

Tab 1

LOCAL NEWS MATTERS

BAY AREA

Bay City News has been the backbone of Bay Area journalism since 1979, delivering 24/7 reporting across 13 counties. Our hybrid model includes a commercial newswire for media and business subscribers and a nonprofit arm that publishes public service journalism on Local News Matters and The Mendocino Voice.



Ciara Zavala

Ciara Zavala, now the digital editor at BCNF, was a Dow Jones News Fund fellow in 2023 and 2024 before joining as an intern last summer. She has experience in digital media, audience engagement, web design, and analytics through internships at The Greater Sacramento Economic Council, The Colorado Sun and Fox 8.

Katherine Ann Rowlands, president of Bay City News Service and founder of its nonprofit arm, has been a journalist for 40 years. After a JSK Fellowship at Stanford, she bought Bay City News in 2018. She co-founded WomenDoNews and leads the First Amendment Coalition board. As a Berkeley native, she started as a Bay City News intern.



Kat Rowlands

A Playbook for Newsrooms: Revolutionizing Election Coverage with AI Part II

This is the second installment of the election playbook that we are iterating upon as we learn and as tools develop. Here's a link to the [first playbook](#) published in 2024.

Our Goal

In the fast-evolving landscape of AI, we saw an [opportunity to revolutionize local election coverage in our newsroom](#) by reducing manual, repetitive tasks so our journalists could focus on in-depth reporting and storytelling. Election season is our busiest time of year, and methodically copying and pasting candidate names and results from dozens of county, city and school board races was not an efficient use of our team's time. We sought to use AI-powered data scraping and automation to streamline our workflow and provide real-time local election results to our readers -- something not provided by AP, Ballotpedia or other large services focused only on top-of-ticket races and legislative contests. Our [Election Hub](#) was the result.

1. [Revisiting the Problem](#): Why election data remains difficult to automate and what changed this cycle
2. [What We Learned from the 2024 Election Cycle](#): A quick recap of what worked, what didn't and why it mattered
3. [Exploring New Paths Forward](#): Partnerships, platforms and why collaborations fell short
4. [Rethinking the Approach & Standardizing the Input](#): Using screenshots and automation to force consistency
5. [Where the Automation Broke Down](#): Why AI interpretation introduced inconsistency
6. [Returning to Traditional Scraping](#): Using AI to build the tool, not touch the data
7. [Creating this workflow for your newsroom's use](#): A guide to what features are, what you need to feed the bot and tips for better outcomes.
8. [Suggested query to replicate in your newsroom](#): Feed this prompt into Claude after you open it in the terminal.

9. **Step-by-Step Guide:** Now that you have an understanding of the platforms and how to interact with code, follow these steps.
10. **Conclusion:** What this means for future election cycles

1. Revisiting the Problem

Why election data remains difficult to automate and what changed this cycle

In the 2026 election cycle, we set out to iterate on our previous election project, which aimed to automate the collection of election data and results into a single, centralized destination for our audience across 13 counties. This challenge is not unique to our newsroom. News organizations of all sizes grapple with the same problem each election cycle because there is no easily standardized pipeline for pushing results from county registrar websites directly to a newsroom's website. Registrars are also not held accountable to a deadline to post zero reports, which are templates for the way they plan to present results on election night and are extremely vital for testing scrapers and different codes.

With the growing emergence of AI as a tool for journalists, we saw an opportunity to revisit this problem through an experimental lens. But the question wasn't whether to use AI, it was how. There are countless ways AI can be applied to workflows, and part of this project involved testing multiple approaches before arriving at a final methodology. The 2025 cycle became less about finding a single perfect solution and more about understanding the limits, tradeoffs and practical applications of AI in real-world election coverage.

2. What We Learned from the 2024 Election Cycle

A quick recap of what worked, what didn't and why it mattered

During the 2024 election cycle, our newsroom piloted an AI-assisted workflow designed to reduce the manual labor involved in election coverage and redirect staff time toward reporting and analysis. The project focused on two core goals: streamlining ballot data collection and building a scalable system to display election results across 13 counties in a clear, accessible format.

What worked

- **AI-assisted ballot previews:** We successfully used AI to extract candidate and ballot measure information from county registrar PDFs and webpages and convert it into structured spreadsheets. This significantly reduced manual data entry and set the foundation of our public-facing election dashboard. It also allowed our audience from all 13 counties to preview ballot information ahead of voting to help them assess local candidates and measures.

- **HTML generation without a data team:** By using AI-generated spreadsheet formulas to produce custom HTML layouts, we were able to build election result pages that mirrored official formats without hiring a developer or building custom software.
- **All-in-one dashboard:** We created a central hub that equipped our readers with everything they needed to navigate the election process across all 13 counties we cover in the greater Bay Area. It answered voter questions and provided key deadlines, polling locations and ballot details to ensure every voter had the information they needed at their fingertips.

What didn't work

- **Real-time election night automation:** County registrar websites varied widely in structure, update frequency and reporting systems, making it difficult to deploy a single, reliable scraping solution across all 13 counties.
- **Lack of access to “zero ballots,” the templates for election night results:** Obtaining zero ballots from county registrar offices proved inconsistent and, in some cases, impossible. Some counties released them just days or hours before polls closed, while others did not release them at all. This limited our ability to test scrapers in advance and left little room for troubleshooting under live conditions.
- **Standardization barriers:** Changes in candidate order, layout shifts and inconsistent data presentation caused AI tools and scrapers to break during live updates.

Key takeaway

Human oversight remained essential. Even with automation in place, live election results required constant monitoring, verification and manual intervention on election night. AI could assist with preparation and structure, but it could not fully account for the unpredictability of live reporting environments, from last-minute website changes to delayed or missing data from official sources. Rather than replacing editorial judgment, AI introduced a new layer of responsibility: understanding when to trust automation, when to intervene and how to verify results quickly without sacrificing accuracy. This reality emphasized the need for workflows that balance efficiency with quality control, especially during high-stakes moments like election night when we cannot afford to be wrong.

