

@andrew_clegg

Signatures, Patterns & Trends

Timeseries data mining at Etsy



saturdayproject.etsy.com



xanthippew.etsy.com



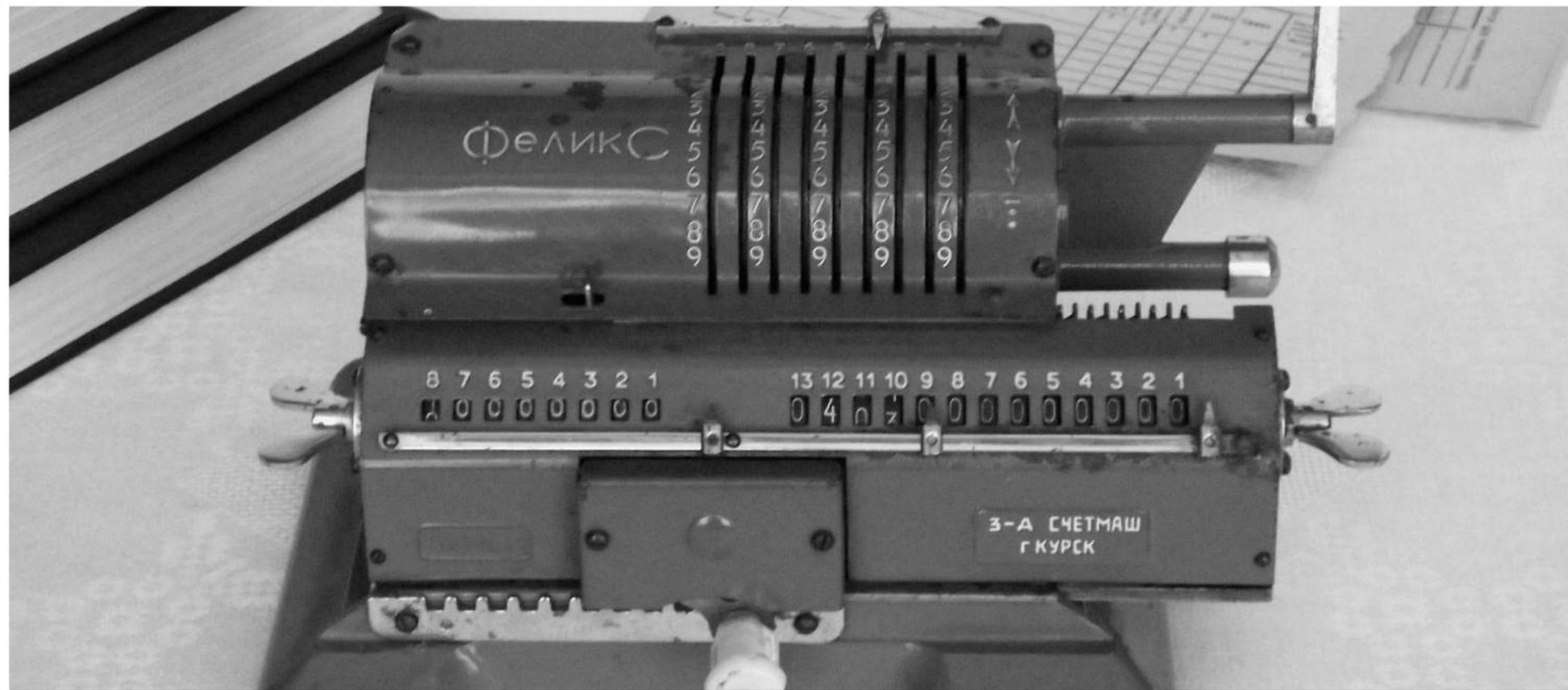
sewmuchfrrippery.etsy.com



Helping visitors to our marketplace find personally relevant items that they'll love

Data Science @ Etsy

- ❖ Shop/item/user recommendations
- ❖ Trending items and shops
- ❖ Data-driven ranking factors
 - ❖ e.g. for search, ads, navigation
- ❖ Text mining for search
 - ❖ autosuggest, related queries



Data mining tools to improve the efficiency, security, growth, operations and value of Etsy

