Patterns

SafeGraph's Patterns dataset includes visitor and demographic aggregations for points of interest (POIs) in the US. This contains aggregated raw counts of visits to POIs from a panel of mobile devices, answering how often people visit, how long they stay, where they came from, where else they go, and more.

Contents:

Patterns Schema

[patterns.csv]

Patterns is a dataset of visitor and demographic aggregations available in the US only. For Patterns data in Canada, see Weekly Patterns.

Column Name	Description	Type	Example
`placekey`	Unique and persistent ID tied to this POI. See the Placekey Key Concept for details on placekey design.	String	`[email protected]`
`parent_placekey`	If place is encompassed by a larger place (e.g. mall, airport), this lists the placekey of the parent place; otherwise `null`. See more on parent-child relationships in Spatial Hierarchy.	String	`[email protected]`
`location_name`	The name of the place of interest.	String	`Salinas Valley Ford Lincoln`
`street_address`	Street address of the place of interest.	String	`1100 Auto Center Circle`
`city`	The city of the point of interest.	String	`Irvine`
`region`	The state, province or county of the place of interest. See region.	String	`CA`
`postal_code`	The postal code of the place of interest.	String	`92602`
`safegraph_brand_ids`	Unique and consistent ID that represents this specific brand.	List	`SG_BRAND_59dcabd7cd2395a2, SG_BRAND_8310c2e3461b8b5a`
`brands`	If this POI is an instance of a larger brand that we have explicitly identified, this column will contain that brand name. See brands.	List	`ford, lincoln`

Column Name	Description	Type	Example
`date_range_start`	Start time for measurement period in ISO 8601 format of YYYY-MM-DDTHH:mm:SS±hh:mm (local time with offset from GMT).	String	`2020-03-01T00:00:00-06:00`
`date_range_end`	End time for measurement period in ISO 8601 format of YYYY-MM-DDTHH:mm:SS±hh:mm (local time with offset from GMT). The end time will be the last day of the month at 12 a.m. local time.	String	`2020-03-31T00:00:00-06:00`
`raw_visit_counts`	Number of visits in our panel to this POI during the date range. See raw_visit_counts.	Integer	`1542`
`raw_visitor_counts`	Number of unique visitors from our panel to this POI during the date range.	Integer	`1221`
`visits_by_day`	The number of visits to the POI each day (local time) over the covered time period. See visits_by_day.	JSON [Integer]	`[33, 22, 33, 22, 33, 22, 22, 21, 23, 33, 22, 11, 44, 22, 22, 44, 11, 33, 44, 44, 44, 33, 34, 44, 22, 33, 44, 44, 34, 43, 43]`
`poi_cbg`	The census block group the POI is located within.	String	`560610112022`
🛡 `visitor_home_cbgs`	The number of visitors to the POI from each census block group based on the visitor's home location. See visitor_home_cbgs.	JSON {String: Integer}	`{"360610112021": 603, "460610112021": 243, "560610112021": 106, "660610112021": 87, "660610112021": 51}`
🛡 `visitor_home_aggregation`	The number of visitors to the POI from each census tract based on the visitor's home location.	JSON {String: Integer}	`{"17031440300": 1005, "18089021500": 522, "17197883516": 233, "17031826402": 5, "17031826301": 4, "04013115802": 4}`
🛡 `visitor_daytime_cbgs`	The number of visitors to the POI from each census block group based on primary daytime location between 9 am - 5 pm. See visitor_daytime_cbgs.	JSON {String: Integer}	`{"360610112030": 9872, "880610112021": 8441, "569610112020": 5671, "160610112041": 2296, "980610112021": 1985}`
🛡 `visitor_country_of_origin`	The number of visitors to the POI from each country based on visitor's home country code. See visitor_country_of_origin.	JSON {String: Integer}	`{"US": 98,"CA": 12}`
`distance_from_home`	Median distance from home travelled by visitors (of visitors whose home we have identified) in meters. See distance_from_home.	Integer	`1211`
`median_dwell`	Median minimum dwell time in minutes. See median_dwell.	Double	`5`

