ASAP: Automatic Smoothing for Attention Prioritization in Time Series Visualization

Kexin Rong, Peter Bailis
Stanford University
IoT generates lots of time series data
Problem: Noisy Visualizations

Short-term fluctuations can obscure long-term trends

HARD TO READ!

SMOOTHED: MUCH BETTER!

This talk: how to get the smooth plot automatically
This talk: how to smooth plots automatically

Motivation:
more informative time series visualizations

Big idea: smooth your visualizations!
this talk: how much to smooth?

Why smooth?
38% more accurate + 44% faster responses

Try it yourself:
JavaScript library ASAP.js
What do my visualizations tell me today?

Many plotting libraries we’ve seen plot raw data directly!

Is plotting raw data always the best idea?
Is plotting raw data always the best idea?

Example: Two servers from same cluster (production data)

Are these two servers really the same?
Is plotting raw data always the best idea?

Example: Monthly temperature in England from 250 years

Temperature fluctuates on a yearly cycle => 250 spikes

Excel
Key Takeaway: Smooth your visualizations!

A little smoothing can go a long way

Average temperature increases from the early 1900s
Q: What’s distracting about raw data?
A: In many cases, spikes dominate the plot.

Short-term fluctuations are overrepresented relative to the overall trends.
Talk Outline

Motivation:
raw data is often noisy

Observation:
smoothing helps highlight trends

Our research:
smooth automatically with ASAP

Going fast:
optimizations for fast rendering
How should we smooth visualizations?

Q: What smoothing function should we use?

A: Moving average works

Signal Processing Literature: Optimal for removing noise
How should we smooth visualizations?

Q: How much to smooth? (What window size to use?)

Original

Window too small?
Noisy

Window too large?
Lose structure
How should we smooth visualizations?

Q: How much to smooth?

A: New approach called ASAP! Make your plots:

\[
\begin{array}{ll}
\text{As} & \text{while} \\
\text{Smooth} & \text{preserving} \\
\text{As} & \text{long-term} \\
\text{Possible} & \text{deviations}
\end{array}
\]
How should we smooth visualizations?

How should we quantify smoothness?

As Smooth As Possible while preserving long-term deviations
How should we quantify smoothness?

<table>
<thead>
<tr>
<th>Measure</th>
<th>Series A</th>
<th>Series B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Standard Deviation</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Point-to-Point Variance</td>
<td>4</td>
<td>0</td>
</tr>
</tbody>
</table>

Point-to-point differences?

Not Smooth:

- Time: 0, Value: 2
- Time: 1, Value: -2
- Time: 2, Value: 2
- Time: 3, Value: -2
- Time: 4

Smooth:

- Time: 0
- Time: 1
- Time: 2
- Time: 3
- Time: 4
- Value: .7
- Value: .7
- Value: .7
How should we smooth visualizations?

How should we quantify smoothness?

- As smooth as possible while preserving long-term deviations
- Increase window size until...?
Constraint: Preserve deviations in plots

**Goal:** avoid oversmoothing

**Idea:** measure the “outlyingness” of the plot

**Metric:** measure the kurtosis of the plot

- **Original:** noisy
- **Good:** retains “outlyingness”
- **Bad:** loses “outlyingness”
Constraint: Preserve deviations in plots

**Metric:** measure the *kurtosis* of the plot

\[
Kurt[X] = \frac{E[(X - \mu)^4]}{E[(X - \mu)^2]^2} \quad \rightarrow \quad \text{Fourth moment}
\]

\[
\Rightarrow \text{Variance}^2
\]

High kurtosis: heavy tails, outliers
Low kurtosis: light tails, uniform

![Plot 1](kurtosis = 4.1)

![Plot 2](kurtosis = 4.3 (> 4.1))

![Plot 3](kurtosis = 2.8 (< 4.1))

**Good:** retains “outlyingness”  **Bad:** loses “outlyingness”
**ASAP Recap**

**procedure:**

minimize point-to-point variance by adjusting window size while preserving kurtosis

\[ \hat{w} = \operatorname{argmin} \text{roughness}(Y) \text{ s.t. } \text{Kurt}[Y] \geq \text{Kurt}[X] \]
Try it yourself! ASAP.js

1) **Import:** Include JavaScript library

```html
<script src="ASAP.js" type="application/javascript"></script>
```

2) **Smooth:** Call smooth() before you plot

```javascript
Plotly.newPlot(graphDiv, layout [{
  x: time,
  y: data }]);
```

```javascript
Plotly.newPlot(graphDiv, layout [{
  x: time,
  y: smooth(data, pixels) }]);
```

http://futuredata.stanford.edu/asap/
User study: quantifying ASAP benefits

In which time period did a drop in taxi volume occur?

original

ASAP

28% 98.0

20.6 11.5

44%
User study: quantifying ASAP benefits

In which time period did a drop in taxi volume occur?

250 users, 5 datasets:
Accuracy: up to 38% increase (avg 21%)
Response time: up to 44% decrease (avg 24%)
Talk Outline

Motivation: Raw data is often noisy

Observation: smoothing helps highlight trends

Our research: smoothing automatically with ASAP
  Smoothing function: moving average
  Objective function: minimize point-to-point variance
  Constraint: preserve kurtosis of original data

Going fast: optimizations for fast rendering
Optimization: Exploit periodicity

Periodicity can be captured by autocorrelation

- Similarity between observations as a function of the time lag between them

\[
ACF(X, w) = \frac{E[(X_t - \mu)(X_{t+w} - \mu)]}{\sigma^2}
\]

We showed that

\[
\text{roughness}(Y) = \frac{\sqrt{2\sigma}}{w} \sqrt{1 - \frac{N}{N - w} ACF(X, w)}
\]

Only search for windows corresponding to high autocorrelation
Other optimizations

Limited pixels
• Pre-aggregate time series according to resolution

Update rate matters
• Even if data arrives quickly, don’t update faster than users can notice
This talk: how to smooth plots automatically

New research: more informative time series visualization

Big idea: smooth your visualizations!
this talk: how much to smooth?

Why smooth?
38% more accurate + 44% faster responses

Try it yourself:
JavaScript library ASAP.js

Demo, code and paper: http://futuredata.stanford.edu/asap/
Kexin Rong, kexinrong.github.io