

Figure 4-72. Annotated average birth rates by day

You'll notice that the specifications of the arrows and text boxes are very detailed: this gives you the power to create nearly any arrow style you wish. Unfortunately, it also means that these sorts of features often must be manually tweaked, a process that can be very time-consuming when one is producing publication-quality graphics! Finally, I'll note that the preceding mix of styles is by no means best practice for presenting data, but rather included as a demonstration of some of the available options.

More discussion and examples of available arrow and annotation styles can be found in the Matplotlib gallery, in particular http://matplotlib.org/examples/pylab_examples/annotation_demo2.html.

Customizing Ticks

Matplotlib's default tick locators and formatters are designed to be generally sufficient in many common situations, but are in no way optimal for every plot. This section will give several examples of adjusting the tick locations and formatting for the particular plot type you're interested in.

Before we go into examples, it will be best for us to understand further the object hierarchy of Matplotlib plots. Matplotlib aims to have a Python object representing everything that appears on the plot: for example, recall that the `figure` is the bounding box within which plot elements appear. Each Matplotlib object can also act as a container of sub-objects; for example, each `figure` can contain one or more axes objects, each of which in turn contain other objects representing plot contents.

The tick marks are no exception. Each `axes` has attributes `xaxis` and `yaxis`, which in turn have attributes that contain all the properties of the lines, ticks, and labels that make up the axes.

Major and Minor Ticks

Within each axis, there is the concept of a *major* tick mark and a *minor* tick mark. As the names would imply, major ticks are usually bigger or more pronounced, while minor ticks are usually smaller. By default, Matplotlib rarely makes use of minor ticks, but one place you can see them is within logarithmic plots (Figure 4-73):

```
In[1]: %matplotlib inline
import matplotlib.pyplot as plt
plt.style.use('seaborn-whitegrid')
import numpy as np

In[2]: ax = plt.axes(xscale='log', yscale='log')
```

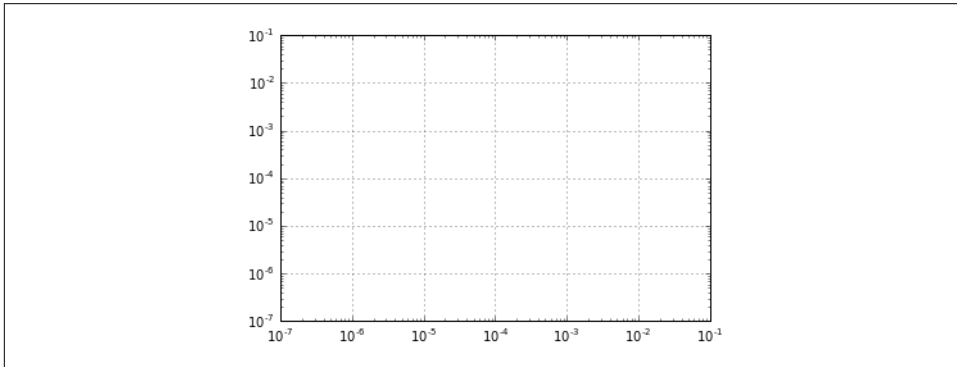


Figure 4-73. Example of logarithmic scales and labels

We see here that each major tick shows a large tick mark and a label, while each minor tick shows a smaller tick mark with no label.

We can customize these tick properties—that is, locations and labels—by setting the formatter and locator objects of each axis. Let's examine these for the x axis of the plot just shown:

```
In[3]: print(ax.xaxis.get_major_locator())
print(ax.xaxis.get_minor_locator())

<matplotlib.ticker.LogLocator object at 0x107530cc0>
<matplotlib.ticker.LogLocator object at 0x107530198>

In[4]: print(ax.xaxis.get_major_formatter())
print(ax.xaxis.get_minor_formatter())

<matplotlib.ticker.LogFormatterMathtext object at 0x107512780>
<matplotlib.ticker.NullFormatter object at 0x10752dc18>
```

We see that both major and minor tick labels have their locations specified by a `LogLocator` (which makes sense for a logarithmic plot). Minor ticks, though, have their labels formatted by a `NullFormatter`; this says that no labels will be shown.