California Self-Driving Vehicle Data Analysis

Kyle Markwardt, DB

The Great Seal of the California Department of Motor Vehicles

State of CA Regulations Info

Regulations v1.0 from 2014-2019 “The Test Drive Era”:

- Limited approval as a developing technology
- Inconsistent data reporting standards
- Data available in scholarly research or by request from DMV
    - Not readily publicly available.

Regulations v2.0 from 2020- Today:

- Standardized reporting structure
- Data publicly available from DMV website
- Special numbered permit is issued by company, with special vehicle registration requirements
- 2x Permit Types: Autonomous Testing with a Driver & Driverless Testing
- Each company has slightly different restrictions. All require ‘decent’ weather and reasonable speeds

The Great Seal of the California Department of Motor Vehicles

About California Regulations v2.0 Raw Data

- Data is self-reported, submitted on standardized forms, and compiled by the State into .csv. Covers important categories like monthly miles, date and location of disengagement, driver present
- Yearly Disengagement Reports and Mileage Reports provided by State
- Raw data covers 18 different .csv files, ~20,000 overall rows, ~250,000 overall cells, ~2,000 vehicles
- Each individual Crash Report PDF provided by State (but uncompiled) (~700)

Processing California Regulations v2.0 Raw Data

- Varieties of highly similar terms (“Highway”, “Freeway”) (“Computer Disengagement” “AV System Disengagement”) standardized into main categories
- When possible, VIN problems were cleaned by finding correct VIN in a different report
- Data was combined & cross-referenced to enable clear multi-year insights for VINs, Companies, Dates, etc

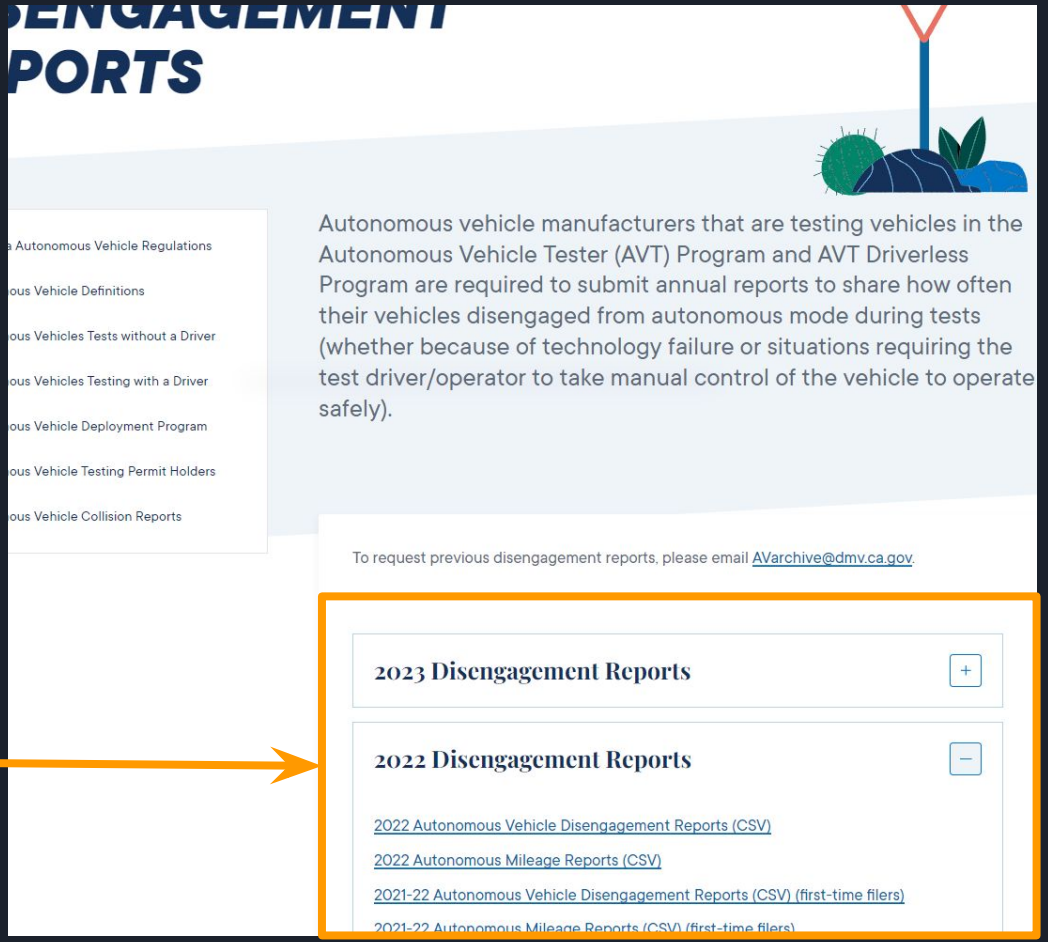
Data Sources

Mileage Reports

Disengagement Reports

- Vehicle in autonomy
- Comes out
- Regardless of driver

CSVs per year



Report Form

STATE OF CALIFORNIA DMV A Public Service Agency

ANNUAL REPORT OF AUTONOMOUS VEHICLE DISENGAGEMENT

Instructions: Print as many pages as needed. Submit completed report to: Department of Motor Vehicles, Autonomous Vehicles Branch, 2415 1st Avenue, MS D405, Sacramento, CA 95818

SECTION 1 — MANUFACTURER INFORMATION

NAME OF MANUFACTURER				AVT NUMBER
BUSINESS MAILING ADDRESS	CITY	STATE	ZIP CODE	TELEPHONE NUMBER ( )

SECTION 2 — DISENGAGEMENT EVENT DETAIL Use one row for each disengagement event.

DATE	VIN NUMBER	DISENGAGEMENT INITIATED BY (AV System, Test Driver, Remote Operator, or Passenger)	DISENGAGEMENT LOCATION (Interstate, Freeway, Highway, Rural Road, Street, or Parking Facility)	DESCRIPTION OF FACTS CAUSING DISENGAGEMENT *
	VEHICLE IS CAPABLE OF OPERATING WITHOUT A DRIVER ☐YES ☐NO	DRIVER PRESENT ☐YES ☐NO		
	VEHICLE IS CAPABLE OF OPERATING WITHOUT A DRIVER ☐YES ☐NO	DRIVER PRESENT ☐YES ☐NO		
	VEHICLE IS CAPABLE OF OPERATING WITHOUT A DRIVER ☐YES ☐NO	DRIVER PRESENT ☐YES ☐NO		
	VEHICLE IS CAPABLE OF OPERATING WITHOUT A DRIVER ☐YES ☐NO	DRIVER PRESENT ☐YES ☐NO		
	VEHICLE IS CAPABLE OF OPERATING WITHOUT A DRIVER ☐YES ☐NO	DRIVER PRESENT ☐YES ☐NO		
	VEHICLE IS CAPABLE OF OPERATING WITHOUT A DRIVER ☐YES ☐NO	DRIVER PRESENT ☐YES ☐NO		
	VEHICLE IS CAPABLE OF OPERATING WITHOUT A DRIVER ☐YES ☐NO	DRIVER PRESENT ☐YES ☐NO		
	VEHICLE IS CAPABLE OF OPERATING WITHOUT A DRIVER ☐YES ☐NO	DRIVER PRESENT ☐YES ☐NO		

** Additional information regarding the causes of the disengagement may be submitted as an attachment. If an attachment is provided, indicate the specific attachment number for the disengagement event.*

OL 311 R (REV. 7/2020) WWW

Print | Clear Form

SECTION 3 — DISENGAGEMENT AND AV MILES SUMMARY PER VEHICLE

VIN Number	Annual Total of Disengagements	Total Number of Miles Tested in Autonomous Mode (December ____ to November ____)												
		December	January	February	March	April	May	June	July	August	September	October	November	ANNUAL TOTAL
														0
														0
														0
														0
														0
														0
														0
														0
														0
														0
														0
														0
														0
														0
														0
														0

