

*Andrew Clegg*

---

# Signatures, Patterns & Trends

Timeseries data mining at Etsy

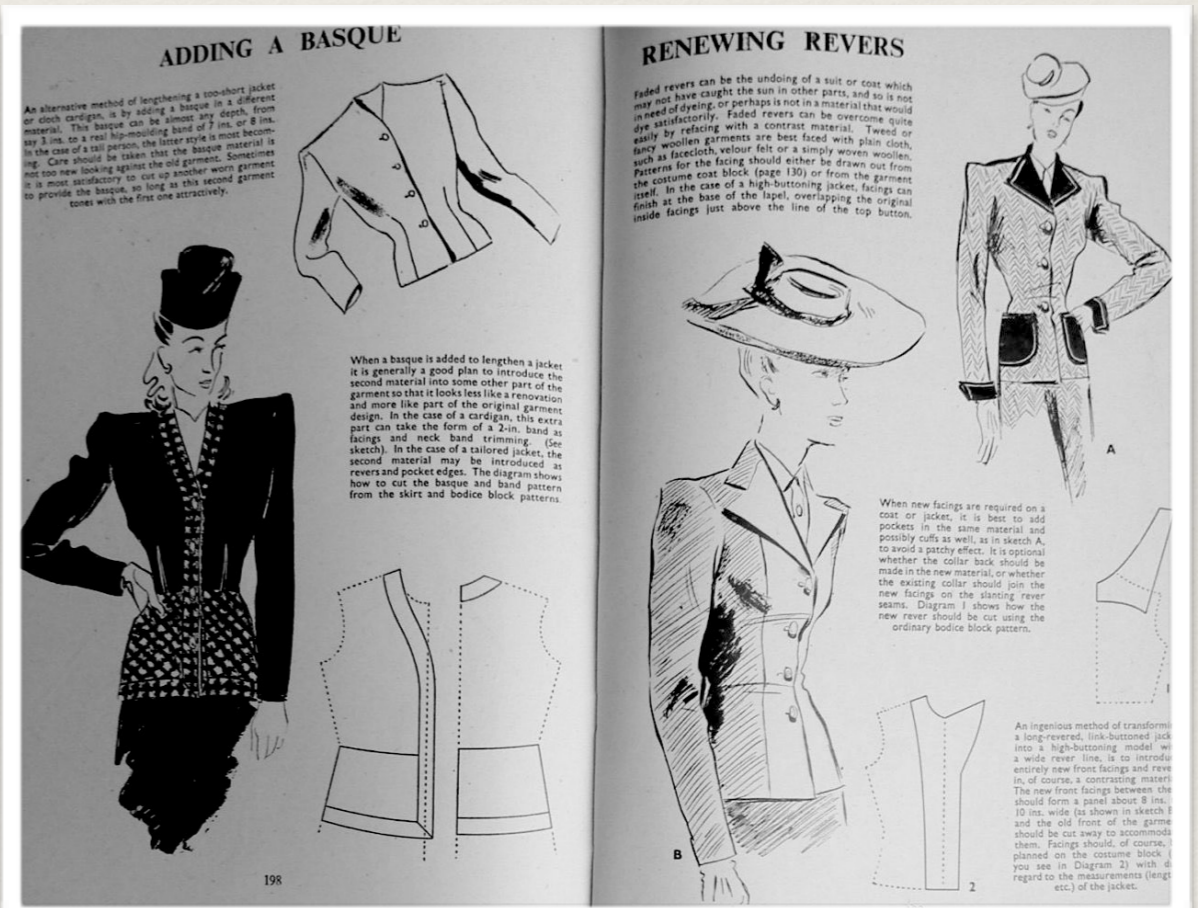
---



saturdayproject.etsy.com



xanthippew.etsy.com

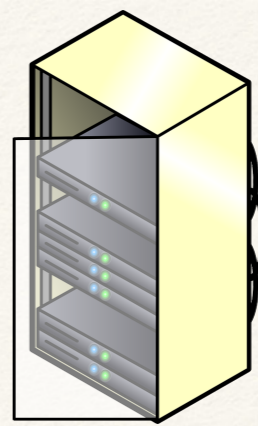


sewmuchfrippersy.etsy.com

# KALE

turns  
me on.