Column Name	Description	Type	Example
`bucketed_dwell_times`	The distribution of visit dwell times based on pre-specified buckets. Key is the range of dwell time in minutes and value is number of visits that were within that range. See bucketed_dwell_times.	JSON {String: Integer}	`{ "<5": 40, "5-20": 22, "21-60": 45, "61-240": 3, ">240": 5}`
`related_same_day_brand`	Other brands that the visitors to this POI visited on the same day as the visit to this POI. Limited to top 20. See related_same_day_brand.	JSON {String: Integer}	`{"mcdonalds": 7,"amc": 5,"target": 3}`
`related_same_month_brand`	Other brands that the visitors to this POI visited in the same month as the visit to this POI. Limited to top 20. See related_same_month_brand.	JSON {String: Integer}	`{"mcdonalds": 7,"amc": 5,"target": 3}`
`popularity_by_hour`	The number of visits in each hour over the course of the date range, in local time. First element in the array corresponds to the hour of midnight to 1 am, second is 1am to 2am, etc. See popularity_by_hour.	JSON [Integer]	`[ 0, 0, 0, 0, 0, 0, 0, 222, 546, 444, 333, 232, 432, 564, 456, 345, 678, 434, 545, 222, 0, 0, 0, 0 ]`
`popularity_by_day`	The number of visits in total on each day of the week (in local time) over the course of the date range. See popularity_by_day.	JSON {String: Integer}	`{"Monday": 3300,"Tuesday": 1200,"Wednesday": 898,"Thursday": 7002,"Friday": 5001,"Saturday": 5987,"Sunday": 0}`
🛡 `device_type`	The number of visitors to the POI that are using Android vs. iOS.	JSON {String: Integer}	`{"android": 6, "ios": 8}`
🛡 `†carrier_name`	The number of visitors to the POI based on the wireless carrier of the device. See carrier_name.	JSON {String: Integer}	`{"Verizon": 342, "T-Mobile": 288, "AT&T": 265}`
`normalized_visits_by_state_scaling`	`raw_visit_counts` scaled using the mobile device sampling rate for the state in which the POI is located.	Float	`715.08396...`
`normalized_visits_by_region_naics_visits`	`raw_visit_counts` divided by the `sum(raw_visit_counts)` to the `naics_code` in the same state or province during the same time period. This measures changes in the category-specific popularity of the POI over time.	Float	`0.00411...`
`normalized_visits_by_region_naics_visitors`	`raw_visit_counts` divided by the `sum(raw_visitor_counts)` to the `naics_code` in the same state or province during the same time period. This measures changes in the visits per devices that visited the	Float	`0.0127...`

Column Name	Description	Type	Example
	same category in SafeGraph's panel to the POI over time.		
`normalized_visits_by_total_visits`	`raw_visit_counts` divided by the `total_visits` in the same state or province during the same time period. This measures changes in the relative popularity of POI over time.	Float	`0.0000567...`
`normalized_visits_by_total_visitors`	`raw_visit_counts` divided by the `total_devices_seen` in the same state or province during the same time period. This measures changes in the visits per device in SafeGraph's panel to the POI over time.	Float	`0.0000913...`

🛡️ We do not report data unless at least 2 visitors are observed from that group. If there are between 2 and 4 visitors this is reported as 4. See more on privacy here.

† *carrier_name is a premium column. Please Contact Sales for more details.*

Panel Overview Data

Along with the Patterns file, we also deliver Panel Overview Data (see tables below) to help you better understand the context of the data appearing in Patterns.

Home Location Distributions by State/Census Block Group

[home_panel_summary.csv]

Column Name	Description	Type	Example
`year`	Calendar Year	Integer	`2018`
`month`	Calendar month starting from 1 as January	Integer	`1`
`state`	Lowercase abbreviation of U.S. state or territory	String	`ca`
`census_block_group`	FIPS code for this Census block group	String	`530330080012`
`number_devices_residing`	Number of distinct devices observed with a primary nighttime location in the specified census block group.	Integer	`54481`
`number_devices_primary_daytime`	Number of distinct devices observed with a primary daytime location in the specified census block group.	Integer	`54482`

Number of Visits/Visitors by State

[visit_panel_summary.csv]

Note: Includes one row with ALL_STATES to provide total visitors seen in the month (might be less than the sum of visitors by state due to same visitors having visits in multiple states).

