Creating CountDracula: an Open Source Counts Management Tool

Daniel Tischler*
San Francisco County Transportation Authority
Tel: 415-593-1660
Fax: 415-522-4829
Email: dan.tischler@sfcta.org
1455 Market Street, 22nd Floor
San Francisco, CA 94103

Lisa Zorn
San Francisco County Transportation Authority
Tel: 415-593-1660
Fax: 415-522-4829
Email: lisa.zorn@sfcta.org
1455 Market Street, 22nd Floor
San Francisco, CA 94103

Elizabeth Sall
San Francisco County Transportation Authority
Tel: 415-522-4810
Fax: 415-522-4829
Email: elizabeth.sall@sfcta.org
1455 Market Street, 22nd Floor
San Francisco, CA 94103

* indicates corresponding author

Submission Date: November 15, 2013

Word Count: 5,054 + 8 Figures x 250 = 7,054

Submitted for presentation at the 2014 Transportation Research Board Annual Meeting.

This paper is for the Transportation Planning Applications Committee to review.
Abstract

One of the most persistent problems faced by the San Francisco County Transportation Authority (the Authority) is that of handling a growing collection of counts. Traffic, pedestrian and bicycle counts have been collected by staff, consultants and sister agencies for numerous planning studies at various locations in San Francisco over the years. But how should these counts be organized? Some are in Excel workbooks of varying and spontaneous formats, others consist of scanned handwritten documents, and finally some are simply on paper.

Since the modeling team at the Authority has a continuous need for these counts in order to calibrate and validate the travel demand model as well as to inform model development, these counts have come under the team’s purview. After a couple of failed attempts to standardize Excel formats and directory structures, the modeling team decided to modernized its counts management system. The Authority first explored proprietary software products, but found these solutions to be too expensive, cumbersome, or inflexible. Instead, Authority staff embarked on developing CountDracula, an open source counts management tool. The aim of CountDracula is to make uploading, downloading and querying counts easy for Authority staff as well as other interested parties outside the organization. The CountDracula code base has been designed to be reusable by other agencies with similar needs, and it is built on GeoDjango, a geographic web framework.

CountDracula includes a web-based map user interface for visualizing the locations of counts (and as a side effect, the locations of where more counts are needed), and it includes a query interface so that specific types of counts can be batch-downloaded (for example, midweek counts from the last three years). As it was developed by a modeling team, there is a specific emphasis on counts seamlessly interfacing with model transportation networks. Counts can also be uploaded using this interface, and moderated through an admin interface. This paper explains the development of Count Dracula, and explores how CountDracula fits into the open data and open source movement.
Motivation and Background

Like many cities, San Francisco tries to adhere to data-driven decision-making about the most effective ways to invest in its transportation system. The city has a multitude of databases that represent the transit network (San Francisco Municipal Transportation Agency n.d.) as well as a state-of-the-art travel demand model (Outwater and Charlton 2008, Sall, Bent, et al. 2010, Zorn, Sall and Wu 2012) and citywide dynamic traffic assignment model (Sall, Erhardt, et al. 2013) to forecast various future scenarios. However, despite its location near Silicon Valley and possession of high-tech forecasting technology, San Francisco did not have any comprehensive system for measuring the use of its roadways by walking, cycling, or automobile. Specific projects had conducted numerous traffic counts, but there was no systematic or electronic way of storing these counts - and those that were stored in a central location lacked meta-data, context, and were often rolled up into single numbers thereby losing the richness of the original data. Answering seemingly simple questions such as “how bad was traffic in the Northwest part of San Francisco in 2005 compared to 2000” would have been a several week-long task, and there was no hope of being able to do simple mappings of spatial and temporal trends across the years. The most accessible form of the count data was in an excel file used to validate the SF-CHAMP travel demand model, which was hardly user-friendly.