lotusleafcreations.etsy.com



*metrics*



*Skyline:  
anomaly detection*



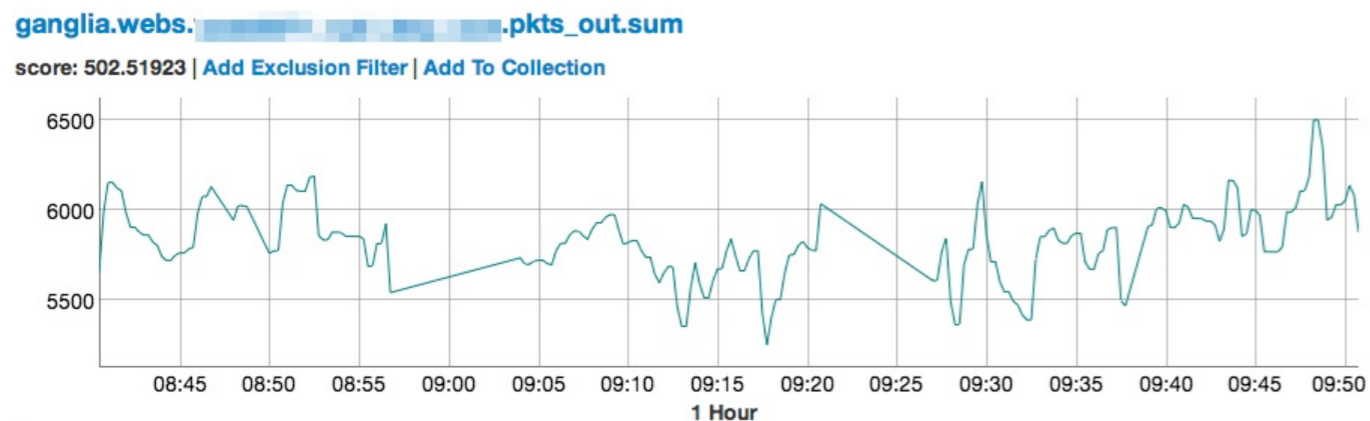
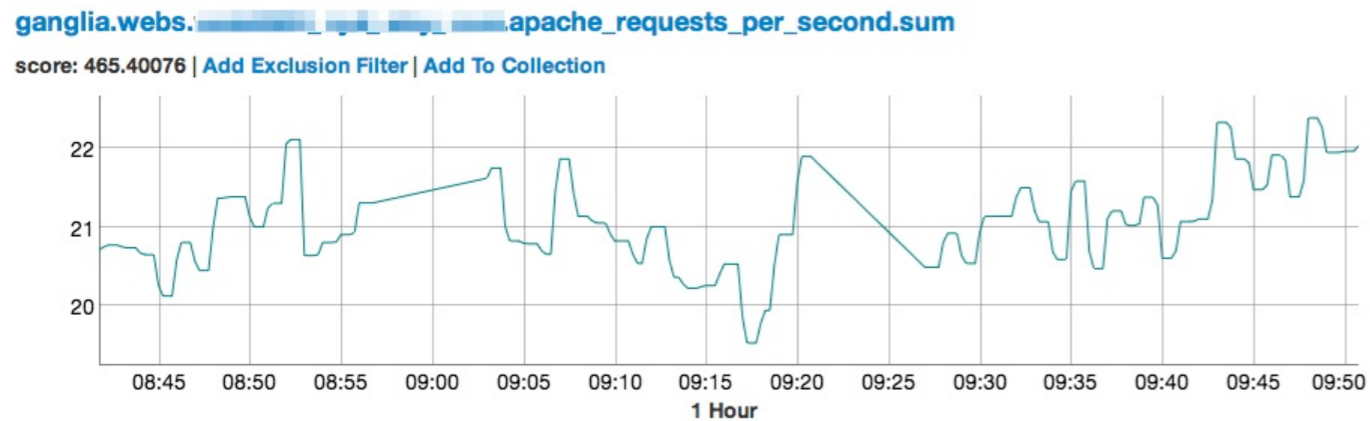
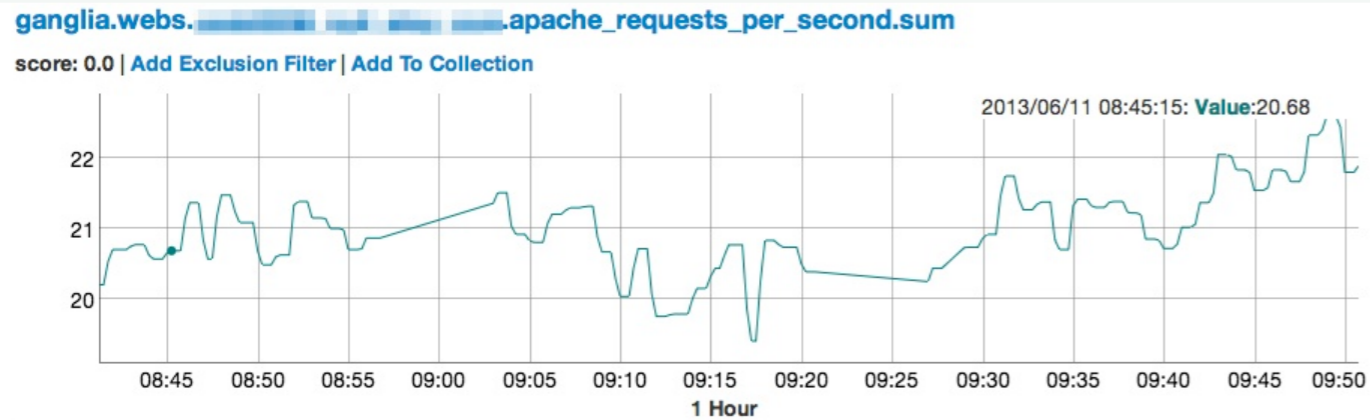
*Oculus:  
similarity search*

NOTHING  
*Easy*  
IS WORTH  
DOING




Kale 1.0: What worked well?