Column Name	Description	Type	Example
`year`	Calendar Year	Integer	`2018`
`month`	Calendar month starting from 1 as January	Integer	`1`
`state`	Lowercase abbreviation of U.S. state or territory	String	`ca`

Column Name	Description	Type	Example
`num_visits`	Number of point-of-interest visits observed in the specified state	Int	`8900`
`num_unique_visitors`	Number of unique visitors observed with at least 1 point-of-interest visit in the specified state	Integer	`966`

Normalization Stats

[normalization_stats.csv]

Column Name	Description	Type	Example
`year`	Calendar Year	Integer	`2018`
`month`	Calendar month starting from 1 as January	Integer	`1`
`day`	Calendar day	Integer	`1`
`region`	When iso_country_code == US, then this is the USA state or territory. When iso_country_code == CA, then this is the Canadian Province or territory.	String	`CA`
`total_visits`	All visits we saw on the given day in local time (includes visits to POI and visits to homes)	Integer	`200`
`total_devices_seen`	Total devices in our panel which we saw on the given day with any visit in local time (POI or home visit)	Integer	`50`
`total_home_visits`	Visits we saw on the given day in local time to the device's home geohash-7	Integer	`120`
`total_home_visitors`	Total devices we saw on the given day with at least 1 visit to the device's home geohash-7	Integer	`35`

Key Concepts

Geographic Bias

Small geographic bias exists in our panel based on our understanding of the home locations of the devices in the panel. SafeGraph tested for geographic bias by comparing its determination of the state-by-state numbers of home location of the devices in the panel to the true proportions reported by the 2016 US Census. Based on that analysis, SafeGraph panel density closely mirrors true population density. The overall average percentage point difference is < 1% with a maximum of +/-3% per state. For a deep dive on geographic bias in the panel, see Quantifying Sampling Bias in SafeGraph Patterns.

Panel Growth

The panel has grown significantly since its inception. As such, it may be important to normalize the data when doing time series analysis across long periods of time or multiple releases. We have seen success by normalizing visits by the total number of visits in the SafeGraph Panel, month by month. It is also worth exploring normalizing based on state or census block group. With each delivery, we provide you with the Panel Overview Data files to enable you to do these calculations.

Predicting Financial Indicators

SafeGraph data can be used to estimate foot traffic and predict financial indicators of companies (eg. number of visitors, revenue, etc.). Please see our Data Science Resources on Normalization for approaches to improving predictions.

Correlation between reported company KPIs and SafeGraph visits will vary depending on multiple factors related to the company:

- Does the business separately report online vs in-store sales and revenue?
- How much do online sales contribute to the overall revenue?
- How much revenue is generated outside of the US and Canada? (SafeGraph visits are US and CA only)
- What is the ground truth correlation between foot traffic and sales for that business? For example, the relationship between foot traffic and sales at a car dealership has a very different pattern than at a convenience store.

Visit Attribution for Special Cases

- **Dense urban areas** - Visits to urban, suburban, and rural areas have varying precision levels, so care should be taken performing these types of comparisons. In general, it is more difficult to measure visits, say, to a midtown Manhattan Starbucks than a visit to a suburban standalone Starbucks.
- **Large structures/indoor malls** - We attribute visits to the parent POI only when we determine that the parent completely encloses its children indoors. We believe this is the most accurate option given the limitations of GPS inside such structures and currently do this for the following parent POI types:
  - Airports
  - Indoor malls (not including outdoor, open-air malls)
  - Major medical facilities
  - Hotels
  - Casinos
- **Parent POIs** - We attribute visits to both the parent POI and its children if the parent does not contain the children indoors. `raw_visit_counts` at a parent POI **=** SUM(`raw_visit_counts`) at all child POIs **+** other visits that the parent picks up itself (not through any of its children). Therefore, if you count visits at the parent level and then again for all children, you are double counting visits. For example:
  - A shopping center POI may have `raw_visit_counts` **=** SUM(`raw_visit_counts`) for all of its children because there are not any gaps within the shopping center to visit without the presence of a child POI.
  - A parent POI, like a golf course, will likely have more visits than the sum of its children because there are plenty of places within the golf course to visit without making a visit to a child POI (ex: playing 18 holes on the green but not making a visit to the clubhouse or restaurant).
- **Strip malls** - We attribute visits to the individual stores as well as the parent strip mall (assuming we have a POI for the entire strip mall). There will be instances where we have not divided a strip mall polygon into its constituent stores. Our model to determine visits does take a number of factors into account, including distance from centroid, so even though there are multiple POIs in one strip mall polygon, we attempt to allocate visits within the strip mall to the POI most likely to have received the visit.

