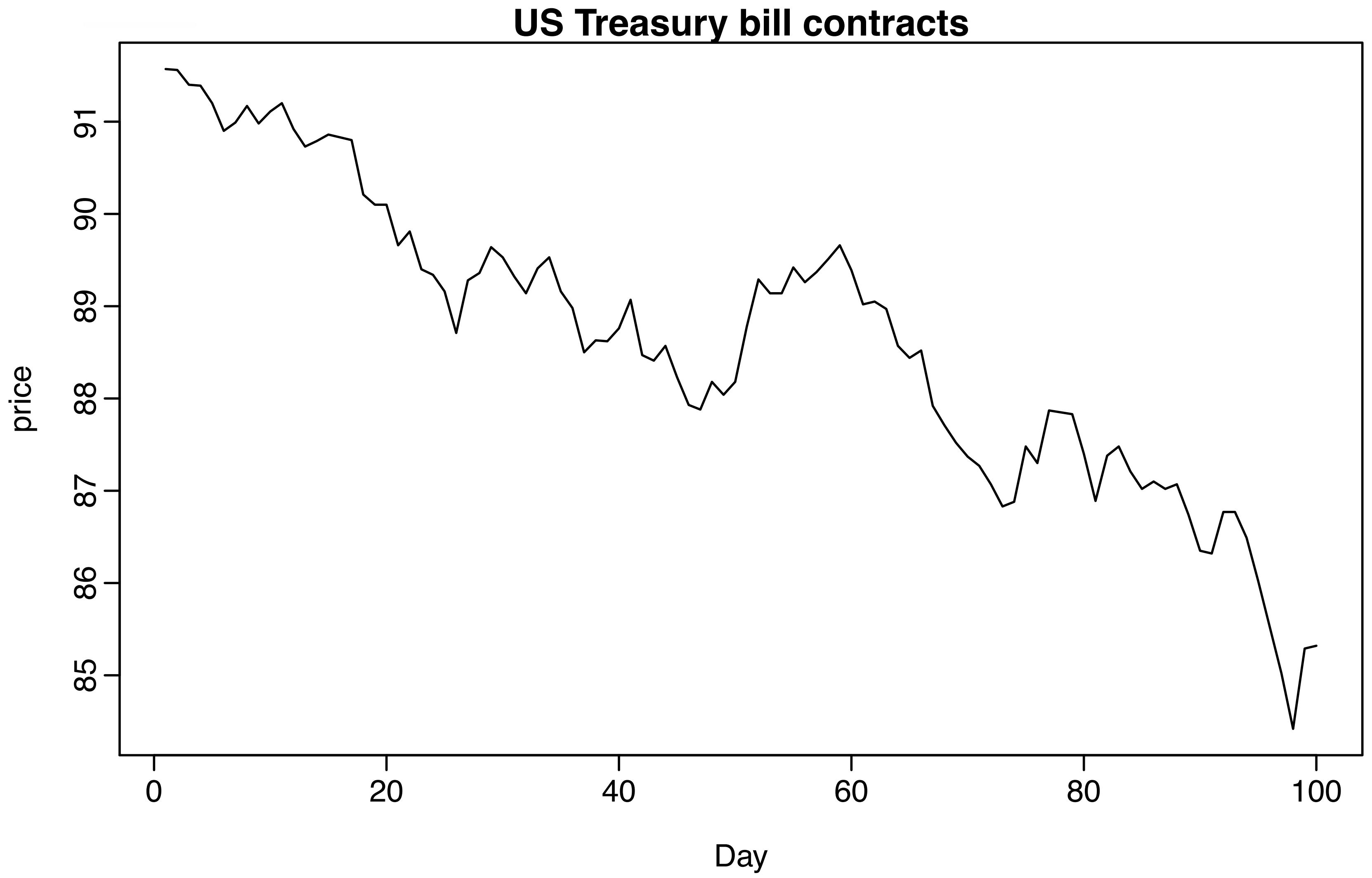
Trend

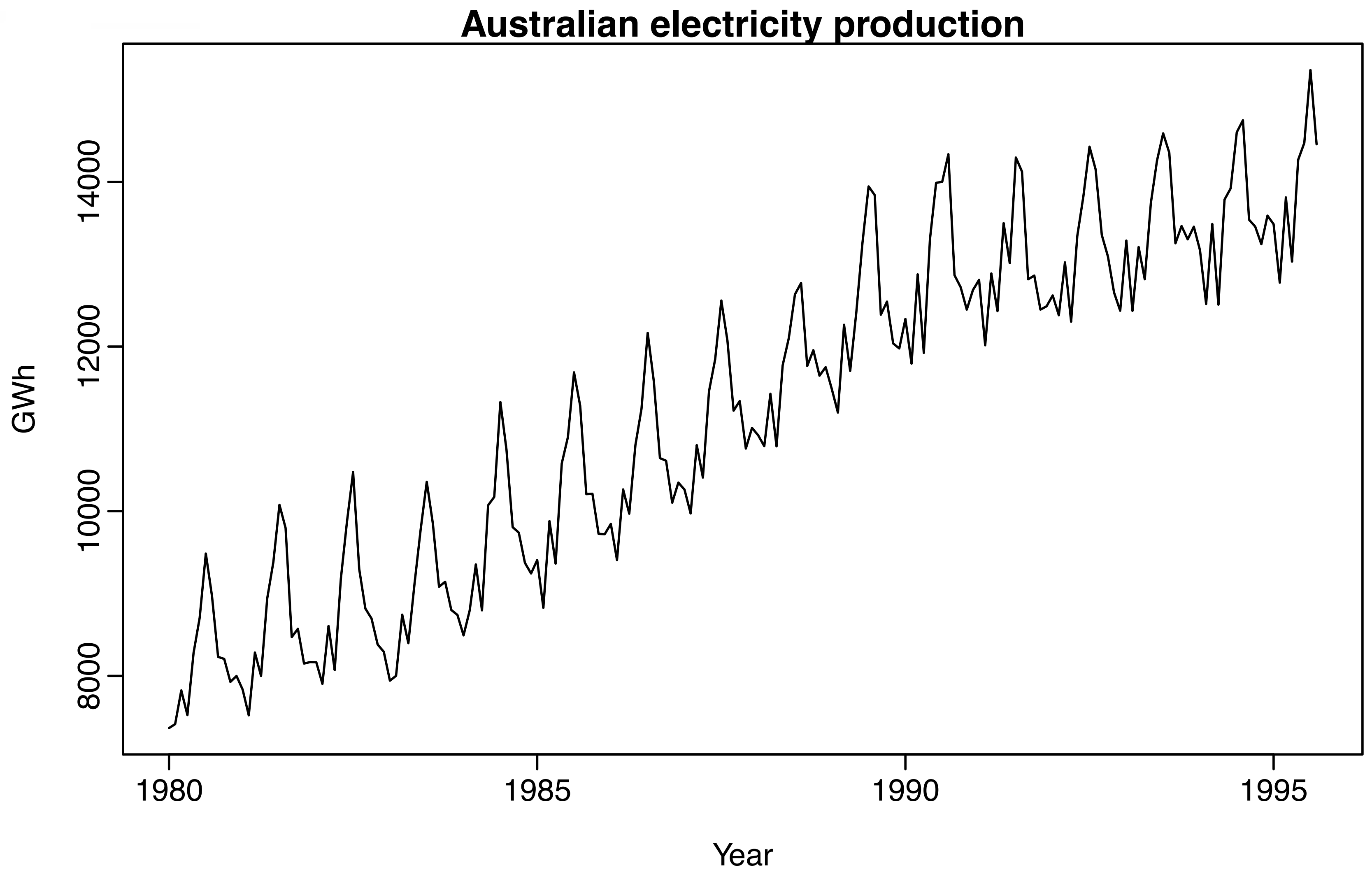


[R. J. Hyndman]