# Timeseries similarity search: 👍





## Shape Description Alphabet: a lovely hack.

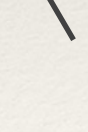
1. Map line segments to tokens based on gradient.
2. Index tokens with Elasticsearch.
3. Search for similar subsequences using sloppy phrase queries.

“a” 

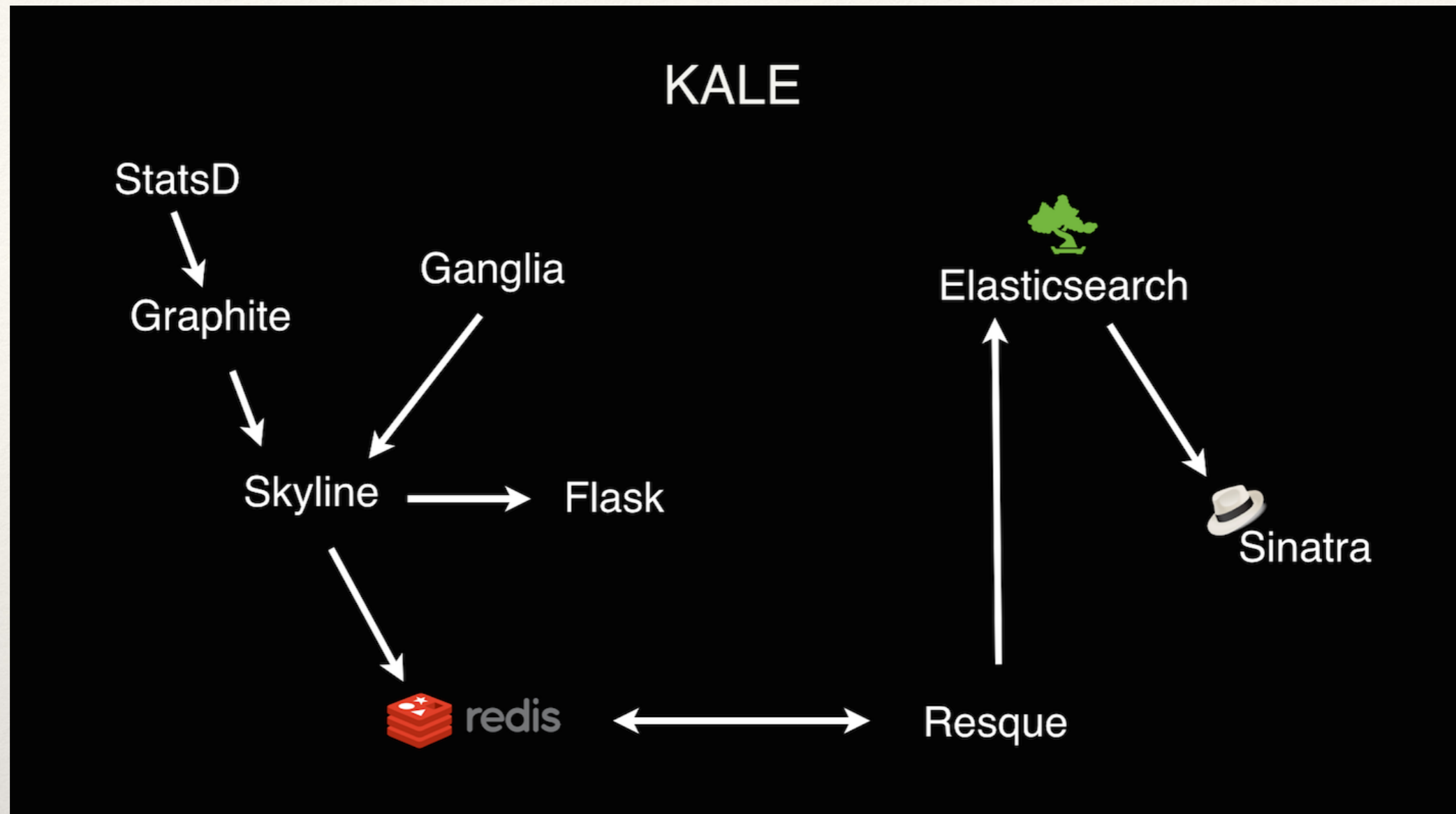
“b” 

“c” 

“d” 

“e” 

Kale 1.0: What proved hard?



