Outline
Exec Summary
    Major Findings
Acknowledgements of Project Team
    ITA: Tiffany Chu, Hunter Owens (Project Manager), Maryam Abbassi, Ian Rose

    DCP: Ashley Atkinson, Roy Samaan, Jonathan Chiu, Betty Dong, Tom Tran, Mony Patel, David Terukina, DCP to provide names we missed

Tiffany To Do for final edit:
- Format footnotes for uniformity
- Check footnotes, some articles need extra citation (not the data sources)
- Make sure hyperlinks are blue in footnotes

DCP To Do:
- Tom to verify some sections that are tagged for him
- PMU team to write up applications of this work

Title page

Exec Summary

Acknowledgements

I. Introduction

The Department of City Planning (DCP) Performance Management Unit (PMU) submitted a project proposal to ITA's Data Science Federation (DSF) call for projects. DSF projects are typically paired with universities. Due to the complexity of the database and longer project timeline, beyond what students can reasonably finish in one semester, the ITA data science team felt it was better to do the project internally.

DCP records and tracks the entitlements in the Planning Case Tracking System (PCTS). PMU wanted to use data analytics and GIS tools to provide insights into whether various types of entitlements submitted were correlated with neighborhood or location characteristics. The "entitlement" is a request from a property owner for an exemption or exception to what is permitted under current land use regulations. The entitlement is then subject to planning review before being allowed to proceed. Tracking and reviewing these entitlements is one of DCP's core functions; PMU was interested in diving into the last decade's worth of data and gleaning key geographic and socioeconomic insights.

Once the ITA Data team started cleaning and processing the PCTS data, it was clear that this project needed a standardized data processing pipeline. Data analytics projects often suffer from reproducibility issues; it is hard to reproduce the same counts, trends, or results when different people answer the same question with the same dataset. It is imperative to have both a canonical data source (PCTS) and a standardized data processing pipeline. For this project, we had a canonical data source, but the project scope changed to accommodate a standardized data processing pipeline as one of the deliverables.

A. Project Scope

This analysis portion of the project answers the following questions:

(1) *Transit / geographic analysis for TOC suffix only*

What kind of entitlement activity occurs near high-quality transit?
Are TOC entitlements spatially concentrated near certain rail stations, rail lines, or bus corridors?

This is covered in **Section II** of the report.

(2) *Socioeconomic analysis across all entitlement suffixes*
How are socioeconomic characteristics correlated with various entitlement suffixes?

(a) Transit Oriented Communities (TOC) in-depth regression analysis

(b) Stand-alone alcohol sale (CUB) in-depth regression analysis.

This is covered in **Section III** of the report.

Establishing a data pipeline for this project meant familiarizing ourselves with how DCP generates its internal SQL queries for its internal PCTS reports. We also developed several tools that make this project reproducible and lay the groundwork for future data analytics projects DCP would undertake using the PCTS data. We created a Python package called `laplan` and use the Civis Analytics platform to automate the data cleaning pipeline and deploy dashboards. The data pipeline process and tools developed and used are covered in Section IV.

B. Data Sources

Table 1 lists all the data sources used to conduct the analysis. The vast majority of data came from open data portals, with the exception of DCP's PCTS database. All the data sources are documented in our data catalog and are stored in an S3 bucket.[1]

Table 1 Data Sources

Data	Source	Years Available	Data Used In
Entitlements	DCP - Planning Case Tracking System (PCTS)	2010-2019	All analyses
PCTS documentation	DCP website for prefixes, suffixes[2], DCP emailed PCTS codebook[3]		All analyses
Parcels	County of LA, Open Data Portal, Tax Assessor Parcels[4]	2006-2019	All analyses
Zoning (ZIMAS)	City of LA, DCP, GeoHub (geospatial open data portal)[5]	2019	All analyses
Zoning documentation	City of LA, DCP (emailed)[6] [7]		All analyses
TOC tiers and TOC parcels	City of LA, DCP (emailed)	2017	Geographic / transit analysis
Metro existing rail stations and rail lines	LA Metro Developer GIS Data[8], `Metro_Rail_Lines`, `Metro_Rail_Stations`	2019	Geographic / transit analysis

[1] To ensure that we have the data from open data portals even if URLs break, we save versioned copies in our S3 bucket. The `make mirror` command in our Makefile creates a versioned copy of the open data sources in the S3 bucket.
[2] https://planning.lacity.org/resources/prefix-suffix-report
[3] https://github.com/CityOfLosAngeles/planning-entitlements/blob/master/references/PCTS_TABLE_FIELDS.xlsx
[4] https://data.lacounty.gov/Parcel-/Assessor-Parcels-Data-2006-thru-2019/9trm-uz8i
[5] https://geohub.lacity.org/datasets/49ad06a6b8c945debbbea865b1832ee2_0
[6] https://github.com/CityOfLosAngeles/planning-entitlements/blob/master/references/Zoning_Types_Master_Table_092518.xlsx
[7] https://github.com/CityOfLosAngeles/planning-entitlements/blob/master/references/Zoning_Code_Summary_March2020.pdf
[8] https://developer.metro.net/bus-rail-gis-data/

Data	Source	Years Available	Data Used In
Metro planned rail stations	LA Times, Data Desk, LA Metro Maps[9], \`planned-crenshaw-stops\`, \`planned-purple-line-extension-stops\`, \`planned-regional-connector-stops\`	2015	Geographic / transit analysis
Metro bus stations and bus lines	LA Metro Developer GIS Data[10], \`BusStopServingLines1219\`, \`BusLineServingStops1219\`, \`RapidBRT1219\`,	2019	Geographic / transit analysis
Metrolink rail stations	City of LA, GeoHub[11]	2019	Geographic / transit analysis
Metrolink rail lines	Southern California Association of Governments, Open Data Portal[12]	2019	Geographic / transit analysis
General Transit Feed Specification (GTFS) for LA Metro	LA Metro, GTFS[13], \`gtfs_bus\`	2020	Geographic / transit analysis
GTFS for Big Blue Bus	Big Blue Bus, GTFS Service, \`current\`	2020	Geographic / transit analysis
GTFS for Culver City Bus	Open Mobility Data, January 2020[14]	2020	Geographic / transit analysis
City boundary	City of LA, GeoHub[15]	2020	All analyses
Census tract	City of LA, GeoHub[16]	2010	All analyses
American Community Survey, 5-year tables	US Census Bureau[17], tables downloaded: B01001, B02001, B19001, B25008, S1903, S0801, S0802	2010-2018 (used 2018 only)	Socioeconomic analysis

[9] https://github.com/datadesk/lametro-maps
[10] https://developer.metro.net/bus-rail-gis-data/
[11] https://geohub.lacity.org/datasets/6f6c4677365b4418bd585db2ef8e201f_186
[12] https://gisdata-scag.opendata.arcgis.com/datasets/ac623114be664c7488836db5c22e3566_0
[13] https://gitlab.com/LACMTA/gtfs_bus/-/raw/aef844c79c31c40ef751a3472b3882406307b05a/gtfs_bus.zip?inline=false
[14] http://transitfeeds.com/p/culver-city-bus/1057
[15] https://geohub.lacity.org/datasets/09f503229d37414a8e67a7b6ceb9ec43_7
[16] https://geohub.lacity.org/datasets/census-tracts-2010-population
[17] https://data.census.gov/cedsci/

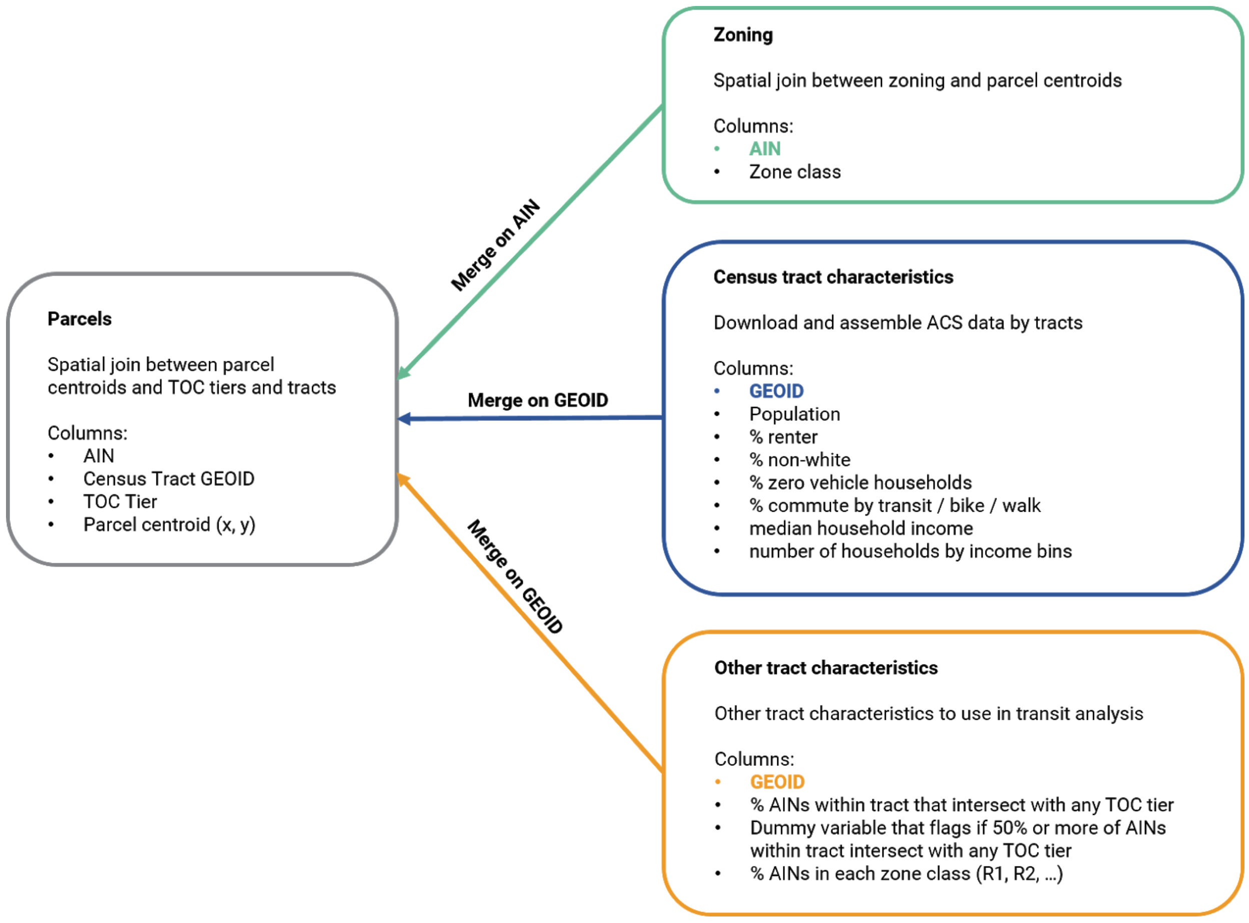
C. Methodology for Parcel Base File

Since parcels are the most granular level of data we have, we use it as the base file. We create several **crosswalks** that link parcels with other characteristics we want, such as, including census tract, zone class, and entitlements. These crosswalks are key "puzzle pieces" in constructing the analysis table.

The process for compiling the PCTS data, which comes from four tables in the PCTS database, is discussed more in detail in **Section IV - Data Pipeline**.