As the San Francisco County Transportation Authority (the Authority) looked towards being able to validate its Dynamic Traffic Assignment model (Parsons Brinckerhoff; San Francisco County Transportation Authority 2012) with 15-minute count data for the entire city, it became obvious that investment in a robust count database tool would not only save time for that specific model development project, but it could very easily be adapted to be useful to a variety of users including transportation planners investigating travel patterns for a neighborhood study, consultants doing traffic impact analysis and environmental review, and in the spirit of San Francisco’s open data policy (Board of Supervisors, City and County of San Francisco 2010), the general public.

Design Considerations

The Authority team developed a list of user-requirements for the count database, née CountDracula, based on four user types: local agency transportation planner/engineer, travel modeler, traffic engineering consultant, and the general public. The goals of CountDracula are to:

1. Store count data electronically in a single location (as opposed to emailing a dozen people to ask them to check their file cabinets).
2. Allow universal access to the data via web-based user interface.
3. Query count data to find specific times, dates, or locations both efficiently (via application programming interface –API-- access) and intuitively (web-based Graphical User Interface, or GUI).
4. Download data into commonly used data formats (such as the Universal Traffic Data Format, or UTDF) or structured data.
5. Include varying levels of account permissions (to allow for some users to write, and others to just read).
6. Allow user-uploads of data in a variety of formats that integrate into existing workflows.
(7) Maintain important meta-data such as: count methodology, associated projects, requesting agency, etc.

The main technology consideration was to use existing tools as much as possible in order to minimize both the development and maintenance burden. The Authority team first sought to find an existing tool and considered the following solutions, although they all failed to meet one or more of our requirements (Midwestern Software Solutions n.d., Cambridge Systematics n.d., California Department of Transportation n.d.). After doing a thorough review of existing options, the Authority decided to create CountDracula, our own open-source count data management system. In particular, many existing systems lack API-access to the data so while it was easy to query a single count, it was more difficult to seamlessly link the data to our travel model, our DTA model, and to do flexible queries and data analysis (e.g. what are all the counts that have more than 10% trucks? or what is the typical standard deviation of the P.M. peak hour in the northwest part of San Francisco?).

Implementation

CountDracula utilizes existing tools as much as possible in order to maximize efficiency and minimize code maintenance. After implementing an initial version of CountDracula in pure Python that directly issued SQL commands to a PostgreSQL database, the development team realized that resources would be used more efficiently by leveraging existing tools and frameworks that already accomplish: database manipulation, user-friendly account provisioning and an administrator interface, and webpage templating (custom views). After considering a few options, the Authority team settled on Django, a Python web framework with the following appealing features (Django Software Foundation n.d.):

- Django is open source. This means that the team can use it for free, and in turn, that other CountDracula users will also be able to use it for free. It also means that users can see “under the hood” to understand what is going on.
- Django has an active development community and it is backed by its own non-profit organization. This means that Django is well peer-reviewed and tested and that the CountDracula development team can rely on the Django community to do frequent maintenance and keep the code base relevant and up-to-date.
- Django is in Python, which is highly readable and Authority Staff are already proficient in it.
- Django has a high quality object-relational mapper, so that data models can be defined entirely in Python and writing SQL queries can be avoided.
- Django comes with an automatic admin interface, elegant URL design, a template system that allows us to customize views, and caching to improve performance.
- Django has an add-on, GeoDjango, which adds a spatial data types and enables efficient spatial-based queries and uses PostGIS (the spatial add-on to PostgreSQL).

Models

In Django, an object and its attributes are defined in “a model”; these typically map to a single database table. Models may also have relationships defined with other models. The CountDracula models and their attributes are as follows:
**Node**: This object represents an intersection or physical location of interest.

- **point**: The location of the intersection. This is a `PointField`, which is a GeoDjango field type.

**StreetName**: This object represents a textual street name in the network. As a convention, our import scripts store all StreetName fields as all upper case.