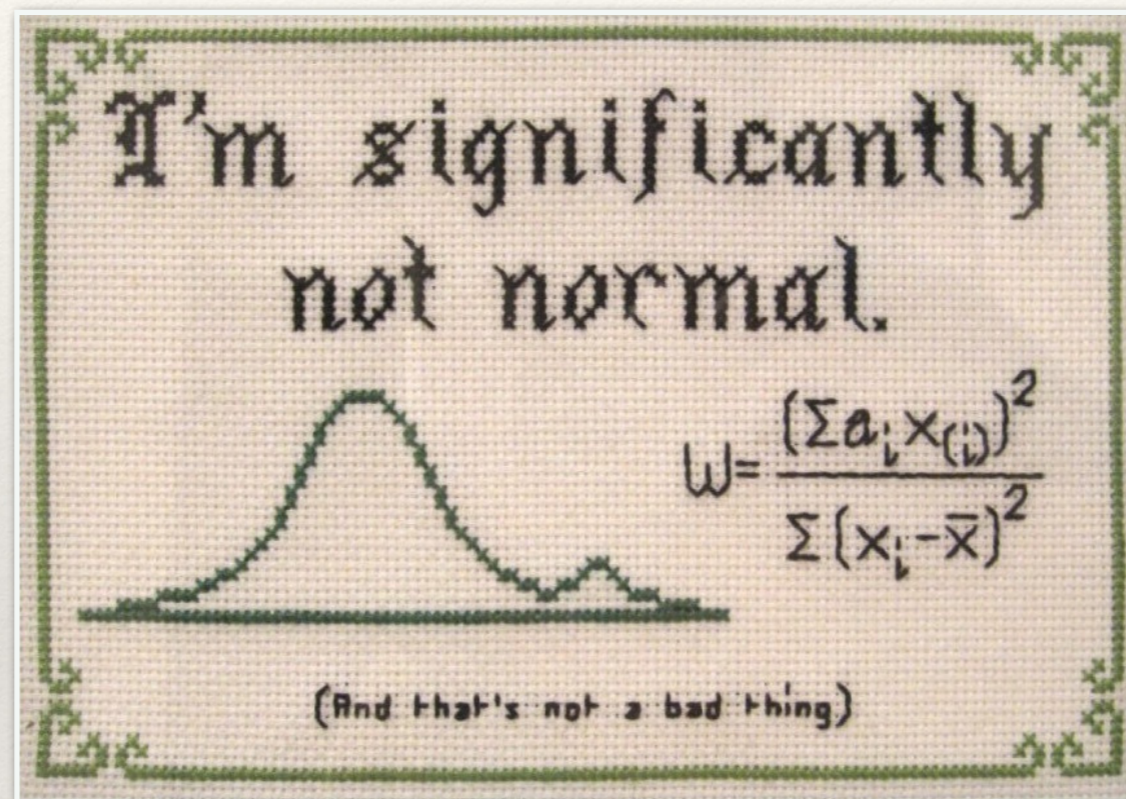
# Architecture

Languages used:

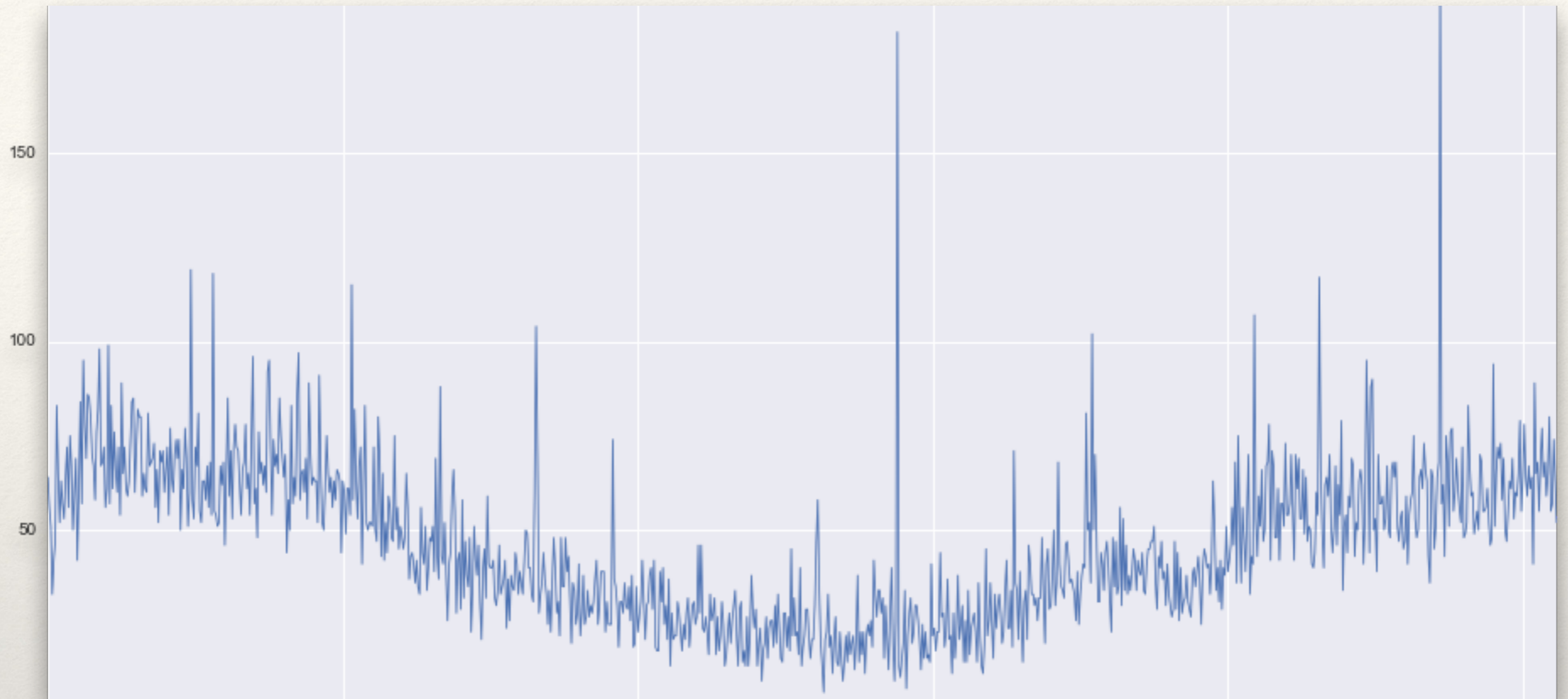
Python (Skyline app & workers)  
Ruby (Oculus search app)  
Java (Oculus ES plugin)  
JS (Skyline and Oculus apps)