Figure 1 shows how we approached bringing all these various data sources together at the parcel-level. Parcel with tier information, parcel with tract GEOID, zoning, and other tract characteristics were constructed through spatial joins. The census tract characteristics table was constructed by downloading relevant American Community Survey (ACS) 5-year tables and merging them together.

Figure 1 Creating the Parcel Base File


Zoning
Spatial join between zoning and parcel centroids
Columns:
• AIN
• Zone class
Merge on AIN
Census tract characteristics
Download and assemble ACS data by tracts
Columns:
• GEOID
• Population
• % renter
• % non-white
• % zero vehicle households
• % commute by transit / bike / walk
• median household income
• number of households by income bins
Merge on GEOID
Parcels
Spatial join between parcel centroids and TOC tiers and tracts
Columns:
• AIN
• Census Tract GEOID
• TOC Tier
• Parcel centroid (x, y)
Merge on GEOID
Other tract characteristics
Other tract characteristics to use in transit analysis
Columns:
• GEOID
• % AINs within tract that intersect with any TOC tier
• Dummy variable that flags if 50% or more of AINs within tract intersect with any TOC tier
• % AINs in each zone class (R1, R2, …)

Source: Analysis by ITA Data Science

LA County Assessor Parcels

We used LA County parcel data from 2006-2019 to construct our full list of unique parcels. When we used just the 2019 parcel file, about 10% of the entitlements went unlinked. Including all the historical AINs allowed us to attach all the entitlements to parcel IDs. The trade-off in using the 2006-2019 AINs is that we only have parcel centroids, whereas the 2019 AIN file came with parcel polygons. Doing spatial joins between two sets of polygons is computationally expensive. Our spatial joins always involved finding a point-in-polygon, such as which parcel centroid fell in which tract polygon, making parcel centroids a reasonable choice.

Duplicate Assessor Identifier Numbers (AIN) were removed, but duplicates based on parcel centroids are kept. This means that the same parcel (based on the same X, Y point) can be linked to different AINs over time. Unsure of which AIN is actually used in PCTS, we keep them all. Parcels are then spatially joined to census tracts and TOC tiers using parcel centroids. Counting the number of unique parcels within a census tract is done by counting unique parcel centroids, not AINs; the same physical boundary of a parcel would only be counted once, regardless of how many AINs are associated with it.

Zoning

Zoning information is stored in ZIMAS. We use the version provided on GeoHub, with fewer columns than what DCP typically can access. DCP's Guide to Zoning String shows that the zoning string is made up of component parts. The zoning string contains information about prefix on (Q)ualified or (T)entative zone classifications, zone class, the height district, (D)evelopment limits, and specific plans and overlays applicable.

We parsed the zoning string to get the relevant component parts, and used a spatial join to attach zoning information to parcels. We also count the total number of unique parcels (based on parcel centroid) within a tract and calculate the percentages of parcels within the tract belonging to each zone class. There are 63 zone classes captured. Keeping the zone classes as disaggregated as possible provides the flexibility needed for the crosswalk to be used in multiple analyses.

The zone classes included are:

'A1', 'A2', 'A2P',

'C1', 'C1.5', 'C2', 'C4', 'C5', 'CM', 'CR',

'GW',

'HJ', 'HR',

'M', 'M1', 'M2', 'M3', 'MR1', 'MR2',

'NI',

'OS',

'P', 'PB', 'PF',

'R1', 'R1H1', 'R1P', 'R1R3', 'R1V1', 'R1V2', 'R1V3', 'R2', 'R2P', 'R3', 'R3P', 'R4', 'R4P', 'R5', 'R5P',

'RA', 'RAS3', 'RAS4',

'RD1.5', 'RD2', 'RD3', 'RD4', 'RD5', 'RD6', 'RE', 'RE11', 'RE15', 'RE20', 'RE40', 'RE9',

'RMP', 'RS', 'RU', 'RW1', 'RW2',

'RZ2.5', 'RZ3', 'RZ4', 'RZ5'

See **Section IV - Data Pipeline** and **Appendix A** for more detail.

Census Tract Characteristics

The American Community Survey (ACS) 5-year estimates has census tract-level socioeconomic characteristics. We downloaded all the tables using the Census API and processed all the data to fit our analysis needs in scripts (download data, clean data part 1, clean data part 2, subset for analysis, and assemble as a crosswalk). We were interested in socioeconomic characteristics such as the population, race/ethnicity, median income, zero vehicle households, renter-occupied households, and commute mode.

Table 2 lists the ACS tables that were downloaded and stored.[18] Although we downloaded years 2010 through 2018, we only used the 2018 data in our analysis. However, we cleaned the data in the same way for all years and accounted for any major data changes that may have occurred in 2014.

Table 2 ACS Tables

Outcome	ACS Table Name	Main Dataset	Years
median income	HOUSEHOLD INCOME IN THE PAST 12 MONTHS (INFLATION-ADJUSTED) (B19001) by income ranges and has supplemental tables for each race	ACS 5 yr	2010-2018
median income	MEDIAN INCOME IN THE PAST 12 MONTHS (INFLATION-ADJUSTED) (S1903) for race categories	ACS 5 yr Subject	2010-2018
% zero vehicle workers	MEANS OF TRANSPORTATION TO WORK BY SELECTED CHARACTERISTICS (S0802)	ACS 5 yr Subject	2010-2018
commute mode	COMMUTING CHARACTERISTICS BY SEX (S0801)	ACS 5 yr Subject	2010-2018
% renter-occupied	TOTAL POPULATION IN OCCUPIED HOUSING UNITS BY TENURE (B25008)	ACS 5 yr	2010-2018
% white or % non-white	RACE (B02001)	ACS 5 yr	2010-2018
% white, non-Hispanic or converse	SEX BY AGE (B01001) for race and ethnicity categories	ACS 5 yr	2010-2018

[18] A more comprehensive version of Table 2 is available at: https://github.com/CityOfLosAngeles/planning-entitlements/blob/master/references/Census_Tables.xlsx

Source: US Census Bureau, American Community Survey

See **Section IV - Data Pipeline** and **Appendix A** for more detail.

Other Tract Characteristics

We created a crosswalk containing other tract characteristics further derived from the zoning and Census data already gathered. This enabled us to use these tract characteristics across multiple analyses without generating it each time. This crosswalk was adapted and expanded through iterations, as we tried different ways to slice the data and found the characteristics that could be used across multiple analyses. The main categories of characteristics involved understanding the relationship between census tracts and TOC tiers and zoning.

Knowing how many parcels within a tract fell into one of the TOC tiers was one piece of information, but a further step was grouping a tract as being a TOC tract or not. A TOC tract was defined as 50% or more of the parcels within a tract fell into one of the TOC tiers.

Through some of our descriptive charts, we found that the majority of TOC entitlements occurred in just a handful of zone classes. Even though a much larger list of zone classes was eligible for the TOC entitlement, these “favorable” zone classes were a proxy for unobserved incentives that developers faced or operated under. Including these favorable zone classes as a predictor in our regression analysis proved extremely helpful.

II. Geographic / Transit Analysis

In 2016, Measure JJJ passed, which allowed for the zoning code to include a Transit Oriented Communities (TOC) program. DCP "create[d] TOC guidelines for all housing developments within a half-mile radius of a major transit stop", providing "a package of new incentives for building affordable housing near public transit (City of Los Angeles, Department of City Planning, Transit Oriented Communities Incentive Program)."[19] The goal of that policy was to encourage affordable housing equally across all bus / rail corridors. The entitlement activity in these corridors could be a proxy for increased affordable housing or denser housing in transit-rich areas.

One major question of interest was: **what kind of entitlement activity occurs near high-quality transit?** Of course, the TOC suffix could occur near high-quality transit, but perhaps, relative to other suffixes, TOC entitlements are only a small part of entitlement activity in those corridors. Another question to explore is **where *are* the spatial concentrations and where activity is lacking.** Which rail stations or bus corridors are they concentrated along? Those lacking entitlement would benefit from other policy and planning programs to encourage increased affordable housing.

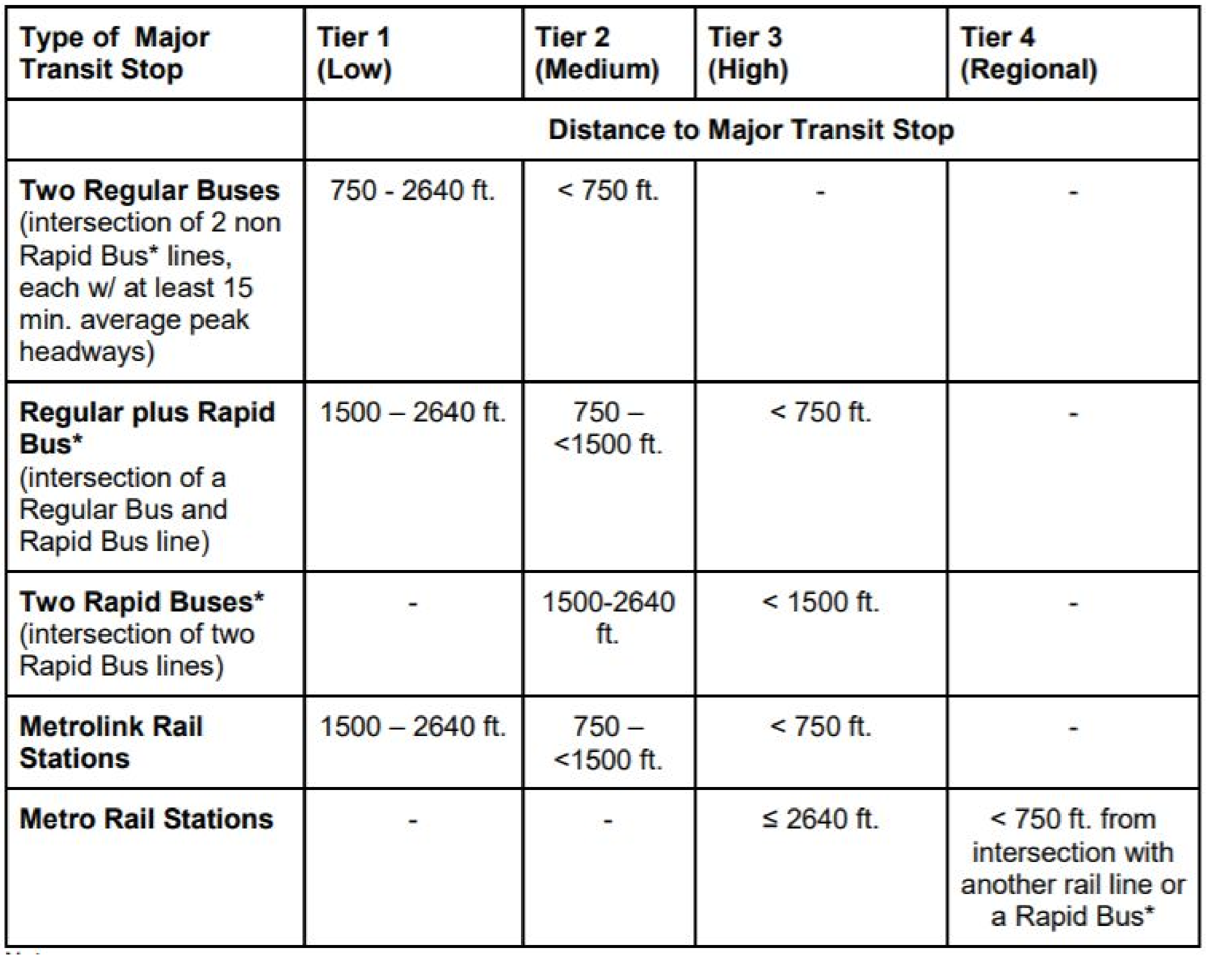
A. Methodology

DCP published TOC guidelines in November 2017 in response to the passage of Measure JJJ. The guidelines defined what was considered being near high-quality transit. Four tiers were established, with Tier 4 being the closest to the intersection of Metro rail stations and Rapid bus lines, and Tier 1 being much further out from Metro rail stations, the intersection of Regular and Rapid bus lines, or the intersection of two Regular bus lines. **Figure 2** shows how each tier was defined based on proximity to various types of transit.

