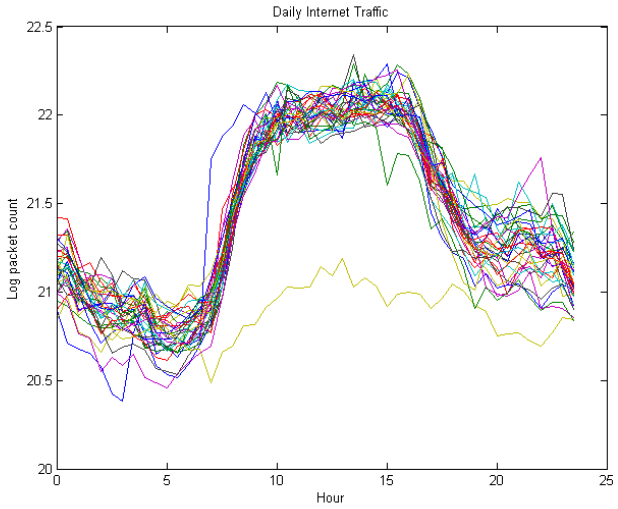
# Data analysis using Matlab functions EMt0 and EMt

With this data analysis we illustrate the use of the Matlab functions EMt0.mat and EMt.mat, which compute the estimators introduced in Gervini (2008). The data we analyze are Internet traffic data from the University of South Carolina, provided by Lingsong Zhang (see Zhang, Marron, Shen and Zhu, 2007). The observations are daily trajectories of packet counts measured every ½ hour, so there are 48 observations per curve, for a total of 35 week days. One of these days, the 4$^{th}$ of July, is a US holyday and then an outlying trajectory when compared with working days. The "4$^{th}$ of July" outlier is the yellow line in the plot below. There is another outlying curve, although less pronounced: the blue curve that spikes up earlier than normal.




Although a *q*-component model could be estimated directly from arbitrary starting points, it's safer to build up from a mean-only model (*q=0*), adding one component at the time until the desired *q*. The advantage of this procedure is that arbitrary starting points are required for only one of the principal components (the last one), because for the other *q-1* components we can use as initial estimators the final estimators of the previous model.

Concretely, we proceed as follows. The mean-only model estimators are computed with the function EMt0.mat:

```
>> [theta,sigma2,w] = EMt0(id,t,x,a,b,theta,sigma2,nk,order,nu,maxit)
```

The input requires some explanation because the function is tailored to irregularly sampled data. So *id*, *t* and *x* are vectors of equal length; *t* is "time" (the points where the curves are observed) and *x* are the observed values of the curves. The variable *id* are labels that identify the individuals; the actual values do not matter very much, as long as they are the same for each individual and different for different individuals. The variables

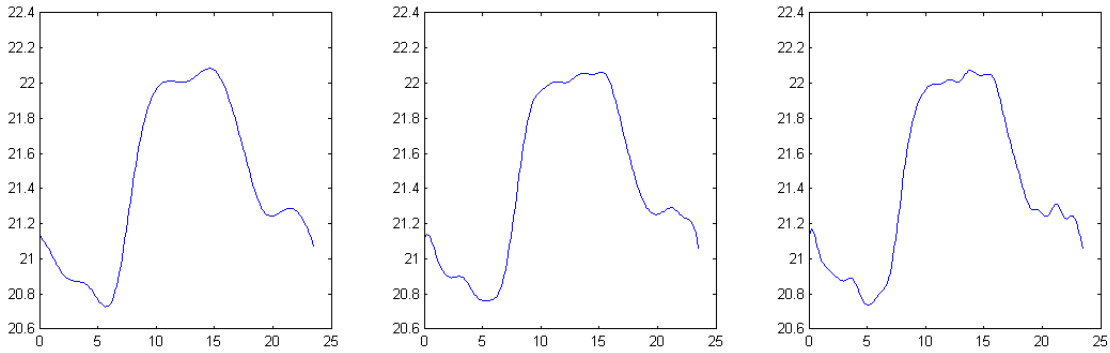
*a* and *b* are the endpoints of the "time" interval on which the mean and the components are estimated, which are not necessarily the smallest and largest *t*, respectively; when dealing with irregular or missing data, estimation can be hard at the borders, so a narrower interval may be advisable.

The input vector *theta* is the initial estimator of the spline coefficients of the mean, and the input *sigma2* is the initial estimator of the error variance. We're going to use empty vectors as input, which triggers the default values: zeros for *theta*, and the pooled sample variance for *sigma2*. The variable *nk* is the number of knots for the spline basis; the program will use *nk* equispaced knots within the interval [*a*,*b*], not counting *a* and *b*. The variable *order* is the spline order, typically chosen as 4 (cubic splines). The variable *nu* is the degrees of freedom of the *t* model; we take *nu=1*, the Cauchy model; *maxit* is the maximum number of iterations.

