Statistics for Astronomers
Solutions to Homework #5

Prof. Sundar Srinivasan

November 19, 2020

Note: the solutions below use the script `hw5.py`.

1. The module `hw5p1` in the script produces the left panel in Figure 1, which compares the distribution resulting from 100, 1000, and 10000 bootstrap resamples. The shape of the distribution is decided primarily by the content of the original dataset, and isn't very sensitive to the number of resamples. The sparse nature of the dataset results in a wide, mostly flat resampled distribution for the sample median. The code prints out

```
The mean of the distribution of the sample median for Nboot = 100 is 4.36.
The mean of the distribution of the sample median for Nboot = 1000 is 4.33.
The mean of the distribution of the sample median for Nboot = 10000 is 4.32.
```

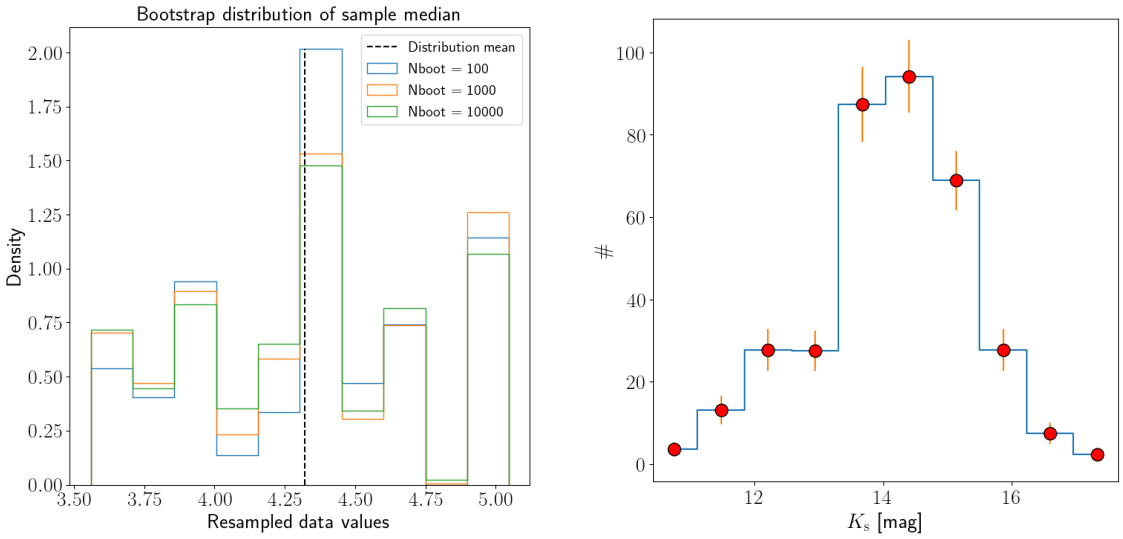

Figure 1: *Left*: bootstrap distributions for 100, 1000, and 10000 resamples for the median. The dashed line shows the mean of the distributions for the median. *Right*: Histogram of K_s magnitudes from the original dataset (blue line) with the mean and standard deviation of the bootstrap distributions overlaid (red circles and vertical line segments).

2. The `hw5p2` module in the script performs 100 bootstrap resamples of the magnitudes from the data file, and computes a histogram each time. The bin edges are set by using `numpy.histogram` on the

original dataset, then these bin edges are fed to each subsequent `numpy.histogram` call, one for each bootstrap resample. This procedure results in 100 bootstrapped counts for each bin, which can be used to compute a mean count and a standard deviation in the counts for each bin. The right panel in Figure 1 shows the histogram of the original dataset over which the bootstrap mean is overlaid (red circles) along with the standard deviations as error bars (red vertical line segments).