Visit Nuances

- **Worker & non-worker visits** - In an earlier release (May 2020), we attempted to exclude workers at a POI from our visit counts to the POI. However, we were only able to determine a limited number of workers so we decided to remove this filter, and in all up-to-date, current, and historical data, visits may include worker visits. The best way to determine how many of the visitors to a POI are workers is by looking at the `bucketed_dwell_time` column - these visitors will be disproportionately represented in the highest bucket.
- **GPS data** - The visits are determined using GPS data, we do not include any GPS data with a horizontal accuracy >160 meters. See our Visits Attribution Whitepaper for more details.
- **Very long visits** - Sometimes a visit lasts a very long time (e.g., > 24 hours). Please see our Documentation on how Patterns handles Long Visits for a number of nuances about these edge scenarios. If you are seeing visits to a POI that are longer than expected, this is likely due to picking up an employee device or picking up someone in a place above a POI (such as residential or retail over office).
- **Relationship with POI opening and closing dates** - Where the dates of POI opening and closing are known (see `opened_on` and `closed_on` columns in Core Places), no visits will be attributed to the POI before the `opened_on` date or after the `closed_on` date. If these dates are not known, it is possible for the algorithm to mis-attribute visits for those POIs (e.g., after a POI has closed in real life but where this knowledge has not yet been captured in our data).
  - *When* SafeGraph learns of a POI opening and closing can occasionally matter as well: e.g., in a few edge cases, our algorithm may capture that a POI closed several months after it actually did (i.e., we assign a `closed_on` date of `2021-01-01` to a POI in the Apr 2021 release). In cases like this, we may incorrectly assign visits to the POI in Feb and Mar 2021. These small discrepancies will occur until we complete a backfill, after which our visits algorithm will correctly account for known opening and closing dates.

Using Census Block Groups

The census block group is the highest geographic resolution for which the US Census Bureau provides demographic information. This demographic data is publicly available through APIs maintained by the US Census Bureau. SafeGraph provides census block group demographic data to download for free.

For the `poi_cbg` column and any columns referencing census block groups, note that we are using the 2010-2019 version of the census block group boundaries and will update our documentation and customers in advance of transitioning to the 2020 version of the census block group boundaries.

There are also resources for developers on Github and Stackoverflow for working with the US Census Bureau APIs. Some of the most common APIs are the Population Estimates API and the Decennial Census

Determining Home Location

Many columns in SafeGraph datasets rely on estimates of device “home location” (e.g., at the level of a census block group). “Home location” is an abbreviation for “common nighttime location”.

SafeGraph dataset	Columns and files
Monthly and Weekly Patterns	• `visitor_home_cbg` • `visitor_home_aggregation` • `distance_from_home` • `visitor_country_of_origin` • `home_panel_summary.csv` • `normalization_stats.csv`
Neighborhood Patterns	• `device_home_areas` • All of the `*_device_home_areas` columns • `distance_from_home` • `home_panel_summary.csv` • `normalization_stats.csv`
Social Distancing Metrics	All the data reported rely on an estimate of the “home location” for a device, listed in the column `origin_census_block_group` .

- `visitor_home_cbg` , `visitor_home_aggregation` , and `visitor_country_of_origin` are determined by analyzing 6 weeks of data during nighttime hours (between 6pm and 7am). We require a sufficient amount of evidence (total data points and distinct days) to assign a home (common nighttime) geohash-7 for the device, which is then mapped to a census block group, census tract, and country of origin.
- `visitor_daytime_cbgs` is determined by looking at 45 days of data and determining where the device is most frequently during daytime hours (9am-5pm local time). It is easier to determine a home/common nighttime location of a device than it is to determine the daytime census block group, so our data contains more `home_cbgs` than `daytime_cbgs` .