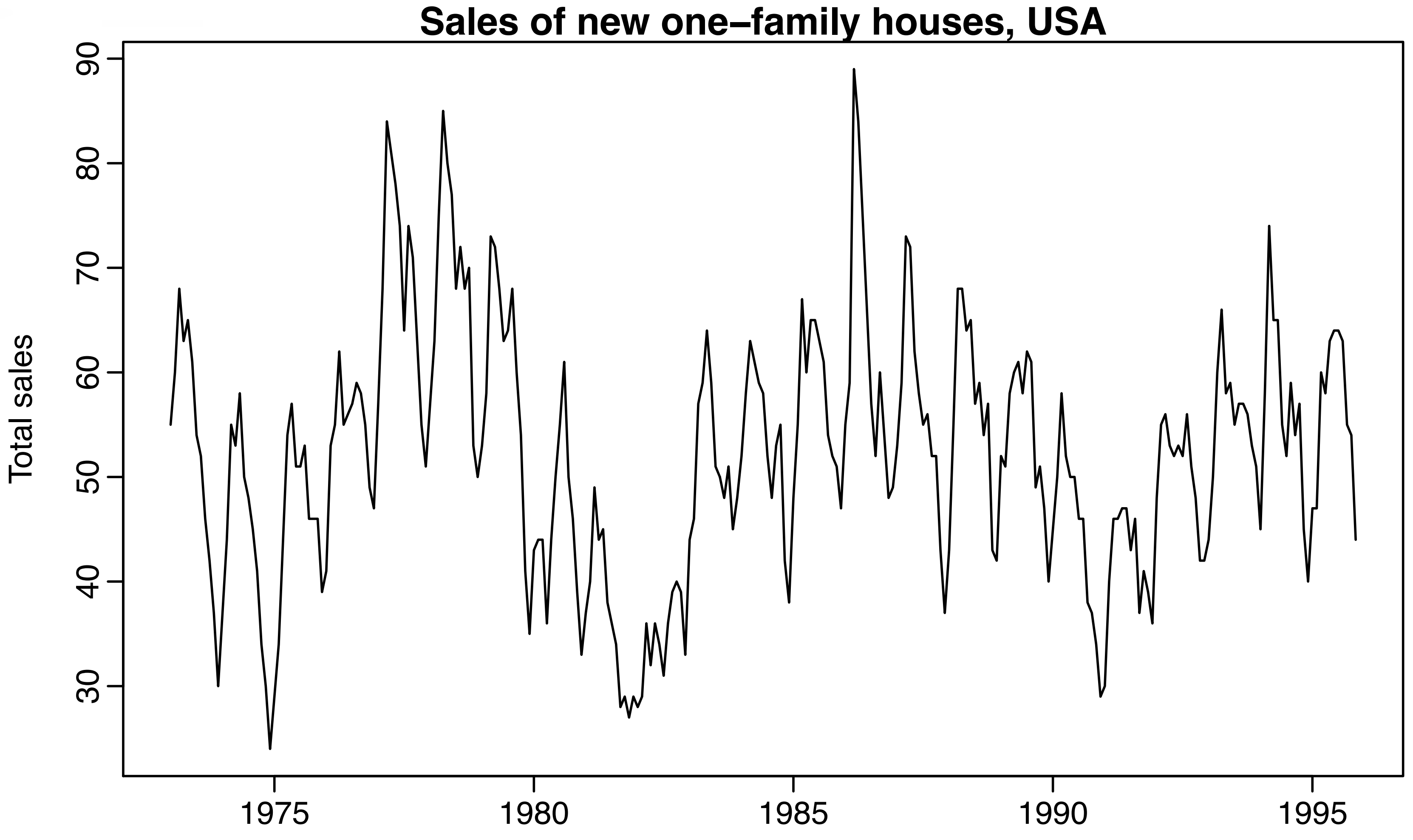
Time Series Patterns

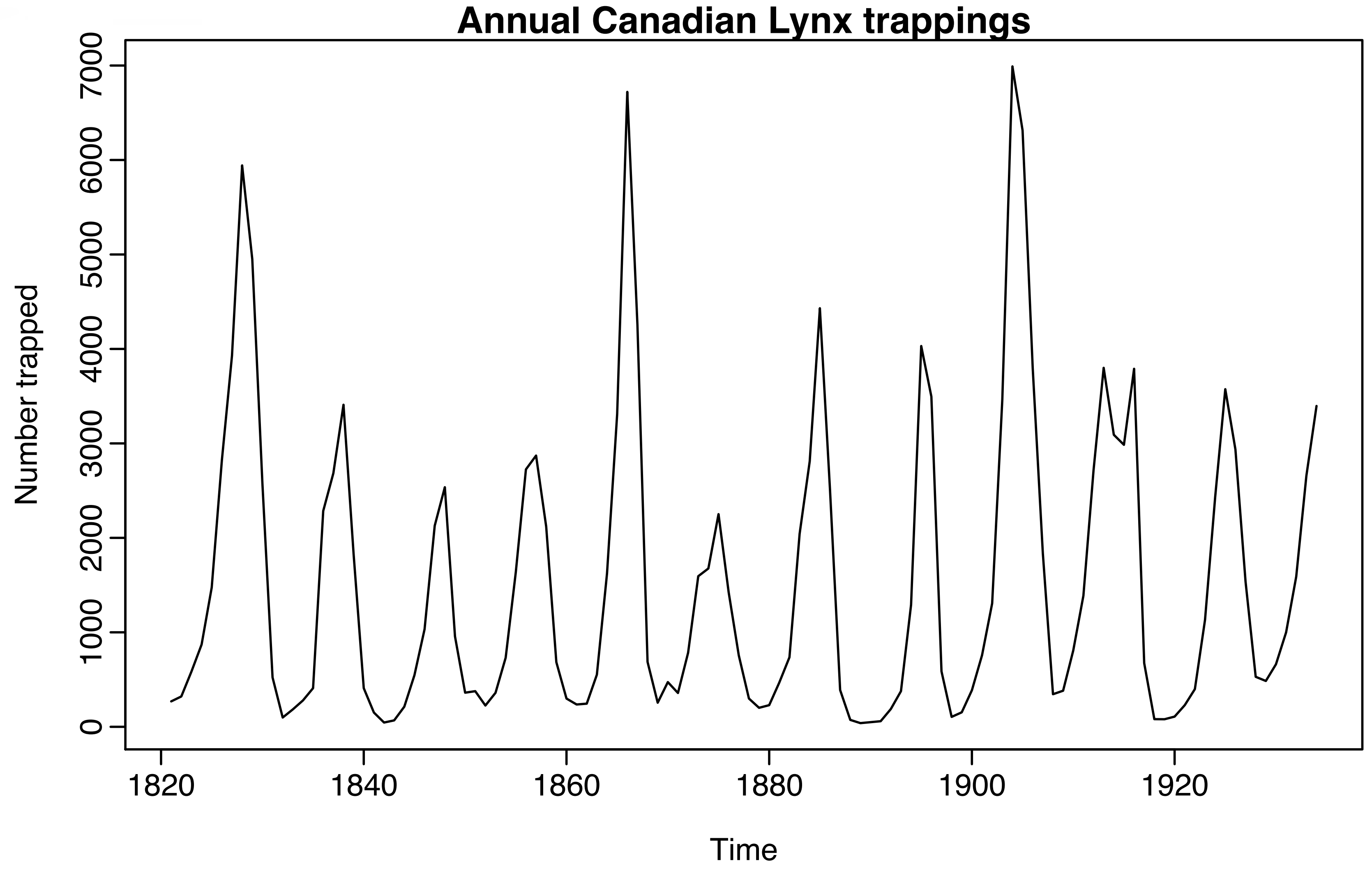
Trend





Trend + Seasonality

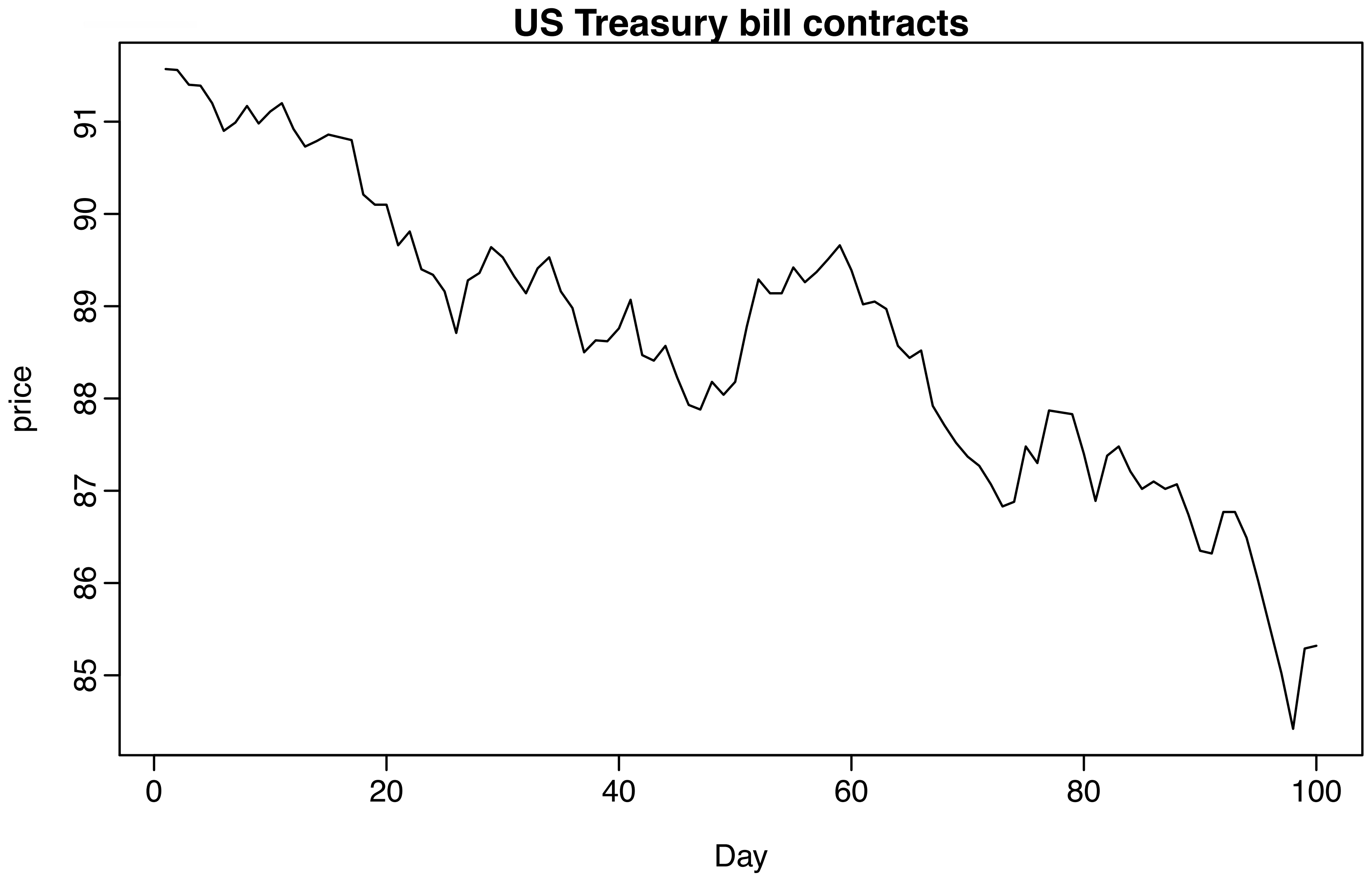




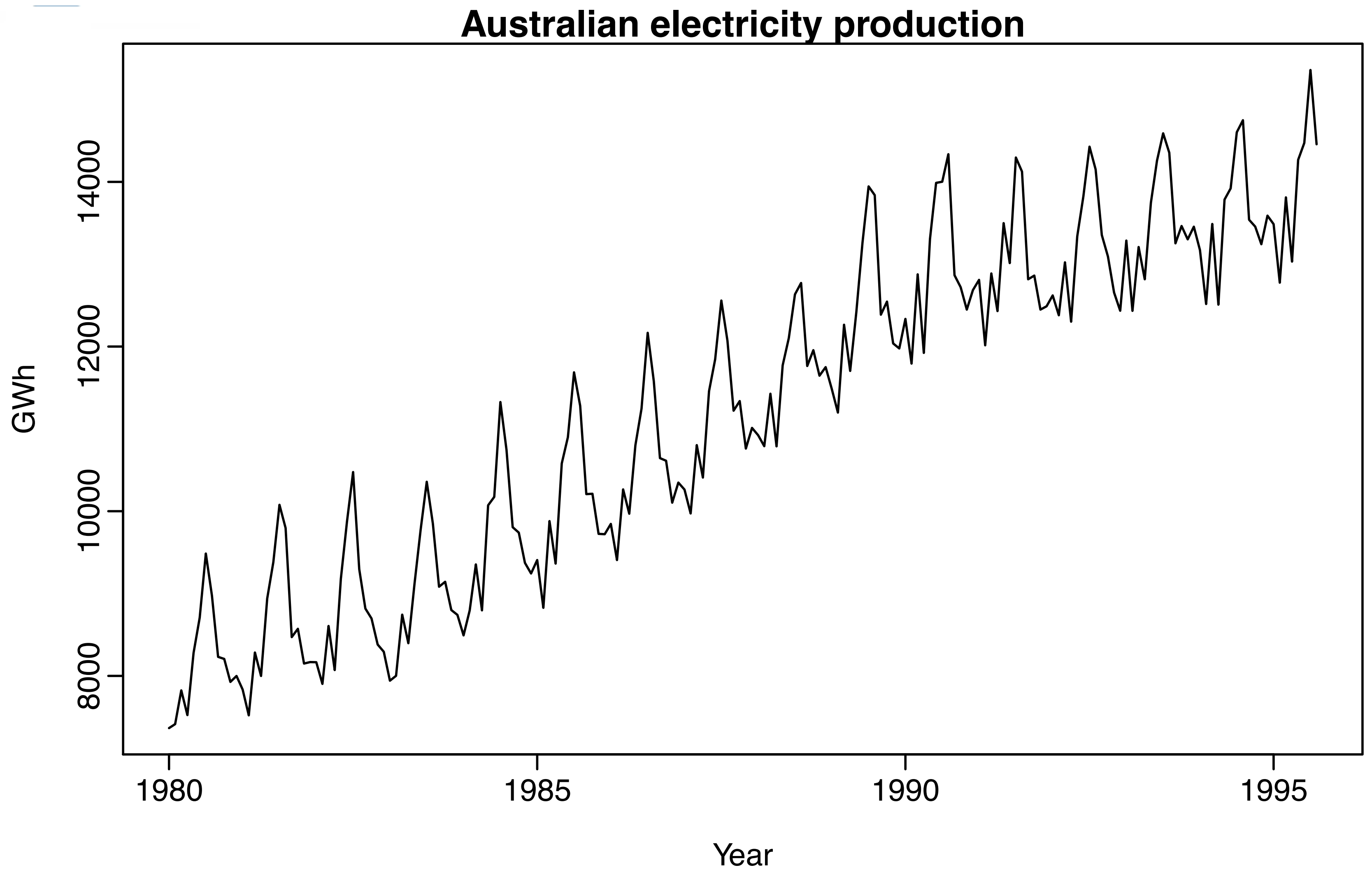
[R. J. Hyndman]

Time Series Patterns

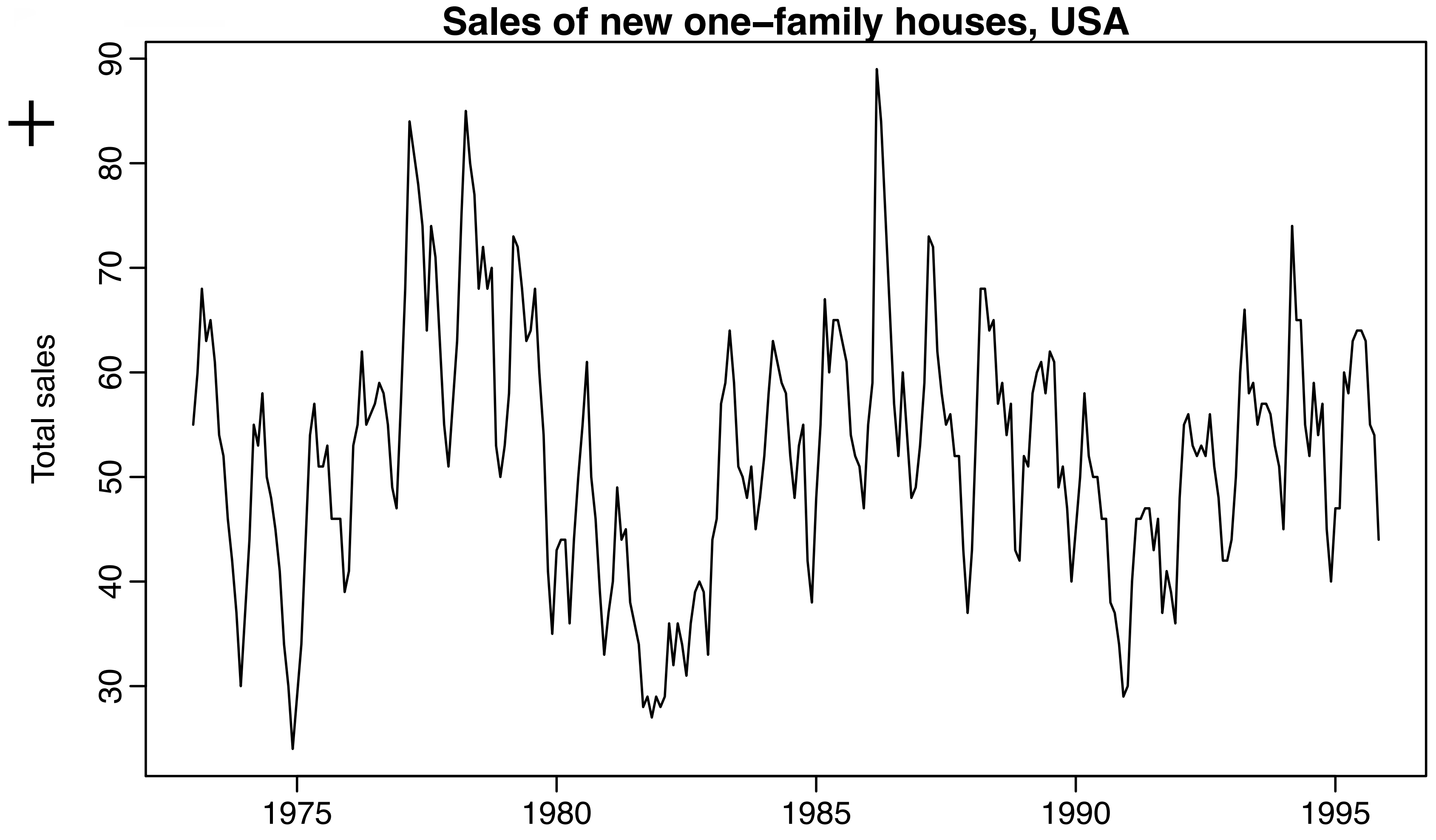
Trend

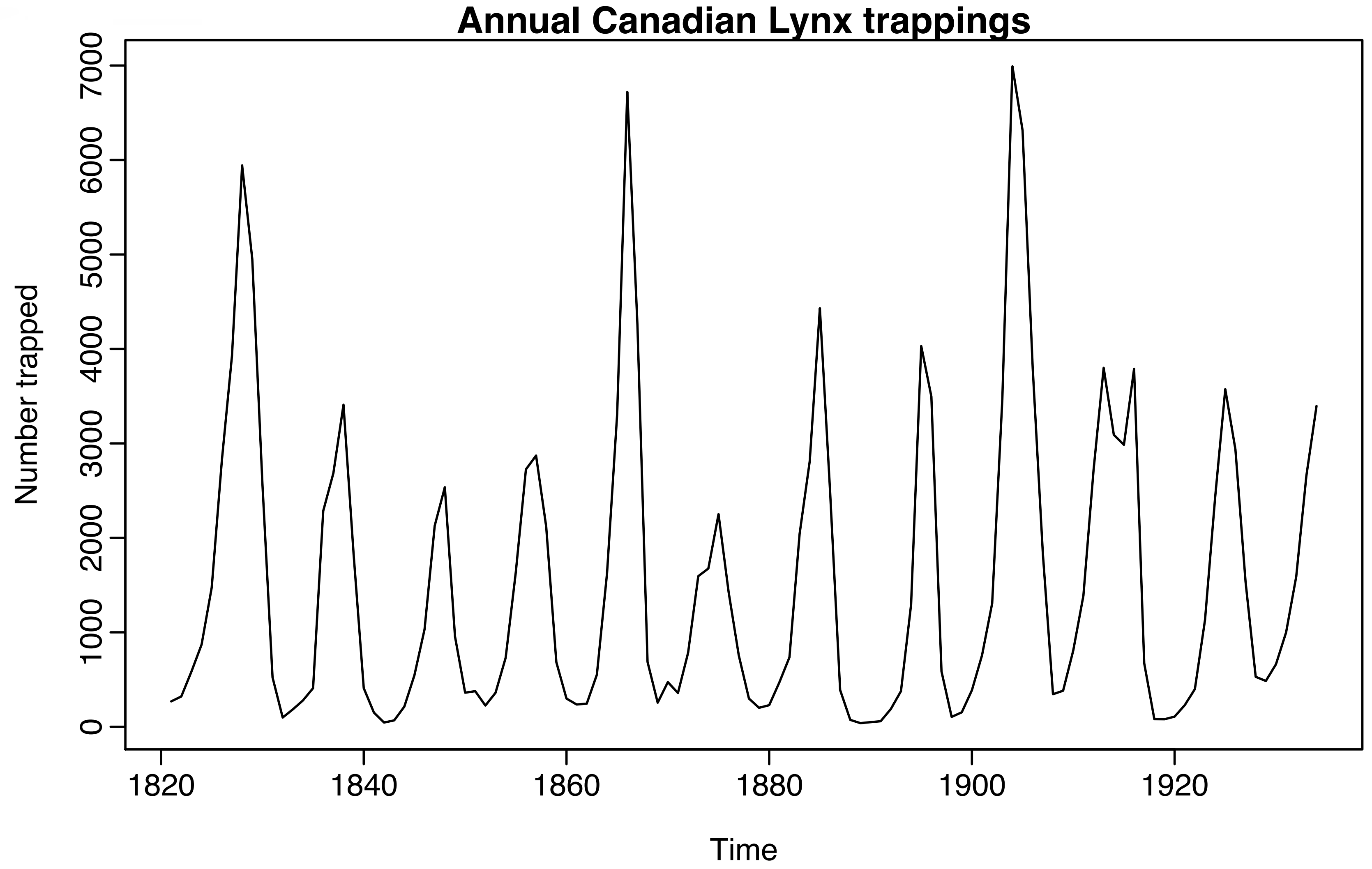


Trend + Seasonality



Seasonality + Cyclic

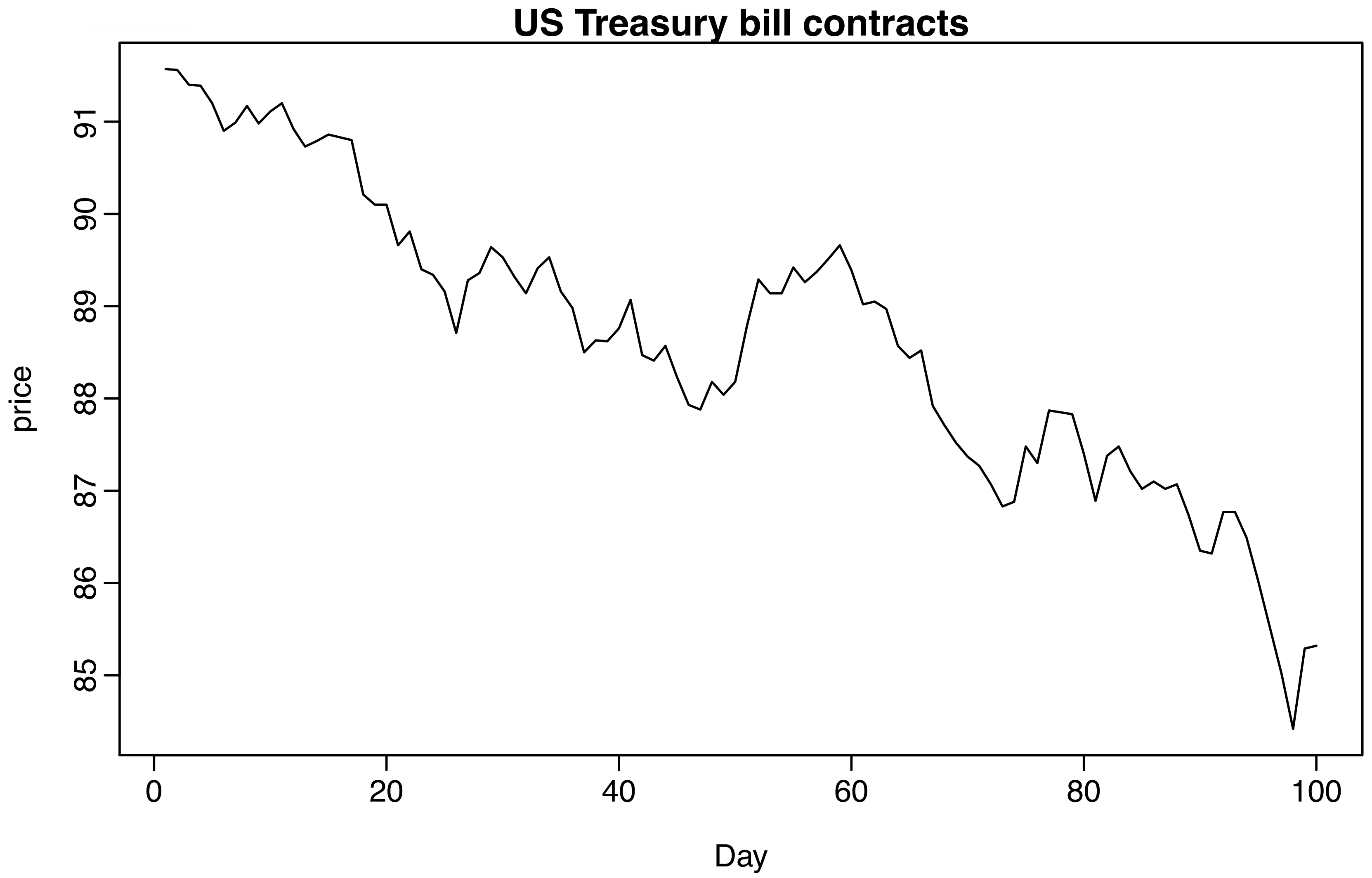


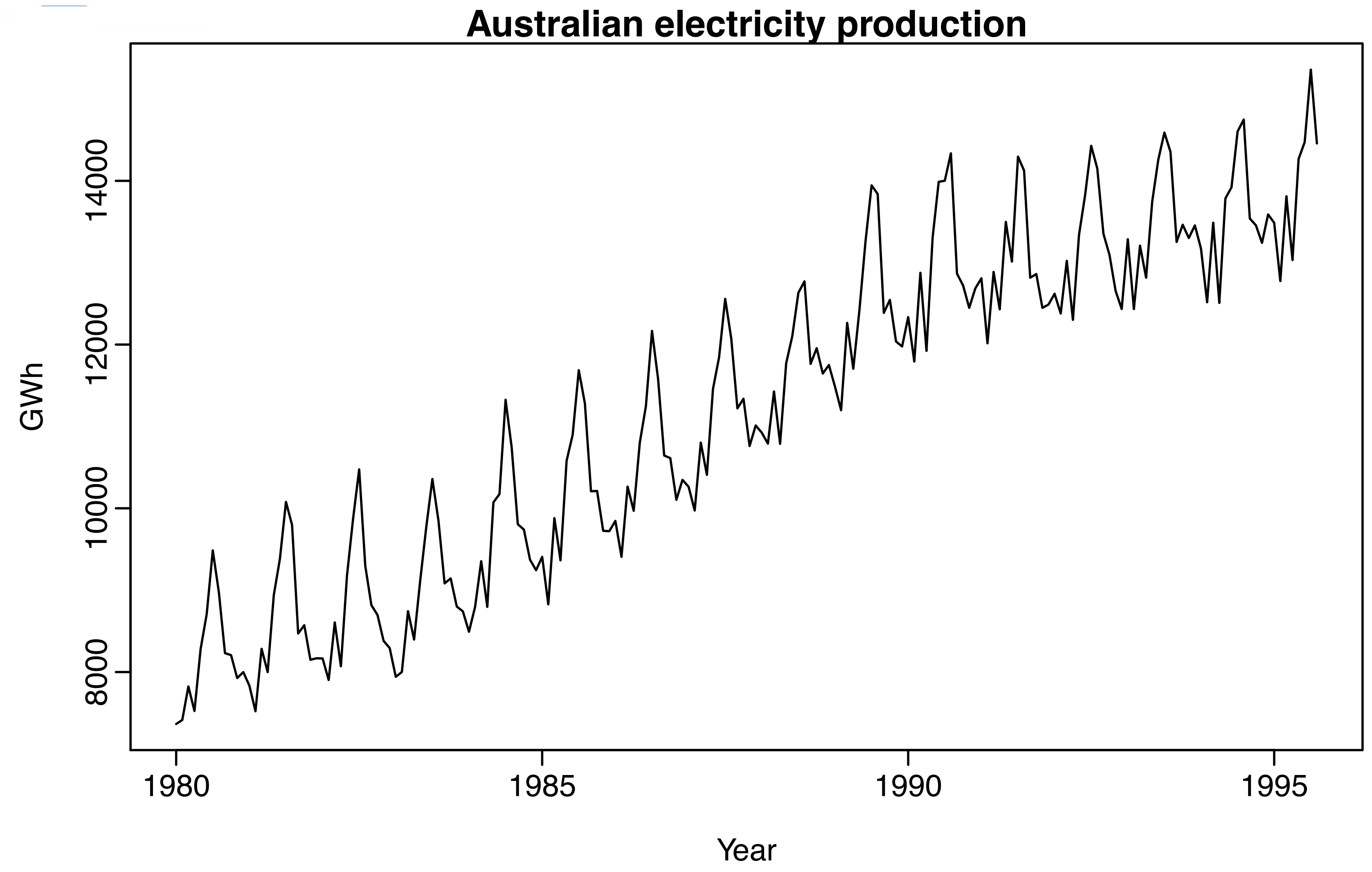


[R. J. Hyndman]

