Getting to Maybe:

Making the case for changes
USRSE Community Call

Andy Boughton

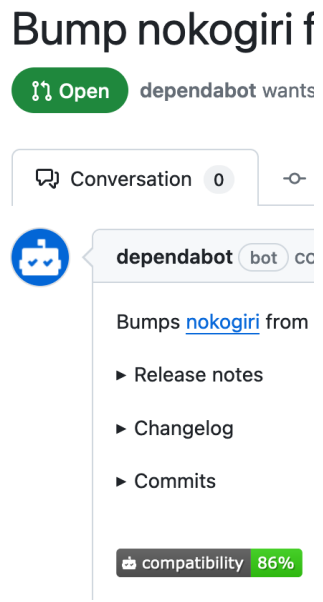
We all have tech debt

How do you convince your manager to spend the resources to change things?

Suggestion 1: Do you have requirements?

- You may be affected by rules without knowing it!
  - ITAR, HIPAA, GDPR, IRB agreements, NIH CoC, SLAs, contract timelines....
  - Organizational policies
- These may define your maintenance schedule
- They will determine your end-of-life schedule
  - Protip: your vendors have end-of-life schedules too!

Security means maintenance



- Your institution/ grant / funders may have explicit mandatory timelines
  - Public systems may have more specific timelines
- Use automated tools to keep up
  - Dependabot, snyk, npm audit...
  - Reproducible deployments (like containers) to make releases safer

Remediation Timeframes		
Risk Level	**Public-Facing**	**Internal**
Critical	15 days	30 days
High	30 days	60 days
Medium	60 days	90 days
Low	6 months	6 months

Sample NIH policy

Priority Level	Action Plan By	Resolved By
Critical (CVSS 9-10)	2 weeks	1 month
High (CVSS 7-8.9)	1 month	3 months

https://it.umich.edu/information-technology-policies/general-policies/DS-21

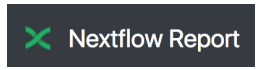
Suggestion 2: Chores are opportunities

- Federal procurement standards require considering accessibility of tools (sect 508, WCAG, etc)
  - This can be a good chance to incorporate automated tools into your dev process and drive improvements going forward
- Tie internal goals to external features
  - “We’ve always wanted to x and it will make y easier”
  - Give downstream users a reason to upgrade