See also appendix on algorithms for more information on how the home location algorithm has changed in historical data.

Privacy

To preserve privacy, we apply differential privacy techniques to the following columns: `visitor_home_cbgs` , `visitor_home_aggregation` , `visitor_daytime_cbgs` , `visitor_country_of_origin` , `device_type` , `carrier_name` . We have added Laplacian noise to the values in these columns. After adding noise, only attributes (e.g., a census block group) with at least two devices are included in the data. If there are between 2 and 4 visitors this is reported as 4.

Wondering where the device data used in Patterns comes from? See this FAQ.

Patterns Backfill

Backfill is when we take our most recent version of Places (i.e., Core + Geometry) and run our visit attribution algorithm backward in time to generate a new history of “backfilled” Patterns. It happens no more than twice a year, most commonly in July ☀️ and in December ❄️ as needed. This means historical Patterns will only be present for all POI that were released at a specific date; that is, historical Patterns data exist in the July 2021 backfill for all POIs that were available in the July 2021 release. As of July 2021, this includes industrial POIs introduced in early 2021 and corporate offices.

Check out our FAQs for more details and the Release Notes for guidance on the latest backfill.

Column Name Detailed Descriptions

`street_address`

- We implement a number of steps to clean, validate and standardize `street_address` .
- You should expect `street_address` to be title-cased, consistent, and friendly for human reading. Please send us your feedback if you see otherwise.
- If you care about street addresses as much as we do, we also have more specific address columns to split out address components. These are optional and available upon request for future deliveries.
    - `primary_number`

- `street_predirection`
- `street_name`
- `street_postdirection`
- `street_suffix`

`city`

- In the US, all centroids (latitudes/longitudes) are referenced against a geospatial file of city boundaries as defined by the US Census Bureau (browse the boundaries here). In edge cases, the preferred city name in the address line reflects a pre-annexed city name, and we try our best to preserve those city names where possible.
- In Canada, city names are the output of normalized address strings from POI sources.
- In Great Britain, city names are the output of normalized address strings from POI sources, but in edge cases, we allow POIs to have a null city name as long as `region` is populated. The `region` column in Great Britain refers to county boundaries, and counties are a decent alternative to cities for geographic filtering.

`region`

- When `iso_country_code == US`, then this is the US state or territory.
- When `iso_country_code == CA`, then this is the Canadian Province or territory.
- When `iso_country_code == GB`, then this is the United Kingdom county.

`postal_code`

- When `iso_country_code == US`, then this is the US 5 digit zip code.
- When `iso_country_code == CA`, then this is the Canadian postal code in the form of a 3 digit Forward Sortation Area (FSA), a space, and the 3 digit Local Delivery Unit (LDU).
- When `iso_country_code == GB`, then this is the British postal code. Learn more about Great Britain postal code precision here.

`raw_visit_counts`

These are the aggregated raw counts that we see visit the POI from our panel of mobile devices. The duration of the visit must last at least 4 minutes to count as a visit to a given POI.

`visits_by_day`

- This is an array of visits on each day in the month.
- We are breaking up days based on local time.
- See also: How do I work with Patterns columns that contain JSON?

`visitor_home_cbgs`

- These are the home census block groups of the visitors to the POI.
- For each census block group, we show the number of associated *visitors* (as opposed to the number of *visits*). If visits by home cbg is desired, we recommend taking the *visitors* from each CBG and multiplying by the average visits/visitor (i.e., `raw_visit_counts / raw_visitor_counts`) as an approximation.
- We do not have a home census block group for each visitor and not each visitor originates from the US. The number of US visitors listed in the `visitor_country_of_origin` column represents the total number of visitors which we have determined originate from the US versus Canada.
- See also: How do I work with Patterns columns that contain JSON?, Determining Home Location and Privacy.

`visitor_daytime_cbgs`

- These are the daytime census block groups of the visitors to the POI.
- For each census block group, we show the number of associated *visitors* (as opposed to the number of *visits*).
- See also: How do I work with Patterns columns that contain JSON?, Determining Home Location and Privacy.

`visitor_country_of_origin`

- These are the countries of origin of the visitors to the POI.
- See also: How do I work with Patterns columns that contain JSON?, Determining Home Location and Privacy.