Time Series Patterns

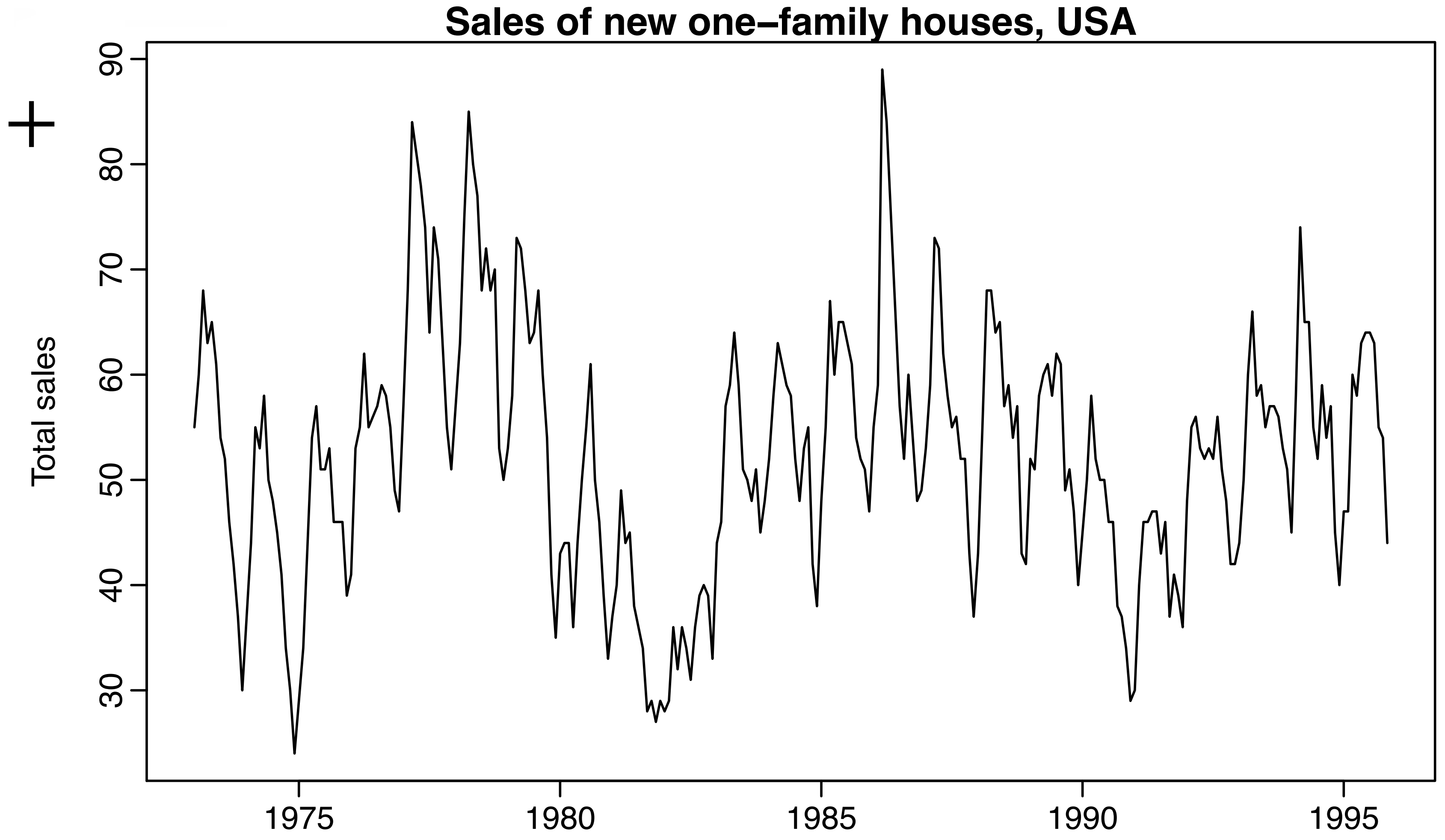
Trend

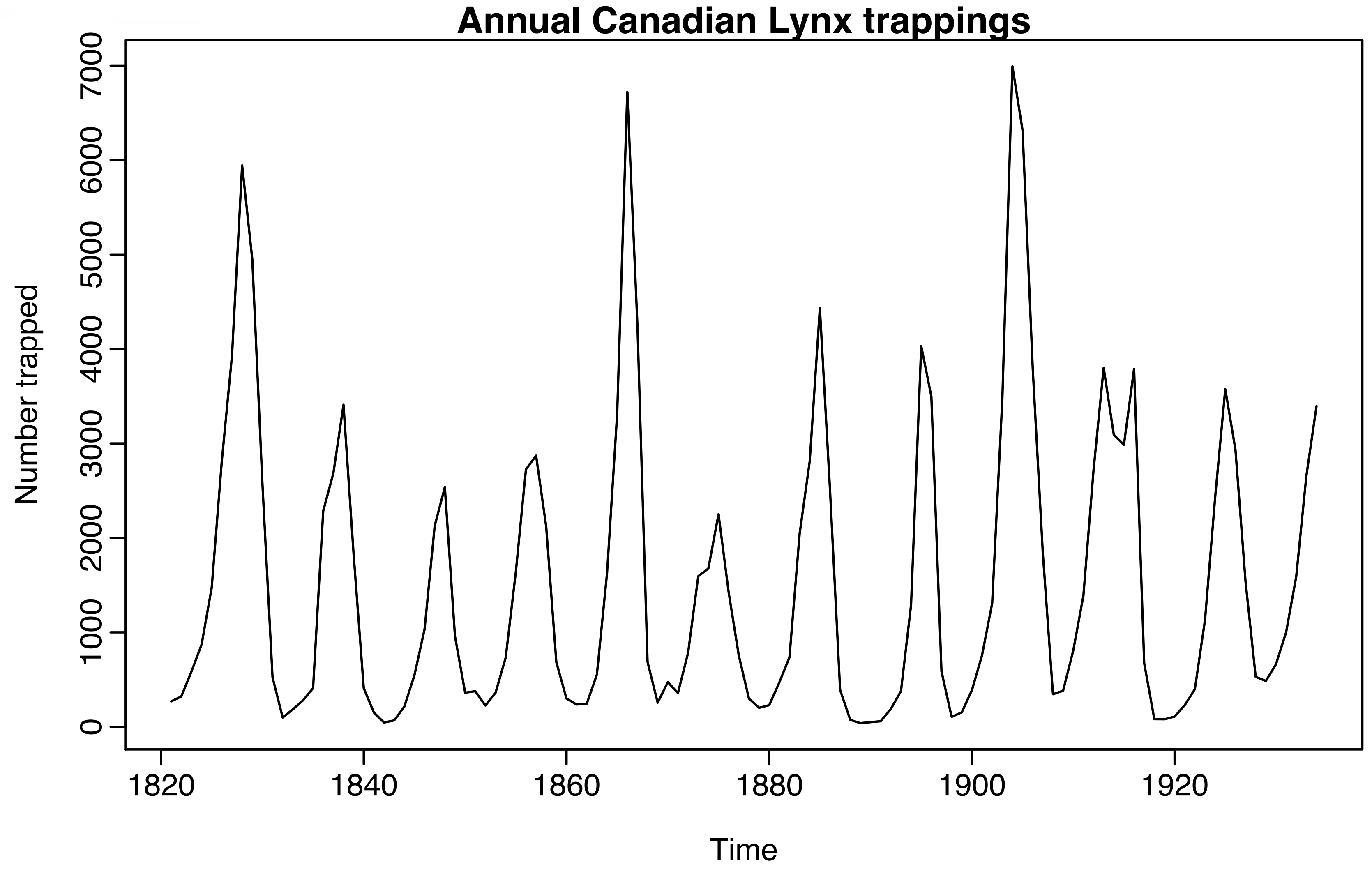




Trend + Seasonality

Seasonality + Cyclic





Stationary

[R. J. Hyndman]

Pandas Support for Datetime

- `pd.to_datetime`:
  - convenience method
  - can convert an entire column to datetime
- Has a `NaT` to indicate a missing time value
- Stores in a `numpy.datetime64` format
- `pd.Timestamp`: a wrapper for the `datetime64` objects

Assignment 5

- Chicago Bike Sharing Data
    - Spatial Analysis
    - Temporal Analysis
    - Graph Database (neo4j)

Reading Critique

- Read VisTrails and Reproducibility paper
- Write critique as before
- Due Monday before class

Time Zones

- Why?
- Coordinated Universal Time (UTC) is the standard time (basically equivalent to Greenwich Mean Time (GMT)
- Other time zones are UTC +/- a number in [1,12]
- DeKalb is UTC-6 (aka US/Central); Daylight Saving Time is UTC-5

Python, Pandas, and Time Zones

- Time series in pandas are **time zone native**
- The pytz module keeps track of all of the time zone parameters
  - even Daylight Savings Time
- Localize a timestamp using `tz_localize`
  - ```
    ts = pd.Timestamp("1 Dec 2016 12:30 PM")
    ts = ts.tz_localize("US/Eastern")
    ```
- Convert a timestamp using tz_convert
  - ```
    ts.tz_convert("Europe/Budapest")
    ```
- Operations involving timestamps from different time zones become UTC

Frequency

- Generic time series in pandas are **irregular**
  - there is no fixed frequency
  - we don't necessarily have data for every day/hour/etc.
- Date ranges have frequency

```
In [76]: pd.date_range(start='2012-04-01', periods=20)
Out[76]:
DatetimeIndex(['2012-04-01', '2012-04-02', '2012-04-03', '2012-04-04',
               '2012-04-05', '2012-04-06', '2012-04-07', '2012-04-08',
               '2012-04-09', '2012-04-10', '2012-04-11', '2012-04-12',
               '2012-04-13', '2012-04-14', '2012-04-15', '2012-04-16',
               '2012-04-17', '2012-04-18', '2012-04-19', '2012-04-20'],
              dtype='datetime64[ns]', freq='D')
```

Lots of Frequencies (not comprehensive)

Alias	Offset type	Description
D	Day	Calendar daily
B	BusinessDay	Business daily
H	Hour	Hourly
T or min	Minute	Minutely
S	Second	Secondly
L or ms	Milli	Millisecond (1/1,000 of 1 second)
U	Micro	Microsecond (1/1,000,000 of 1 second)
M	MonthEnd	Last calendar day of month
BM	BusinessMonthEnd	Last business day (weekday) of month
MS	MonthBegin	First calendar day of month
BMS	BusinessMonthBegin	First weekday of month
W-MON, W-TUE, ...	Week	Weekly on given day of week (MON, TUE, WED, THU, FRI, SAT, or SUN)
WOM-1MON, WOM-2MON, ...	WeekOfMonth	Generate weekly dates in the first, second, third, or fourth week of the month (e.g., WOM-3FRI for the third Friday of each month)
Q-JAN, Q-FEB, ...	QuarterEnd	Quarterly dates anchored on last calendar day of each month, for year ending in indicated month (JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, or DEC)
BQ-JAN, BQ-FEB, ...	BusinessQuarterEnd	Quarterly dates anchored on last weekday day of each month, for year ending in indicated month
QS-JAN, QS-FEB, ...	QuarterBegin	Quarterly dates anchored on first calendar day of each month, for year ending in indicated month
BQS-JAN, BQS-FEB, ...	BusinessQuarterBegin	Quarterly dates anchored on first weekday day of each month, for year ending in indicated month
A-JAN, A-FEB, ...	YearEnd	Annual dates anchored on last calendar day of given month (JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, or DEC)
BA-JAN, BA-FEB, ...	BusinessYearEnd	Annual dates anchored on last weekday of given month
AS-JAN, AS-FEB, ...	YearBegin	Annual dates anchored on first day of given month
BAS-JAN, BAS-FEB, ...	BusinessYearBegin	Annual dates anchored on first weekday of given month

[W. McKinney, Python for Data Analysis]

Resampling

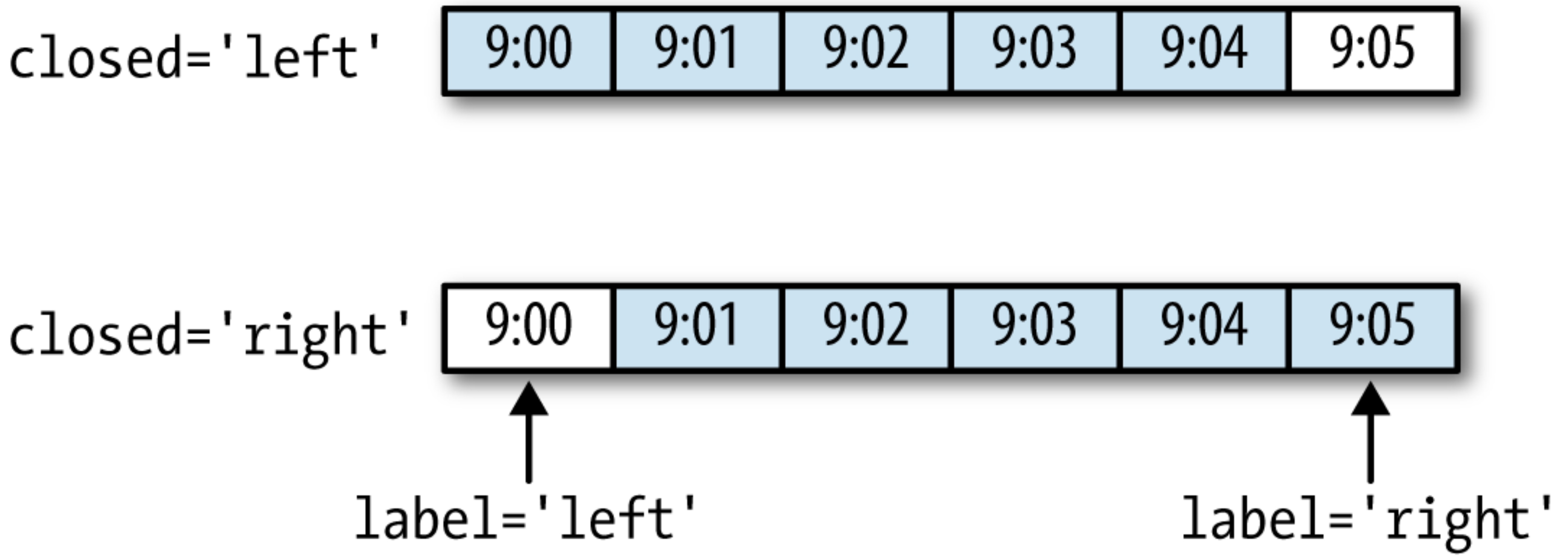
- Could be
  - downsample: higher frequency to lower frequency
  - upsample: lower frequency to higher frequency
  - neither: e.g. Wednesdays to Fridays
- resample method: e.g. `ts.resample('M').mean()`

Argument	Description
`freq`	String or DateOffset indicating desired resampled frequency (e.g., ‘M', '5min', or `Second(15)`)
`axis`	Axis to resample on; default axis=0
`fill_method`	How to interpolate when upsampling, as in `'ffill'` or `'bfill'`; by default does no interpolation
`closed`	In downsampling, which end of each interval is closed (inclusive), `'right'` or `'left'`
`label`	In downsampling, how to label the aggregated result, with the `'right'` or `'left'` bin edge (e.g., the 9:30 to 9:35 five-minute interval could be labeled `9:30` or `9:35`)
`loffset`	Time adjustment to the bin labels, such as `'-1s'` / `Second(-1)` to shift the aggregate labels one second earlier
`limit`	When forward or backward filling, the maximum number of periods to fill
`kind`	Aggregate to periods (`'period'`) or timestamps (`'timestamp'`); defaults to the type of index the time series has
`convention`	When resampling periods, the convention (`'start'` or `'end'`) for converting the low-frequency period to high frequency; defaults to `'end'`

[W. McKinney, Python for Data Analysis]

Downsampling

- Need to define **bin edges** which are used to group the time series into **intervals** that can be aggregated
- Remember:
  - Which side of the interval is closed
  - How to label the aggregated bin (start or end of interval)



Upsampling

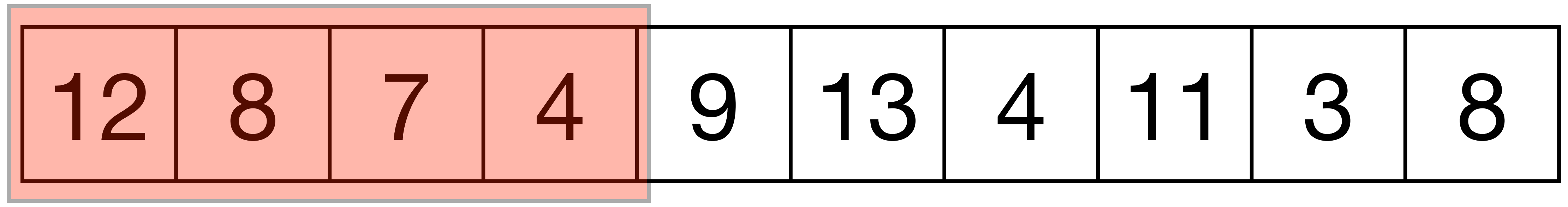
- No aggregation necessary

```
In [222]: frame
Out[222]:
            Colorado     Texas  New York      Ohio
2000-01-05 -0.896431  0.677263  0.036503  0.087102
2000-01-12 -0.046662  0.927238  0.482284 -0.867130
```