- **street_name**: This is the standard full street name, including spaces and an abbreviated generic suffix (St, Rd, Ave, etc.) e.g. “CESAR CHAVEZ ST”
- **nospace_name**: This is a version with the spaces stripped. This is used for matching in situations where the spacing is ambiguous. e.g. “CESARCHAVEZST”
- **short_name**: The street name without the generic suffix. e.g. “CESAR CHAVEZ”
- **suffix**: The generic suffix, abbreviated. e.g. “ST”
- **nodes**: The relationship between StreetName instances and Node instances is many-to-many. This enables look-ups in both directions: one can query an intersection node for the street names of those streets that meet at that intersection, or query a street name for all the nodes corresponding to the street.

**TurnCountLocation**: This object represents a location for a turn count.

- **from_street**: A reference to the StreetName instance describing the street from which this turn originates.
- **from_dir**: The direction going into the turn (one of N,S,E or W).
- **to_street**: A reference to the StreetName instance describing the street to which the turn is destined.
- **to_dir**: The direction coming out of the turn (one of N, S, E or W).
- **intersection_street**: A StreetName reference for identifying a cross street at the turn count intersection location, in case the turn movement is a through movement, and the from_street is the same is the to_street.
- **intersection**: A Node reference for the turn count location.

**TurnCount**: This object represents a single turn count or an average of a set of turn counts.

CountDracula does allow count averages, although they are not preferred, in cases where the raw count data is not available and only the average is available.

- **location**: A reference to the TurnCountLocation where this count was observed.
- **count**: The count itself. This is a decimal number because it may represent an average.
- **count_date**: The date that the count was collected. This can be null if the count is an average over multiple dates.
- **count_year**: The year that the count was collected; this is a required field.
- **start_time**: The start time for the count period.
- **period_minutes**: This is an integer and it represents the period of time over which the count took place.
- **vehicle_type**: This is an integer code representing the vehicle type.
- **sourcefile**: This is the name of the source file from which the count was imported. These files will be archived for the system, for situations when a count needs to be investigated for possible misinterpretation or error.
- **project**: A string for tracking for what project the count was collected, if any.
- **upload_user**: A reference to the user (a built-in Django model) that uploaded the count data.

**MainlineCountLocation**: this object represents a location for a mainline count.

- **on_street**: A reference to the `StreetName` instance describing the street on which the mainline count is observed.
- **on_dir**: The direction of the link on which the count is observed (one of N, S, E or W).
- **from_street**: A reference to a `StreetName` instance describing a cross street upstream of the count location.
- **from_int**: A reference to a `Node` instance at the intersection of on_street and from_street.
- **to_street**: A reference to a `StreetName` instance describing a cross street downstream of the count location.
- **to_int**: A reference to a `Node` instance at the intersection of on_street and to_street.

**MainlineCount**: This object represents a single mainline count or an average of a set of mainline counts. These can be average counts, but raw counts are preferred.

- **location**: A reference to the `MainlineCountLocation` where this count was observed.
- **count, count_date, count_year, start_time, period_minutes, vehicle_type, sourcefile, project, upload_user**: these are the mainline versions of the same variables in the `TurnCount` model
- **reference_position**: How far along the link the count was actually observed. A value of -1 represents unknown.

### Tables

Given the model definitions discussed above, Django automatically creates the PostGIS database tables to support them and their relationships. Additionally, Django creates some additional tables for users, groups, and permissions, as well as for sessions and logging.

---

*Figure 1* Login screen for CountDracula; this functionality comes built-in to Django.
User Interface

As previously mentioned, Django automatically creates a web-based admin interface for viewing and editing the data. This interface comes with a user login system, as well as basic edit screens for each model. Figure 1, Figure 2 and Figure 3 show some examples of these admin interface web pages.

Figure 2 This is the view of the StreetName instances.
Figure 3  The edit view for a StreetName instance.
On top of the basic Django administrator interface, CountDracula adds two additional views. The first is the map view, shown in Figure 4, which shows all of the available data on a map, giving users a more intuitive understanding of the location of counts. Mainline count locations are indicated by a line segment, while turn count locations are indicated by a node at the intersection of the movement. This view includes filtering widgets along the left hand side, so that users can see where counts have been collected for various vehicle types or for a specific set of years. Additionally, users can enter in an address or intersection, and filter count locations to those within a specific radius of that point (Figure 5). Finally, the raw counts can be downloaded as a comma-separated value file, for all mainline or turn count locations or for those locations filtered by the widgets.