3. Exploring New Paths Forward

Partnerships, platforms and why collaborations fell short

At the beginning of planning how we would approach this project differently in 2025, we revisited several leads from our 2024 experiment that we believed could lead to a better solution.

One promising avenue was a previous conversation with the election data team at The Press Democrat, who had successfully built custom scrapers tailored to the specific formats (html, csv and pdf) used by each county they cover. We explored the possibility of collaborating with their team to replicate a similar approach at a larger scale. However, that option ultimately fell off the table when The Press Democrat was acquired by Bay Area News Group, and the data team was reassigned within the larger organization.

We also revisited discussions with the Associated Press to explore whether they might develop a workflow to distribute local election data directly to newsrooms. While the AP collects comprehensive election results, their process prioritizes top-of-ticket national and state races and only selective, high-profile local races. Unfortunately, they did not have the capacity to build or support a pipeline that would deliver down-ballot election data to local news newsrooms.

Another potential avenue involved Civera, an election data systems company we connected with during the previous election cycle. While they expressed interest in the project, they ultimately did not have the capacity to collaborate with us during the 2025 cycle. However, they shared that they plan to invest more time and resources into developing a real-time election data strategy in 2026.

So we were back to bootstrapping a solution on our own.

4. Rethinking the Approach & Standardizing the Input

Using screenshots and automation to force consistency

As potential collaborations with other organizations and publications fell off the table, we shifted our focus to rethinking how we would approach this election cycle just with our internal team. We kept the challenges and limitations from the 2024 project top of mind, particularly around data access and standardization. Our initial instinct was to move away from scrapers altogether and reconsider how election data could be collected. This led us to test an entirely different approach. Because the lack of standardization across county election websites proved to be the biggest obstacle last year, we asked a simple question:

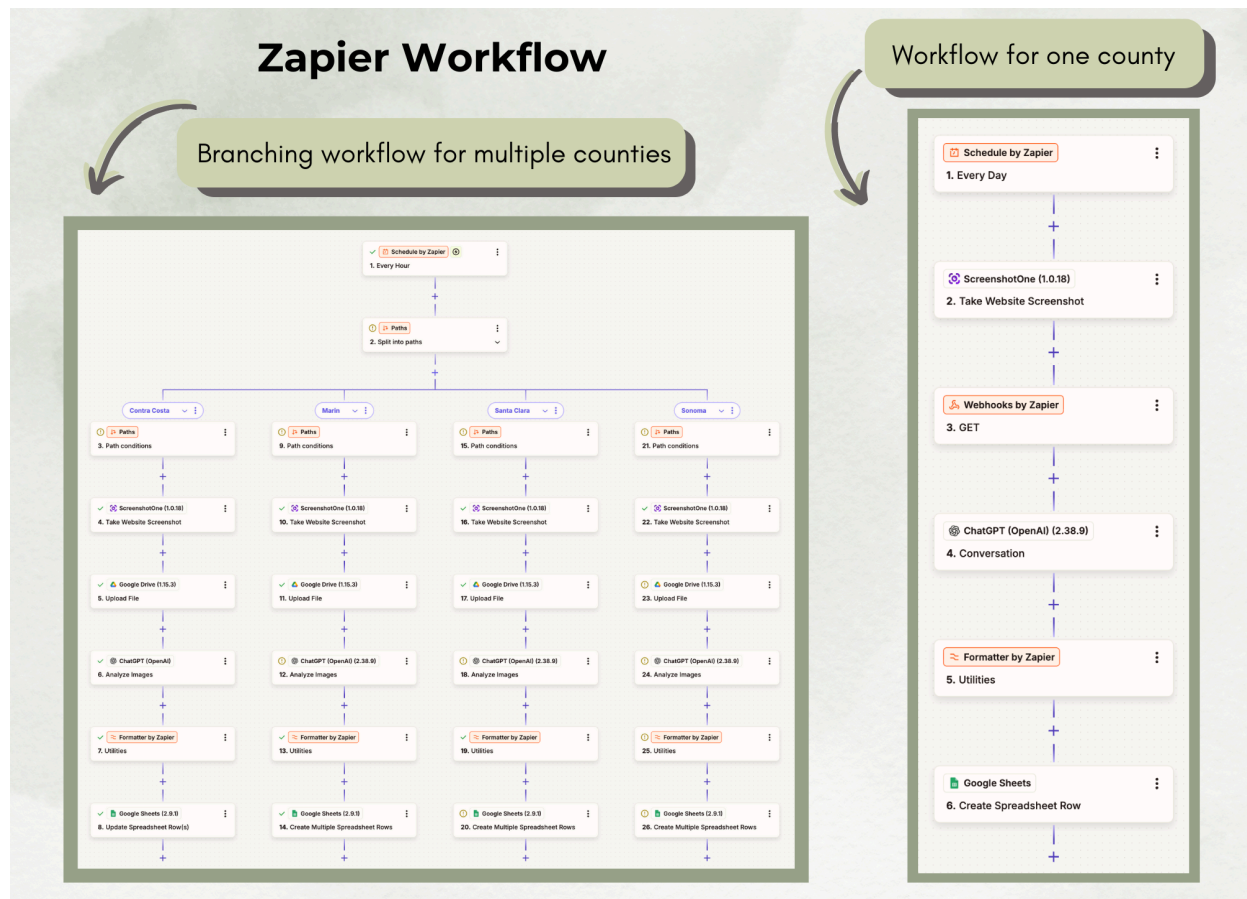
What if we could force all election results into the same format before attempting to extract the data?

Our solution was to standardize the input rather than the source. We hypothesized that if every county's results were captured in a uniform format, it would be easier for AI tools to interpret the data consistently. Another advantage was that the 2025 election cycle had just one statewide measure, Prop 50, along with a handful of local contests scattered across a few counties. This gave us the perfect opportunity to test the system with lower stakes and fewer items to automate. The format we believed would work best was a screenshot of each county's election results page.

It took some experimenting and researching through different screenshot software programs before we landed on the one we thought would work best. We also entertained contracting a few different screenshot companies who believed they'd be able to scrape the information from a screenshot, but unfortunately they were not able to commit to this type of experimental project. Ultimately, we designed an automated pipeline using Zapier, triggering a series of actions to extract and process the data.