# Anomaly detection

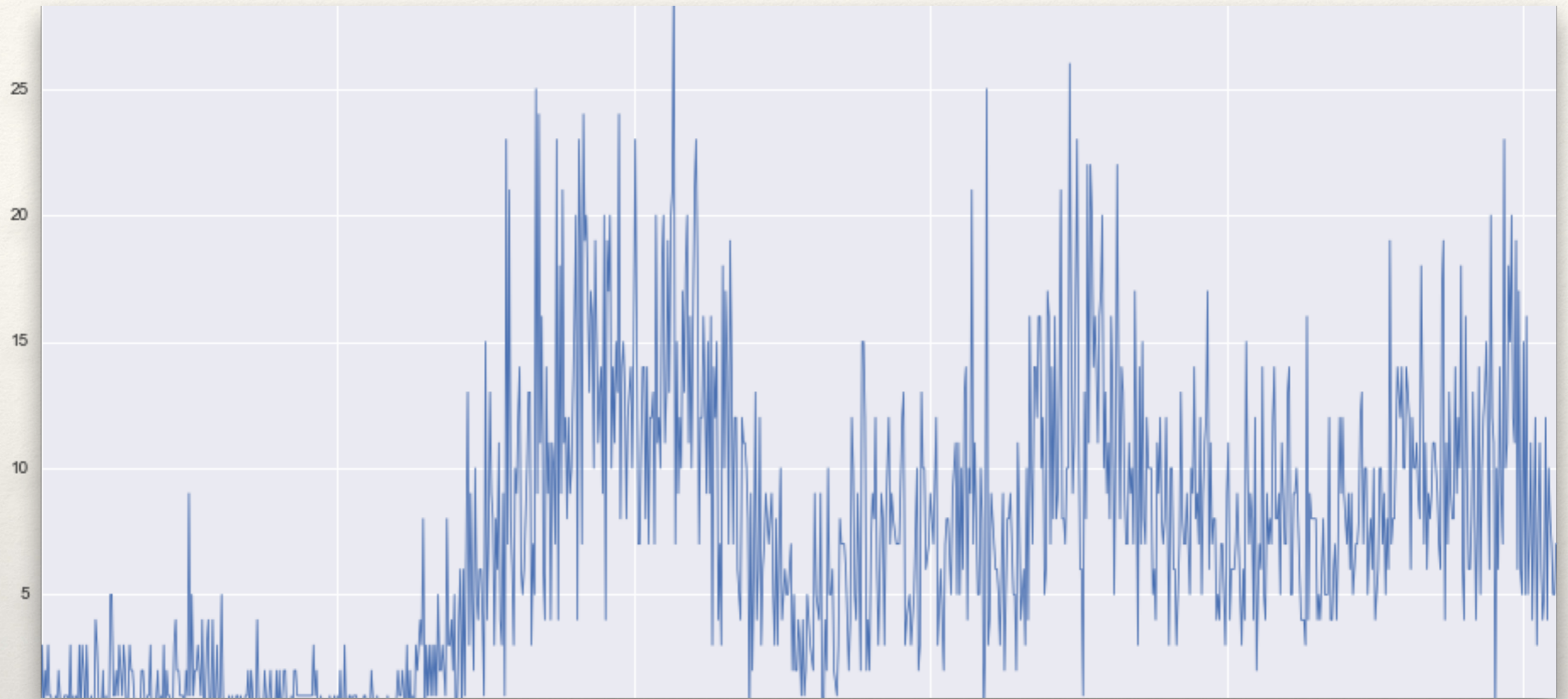


[nausicaadistribution.etsy.com](http://nausicaadistribution.etsy.com)