Each tier is associated with base incentives, which allow for increased residential density (higher density for higher tiers), increased floor area ratios, and the decreased parking minimum requirements. There are also additional incentives, which further adjust setbacks, open space, lot coverage, and building heights. Each tier is also bundled with certain affordable housing requirements, which set forth certain percentages of units to be made available for extremely low or very low households. Taken together, the goal is to increase housing density near transit while making sure that a portion of that increase is set aside as affordable units.

[19] https://planning.lacity.org/plans-policies/transit-oriented-communities-incentive-program

Figure 2 TOC Affordable Housing Incentive Area Tiers



Type of Major Transit Stop	Tier 1 (Low)	Tier 2 (Medium)	Tier 3 (High)	Tier 4 (Regional)
	Distance to Major Transit Stop			
Two Regular Buses (intersection of 2 non Rapid Bus* lines, each w/ at least 15 min. average peak headways)	750 - 2640 ft.	< 750 ft.	-	-
Regular plus Rapid Bus* (intersection of a Regular Bus and Rapid Bus line)	1500 – 2640 ft.	750 – <1500 ft.	< 750 ft.	-
Two Rapid Buses* (intersection of two Rapid Bus lines)	-	1500-2640 ft.	< 1500 ft.	-
Metrolink Rail Stations	1500 – 2640 ft.	750 – <1500 ft.	< 750 ft.	-
Metro Rail Stations	-	-	≤ 2640 ft.	< 750 ft. from intersection with another rail line or a Rapid Bus*

Source: DCP, Transit Oriented Communities Guidelines, Chart 1

In order to quantify how much of the entitlement activity occurred near certain rail stations or bus lines, we needed to reconstruct the TOC tiers using the General Transit Feed Specification (GTFS) data for LA Metro, Santa Monica Big Blue Bus, and Culver CityBus routes. GTFS is a standard format for transit agencies to report the bus schedules for each route. The guidelines state that only bus lines with at least an average of 15 minute headway should be counted. We allow for an extra 20% buffer and err on the side of selecting ineligible bus routes, to make sure we catch all bus routes. We note that the DASH bus service is also included in DCP's list of transit agencies, but is excluded from our analysis. However, DASH buses that serve the downtown core are often co-located near other major bus routes, located close to Metro, Big Blue Bus, and Culver CityBus stops, and the entitlements would still be captured otherwise. The GTFS data allows us to calculate AM and PM weekday peak frequencies for bus routes, and select only the bus routes that meet this 15 minute cut-off.[20]

Next, buffers around the bus stops and rail stations are drawn. DCP does have the boundaries of the four TOC tiers as well as the list of parcels that are TOC-eligible, which was compiled with additional verification. We use spatial joins to attach the tier to each parcel. Parcels that are attached to multiple tiers are assigned to the higher tier (with more attractive incentives). Then,

[20] This GTFS work is documented in this script and this notebook.

We attach the bus stop(s), bus line(s), and rail stations(s) to each parcel. At this point, the same parcel can appear multiple times if it is near multiple bus stop(s) and rail station(s).[21]

Only parcels that fall within the TOC tiers are TOC-eligible, and we focus on the entitlement cases that took place in those areas. In terms of PCTS entitlements, we include cases from October 2017 through March 2020. Cases with the prefixes ENV (environmental), PAR (pre-application review), and ADM (administrative) are excluded. If a case contains "TOC" suffix, it is counted as a TOC entitlement, even if other suffixes are also requested. Entitlements without the "TOC" suffix are counted as Non-TOC entitlements.

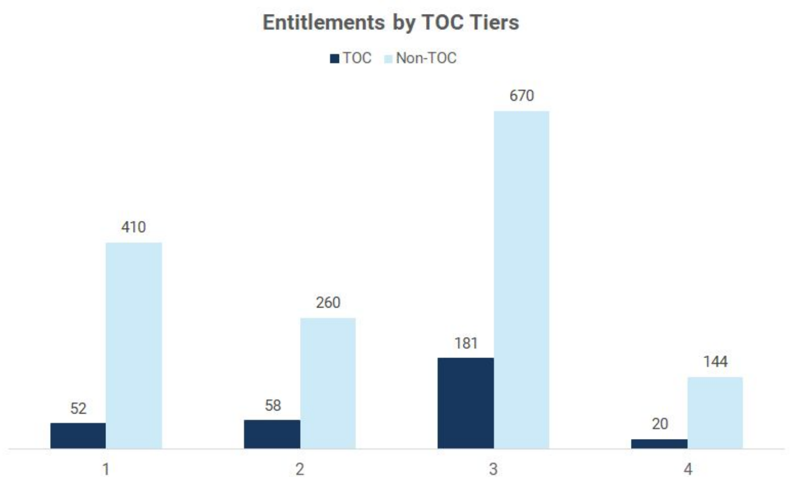
B. Analysis Findings

By Tier and Zone Class

First, we look at the distribution of TOC and Non-TOC entitlements across the TOC tiers. Of course, TOC entitlements can only occur in one of the TOC tiers. Non-TOC entitlements can occur inside or outside the TOC tiers. Entitlements that ask for verification of TOC-eligibility are excluded from the analysis.

Figure 3 shows that 58% of the TOC entitlements across the four eligible tiers occur in Tier 3. Being <750 ft of a Regular + Rapid, <1,500 ft of two Rapid lines, <750 ft of Metrolink rail station, or <=2,640 ft of a Metro rail station all qualify as Tier 3. Tier 3 has the biggest area in terms of eligible zones than other tiers, leading the tier to capture also the majority of Non-TOC entitlements. In terms of the fraction of entitlements that were TOC within a tier, it was Tier 3 (21%) and Tier 2 (18%) with the highest proportions of TOC to Non-TOC entitlements.

Figure 3 Entitlements by TOC Tiers



Source: PCTS, TOC Tiers; analysis by ITA Data Science

[21] An individual bus stop that has several bus routes passing through would be listed as the same bus stop ID. But, bus stops even 20 feet apart would have separate IDs. For example, there are two unique bus stops on Flower & 7th St. One bus stop has Metro bus routes 438, 448, and 534. The second bus stop, 50 feet away, has Metro Silver Line and Metro route 460.

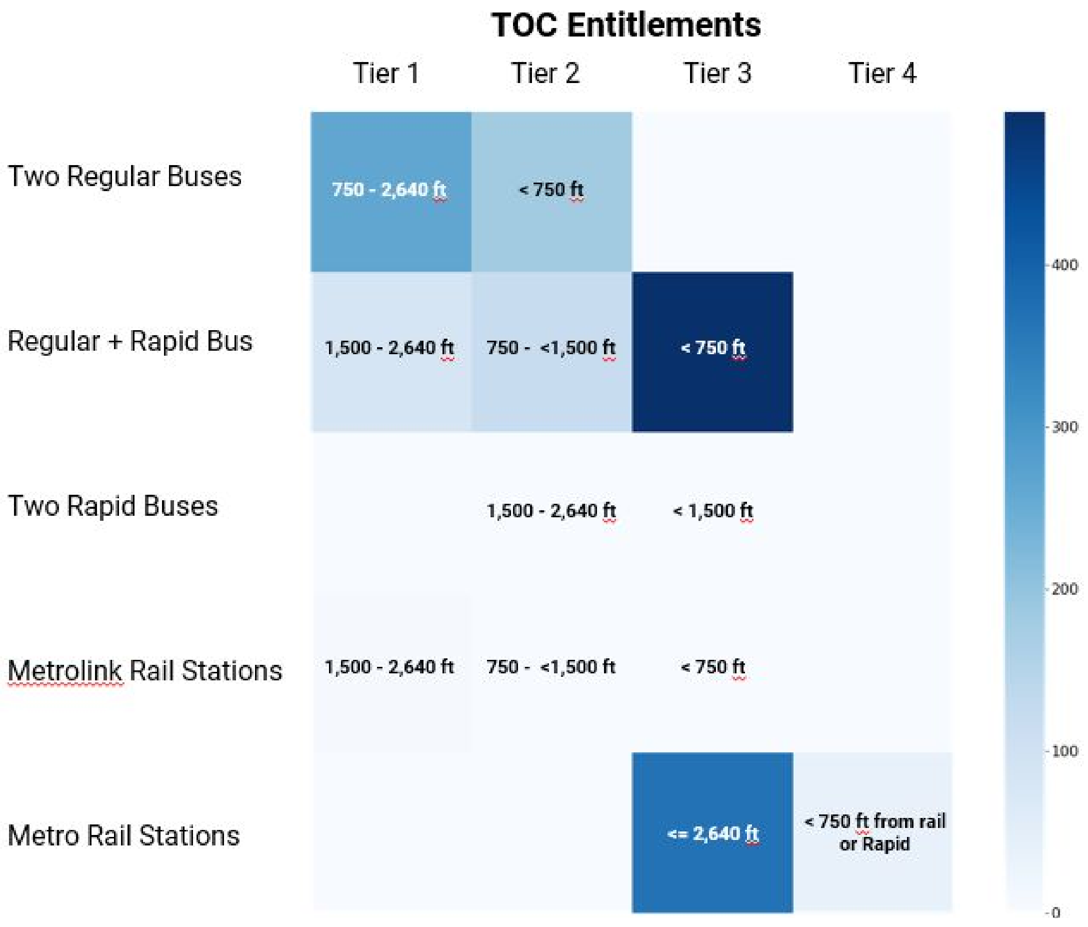
Figure 4 shows the number of TOC entitlements according to which rule was used to determine the tier assignment. Rather than only knowing that an entitlement was Tier 3, we identify which rule or rules with which the parcel is associated. That is, was the parcel was <750 ft of a Regular + Rapid and/or <1,500 ft of two Rapid lines and/or <750 ft of Metrolink rail station and/or <=2,640 ft of a Metro rail station? Certainly, multiple rules could be applicable. For example, a parcel in the downtown core near a Metro rail station would likely also be near the intersection of two Rapid bus lines, since bus stops tend to be co-located near rail stations for better connectivity. In that case, if the parcel had a TOC entitlement, it would be counted once for being near a Metro rail station, and a second time for being near the intersection of two Rapid lines.