Data Science @ Etsy

- ❖ Detecting stolen and watermarked images
- ❖ Classifiers for marketplace abuse and suspicious activity
- ❖ Customer lifetime value models
- ❖ **Detecting/diagnosing anomalies**

What's in this talk

Kale 1.0

- ❖ What it attempted, what succeeded, and what failed

More on anomaly detection

- ❖ Why it's hard, and why most tools over-simplify the issues

Kale 2.0 and Thyme

- ❖ What we're doing differently, and why
- ❖ Architecture overview
- ❖ ✨ 🍰 🎉 🎈 ≈ Live demo! ≈ 👨‍👩‍👧‍👦 🐱 💥 🚑

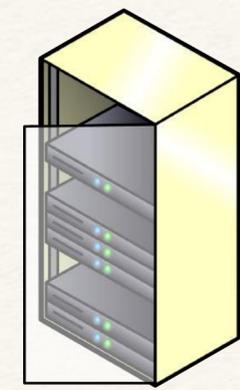
Lessons learnt

- ❖ Both technical and organizational

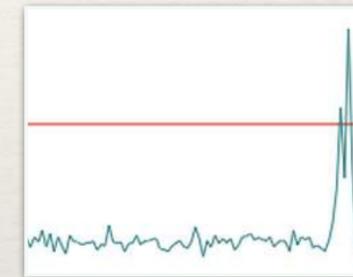
KALE

turns
me on.♥

lotusleafcreations.etsy.com



metrics



*Skyline:
anomaly detection*



*Oculus:
similarity search*

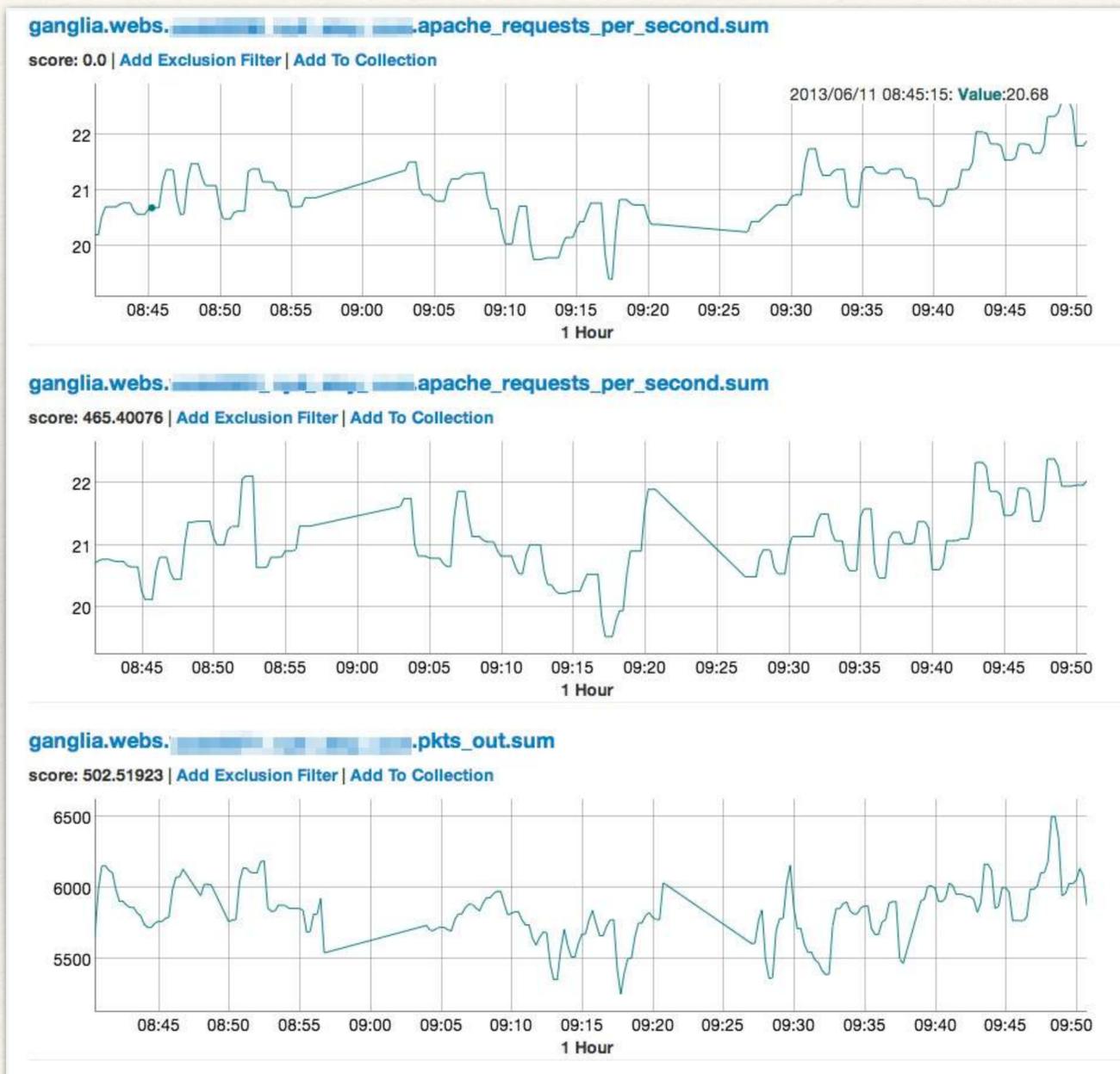
Kale 1.0
June 2013

NOTHING
Easy
IS WORTH
DOING



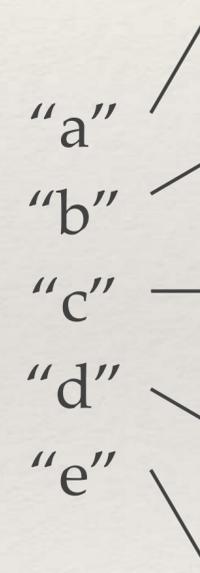
Kale 1.0: What worked well?

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.