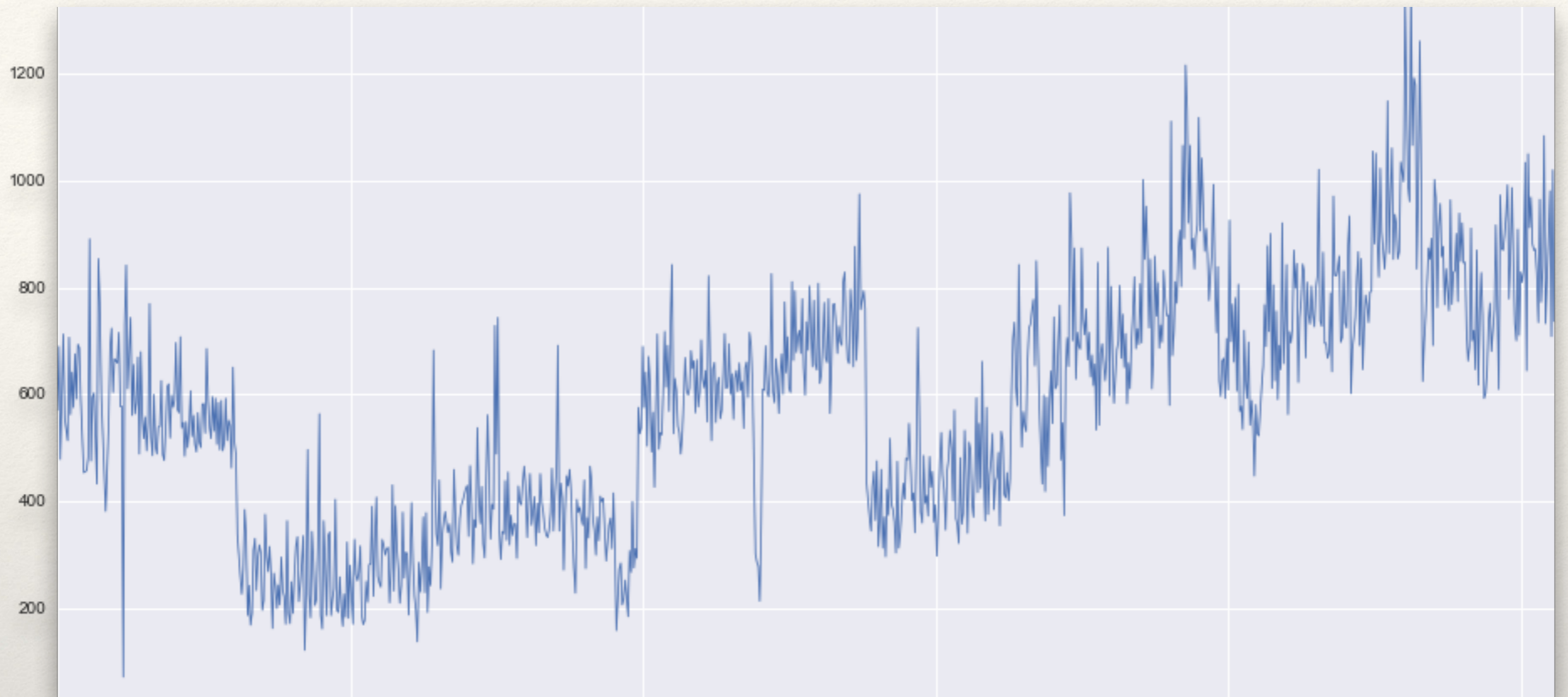
**Not every anomaly  
is a point outlier.**

And not every point outlier still  
looks like one, if you step back  
and look at more data.



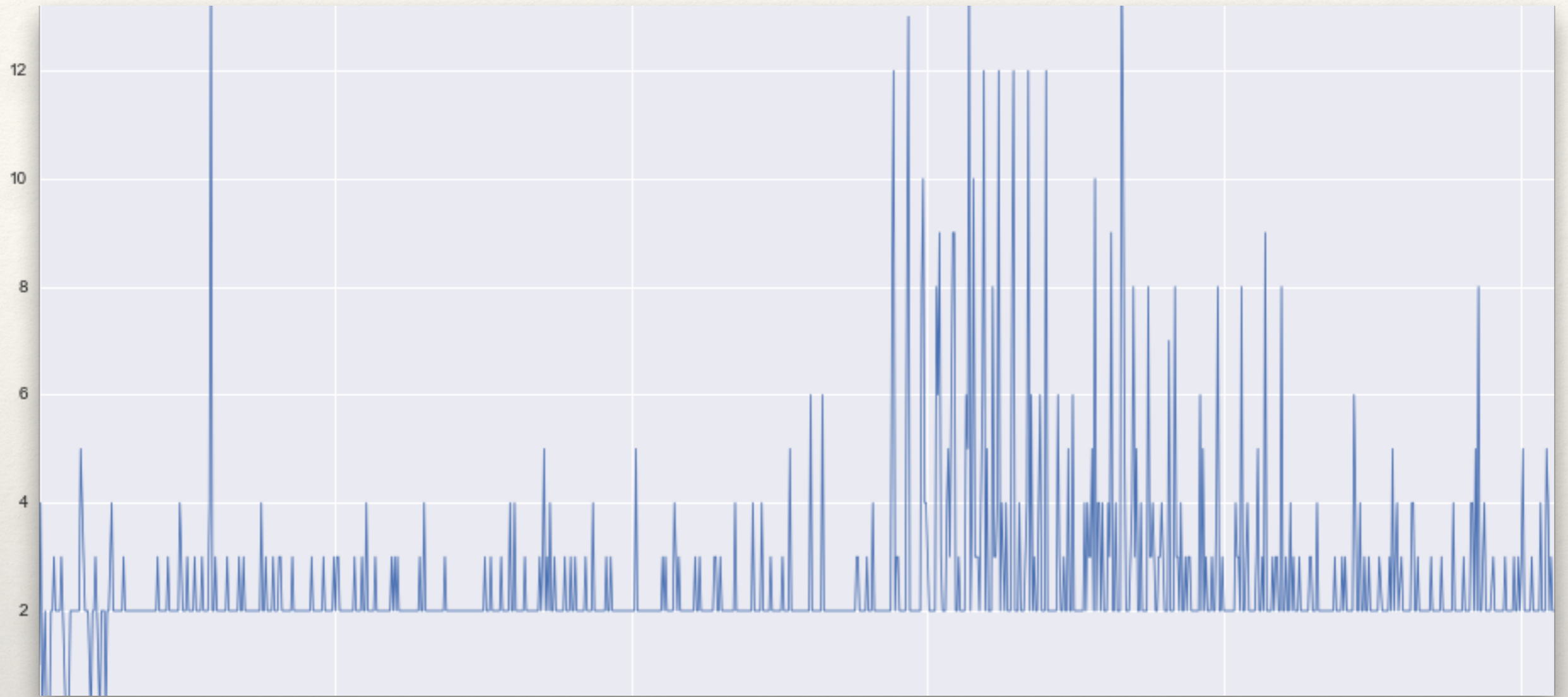
Periodic oscillations  
sometimes appear  
out of nowhere.

