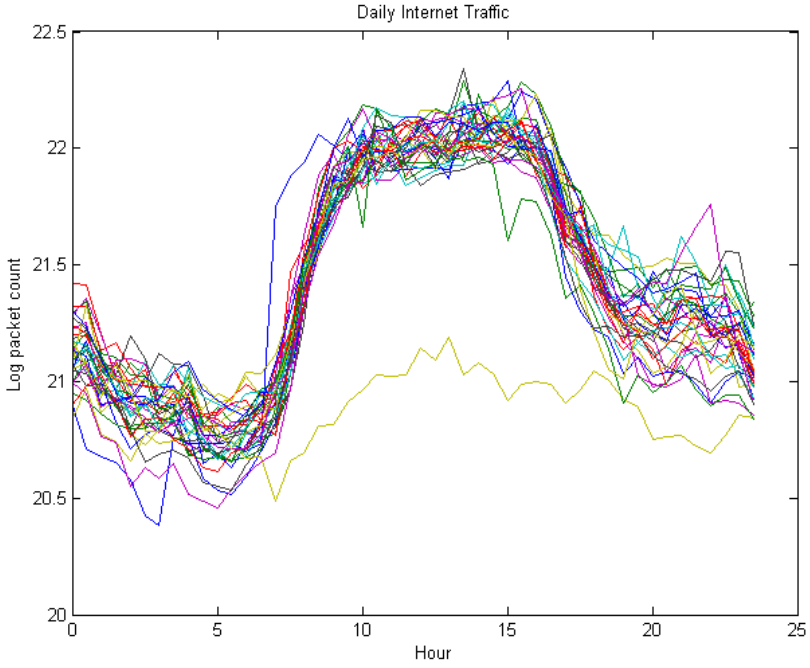
Data analysis using Matlab functions SpMed and SpPC

This data analysis illustrates the use of the Matlab functions SpMed.mat and SpPC.mat. It is assumed you have already read Gervini (2008), where the estimators were introduced.

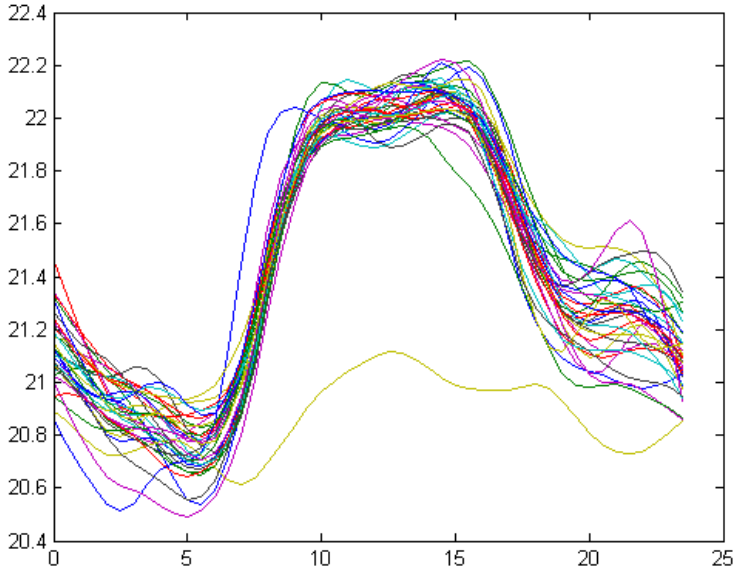
Here we analyze Internet traffic data from the University of South Carolina. The data was provided by Lingsong Zhang (see Zhang, Marron, Shen and Zhu, 2007). The observations are daily trajectories of packet counts every ½ hour, so there are 48 observations per curve, for a total of 35 week days. One of these days, however, was the 4^{th} of July (US holyday), so the curve is an outlier. This is easy to see in the accompanying figure (yellow line). There is another outlying curve, although less pronounced: the blue curve that spikes up earlier than normal.