The workflow operated as follows: 1) a newly updated county election's page triggered a Zapier automation. That link was sent to ScreenshotOne, which captured a screenshot of the results page. 2) The image was then saved to Google Drive and sent to ChatGPT with a fine-tuned prompt designed to extract only the necessary election data.



3) Once extracted, the cleaned data was routed back through Zapier, formatted into a Google Sheet, and connected to the same HTML-coded, public-facing dashboard we built during the 2024 election cycle.

Spreadsheet

This is the spreadsheet used to generate the HTML code.

Candidate HTML (Full List)

File Edit View Insert Format Data Tools Extensions Help

Menus

100%

123

Calibri

11

B

I

A

Ideally, this is where the zap would place the ballot information.

These are the HTML formulas that get pulled into WordPress. Zapier doesn't touch this part of the spreadsheet.

At first, this approach showed promise, particularly because it relied on an automation tool like Zapier, which is more accessible and intuitive than building a complex automation pipeline entirely in code. Zapier allowed us to move quickly without heavy engineering support and made it easier to test and iterate on the workflow.

It was also useful because it supported a branching logic system. Rather than building and maintaining 13 separate automations for each county, we were able to create a single Zap with conditional triggers that handled all 13 counties within the same workflow. This reduced complexity, simplified maintenance and made the system easier to scale or modify as county formats changed.

5. Where the Automation Broke Down

Why AI interpretation introduced inconsistency

Despite those advantages, the workflow still had significant drawbacks. The most persistent and familiar failure point was the chatbot component. The AI consistently struggled to follow the parameters and rules outlined in our prompts. Even when we copied and pasted the ballot and candidate information directly into the query, the bot failed to return accurate or consistent results.

Query + Parameters for Chatgpt & Claude

- Extract results from this screenshot. Some images will show more than 1 campaign. make sure to extract all of them!!
- RETURN ONLY CSV. Use comma as delimiter and wrap EVERY field in double quotes.
- HEADER (include exactly once):
- "Location","Measure Name","Yes Votes","No Votes","Updated On","Yes Votes Pct","No Votes Pct","Precincts Reporting Pct"
- Rules:
 - for each row include a row number
 - Output one row per fully visible contest/measure option (e.g., Yes/No).
 - If any value is unreadable, output "" for that field but still output EXACTLY 8 fields.
 - "Updated On" must be in YYYY-MM-DD HH:mm:ss.
 - No code fences. No extra text.
 - Make sure all contests are extracted
- Here is what needs to be collected:
 - Town of Fairfax Recall of LISEL BLASH (Vote For 1)
 - Town of Fairfax Recall of STEPHANIE HELLMAN (Vote For 1)
 - Proposition 50 (Vote For 1)
 - Measure I Petaluma Joint Union High School District (Vote For 1)
 - Measure J City of Sausalito (Vote For 1)
 - Measure K City of Sausalito (Vote For 1)
- Example:
 - "Marin County","Town of Fairfax Recall of LISEL BLASH","0","0","2025-11-04 20:15:00","0.00%","0.00%","0%"

This was particularly surprising given that we were testing the system before the election, when all vote tallies were set to zero. In theory, this should have simplified the task. Instead, the bot was unable to reliably identify races and measures on its own, which forced us to manually label every item we wanted it to extract. **We provided the exact data structure and formatting requirements, yet each time the Zap was triggered, the chatbot produced different outputs.** We tried both Chatgpt and Claude and unfortunately had issues with both tools. It seemed as though the intelligence of the chatbot degraded when it was added to the zapier automation workflow.

6. Returning to Traditional Scraping

Using AI to build the tool, not touch the data

With this in mind, we decided to try a fundamentally different approach. Our earlier experiments made one conclusion clear: we could not rely on AI to interpret, collect or analyze election data directly. The tools we tested were prone to hallucinations, errors and inconsistent outputs.

The question then became how AI could still be used to reduce the most tedious parts of data collection without touching the data itself. Our solution was to return to traditional scraping methods, similar to those used by product and engineering teams at larger news organizations.

Rather than hiring a dedicated technical team, we used an AI tool, Claude), to help us build a conventional scraper. With no computer science or data background, our team had to learn things on the fly and ask the bot constantly about what the code errors meant, which settings to use and how to access the output it produced. In this workflow, AI did not interact with the election data at all. Instead, it supported the development of the scraper itself, a role that would typically require a software engineer.

For a small newsroom like ours, which does not have the resources to fund a full tech team, this approach offered a practical way to access technical capabilities without sacrificing accuracy or editorial control. But with our limited skills, we had to somewhat give ourselves a mini crash course on how to build code, run code and ultimately iterate with the help of Claude.

Here's what you will need

 **CURSOR**

×

**CLAUDE
CODE** 

Cursor is a code editor where the work actually happens. It is the space where you write, edit, and run code. For journalists with no coding background, Cursor makes technical work more approachable by keeping everything in one place and allowing you to interact with code without needing to understand programming language in advance. There are paid versions, but only the free version is needed. - **Free**