Figure 4 CountDracula map view of count locations.
Figure 5  Map view with year- and location-based filtering of count locations.

Figure 6  Upload view
The second custom CountDracula view is the upload view, shown in Figure 6. This view allows logged in users to upload a workbook of count data of a predetermined format, described in a later section. CountDracula parses this count data and stores it, if no errors are found during processing. If errors are found, they are reported back to the user for correction. In the future, a process for moderation or validation of uploaded data could be implemented.

**Data Population**

Although CountDracula was written to be a generic counts management tool for any interested party with similar needs, in order to create an installation of the system that fit our needs, some San Francisco-specific scripts were created to import data. First, a script called `insertSanFranciscoIntersectionsFromCube.py` gets called to:

1. Read the Authority’s macroscopic static assignment network. This is in the Citilabs Cube Voyager .network format (Citilabs n.d.), but it could be easily adapted to read any format that exports a series of nodes (locations) and links (attributes for an ordered pair of nodes).
2. The nodes are added as new Node instances to CountDracula. Nodes that are outside of San Francisco (such as those in neighboring counties) are skipped.
3. The street names for links are added as StreetName instances to CountDracula. Additionally, the relationship between the StreetName instances and their corresponding Nodes is added. Links that are outside of San Francisco are skipped, as are links that are missing street names.

Following this, several scripts are also run to import various types of data into CountDracula. One of them reads downloaded data from the Caltrans Performance Measurement System, or PeMS (California Department of Transportation n.d.). This data set is an aggregate of data for all of calendar year 2010, cleaned and restricted to non-holiday mid-weekdays and it is already being used for 2010 vehicular traffic model validation of SF-CHAMP. A second set of legacy data is similarly imported by another script, which imports 2007-2008 traffic counts for major routes at county border locations collected for the Metropolitan Transportation Commission, the regional metropolitan planning organization.
The third import script is the most general, importing an excel workbook file format that was devised to be a generic template representing how raw traffic counts are often stored. **Figure 7** shows an example of this format; each separate day of counts is on its own worksheet and within each sheet, each row represents a time period. Blank lines separate sections for different vehicle types. This generic count format workbook is the type that is accepted by the current web-based upload mechanism discussed above.

**Deployment Environment**

At the time of this writing, CountDracula requires the following tools to run:

- Django and GeoDjango, described in detail above
- PostgreSQL: an open source object-relational database system
- PostGIS: an open source add-on that spatially enables PostgreSQL with GIS functionality
- Apache: an open source HTTP server; this is necessary to provide the web-based user interface
- python: the open source programming language on which Django, GeoDjango and CountDracula are built

**Figure 7** Standardized movement count input spreadsheet.
modwsgi: WSGI (the Web Server Gateway Interface) is a specification for web servers to communicate with web applications. modwsgi is a way for Apache to run a web application in python, and it’s the recommended way to get Django to work with Apache.

Additionally, the following python modules are required:

- psycopg: a PostgreSQL adaptor for Python
- xlr: a python module for reading Microsoft Excel files
- python-memcached: a memory-based caching framework to improve performance

Creating Open Source Tools at a Public Agency

Why open source

Over the past few years, the Authority has increasingly been developing tools in the open source domain. This has been for a couple of reasons. Many of the Authority’s modeling projects are funded by sales tax dollars and Federal Highway Administration money, and many of them address common problems facing transportation agencies and modelers. By making these projects open source, the Authority team hopes that these public funds will not be spent doing something more than once, and that other interested municipalities and teams can fork or improve on these projects rather than starting from scratch. In turn, when other teams can spend resources improving these shared tools, then everyone will benefit, including the Authority.