Or previously-reliable ones can  
vanish just as suddenly.



Trends change,  
baselines can  
suddenly shift.

Healthy upward growth can  
drop out suddenly, flatten off or  
begin to fall.



Rare, discrete events  
can suddenly become  
more frequent.

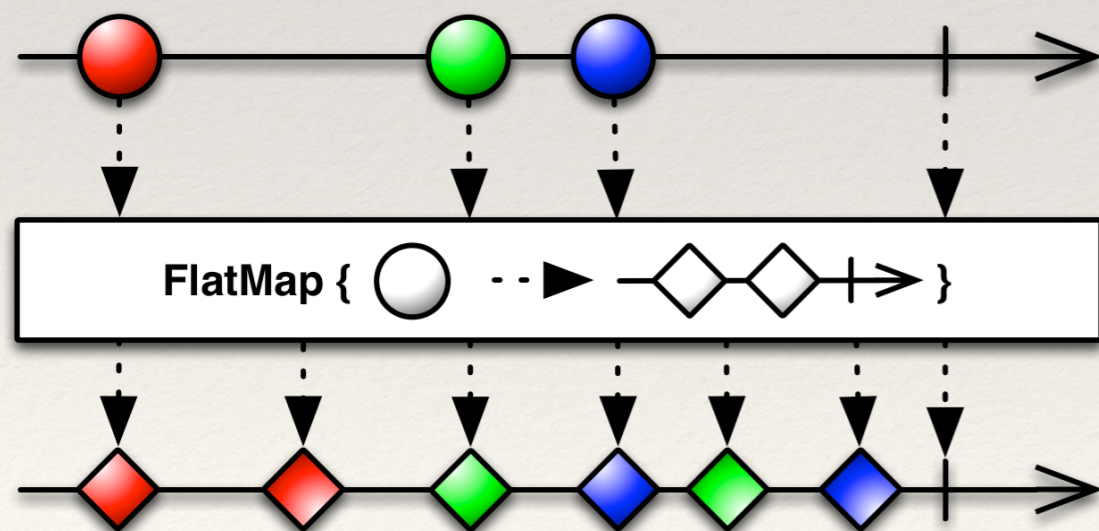
Conversely, events you expect  
to see with some regularity can  
become much sparser or  
disappear completely.

And the best bit is...

**... it's usually not even a  
problem.**

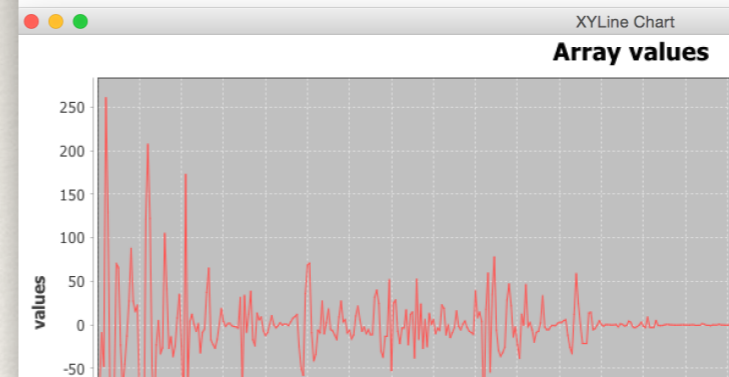
# Kale 2.0, Phase 1: **Thyme**

- ❖ Library of algorithms and composable processing steps
- ❖ Aims to be memory-efficient and cache-friendly (for Java)
- ❖ Platform and infrastructure agnostic
- ❖ Supports flexible experimentation and prototyping