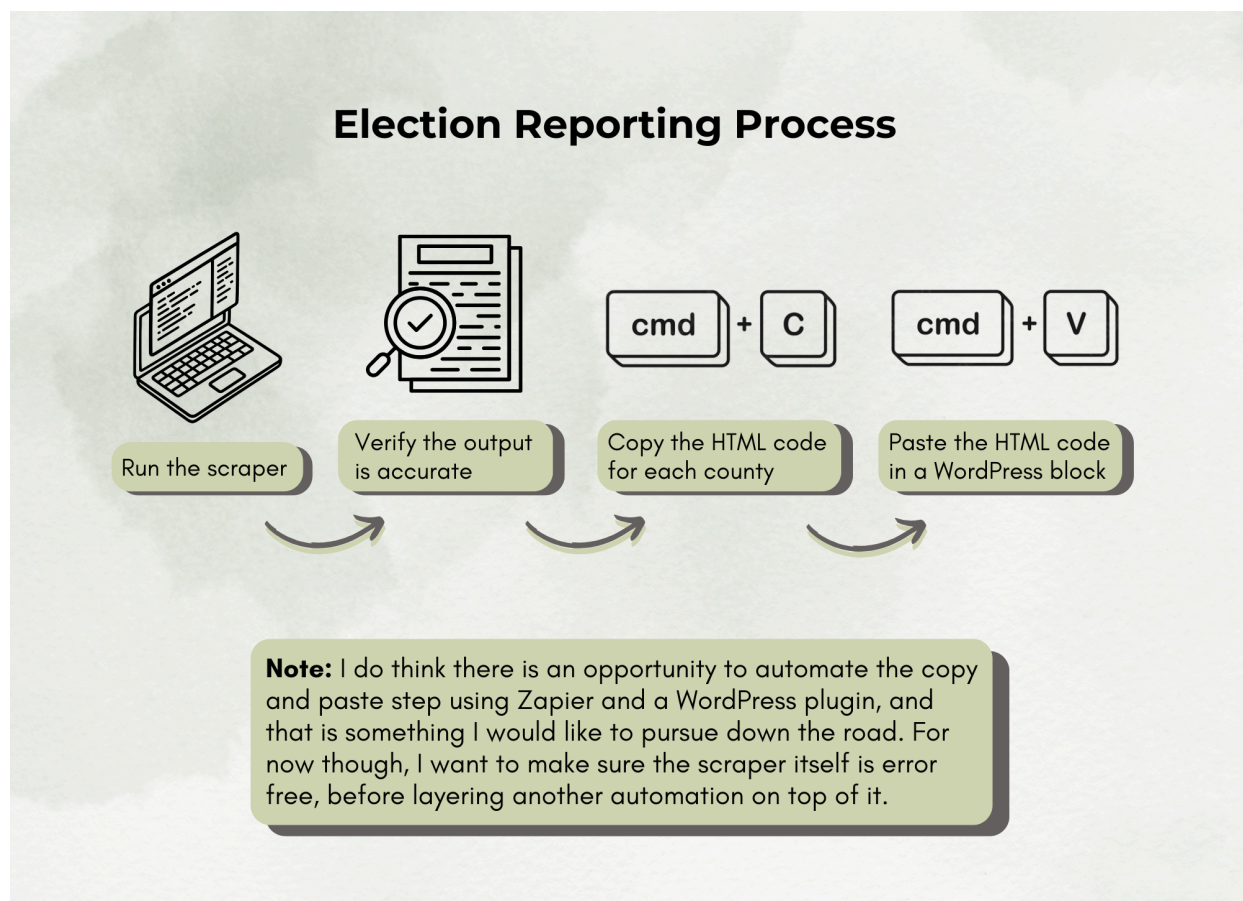
Claude Code is the AI assistant inside the coding environment. It helps explain what the code does, suggests what to write next, and helps fix errors when something breaks. Instead of touching or interpreting data, Claude Code builds the code itself while the user defines what the output should be.
***Note:** Claude Code is a feature of the Pro version of Claude, but it is different from the regular chatbot. Claude Code is used within a terminal or coding environment, while the standard Claude is used in a chat. - **\$20/month**

Cursor serves as the wrapper that brings everything together, offering a clean interface that houses both the terminal and Claude in one place. Technically you don't need Cursor to generate or run the code, but we found it delivers a much better overall experience for housing the project. A terminal alone is fairly bare bones, but Cursor lets you see all the files in your project at a glance and watch in real time as new ones get added. As long as you have a Claude account, you can access Claude Code directly through Cursor's terminal. Cursor also has its own built-in chatbot, but there's really no need to use it when Claude is right there.

Cursor is free, unless you want to pay to use its chat feature, and Claude Code runs for \$20 a month.

7. Creating this workflow for your newsroom's use

A guide to the features, what you need to feed the bot, and tips for better outcomes.

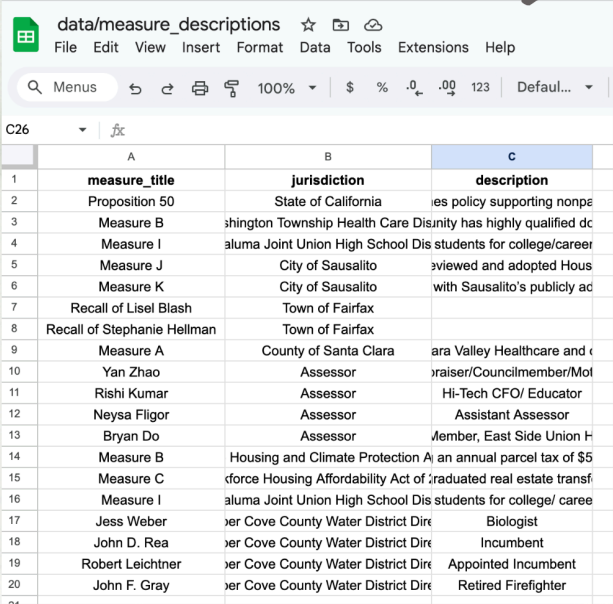


Recommendations: You can work through the steps by pulling the code from [GitHub](#), if you are comfortable pulling code and building from there. But we would recommend a different approach if you are not well-versed in computer science and code by “vibe coding” as they say these days. Take the code from this project, feed it into the Claude chatbot, and use it to get Claude up to speed on the challenges and nuances of this specific project. From there, start collecting your county sites and feeding them into Claude one at a time. Doing this process step by step is important, both to avoid overwhelming the chat and because counties often have different systems and formats. That way Claude can focus on learning each system, its firewalls and other nuances while building a strong scraper before moving on to the next. To get started, feed Claude the following query below with the [GitHub code](#), filling in the county names and links as you go.

This ensures your project starts smarter than mine did, because Claude already knows what challenges to anticipate based on our experience and can iterate with that context in mind. It will also understand from the start that this cannot be built like a typical scraper but needs to be stress tested for the demands of election night.

Note: Create a folder that houses all the reference docs you want the bot to be aware of (just like a regular chat). You should also add the [Github code](#) and both of the playbooks Bay City News has created, so it has documented context to similar projects and is aware of potential roadblocks.