The rules that generated the most TOC entitlements are being located within 750 ft of the intersection of Regular + Rapid bus (Tier 3), being within a 0.5 mile of a Metro rail station (Tier 3), and being between 750 ft - 2,640 ft of the intersection of two Regular buses (Tier 1). Tier 3 had the most number of TOC entitlements, but most of these occurred near where a Regular + Rapid bus intersected and near Metro rail stations. Unsurprisingly, TOC entitlements within 0.5 mile of Metro rail stations are popular, as much of the public transit literature agrees people will comfortably undertake a quarter to half mile walk to reach transit, and are even willing to walk up to 0.7 mi to access heavy rail.[22] [23] [24] ADD SENTENCE ABOUT METROLINK TOO. NONE? OR FEW CASES?

Interestingly, few occurred near where two Rapid buses intersected, which could be due to the relative scarcity of when that occurs compared to when Rapid buses intersect with Regular buses. Rapid buses tend to run along major corridors, such as Wilshire Blvd, Santa Monica Blvd, Olympic Blvd, Venice Blvd, Sepulveda Blvd, Vermont Av, and Western Av. Many of these are parallel routes, and there are few intersections between the east-west routes with the north-south routes.

[22] https://safety.fhwa.dot.gov/ped_bike/ped_transit/ped_transguide/ch4.cfm
[23] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4941821/
[24] https://media.metro.net/images/service_changes_transit_service_policy.pdf

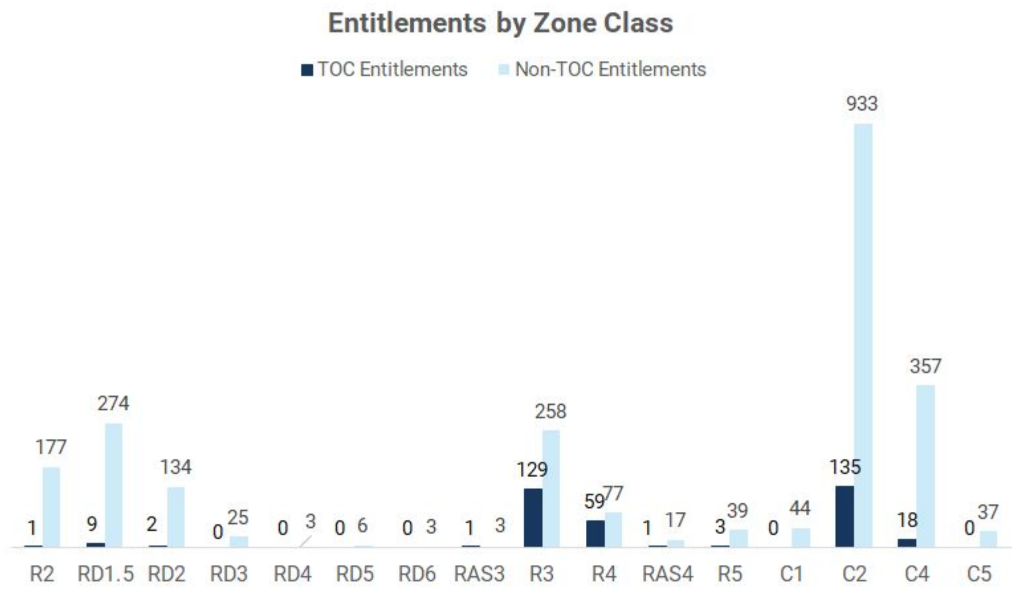
Figure 4 Number of TOC Entitlements by Tier Rule



Source: PCTS, TOC Tiers, GTFS; analysis by ITA Data Science

In terms of zoning, even though there are theoretically many zone classes that qualify for TOC entitlements, 90% of TOC entitlements occur in the R3, R4, and C2 zone classes (**Figure 5**). Developers must view the R3, R4, and C2 zone classes as being financially viable for this type of denser housing construction with affordable housing requirements sought after in the TOC guidelines. The Non-TOC entitlements that are occurring on TOC-eligible parcels are mostly taking place in RD1.5, R2, and R3 zone classes. That a significant portion of TOC and Non-TOC entitlements occur in the R3 zone areas could mean that there are certain attractive or particular characteristics about this zone that lead to a lot of entitlements being generated. It could be a sweet spot for certain types of development, and warrant additional exploration and planning domain knowledge to determine those factors.

Figure 5 Entitlements by Zone Classes



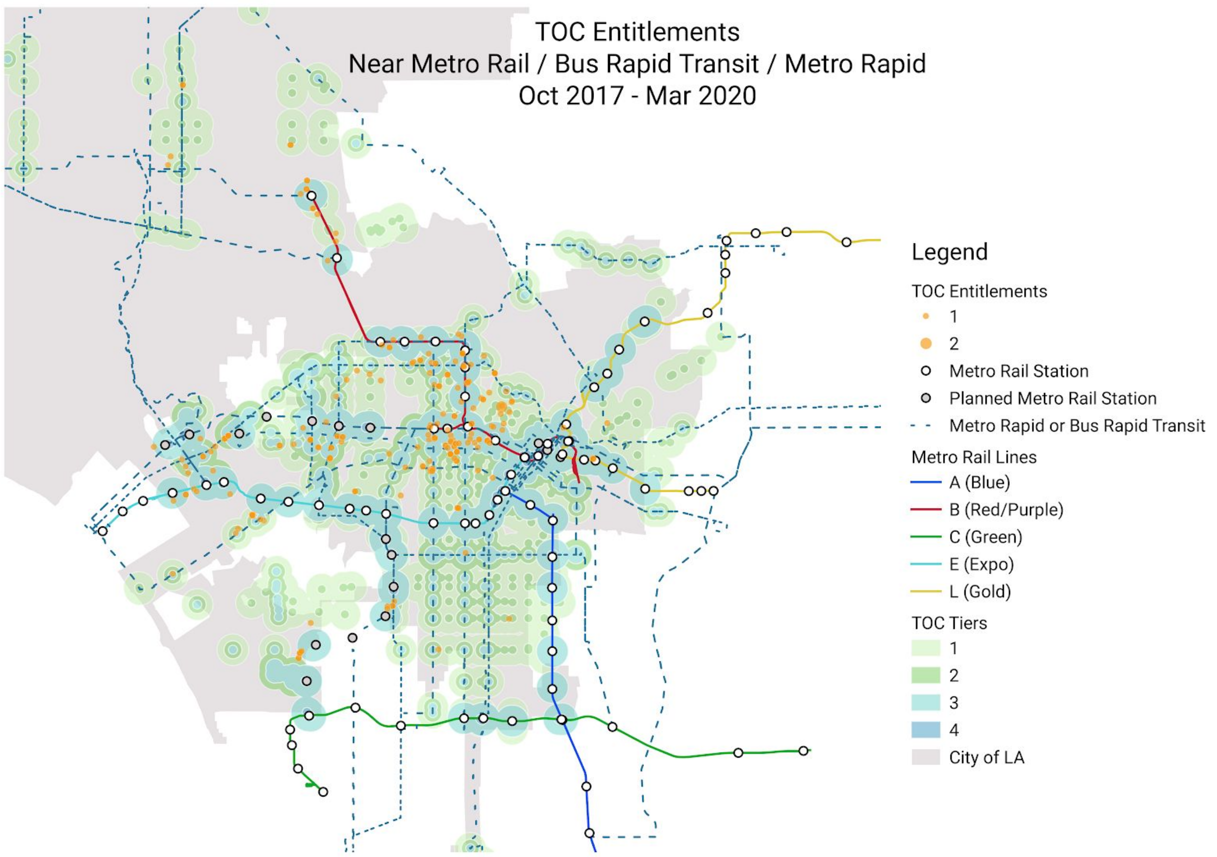
Source: PCTS, ZIMAS zoning; analysis by ITA Data Science

Putting all the TOC entitlements onto a map with the Metro rail, bus rapid transit, and Rapid lines shows that most of the TOC entitlements appear to be concentrated around the B - Red and D - Purple Lines (**Figure 6**). Along the planned Metro lines, there is some activity around the Purple Line Extension near Century City and the Crenshaw Line around Crenshaw Blvd / Slauson Ave.

Where there is little or zero TOC entitlement activity is also notable, and there appears to be few TOC entitlements around the existing light rail lines, including the A - Blue Line, C - Green Line, and E - Expo Line, L - Gold Line. Areas near the A - Blue Line have long been documented as areas of disinvestment in the urban planning literature. READ https://escholarship.org/content/qt8jd663ht/qt8jd663ht.pdf and draw some conclusions about why more investment is needed...equity implications.

Most of the rail lines converge on the downtown core, and there are specific plans and overlays there that provide better incentives for developers than what is listed under the TOC guidelines.

Figure 6 Map of TOC Entitlements

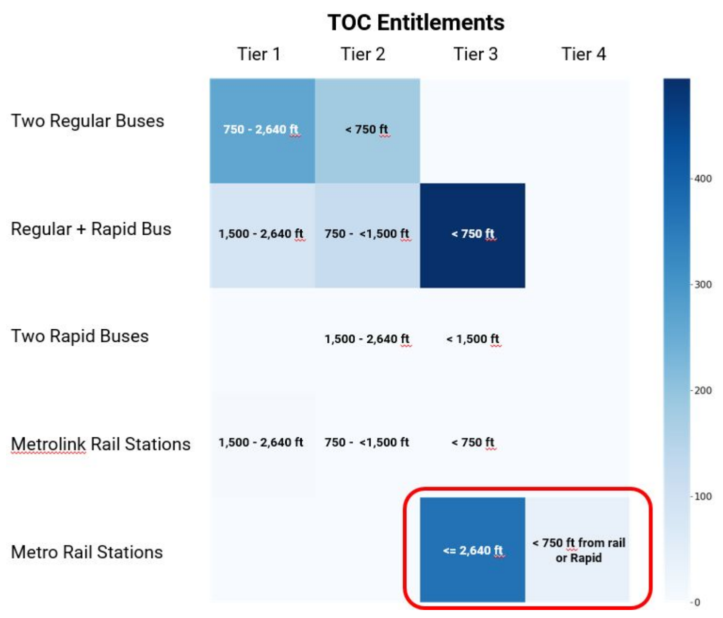


Note: The B - Red Line and D - Purple Line are shown as one line. Prior to LA Metro's 2020 renaming of the lines, these two lines were considered as one heavy rail line, sharing a significant portion of its stops. In light of the Purple Line Extension connecting between Koreatown to West LA, LA Metro renamed the Purple Line the D line.

Source: PCTS, LA Metro Developer GIS Data; TOC Tiers; analysis by ITA Data Science

By Rail Station

After establishing that TOC entitlements occur mostly in Tier 3 with R3, R4, and C2 zoning, we quantify TOC entitlement activity by Metro rail stations (circled in red below). We are looking across Tiers 3 and 4 to see where TOC entitlements are occurring.



Half of the TOC entitlements associated with Metro Rail (Tiers 3 and 4) occur near seven stations; these seven stations are all along the B - Red/Purple Lines (**Table 3**). The Wilshire / Western Station in Koreatown is associated with the most TOC entitlements, followed by the Vermont / Santa Monica Station in East Hollywood. The full table is available in **Appendix B**.

Table 3 TOC Entitlements by Rail Station (Top Rail Stations)

Station	Line	# TOC	Station's % TOC Entitlements
Wilshire / Western Station	Purple	51	12.7%
Vermont / Santa Monica Station	Red	34	8.5%
Wilshire / Vermont Station	Red / Purple	28	7.0%
Wilshire / Normandie Station	Purple	23	5.7%
Hollywood / Vine Station	Red	22	5.5%
Vermont / Beverly Station	Red	21	5.2%
Westlake / MacArthur Park Station	Red / Purple	20	5.0%

Source: PCTS, LA Metro Developer GIS Data; analysis by ITA Data Science

By Rail Line

We also explore how TOC entitlements are concentrated by rail lines and bus routes. By aggregating station-level data to line-level,, we better understand specific corridors that are benefiting from this policy.