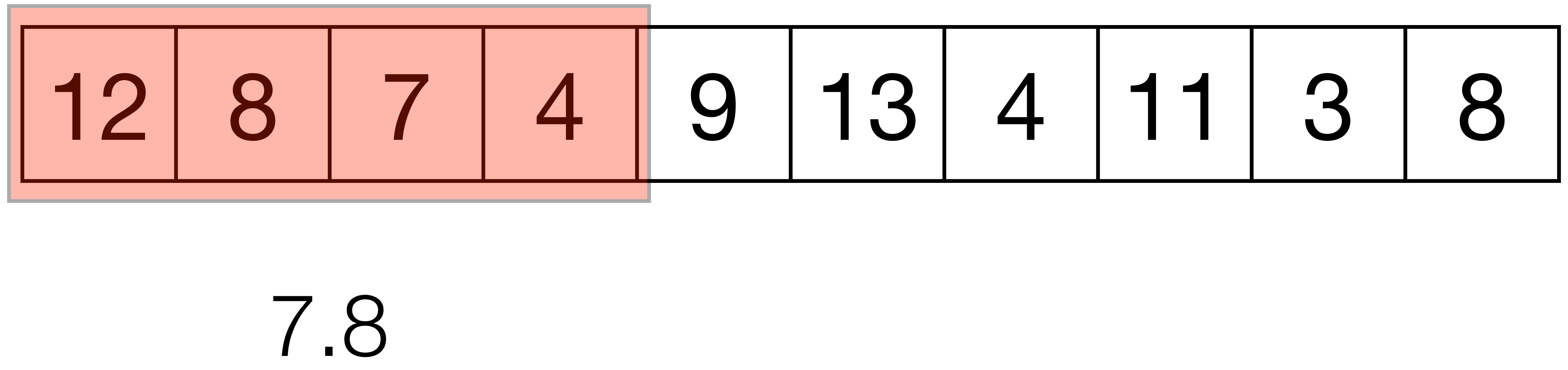
```
In [223]: df_daily = frame.resample('D').asfreq()

In [224]: df_daily
Out[224]:
            Colorado     Texas  New York      Ohio
2000-01-05 -0.896431  0.677263  0.036503  0.087102
2000-01-06       NaN       NaN       NaN       NaN
2000-01-07       NaN       NaN       NaN       NaN
2000-01-08       NaN       NaN       NaN       NaN
2000-01-09       NaN       NaN       NaN       NaN
2000-01-10       NaN       NaN       NaN       NaN
2000-01-11       NaN       NaN       NaN       NaN
2000-01-12 -0.046662  0.927238  0.482284 -0.867130
```

```
In [225]: frame.resample('D').ffill()
Out[225]:
            Colorado     Texas  New York      Ohio
2000-01-05 -0.896431  0.677263  0.036503  0.087102
2000-01-06 -0.896431  0.677263  0.036503  0.087102
2000-01-07 -0.896431  0.677263  0.036503  0.087102
2000-01-08 -0.896431  0.677263  0.036503  0.087102
2000-01-09 -0.896431  0.677263  0.036503  0.087102
2000-01-10 -0.896431  0.677263  0.036503  0.087102
2000-01-11 -0.896431  0.677263  0.036503  0.087102
2000-01-12 -0.046662  0.927238  0.482284 -0.867130
```

Rolling Window Calculations



Rolling Window Calculations

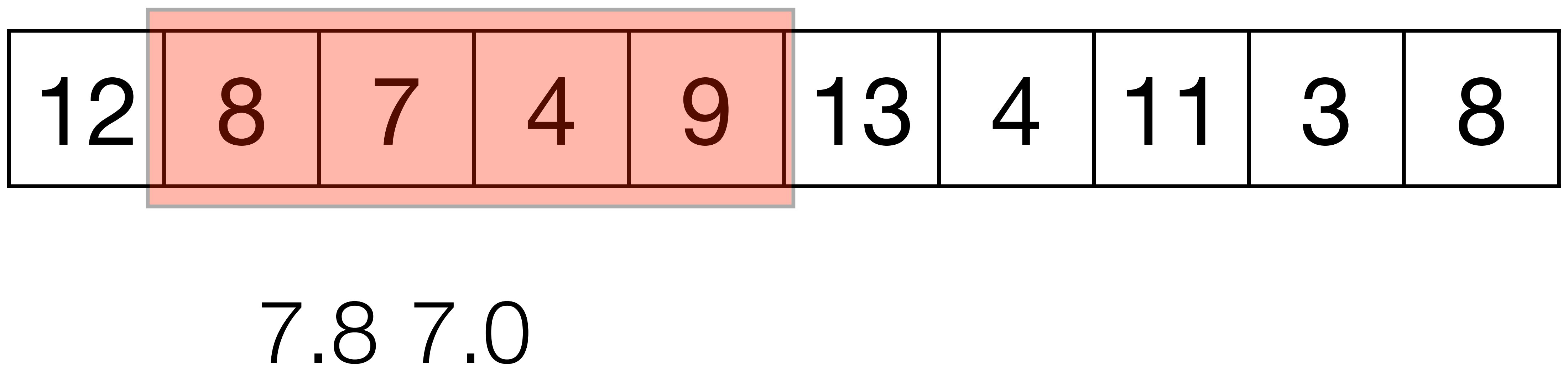


Rolling Window Calculations

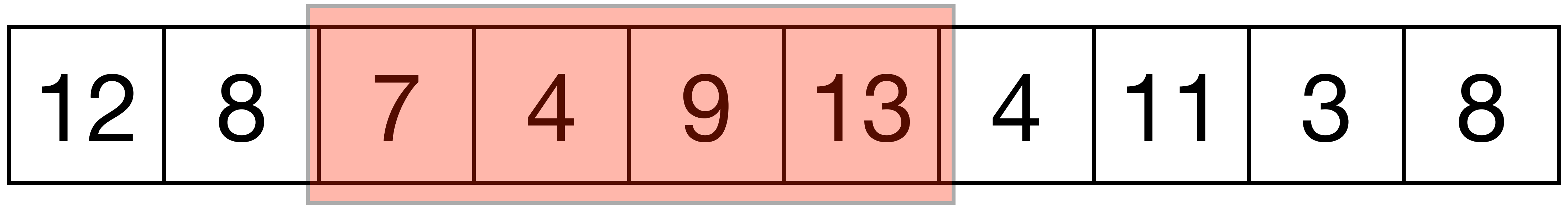
12	8	7	4	9	13	4	11	3	8

7.8

Rolling Window Calculations

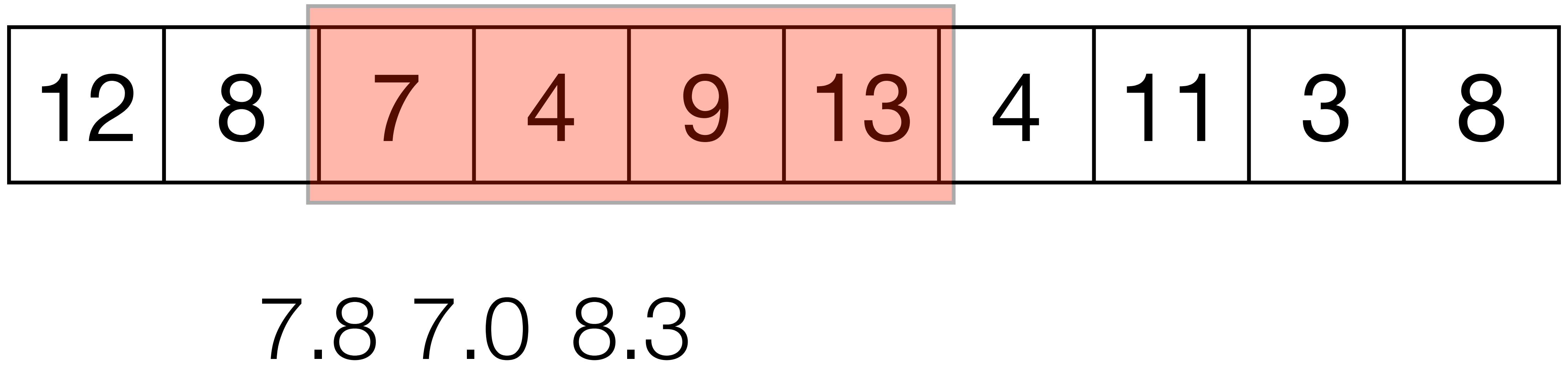


Rolling Window Calculations



7.8 7.0

Rolling Window Calculations



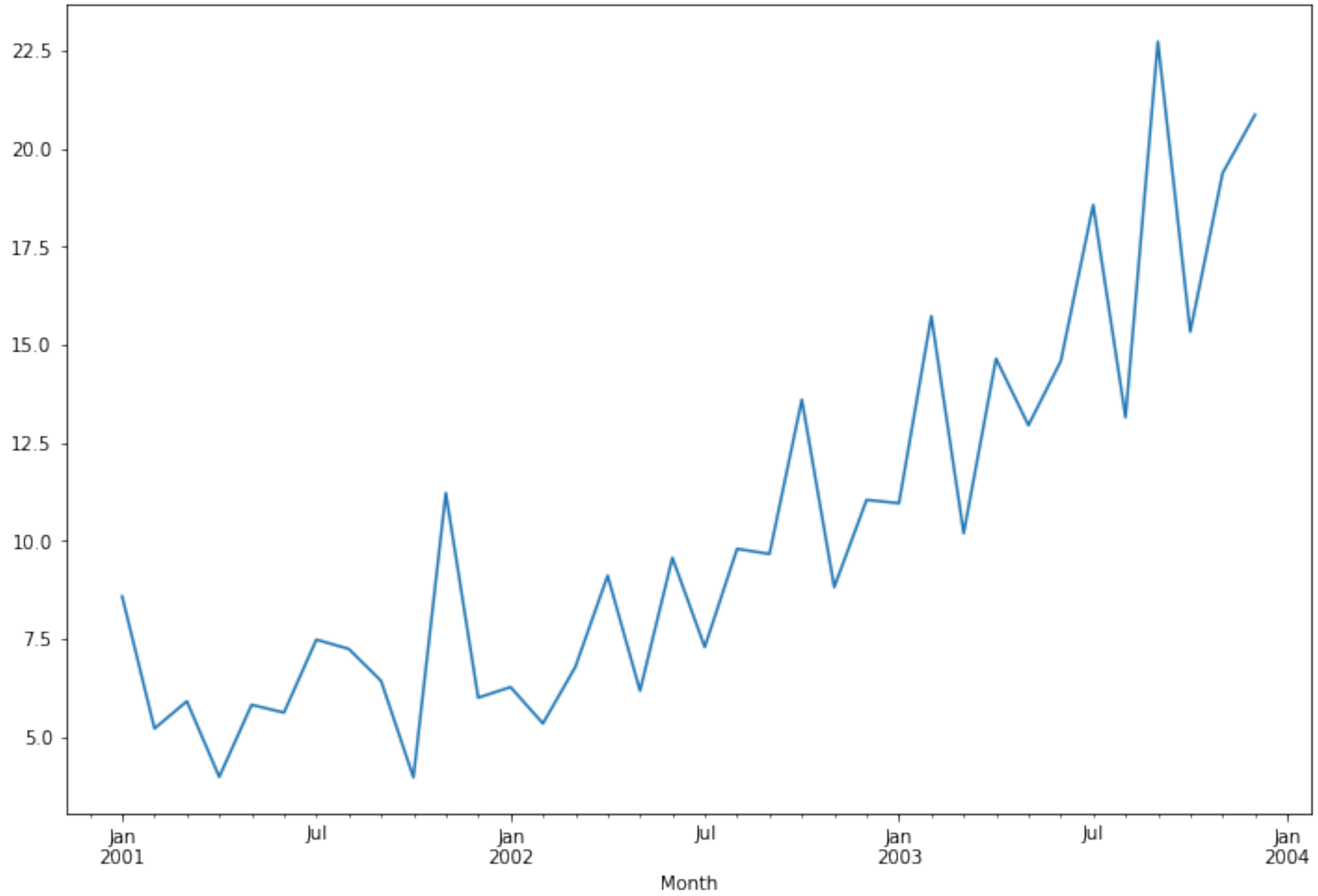
Window Functions

- Idea: want to aggregate over a window of time, calculate the answer, and then slide that window ahead. Repeat.
- `rolling`: smooth out data
- Specify the window size in rolling, then an aggregation method
- Result is set to the right edge of window (change with `center=True`)
- Example:
  - `df.rolling('180D').mean()`
  - `df.rolling('90D').sum()`

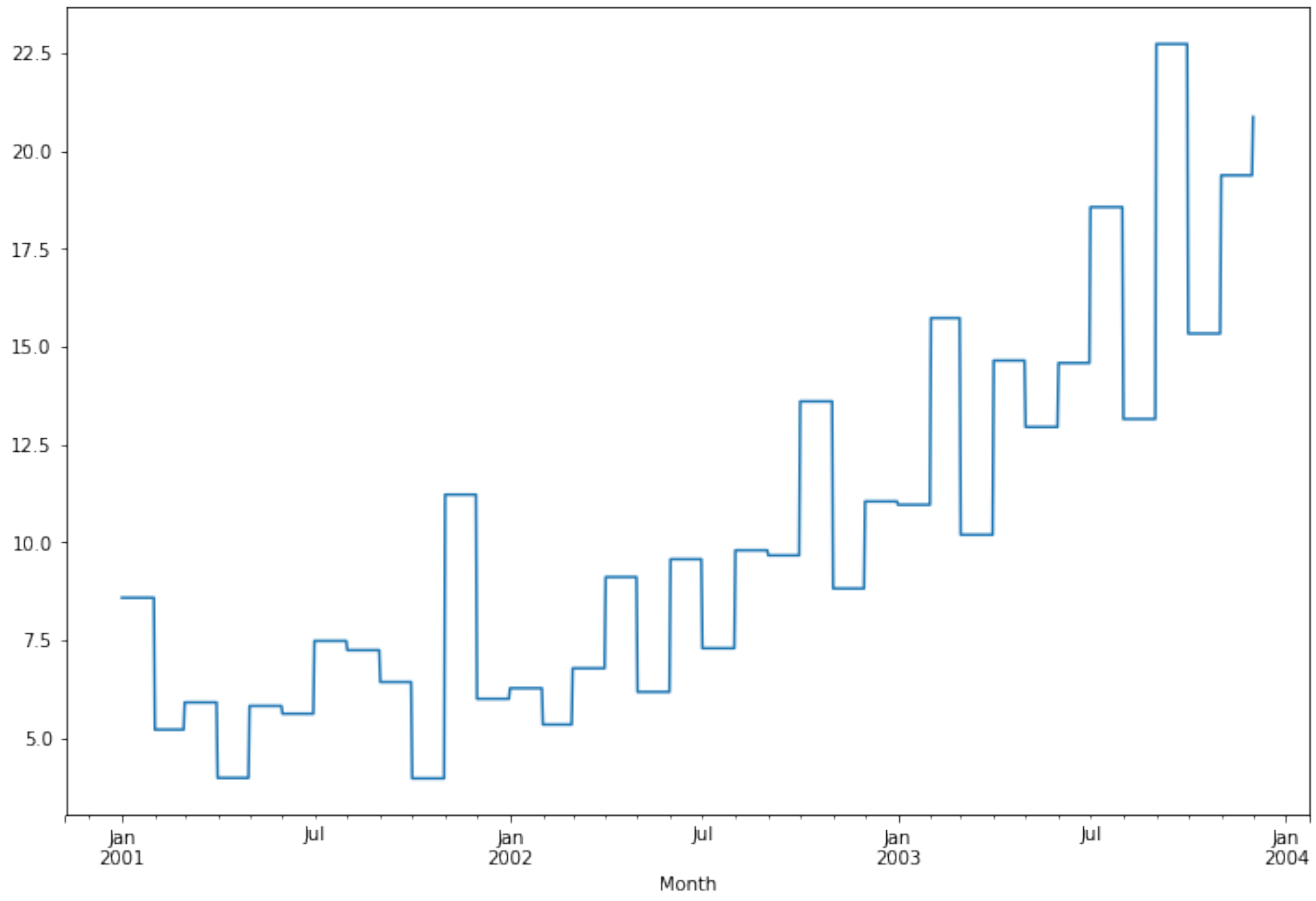
Interpolation

- Fill in the missing values with computed best estimates using various types of algorithms
- Apply after resample