You should also create a spreadsheet including every contest you would like to pull on election night. This should include any information about the race or measure. Many counties will only include the definition of a measure or the profession of a candidate in the ballot preview, but will not include this information on election night. This spreadsheet ensures the scraper pulls this information and adds it to the HTML it generates.



Download as a CSV and add it to the project file. Tell Claude the name of the file and where to find it. Prompt it to attach this information to the code. Make sure to test this step incase the title is case sensitive or there are any other errors between the spreadsheet and the content the scraper is grabbing.

	A	B	C
	measure_title	jurisdiction	description
1	Proposition 50	State of California	ies policy supporting nonpe
2	Measure B	shington Township Health Care Disunity has highly qualified dc	
3	Measure I	aluma Joint Union High School Dis students for college/career	
4	Measure J	City of Sausalito	viewed and adopted Hous
5	Measure K	City of Sausalito	with Sausalito's publicly ad
6	Recall of Lisel Blash	Town of Fairfax	
7	Recall of Stephanie Hellman	Town of Fairfax	
8	Measure A	County of Santa Clara	ara Valley Healthcare and c
9	Yan Zhao	Assessor	raiser/Councilmember/Mot
10	Rishi Kumar	Assessor	Hi-Tech CFO/ Educator
11	Neysa Fligor	Assessor	Assistant Assessor
12	Bryan Do	Assessor	Member, East Side Union H
13	Measure B	Housing and Climate Protection A	an annual parcel tax of \$5
14	Measure C	force Housing Affordability Act of /raduated real estate transf	
15	Measure I	aluma Joint Union High School Dis students for college/ caree	
16	Jess Weber	er Cove County Water District Dire	Biologist
17	John D. Rea	er Cove County Water District Dire	Incumbent
18	Robert Leichter	er Cove County Water District Dire	Appointed Incumbent
19	John F. Gray	er Cove County Water District Dire	Retired Firefighter

Measure Descriptions

Candidate Professions

Dashboard

Alameda County Election Results

Search for measures, candidates, contests...

VOTER TURNOUT

52.67%

BALLOTS CAST

508,730

REGISTERED VOTERS

965,876

PRECINCTS REPORTING

539 of 539 Precincts Reported

State of California

Proposition 50

Requires temporary use of new congressional district maps through 2030. Directs independent Citizens Redistricting Commission to resume enacting congressional district maps in 2031. Establishes policy supporting nonpartisan redistricting commissions nationwide. Fiscal Impact: One-time costs to counties of up to a few million dollars statewide to update election materials to reflect new congressional district maps.

✓ PROPOSITION 50 IS PASSING 507,954 votes counted

RESPONSE	% VOTES	PCT	VOTES
✓ Yes		80.58%	409,301
No		19.42%	98,653

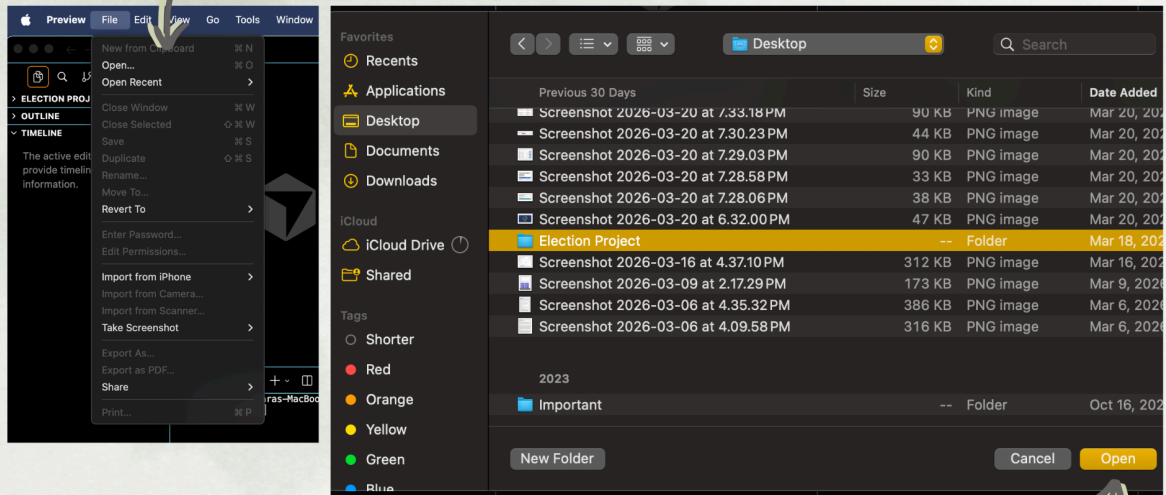
It's helpful to have the end product in mind as you are building. Your dashboard will look like ours (shown above) if you use the same code we attached from GitHub. If you'd prefer a different output, then screenshot other news outlet election dashboards as inspiration and feed them into a regular Claude chat, not the Claude Code chat in the terminal. This ensures that the bot doesn't get confused and accidentally edit the actual project. By doing it in the regular chat, the stakes will be lower and there will be less risk of messing up the core code. Once you add the screenshots, ask it to create a similar election dashboard with these fields in mind:

- Voter turnout
- Turnout percentage
- Ballots cast
- Registered voters
- Precincts reported
- Precincts reporting per contest
- Title
- Contest/race name
- List of choices, each parsed into:
 - candidate/response name
 - votes
 - percentage

If for whatever reason the counties you are scraping do not have some of these fields or have others you'd like added, then edit them accordingly.

Claude will generate a dashboard. You can ask the bot to change anything you want to alter the appearance. Once you have a dashboard you like, then feed this code into Claude Code in the terminal of Cursor and it will use the code as a template to generate the HTML code you will add to your site.

Click Open.