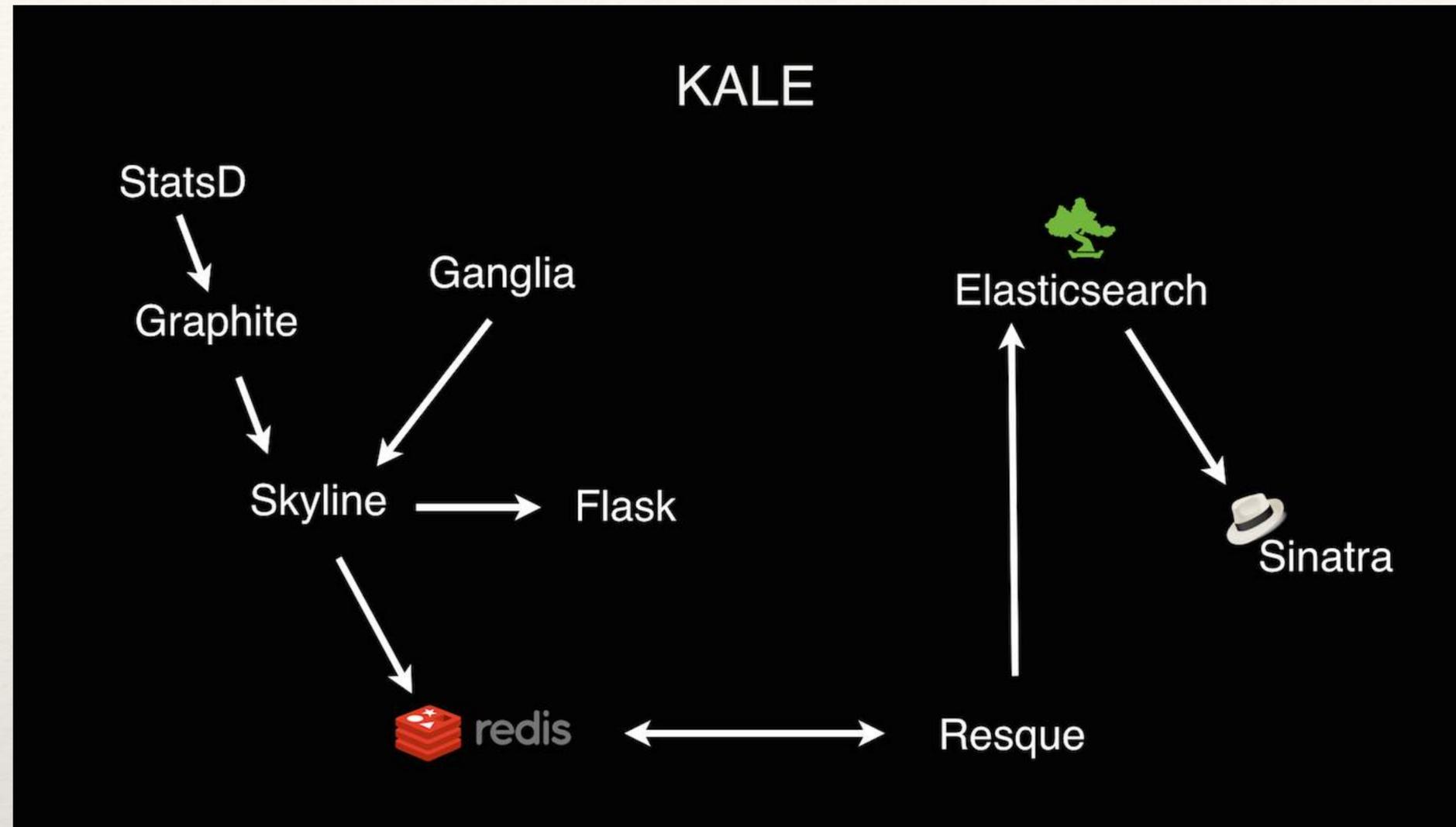


Then search these candidates exhaustively, using Fast Dynamic Time Warping.