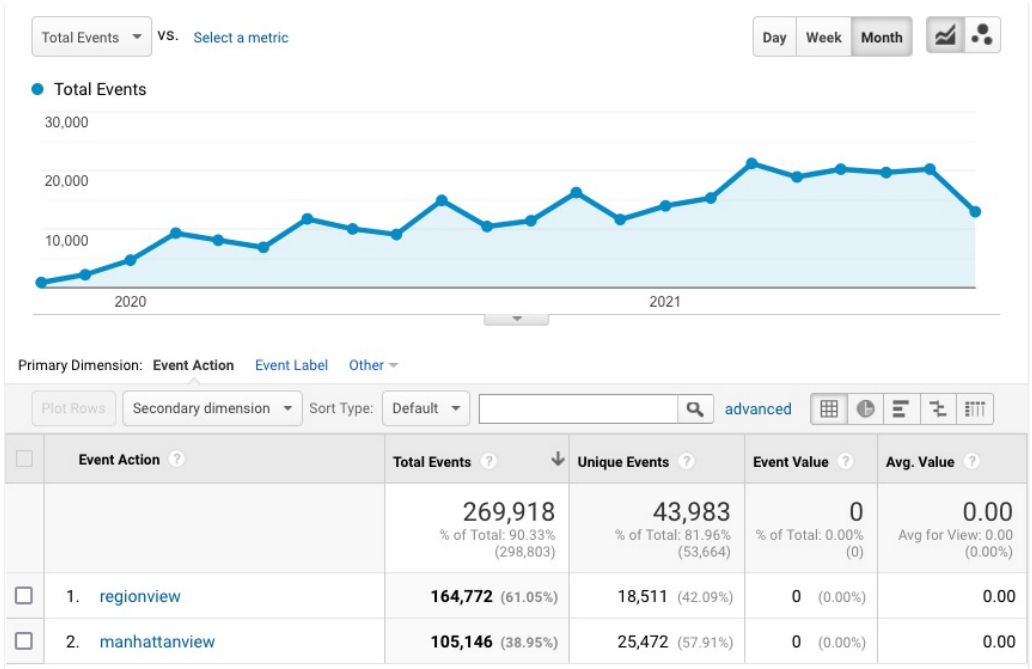
https://github.com/dequelabs/axe-core

Suggestion 3: Measure now, decide later

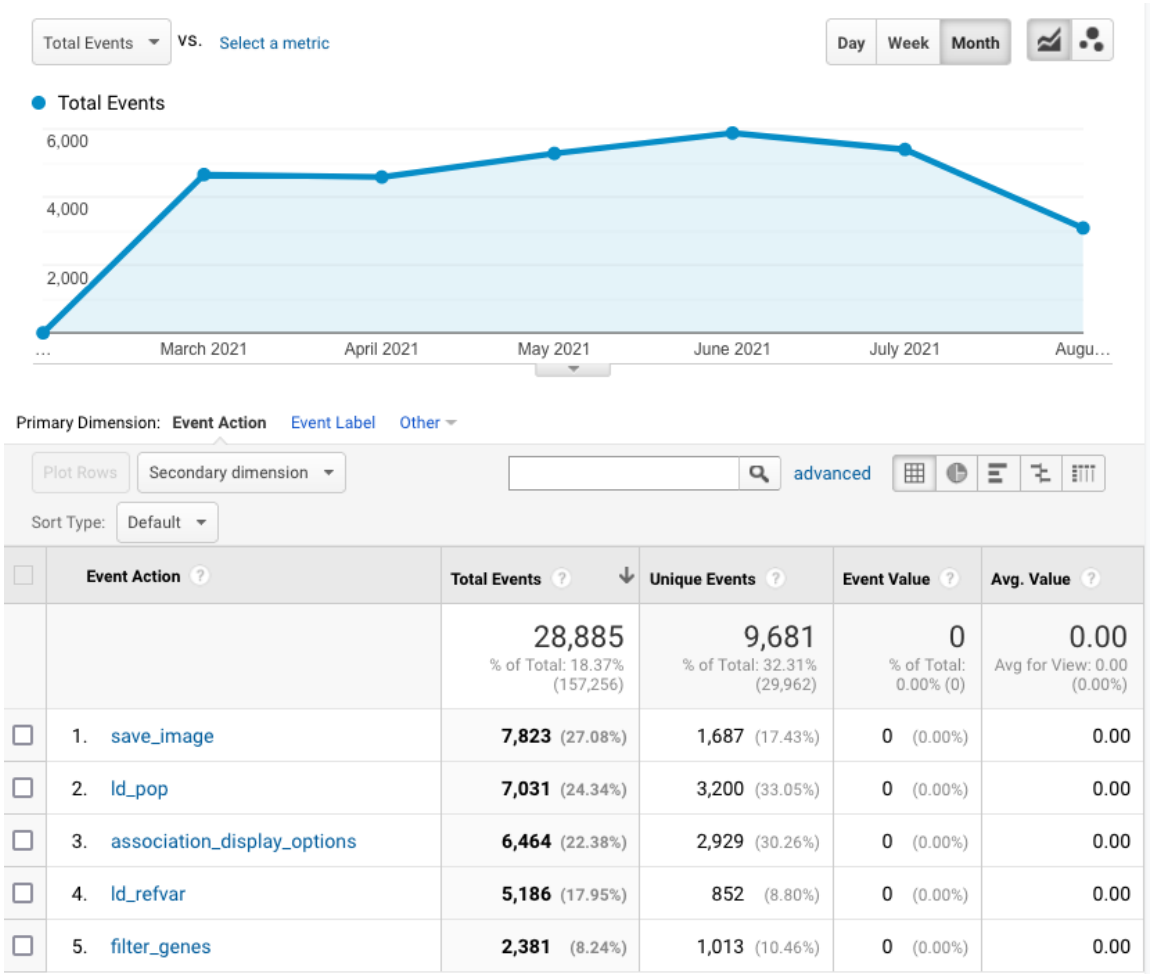
- Find a way to track ongoing usage
  - Analytics for websites (matomo, Google analytics, Goaccess, etc)
  - For CLIs: Download counts for plugins/assets/datasets (find a place where users engage with your infra)
- Use what your metrics provide
  - Automated error reports help make the case for refactoring
  - Performance/runtime metrics make the case for optimization
  - Feature level analytics make the case for deprecations
- Don’t send reports from secure systems; people **will** get cranky



Beyond Pageviews: analytics by feature/ event



Event Action	Total Events	Unique Events	Event Value	Avg. Value
	269,918 % of Total: 90.33% (298,803)	43,983 % of Total: 81.96% (53,664)	0 % of Total: 0.00% (0)	0.00 Avg for View: 0.00 (0.00%)
1. regionview	164,772 (61.05%)	18,511 (42.09%)	0 (0.00%)	0.00
2. manhattanview	105,146 (38.95%)	25,472 (57.91%)	0 (0.00%)	0.00



Event Action	Total Events	Unique Events	Event Value	Avg. Value
	28,885 % of Total: 18.37% (157,256)	9,681 % of Total: 32.31% (29,962)	0 % of Total: 0.00% (0)	0.00 Avg for View: 0.00 (0.00%)
1. save_image	7,823 (27.08%)	1,687 (17.43%)	0 (0.00%)	0.00
2. ld_pop	7,031 (24.34%)	3,200 (33.05%)	0 (0.00%)	0.00
3. association_display_options	6,464 (22.38%)	2,929 (30.26%)	0 (0.00%)	0.00
4. ld_refvar	5,186 (17.95%)	852 (8.80%)	0 (0.00%)	0.00
5. filter_genes	2,381 (8.24%)	1,013 (10.46%)	0 (0.00%)	0.00

Example: ~10% of region plots are saved for followup

Avg 10 region plots generated for every file uploaded

Suggestion 4: Define victory

- Who can issue updates?
- When are you willing to stop development?
  - What happens if your vendor stops development first?
- Identify path to usage when your support ends
  - What components need to be flexible to keep the tool running?
  - Eg data retrieval, type of compute environment