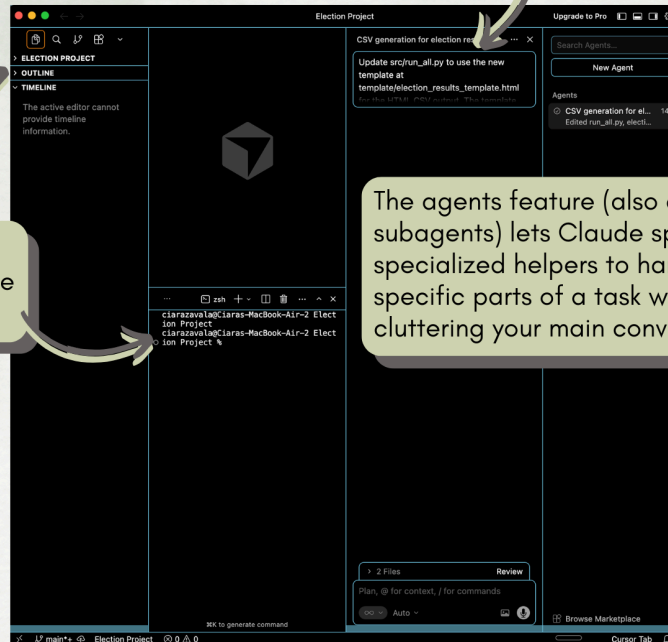


Find the election project folder. It should include any information the bot needs for reference. This will also be the folder the bot will add the code to, scrapers and any guides it creates. Then click open.

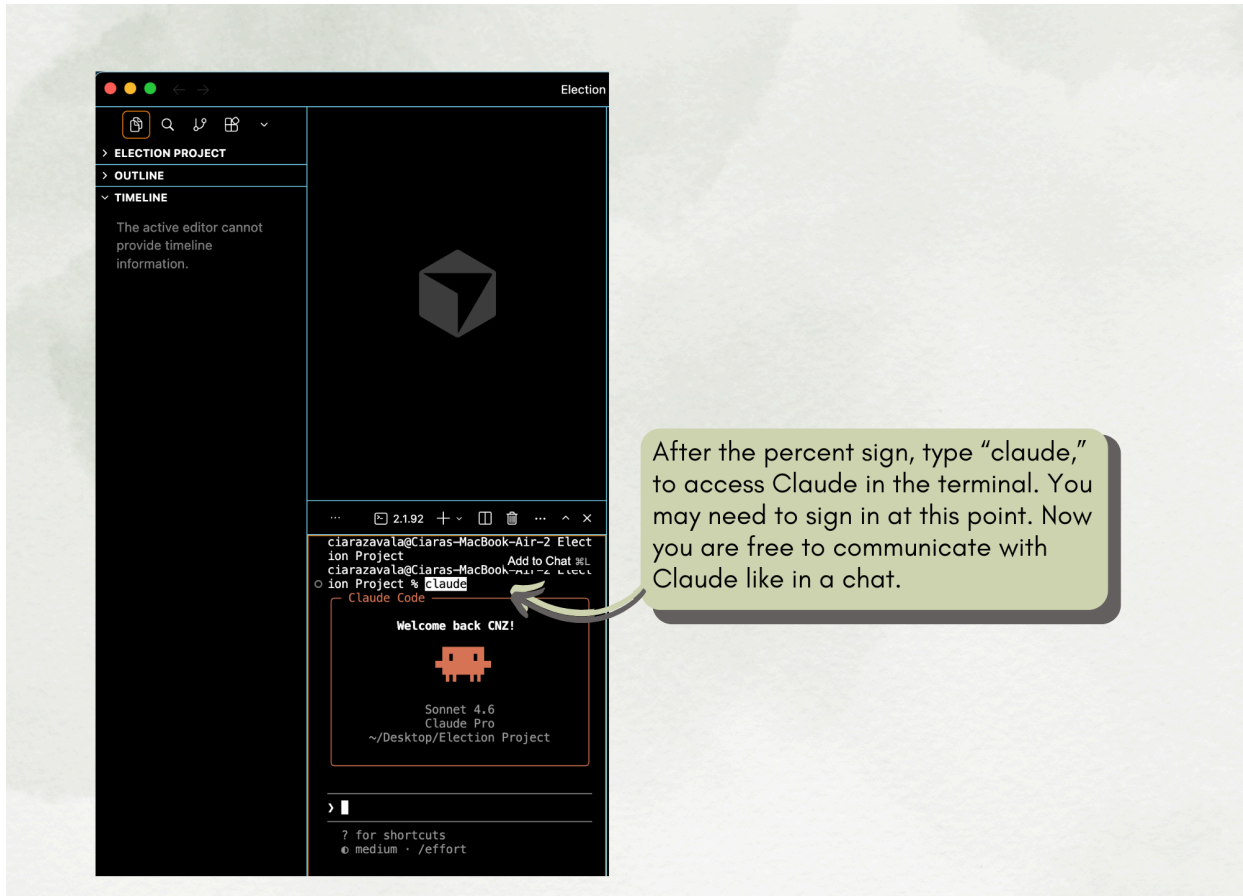
The project folder shows all of the documents, code and overall information that Claude is using to generate the scraper.

This is the chat for Cursor, but is not needed since Claude is already part of the terminal.

This is the terminal that will house the code and also the chat with Claude.



The agents feature (also called subagents) lets Claude spin up specialized helpers to handle specific parts of a task without cluttering your main conversation.



8. Suggested query to replicate in your newsroom

Feed this prompt into Claude after you open it in the terminal.

Note: This is the exact query Claude gave us when we prompted it to generate a replicable project query. There may be some technical jargon in there that is unfamiliar, but the important thing is that Claude knows exactly what it means.

I need a Python election night results scraper for the following counties in [STATE]:

- [County A] — results URL: [URL]

- [County B] — results URL: [URL]

- [County C] — results URL: [URL]

- For each county, scrape: contest names, candidate/choice names, vote totals, percentages, precincts reporting, and voter turnout (ballots cast, registered voters, turnout %).]

