Max-Linear Models

Jean-Yves Djamen

Fall 2019

Contents

Given a graph, Generate ML Coefficient Matrix

We will assume that the input is a graph $\mathcal{D} = (V, E)$ with n vertices. Our function `ml_gen` will output the ML coefficient matrix B as shown bellow.

```
m<- 4
m_mat<-c(2,3,4,1,
         0,2,0,4,
         0,2,3,5,
         0,0,0,2)
coef.mat<-ml_gen(m,m_mat)
```

Generating some data

To generate the data, we need to use the coefficient matrix B. From the analysis of this matrix, we can see that any comlumn i denotes the maximum path between node i and each of its ancestors (denoted $An(i)$). In each column, a zero in the j^{th} element means that there is no path between i and j. So, to build the function that will generate data from the coefficient matrix, we compute the element wise product of our random vector Z and each column B_i. Then, the maximum element in each of these will give us the value for our recursive structural model.

```
#given a max linear matrix and an optional input, generate some data
data_gen<-function(ml.mat, dist="exponential",lambda=1, s=1, alpha=1, m=0){
  #number of rvs to create
  n<-nrow(ml.mat)
  if(dist=="frechet"){
    #s is scale, alpha is shape and m is location
    z<-rfrechet(n, loc=m, scale=s, shape=alpha)
  }
  else{
    z<-rexp(n, rate=1)
  }
  X=c()
  #Now, z is our matrix of n samples
  for(col in 1:n){
    #take each of the columns in the Coefficient matrix
    row=ml.mat[,col]
    #compute the element wise product (b_{ji}Z_j)
    #intuitively each column is the b_{ji} element
    prod=(z*row)
    #we find X.i by maximizing all of these
    X.i=max(prod)
    X<-c(X,X.i)
  }
  print(ml.mat)
  return(data.frame("noise"=z,"data"=X ))
}
lete=data_gen(coef.mat,dist="exponential")
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    2   16    8   64
## [2,]    0    2    0    8
## [3,]    0    6    3   24
```

```
## [4,]    0    0    0    2
```

```
typeof(lete)
```

```
## [1] "list"
```

```
exponential.1<-data_gen(coef.mat,dist="exponential")
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    2   16    8   64
## [2,]    0    2    0    8
## [3,]    0    6    3   24
## [4,]    0    0    0    2
```

```
exponential.2<-data_gen(coef.mat,dist="exponential")
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    2   16    8   64
## [2,]    0    2    0    8
## [3,]    0    6    3   24
## [4,]    0    0    0    2
```

