# leather

***Release 0.3.2***

November 11, 2016

Contents

Leather is the Python charting library for those who need charts *now* and don't care if they're perfect.

Leather isn't picky. It's rough. It gets dirty. It looks sexy just hanging on the back of a chair. Leather doesn't need your accessories. Leather is how Snake Plissken would make charts.

Get it?

Important links:

- Documentation: http://leather.rtfd.io
- Repository: https://github.com/wireservice/leather
- Issues: https://github.com/wireservice/leather/issues

CHAPTER 1

# About leather

## 1.1 Why leather?

- A readable and user-friendly API.
- Optimized for exploratory charting.
- Produces scale-independent SVG charts.
- Completely type-agnostic. Chart your data, whatever it is.
- Designed with iPython, Jupyter and atom/hydrogen in mind.
- Pure Python. No C dependencies to compile.
- MIT licensed and free for all purposes.
- Zealously zen.
- Made with love.

CHAPTER 2

# Installation

## 2.1 Users

To use leather install it with pip:

```
pip install leather
```

## 2.2 Developers

If you are a developer that also wants to hack on leather, install it from git:

```
git clone git://github.com/wireservice/leather.git
cd leather
mkvirtualenv leather

# If running Python 3 (strongly recommended for development)
pip install -r requirements-py3.txt

# If running Python 2
pip install -r requirements-py2.txt

python setup.py develop
tox
```

**Note:** To run the leather tests with coverage:

```
nosetests --with-coverage tests
```

## 2.3 Supported platforms

leather supports the following versions of Python:

- Python 2.7
- Python 3.3+
- PyPy

It is tested primarily on OSX, but due to its minimal dependencies it should work on both Linux and Windows.

**Note:** iPython or Jupyter user? Leather works great there too.

CHAPTER 3

# Examples

## 3.1 Data series

### 3.1.1 Simple pairs

```
import leather

data = [
    (0, 3),
    (4, 5),
    (7, 9),
    (8, 4)
]

chart = leather.Chart('Simple pairs')
chart.add_dots(data)
chart.to_svg('examples/charts/simple_pairs.svg')
```

### 3.1.2 Table from csv.reader

Sequence row data, such as is returned by `csv.reader()` can be accessed by specifying the indices of the columns containing the `x` and `y` values.

Note that leather does not automatically convert numerical strings, such as those stored in a CSV. If you want that you'll need to use a smarter table reader, such as agate

```
import csv

import leather

with open('examples/realdata/gii.csv') as f:
    reader = csv.reader(f)
    next(reader)
    data = list(reader)[:10]

    for row in data:
        row[1] = float(row[1]) if row[1] is not None else None

chart = leather.Chart('Data from CSV reader')
```

```
chart.add_bars(data, x=1, y=0)
chart.to_svg('examples/charts/csv_reader.svg')
```

### 3.1.3 Table from csv.DictReader

Dict row data, such as is returned by `csv.DictReader` can be accessed by specifying the indices of the columns containing the `x` and `y` values.

See previous example for note on strings from CSVs.

```
import csv

import leather

with open('examples/realdata/gii.csv') as f:
    reader = csv.DictReader(f)
    data = list(reader)[:10]

    for row in data:
        mmr = row['Maternal mortality ratio']
        row['Maternal mortality ratio'] = float(mmr) if mmr is not None else None

chart = leather.Chart('Data from CSV reader')
chart.add_bars(data, x='Maternal mortality ratio', y='Country')
chart.to_svg('examples/charts/csv_dict_reader.svg')
```

### 3.1.4 Custom data

Completely custom data formats are also supported via accessor functions.

```
import leather

data = [
    { 'x': 0, 'q': { 'y': [3] } },
    { 'x': 4, 'q': { 'y': [5] } },
    { 'x': 7, 'q': { 'y': [9] } },
    { 'x': 8, 'q': { 'y': [4] } }
]

def x(row, index):
    return row['x']

def y(row, index):
    return row['q']['y'][0]

chart = leather.Chart('Line')
chart.add_line(data, x=x, y=y)
chart.to_svg('examples/charts/custom_data.svg')
```