Table 4 shows that 76% of the TOC entitlements occur along the Red, Purple, or Red / Purple Lines. The B - Red Line and D - Purple Line share six stations, between Union Station and Wilshire / Vermont. The existing D - Purple Line runs between Koreatown and Downtown LA; the future D - Purple Line extends to Westwood (UCLA). TOC entitlements are eligible and do occur along the planned extension of the D - Purple Line. The existing B - Red Line runs between North Hollywood and Downtown LA.

There is little activity around the existing E - Expo Line and L - Gold Line, light rail lines that have been around for quite some time. The E - Expo Line is the newest, with its first phase opening in 2012, and its second phase to Santa Monica completed in 2016. The Crenshaw / LAX Line (possibly named K - Line), still under construction with its first phase opening in 2021/2022, has the same level of TOC entitlement activity as the E - Expo and L - Gold Lines.

In addition to looking at how many TOC entitlements occur along a rail line, we also count the number of stations with TOC entitlements in order to calculate the average number of entitlements per station. This helps us understand whether TOC entitlements are dispersed around along all the stations along a rail line, or if they are fairly concentrated around a few stations. For example, the Red / Purple segment has 53 TOC entitlements over 3 stations, and its 17.7 average TOC entitlements per station shows there is a high concentration of TOC entitlements around those 3 stations. That differs from the Expo Line, which has 49 TOC entitlements, roughly similar to the Red / Purple, but these entitlements are spread across 10 stations, resulting in 4.9 average TOC entitlements per station, much lower than the 17.7 for the Red / Purple.

The average TOC entitlements per station metric doesn't speak to whether concentration is desirable or not, but simply helps us understand whether the TOC entitlements are occurring at punctuated intervals throughout a line or heavily concentrated in a few neighborhoods.

Table 4 TOC Entitlements by Rail Line

Rail Line	# TOC	# Stations	Line's %TOC Entitlements	Avg Entitlements per Station (# TOC / # stations w/ TOC)
Purple	129	7	32.1%	18.4
Red	125	8	31.1%	15.6
Red/Purple	53	3	13.2%	17.7
Expo	49	10	12.2%	4.9
Crenshaw	26	4	6.5%	6.5
Gold	16	6	4.0%	2.7
Blue	2	1	0.5%	2.0
Blue/Expo	2	1	0.5%	2.0

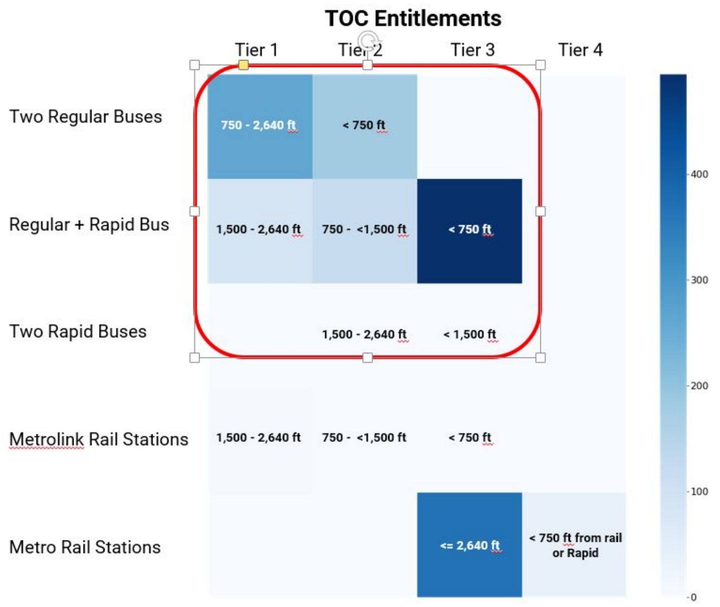
Source: PCTS, LA Metro Developer GIS Data; analysis by ITA Data Science

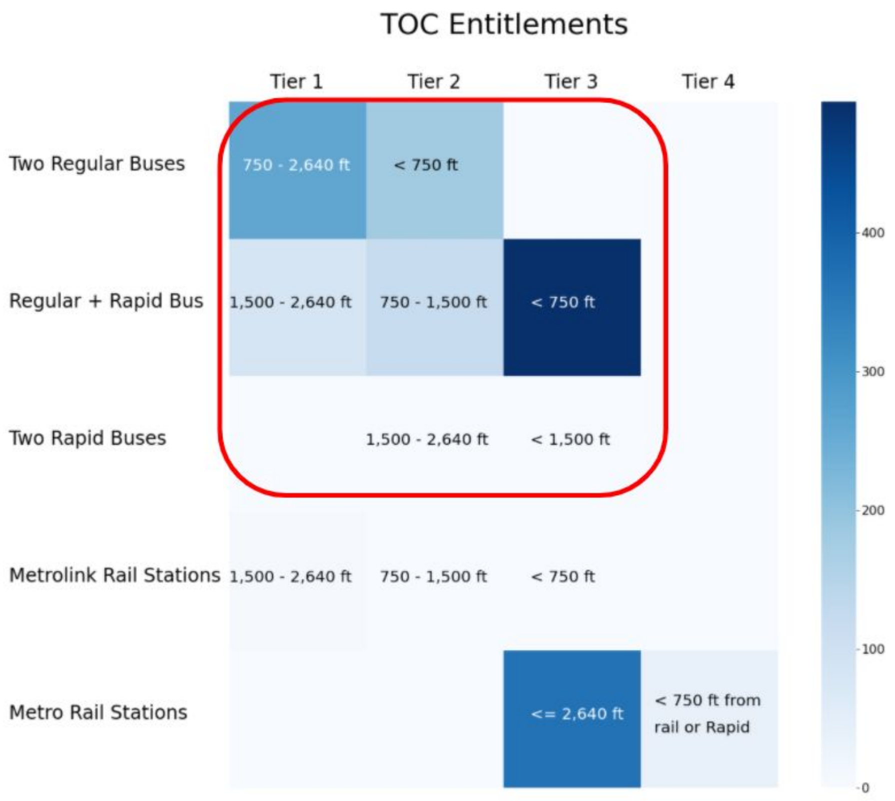
By Metrolink Station

Metrolink is considered commuter rail, but very few of the TOC entitlements are associated with being near any Metrolink station. In fact, we found only four TOC entitlements associated with the Van Nuys Metrolink Station in Tier 1, located on parcels 1,500 - 2,640 ft away from the station. We conclude that although Metrolink is included in the guidelines for TOC development, it is hardly a major contributor of TOC entitlements.

By Bus Route

As similarly done with rail, we look across multiple tiers and see which bus routes have concentrations of TOC entitlements. This is equivalent to looking across tiers by transit mode to see which stations or lines have concentrations of TOC entitlements (circled in red below).





In terms of bus routes, 40% of the TOC entitlements occur along seven bus routes, and 60% of the entitlements occur along 13 bus routes (**Table 5**). The full table is in **Appendix B**. Of these 13 bus routes, only one is from Santa Monica Big Blue Bus (R12), and all others are LA Metro bus routes. The R12 connects UCLA / Westwood to the Expo Line.

Of the top seven routes, five are Rapid lines:

- 754 (Hollywood - 105 Fwy via Vermont)
- 757 (Hollywood - Hawthorne via Western)
- 720 (Commerce - DTLA - Santa Monica via Wilshire)

- 780 (Pasadena - Mid City via Fairfax / Hollywood / Colorado)
- 710 (Hollywood / Vine - South Bay Galleria via Crenshaw)

Table 5 TOC Entitlements by Bus Route

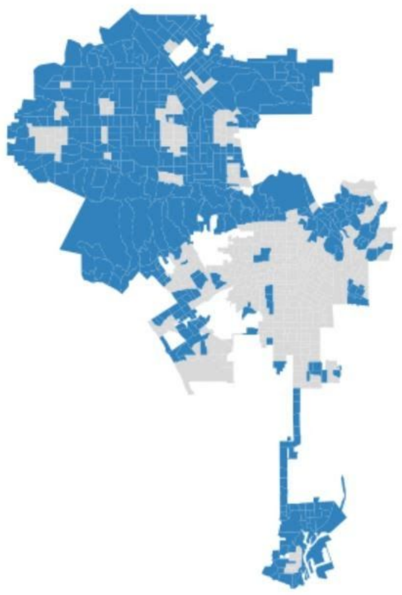
Bus Route	# TOC	Route's % TOC Entitlements
754	141	9.3%
757	125	8.2%
720	102	6.7%
780	86	5.7%
207	63	4.2%
710	61	4.0%
10/48	57	3.8%
33	53	3.5%
704	49	3.2%
204	47	3.1%
603	43	2.8%
R12	42	2.8%
206	41	2.7%

Source: PCTS, GTFS, LA Metro Developer GIS Data; analysis by ITA Data Science

Socioeconomic Characteristics of TOC Tracts

Bringing in socioeconomic characteristics from ACS data, we can see whether TOC-eligible tracts look similar or different than Non-TOC-eligible tracts. A tract is defined as a TOC tract if 50% or more of the parcels within the tract fall within one of the TOC Tiers. Given how TOC Tiers are drawn as concentric rings of circular buffers around rail stations and bus stops, these tiers divide tracts into various shapes. The simplest way is to look across all the tiers and see if half or more of the parcels fall into one of the tiers. **Figure 6** plots these TOC tracts in gray.

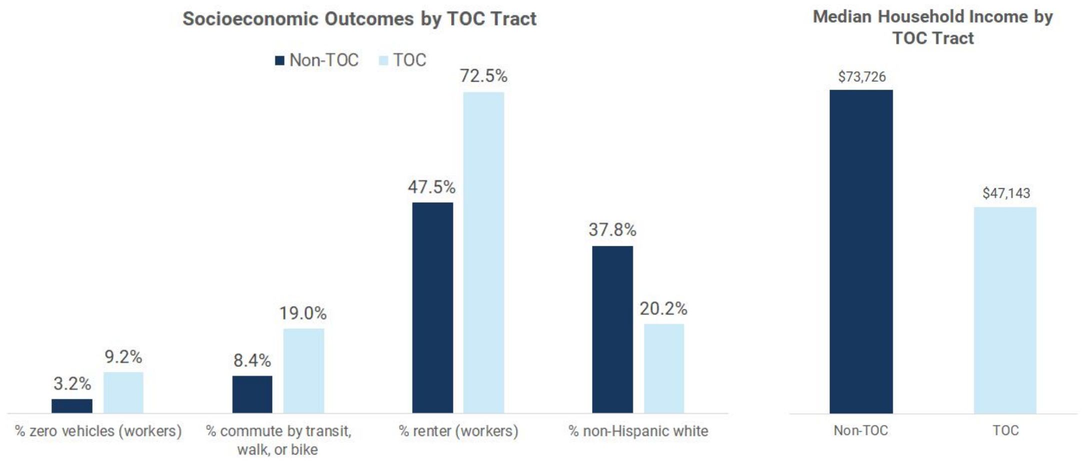
Figure 6 TOC Tracts Map



Source: TOC Tiers, census tracts; analysis by ITA Data Science

Figure 7 compares socioeconomic outcomes between TOC vs Non-TOC tracts. TOC tracts are associated with more workers from zero-vehicle households, more workers who commute by transit, walking, or biking, more workers from renter-occupied households, have lower proportions of non-Hispanic white populations, and lower median household incomes.