`distance_from_home`

- This is the median distance from home to the POI in meters for the visitors we have identified a home location.

- This is calculated by taking the haversine distance between the visitor's home geohash-7 and the location of the POI for each visit. We then take the median of all of the home-POI distance pairs.
- If we have fewer than 5 visitors to a POI, the value will be null.
- We do not adjust for visits - each visitor is counted equally.

`median_dwell`

- This is the median of the minimum dwell times we have calculated for each of the visits to the POI.
- We determine the minimum dwell time by looking at the first and last ping we see from a device during a visit. This is a minimum dwell because it is possible the device was at the POI longer than the time of the last ping.
- It is possible to have a minimum dwell of 0 if we only saw 1 ping and determined the visit based on factors such as wifi.

`bucketed_dwell_times`

- This is a dictionary of different time spans and the number of visits that were of each duration.
- The time spans are in minutes.
- Data delivered from December 2020 onward, including all backfilled historical data, contains the following bins: { "<5", "5-10", "11-20", "21-60", "61-120", "121-240", ">240"}
- Data delivered prior to December 2020 contains the following bins: { "<5", "5-20", "21-60", "61-240", ">240"}
- See also: How do I work with Patterns columns that contain JSON?

`related_same_day_brand`

- These are the brands that the visitors to this POI also visited, on the same day that they visited the POI. The number mapped to each brand is an indicator of how highly correlated a POI is to a certain brand. The value is a simple percent of POI visitors that visited the other brand on the same day.
- Note that the way this column was calculated changed in data released in July 2021 and onward. At that time, all historical data since 2018 was re-computed with the new way of calculation. There should be no inconsistency if you are using the latest data.
- Only the first 20 brands are returned.
- See also: How do I work with Patterns columns that contain JSON?

`related_same_month_brand`

These are the brands that the visitors to this POI visited over the course of the month. Interpreted and calculated in the same way as `related_same_day_brand` .

- See also: How do I work with Patterns columns that contain JSON?

`popularity_by_hour`

- This is an array of visits seen in each hour of the day over the course of the month.
- Local time is used.
- If a visitor stays for multiple hours, an item in the array will be incremented for each hour during which the visitor stayed. This means that if you sum the numbers in the `popularity_by_hour` array the sum will likely be greater than the amount shown in the `raw_visit_counts` column (since the `raw_visit_counts` counts a multiple-hour visit as one visit).

`popularity_by_day`

- This is a mapping of the day of the week to the total number of visits seen on each day of the week during the course of the month.
- Local time is used.
- See also: How do I work with Patterns columns that contain JSON?

`carrier_name`

- This is a premium column that maps wireless carrier names to the number of visitors to the POI whose device uses that wireless carrier.
- Below is a breakdown of our panel of devices by wireless carrier as of the July-2020 release:
- See also: How do I work with Patterns columns that contain JSON? and Privacy.

Carrier	Count	Ratio
Verizon	10,303,871	35.64%

Carrier	Count	Ratio
AT&T	7,267,146	25.13%
T-Mobile	7,129,894	24.66%
Sprint	3,685,988	12.75%
Altice	323,221	1.11%
C-Spire	204,800	0.71%

Known Data Issues or Artifacts

See page in Developer Tools for more information.

Algorithms

Home Algo v2 “Incremental Updates”

- This is the current production version of the home algorithm, and applies to all new releases of Patterns.
- Home Algo v2 went into production in May 2020 (see Rollout of Home Algo v2 below).
- Each day:
  - All pings for all devices are clustered, filtered to only include clusters during nighttime hours (6pm - 7 am local time), and the frequency of clusters per unit space (e.g., 3 clusters in census block group A, 4 clusters in census block group B, etc.) are computed. The census block group with the most clusters is internally recorded as the daily “winner”. We also record how many pings are observed in all of the clusters at the winning location.
  - Then, any device that has not had a home location updated within 30 calendar days is “updated” by re-computing the common nighttime location (see next).
- To compute a common nighttime location:
  - Lookback over the previous 6 weeks of daily “winning” common nighttime locations, and identify the most frequently “winning” common nighttime location. This is the new home location for the next 30 calendar days.
  - We also compute an internal “confidence score” based on the number of unique hours and unique days for which pings were observed at this home location.
  - Only devices with a confidence score > a threshold are considered high confidence home locations.
  - The new home location is recorded internally, along with its confidence rating, along with the date.
  - The new home location (or lack thereof) immediately takes effect for that device.
  - The home location for this device will not be re-computed for 30 calendar days.
  - Note: Only high confidence home locations are used in SafeGraph Patterns and Social Distancing Metrics products and reflected in the home_panel_summary.csv. Devices that do not have a high confidence home location are treated as if the home location is unknown.
- **New Devices**. When a new device enters the panel, there is no historical data. A new device must accumulate at least 5 unique days of data (this may be > 5 calendar days if the device does not generate pings every day) before it is eligible to determine a high-confidence home. After 5 unique days of data are collected the home location will be computed, and it will not be recomputed for another 30 calendar days.

