midterm

November 22, 2018

0.1 Load data from the website

In [1]:
```
import pandas as pd
mydateparser = lambda x: pd.datetime.strptime(x, "%Y-%m-%dT%H:%M:%S+08:00")
df = pd.read_csv('http://farmer.iyard.org/cwb/C0D660.txt', sep=',', header=None,
                 encoding='big5', usecols=[0,1,4,7], names=['lat','lon','datetime','wi
                 parse_dates=['datetime'], date_parser=mydateparser)
df.head()
```

Out[1]:

	lat	lon	datetime	windspeed
0	24.800436	120.978742	2018-05-01 00:00:00	0.4
1	24.800436	120.978742	2018-05-01 01:00:00	0.7
2	24.800436	120.978742	2018-05-01 02:00:00	0.3
3	24.800436	120.978742	2018-05-01 03:00:00	1.1
4	24.800436	120.978742	2018-05-01 04:00:00	0.7

In [2]:
```
df.index = df['datetime']
del df['datetime']
df.head()
```

Out[2]:

datetime	lat	lon	windspeed
2018-05-01 00:00:00	24.800436	120.978742	0.4
2018-05-01 01:00:00	24.800436	120.978742	0.7
2018-05-01 02:00:00	24.800436	120.978742	0.3
2018-05-01 03:00:00	24.800436	120.978742	1.1
2018-05-01 04:00:00	24.800436	120.978742	0.7

In [3]:
```
import numpy as np
x = df['windspeed'].values
n = df['windspeed'].size
```

0.2 MoM

In [4]:
```
from scipy.optimize import root
from scipy.special import gamma
def momeq(theta, data):
    k = theta[0]
    lam = theta[1]
```

```
        return [lam*gamma(1+1/k)-data.mean(),
                lam*lam*(gamma(1+2/k)-(gamma(1+1/k))**2)-data.var()]

momcond = lambda theta: momeq(theta, x)
sol = root(momcond, [1,1])
sol
```

```
Out[4]:     fjac: array([[-0.08153496, -0.99667048],
               [ 0.99667048, -0.08153496]])
            fun: array([-1.24344979e-14, -3.99680289e-15])
        message: 'The solution converged.'
           nfev: 11
            qtf: array([-6.95769620e-12,  2.34921836e-11])
              r: array([ 2.8076421 , -1.78398658,  0.80765522])
         status: 1
        success: True
              x: array([1.61426914, 2.62979162])
```

```
In [5]: theta_mom = sol.x
        theta_mom
```

```
Out[5]: array([1.61426914, 2.62979162])
```

0.3 MLE by R

```
In [6]: epsilon = 1e-3
        y = x + epsilon
```

```
In [7]: from rpy2.robjects import r, numpy2ri, pandas2ri
        numpy2ri.activate()
        from rpy2.robjects.packages import importr
        stats = importr('stats')
        def objfun(theta, data):
            loglik = sum(stats.dweibull(data, shape=theta[0], scale=theta[1], log=True))
            return -loglik
```

```
In [8]: negativeweibull_lik = lambda theta: objfun(theta,y)
```

```
In [9]: from scipy.optimize import minimize
        res = minimize(negativeweibull_lik, [1,1], method='Powell')
        res
```

```
Out[9]:    direc: array([[0.        , 1.        ],
               [0.08589708, 0.0401073 ]])
              fun: array(8475.27632071)
          message: 'Optimization terminated successfully.'
             nfev: 77
              nit: 3
           status: 0
          success: True
                x: array([1.46809023, 2.56947397])
```

```
In [10]: theta_mle = res.x
         theta_mle

Out[10]: array([1.46809023, 2.56947397])
```