SECTION 4 — ACKNOWLEDGMENT

PRINTED NAME OF AUTHORIZED REPRESENTATIVE			TITLE
SIGNATURE X			DATE SIGNED
STREET ADDRESS	CITY	STATE	ZIP CODE
EMAIL ADDRESS	FAX NUMBER ( )		TELEPHONE NUMBER ( )

Print | Clear Form

OL 311 R (REV. 7/2020) WWW

Data Processing

Data Cleaning and QA

Drop Columns

```
# What are "Unnamed" colums? Probably useless "reserved"
# This and below are from kyle, and deal with the initial mileage_df
unnamed_cols = ['Unnamed: 17', 'Unnamed: 18', 'Unnamed: 19', 'Unnamed: 20', 'Unnamed: 21']
for check in unnamed_cols:
    non_null_count = mileage_df[check].notna().sum()
    print(f'Number of non values in {check}: {non_null_count}')

# Empty. Drop 'em
mileage_df = mileage_df.drop(columns=unnamed_cols)
mileage_df.columns
```

✓ 0.0s

```
Number of non values in Unnamed: 17: 0
Number of non values in Unnamed: 18: 0
Number of non values in Unnamed: 19: 0
Number of non values in Unnamed: 20: 0
Number of non values in Unnamed: 21: 0
```

```
Locations: 15 - ['Express Way', 'Freeway', 'Freeway ', 'HIGHWAY', 'HIghway', 'Highway', 'Interstate', 'Parking Facility', 'Parking facility', 'Rural Roac
Operators: 51 - ['AIMOTIVE INC.', 'APOLLO AUTONOMOUS DRIVING USA LLC', 'APPLE INC.', 'ARGO AI, LLC', 'AURORA OPERATIONS, INC.', 'AUTOX TECHNOLOGIES, INC'

Cleaned locations: 8 - ['EXPRESS WAY', 'FREEWAY', 'HIGHWAY', 'INTERSTATE', 'PARKING FACILITY', 'RURAL ROAD', 'STREET', 'URBAN']
Cleaned operators: 38 - ['AIMOTIVE', 'APOLLO', 'APOLLO AUTONOMOUS DRIVING USA', 'APPLE', 'ARGO AI', 'AURORA OPERATIONS', 'AUTOX TECHNOLOGIES', 'BOSCH', '

Original number of operators:  50
Actual number of unique operators:  32
'AIMOTIVE'
'APOLLO'
'APPLE'
'ARGO AI'
'AURORA OPERATIONS'
'AUTOX'
'BOSCH'
'CRUISE'
```

Data Sources

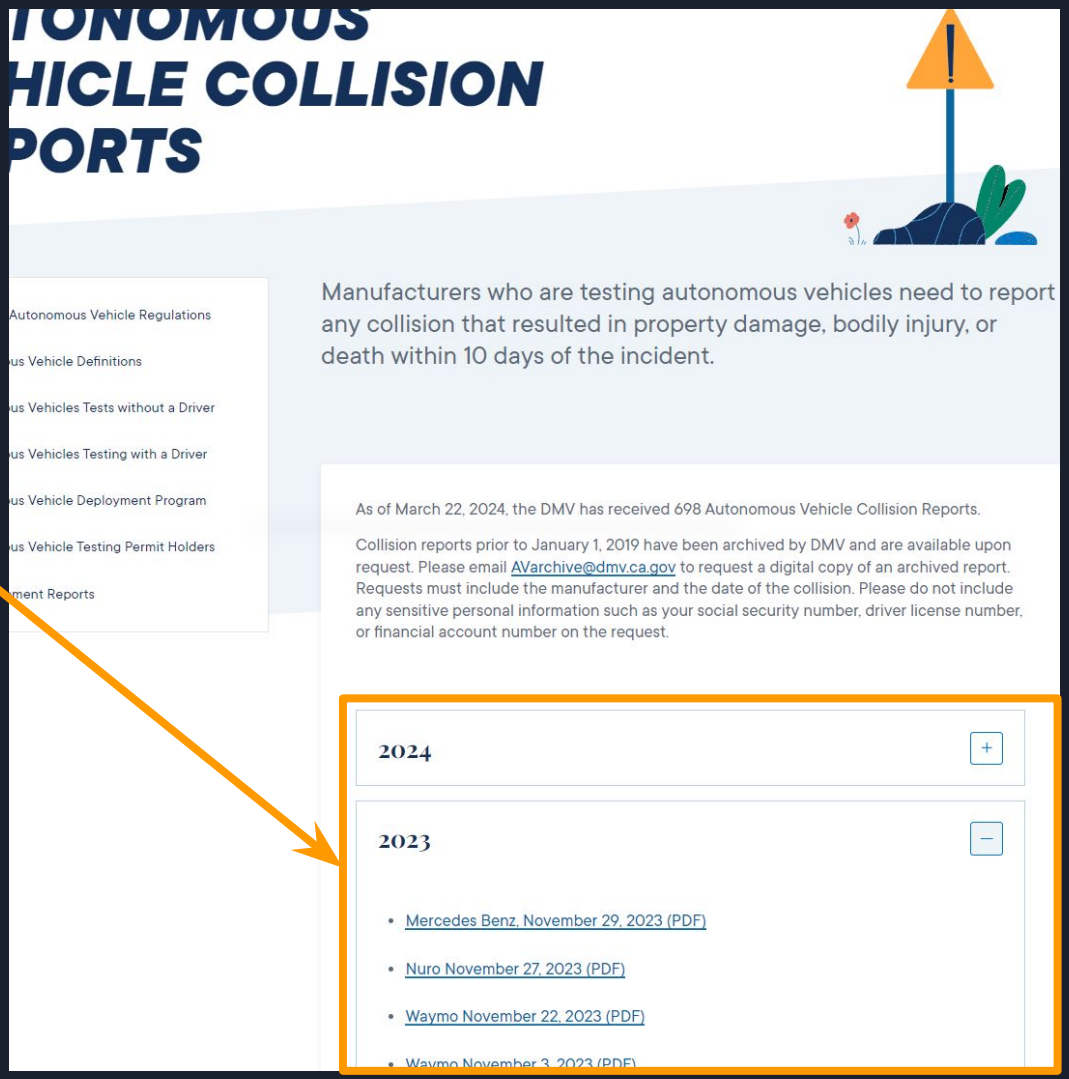
Collision reports

Individual PDFs per incident

Access problem:

- No API
- Parse failed (href)
- Clicker failed (JS)

Manual download



PDF Mining

1. Download PDFs
2. Zip for memory (568 files)
3. Read fields

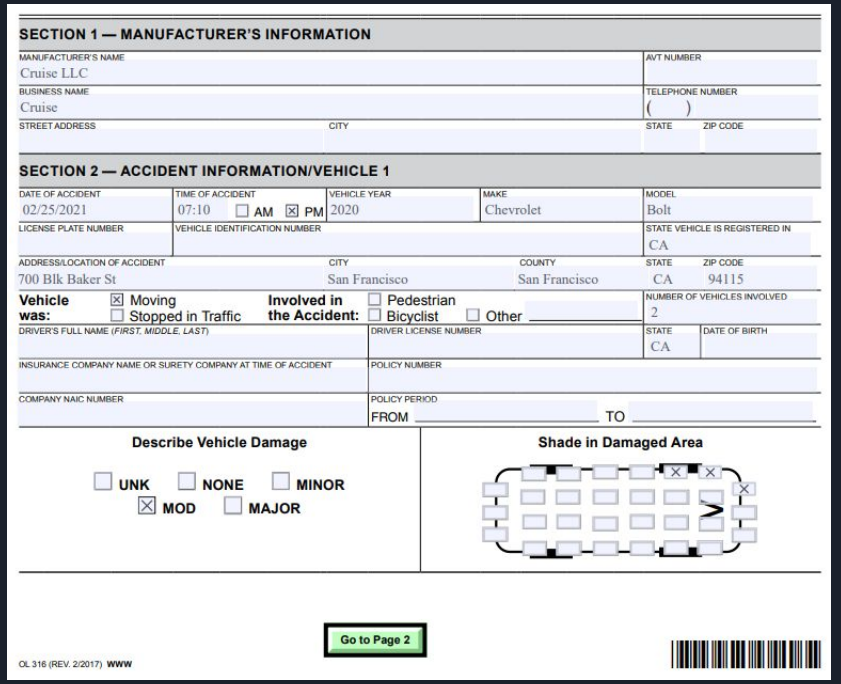
   [pdfplumber]