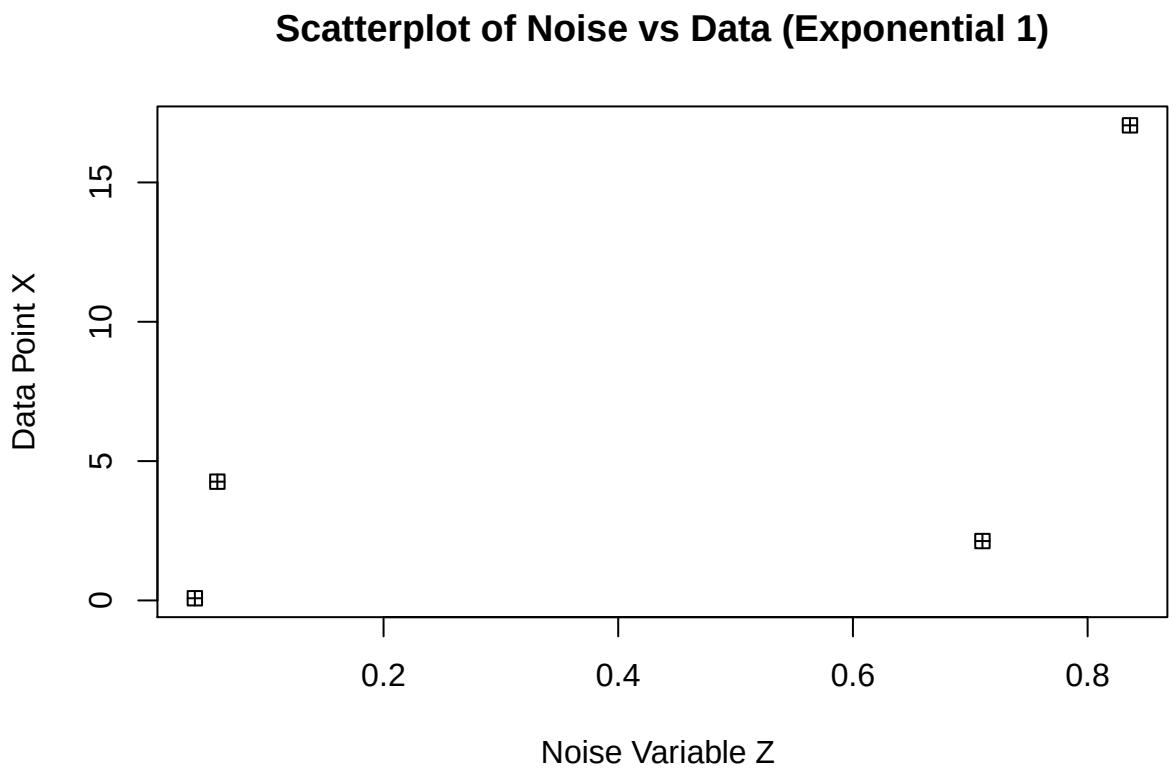
```
frechet.1<-data_gen(coef.mat,dist="frechet")
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    2   16    8   64
## [2,]    0    2    0    8
## [3,]    0    6    3   24
## [4,]    0    0    0    2
```

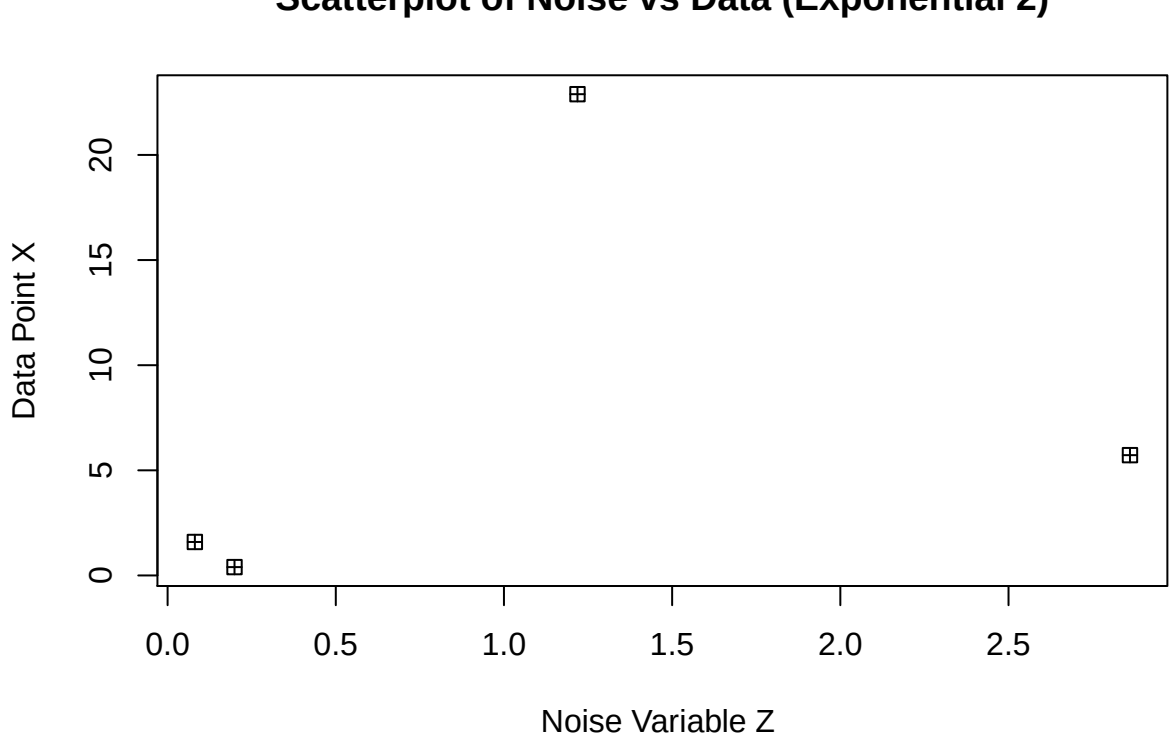
```
frechet.2<-data_gen(coef.mat,dist="frechet")
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    2   16    8   64
## [2,]    0    2    0    8
## [3,]    0    6    3   24
## [4,]    0    0    0    2
```