After creating the appropriate vectors *id*, *t* and *x*, we call the program with three different numbers of knots (10, 20 and 30), just to see what the estimators look like:

```
>> [theta1,sigma2,w] = EMt0(id,t,x,0,23.5,[],[],10,4,1,500);
>> [theta2,sigma2,w] = EMt0(id,t,x,0,23.5,[],[],20,4,1,500);
>> [theta3,sigma2,w] = EMt0(id,t,x,0,23.5,[],[],30,4,1,500);
>> tt = linspace(0,23.5,200);
>> knots1 = linspace(0,23.5,12);
>> knots2 = linspace(0,23.5,22);
>> knots3 = linspace(0,23.5,32);
>> B1 = bspl(tt,4,knots1,0);
>> B2 = bspl(tt,4,knots2,0);
>> B3 = bspl(tt,4,knots3,0);
>> subplot(131),plot(tt,B1*theta1)
>> subplot(132),plot(tt,B2*theta2)
>> subplot(133),plot(tt,B3*theta3)
```

Note that when calling *bspl*, the knot sequence must include the interval endpoints *a* and *b*; that's why, for example, if we call EMt0 with *nk=10* then we have to create a vector *knots* of 12 equispaced points to plot the mean. Also note that *maxit* has to be large, since the EM algorithm is fast but takes many iterations to converge (for this data, it took about 150 iterations in less than 9 seconds on a 2 GHz Pentium notebook). The three means are shown below:



Apparently there isn't much gain in using more than 10 knots; the left-most panel already shows all the interesting features of the function. So we'll use *nk=10* from now on. We call the function again:

```
>> [thetaC0,sigma2C0,wC0] = EMt0(id,t,x,0,23.5,[],[],10,4,1,500);
```

The "C0" suffix stands for Cauchy model, *q=0*. The variable *wC0* are the individual weights that define the estimator (which is a weighted average); unusually small weights would single out the outliers, but we do not detect anything unusual in the boxplot.

Now we're going to estimate a one-component model (*q=1*) using the function EMt. It's run in a similar (but not identical) way to EMt0:

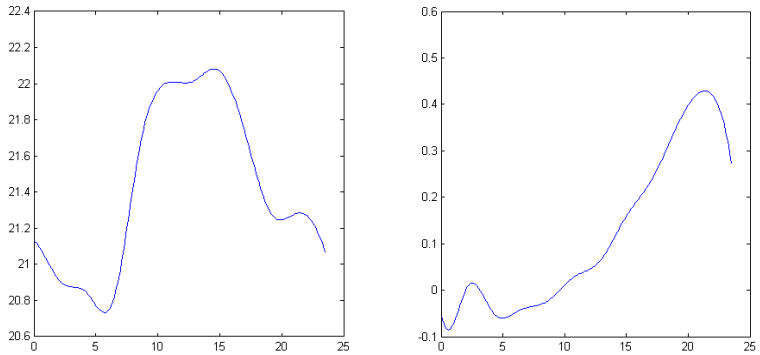
```
>> [theta,etas,lambdas,sigma2,y,xhat,w] = …
                EMt(id,t,x,a,b,theta,etas,lambdas,sigma2,nk,order,nu,maxiter)
```

The input variable *theta*, as before, is the initial estimator of the spline coefficients of the mean function, for which we can use *thetaC0*. Also, the input variable *sigma2* is the initial estimator of the error variance, for which we use *sigma2C0*. The variable *etas* is a matrix with the spline coefficients of the principal components, and *lambdas* is a row vector with the corresponding eigenvalue estimators. Since one normally doesn't have a clue what these values will be like, as initial estimators we take a vector of ones for *etas* and *2*sigmaC0* for *lambdas*. Then we run

```
>> [thetaC1,etasC1,lambdasC1,sigma2C1,yC1,xhatC1,wC1] =  …
EMt(id,t,x,0,23.5,thetaC0,ones(14,1),2*sigma2C0,sigma2C0,10,4,1,500);
```

and plot the resulting mean and principal component,

```
>> subplot(121),plot(tt,B1*thetaC1)
>> subplot(122),plot(tt,B1*etasC1)
```




Note that there isn't much difference between this mean and the mean obtained before for *q=0*. The first principal component is a contrast (mostly negative between 0 and 10, and positive thereafter). The eigenvalue estimator *lambdasC1* is 0.1745 and the error variance

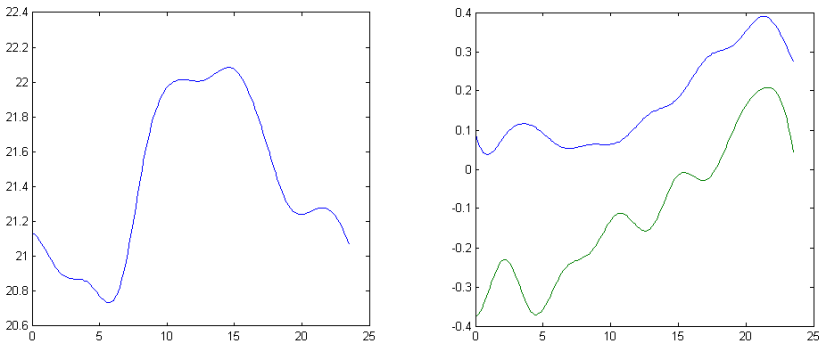
estimator *sigma2C1* is 0.0116, so there probably is some more systematic variability to be modeled and we will try a two-component model.