Graphical user interface: 👍



Kale 1.0: What proved hard?



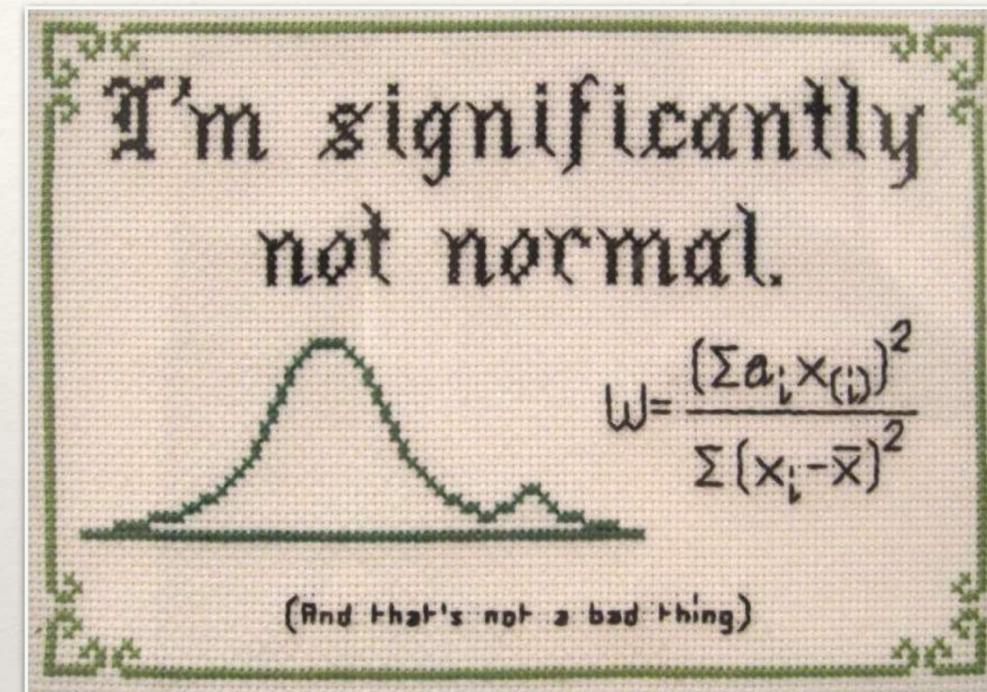
Architecture

Languages used:

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

Anomaly detection

- ❖ Non-normal distributions
- ❖ Spikes and periodic cycles
- ❖ 250,000+ metrics = inevitable false alarms
- ❖ Some mitigations helped a little:
 - ❖ Majority voting ensembles
 - ❖ Auto-silencing repeated alerts
- ❖ Too much effort focused on spotting *point outliers*
 - ❖ Single extreme values



nausicaadistribution.etsy.com



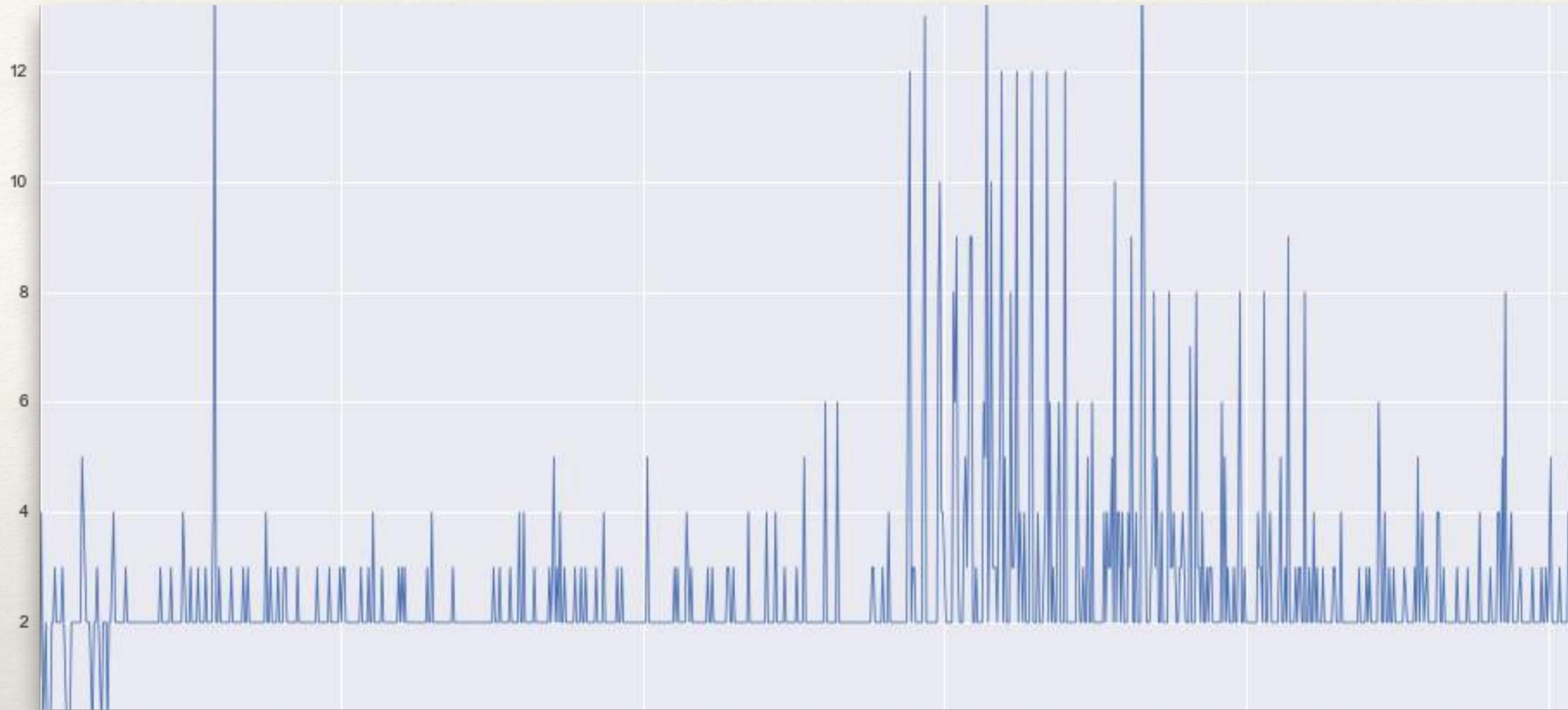
Not every anomaly
is a point outlier.

