Also surprisingly, the 80-year-old women seem to outperform *everyone* in terms of their split time. This is probably due to the fact that we're estimating the distribution from small numbers, as there are only a handful of runners in that range:

```
In[38]: (data.age > 80).sum()

Out[38]: 7
```

Back to the men with negative splits: who are these runners? Does this split fraction correlate with finishing quickly? We can plot this very easily. We'll use `regplot`, which will automatically fit a linear regression to the data (Figure 4-132):

```
In[37]: g = sns.lmplot('final_sec', 'split_frac', col='gender', data=data,
                       markers=".", scatter_kws=dict(color='c'))
        g.map(plt.axhline, y=0.1, color="k", ls=":");
```

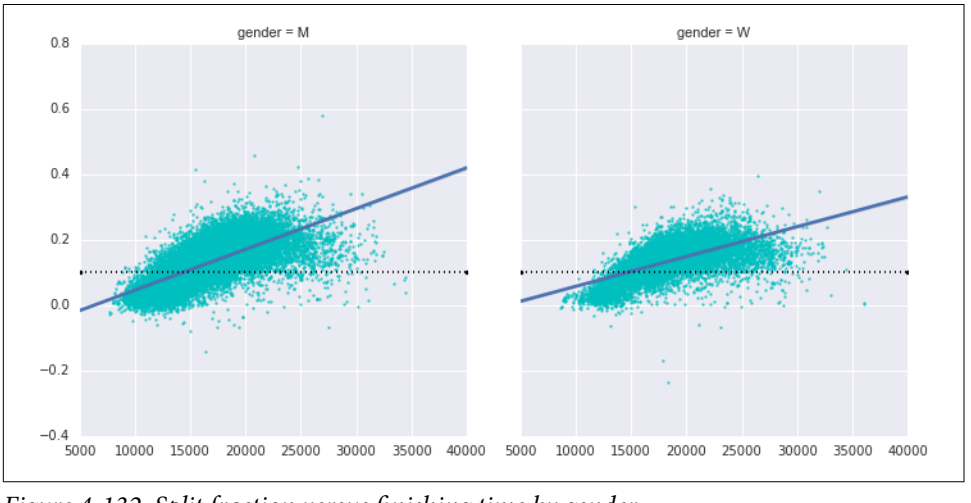

Figure 4-132. Split fraction versus finishing time by gender

Apparently the people with fast splits are the elite runners who are finishing within ~15,000 seconds, or about 4 hours. People slower than that are much less likely to have a fast second split.

Further Resources

Matplotlib Resources

A single chapter in a book can never hope to cover all the available features and plot types available in Matplotlib. As with other packages we've seen, liberal use of IPython's tab-completion and help functions (see "Help and Documentation in IPython" on page 3) can be very helpful when you're exploring Matplotlib's API. In addition, Matplotlib's online documentation can be a helpful reference. See in particular the

Matplotlib gallery linked on that page: it shows thumbnails of hundreds of different plot types, each one linked to a page with the Python code snippet used to generate it. In this way, you can visually inspect and learn about a wide range of different plotting styles and visualization techniques.

For a book-length treatment of Matplotlib, I would recommend *Interactive Applications Using Matplotlib*, written by Matplotlib core developer Ben Root.

Other Python Graphics Libraries

Although Matplotlib is the most prominent Python visualization library, there are other more modern tools that are worth exploring as well. I'll mention a few of them briefly here:

- Bokeh is a JavaScript visualization library with a Python frontend that creates highly interactive visualizations capable of handling very large and/or streaming datasets. The Python frontend outputs a JSON data structure that can be interpreted by the Bokeh JS engine.
- Plotly is the eponymous open source product of the Plotly company, and is similar in spirit to Bokeh. Because Plotly is the main product of a startup, it is receiving a high level of development effort. Use of the library is entirely free.
- Vispy is an actively developed project focused on dynamic visualizations of very large datasets. Because it is built to target OpenGL and make use of efficient graphics processors in your computer, it is able to render some quite large and stunning visualizations.
- Vega and Vega-Lite are declarative graphics representations, and are the product of years of research into the fundamental language of data visualization. The reference rendering implementation is JavaScript, but the API is language agnostic. There is a Python API under development in the Altair package. Though it's not mature yet, I'm quite excited for the possibilities of this project to provide a common reference point for visualization in Python and other languages.

The visualization space in the Python community is very dynamic, and I fully expect this list to be out of date as soon as it is published. Keep an eye out for what's coming in the future!