To estimate a two-component model we run

```
>> [thetaC2,etasC2,lambdasC2,sigma2C2,yC2,xhatC2,wC2] =  …
        EMt(id,t,x,0,23.5,thetaC1,[etasC1 ones(14,1)], …
        [lambdasC1;lambdasC1(end)/2],sigma2C1,10,4,1,500);
```

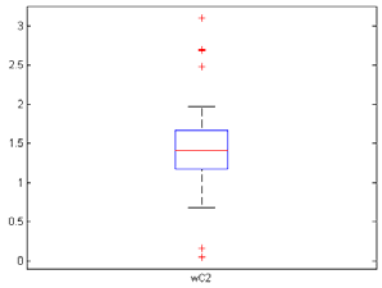
Note how we build up the initial *etas* and *lambdas*: a more or less random guess is appended at the end of the one-component model estimators *etasC1* and *lambdasC1*. A plot of the mean and principal components is obtained as follows:

```
>> subplot(121),plot(tt,B1*thetaC2)
>> subplot(122),plot(tt,B1*etasC2)
```

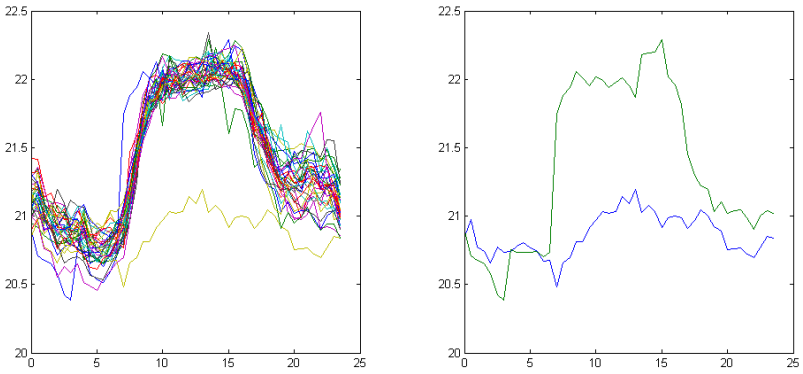



The green line on the right-hand side would be the second component. The mean is practically identical to the mean of previous model, but the first principal component is slightly different: it has the same shape but is everywhere positive, so it's not quite a contrast, it's a vertical-shift effect although with more emphasis after 10 am. The eigenvalue estimators (*lambdasC2*) are 0.218 and 0.081, and the error variance estimator *sigma2C2* is 0.0095, so there's room for more principal components.

To detect potential outliers, it's instructive to take a look at a boxplot of *wC2*:

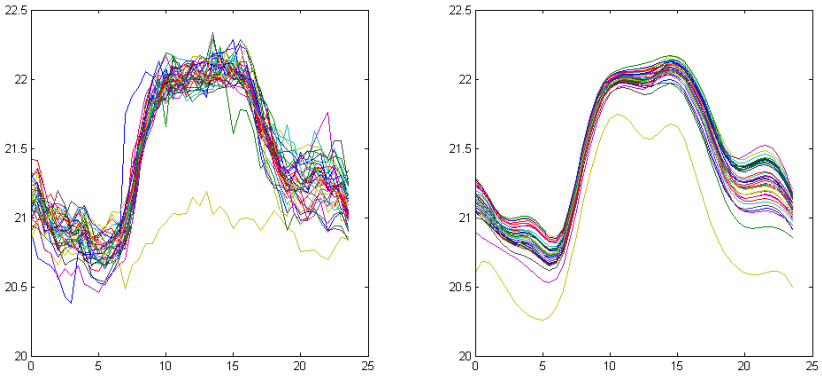



Clearly, there are two observations with suspiciously small weights that may be outliers. These two are plotted on the right-hand side panel below, next to the whole dataset:




They happen to be the two outliers mentioned at the beginning: the "4th of July" and the curve that spikes up earlier than the others.

Another interesting plot to look at is the plot of the estimated trajectories, which are given in the output variable *xhatC2* (plotted on the right-hand side, next to the raw data):




We see that the estimated trajectories are pretty accurate, except of course for the outliers, as it is to be expected of a robust method.

## References

1. Gervini, D. (2008). Robust estimation of the mean and principal components for sparse functional data. *Submitted*.
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.