Additionally, creating open source projects has other advantages. It allows the Authority team to be flexible with consultants because these tools are not proprietary, nor do they include proprietary components. It also creates incentive for these projects to be less vendor-specific. Having an open source project encourages the development team to follow best practices for some tasks which are more easily neglected in a closed environment, such as thorough documentation and portability. Finally, even if other public agencies or interested parties do not use the codebase itself, developing the project in an open source environment means that the codebase and the process can still serve as a reference and help to expose the project’s lessons learned.

Why in-house development

While the Authority team does recognize that this is not typically work done in-house by other agencies, the decision to develop an initial version in-house was made for several reasons. First, the Authority team had appropriate in-house resources: many of the count data files had already been standardized for previous script work, and the programming work was minimal once the decision was made to use the Django (and GeoDjango) framework. Given this relatively low level of effort needed to get an initial version running, the overhead of writing a scope of work and hiring outside consultants would likely have been more work. Further, the consulting teams that were readily available to the Authority team did not necessarily have particular expertise in the technology involved in this project.

Second, since the first use of CountDracula was the DTA Anyway project which was already being managed in-house, the Authority team had a unique understanding of its requirements. In addition, developing this initial version of CountDracula in-house was more flexible to changing demands and
needs. For future development, the Authority team will seek outside assistance from both collaborators at other agencies and/or from consultants for implementing future features.

Complementary Projects

As mentioned previously, the initial impetus for the development of CountDracula was the development of San Francisco’s citywide DTA model, which benefited from a large selection of count data for model validation (over 1,100 counts were used for this, with 15-minute movement counts making up the majority). Model validation in general is an excellent use case for count data, and CountDracula complements these projects easily by giving users an API to the dataset, so scripts can algorithmically relate the counts to their modeled links and movements.

Other planning/modeling projects which could complement CountDracula include:

1. Pedestrian/Bicycle models. When analyzing pedestrian and bicycle crash data, it is also useful to take exposure into account, which means that vehicle volumes as well as bicycle and pedestrian volumes must be analyzed as well. CountDracula will be helpful for both of these datasets.

2. Transit service analysis. While transit vehicle data (such as automatic vehicle location data and automatic passenger count data) are necessary for transit service analysis, other information on local streets is also relevant. Conflict with auto vehicles, pedestrians and bicyclists likely has an effect on transit performance, and CountDracula could assist in this analysis.

3. On-the-fly micro-simulation network creation. Acquiring detailed counts in the study area is crucial for creating an accurate micro-simulation network. By accessing count data from CountDracula, one could map these counts to a study area in an automated fashion (including automatic network balancing), which could lead to possibilities like on-the-fly micro-simulation creation.

The Future

CountDracula’s Future

Since CountDracula is an open source tool, the Authority team hopes that other users and collaborators will become involved and help define the future of CountDracula. Some priorities have been discussed in the context of San Francisco’s needs. For example, pedestrian counts are not currently included in CountDracula, but they are critical to the transportation landscape in San Francisco, and so the addition of pedestrian counts is a high priority. These counts are typically observed as crossings at intersections, so the counts and their locations would need to be defined in a new model. Other vehicular data could be included as well, such as vehicle speed data, enabling CountDracula to support Congestion Management Programs and other types of model validation.

There is also room for improvement in the admin interface of CountDracula and many of the pages would benefit from a more map-based representation of points to check that imported data looks correct. The StreetName admin page, for example, would benefit from showing a map of all of relevant Node instances rather than a textual list of them (Figure 2, above).
Additionally, the current representation of street direction could be re-examined. In San Francisco, where most blocks are short and most streets are straight and form a grid-like pattern, defining links using the simple 4-cardinal directions works reasonably well. However, in other cities where streets may wind and change direction, this may be too ambiguous and require a more nuanced handling. Indeed, in some parts of San Francisco, where the grid alignment is at an angle, the Authority team had to choose a convention and call all northwest-bound streets northbound instead of westbound, for example.