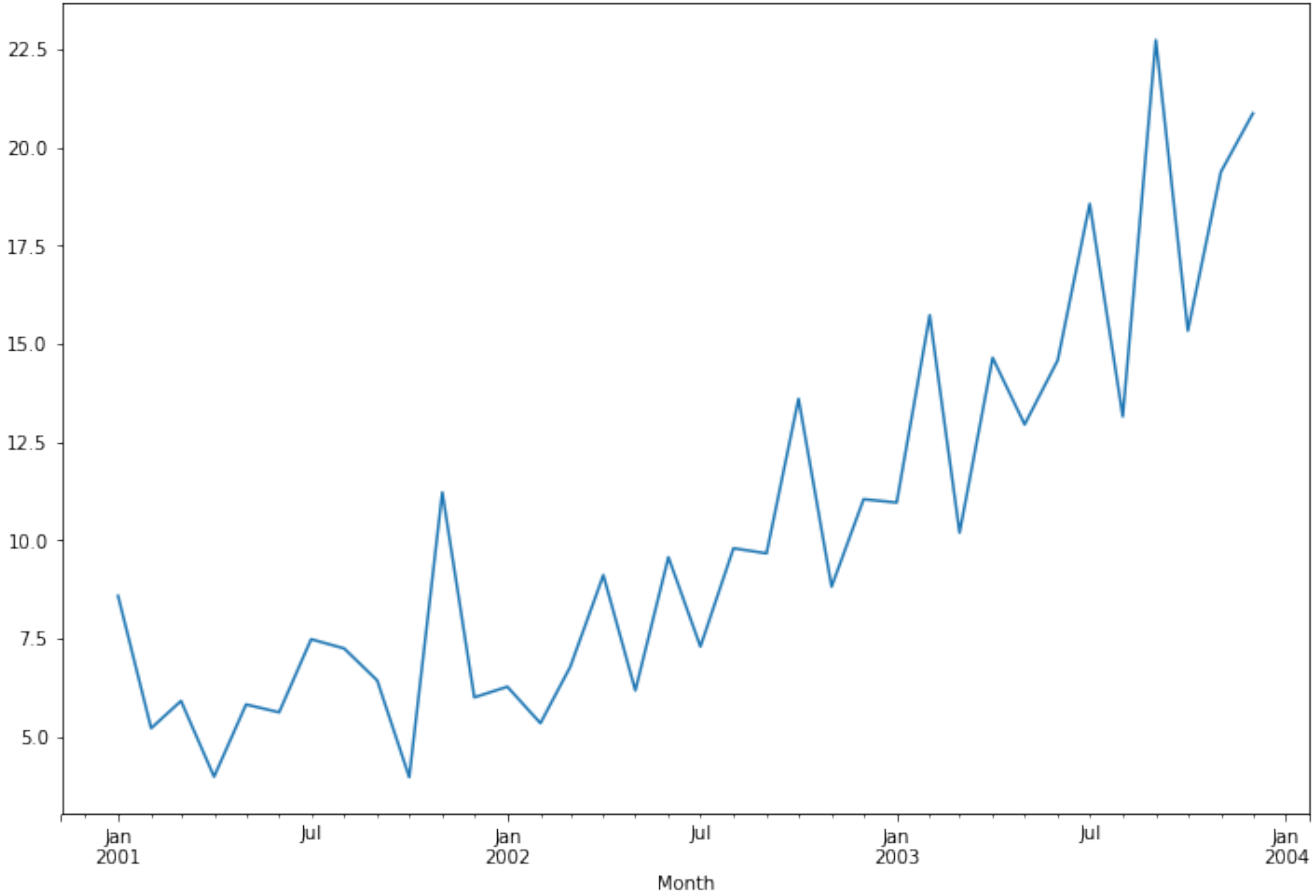
Sales Data by Month



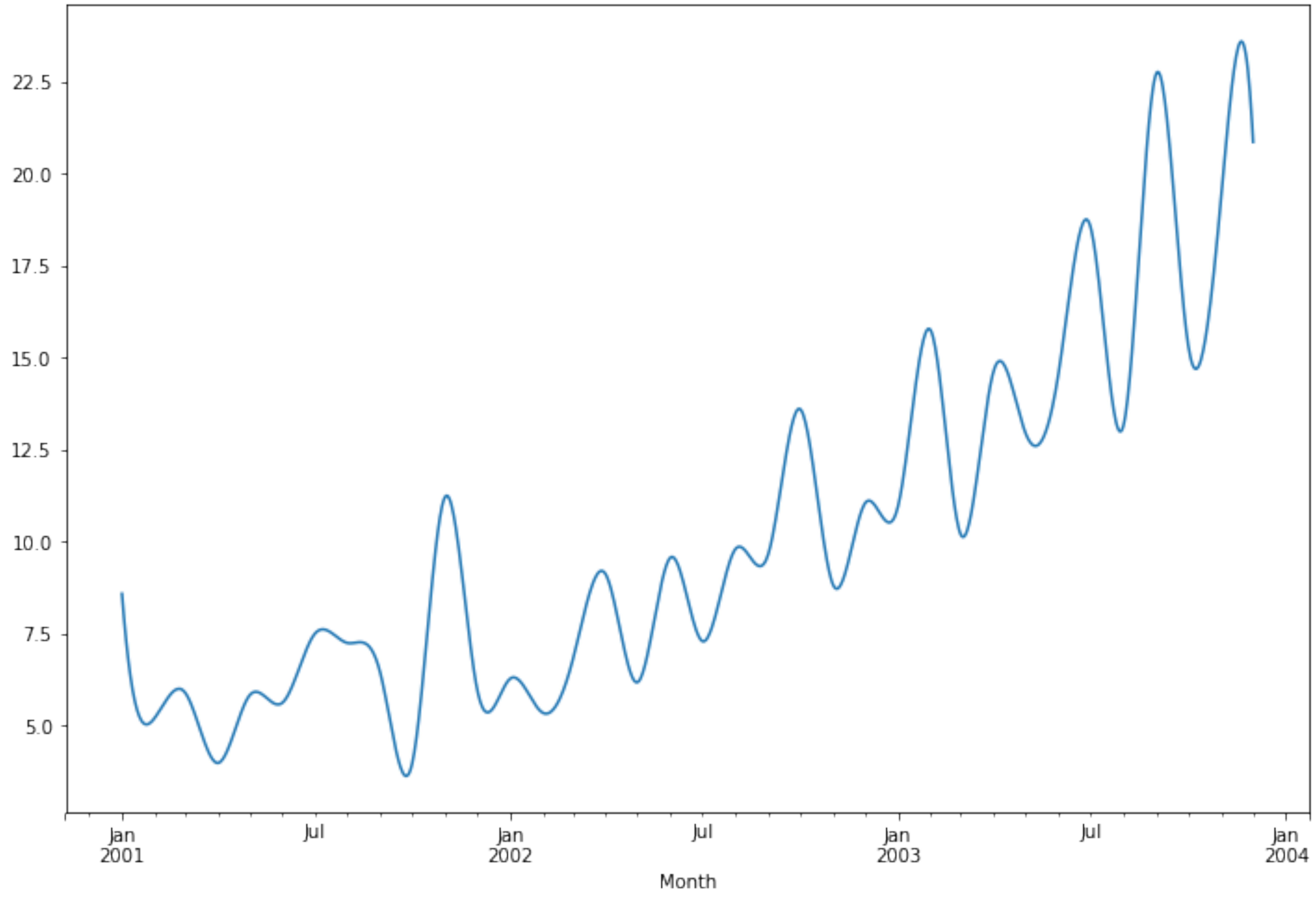
Resampled Sales Data (ffill)



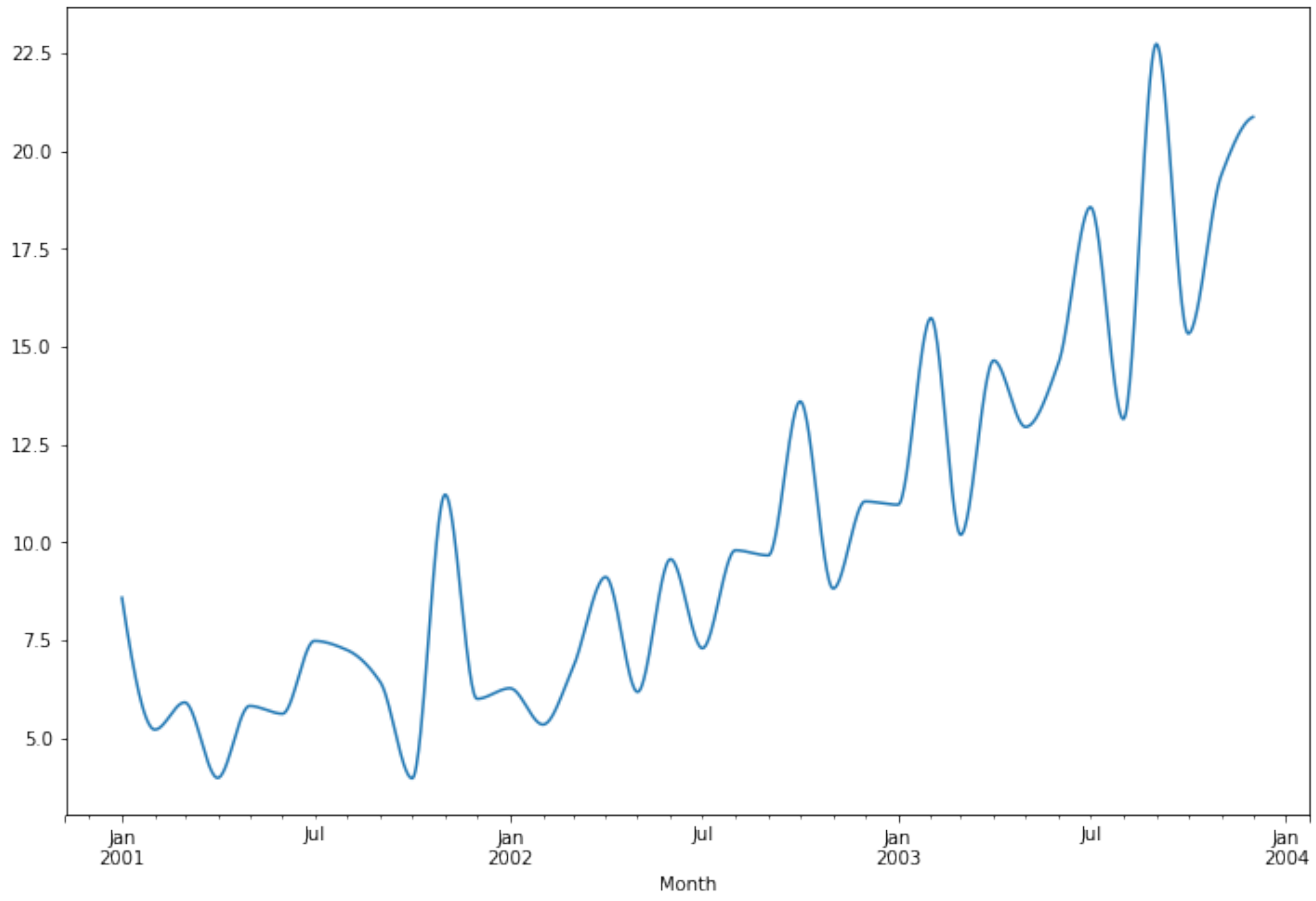
Resampled with Linear Interpolation (Default)



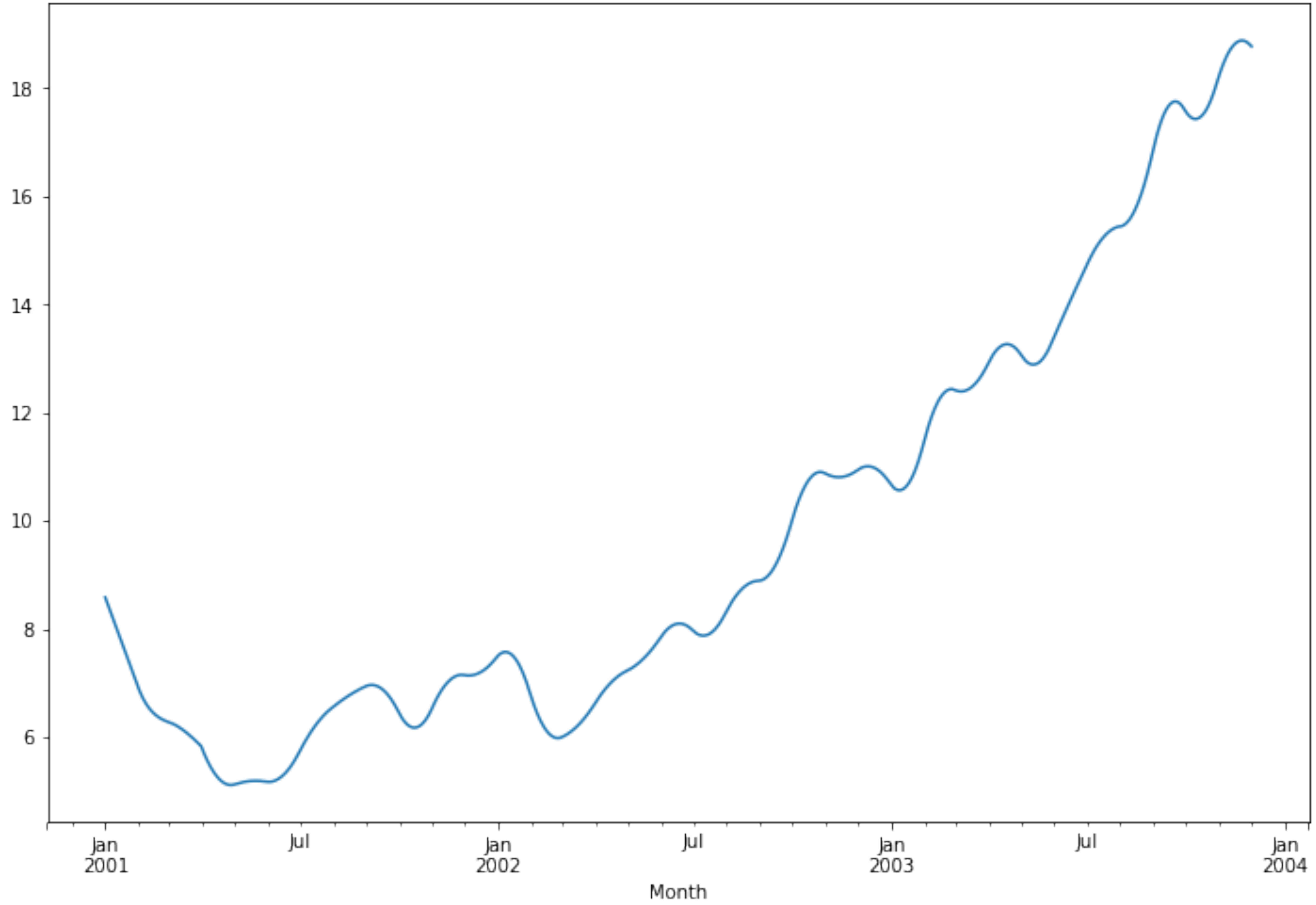
Resampled with Cubic Interpolation



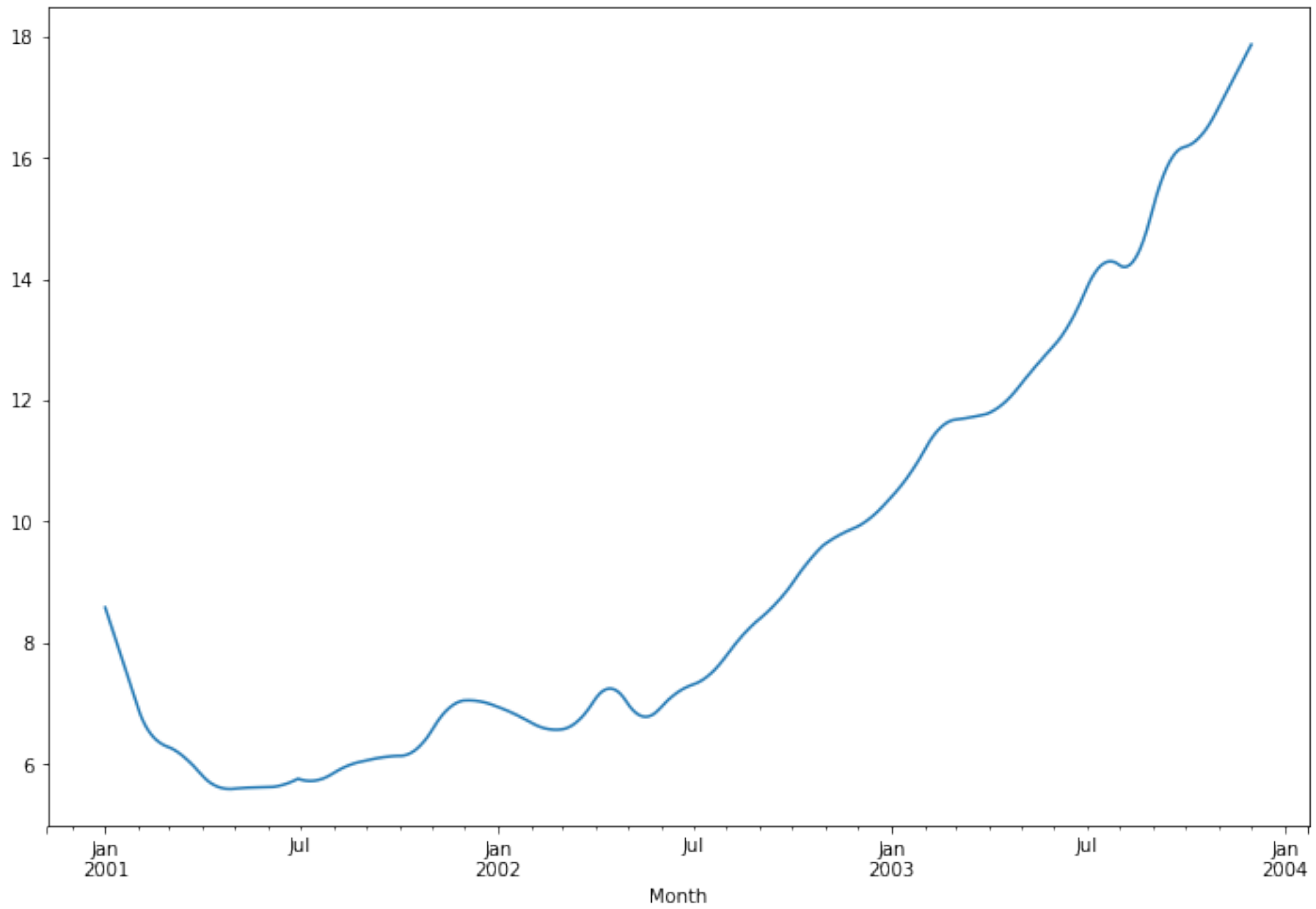
Piecewise Cubic Hermite Interpolating Polynomial



90-Day Rolling Window (Mean)



180-Day Rolling Window (Mean)



Provenance

What actually happened in a computational experiment?

Provenance in Art

Rembrandt van Rijn
Dutch, 1606 - 1669
Self-Portrait, 1659
oil on canvas
Andrew W. Mellon Collection
1937.1.72

Provenance

George, 3rd Duke of Montagu and 4th Earl of Cardigan [d. 1790], by 1767;[1] by inheritance to his daughter, Lady Elizabeth, wife of Henry, 3rd Duke of Buccleuch of Montagu House, London; John Charles, 7th Duke of Buccleuch; (P. & D. Colnaghi & Co., New York, 1928); (M. Knoedler & Co., New York); sold January 1929 to Andrew W. Mellon, Pittsburgh and Washington, D.C.; deeded 28 December 1934 to The A.W. Mellon Educational and Charitable Trust, Pittsburgh; gift 1937 to NGA.

[1] This early provenance is established by presence of a mezzotint after the portrait by R. Earlom (1743-1822), dated 1767. See John Charrington, A Catalogue of the Mezzotints After, or Said to Be After, Rembrandt, Cambridge, 1923, no. 49.

Associated Names

- Buccleuch, Henry, 3rd Duke of
- Buccleuch, John Charles, 7th Duke of
- Colnaghi & Co., Ltd., P. & D.
- Knoedler & Company, M.
- Mellon, Andrew W.
- Mellon Educational and Charitable Trust, The A.W.
- Montagu, and 4th Earl of Cardigan, George, 3rd Duke of

[National Gallery of Art]

Provenance in Art

Rembrandt van Rijn
Dutch, 1606 - 1669
Self-Portrait, 1659
oil on canvas
Andrew W. Mellon Collection
1937.1.72

Provenance

George, 3rd Duke of Montagu and 4th Earl of Cardigan [d. 1790], by 1767;[1] by inheritance to his daughter, Lady Elizabeth, wife of Henry, 3rd Duke of Buccleuch of Montagu House, London; John Charles, 7th Duke of Buccleuch; (P. & D. Colnaghi & Co., New York, 1928); (M. Knoedler & Co., New York); sold January 1929 to Andrew W. Mellon, Pittsburgh and Washington, D.C.; deeded 28 December 1934 to The A.W. Mellon Educational and Charitable Trust, Pittsburgh; gift 1937 to NGA.

[1] This early provenance is established by presence of a mezzotint after the portrait by R. Earlom (1743-1822), dated 1767. See John Charrington, A Catalogue of the Mezzotints After, or Said to Be After, Rembrandt, Cambridge, 1923, no. 49.