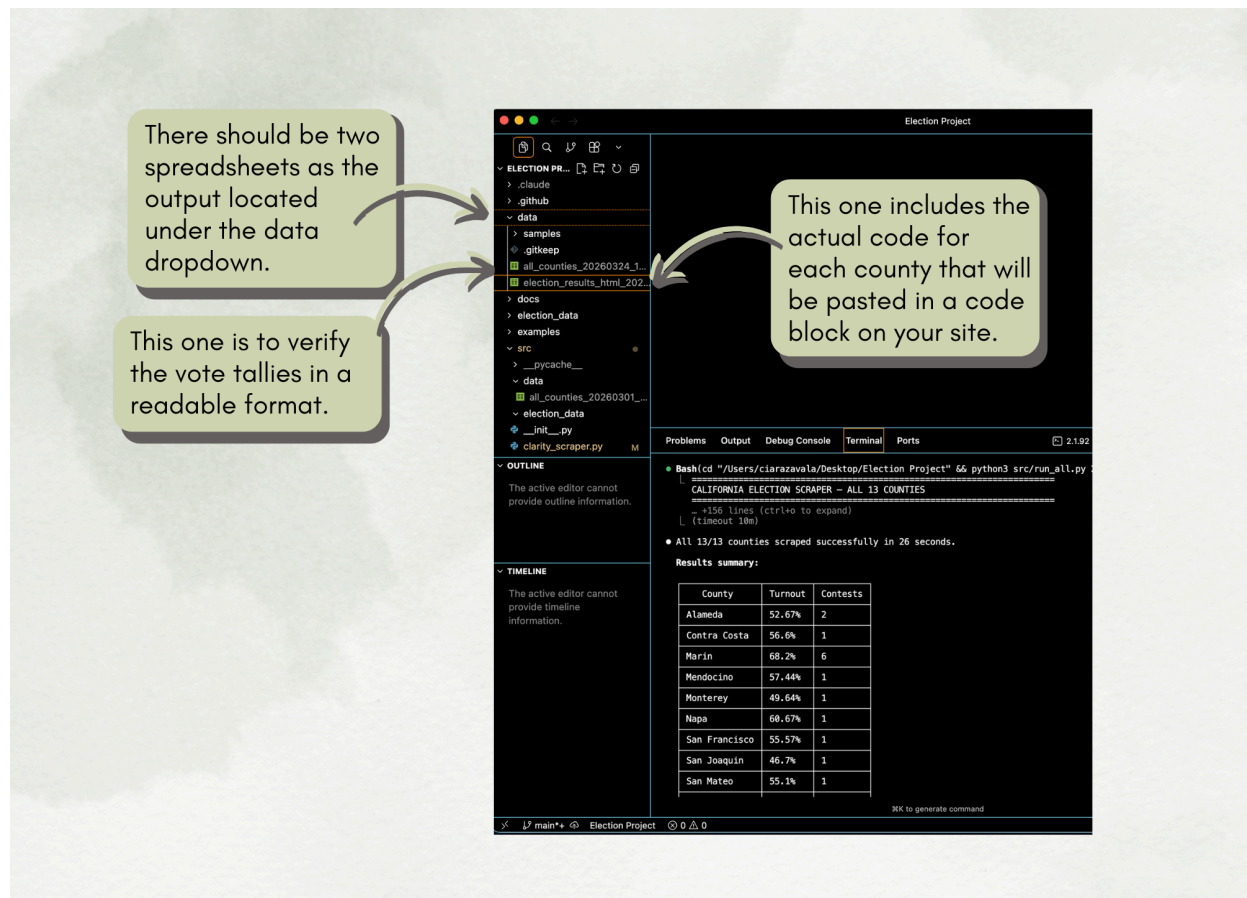
- Some counties use the Clarity Elections platform (results.enr.clarityelections.com), which requires Selenium because the pages are JavaScript-rendered SPAs behind CloudFront. Others use simpler HTML pages that can be scraped with requests + BeautifulSoup.
- Requirements:
 - Run all counties in parallel using ThreadPoolExecutor
 - Retry failed scrapers once after a short delay
 - Output a single CSV with columns: county, contest, choice, votes, percentage]
 - Headless Chrome for Selenium scrapers
 - Handle election night high load (extended timeouts, backoff on retry)
 - Structure the project as:
 - clarity_scraper.py — reusable Clarity scraper class
 - multi_platform_scraper.py — one class per non-Clarity county
 - run_all.py — parallel orchestrator that writes the CSV
- Tips for getting the best result:
 - 1. Identify platforms first: Before running the query, visit each county's results page and note whether it's Clarity Elections, LiveVoterTurnout, a custom site, or a PDF. Include that in the prompt so Claude doesn't have to guess.
 - 2. Provide real URLs: Live or sample URLs are essential; Clarity URLs embed election-specific IDs (e.g. /CA/Marin/124182/) that change each election cycle.
 - 3. Specify output format: If the newsroom needs something other than CSV (e.g. JSON for a live results page, or a specific schema for a CMS), say so explicitly.
 - 4. Iterate per county: Ask Claude to build and test one non-Clarity county at a time, since each site is different. Trying to do all at once in one prompt often produces untested code.
 - 5. Mention election night conditions: Explicitly ask for retry logic and timeouts or Claude will generate a scraper that works in testing but fails under real election night server load.

9. Step-by-Step Guide

Now that you have an understanding of the platforms and how to interact with code, follow these steps.

1. Create an elections project folder on your computer.
2. Create and add the descriptions spreadsheet to the folder.
3. Add the [GitHub code](#) from our BCN project to the project folder.
4. If you prefer a different design, then add the new template code you created to the project folder.
5. Add any other information to the project folder that would be useful for the bot to know.

6. Now paste the [suggested query](#) listed from above and sub out information specific to the counties you are covering like the site URLs.
7. Prompt the bot to work on each county separately and one by one.
8. Once the bot is returning the results, it should say 1/1 counties scraped successfully ✓. Once all your other scrapers are good to go, it should say something like 13/13 counties scraped successfully ✓.
9. To trigger the entire process, type “run it,” and it will return the results. The scraper will return two documents in the sidebar under “data.” One will say “all_counties_CSV,” and the other will say “HTML_CSV.” The first one includes a spreadsheet of the data scraped. **This is where a human editor verifies the information and that votes tallies are correct.** The “HTML_CSV,” is a duplicate of the information but in code form. This spreadsheet will have the code for each county in separate cells.