Finally, CountDracula should be expanded to handle more input and output formats that are used by other municipalities, such as the Universal Traffic Data Format (UTDF). CountDracula also needs to have an easier installation process (such as one for a generic cloud server, like an Amazon EC2 instance) as well as a tutorial for typical setup and API usage. In this way, CountDracula instances will be easily set up and populated for other cities and regions.

**Open Data Management’s Benefits**

In the last five years, providing public access to data has been an emphasis of government on both the federal and local levels. On January 21, 2009, President Obama released the Transparency and Open Government memorandum, asking executive departments and agencies to “establish a system of transparency, public participation, and collaboration.” (Obama 2009) On October 21, 2009, then Mayor Gavin Newsom responded on behalf of the city of San Francisco with Executive Directive 09-06 on Open Data, which stated: “This Directive will enhance open government, transparency, and accountability by improving access to City data that adheres to privacy and security policies. Data which often resides in technology systems is unique from information like documents, emails and calendars in that it is structured and can be used by other computer applications for analysis or new uses such as mapping.” (Newsom 2009) The San Francisco Board of Supervisors expanded on this directive by adding Section 22D into the Administrative Code in 2010 (with an amendment in 2013 to create the position of Chief Data Officer). Section 22D lists the benefits of an open data policy (Board of Supervisors, City and County of San Francisco 2010):

1. enhanced government transparency and accountability;
2. development of new analyses or applications based on the unique data the City provides;

Figure 8 CountDracula Process/User Diagram
mobilization of San Francisco’s high-tech workforce to use City data to create useful civic tools
at no cost to the City; and

creation of social and economic benefits based on innovation in how residents interact with
government stemming from increased accessibility to City data sets.

On the community side, groups have also jumped in to help fill this gap to connect data with those
interested in making use of the data (localdata n.d.). Open data policies and practices are gaining
momentum in San Francisco and elsewhere, and CountDracula supports this vision.

Hosting data with both an easy-to-use point-and-click front end coupled with flexible and powerful API
access to the database not only greatly expands the audience of potential users, but also alleviates a
burden normally placed on public employees to provide timely responses to custom data requests.

**Open Source Development’s Benefits**

Many public agencies have similar needs for planning and analysis tools, and building and maintaining
those tools internally leads to inefficient resource allocation and redundant expenditures. To the extent
possible, public agencies should collaborate on these tools that are likely to have shared benefits by
embracing open source development policies, thereby making public funds go farther. Many agencies are
doing this already by becoming involved in open source initiatives; see the referenced website for a
partial list: (Collaboration, Open Solutions, and Innovation n.d.). In order for an open source project to be
successful, it is not sufficient to simply push the code to a public server. Agencies need to share tools,
methods, and resources to develop the project, and joint funding mechanisms are required. To deal with
the legal hurdles involved in this type of collaboration in the urban planning space, the Open Source
Planning Initiative has been formed (Open Source Planning Initiative n.d.) The Mission Statement of this
organization is as follows:

“The Open Source Planning Initiative supports innovation, development, and sharing in open source
urban planning software. This foundation was formed out of an on-going need identified by urban
planning practitioners across the nation to focus on innovation rather than rebuilding the same tools over
and over again. This foundation will serve as an independent legal entity to which community members
can contribute code, funding, and other resources, secure in the knowledge that their contributions will
be maintained for public benefit.”

Future development of CountDracula will likely be performed under the Open Source Planning Initiative.

**Use by Others**

CountDracula should be directly transferable to use in any other jurisdiction or situation that would want
to use the same data structure and that has a list of nodes with intersecting street names. The setup batch
file included in the open-source codebase can be easily modified to point to local files and directories.
Over time, as more jurisdictions use this tool, they can add more functionality to the system (i.e. more
query types, more GUI features, more file types).
Acknowledgements

Special thanks to former intern Varun Kohli for developing the initial version of CountDracula.

References