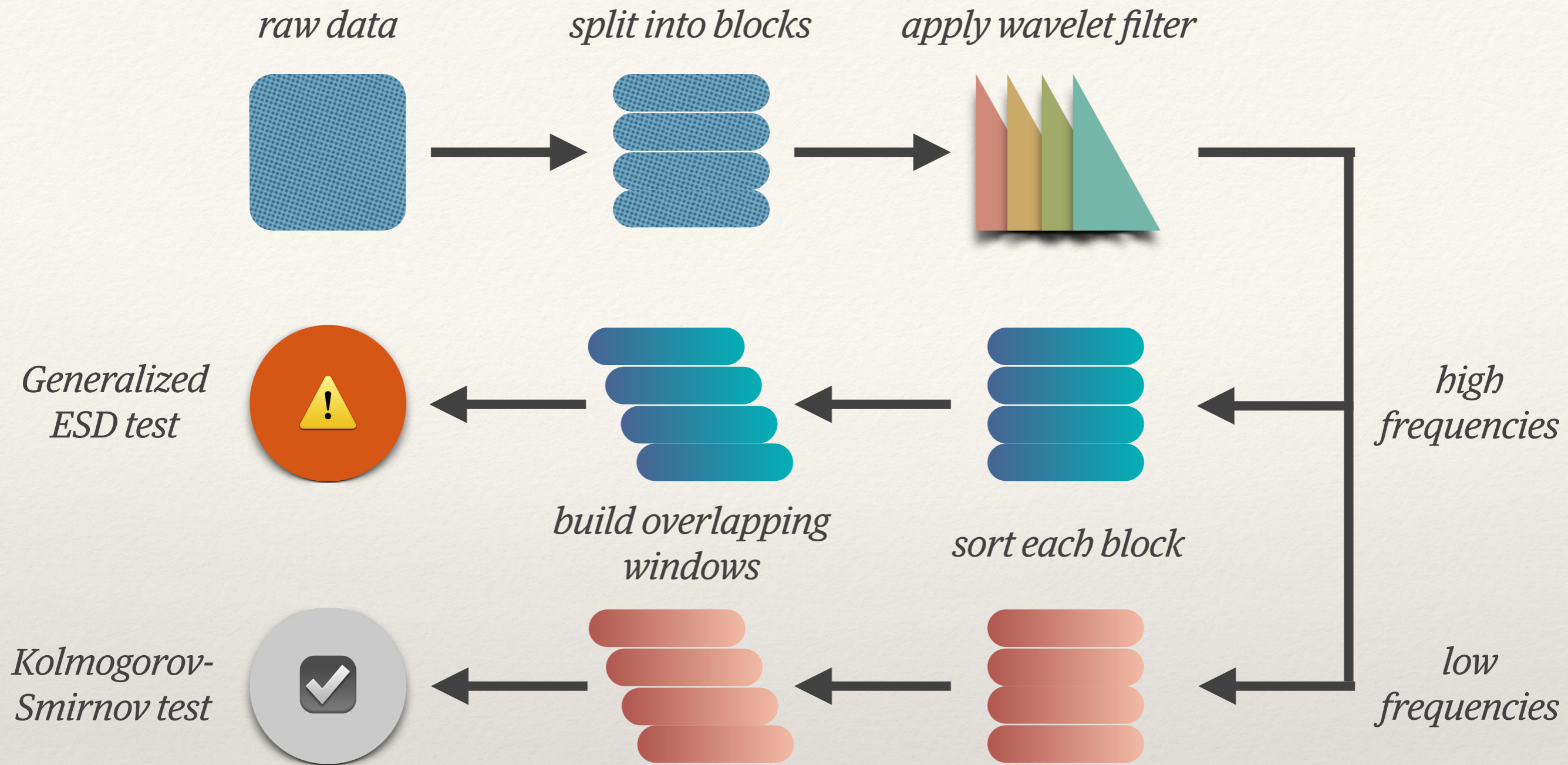
Built using **ReactiveX** framework

```
public Observable<Double> rms;  
  
public Microphone(final int period) throws LineUnavailableException {  
    final AudioFormat format = new AudioFormat(8000.0f, 16, 1, true, true);  
    final TargetDataLine mic = AudioSystem.getTargetDataLine(format);  
    mic.open(format);  
    mic.start();  
  
    final ByteBuffer audio = ByteBuffer.allocate(mic.getBufferSize());  
  
    final Observable<Long> timer = Observable.interval(period, TimeUnit.A  
        Schedulers.trampoline());  
}
```



Interactive sample  
application and  
developer tutorial





# A taste of Thyme

Schematic of a pipeline.  
The component parts can be assembled in various ways.

---

# Lessons learnt from Kale 1.0

---

- ❖ Keep your architecture simple – especially for OSS releases
- ❖ “This is a good fit for this problem” doesn’t always imply “Let’s add this new piece to our stack”
- ❖ Don’t release a product/platform unless you have a good history of using it yourself
  - ❖ ... and no plans to stop

---

# Lessons learnt from Kale 1.0

---

- ❖ Anomaly detection is more than just outlier detection
- ❖ A one-size-fits-all approach probably won't fit at all

---

# Lessons learnt from Kale 1.0

---

- ❖ Ensemble methods and auto-calibration a good idea
- ❖ Timeseries similarity search *is* feasible after all
  - ❖ ... if you constrain the search space by fingerprinting
- ❖ Don't forget human factors: good UI and workflow

@andrew clegg

---

# Thank you!

---

And thanks to everyone who's contributed:  
Abe Stanway, Avleen Vig, Jeff Kim, Jon  
Cowie, Katherine Daniels, Nishan Subedi

Follow us for updates:

<https://github.com/etsy>

<https://codeascraft.com/>

@codeascraft



"Thymus citriodorus" by Forest & Kim Starr  
Shared under CC-BY 3.0 license