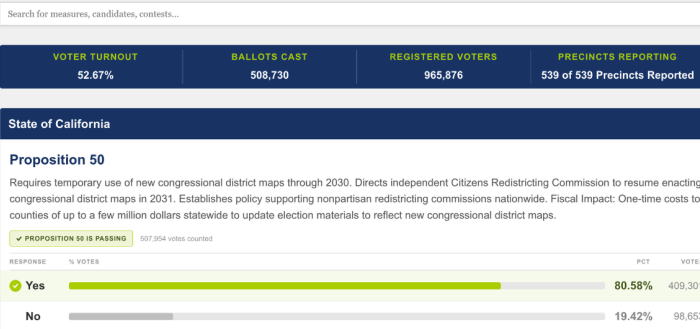
10. Paste the code into an HTML code block in your website's system, like WordPress.
11. Publish and tada! You're done.

Paste the code from the HTML CSV into the corresponding county code block. Hit publish and the page will update as the dashboard with the results.

Alameda County Election Results

```
<!DOCTYPE html> <html lang="en"> <head> <meta charset="UTF-8"> <meta name="viewport" content="width=device-width, initial-scale=1.0"> <title>Election Results</title> <style> *, ::before, ::after { box-sizing: border-box; margin: 0; padding: 0; } body { font-family: Arial, sans-serif; background-color: #f0f0f0; padding: 20px; } /* Turnout bar */ .turnout-bar { background-color: #1a3668; color: white; border-radius: 5px; padding: 8px 16px; margin-bottom: 20px; display: flex; flex-wrap: wrap; gap: 8px; } .turnout-stat { flex: 1; min-width: 120px; text-align: center; padding: 0 16px; border-right: 1px solid rgba(255,255,255,0.2); } .turnout-stat:last-child { border-right: none; } .turnout-stat-label { font-size: 15px; font-weight: 700; text-transform: uppercase; letter-spacing: 0.07em; color: #ccccff; margin-bottom: 4px; } .turnout-stat-value { font-size: 18px; font-weight: 700; color: white; } .turnout-stat-value.green { color: white; } /* Race box */ .race-box { background-color: white; border: 1px solid #ddd; border-radius: 5px; margin-bottom: 20px; overflow: hidden; } .race-title { background-color: #1a3668; color: white; padding: 18px 14px; font-weight: bold; font-size: 1em; } /* Ballot measure - NV style */ .measure-block { padding: 16px 16px 0 16px; } .measure-name { font-size: 1.15em; font-weight: 700; color: #1a3668; margin-bottom: 6px; } .measure-desc { font-size: 0.88em; color: #555; line-height: 1.55; margin-bottom: 12px; } .measure-status-row { display: flex; align-items: center; gap: 12px; margin-bottom: 18px; } .measure-status-badge { display: inline-flex; align-items: center; gap: 5px; font-size: 11px; font-
```

Alameda County Election Results



12. Now continue this process every 30 minutes or whatever cadence you choose to reflect election night results as they come in.

10. Conclusion

What this means for future election cycles

This is an experiment we will continue to iterate upon. Since we were able to develop the scrapers on the spot on election night in 2025 and successfully scrape many of the 13 counties we cover, we are very optimistic for the primary coming up this June 2026. The scrapers will need to be stress tested, however. The high volume of traffic hitting registrar websites simultaneously can cause timeouts, slowdowns or temporary blocks that a scraper built under calm conditions may not be equipped to handle. Testing early and often, and specifically under simulated election night pressure is essential.

What this project ultimately revealed is that AI does not need to touch the data to be useful. Its greatest contribution was not interpreting results or

making editorial decisions, but doing the technical heavy lifting that would have otherwise required a dedicated engineering team. For small and under-resourced newsrooms, that distinction matters enormously. Local newsrooms are constantly being asked to do more with less, and tools like Claude are beginning to close the gap between what a small team can realistically accomplish and what was once only possible with significant engineering resources and large newsroom budgets. That is not a small thing. That is a fundamental shift in what independent and nonprofit journalism can look like.

The verification step built into this workflow is not a contradiction of the time savings, it is part of how they work. The goal was never to replace editorial judgment. It was to cut down on the most repetitive parts of the process so that we can spend more time where our work actually matters. Without automation, we are copying and pasting vote tallies from 13 different county websites with completely different formats, structures and update cadences, all while trying to report on what those numbers mean. The scraper absorbs that labor. What remains for the human is a quick scan to confirm the data came back clean, which is a more manageable task than starting from scratch on every update cycle.

And the importance is even more obvious at scale. One county automated might not feel worth the setup time. But 13 counties automated, starts to look very different. It is using one process to cover a much bigger amount of ground. Scale that to 26 counties and we are covering half of California. Scale it to all 58 and we are doing something no wire service, no national platform and no single newsroom has consistently done which is providing real time local election results to communities that are otherwise invisible on election night.

If there is one thing we hope other newsrooms take away from this project, it is that **preparation and testing are everything.** The single biggest obstacle we have faced across both election cycles has not been the lack of resources or the technology itself, it has been the lack of access to zero ballots, which are the templates registrar offices publish before election night that show exactly how results will be formatted and displayed. Without them, you are essentially building a scraper blind and hoping the structure you assumed matches the structure that actually appears when votes start coming in. The more lead time you have to test

against real data structures, the more confident you can be when election night arrives and every minute counts.

The evolution of AI is also worth acknowledging here, because it is moving fast. The tools available during the 2024 election cycle were meaningfully different from what we used in 2025, and what exists today in 2026 will look different again by the time the next major election rolls around. That is both exciting and humbling. It means the workflows we build now will need to be revisited, not because they failed, but because better options will emerge. Staying curious and staying willing to start over when something better comes along is part of the work.

That is also why the need to keep iterating cannot be overstated. This playbook is not a finished product. It is a snapshot of where we are right now, with the understanding that every election cycle will teach us something new. The scrapers will break in ways we did not anticipate. Counties will change their website formats without warning. New platforms will emerge and old ones will disappear. Each of those moments is an opportunity to learn, adjust and build something more resilient than what came before.