4. Select wanted fields
5. Create DF

[[field, alt, value]
[field, alt, value]
[field, alt, value]
...
[field, alt, value]
[field, alt, value]]

[[field, alt, value]
[field, alt, value]
...
[field, alt, value]]

	alt1	alt2	alt3
PDF1	value1	value2	value3
…	…	…	…
PDF568	value1	value2	value3



SECTION 1 — MANUFACTURER'S INFORMATION

MANUFACTURER'S NAME: Cruise LLC
AVT NUMBER:
BUSINESS NAME: Cruise
TELEPHONE NUMBER: ( )
STREET ADDRESS:
CITY:
STATE:
ZIP CODE:

SECTION 2 — ACCIDENT INFORMATION/VEHICLE 1

DATE OF ACCIDENT: 02/25/2021
TIME OF ACCIDENT: 07:10 ☐ AM ☒ PM
VEHICLE YEAR: 2020
MAKE: Chevrolet
MODEL: Bolt
LICENSE PLATE NUMBER:
VEHICLE IDENTIFICATION NUMBER:
STATE VEHICLE IS REGISTERED IN: CA
ADDRESS/LOCATION OF ACCIDENT: 700 Blk Baker St
CITY: San Francisco
COUNTY: San Francisco
STATE: CA
ZIP CODE: 94115
Vehicle was: ☒ Moving ☐ Stopped in Traffic
Involved in the Accident: ☐ Pedestrian ☐ Bicyclist ☐ Other
NUMBER OF VEHICLES INVOLVED: 2
DRIVER'S FULL NAME (FIRST, MIDDLE, LAST):
DRIVER LICENSE NUMBER:
STATE: CA
DATE OF BIRTH:
INSURANCE COMPANY NAME OR SURETY COMPANY AT TIME OF ACCIDENT:
POLICY NUMBER:
COMPANY NAIC NUMBER:
POLICY PERIOD: FROM TO

Describe Vehicle Damage

☐ UNK ☐ NONE ☐ MINOR ☒ MOD ☐ MAJOR

Shade in Damaged Area

Go to Page 2

OL 316 (REV. 2/2017) WWW

Scraping User-Entered data from PDF

```python
# This function read the filed name, alternate field name, and field values
def parse_field_helper(form_data, field, prefix=None):
    """ appends any PDF AcroForm field/value pairs in `field` to provided `form_data` list

        if `field` has child fields, those will be parsed recursively.
    """
    resolved_field = field.resolve()
    field_name = '.'.join(filter(lambda x: x, [prefix, resolve_and_decode(resolved_field.get("T"))]))
    if "Kids" in resolved_field:
        for kid_field in resolved_field["Kids"]:
            parse_field_helper(form_data, kid_field, prefix=field_name)
    if "T" in resolved_field or "TU" in resolved_field:
        # "T" is a field-name, but it's sometimes absent.
        # "TU" is the "alternate field name" and is often more human-readable
        # your PDF may have one, the other, or both.
        alternate_field_name = resolve_and_decode(resolved_field.get("TU")) if resolved_field.get("TU") else None
        field_value = resolve_and_decode(resolved_field["V"]) if 'V' in resolved_field else None

        # Remove non-printable characters and trailing spaces - This affects every Cruise file.
        field_name = ''.join(char for char in field_name if char.isprintable()).strip()
        alternate_field_name = ''.join(char for char in alternate_field_name if char.isprintable()).strip() if alternate_field_name else None
        field_value = ''.join(char for char in field_value if char.isprintable()).strip() if field_value else None

        form_data.append([field_name, alternate_field_name, field_value])
```

Returns a list of tuples, e.g:
[[field_name_1, alt_name_1, value_1], … [field_name_n, alt_name_n, value_n]]

Filter for what is interesting

```python
# Define filtering criteria
def find_important_tuples(tuple, search_condition):
    # Check if the first element of the tuple is in the search condition list
    return tuple[0] in search_condition
```

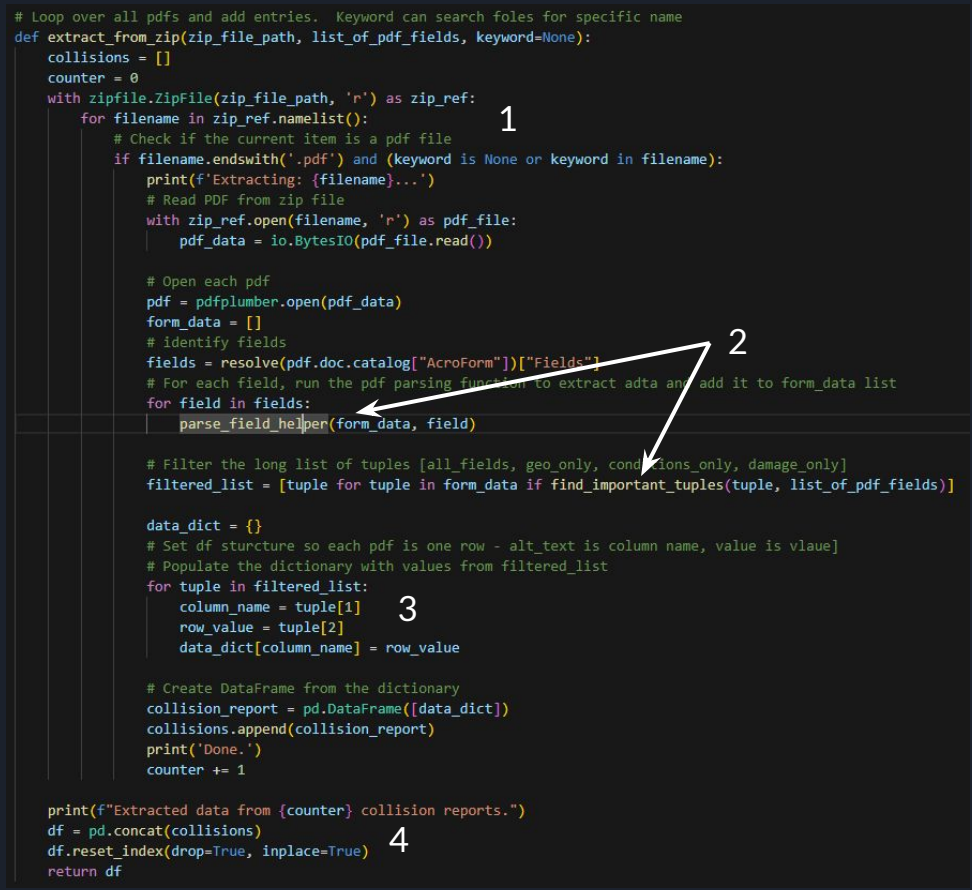
Manually checked whole list for fields we wanted (~½)