### 3.1.5 Multiple series

Multiple data series can be displayed on a single chart so long as they all use the same type of `Scale`.

```
import leather

data1    = [
    (0, 3),
    (4, 5),
    (7, 9),
    (8, 4)
]

data2 = [
    (2, 4),
    (7, 3),
    (6, 2),
    (5, 9)
]

chart = leather.Chart('Multiple series')
chart.add_dots(data1)
chart.add_dots(data2)
chart.to_svg('examples/charts/multiple_series.svg')
```

## 3.2 Shapes

### 3.2.1 Bars

```
import leather

data = [
    (3, 'Hello'),
    (5, 'How'),
    (9, 'Are'),
    (4, 'You')
]

chart = leather.Chart('Bars')
chart.add_bars(data)
chart.to_svg('examples/charts/bars.svg')
```

### 3.2.2 Columns

```
import leather

data = [
    ('Hello', 3),
    ('How', 5),
    ('Are', 9),
    ('You', 4)
```

```
]

chart = leather.Chart('Columns')
chart.add_columns(data)
chart.to_svg('examples/charts/columns.svg')
```

### 3.2.3 Dots

```
import leather

data = [
    (0, 3),
    (4, 5),
    (7, 9),
    (8, 4)
]

chart = leather.Chart('Dots')
chart.add_dots(data)
chart.to_svg('examples/charts/dots.svg')
```

### 3.2.4 Lines

```
import leather

data = [
    (0, 3),
    (4, 5),
    (7, 9),
    (8, 4)
]

chart = leather.Chart('Line')
chart.add_line(data)
chart.to_svg('examples/charts/lines.svg')
```

### 3.2.5 Mixing shapes

You can mix different shapes for different series on the same chart.

```
import leather

column_data = [
    ('Hello', 3),
    ('How', 5),
    ('Are', 9),
    ('You', 4)
]
```

```
line_data = [
    ('Hello', 1),
    ('How', 5),
    ('Are', 4),
    ('You', 3)
]

dot_data = [
    ('Hello', 3),
    ('How', 5),
    ('Are', 9),
    ('You', 4)
]

chart = leather.Chart('Mixed shapes')
chart.add_columns(column_data)
chart.add_line(line_data)
chart.add_dots(dot_data)
chart.to_svg('examples/charts/mixed_shapes.svg')
```

## 3.3 Scales

### 3.3.1 Linear

When using numerical data `Linear` scales are created automatically and by default. You may override the domain by adding a scale manually.

```
import leather

data = [
    (0, 3),
    (4, 5),
    (7, 9),
    (8, 4)
]

chart = leather.Chart('Linear')
chart.add_x_scale(0, 20)
chart.add_y_scale(-10, 10)
chart.add_line(data)
chart.to_svg('examples/charts/linear.svg')
```

### 3.3.2 Ordinal

When using text data `Ordinal` scales are created automatically and by default. It is generally not useful to override these defaults.

```
import leather

data = [
    ('Hello', 3),
```

```
    ('How', 5,),
    ('Are', 9),
    ('You', 4)
]

chart = leather.Chart('Ordinal')
chart.add_columns(data)
chart.to_svg('examples/charts/ordinal.svg')
```

### 3.3.3 Temporal

When using date/time data `Temporal` scales are created automatically and by default. You may override the domain by adding a scale manually.

```
from datetime import date

import leather

data = [
    (date(2015, 1, 1), 3),
    (date(2015, 3, 1), 5),
    (date(2015, 6, 1), 9),
    (date(2015, 9, 1), 4)
]

chart = leather.Chart('Temporal')
chart.add_x_scale(date(2014, 1, 1), date(2016, 1, 1))
chart.add_line(data)
chart.to_svg('examples/charts/temporal.svg')
```

## 3.4 Axes

### 3.4.1 Changing tick values

You can change the list of ticks that are displayed using `Chart.add_x_axis()` and `Chart.add_y_axis()` methods. This will not adjust automatically adjust the scale, so it is possible to pick tick values that are not displayed.

```
import leather

data = [
    (0, 3),
    (4, 5),
    (7, 9),
    (8, 4)
]

chart = leather.Chart('Line')
chart.add_x_scale(0, 10)
chart.add_x_axis(ticks=[0, 10])
chart.add_line(data)
chart.to_svg('examples/charts/ticks.svg')
```

