Installing and getting started

To install the released version of seaborn, you can use `pip` (i.e. `pip install seaborn`). It's also possible to install the released version using `conda` (i.e. `conda install seaborn`), although this may lag behind the version availible from PyPI.

Alternatively, you can use `pip` to install the development version, with the command `pip install git+git://github.com/mwaskom/seaborn.git#egg=seaborn`. Another option would be to to clone the github repository (https://github.com/mwaskom/seaborn) and install with `pip install .` from the source directory. Seaborn itself is pure Python, so installation should be reasonably straightforward.

When using the development version, you may want to refer to the development docs (http://stanford.edu/~mwaskom/software/seaborn-dev/). Note that these are not built automatically and may at times fall out of sync with the actual master branch on github.

Dependencies

- Python 2.7 or 3.3+

Mandatory dependencies

- numpy (http://www.numpy.org/)
- scipy (http://www.scipy.org/)
- matplotlib (matplotlib.sourceforge.net)
- pandas (http://pandas.pydata.org/)

Recommended dependencies

- statsmodels (http://statsmodels.sourceforge.net/)

The *pip* installation script will attempt to download the mandatory dependencies if they do not exist at install-time.

I recommend using seaborn with the Anaconda distribution (https://store.continuum.io/cshop/anaconda/), as this makes it easy to manage the main dependencies, which otherwise can be difficult to install.

I attempt to keep seaborn importable and generally functional on the versions available through the stable Debian channels. There may be cases where some more advanced features only work with newer versions of these dependencies, although these should be relatively rare.

There are also some known bugs on older versions of matplotlib, so you should in general try to use a modern version. For many use cases, though, older matplotlibs will work fine.

Seaborn is tested on the most recent versions offered through `conda`.

Importing seaborn

Seaborn will apply its default style parameters to the global matplotlib style dictionary when you import it. This will change the look of all plots, including those created by using matplotlib functions directly. To avoid this behavior and use the default matplotlib aesthetics (along with any customization in your `matplotlibrc`), you can import the `seaborn.apionly` namespace.

Seaborn has several other pre-packaged styles along with high-level tools (tutorial/aesthetics.html#aesthetics-tutorial) for managing them, so you should not limit yourself to the default aesthetics.

By convention, `seaborn` is abbreviated to `sns` on import.

Testing

To test seaborn, run `make test` in the root directory of the source distribution. This runs the unit test suite (which can also be exercised separately by running `nosetests`). It also runs the code in the example notebooks to smoke-test a broader and more realistic range of example usage.

The full set of tests requires an internet connection to download the example datasets, but the unit tests should be able to run offline.

Bugs

Please report any bugs you encounter through the github issue tracker (https://github.com/mwaskom/seaborn/issues/new). It will be most helpful to include a reproducible example on one of the example datasets (accessed through `load_dataset()`). It is difficult debug any issues without knowing the versions of seaborn and matplotlib you are using, as well as what matplotlib backend you are using to draw the plots, so please include those in your bug report.

Known issues

There is a bug (https://github.com/matplotlib/matplotlib/issues/2654) in the matplotlib OSX backend that causes unavoidable problems with some of the seaborn functions (particularly those that draw multi-panel figures). If you encounter this, you will want to try a different backend (http://matplotlib.org/api/matplotlib_configuration_api.html). In particular, this bug affects any multi-panel figure that internally calls the matplotlib `tight_layout` function.

An unfortunate consequence of how the matplotlib marker styles work is that line-art markers (e.g. `"+"`) or markers with `facecolor` set to `"none"` will be invisible when the default seaborn style is in effect. This can be changed by using a different `markeredgewidth` (aliased to `mew`) either in the function call or globally in the *rcParams*.