Created list to filter and some sub-lists

```python
all_fields = ['MANufACTuRERS NAME','BuSINESS NAME',
              'DATE Of ACCIDENT','Time of Accident','AM','PM',
              'VEhICLE YEAR','MAkE','MODEL',
              'section 2  accident infoRmation.0','section 2  accident infoRmation.1.0','section 2  accident infoRmation
              'Moving', 'Stopped in Traffic', 'Pedestrian', 'Bicyclist', 'undefined', 'Other',
              'NuMBER Of VEhICLES INVOLVED',
              'Unknown','None','minor','Moderate','major',
              'Left Rear 1','Rear Bumper','Right Rear 1','Left Rear 2','Left Rear 3','Right Rear 2','Right Rear 3',
              'Left Rear Passenger 1','Left Rear Passenger 2','Right Rear Passenger 1','Right Rear Passenger 2',
              'Left Rear Passenger 3','Left Rear Passenger 4','Right Rear Passenger 3','Right Rear Passenger 4',
              'Front Driver Side 1','Front Driver Side 2','Front Passenger Side 1','Front Passenger Side 2',
              'Front Driver Side 3','Front Driver Side 4','Front Passenger Side 3','Front Passenger Side 4',
              'Left Front Corner 1','Left Front Corner 2','Right Front Corner 1', 'Right Front Corner 2',
              'Left Front Corner 3','Front Bumper','Right Front Corner 3',
              'Moving_2', 'Stopped in Traffic_2','Pedestrian_2','Bicyclist_2','undefined_2','Other_2',
              'ADDRESS_2.1.0.1','Autonomous Mode','Conventional Mode',
              'WEATHER A 1','WEATHER A 2','WEATHER B 1','WEATHER B 2','WEATHER C 1','WEATHER C 2',
              'WEATHER D 1','WEATHER D 2','WEATHER E 1','WEATHER E 2','WEATHER F 1','WEATHER F 2','WEATHER G 1','WEATHER
              'LIGHTING A 1','LIGHTING A 2','LIGHTING B 1','LIGHTING B 2','LIGHTING C 1','LIGHTING C 2',
              'LIGHTING D 1','LIGHTING D 2','LIGHTING E 1','LIGHTING E 2',
              'ROADWAY A 1','ROADWAY A 2','ROADWAY B 1','ROADWAY B 2','ROADWAY C 1', 'ROADWAY C 2','ROADWAY D 1','ROADWA
              'ROAD CONDITIONS A 1','ROAD CONDITIONS A 2','ROAD CONDITIONS B 1','ROAD CONDITIONS B 2', 'ROAD CONDITIONS
              'ROAD CONDITIONS D 1','ROAD CONDITIONS D 2','ROAD CONDITIONS E 1', 'ROAD CONDITIONS E 2', 'ROAD CONDITIONS
              'ROAD CONDITIONS G 1', 'ROAD CONDITIONS G 2', 'ROAD CONDITIONS H 1', 'ROAD CONDITIONS H 2',
              'MOVEMENT A 1','MOVEMENT A 2','MOVEMENT  B 1', 'MOVEMENT  B 2','MOVEMENT C 1','MOVEMENT C 2', 'MOVEMENT  D
              'MOVEMENT  E 1','MOVEMENT  E 2', 'MOVEMENT  F 1', 'MOVEMENT  F 2', 'MOVEMENT  G 1','MOVEMENT  G 2', 'MOVEM
              'MOVEMENT  I 1', 'MOVEMENT  I 2', 'MOVEMENT J 1', 'MOVEMENT J 2', 'MOVEMENT  K 1', 'MOVEMENT  K 2', 'MOVEM
              'MOVEMENT  M 1', 'MOVEMENT  M 2','MOVEMENT  N 1', 'MOVEMENT  N 2', 'MOVEMENT  O 1', 'MOVEMENT  O 2',
              'MOVEMENT  P 1', 'MOVEMENT  P 2', 'MOVEMENT  Q 1', 'MOVEMENT  Q 2', 'MOVEMENT  R 1', 'MOVEMENT  R 2',
              'TYPE A 1', 'TYPE A 2', 'TYPE B 1', 'TYPE B 2','TYPE C 1','TYPE C 2','TYPE D 1','TYPE D 2','TYPE E 1','TYP
              'OTHER A YES','OTHER A NO','OTHER B','OTHER C','OTHER D','OTHER E','OTHER F','OTHER G',
              'OTHER H YES','OTHER H NO','OTHER I','OTHER J','OTHER K','OTHER L']
```

PDF Megafunction

1. Loop through all files in location (zip)
2. Call the scrape and filter functions
3. Make each pdf into one row
4. Concatenate




```python
# Loop over all pdfs and add entries.  Keyword can search foles for specific name
def extract_from_zip(zip_file_path, list_of_pdf_fields, keyword=None):
    collisions = []
    counter = 0
    with zipfile.ZipFile(zip_file_path, 'r') as zip_ref:
        for filename in zip_ref.namelist():
            # Check if the current item is a pdf file
            if filename.endswith('.pdf') and (keyword is None or keyword in filename):
                print(f'Extracting: {filename}...')
                # Read PDF from zip file
                with zip_ref.open(filename, 'r') as pdf_file:
                    pdf_data = io.BytesIO(pdf_file.read())

                # Open each pdf
                pdf = pdfplumber.open(pdf_data)
                form_data = []
                # identify fields
                fields = resolve(pdf.doc.catalog["AcroForm"])["Fields"]
                # For each field, run the pdf parsing function to extract adta and add it to form_data list
                for field in fields:
                    parse_field_helper(form_data, field)

                # Filter the long list of tuples [all_fields, geo_only, conditions_only, damage_only]
                filtered_list = [tuple for tuple in form_data if find_important_tuples(tuple, list_of_pdf_fields)]

                data_dict = {}
                # Set df sturcture so each pdf is one row - alt_text is column name, value is vlaue]
                # Populate the dictionary with values from filtered_list
                for tuple in filtered_list:
                    column_name = tuple[1]
                    row_value = tuple[2]
                    data_dict[column_name] = row_value

                # Create DataFrame from the dictionary
                collision_report = pd.DataFrame([data_dict])
                collisions.append(collision_report)
                print('Done.')
                counter += 1

    print(f"Extracted data from {counter} collision reports.")
    df = pd.concat(collisions)
    df.reset_index(drop=True, inplace=True)
    return df
```

Result!

Additional challenges:

CRUISE!! - non-printable characters
EVERY field

Checkboxes:

- X = “”, “Yes”
  - Change to 1
- Blank = None, NaN, “Off”
  - Change to 0

Timestamps - 12hr/24hr

```
path_to_zip = "data/collisions/Collision_PDFs.zip"
# read in all the data once.
all_collisions_df = extract_from_zip(path_to_zip, all_fields)
all_collisions_df.head()
```

[6] ✓ 4m 16.6s

```
Extracting: Aimotive_091619.pdf...
Done.
Extracting: Apollo-OL316-062923-Redacted.pdf...
Done.
Extracting: Apollo-OL316-090623-Redacted.pdf...
Done.
Extracting: Apollo-OL316-101623-Redacted.pdf...
Done.
Extracting: Apple_012624.pdf...
Done.
Extracting: Apple_022123.pdf...
Done.
Extracting: apple_081921.pdf...
Done.
Extracting: apple_082321.pdf...
Done.
Extracting: Apple_091919.pdf...
Done.
Extracting: Apple_101022.pdf...
Done.
Extracting: Apple_120621.pdf...
Done.
Extracting: Apple_122022.pdf...
Done.
Extracting: Apple_OL316_061422_Redacted.pdf...
...
Done.
Extracting: Zoox-OL316-101223-Redacted.pdf...
Done.
Extracted data from 568 collision reports.
```