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



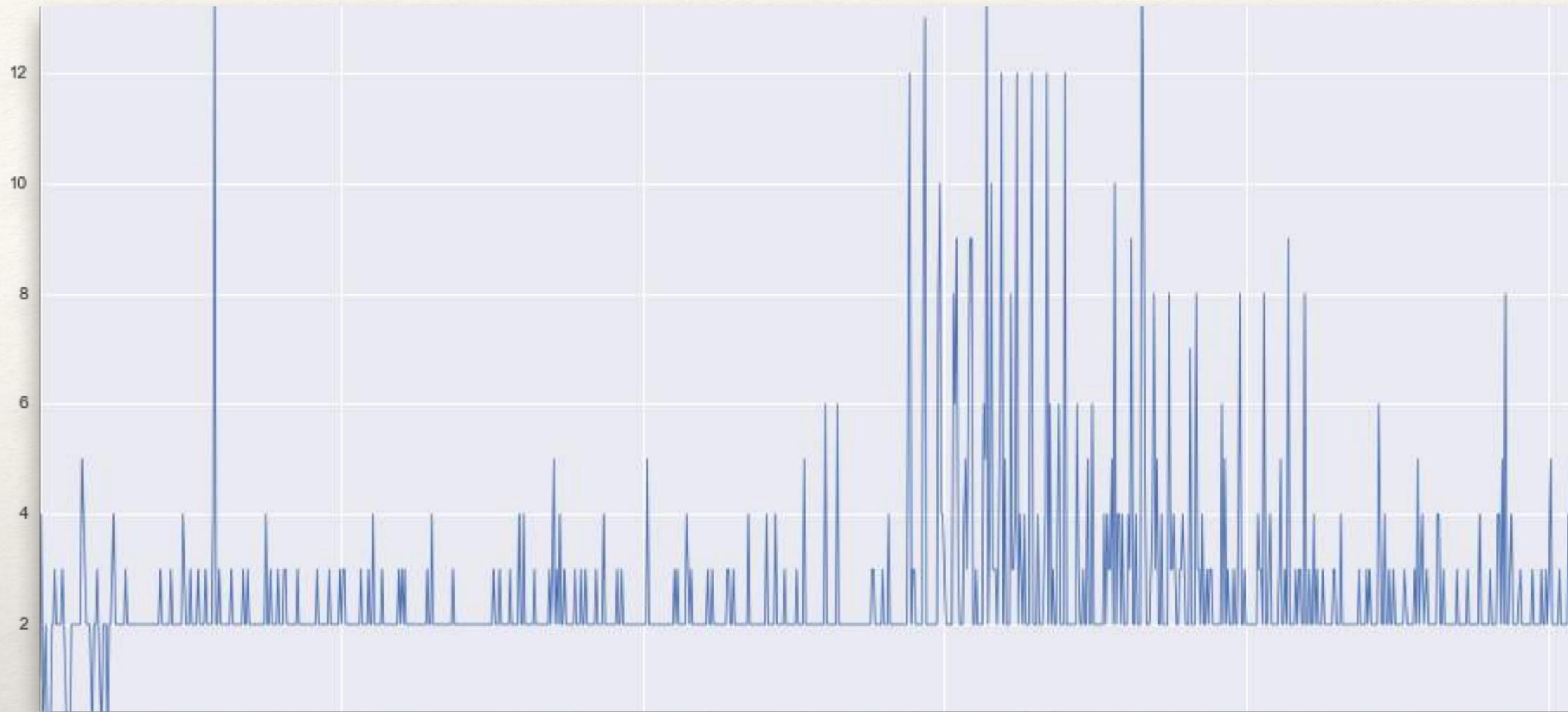
Periodic oscillations
sometimes appear
out of nowhere.

Or previously-reliable ones can
vanish just as suddenly.



Distributions change completely, within the same extremes.

Outlier detection is useless here, since no single value will be high enough (or low enough) to trigger an alarm.



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...