Home Algo v1 “Monthly Batched”

- This is an older version of the home algorithm. It was used in production for data generation prior to May 2020 and is now retired.
- The Home Algo was run 1x / month on the 1st of each month.
- At the start of each month, all pings for the previous 6 weeks were analyzed for each device. These pings were aggregated into clusters, and then filtered to only include clusters during nighttime hours (6pm - 7am local time). We identify the most common nighttime location for each device based on the frequency of clusters. The winning location census block group (CBG) was reported as the “home” for that device for the subsequent month.
  - We also recorded the number of unique hours and unique days for which pings were observed at the common nighttime location, and these numbers were used to form a “confidence score”. Only devices with a confidence score > a threshold were considered high confidence home locations, and only high confidence home locations were used in SafeGraph products.
  - Devices that do not have a high confidence home location were treated as if the home location is unknown.
- Known Issues with Home Algo v1:

- New devices entering the panel were not assigned a home until the first of the following month, and therefore all new devices across a month were added to the panel simultaneously. This created discontinuities at month boundaries.
- Devices leaving the panel across the month caused the apparent sample size to slowly decay across the course of each month, and then regenerate at the start of the month suddenly.
- These issues were fixed with Home Algo v2.

Rollout of Home Algo v2 in 2020

Home Algo v1 was used until May 2020. Forward-facing data generation switched over to use Home Algo v2 on the following dates:

- SDM v2.1: May 18th 2020. (Note SDM v2.1 began on 5/10/20 using Home Algo v1)
- Monthly Places Patterns: May 2020 data (released in June 2020).
- Weekly Places Patterns: Week of 5/04/20

Historical Data and Backfills

For historical backfills of data before May 2020, a hybrid algorithm is used, rather than back-computing Home Algo v2. Hybrid in this instance means using Home Algo v1 before a certain time, and then using Home Algo v2 afterward.

Home Algo v1 is applied to the following time periods:

- Backfills of Weekly Places Patterns from Jan 1 2018 through May 2020
- Backfills of Monthly Patterns from Jan 1 2018 through May 2020
- Backfills of Social Distancing Metrics v2.1 (Jan 1 2019 - Dec 31 2019)

Modified Home Algo v1 during the Dec 2020 backfill

The main shortcoming of Home Algo v1 is that it failed to identify the home locations of new devices in a timely manner, and added all new devices to the panel at the start of each month.

- In the Dec 2020 backfill, we attempted to work around this limitation by modifying Home Algo v1 slightly. We allowed back-propagated Home Algo v1 to use data “from the future” (e.g., 30 days following the first of the month); however, this ended up being incomparable to previous backfilled data and so we have reverted to using the "standard" Home Algo v1 (no forward-looking) in the July 2021 backfill.
- Although this is not ideal, it is consistent with the data that were provided at that time, and so for comparability we have decided to maintain the non-forward-looking Home Algo v1 historically for now prior to May 2020.

Why does SafeGraph not use Home Algo v2 for historical data prior to May 2020?

- Backfilling Home Algo v2 on the history of SafeGraph data has non-trivial compute costs because the computation must occur on every day historically for every device.
- When feasible, we will back-propagate Home Algo v2 so that the historical data are the most accurate possible (i.e., homes added on a rolling basis, rather than at the first of each month). This will result in an inconsistency with previous backfills, so we will provide sufficient notice prior to implementation.

Video Overview