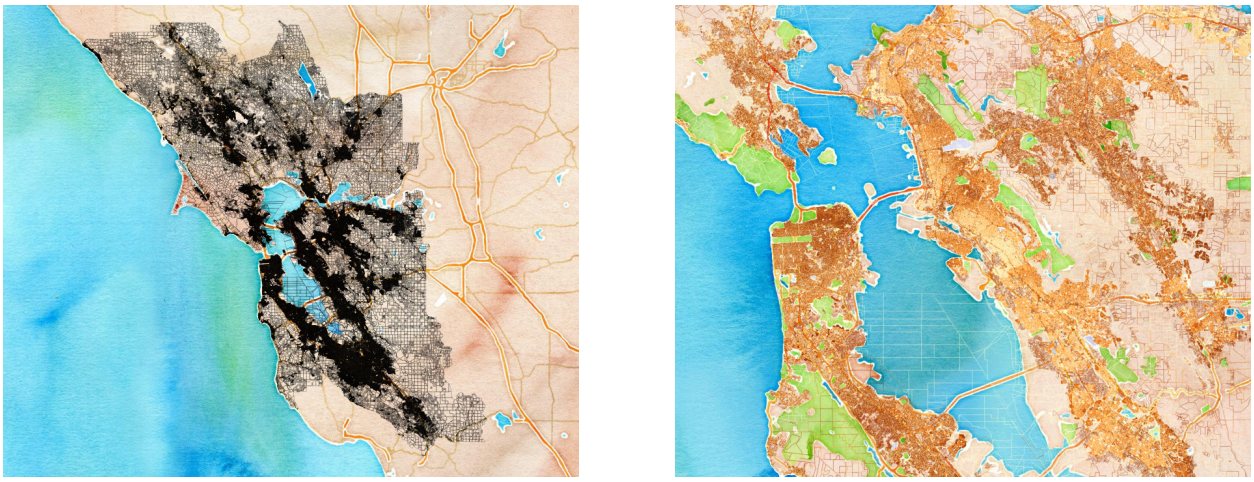
BAY AREA REGENERATION INSTALLATION GUIDE

SYNTHICITY



Base map tiles by Stamen Design and data by OpenStreetMap contributors.

This guide is provided for information purposes only. Synthicity does not assume any legal liability or responsibility for its accuracy, completeness, or usefulness. References to third-party software are made for convenience and do not imply endorsement.

OVERVIEW

The regional regeneration tools are written in Python and SQL. They depend on software developed by Synthicity, spandex and urbansim, and third-party open source software, including Python, pandas, PostgreSQL, and PostGIS. Refer to table 1 for the purpose of each dependency.

Although not required, a visualization tool, like UrbanCanvas or QGIS, is recommended.

Assumptions. This guide makes several assumptions.

(1) The operating system is Windows 64-bit. The regeneration tools will run on other platforms, including Linux and OS X, but only Windows installation is documented in this guide.
(2) The target system has sufficient memory. 16GB+ is recommended.
(3) Git is installed.
(4) No other Python builds are installed to the system `PATH`, as they may interfere with the desired Anaconda Python distribution.
(5) The user has familiarity with Git, Python, and PostgreSQL.

Dependency	Purpose
spandex	Spatial data extraction functions and database interaction.
urbansim	Parcel attribute loading using simulation framework.
Python	Scripting language. Both version 2 and 3 are supported.
NumPy	Numeric data types. Included with Anaconda.
pandas	Data manipulation. Included with Anaconda.
PostgreSQL	Relational database.
PostGIS	Spatial operations.

TABLE 1. Overview of dependencies.

INSTALLATION

PostgreSQL and PostGIS. Installation of PostgreSQL and PostGIS is described in the PostGIS documentation. Be sure to install the 64-bit builds of PostgreSQL and PostGIS.

PostgreSQL can be configured for better performance, especially on high-end hardware and with large, spatial queries. The third-party web application PgTune may suggest parameters that take advantage of available hardware, like more memory. Specify "Data warehouses" for the DB Type if using PgTune.