### 3.4.2 Customizing tick format

You can provide a tick formatter method to change how ticks are displayed using the `Chart.add_x_axis()` and `Chart.add_y_axis()` methods.

```
import leather

data = [
    (0, 3),
    (4, 5),
    (7, 9),
    (8, 4)
]

def tick_formatter(value, index, tick_count):
    return '%i (%i/%i)' % (value, index, tick_count)

chart = leather.Chart('Line')
chart.add_x_axis(tick_formatter=tick_formatter)
chart.add_line(data)
chart.to_svg('examples/charts/tick_format.svg')
```

## 3.5 Styling

### 3.5.1 Changing theme values

Chart styles are set using a dead simple `theme` system. Leather is meant for making quick and dirty charts. It is neither expected nor recommended for user's to customize these styles.

```
import leather

column_data = [
    ('Hello', 3),
    ('How', 5),
    ('Are', 9),
    ('You', 4)
]

line_data = [
    ('Hello', 1),
    ('How', 5),
    ('Are', 4),
    ('You', 3)
]

dot_data = [
    ('Hello', 3),
    ('How', 5),
    ('Are', 9),
    ('You', 4)
]
```

```
leather.theme.title_font_family = 'Comic Sans MS'
leather.theme.legend_font_family = 'Comic Sans MS'
leather.theme.tick_font_family = 'Comic Sans MS'
leather.theme.default_series_colors = ['#ff0000', '#00ff00', '#0000ff']

chart = leather.Chart('Theme')
chart.add_columns(column_data)
chart.add_line(line_data)
chart.add_dots(dot_data)
chart.to_svg('examples/charts/theme.svg')
```

### 3.5.2 Changing series colors

More practically, individual default `Series` colors can be overriden when they are created.

```
import leather

data = [
    ('Hello', 3),
    ('How', 5),
    ('Are', 9),
    ('You', 4)
]

chart = leather.Chart('Series color')
chart.add_columns(data, fill_color='#000000')
chart.to_svg('examples/charts/series_color.svg')
```

### 3.5.3 Styling data based on value

Style attributes of individual data points can be set by value using a `style_function()`.

```
import random

import leather


dot_data = [(random.randint(0, 250), random.randint(0, 250)) for i in range(100)]

def colorizer(d):
    return 'rgb(%i, %i, %i)' % (d.x, d.y, 150)

chart = leather.Chart('Colorized dots')
chart.add_dots(dot_data, fill_color=colorizer)
chart.to_svg('examples/charts/colorized_dots.svg')
```

## 3.6 Chart grids

### 3.6.1 With mixed scales

You can add charts of completely different types to a single graphic by using `Grid`.

```
import leather

data1 = [
    (0, 3),
    (4, 5),
    (7, 9),
    (8, 4)
]

data2 = [
    (3, 4),
    (5, 6),
    (7, 10),
    (8, 2)
]

chart1 = leather.Chart('Dots')
chart1.add_dots(data1)

chart2 = leather.Chart('Lines')
chart2.add_line(data2)

grid = leather.Grid()
grid.add_one(chart1)
grid.add_one(chart2)
grid.to_svg('examples/charts/grid.svg')
```

### 3.6.2 With consistent scales

A grid of charts can automatically be synchronized to a consistent view using `Lattice`.

```
import leather

data1 = [
    (0, 3),
    (4, 5),
    (7, 9),
    (8, 4)
]

data2 = [
    (3, 4),
    (3, 4),
    (5, 6),
    (7, 10),
    (8, 2)
]

data3 = [
```

```
    (2, 4),
    (3, 5),
    (6, 2),
    (8, 3),
    (10, 5)
]

lattice = leather.Lattice()
lattice.add_many([data1, data2, data3], titles=['First', 'Second', 'Third'])
lattice.to_svg('examples/charts/lattice.svg')
```

CHAPTER 4

# API

## 4.1 Chart

leather.Chart

### 4.1.1 Adding data

leather.Chart.add_series
leather.Chart.add_bars
leather.Chart.add_columns
leather.Chart.add_dots
leather.Chart.add_line

### 4.1.2 Customizing

leather.Chart.set_x_scale
leather.Chart.set_y_scale
leather.Chart.set_x_axis
leather.Chart.set_y_axis

### 4.1.3 Rendering

leather.Chart.to_svg
leather.Chart.to_svg_group

#### 4.1.4 Detailed list

## 4.2 Grid

## 4.3 Lattice

## 4.4 Scales

## 4.5 Axis

## 4.6 Series

## 4.7 Shapes

## 4.8 Theme

CHAPTER 5

# Changelog

## 5.1 0.3.2 - November 11, 2016

- Fix trove classifiers.