Self-Driving Trends in California

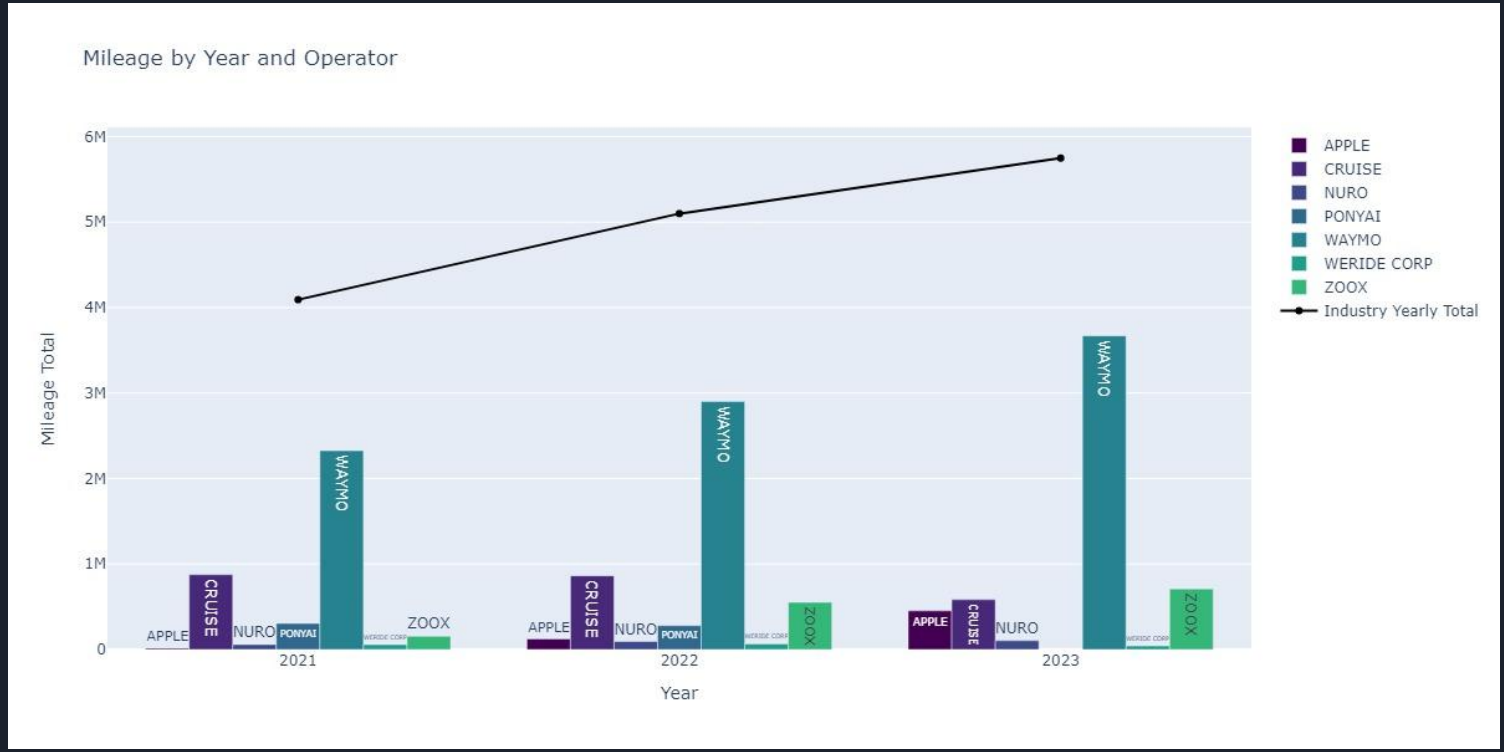
2021-2023 Total Autonomous Miles in CA:

14,938,551

US-20 trips (3300m) equivalent:

Over 4,500

Top Company Annual Mileage



Things break down

Average Rate of Autonomous Collisions

All Company aggregate:

Autonomous collisions:

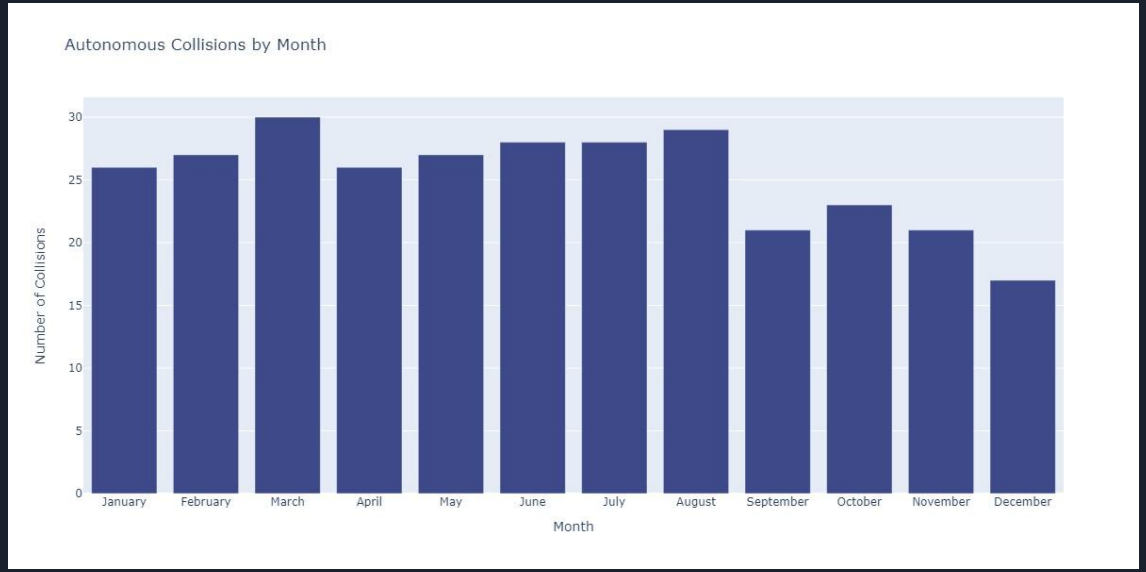
~25/mo

2023 autonomous mileage:

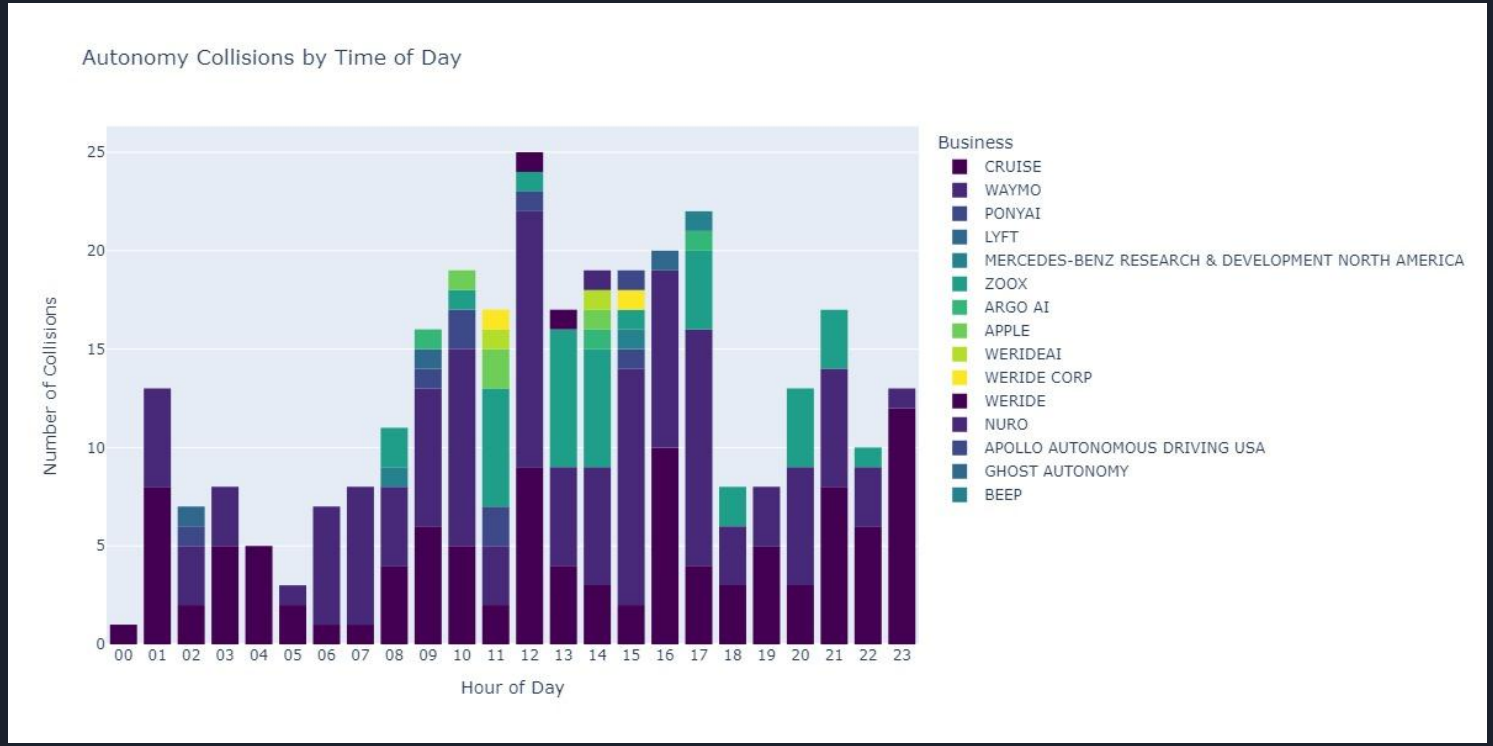
5.75M mi

Per 100M mi:

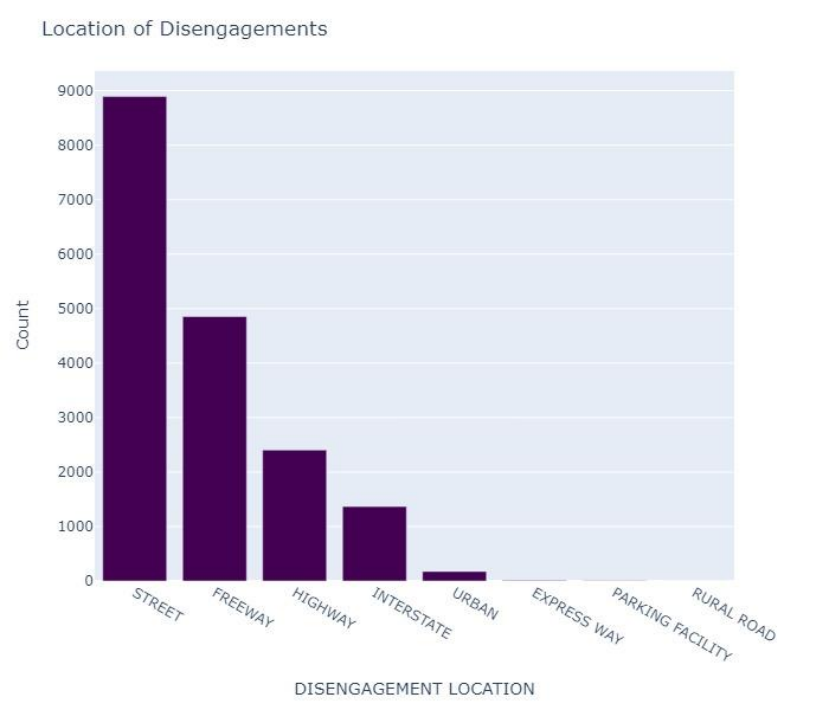
~5200

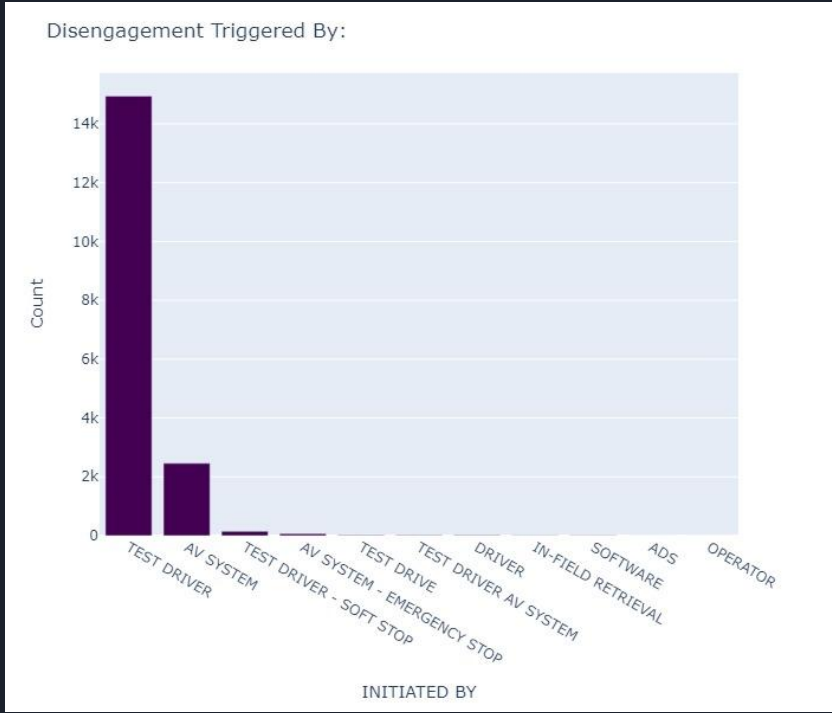


What Time Did Collisions Occur?