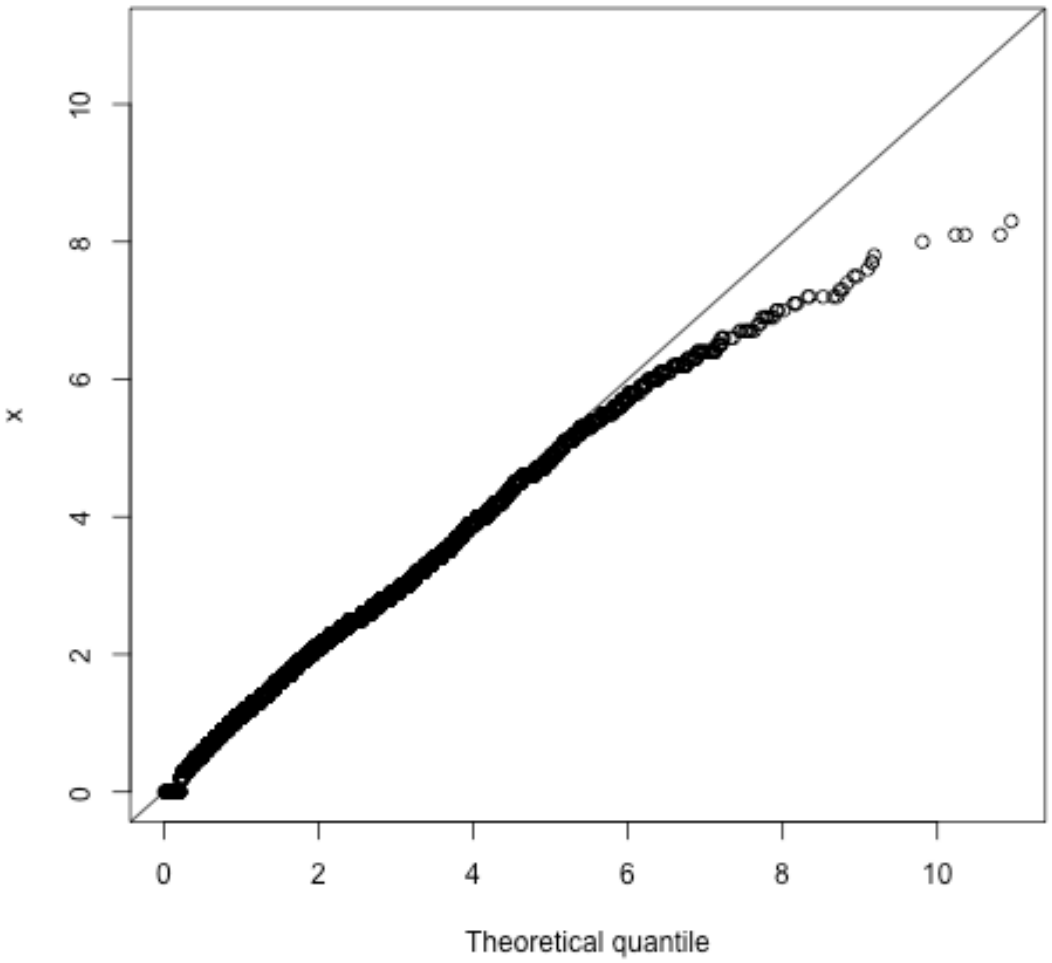
0.4 Kolmogorov-Smirnov test by R

```
In [11]: ks = stats.ks_test(x, 'pweibull', shape=theta_mle[0], scale=theta_mle[1])
         ks

Out[11]: R object with classes: ('htest',) mapped to:
         <ListVector - Python:0x1131d7f48 / R:0x7fbc0c99f648>
         [FloatVe..., FloatVe..., StrVector, StrVector, StrVector]
           statistic: <class 'rpy2.robjects.vectors.FloatVector'>
           R object with classes: ('numeric',) mapped to:
         <FloatVector - Python:0x1021bbcd88 / R:0x7fbc0cc118b8>
         [0.053161]
           p.value: <class 'rpy2.robjects.vectors.FloatVector'>
           R object with classes: ('numeric',) mapped to:
         <FloatVector - Python:0x1021cc7608 / R:0x7fbc0cc11340>
         [0.000000]
           alternative: <class 'rpy2.robjects.vectors.StrVector'>
           R object with classes: ('character',) mapped to:
         <StrVector - Python:0x1021bbcc88 / R:0x7fbc09c8a9f0>
         ['two-sided']
           method: <class 'rpy2.robjects.vectors.StrVector'>
           R object with classes: ('character',) mapped to:
         <StrVector - Python:0x1021cbcd48 / R:0x7fbc09c8aec0>
         ['One-sample Kolmogorov-Smirnov test']
           data.name: <class 'rpy2.robjects.vectors.StrVector'>
           R object with classes: ('character',) mapped to:
         <StrVector - Python:0x1021cbcf88 / R:0x7fbc0b8d6400>
         ['structur..., '4, 3.6, ..., '0.9, 1.1..., '4.2, 4.2..., ..., '0.4, 0.7..., '1, 1.6, ..

In [12]: pvalue = ks[1]
         np.asarray(pvalue)

Out[12]: array([3.13693516e-12])
```

0.5 qqplot with R

```
In [13]: import warnings
         warnings.filterwarnings('ignore')
         # Load in the r magic

         import rpy2.ipython
         %reload_ext rpy2.ipython
```

```
In [14]: %%R -i x,theta_mle,n
         w <- rweibull(n,shape=theta_mle[1], scale=theta_mle[2])
         lower <- min(c(x,w))
         upper <- max(c(x,w))
         qqplot(w,x,xlim=c(lower,upper),ylim=c(lower,upper),xlab='Theoretical quantile')
         abline(0,1)
```



0.6 Bootstrap CI