Figure 7 Socioeconomic Outcomes by TOC vs Non-TOC Tract



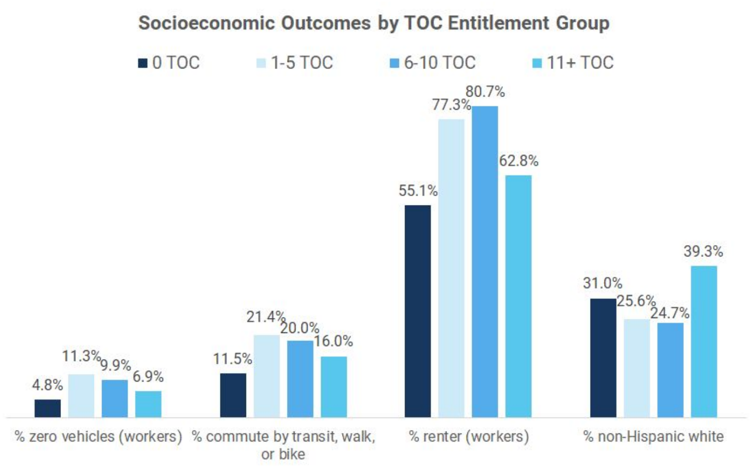
	% zero vehicles (workers)	% commute by transit, walk, or bike	% renter (workers)	% non-Hispanic white	median household income	# tracts
Non-TOC	3.2%	8.4%	47.5%	37.8%	$73,726	639
TOC	9.2%	19.0%	72.5%	20.2%	$47,143	510

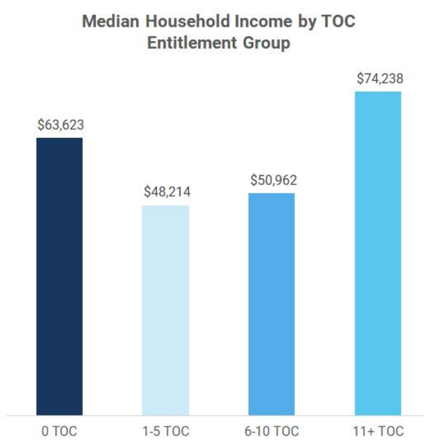
Source: PCTS, ACS 2018 5-year estimates, TOC Tiers; analysis by ITA Data Science

Instead of grouping tracts into a dichotomy of being a TOC tract or not, we also use a gradient approach, grouping tracts by the number of TOC entitlements that occurred. Tracts can have zero TOC entitlements, 1 - 5, 6 - 10, or 11 or more. This categorizes tracts according to the number of entitlements it *actually* had, not whether the tract was composed of many parcels that were *eligible*.

Figure 8 compares the socioeconomic outcomes. Except for tracts with 11 or more TOC entitlements, tracts that have between 1 and 10 TOC entitlements have the pattern seen before, being associated with more workers from zero-vehicle households, commute by transit / walking / biking, more workers from renter-occupied households, fewer non-Hispanic whites, and lower median household income. The tracts with 11 or more TOC entitlements, of which there are only 3 tracts, look much more similar to tracts with no TOC entitlements.

Figure 8 Socioeconomic Outcomes by TOC Entitlements Group





	% zero vehicles (workers)	% commute by transit, walk, or bike	% renter (workers)	% non-Hispanic white	median household income	# tracts
0 TOC	4.8%	11.5%	55.1%	31.0%	$63,623	973
1-5 TOC	11.3%	21.4%	77.3%	25.6%	$48,214	149
6-10 TOC	9.9%	20.0%	80.7%	24.7%	50,962	24
11+ TOC	6.9%	16.0%	62.8%	39.3%	$74,238	3

Source: PCTS, ACS 2018 5-year estimates, TOC Tiers; analysis by ITA Data Science

We rely on **Section III - Socioeconomic Analysis** and particularly the TOC regressions to confirm what we found in the descriptive findings, and to look for further factors that are associated with where TOC entitlements are located.

III. Socioeconomic Analysis

Add section from other draft when finished

A. Methodology

asdfasdf

B. Analysis Findings

Asdfasd

IV. Data Pipeline

Reproducibility is one of the key tenets of completing data analytics projects on behalf of City departments. To this end, we've documented all our code in this GitHub repository, saved all the datasets in a versioned S3 bucket, created the `laplan` Python package, and automate the data cleaning process in Civis to produce ongoing processed data.

One major hurdle in data science is reproducibility. Two people might get slightly different answers from answering the same question with the same data source. Two things are needed to improve reproducibility: (1) canonical data source, in this case, the PCTS data, and (2) standardized data cleaning pipeline.

This section describes not only specific decisions made within the data cleaning pipeline, but covers some of the tools the team leveraged. To make our analysis reproducible, all the code is stored in GitHub (version control), the data is stored in an S3 bucket, and we've also made the `laplan` package to generalize the data cleaning process. We also use Civis Analytics to create some dashboards as intermediary work products.

A. Data Cleaning Issues Resolved for PCTS

Getting the canonical data source was simple, as DCP owns and maintains the PCTS data, and gave us access to the data. The standardized data cleaning pipeline was less simple to construct. Some of the issues we ran into and resolved were:

- **How should we count parent and child cases? Should we drop child cases and count parent cases only?**

  We built a flexible and open-ended option. Within the `laplan` package, the function has an optional argument to keep or drop child cases. Depending on the question answered, the user should choose whether only parent cases are needed or parent and child cases are needed.

- **How should prefixes and suffixes be stored for each case?**

  We store prefixes and suffixes associated with each case in dummy variables. The dummy variable is True if that prefix or suffix appears in the case string, and False otherwise.

- **How do we reconcile the prefixes and suffixes that appear in parent and child cases? What if they are slightly different?**

  We built a flexible and open-ended option. Within the `laplan` package, the function has an optional argument to keep the prefixes / suffixes in the child cases and "roll up" the history into the parent case's history or not.

For example, the parent case might list suffixes A, B, and C, but child case 1 lists suffixes A, B, and D. Instead of losing suffix D when we drop the child case, we store suffix D as part of the parent case. As mentioned above, the prefixes / suffixes are stored in dummy variables. Therefore, suffixes A, B and C are already flagged through the parent case, but now suffix D gets flagged through the child case, and is stored in the row as the parent case. Suffixes A and B appear in both the parent and child cases, but don't get double-counted, as our dummy variable is True/False.

Parent / child case starts out as:

CASE_NUMBER	parent_or_child	DIR	TOC	1A
DIR-2018-4336-TOC	parent	True	True	False
DIR-2018-4336-TOC-1A	child	True	True	True

Parent / child case ends up as (if we only want to keep parent case, and keep child's history, look at how 1A changes without changing anything else):

CASE_NUMBER	parent_or_child	DIR	TOC	1A
DIR-2018-4336-TOC	parent	True	True	**True**

- **Can we subset by start or end date?**

  Yes, start and end dates are optional arguments, and are based on the CASE_FILE_DATE. If no start or end dates are given, it defaults to Jan 1, 2010 as the start date, and today's date as the end date.

- **What if we want to remove prefixes or suffixes?**

  A list of prefixes and/or suffixes to include is optional. If no lists are given, then all the valid prefixes and suffixes are returned.

  If we define a smaller list of prefixes as the "allow list", then only cases with prefixes on the "allow list" are returned. Cases only have one prefix; by excluding some prefixes, we automatically exclude those cases, without looking at what suffixes those cases contained.

  If we define a smaller list of suffixes as the "allow list", then all cases where those suffixes appear are returned, even if some of those cases also include suffixes that are **not** on the "allow list".

  Excluding by prefixes is more stringent than excluding by suffixes.

  For example, if only one suffix, ["TOC"], is set as the "allow list" for suffixes, a case that is *DIR-2018-4336-TOC-1A* would also be included. "TOC" appears in the case string, which means the case will be returned, even if "1A" is not part of the "allow list".

The exact functions for subsetting the PCTS data is here.

In terms of joining the various tables from PCTS to come up with entitlement data, we use the same SQL query that DCP uses internally for its PCTS internal report. The code for that is available here. After that, we use our functions in `laplan` to further subset the PCTS data by various start / end dates, a list of prefixes and/or suffixes to include or exclude, and deciding whether child cases should be kept or dropped.

After coming up with the sample of entitlements we want to analyze, we aggregate entitlements up to the parcel or tract-level. Refer back to **Figure 1** for how we would assemble the various pieces of data needed to answer a policy/planning research question.

For the transit / geographic analysis, we created a master dataset that looked at entitlements from October 2017 - March 2020, attached the relevant transit information along with entitlements aggregated to the parcel-level, and kept only parcels that fell within TOC tiers.

For the socioeconomic analysis, we created a master dataset that looked at entitlements from January 2010 - March 2020, removed a smaller list of suffixes, aggregated entitlements to the tract-level, and removed outlier cases that touched 20 or more parcels.

Given that different analyses required different master datasets, we were careful in creating our dataset “puzzle pieces". Each component had to be modular and flexible enough to accommodate various types of analyses easily assembled to build a master dataset.

The challenges and complexities we faced in assembling the “puzzle pieces” for our two analyses led us to propose a desired solution for the data cleaning pipeline to increase reproducibility. Rather than creating datasets that were designed for a specific analysis in mind, we wanted the PCTS data to undergo a similar pipeline process so that future users have several options in place. Instead of making the decisions we made over and over, which could have easily been different decisions (not wrong, just different), it was important we laid out clearly what decisions were made in consultation with the PMU team. As a result, users could receive the same processed PCTS data by using this data pipeline, after the canonical PCTS database went through a series of processing steps.

B. Python Package: laplan

The `laplan` package was borne out of the desire to generalize the data cleaning and processing pipeline. This is a Python package that can be installed on its own for use elsewhere. It is installed in our Python environment (all documented in our Dockerfile, requirements, and conda-requirements). The functions included are all fairly general and are designed to standardize the initial stages of cleaning and processing. Our analyses all do further processing.

We created our master datasets in an ad-hoc fashion, but after creating the two we used for our analyses, it was obvious that the steps to clean and process the data could easily be generalized. This was especially needed if we wanted to swap out our static PCTS dataset that ended in March 2020 for a live connection to the PCTS dataset. It would also clearly list the pieces that needed maintenance, such as the ACS data, which is updated once a year.

The `laplan` package is comprised of three cleaning scripts:

1. `pcts`: functions to subset PCTS data by prefixes, suffixes, start / end dates, parents and child cases are all included here.

2. `zoning`: functions to parse the zoning string given in ZIMAS, so we could more easily access characteristics such as zone class, height district, etc.

3. `census`: functions to download a select group of ACS tables using the Census API, and putting those tables through several steps to clean, standardize, reshaping, and aggregating so the table easily merges at the census tract level.

The documentation associated with this package (with examples) is included in full in **Appendix A**. However, the most updated documentation will always be in our GitHub repository.

C. GitHub

Our `laplan` package is stored in this project's GitHub repository. GitHub is a platform for source code management and version control. It facilitates working collaboratively on a project, providing a version of the code for all team members to make and track changes. As only code, not any data, is stored in GitHub, the repository for this project is open and public.