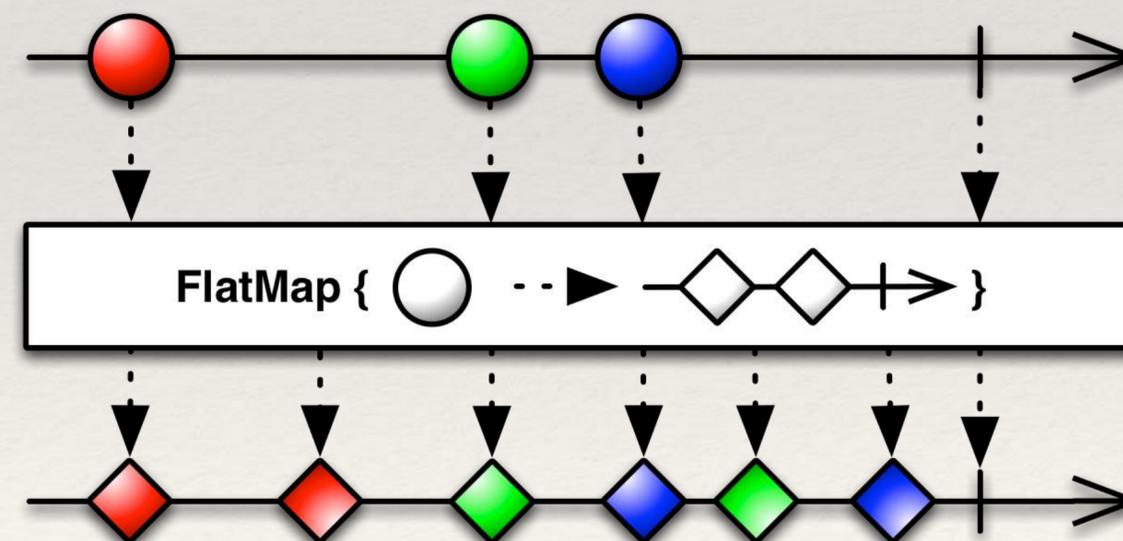
**... it's usually not even
a problem.**

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 on **ReactiveX** framework

- ❖ Stream-driven functional API
- ❖ Originally from .NET world
- ❖ **RxJava**: a Netflix Original



Components for first release



- ❖ **Signal processing and feature extraction**

- ❖ Centering, scaling, smoothing and detrending
- ❖ Wavelet decomposition and filtering
- ❖ Discrete Fourier and cosine (FFT/DCT)



- ❖ **Statistical anomaly tests and ensemble strategies**

- ❖ Generalized ESD
- ❖ Kolmogorov-Smirnov
- ❖ Mann-Whitney



- ❖ **Fingerprinting and search**

- ❖ Shape Description Alphabet
- ❖ Binary clipping
- ❖ Fast Dynamic Time Warping (DTW)