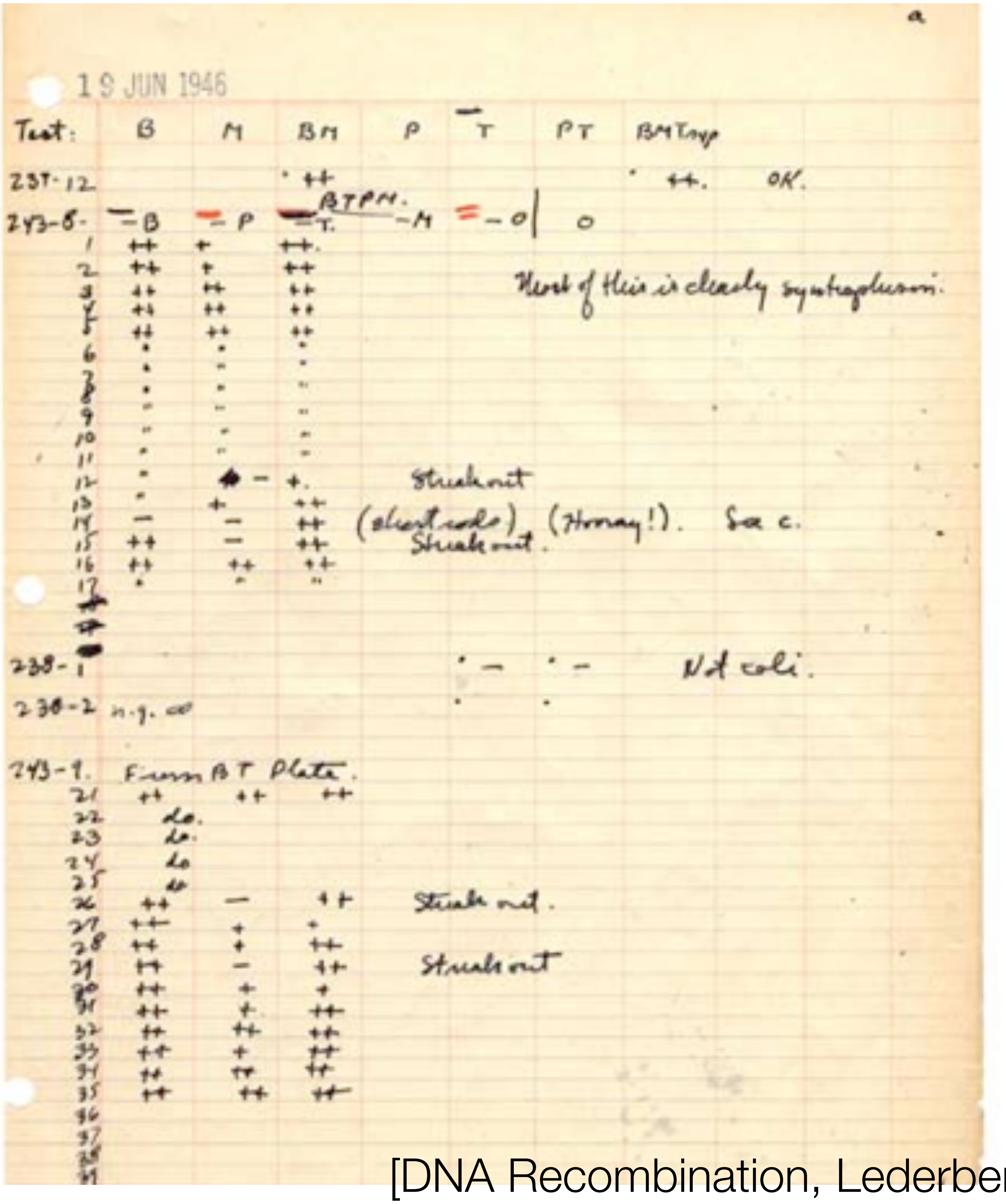
Associated Names

- Buccleuch, Henry, 3rd Duke of
- Buccleuch, John Charles, 7th Duke of
- Colnaghi & Co., Ltd., P. & D.
- Knoedler & Company, M.
- Mellon, Andrew W.
- Mellon Educational and Charitable Trust, The A.W.
- Montagu, and 4th Earl of Cardigan, George, 3rd Duke of

[National Gallery of Art]

Provenance in Science

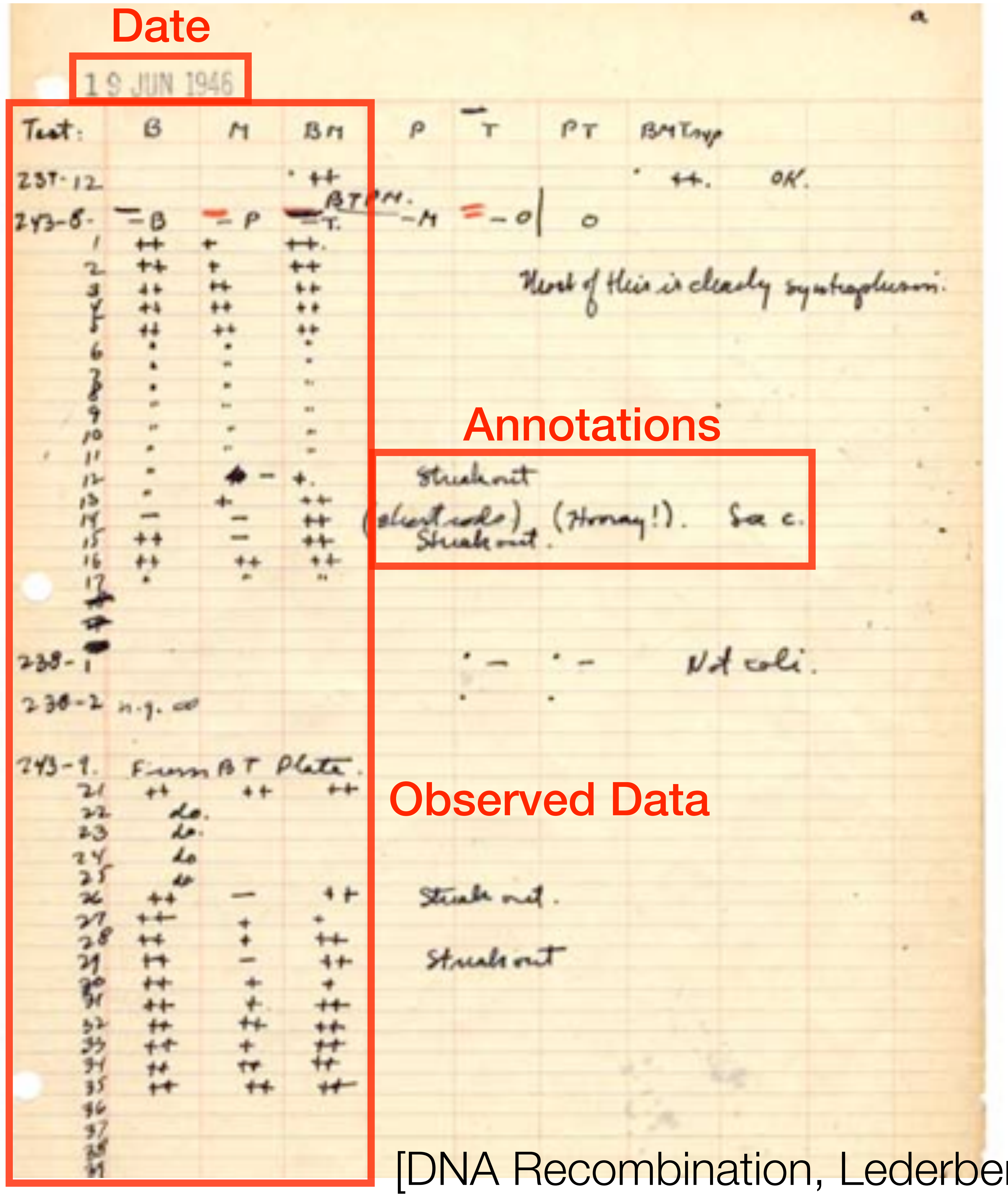
- Provenance: the lineage of data, a computation, or a visualization
- **Provenance is as (or more) important as the result!**
- Old solution:
  - Lab notebooks
- New problems:
  - Large volumes of data
  - Complex analyses
  - Writing notes doesn't scale



[DNA Recombination, Lederberg]

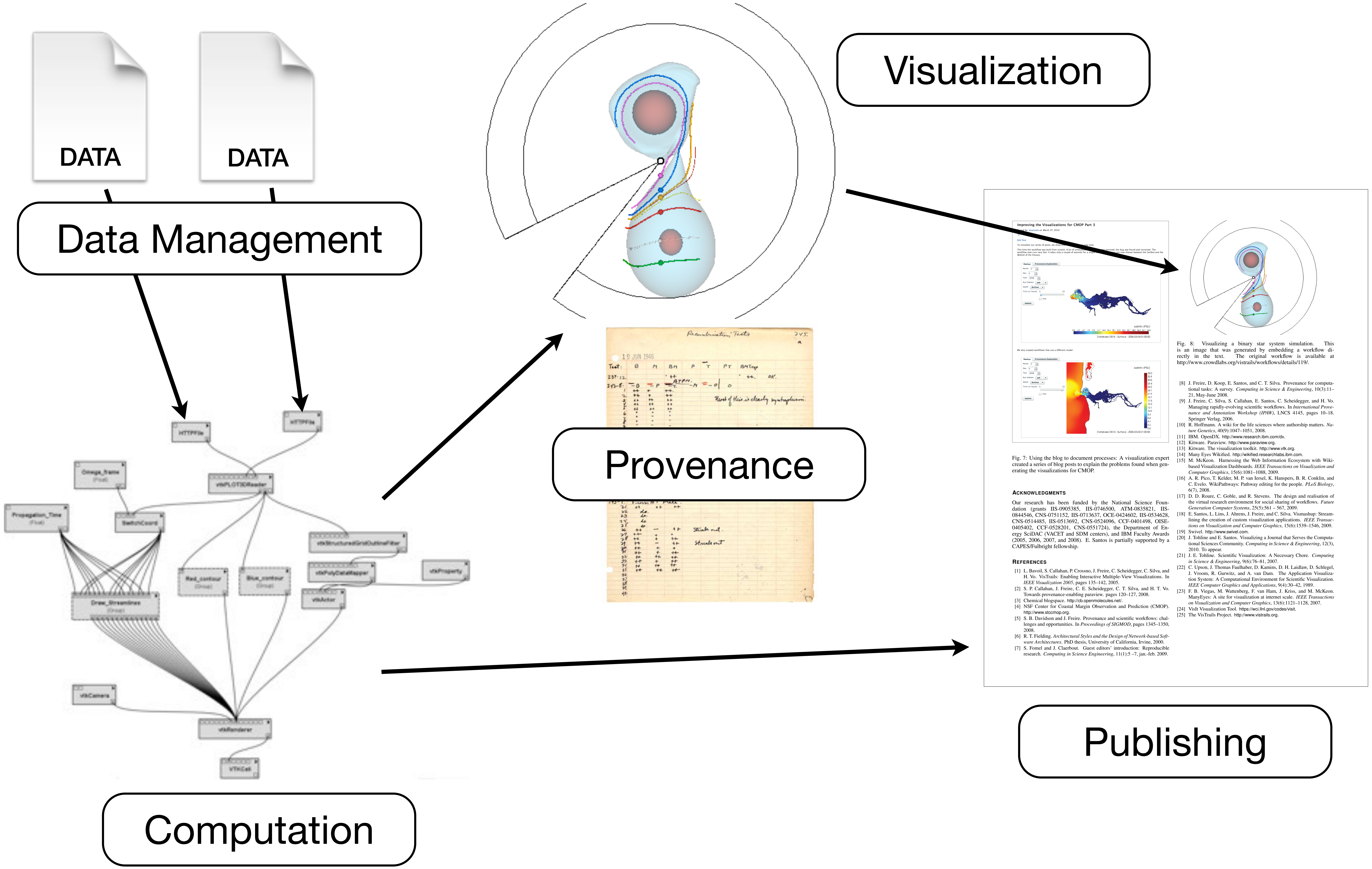
Provenance in Science

- Provenance: the lineage of data, a computation, or a visualization
- **Provenance is as (or more) important as the result!**
- Old solution:
  - Lab notebooks
- New problems:
  - Large volumes of data
  - Complex analyses
  - Writing notes doesn't scale



[DNA Recombination, Lederberg]

Provenance in Computational Science



Evolution of Publication

- Publish paper
- Publish code
- Publish computational experiments/tests
- Publish provenance (what actually happens during your runs)

Provenance-Rich Publication

Galois Conjugates of Topological Phases

M. H. Freedman,[1] J. Gukelberger,[2] M. B. Hastings,[1] S. Trebst,[1] M. Troyer,[2] and Z. Wang[1]
[1]*Microsoft Research, Station Q, University of California, Santa Barbara, CA 93106, USA*
[2]*Theoretische Physik, ETH Zurich, 8093 Zurich, Switzerland*
(Dated: July 6, 2011)

Galois conjugation relates unitary conformal field theories (CFTs) and topological quantum field theories (TQFTs) to their non-unitary counterparts. Here we investigate Galois conjugates of quantum double models, such as the Levin-Wen model. While these Galois conjugated Hamiltonians are typically non-Hermitian, we find that their ground state wave functions still obey a generalized version of the usual code property (local operators do not act on the ground state manifold) and hence enjoy a generalized topological protection. The key question addressed in this paper is whether such non-unitary topological phases can also appear as the ground states of Hermitian Hamiltonians. Specific attempts at constructing Hermitian Hamiltonians with these ground states lead to a loss of the code property and topological protection of the degenerate ground states. Beyond this we rigorously prove that no local change of basis (IV.5) can transform the ground states of the Galois conjugated doubled Fibonacci theory into the ground states of a topological model whose Hermitian Hamiltonian satisfies Lieb-Robinson bounds. These include all gapped local or quasi-local Hamiltonians. A similar statement holds for many other non-unitary TQFTs. One consequence is that the "Gaffnian" wave function cannot be the ground state of a gapped fractional quantum Hall state.

PACS numbers: 05.30.Pr, 73.43.-f

I. INTRODUCTION

Galois conjugation, by definition, replaces a root of a polynomial by another one with identical algebraic properties. For example, i and $-i$ are Galois conjugate (consider $z^2+1=0$) as are $\phi = \frac{1+\sqrt{5}}{2}$ and $-\frac{1}{\phi} = \frac{1-\sqrt{5}}{2}$ (consider $z^2-z-1=0$), as well as $\sqrt[3]{2}$, $\sqrt[3]{2}e^{2\pi i/3}$, and $\sqrt[3]{2}e^{-2\pi i/3}$ (consider $z^3-2=0$). In physics Galois conjugation can be used to convert non-unitary conformal field theories (CFTs) to unitary ones, and vice versa. One famous example is the non-unitary Yang-Lee CFT, which is Galois conjugate to the Fibonacci CFT $(G_2)_1$, the even (or integer-spin) subset of su(2)$_3$.

In statistical mechanics non-unitary conformal field theories have a venerable history.[1,2] However, it has remained less clear if there exist physical situations in which non-unitary models can provide a useful description of the low energy physics of a quantum mechanical system – after all, Galois conjugation typically destroys the Hermitian property of the Hamiltonian. Some non-Hermitian Hamiltonians, which surprisingly have totally real spectrum, have been found to arise in the study of PT-invariant one-particle systems[3] and in some Galois conjugate many-body systems[4] and might be seen to open the door a crack to the physical use of such models. Another situation, which has recently attracted some interest, is the question whether non-unitary models can describe 1D edge states of certain 2D bulk states (the edge holographic for the bulk). In particular, there is currently a discussion on whether or not the "Gaffnian" wave function could be the ground state for a *gapped* fractional quantum Hall (FQH) state albeit with a non-unitary "Yang-Lee" CFT describing its edge.[5–7] We conclude that this is not possible, further restricting the possible scope of non-unitary models in quantum mechanics.

We reach this conclusion quite indirectly. Our main thrust is the investigation of Galois conjugation in the simplest non-Abelian Levin-Wen model.[8] This model, which is also called "DFib", is a topological quantum field theory (TQFT) whose states are string-nets on a surface labeled by either a trivial or "Fibonacci" anyon. From this starting point, we give a rigorous argument that the "Gaffnian" ground state cannot be locally conjugated to the ground state of any topological phase, within a Hermitian model satisfying Lieb-Robinson (LR) bounds[9] (which includes but is not limited to gapped local and quasi-local Hamiltonians).

Lieb-Robinson bounds are a technical tool for local lattice models. In relativistically invariant field theories, the speed of light is a strict upper bound to the velocity of propagation. In lattice theories, the LR bounds provide a similar upper bound by a velocity called the LR velocity, but in contrast to the relativistic case there can be some exponentially small "leakage" outside the light-cone in the lattice case. The Lieb-Robinson bounds are a way of bounding the leakage outside the light-cone. The LR velocity is set by microscopic details of the Hamiltonian, such as the interaction strength and range. Combining the LR bounds with the spectral gap enables us to prove locality of various correlation and response functions. We will call a Hamiltonian a *Lieb-Robinson Hamiltonian* if it satisfies LR bounds.

We work primarily with a single example, but it should be clear that the concept of Galois conjugation can be widely applied to TQFTs. The essential idea is to retain the particle types and fusion rules of a unitary theory but when one comes to writing down the algebraic form of the F-matrices (also called $6j$ symbols), the entries are now Galois conjugated. A slight complication, which is actually an asset, is that writing an F-matrix requires a gauge choice and the most convenient choice may differ before and after Galois conjugation.

Our method is not restricted to Galois conjugated DFib$^{\mathcal{G}}$ and its factors Fib$^{\mathcal{G}}$ and $\overline{\text{Fib}^{\mathcal{G}}}$, but can be generalized to infinitely many non-unitary TQFTs, showing that they will not arise as low energy models for a gapped 2D quantum mechan-

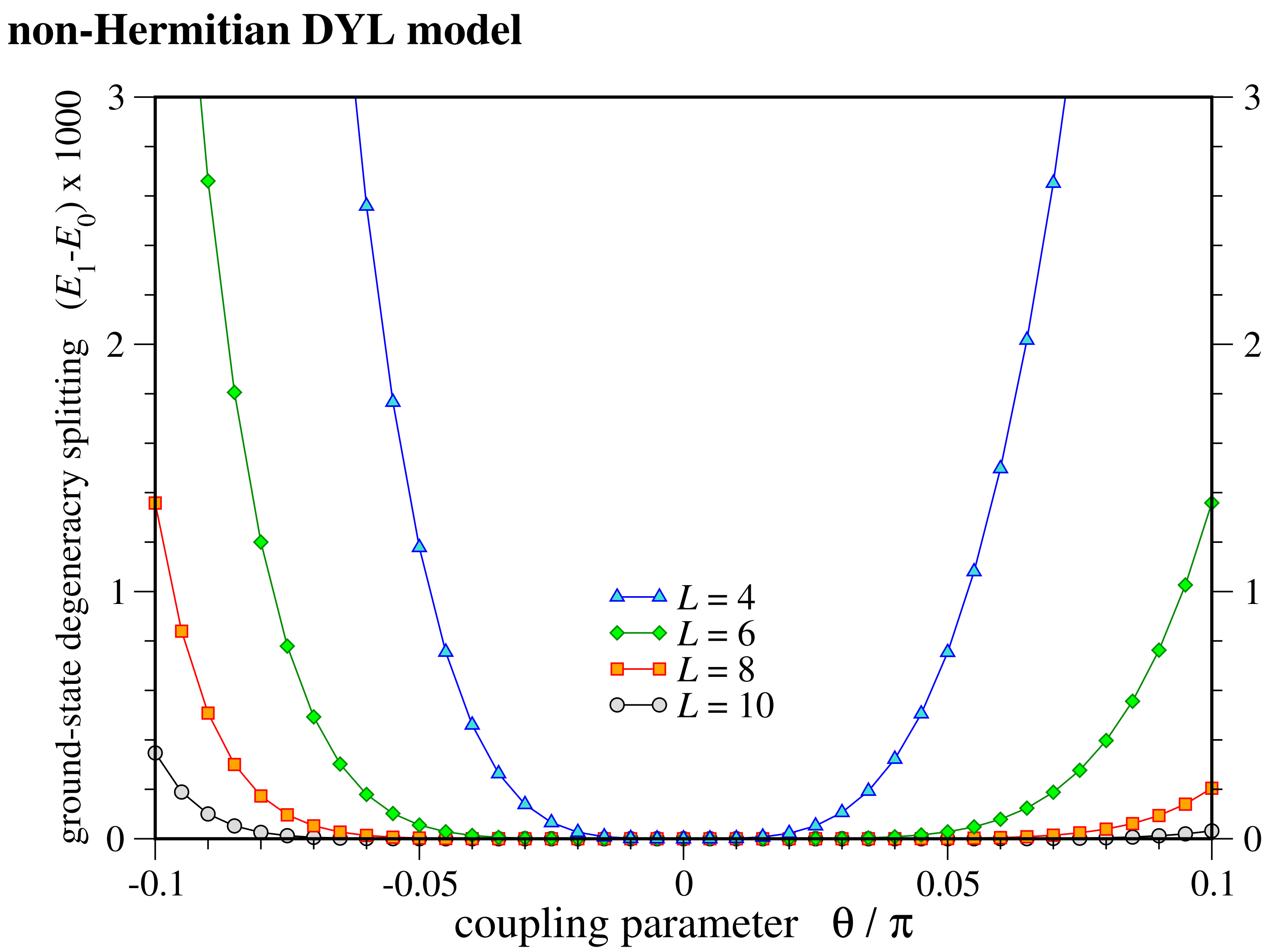


FIG. 6. (color online) Ground-state degeneracy splitting of the non-Hermitian doubled Yang-Lee model when perturbed by a string tension ($\theta \neq 0$).