Additionally, if the database is stored on a solid-state drive, the `random_page_cost` parameter can be set to a lower value closer to `seq_page_cost`, for example 1.1.

Restart PostgreSQL by running `services.msc`, right-clicking the service, and clicking restart.

Create a new database called `mtc`. This can be done from within pgAdmin III after connecting to the server. The database name will be later used in the spandex configuration file.

Python. Anaconda Python is developed and distributed by Continuum Analytics. Be sure to install the 64-bit build of the Anaconda Python distribution.

Then, update Anaconda.

```
conda update conda
conda update anaconda
```

urbansim. urbansim installation is described in its documentation.

```
conda config --add channels synthicity
conda install urbansim
```

Dependency	Version	Description
GDAL	1.7+	Geospatial Data Abstraction Library.
GeoAlchemy2	0.2.1+	SQLAlchemy with PostGIS.
pandas	0.15+	Data analysis library.
Psycopg2	2.5+	PostgreSQL driver.
PyGraphViz		Graphviz graph visualization (optional).
six	1.4+	Python 2 and 3 compatibility.
SQLAlchemy	0.8+	SQL toolkit and object relational mapper.

TABLE 2. spandex Python dependencies.

spandex. spandex is distributed in the spandex Git repository on GitHub.

Python dependencies are documented in the setup.py file distributed with spandex. Refer to table 2.

Some dependencies can be installed using `conda`.

```
conda install gdal pandas six sqlalchemy
```

GeoAlchemy2 can be installed using `pip`.

```
conda install sqlalchemy pip
pip install GeoAlchemy2
```

Currently, Psycopg2 is not packaged by the Anaconda distribution and must be manually installed. One workaround is to run the win-psycopg installer maintained by Jason Erickson.

Likewise, the optional dependency PyGraphViz, for plotting simulation framework relationships, is also not packaged by the Anaconda distribution. One workaround is to run the installer maintained by Christoph Gohlke, which requires installation of GraphViz.

Once spandex dependencies are satisfied, spandex itself can be installed from within the cloned Git repository using `pip`. The `-e` option installs spandex in editable mode so that it will track updates to the cloned repository.

```
git clone https://github.com/synthicity/spandex
cd spandex
pip install -e .
```

spandex should be configured with a text file. Refer to table 3 for an example file. This file should be placed in `~/.spandex/user.cfg`, where `~` represents the home directory, e.g., `C:\Users\username`. The data directory can be specified with forward slashes or double backslashes.

```
[database]
# Database configuration is passed to psycopg2.connect, which also supports
# libpq connection parameters:
# http://initd.org/psycopg/docs/module.html#psycopg2.connect

database = mtc
user = postgres
password = FIXME
host = localhost
port = 5432

[data]

# Directory containing shapefiles to import. Can be an absolute path or
# relative to the current working directory.
directory = c:/FIXME/badata

# Spatial Reference System Identifier (SRID) to reproject to.
# http://spatialreference.org/
# http://prj2epsg.org/
#
# NAD83(HARN) / California zone 3.
srid = 2768
```

TABLE 3. spandex configuration file, based on `config/example.cfg`

Regeneration. The regeneration tools are distributed with the regional UrbanSim implementation in the bayarea_urbansim Git repository on GitHub.
Current development takes place in the `spandex` branch of the repository and will be periodically merged to `master`.

```
git clone https://github.com/synthicity/bayarea_urbansim
cd bayarea_urbansim
git checkout spandex
```

RUNNING REGENERATION

The `run.py` script under the `data_regeneration` directory in the bayarea_urbansim repository will process all regeneration steps. Other scripts in that directory can be executed individually for debugging purposes.

```
python data_regeneration/load.py
```

Currently, `run.py` executes separate scripts in each regeneration step.

(1) Preprocessing: import shapefiles by county, fix invalid geometries, and reproject.
   (a) `load.py`
(2) Processing: create regional parcels table from county-specific attributes and geometries.
   (a) `counties/*.py`
   (b) `join_counties.sql`
(3) Postprocessing: tag parcels by TAZ.
   (a) `spatialops.py`

CONTACT

For feedback on this guide, please contact Dara Adib <dadib@synthicity.com>.