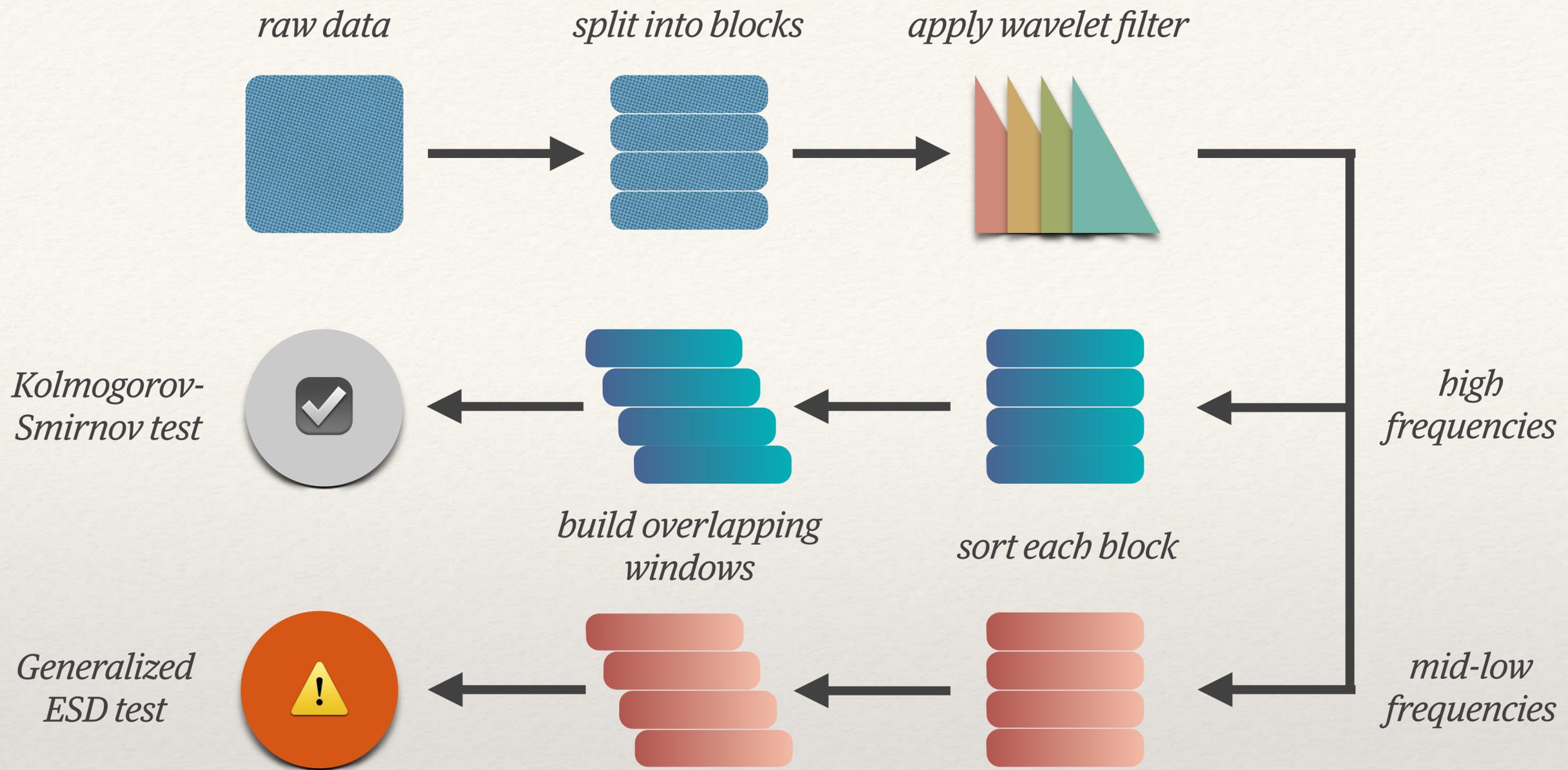
- ❖ **Interactive demo app**

Design choices: why Java?

- ❖ Interop with current and next-gen Etsy platforms:
 - ❖ ELK stack, Kafka + Samza, Spark [Streaming]
 - ❖ Hadoop + Scalding + Conjecture
 - ❖ Android 
- ❖ Plugins for OpenTSDB, DataStax etc. should be easy too
- ❖ Pretty good maths/stats resources to draw on
- ❖ **Why not Scala?** Java more widespread, and generally faster

Design choices: ReactiveX!?

- ❖ *Functional Reactive Programming* pattern is great for data flows
- ❖ Most components are either instances of:
 - ❖ `Func1<double[], double[]>` → preprocessing operators
 - ❖ `Func1<double[], T>` → statistical tests with output type T
- ❖ An `Observable<double[]>` + a series of functions = whole pipeline
 - ❖ RxJava takes care of the plumbing and orchestration
- ❖ Idiomatic DSLs for Java 8, Scala, Clojure, Groovy and Kotlin
 - ❖ Great for third-party extensions and plugins



A taste of Thyme

Simplified schematic of one possible pipeline, using just a handful of Thyme components.

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 open-source a platform or application unless you’ve been running it in production for a while
 - ❖ ... and have 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
- ❖ Not all anomalies should result in pager or email alerts
- ❖ Possible approach:
 - ❖ Alert on anomalies in business and user metrics only
 - ❖ Use anomalies in other metrics for root cause analysis

Lessons learnt from Kale 1.0

- ❖ Good UI and workflow will be vital for production usage
 - ❖ Kale 1.0 was a victim of its own success in this respect!
- ❖ Ensemble methods and auto-calibration a good idea
- ❖ Timeseries similarity search *is* feasible after all
 - ❖ ... if you constrain the search space by fingerprinting

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

Thank you!

First release Q3 – follow us for news:

[@andrew clegg](#)

<https://github.com/etsy>

<https://codeascraft.com/>

[@codeascraft](#)



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