Before computing the estimators, we do a quick-and-dirty spline smoothing of the data (the function *bspl* that computes B-splines is available on my website):

```
>> B = bspl(t,4,t,0);
>> B2 = bspl(t,4,t,2);
>> xx = x*B*inv(B'*B+0.1*B2'*B2)*B';
```

The smoothed data looks like this:



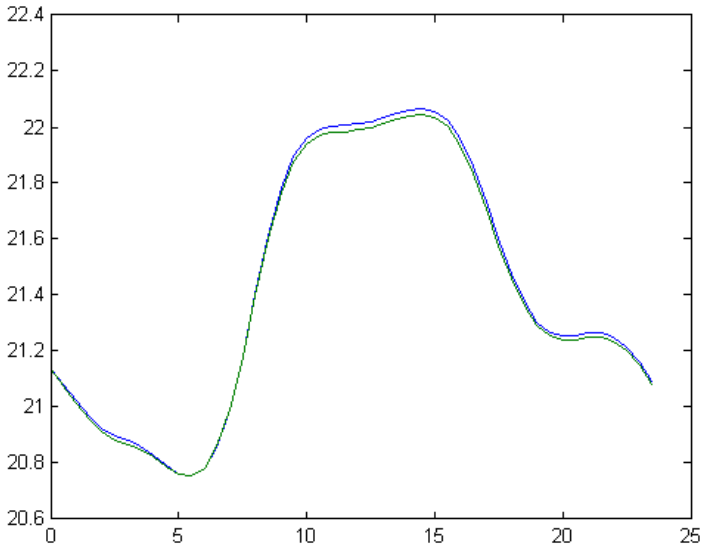
We didn't lose much information by smoothing, but got rid of considerable noise.

To compute the spatial median we do

```
>> [med,wmu,norms] = SpMed(t,xx,'s');
```

where *t* is the time vector and *xx* is the (smoothed) data matrix. The option "s" is used because we're using smooth data. To compare the spatial median with the sample mean we do

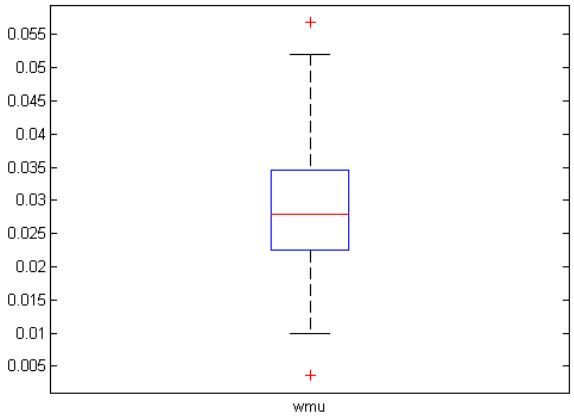
```
>> plot(t,med,t,mean(xx))
```



We see that there isn't much of a difference between the mean and the median. The mean (green line) is always below the median (blue line) because the "4th of July" outlier pulls it down, but the effect is not very important for practical purposes.

To identify potential outliers, boxplots of the data weigths (*wmu*) or the distances to the median (*norms*) may be instructive. For example, we do

```
>> boxplot(wmu,'labels','wmu')
```



and see one observation with very small weight, which turns out to be the "4th of July" outlier (note that the median of the weights is .0279, very close to 1/35 = .0286).

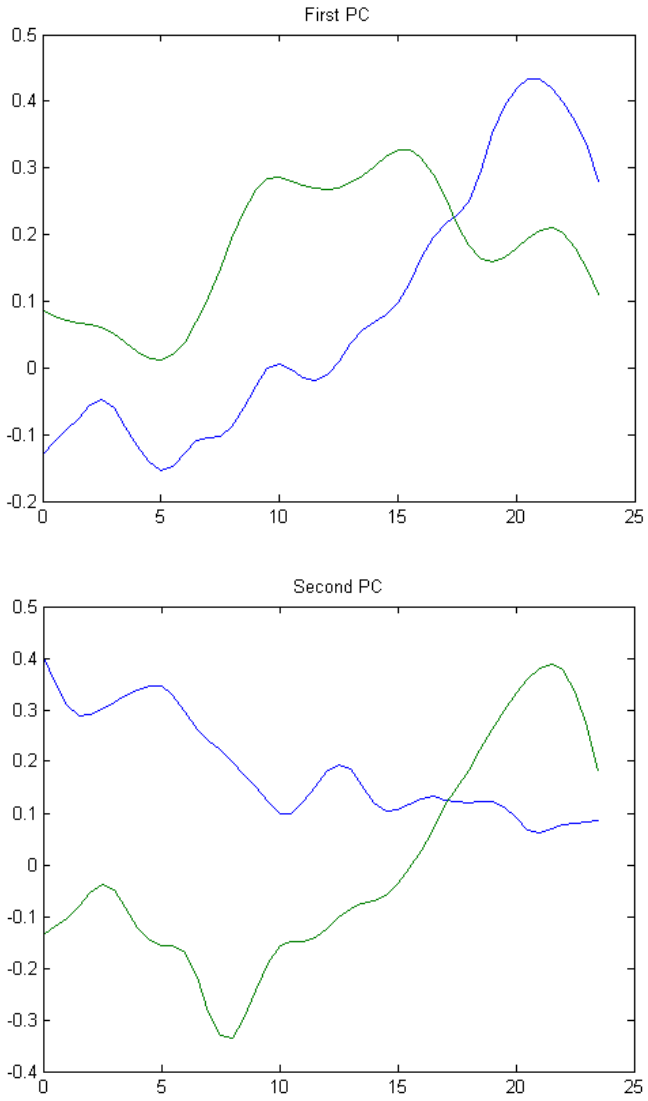
We now compute the spherical principal components (say, 10 of them) and the sample principal components for comparison:

```
>> [pc,wpc,lambda,scores] = SpPC(t,xx,10,wmu,'s');
>> [rawpc,rawlambda,rawscores] = RawPC(t,xx,10,'s');
```