[Freedman et al., 2012]

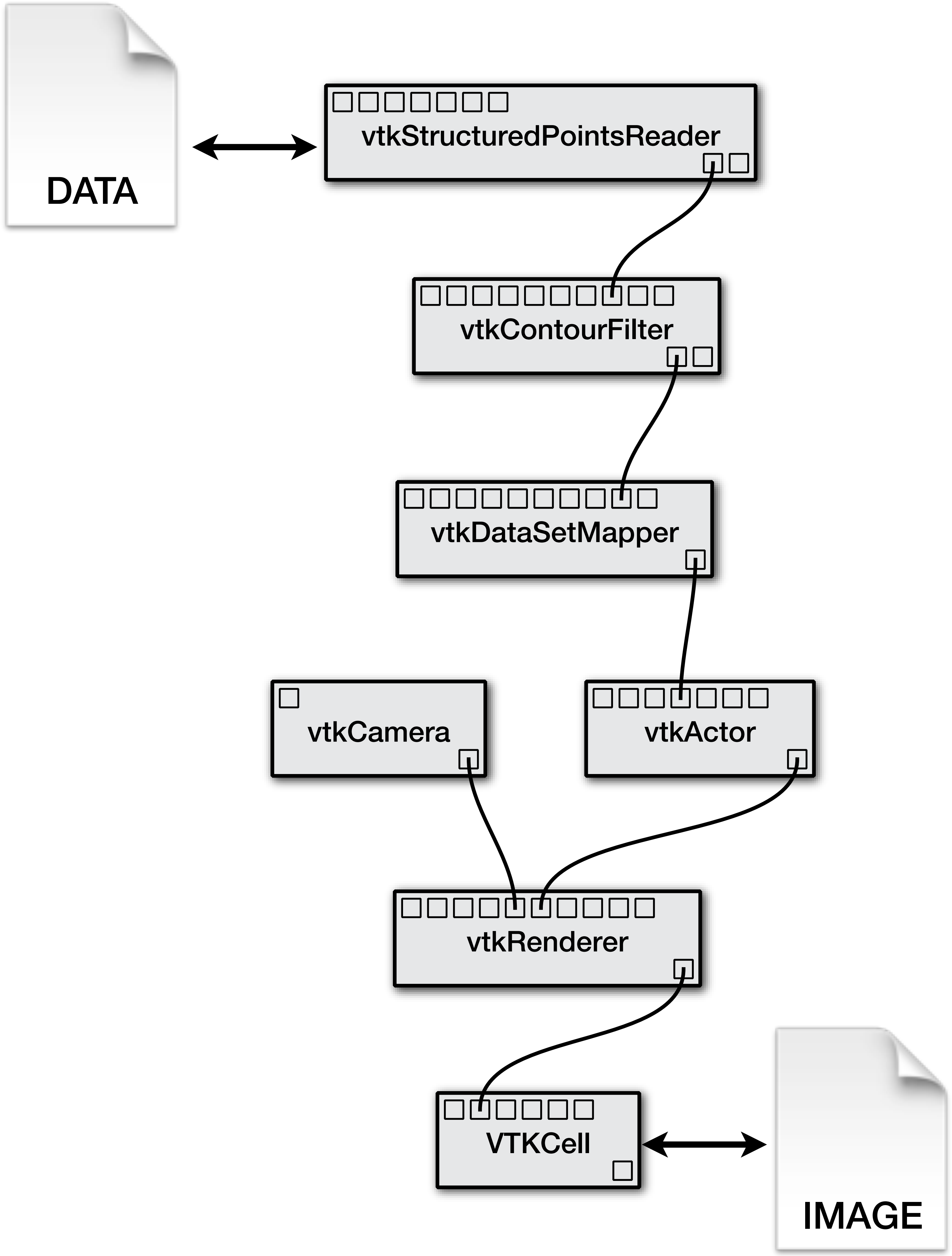
Benefits of Provenance-Rich Publications

- Produce more knowledge–not just text
- Allow scientists to stand on the shoulders of giants (and their own)
- Science can move faster!
- Higher-quality publications
- Authors will be more careful
- Many eyes to check results
- Describe more of the discovery process: people only describe successes, can we learn from mistakes?
- Expose users to different techniques and tools: expedite their training; and potentially reduce their time to insight

Provenance Definitions

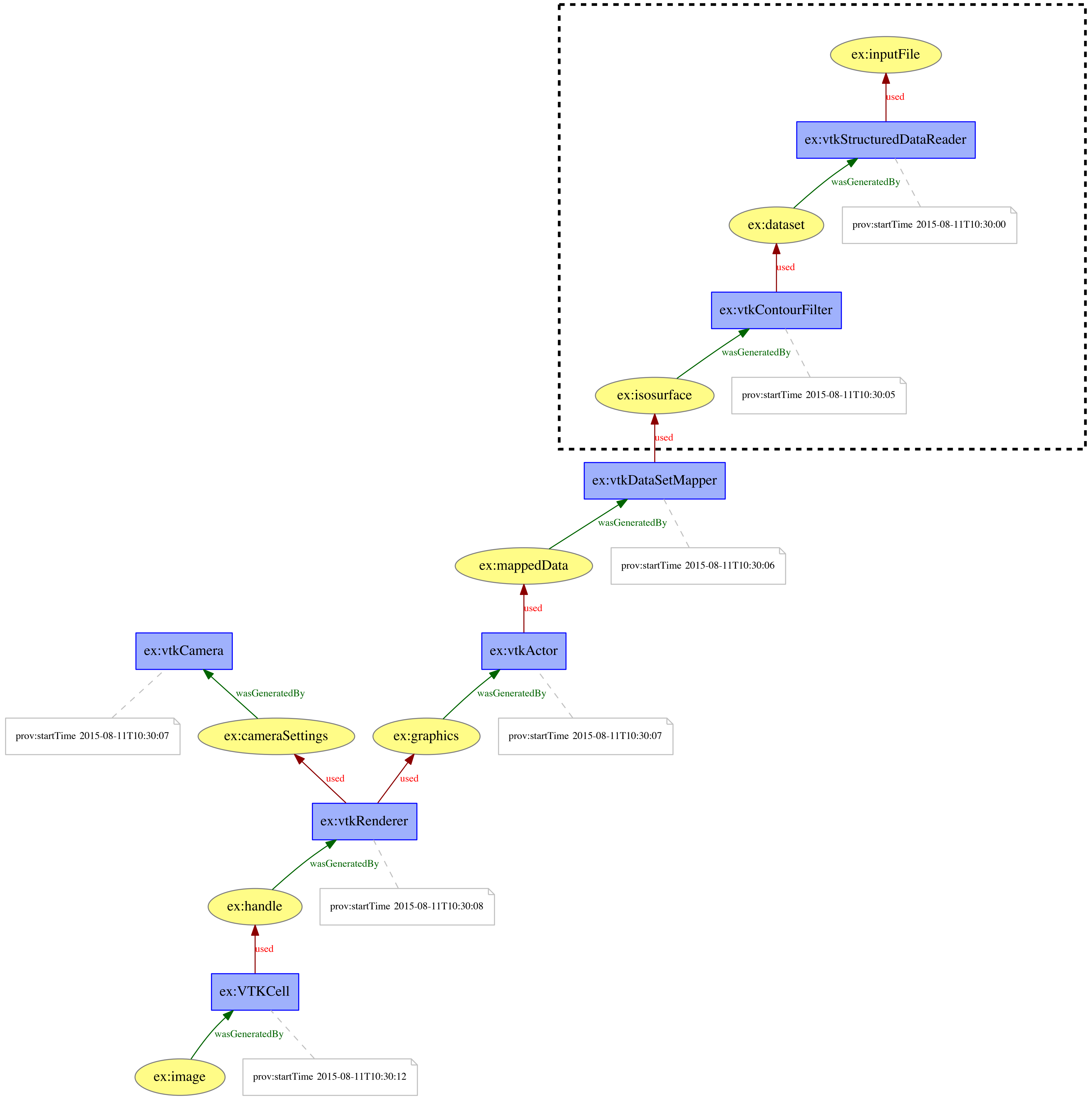
- Dictionary: "the source or origin of an object; its history and pedigree; a record of the ultimate derivation and passage of an item through its various owners."
- Focus on **causality**—the sequence of steps that detail how a result was generated and/or **derivation**—what data a result depended on
- Provenance itself is **data**, this list of steps along with metadata for each step: when it occurred, who initiated it, notes about it
- Can be used to preserve information about an experiment and to answer many questions

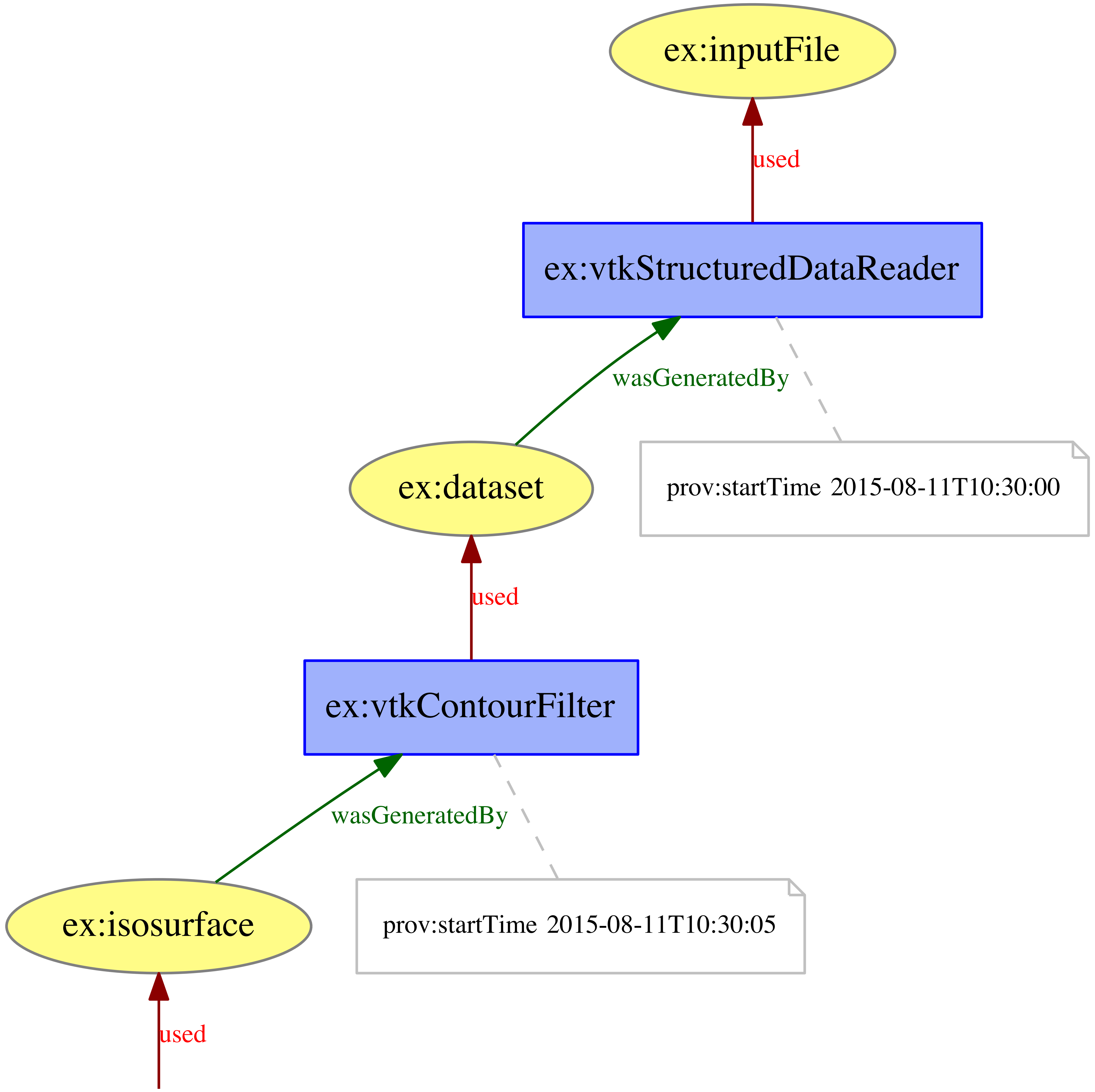
Workflows



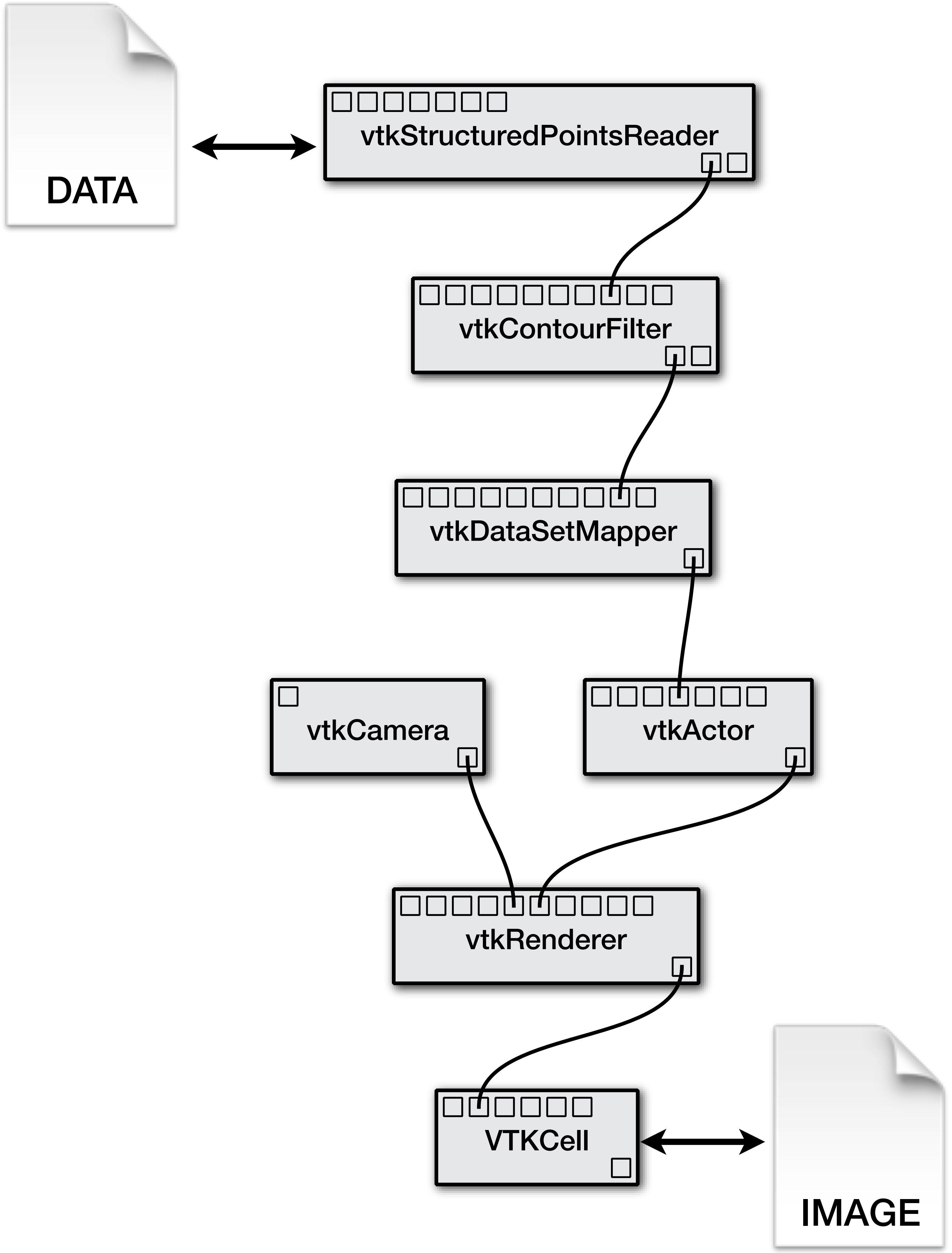
- Abstract computation
- Computational modules connected through input and output ports
- Data flows along the connections

Provenance Graph





Provenance Questions



- What process led to the output image?
- What input datasets contributed to the output image?
- What workflows create an isosurface with isovalue `57`?
- Who create this data product?
- When was this data file created?
- Why was `vtkCamera` used?
- Why do two output images differ?

Questions about Provenance

- How does one capture provenance?
- How does one manage provenance for later use?
- How do we answer questions about our provenance?
- How do we use provenance for good?