In conjunction with a Docker image, which provides a standardized Python environment for users of this project, all the scripts and analyses of this project are completely reproducible.

D. Civis Analytics

Civis Analytics is a data science platform ITA purchased while this project was ongoing. Civis is a platform designed for solving various environment issues users might face while using Python or R. Many issues arise because environments are hard to standardize, packages may get updated or outdated, or updated packages may break functionalities with other installed packages. The ITA data science team uses Docker to solve the environment issue, and Civis can both take Docker images users upload and provide Docker images for the user.

In addition, Civis has the ability to deploy dashboards. Jupyter Notebooks can be turned into interactive dashboards using Voila. We created a dashboard to show how many entitlements were filed for various suffixes and years. That dashboard, while not a deliverable in itself, provided us with a way to communicate with PMU on whether the PCTS cleaning scripts were getting the desired counts. It was a means for us to fine-tune our cleaning script without talking about the code itself, but allowed us to explain what was happening in the background and show how the results differed by using different cleaning methods. This underscores again the importance of a standardized data cleaning pipeline, as steps taken in different orders can lead to different results in the end! Civis makes it fairly easy to deploy, update, start or stop sharing the dashboards.

An area that the ITA data science team has not explored much in Civis is connecting the live PCTS database and scheduling a daily data update for processed PCTS dataset stored in a new database or exporting it as a Google Sheet or another format. This would be an area of future phase of work that requires DCP's system support to come up with how best to implement the data cleaning pipeline that is done in Python currently in conjunction with DCP's PowerBI and ESRI use.

Other areas DCP might want to use Civis for is its Jupyter Notebooks or R Notebooks with Civis's own environment, or to input the same environment that the ITA data science team uses.

V. Conclusion / Recommendations

Recommendations from TOC section

- Further explore what R3 zoning, why particularly attractive about those to TOC and Non-TOC entitlements alike
- The fact that most TOC entitlements occur on Red/Purple lines, could be because it's heavy rail, but also, those are areas with a lot of dense development already
- Other light rails also connect to downtown, which is a major job center...the fact that little affordable housing or any eligible housing is popping up along indicates something of a policy gap
    - We know what kinds of zoning are attractive for developers
    - How can we bridge the gap between what makes the R3/R4/C2 zoning attractive/sweet spot for developers and create similar planning conditions / incentives for developers to develop elsewhere along transit lines?
        - First, all those eligible areas that don't have developments, why? Downtown core has specific plans and overlays that have even more attractive incentives than the TOC program, so that is one raeson why developers don't choose TOC and choose other programs to benefit
        - But is that true for everywhere else? Is there no low-hanging fruit for already eligible areas to be made more attractive with additional policy programs?
        - Next is to convert other ineligible areas near transit into eligible areas. Bit harder, also more long-term, but would expect a lot of resistance from low-density residential neighborhoods. But, there are probably also ways to design planning programs, incentives, policies so that you don't force the conversion, but you allow for such a conversion to take place gradually, and as the market allows, there can be such opportunities
    - Ultimately, DCP has to work toward developing the areas near transit in conjunction with where significant capital outlay and investment has occurred to build light rail lines. Those are expensive projects, and without the land use planning needed to support it, neither of the dual goals (more transit ridership, decreased car usage and more housing, affordable housing) can be achieved. They rely on each other, positive feedback loop.

Recommendations from socioeconomic section

- Equity implications?
- Are certain characteristics associated with more TOC entitlements or more alcohol permits? What are the implications for displacement?
- Policies that would protect those areas while still achieving dual goals of denser development near transit to increase the transit ridership base, and building more affordable housing near transit, because those tend to be the transit riders

Future Areas of Work

Implement Data Cleaning Pipeline

One area of future work would depend on DCP's own implementation of the data cleaning pipeline. It is currently a Python package, but would likely have to be adapted to fit into DCP's existing pipeline of work. Whether DCP wants to convert it to additional SQL scripts or keep it as Python is up to them, but our Python functions serve as a model for what optional arguments are needed to accommodate various analyses.

This could be a short-term project in and of itself, as it requires connecting to the live PCTS database, putting it through a processing pipeline, figuring out where the processed data would be stored and accessed by end users, updating it daily, and connecting it to PowerBI and ESRI for DCP end users.

Some initial challenges would be to figure out where and what language the processing is done, whether Civis would be used for some part of this process, and how PowerBI and ESRI users can access the cleaned data scaled across multiple users with minimal setup work.

Accommodate End User Needs

PowerBI should primarily be leveraged for its visualization capability, instead of doing the processing in the DAX language. ESRI remains a tool for geospatial work and visualization, which means that data cleaning and processing should be offloaded elsewhere. However, a canonical version of the processed data (with tract characteristics, etc) should be produced as a shapefile or geojson for ESRI users. As much as possible, these joins can be done outside of ESRI for cleaner processed data.

Also, one area of maintenance work is that the ACS data needs an annual update. We have cleaned the data from 2010-2018 for the ACS tables we downloaded, even though we only used the 2018 ACS data. Soon, the 2019 5-year estimates will be available and need to be run through the cleaning scripts to the latest available data.

Application of this work

- DCP person-hours allocation across various planning areas, etc
- Get DCP to help flush this section out

References

GitHub repository: https://github.com/CityOfLosAngeles/planning-entitlements

Various documentation in GitHub repo:
- Data Workflow
- Scripts and Notebooks
- laplan package

Civis jobs: dashboard, pipeline

TOC guidelines: https://planning.lacity.org/ordinances/docs/toc/TOCGuidelines.pdf

Appendix A `laplan` documentation

The Python package `laplan` is included in our GitHub repository here, along with the README. Always refer to the GitHub repository for the most updated version.

A copy of the README is shown below.

The `laplan` package is created for the Los Angeles Department of City Planning. There are 3 sub-modules, each of which can be used independently.

The sub-modules that allow users to clean up zoning data from ZIMAS, entitlement data from PCTS, and Census data from the American Community Survey.

Getting Started

This package is installed in our Docker image.

To install it into another GitHub repo:

```
pip install
"git+https://github.com/CityOfLosAngeles/planning-entitlements#subdirectory=lap
lan"

# Ways to use within notebook/script
import laplan
import laplan.census
from laplan import census
```

Zoning

The sub-module is `zoning.py`. Zoning data comes from ZIMAS, and is publicly available on the GeoHub. Planning's Guide to Zoning String shows that the zoning string is made up of component parts.

The zoning string contains information about prefix on (Q)ualified or (T)entative zone classifications, zone class, the height district, (D)evelopment limits, and specific plans and overlays applicable.

The `ZoningInfo` dataclass takes a zoning string and returns any or all of the components as a new dataframe.

Ex 1: Return all the components

```
import laplan

parsed_col_names = ['Q', 'T',
                    'zone_class', 'specific_plan',
                    'height_district', 'D', 'overlay']

# ZONE_CMPLT is the column to be parsed.
def parse_zoning(row):
    try:
        z = laplan.zoning.ZoningInfo(row.ZONE_CMPLT)
        return pd.Series([z.Q, z.T,
                          z.zone_class, z.specific_plan,
                          z.height_district, z.D, z.overlay],
                          index = parsed_col_names)
    # If it can't be parsed, return either a failed or blank string
    except ValueError:
        return pd.Series(['failed', 'failed',
                          'failed', 'failed',
                          'failed', 'failed', ''],
                          index = parsed_col_names)

parsed = df.apply(parse_zoning, axis = 1)
df = pd.concat([df, parsed], axis = 1)
```

ZONE_CMPLT	Q	T	zone_class	height_district	D	overlay
C2-1-SP	False	False	C2	1	False	[SP]
[Q]C1.5-1VLD-RIO	True	False	C1.5	1	True	[RIO]

Ex 2: Return just one of the components

```
parsed_col_names = ['zone_class']

def parse_zoning(row):
    try:
        z = laplan.zoning.ZoningInfo(row.ZONE_CMPLT)
        return pd.Series([z.zone_class],
                         index = parsed_col_names)
    except ValueError:
        return pd.Series(['failed'],
                         index = parsed_col_names)


parsed = df.apply(parse_zoning, axis = 1)
```

PCTS

The sub-module is `pcts.py`. PCTS case strings contain prefixes and suffixes. Planning's PCTS Prefix & Suffix Report lists the valid values.

The `PCTSCaseNumber` dataclass takes a string and returns any or all of the components as a new dataframe (note that `year` and `case` are available columns in PCTS, and parsing these may not be necessary). This dataclass is used infrequently.

The function `subset_pcts` can be used once a PCTS connection is made. It standardizes the initial steps in the data cleaning pipeline so that the PCTS data is extracted and parent/child cases are combined in a standardized way before analysis. The function has optional args. `subset_pcts` and `drop_child_cases` should be used in conjunction with one another. The default is that the full dataset is returned.

- pcts: pandas.DataFrame of PCTS data.
- start_date: defaults to "1/1/2010".
- end_date: defaults to present day.
- prefix_list: a list of prefixes of interest, defaults to all prefixes.
- suffix_list: a list of suffixes of interest, defaults to all suffixes.
- get_dummies: bool, defaults to False. True returns columns for all the prefixes/suffixes of interest.
- verbose: bool, defaults to False. True returns some comments for prefixes/suffixes that have no cases.

Ex: Return PCTS entitlement cases between Oct 2017-Dec 2019 for the ADM and DIR prefixes and TOC suffixes.

```
import laplan

prefix_list = ['ADM', 'DIR']
suffix_list = ['TOC']

df = laplan.pcts.subset_pcts(
    pcts,
    start_date = "10/1/17",
    end_date = "12/31/19",
    prefix_list=prefix_list,
    suffix_list=suffix_list,
    get_dummies=True,
    verbose=True,
)
```

CASE_NBR	CASE_FILE_RCV_DT	ADM	DIR	TOC
DIR-2017-81-TOC-SPR	2018-10-19	False	True	True

ADM-2017-4594-TOC	2017-11-08	True	False	True

The function `drop_child_cases` returns a dataframe of only parent cases.

- df: pandas.DataFrame returned from `subset_pcts`.
- keep_child_entitlements: bool, defaults to True. True means that the parent case should also hold all of the prefixes and suffixes from any child cases. `get_dummies` must be True in `subset_pcts`. False means all the prefix/suffix dummies of the parent case show up, but child cases are dropped, and the prefixes/suffixes of the child cases are not stored. If a child case holds a different suffix not found in the parent case, `keep_child_entitlements = True` would store this information.

```
df2 = laplan.pcts.drop_child_cases(
    df,
    keep_child_entitlements=True
)
```

Census

The sub-module is `census.py`.

Cleaning ACS Data

The American Community Survey (ACS) data for various years, topics, and geographies all follow a similar pattern. Browse the Census Data Catalog or Census API to get the tables needed.

The scripts to download clean Census data are provided below. These scripts can be adapted to include other Census tables; our project dealt with a limited subset of ACS tables for census tracts.

1. Download Census data
2. Clean Census data, part 1
3. Clean Census data, part2
4. Subset Census

The resulting table from these scripts has this form. At minimum, the table MUST have columns `['GEOID', 'year', 'table', 'main_var', 'second_var', 'num']`, in order to use the functions in `census.py`. These necessary columns have stars next to the column name in the table below.