Now the differences between the robust and non-robust estimators are striking:

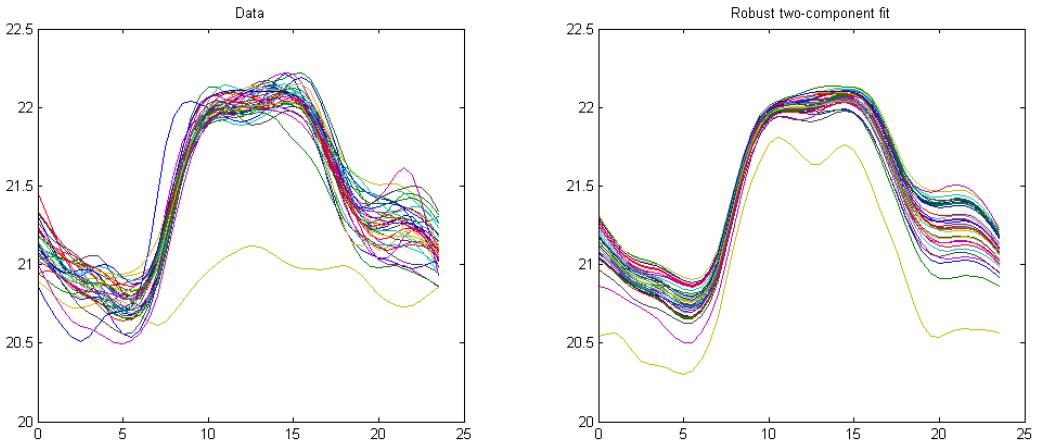
```
>> plot(t,pc(1,:),t,rawpc(1,:))
>> figure, plot(t,pc(2,:),t,rawpc(2,:))
```

In the plots below we see that the first sample pc (green line) is parallel to the mean, so it's basically accounting for vertical-shift variability, whereas the first spherical pc (blue line) is more of a contrast: negative (curve under the median) until 10 am and positive (curve above the median) thereafter. Obviously the discrepancy is due to the "4th of July" outlier, which causes overestimation of the vertical-shift variability. The second components are also very different: the spherical pc (blue line) is positive, so it's a vertical shift effect (with emphasis on the early hours), while the sample pc (green line) is a contrast (negative before 3 pm, positive thereafter).