## 5.2 0.3.1 - November 11, 2016

- Fix unicode rendering issue in Python2.7 and PyPy. (#74)

## 5.3 0.3.0 - November 11, 2016

- Add examples for many more use-cases. (#11)
- Fixed bars so that data are displayed top-down when using `Chart.add_bars()`. (#72)
- Changed default colors. (#51)
- Fixed a rare file handling bug when saving SVG files.
- Leather will now issue a warning if you attempt to render a chart with data exceeding the scale domain. (#42)
- Linear scales will now default to the domain `[0, 1]` if no values are provided. (#66)
- Axis no longer takes a number of ticks as an argument. Instead pass a list of custom tick values.
- Scales `tick` methods no longer take a number of ticks as an argument. (They should self-optimize.)
- Scales that cross `0` will now always have a tick at `0`. (#54)
- Implemented auto-ticking. (#23)
- `style_function()` now takes a `Datum` instances, rather than a list of arguments.
- Renamed the `Lines` class to `Line` to be more accurate.
- Implemented `CategorySeries`.
- Implemented a more elegant pattern for colorizing series.
- Refactored `Series` so `Shape` is no longer a parameter.
- Tick values can now be overridden with the `tick_values` argument. (#56)
- Added methods to customize scales and axes for `Lattice` charts. (#17)

- Expanded unit tests for `Scale` subclasses.
- Zero lines now render above other tick marks. (#31)
- Fixed rendering of `Bar` and `Column` shapes for negative values. (#52)
- Refactored the `Lattice` API.

## 5.4 0.2.0

- Initial prototype

## 5.5 0.1.0

- Never released

CHAPTER 6

# Release process

This is the release process for leather:

1. Verify all unit tests pass with fresh environments: `tox -r`.
2. Check test coverage: `nosetests --with-coverage tests`.
3. Ensure any new modules have been added to setup.py's `packages` list.
4. Ensure any new public interfaces have been added to the documentation.
5. Make sure the example scripts still work: `./examples.sh`.
6. Ensure `CHANGELOG.rst` is up to date. Add the release date and summary.
7. Create a release tag: `git tag -a x.y.z -m "x.y.z release."`
8. Push tags upstream: `git push --tags`
9. If this is a major release, merge `master` into `stable`: `git checkout stable; git merge master; git push`
10. Upload to PyPI: `python setup.py sdist bdist_wheel upload`.
11. Flag the release to build on RTFD.
12. Update the "default version" on RTFD to the latest.
13. Rev to latest version: `docs/conf.py`, `setup.py` and `CHANGELOG.rst` need updates.
14. Commit revision: `git commit -am "Update to version x.y.z for development."`.

CHAPTER 7

# License

CHAPTER 8

# Show me docs

- About - why you should use leather
- Install - how to install for users and developers
- Examples - code + output examples of every feature of leather
- API - technical documentation for the leather API
- Changelog - a record of every change made for each release

CHAPTER 9

# Show me code

```
import random

import leather


dot_data = [(random.randint(0, 250), random.randint(0, 250)) for i in range(100)]

def colorizer(d):
    return 'rgb(%i, %i, %i)' % (d.x, d.y, 150)

chart = leather.Chart('Colorized dots')
chart.add_dots(dot_data, fill_color=colorizer)
chart.to_svg('examples/charts/colorized_dots.svg')
```

CHAPTER 10

# Join us

- Release process - the process for maintainers to publish new releases
- License - a copy of the MIT open source license covering leather

CHAPTER 11

# Who we are

The following individuals have contributed code, documentation, or expertise to leather:

- Christopher Groskopf

CHAPTER 12

# Indices and tables

- genindex
- modindex
- search