Column	**Description**
GEOID *	`str` preferable, but `numeric` works, geographic identifier for county, tract, block group, etc.
variable	`str`, the original Census variable name, such as `B01001_001` or `S0801_C01_001`. This is tagged as more human-readable columns, `main_var` and `second_var`.
year *	`numeric`, year associated with the table
table *	`str`, a human-readable name given to the table in `C2_clean_census.py`. Ex: For `S0801_C01_001`, the table is `S0801`, and is `commute`.
main_var *	`str`, a human-readable name that captures what the variable is mainly about. Ex: For `S0801_C01_001`, the `C01` portion what tags `main_var` as `workers`. `C02` would be `male`, `C03` would be `female`, etc.

last2	`str`. The last 2 digits of `variable`.
second_var *	`str`, a human-readable name that captures what the last two digits from the variable. Ex: For `S0801_C01_001`, the last 2 digits is `01` and designates `total`.
new_var *	`str`, combines `main_var` and `second_var`. Ex: For `S0801_C01_001`, this value is `workers_total`.
pct	`numeric`, holds percent values, ranging from 0-1.
num *	`numeric`, holds count values. The method of standardizing across tables is to have one column holding counts and one column holding percents, and filling in all the values for all tables.

Three Types of ACS Tables

ACS tables always provide a summary statistic with a `total`, the denominator, representing the universe from which this summary statistic is derived. This universe can be the entire population, the population 16 years and up, workers 16 years and up, etc.

When downloading ACS tables through the Census API, remember these things:

- Know the *unit* of the numerator and denominator.
- Does the unit change across years? Sometimes, the table will undergo a change; it will change from reporting count values to percent values after a certain year.
- Are variables are stable in reporting the same information? Particuarly, if the table has undergone a change, new columns might be added, such as `C02`. The *same* information might be found in `C01` from 2010-2013, and then in `C02` from 2014-onward.

We broadly group ACS tables into 3 types in our data cleaning process:

1. Count tables: counts are provided for numerator and denominator. Ex: # households that fall into particular income range, as well as the total # households overall within a census tract (or any other geography)
2. Percent tables: percent for numerator and count for denominator. Ex: 15 for people with less than HS education (which is 15%, not a count of 15 people), and 1,000 households in census tract. These need to be converted to counts from percents using the denominator.
3. Dollar tables: median household income or aggregate income, inflation-adjusted for each year. These tables separate, particularly because median household

income is a tricky topic. Users beware! Do not calculate summary statistics from median income values (average median income is meaningless); only report values as is. Users should think carefully about what the ACS is reporting and how to use it meaningfully in analysis.

General Functions

The main function is `transform_census_percent`, which uses 3 sub-functions, each of which can be used on its own. `transform_census_percent` takes a long, cleaned Census df, grabs one or more columns to aggregate, and reshapes the df to be wide. Example notebook.

`transform_census_percent()`: this function subsets the Census df for a particular table and year, grabs the relevant rows, aggregates them, renames the aggregated row, and then calculates the percent. The specifics of the function are best illustrated in an example.

```
import laplan

commute_group = [
    "workers_transit",
    "workers_walk",
    "workers_bike"
]

# Grab the 2018 commute table for all workers
# Aggregate transit, walk and bike
# Rename aggregated group as "non_car_workers"
# Calculate percent (non_car_workers / workers_total)
# Numerator is non_car_workers
# Denominator is workers_total
# Rename this new column "pct_non_car_workers"

laplan.census.transform_census_percent(
    "commute",
    2018,
    "workers",
    commute_group,
    "non_car_workers",
    "non_car_workers",
    "workers_total"
)
```

Cleaned, long Census df:

GEOID	variable	year	table	main_var	second_var	new_var	num
A	S0801_C01_001	2018	commute	workers	total	workers_ total	1000

A	S0801_C01_009	2018	commute	workers	transit	workers_ transit	50
A	S0801_C01_010	2018	commute	workers	walk	workers_ walk	10
A	S0801_C01_011	2018	commute	workers	bike	workers_ bike	20

`transform_census_percent` returns a wide df that looks like:

GEOID	non_car_workers	workers_total	pct_non_car_workers
A	80	1000	0.08

The sub-functions can be used individually, and should be used to construct reshaped income and race/ethnicity tables.

- Median household income: `subset_census_table` will return those values needed. No aggregation should be done!
- Households by income ranges: this table is used in conjunction with the `income_percentiles` function to re-calculate median household incomes.
- Race/ethnicity: use sub-functions because race/ethnicity groups can sum over 100%; race and ethnicity are *not* mutually exclusive.

`subset_census_table(table, table_name, year, main_var)`: subsets our cleaned, long Census df and grabs a particular table, year, and main_var.

```
# census_df is a df of
# cleaned, long Census data described above.

# 2018 commute mode table
subset_census_table(
    census_df,
    "commute",
    2018,
    "workers"
)
```

`aggregate_group(df, aggregate_me, name="aggregated_group")`: this function takes the df from subset_census_table and aggregates several rows into one. If no aggregation is needed, simply provide a list of 1. The list is made up of value(s) from new_var.

```
# To aggregate 3 groups:
# Rename new_var to "non_car_workers"
commute_group = [
    "workers_transit",
    "workers_walk",
    "workers_bike"
]

aggregate_group(
    df,
    commute_group,
    "non_car_workers"
)

# To aggregate 1 group
# Rename new_var to "zero_veh_workers"

vehicle_group = ["workers_veh0"]
aggregate_group(
    df,
    vehicles_group,
    "zero_veh_workers"
)
```

`make_wide(df, cols)`: this function takes reshapes the df from long to wide, or takes rows and pivots them to be columns. Cols is a list of values in from new_var.

```
reshape_me = ['non_car_workers', 'workers_total']

make_wide(df, reshape_me)
```

Income Functions

The income functions are `make_income_range_wide` and `income_percentiles`.

`make_income_range_wide(df, year, main_var="total")`: subsets the long, cleaned Census df and grabs the `incomerange` table for a particular year and main_var. The default main_var is `total`, which is all households, rather than a specific race or ethnicity.

```
# 2018 all households by income bins
make_income_range_wide(
    census_df,
    2018,
)

# 2018 white households by income bins
```

```
make_income_range_wide(
    census_df,
    2018,
    "white"
)
```

`income_percentiles`: takes a df and returns the estimated income percentiles. This can be used to re-calculate the median household income (50th percentile). The households are aggregated done to a larger geographic area after `make_income_range_wide`, after which the median household income is re-calculated over this larger geographic area. If no aggregation is needed, then using the median household income table is sufficient in itself. The function returns percentiles in thousands of dollars, so multiply by 1,000 to get the result in dollars.

```
# Calculate 25th, 50th, and 75th percentiles.
iqr_df = (df.apply(
        lambda r:
        pd.Series(laplan.census.income_percentiles(
            r, [25,50,75]),
            dtype="float64"),
            axis=1,
    ).rename(
        columns={0: "Q1",
                1: "Q2",
                2: "Q3"})
)
```

`iqr_df` looks like (note: units are thousands of dollars):

GEOID	Q1	Q2	Q3
A	30.5	55.7	82.6
B	40.5	58.7	90.6

Appendix B TOC Entitlements by Rail Station

The full table of number of TOC entitlements by rail station is shown below.

Table 3 TOC Entitlements by Rail Station

Station	Line	# TOC	Station % TOC Entitlements
Wilshire / Western Station	Purple	51	12.7%
Vermont / Santa Monica Station	Red	34	8.5%
Wilshire / Vermont Station	Red/Purple	28	7.0%
Wilshire / Normandie Station	Purple	23	5.7%
Hollywood / Vine Station	Red	22	5.5%
Vermont / Beverly Station	Red	21	5.2%
Westlake / MacArthur Park Station	Red/Purple	20	5.0%
Wilshire / Fairfax	Purple	18	4.5%
Vermont / Sunset Station	Red	15	3.7%
Wilshire / La Cienega	Purple	12	3.0%
Westwood / UCLA	Purple	12	3.0%
Hollywood / Western Station	Red	12	3.0%
North Hollywood Station	Red	11	2.7%
Century City / Constellation	Purple	10	2.5%
Crenshaw / Slauson	Crenshaw	10	2.5%
Hollywood / Highland Station	Red	9	2.2%
Palms Station	Expo	8	2.0%
Culver City Station	Expo	7	1.7%
Westwood / Rancho Park Station	Expo	7	1.7%
Expo / Sepulveda Station	Expo	7	1.7%
Florence / West	Crenshaw	6	1.5%
Florence / Hindry	Crenshaw	6	1.5%

Expo / Bundy Station	Expo	5	1.2%
Expo / Crenshaw Station	Expo	5	1.2%
Farmdale Station	Expo	5	1.2%
Union Station - Metro Red & Purple Lines	Red/Purple	5	1.2%
Union Station - Metro Gold Line	Gold	4	1.0%
Leimert Park	Crenshaw	4	1.0%
Pico / Aliso Station	Gold	3	0.7%
Mariachi Plaza / Boyle Heights Station	Gold	3	0.7%
Wilshire / La Brea	Purple	3	0.7%
Heritage Square / Arroyo Station	Gold	3	0.7%
Soto Station	Gold	2	0.5%
Pico Station	Blue/Expo	2	0.5%
Expo / Vermont Station	Expo	2	0.5%
Expo / La Brea Station	Expo	2	0.5%
Grand / LATTC Station	Blue	2	0.5%
Universal / Studio City Station	Red	1	0.2%
Chinatown Station	Gold	1	0.2%
La Cienega / Jefferson Station	Expo	1	0.2%

Source: PCTS, LA Metro Developer GIS Data; analysis by ITA Data Science

Table 5 TOC Entitlements by Bus Route

Bus Route	# TOC	Route's % TOC Entitlements
754	141	9.3%
757	125	8.2%
720	102	6.7%
780	86	5.7%
207	63	4.2%
710	61	4.0%
10/48	57	3.8%
33	53	3.5%
704	49	3.2%
204	47	3.1%
603	43	2.8%
R12	42	2.8%
206	41	2.7%
217	41	2.7%
200	40	2.6%
728	39	2.6%
14/37	38	2.5%
4	33	2.2%
224	33	2.2%
233	28	1.8%
705	27	1.8%
901	24	1.6%
163/162	23	1.5%
35/38	23	1.5%

30/330	22	1.5%
164	20	1.3%
212/312	20	1.3%
28	15	1.0%
CC 6R	15	1.0%
16/17/316	14	0.9%
165	13	0.9%
105	12	0.8%
CC 1	11	0.7%
7	10	0.7%
110	10	0.7%
501	8	0.5%
20	7	0.5%
111	7	0.5%
3	6	0.4%
751	6	0.4%
152/353	6	0.4%
1	5	0.3%
18	4	0.3%
8	4	0.3%
53	4	0.3%
2/302	4	0.3%
770	4	0.3%
76	3	0.2%
745	3	0.2%
180/181	3	0.2%

760	2	0.1%
910/950	2	0.1%
78/79/378	2	0.1%
40	2	0.1%
66	2	0.1%
750	2	0.1%
55/355	2	0.1%
60	2	0.1%
51/52/351	1	0.1%
605	1	0.1%
150/240	1	0.1%
115	1	0.1%
68	1	0.1%

Source: PCTS, GTFS, LA Metro Developer GIS Data; analysis by ITA Data Science