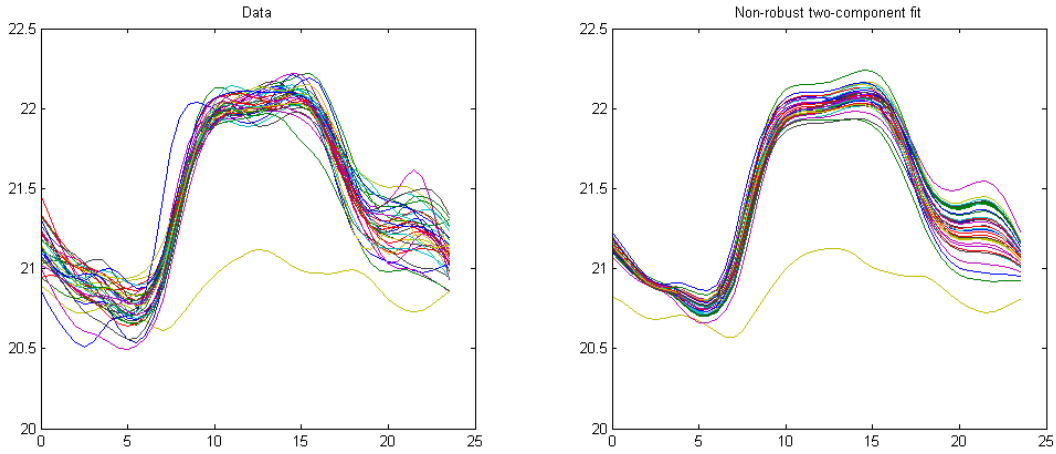
Of course, that the methods produce different components does not necessarily imply that the robust spherical components are the "right" ones. So it's instructive to take a look at the approximation errors. So we do:

```
>> xhat = ones(35,1)*med + scores(:,1:2)*pc(1:2,:);
>> subplot(121),plot(t,xx)
>> axis([0 25 20 22.5])
>> subplot(122),plot(t,xhat)
>> axis([0 25 20 22.5])
```



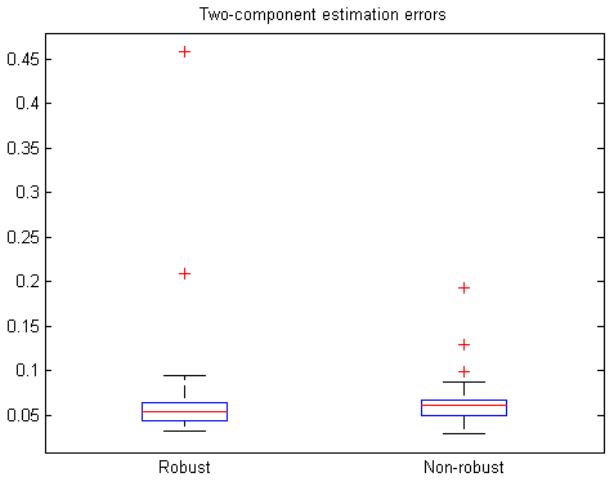
We see that the approximation using the first two spherical PCs is very good, except for the outlying curves (as expected). On the other hand, if we use the sample principal components:

```
>> rawxhat = ones(35,1)*mean(xx) + rawscores(:,1:2)*rawpc(1:2,:);
>> subplot(121),plot(t,xx)
>> axis([0 25 20 22.5])
>> subplot(122),plot(t,rawxhat)
>> axis([0 25 20 22.5])
```

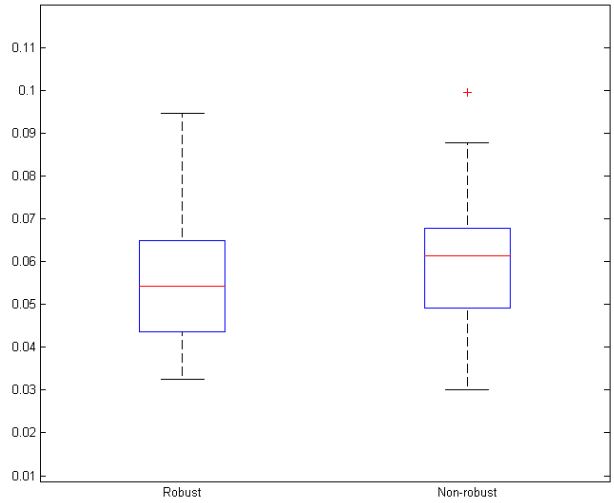


we see that the fit is good for the "4th of July" outlier but not for the rest of the data. We can better compare the estimation errors using boxplots:

```
>> errsph = sqrt(mean((xx-xhat).^2,2));
>> errraw = sqrt(mean((xx-rawxhat).^2,2));
>> boxplot([errsph errraw],'labels',{'Robust';'Non-robust'})
```



The robust fit shows two large approximation errors (the two outlying curves) but the overall fit is much better, as can be seen by the smaller median. For better visualization, we show the boxplots with outliers trimmed:



It is clear that the robust estimators provide a better fit for the central part of the data than the non-robust estimators.

References

1. Gervini, D. (2008). Robust functional estimation using the spatial median and spherical principal components. *Biometrika* **95** 587–600.
2. Zhang, L., Marron, J. S., Shen, H. and Zhu, Z. (2007). Singular value decomposition and its visualization. *Journal of Computational and Graphical Statistics* **16** 833–854.