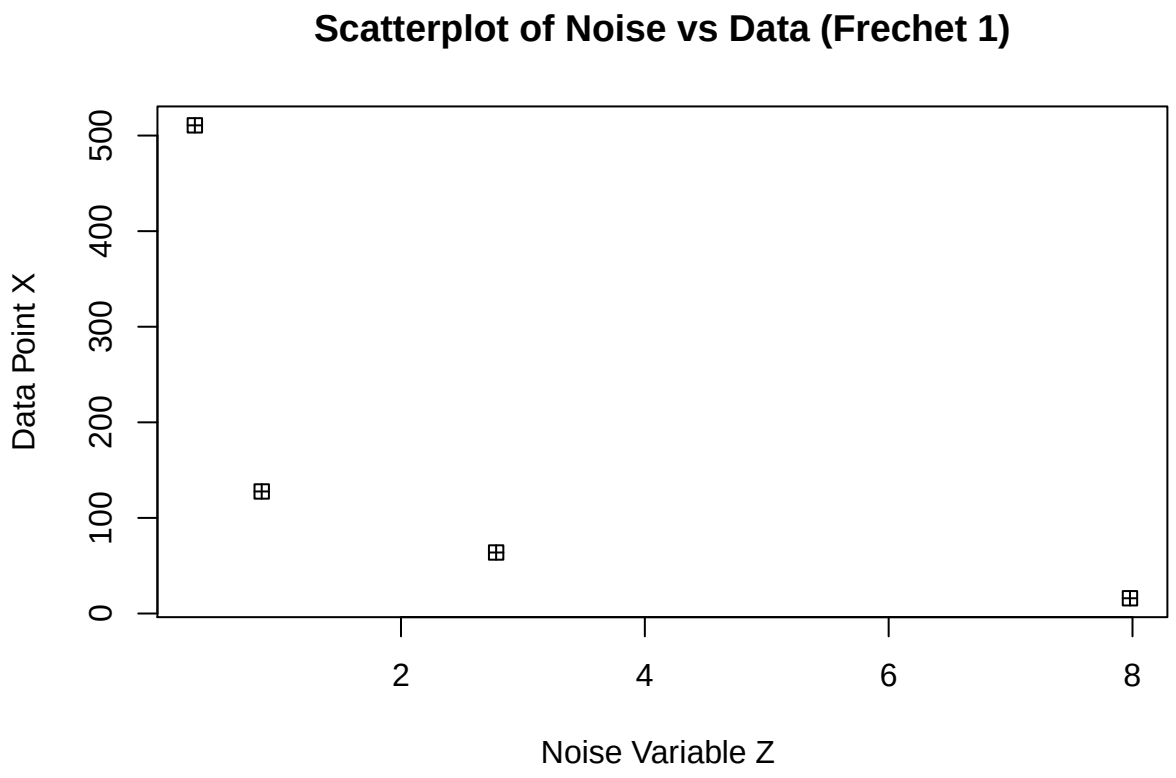
```
plot(exponential.1$noise, exponential.1$data, main="Scatterplot of Noise vs Data (Exponential 1)", xlab=
```




```
plot(exponential.2$noise, exponential.2$data, main="Scatterplot of Noise vs Data (Exponential 2)", xlab=
```




```
plot(frechet.1$noise, frechet.1$data, main="Scatterplot of Noise vs Data (Frechet 1)", xlab="Noise Varia
```




```
plot(frechet.2$noise, frechet.2$data, main="Scatterplot of Noise vs Data (Frechet 2)", xlab="Noise Varia
```