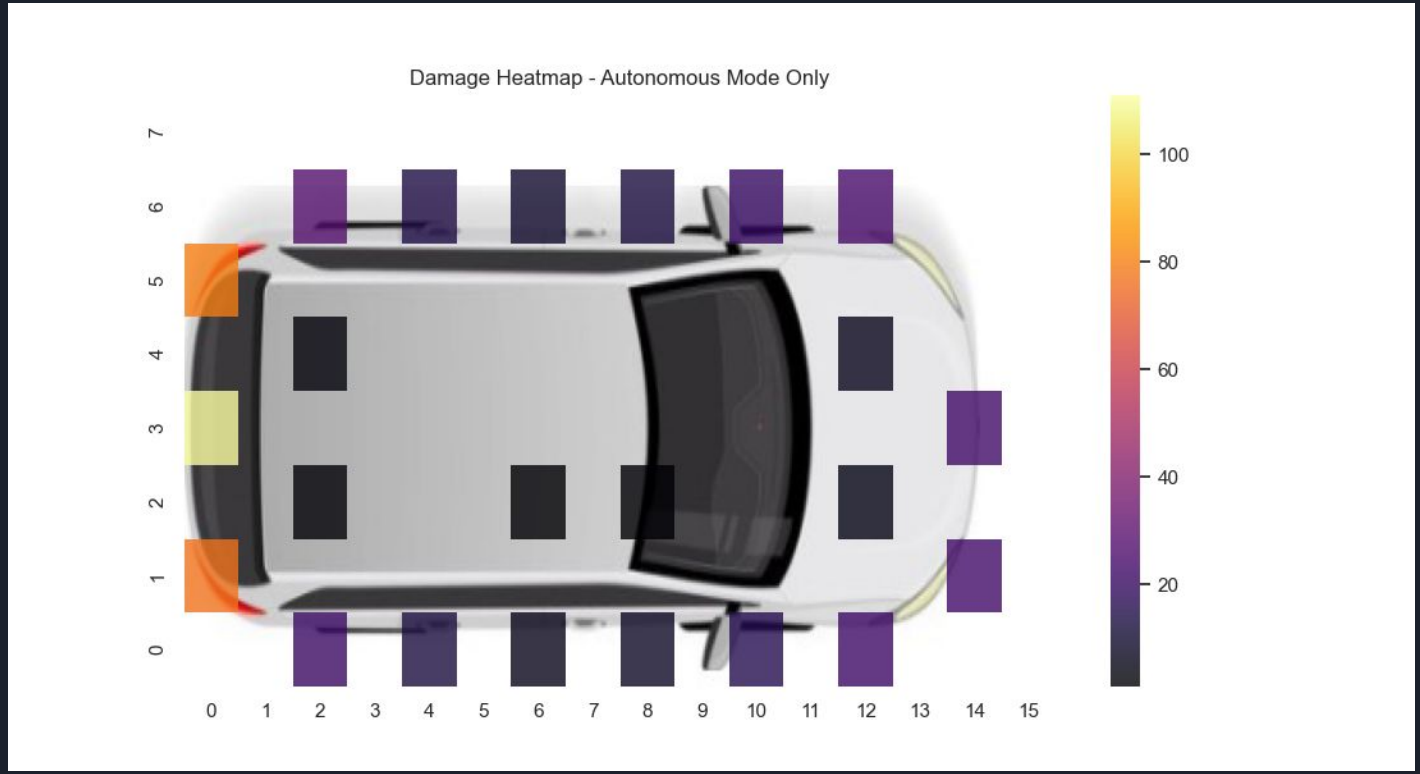


Disengagement Insights





Collision heatmap



Waymo Gets There*

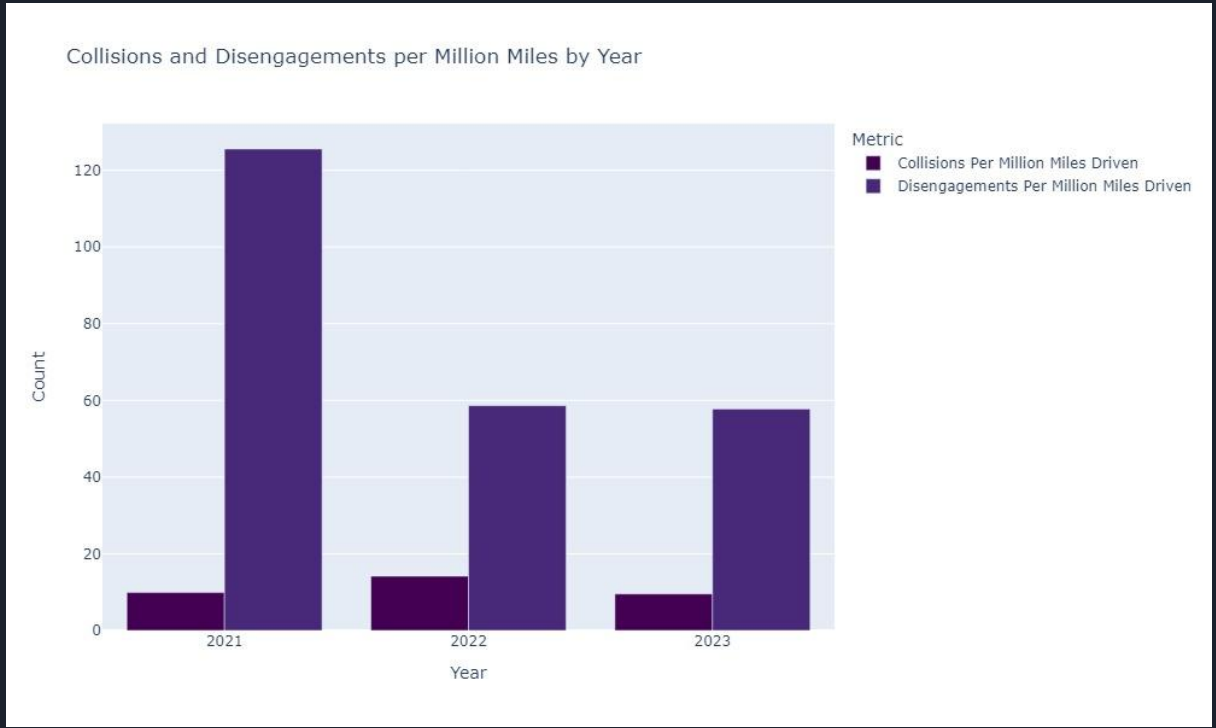
NHTSA - 2021

Crashes/100M miles:

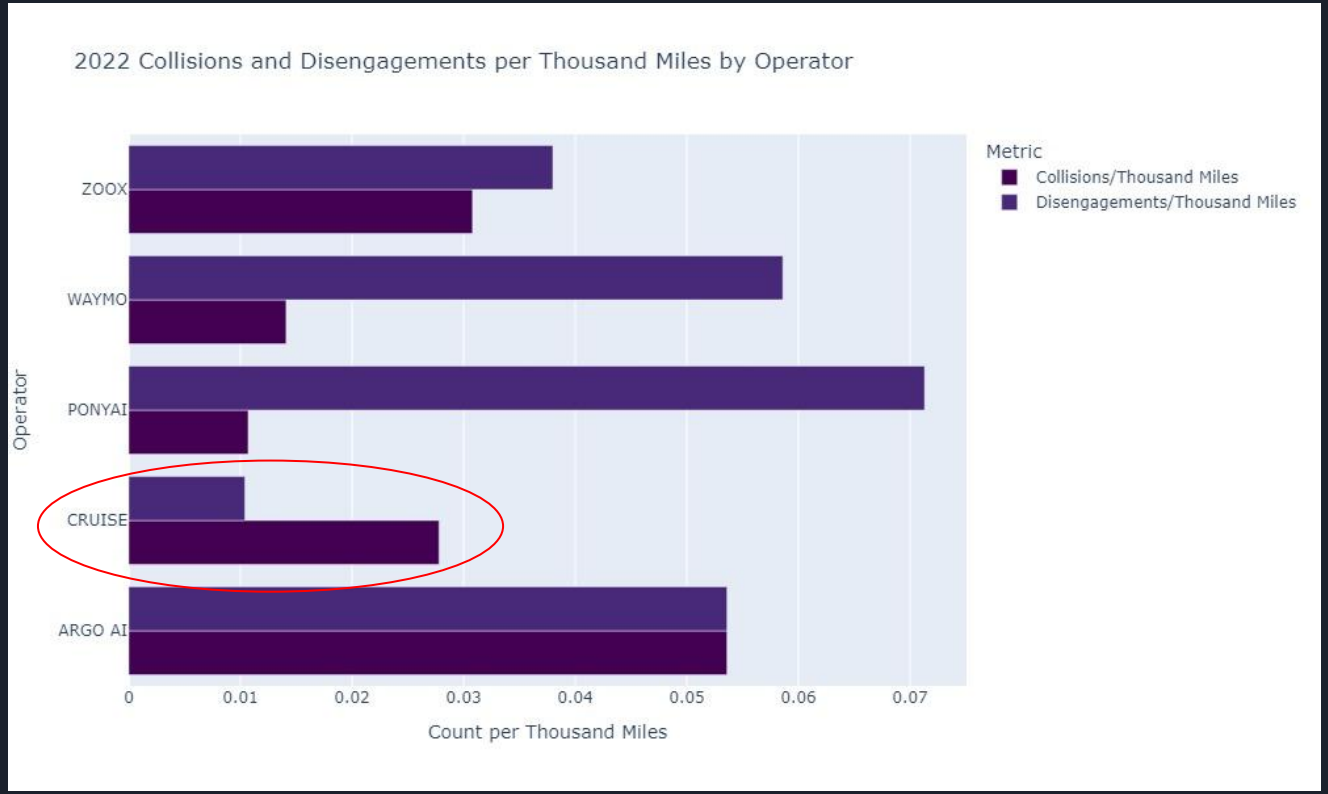
195

Waymo:

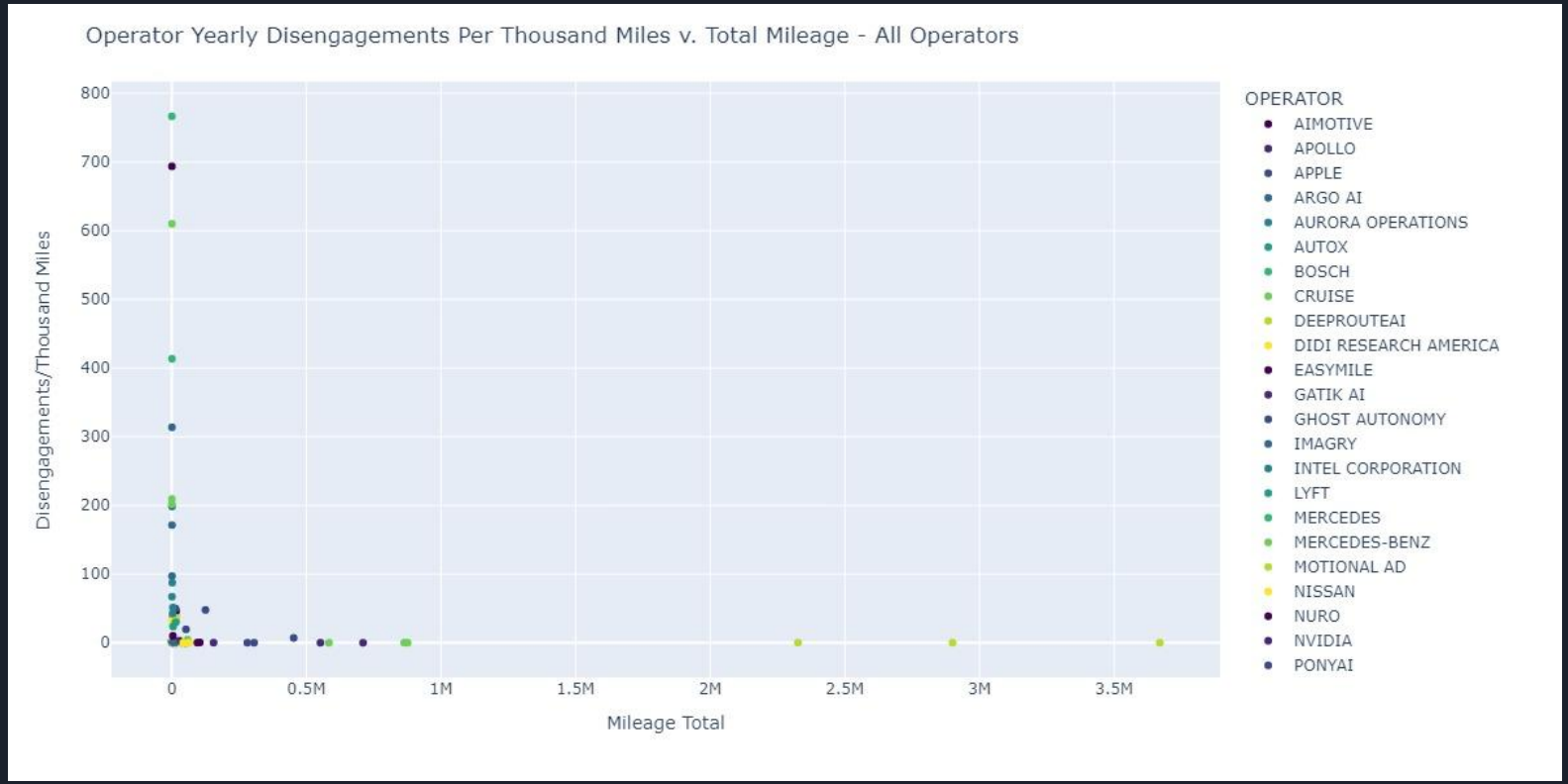
~1400



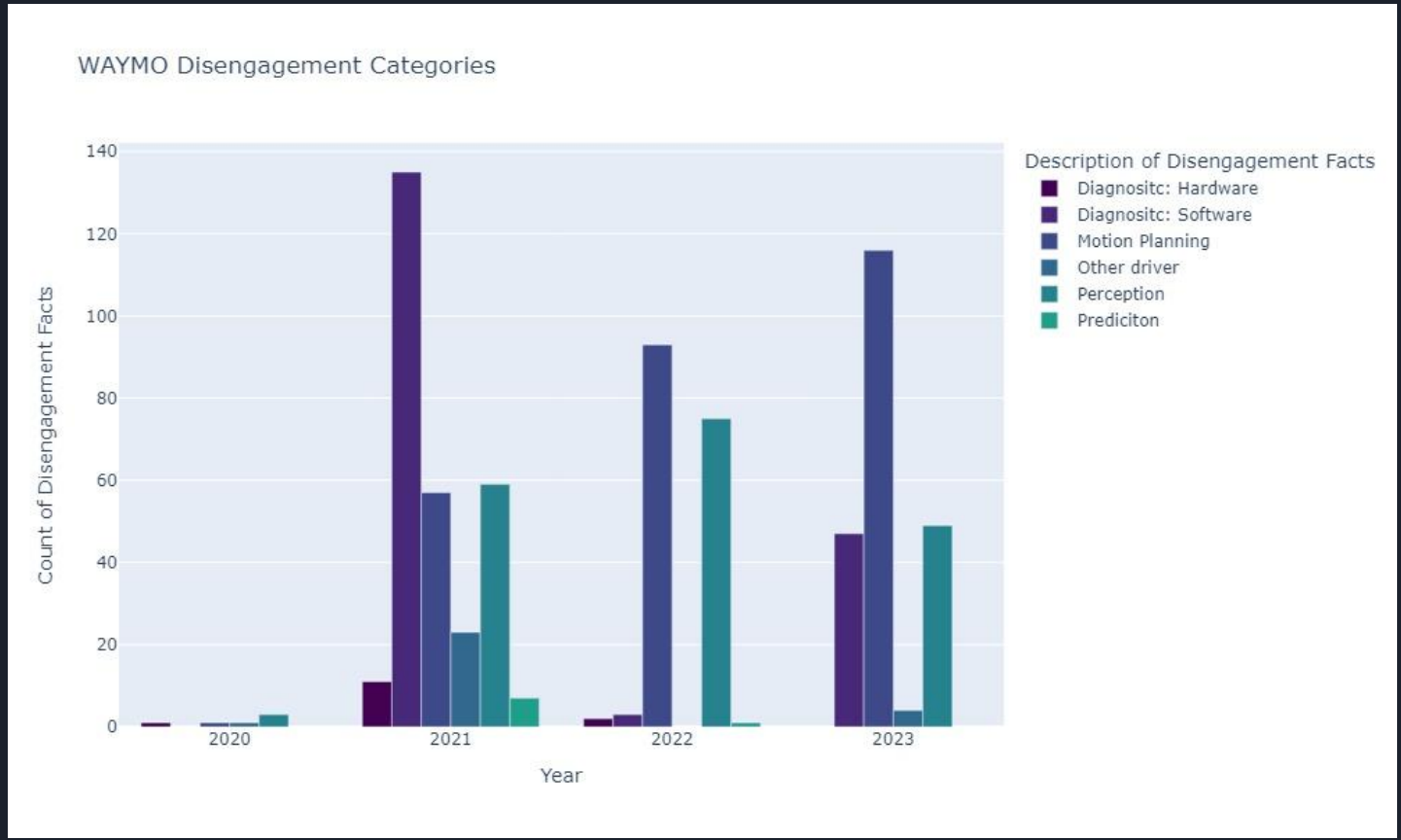
Broader View



Disengagement Rates vs Miles Logged



Waymo - Disengagement Causes



Disengagement Insights

State of California

BAD:

- No one appears to be checking this data
  - Duplicate VINs, wrong format
  - Entries like “same”
  - Total lack of data validation
- Unlikely outliers:
  - E.g., Cruise lack of disengagement reports. Eye opening delta.
    - Could it be real?
- What role do crashes play in relationship to the processed dataset?
  - The State hasn’t compiled crashes in an accessible raw data format.

GOOD:

- Allowed Waymo to get there
- Other states more friendly to self-driving because of CA, but are not publishing this data
- Surprisingly accessible data for the DMV

Wish List:
Zippy & Pokey

Data is granular enough (VIN level) we would have liked to tell the story of the highest and lowest mileage vehicles. Several vehicles have interesting stories:

```
Zippy VIN is SADHW2S13M1616427
Zippy is a WAYMO vehicle, which drove 55273.1 miles.
Pokey VIN is ~5071
Pokey is a ZOOX vehicle, which drove 0.0 miles.
```

Other Directions We Could Go

- Other states
  - Some operators (e.g., Aurora, do not primarily work in CA.)
- Geo-map the collision locations
  - We were unable due to the nature of the locations given (x street near y blvd, for example)
- Other environments (non-urban)
- NLP and Disengagement Reports:
  - Each disengagement report (16,000+) has a short written narrative from the company. What would NLP analysis of this reveal?