Provenance Management

- Provenance can be generated from tasks/programs/scripts/etc.
- Properties of provenance are related to the **computational model**
  - a specific application with a graphical interface
  - a script that automates the use of several command-line tools
  - a scientific workflow that combines several tools

Provenance & Causality

- Knowing what data/steps influenced other data/steps is important!
- Data dependencies: this output file depended on this input file
- Data-process dependencies: this output figure depended on these processes
- Causality can often be represented as a **graph** where connections represent dependencies

User-defined provenance

- Goal: capture lots of provenance automatically based on what steps are executed
- Problem: not everything can be captured automatically
- Annotations offer ability to keep notes about processes
- Users might also specify known causal links that cannot be automatically determined (e.g. a step depends on three system files that were not specified as inputs in the workflow)

Provenance Management

- What is needed to capture, store, and use provenance?
  1. Capture mechanism
  2. Model for representing provenance
  3. Tools to store, query, and analyze provenance

Provenance Capture Mechanisms

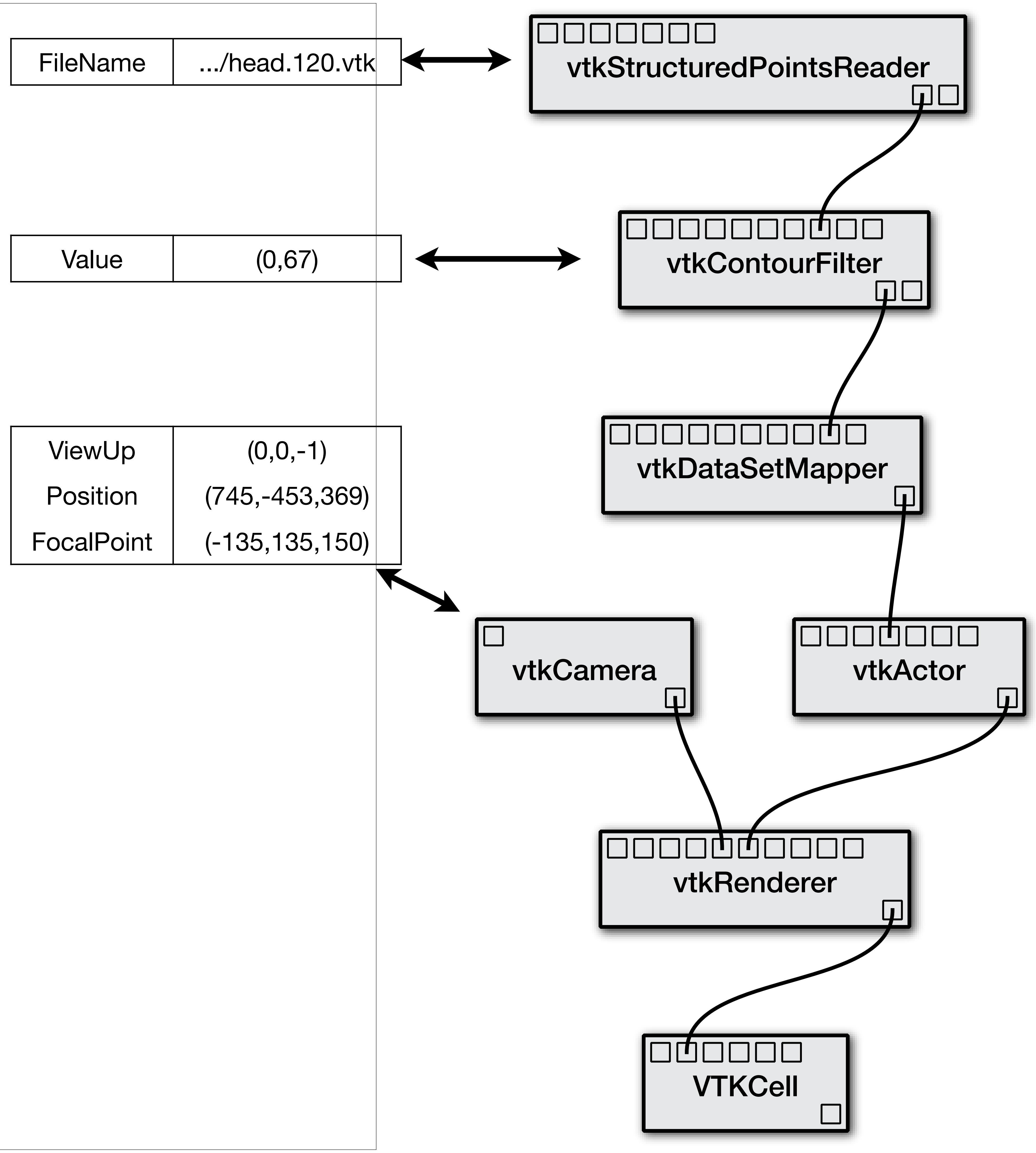
- **Workflow-based**: Since workflow execution is controlled, keep track of all the workflow modules, parameters, etc. as they are executed
- **Process-based**: Each process is required to write out its own provenance information (not centralized like workflow-based)
- **OS-based**: The OS or filesystem is modified so that any activity it does it monitored and the provenance subsystem organizes it
- Tradeoffs:
  - Workflow- and process-based have better abstraction
  - OS-based requires minimal user effort once installed and can capture "hidden dependencies"

Provenance Granularity

- How detailed should our provenance be?
  - **Coarse**: "This program ran with inputs x, y, z and produced outputs a, b, c"
  - **Fine**: "Input x was read into register 4, input y was read in register 5, add operation was performed using registers 4 and 5, …"
- More queries are possible with fine-grained provenance, but…
  - Storage concerns
  - Performance concerns
- Abstraction can help here

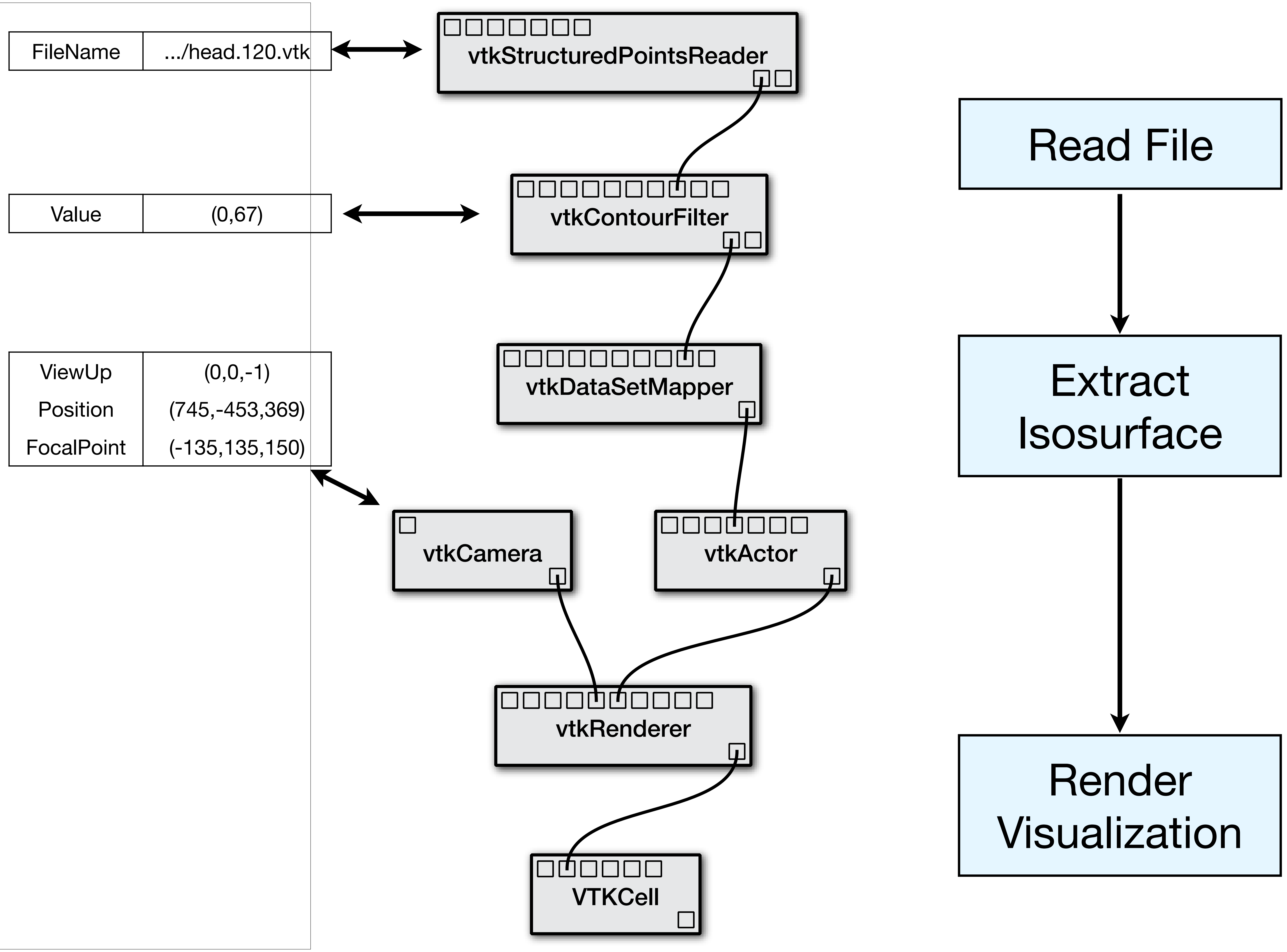
Abstraction: Script, Workflow, Abstract Workflow

```
data = vtk.vtkStructuredPointsReader()
data.SetFileName(../examples/data/head.120.vtk)

contour = vtk.vtkContourFilter()
contour.SetInput(data.GetOutput())
contour.SetValue(0, 67)

mapper = vtk.vtkPolyDataMapper()
mapper.SetInput(contour.GetOutput())
mapper.ScalarVisibilityOff()

actor = vtk.vtkActor()
actor.SetMapper(mapper)

cam = vtk.vtkCamera()
cam.SetViewUp(0,0,-1)
cam.SetPosition(745,-453,369)
cam.SetFocalPoint(135,135,150)
cam.ComputeViewPlaneNormal()

ren = vtk.vtkRenderer()
ren.AddActor(actor)
ren.SetActiveCamera(cam)
ren.ResetCamera()
renwin = vtk.vtkRenderWindow()
renwin.AddRenderer(ren)

style = vtk.vtkInteractorStyleTrackballCamera()
iren = vtk.vtkRenderWindowInteractor()
iren.SetRenderWindow(renwin)
iren.SetInteractorStyle(style)
iren.Initialize()
iren.Start()
```

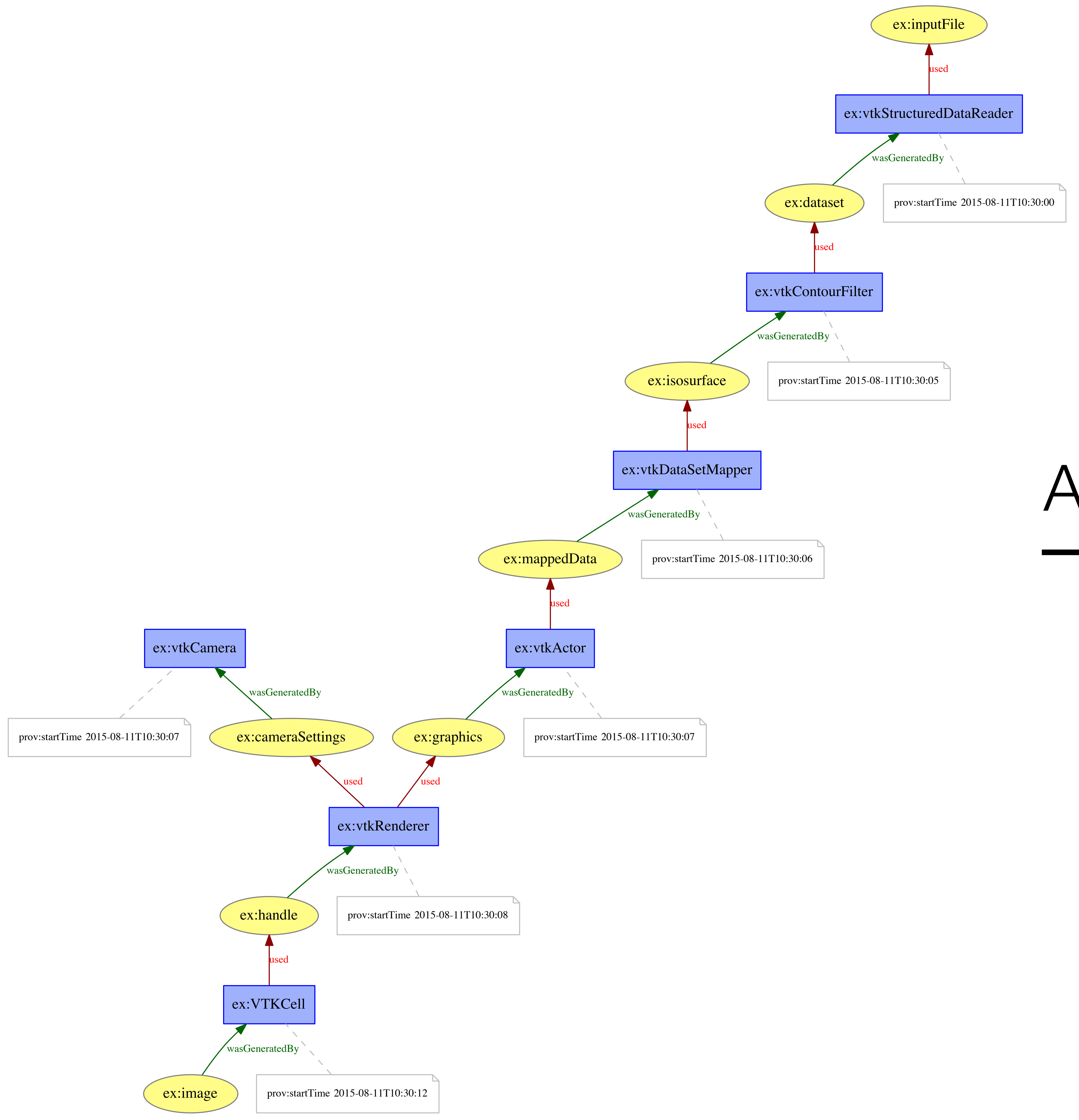


Abstraction: Script, Workflow, Abstract Workflow

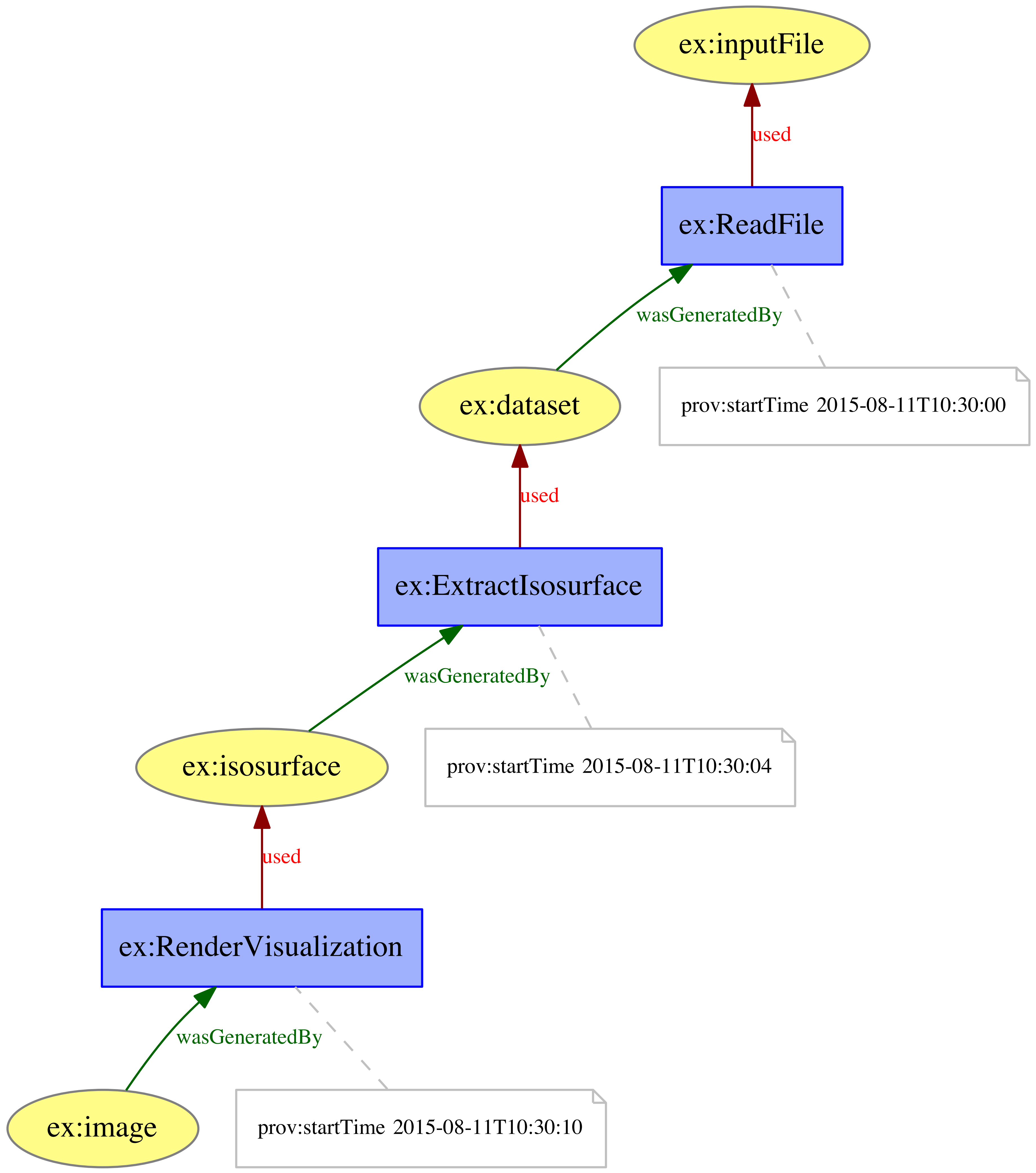
```
data = vtk.vtkStructuredPointsReader()
data.SetFileName(../examples/data/head.120.vtk)

contour = vtk.vtkContourFilter()
contour.SetInput(data.GetOutput())
contour.SetValue(0, 67)

mapper = vtk.vtkPolyDataMapper()
mapper.SetInput(contour.GetOutput())
mapper.ScalarVisibilityOff()

actor = vtk.vtkActor()
actor.SetMapper(mapper)

cam = vtk.vtkCamera()
cam.SetViewUp(0,0,-1)
cam.SetPosition(745,-453,369)
cam.SetFocalPoint(135,135,150)
cam.ComputeViewPlaneNormal()

ren = vtk.vtkRenderer()
ren.AddActor(actor)
ren.SetActiveCamera(cam)
ren.ResetCamera()
renwin = vtk.vtkRenderWindow()
renwin.AddRenderer(ren)

style = vtk.vtkInteractorStyleTrackballCamera()
iren = vtk.vtkRenderWindowInteractor()
iren.SetRenderWindow(renwin)
iren.SetInteractorStyle(style)
iren.Initialize()
iren.Start()
```



Abstraction: Provenance Views



Abstract →



geopandas and neo4j

Teaching Evaluations