3. The module `hw5p3` first computes the least-squares estimators $\widehat{\beta}$ and $\widehat{\alpha}$ from the original dataset. It prints out

   `Least-square estimators of slope and intercept:` -5.43 `and` 12.18

   The code then performs 100 paired bootstrap resamples from the original dataset, and computes the slope and intercept from the least-squares estimator formulae for each resample, generating a distribution for the two quantities. It computes the empirical distribution and identifies the 16^{th} and 84^{th} percentiles, corresponding to the lower and upper limits of the 68% central confidence interval. The code prints out

   `68% central CI around mode for slope:` $[-5.8, -5.03]$
   `68% central CI around mode for intercept:` $[9.94, 13.91]$

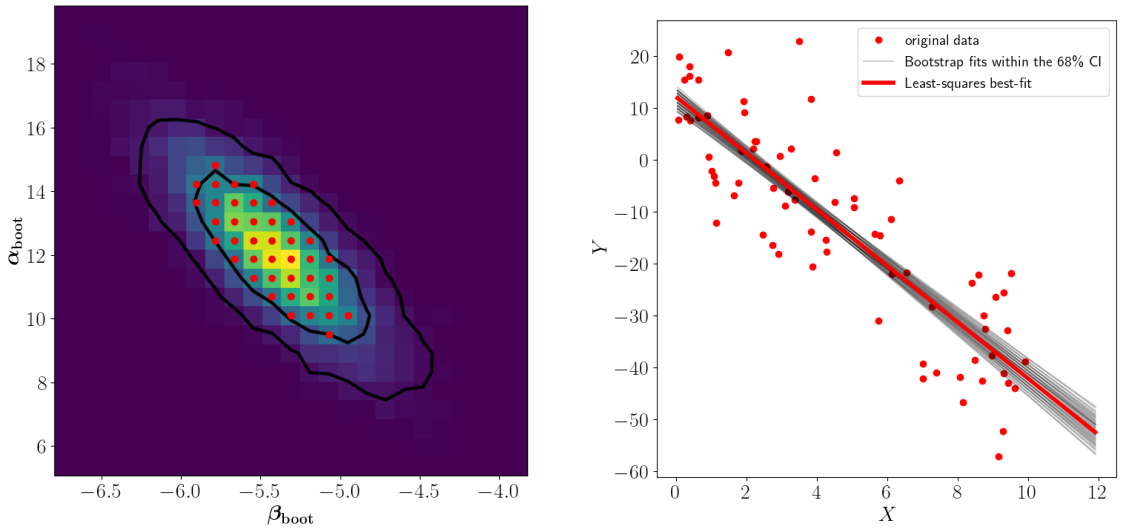

Figure 2: *Left*: joint distribution of the slope and intercept based on 10000 paired bootstrap resamples, with the 68% and 95% confidence regions overlaid (black). The bin centres of the (slope, intercept) pairs within the 68% are also shown (red circles). *Right*: Gray lines showing least-squares linear fits with (slope, intercept) values corresponding to the bins in the left panel are overlaid onto the original dataset (red circles). The red line is the least-squares fit using the slope and intercept computed from the original data.

This is it as far as the original problem which, however, oversimplifies the estimation of confidence intervals for a 2-dimensional case. While the procedure undertaken above works fine for one parameter, in the case of multiple parameters there are, in general, covariances that must be accounted for; the confidence interval is now a **confidence region** enclosing the central 68% of (slope, intercept) pairs around their most likely values. A better way to compute a realistic confidence interval is

executed by calling the `hw5p3` module with the `contours = True` option. The method works better if the number of resamples is increased (as the number of dimensions has increased, the probability is smeared out over a larger "volume". When discretised in the form of a 2D histogram, this reduces the resolution), so we also set `Nboot = 10000`.

The code now computes and displays the bootstrapped **joint distribution** of the slope and intercept (using `pyplot.hist2d`), as shown in the left panel of Figure 2. The counts in the bins are converted to cumulative values and normalised, thus generating the joint cumulative distribution. The code then calls `pyplot.contour` to display the overlay cumulative distribution onto the same figure, with contours for the 68% and 95% confidence regions. The shape and orientation of the contours clearly reflects the linear correlation between the least-squares values of the slope and intercept as defined in the question, showing that the simplistic method employed to obtain separate CIs for the slope and intercept is not the correct approach to this problem.

The code also identifies all (slope, intercept) pairs within the 68% confidence region. These pairs are used to compute the least-squares lines that are overlaid onto the original dataset (right panel of Figure 2).