Assignment 8

In [23]:

```
%matplotlib inline

import numpy as np

import pystan
import stan_utility

import matplotlib
import matplotlib.pyplot as plt

from IPython.display import display, Markdown, Latex

font = {'size': 16}

matplotlib.rc('font', **font)

print('numpy', np.__version__)
print('pystan', pystan.__version__)
```

```
numpy 1.14.1
pystan 2.17.1.0
```

Decision analysis for the factory data

Some configuration values and utility functions:

In [5]:

```
product_cost        = 100
product_price       = 200
quality_threshold   = 85
```

In [7]:

```
def utility(x):
    if x < quality_threshold:
        return -product_cost
    else:
        return product_price - product_cost
```

We load the data

In [3]:

```
data_factory = np.loadtxt('../data/factory.txt')
data_factory.shape
```

Out[3]:

```
(5, 6)
```

Model

We will use the hierarchical model since it performed the best in the previous assignments.

In [57]:

```
stan_code_hierarchical = """
data {
  int<lower=0> N;              // number of data points
  int<lower=0> K;              // number of groups
  int<lower=1,upper=K> x[N]; // group indicator
  vector[N] y;                 // target
}
parameters {
  real mu_prior;               // shared prior mean
  real<lower=0> sigma_prior; // shared prior std
  vector[K] mu;                // group means
  real<lower=0> sigma;         // shared std
}
model {
  for (k in 1:K) {
    mu[k] ~ normal(mu_prior, sigma_prior);
  }
  y ~ normal(mu[x], sigma);
}
generated quantities {
  real mu_new;
  mu_new = normal_rng(mu_prior, sigma_prior);
}
"""

model_hierarchical = pystan.StanModel(model_code = stan_code_hierarchical)
```

```
INFO:pystan:COMPILING THE C++ CODE FOR MODEL anon_model_4613248e8d92bdc69a
f854e6ce557634 NOW.
```

In [58]:

```
x = np.tile(np.arange(1, data_factory.shape[1] + 1), data_factory.shape[0])
y = data_factory.flatten()
N = len(x)
K = np.max(x)

fit_hierarchical = model_hierarchical.sampling(
    data = {
        'N': N,
        'K': K,
        'x': x,
        'y': y
    })
```

```
c:\users\ncp\appdata\local\continuum\anaconda3\envs\stan_env\lib\site-pack
ages\pystan\misc.py:399: FutureWarning: Conversion of the second argument
of issubdtype from `float` to `np.floating` is deprecated. In future, it w
ill be treated as `np.float64 == np.dtype(float).type`.
  elif np.issubdtype(np.asarray(v).dtype, float):
```

We check the fitted parameters and the performance of the fit, i.e. the treedepth, E-BFMI, and divergences:

In [59]:

```
stan_utility.check_treedepth(fit_hierarchical)
stan_utility.check_energy(fit_hierarchical)
stan_utility.check_div(fit_hierarchical)
```

```
0 of 4000 iterations saturated the maximum tree depth of 10 (0.0%)
12.0 of 4000 iterations ended with a divergence (0.3%)
Try running with larger adapt_delta to remove the divergences
```

We see some iterations ended with a divergence, but since it is such small fraction, we accept the performance.

In [60]:

```
params = fit_hierarchical.extract()
```

We calculate the expected utility for each of the six machines:

In [61]:

```
vec_utility = np.vectorize(utility)
```

In [62]:

```
mu = vec_utility(params['mu']).mean(axis = 0)
for i in range(6):
    display(Markdown(r'$\bar{\mathrm{u}}_' + str(i + 1) + r' = ' + '{:.2f}'.format(mu[i
]) + r'$'))
```

$\bar{\mathrm{u}}_1 = -56.15$

$\bar{\mathrm{u}}_2 = 99.25$

$\bar{\mathrm{u}}_3 = 49.60$

$\bar{\mathrm{u}}_4 = 99.90$

$\bar{\mathrm{u}}_5 = 66.35$

$\bar{\mathrm{u}}_6 = 33.15$

We rank the machines based on the expected utilities

In [63]:

```
mu = vec_utility(params['mu']).mean(axis = 0)
rank =  np.argsort(-mu)

for i in range(6):
    display(Markdown(
        'Rank ' + str(i + 1) + r': Machine ' + str(rank[i] + 1) +
        r' ($\bar{\mathrm{u}}_' + str(rank[i] + 1) + r' = ' + '{:.2f}'.format(mu[rank[i
]]) + r')$'))
```

Rank 1: Machine 4 ($\bar{\mathrm{u}}_4 = 99.90)$

Rank 2: Machine 2 ($\bar{\mathrm{u}}_2 = 99.25)$

Rank 3: Machine 5 ($\bar{\mathrm{u}}_5 = 66.35)$

Rank 4: Machine 3 ($\bar{\mathrm{u}}_3 = 49.60)$

Rank 5: Machine 6 ($\bar{\mathrm{u}}_6 = 33.15)$

Rank 6: Machine 1 ($\bar{\mathrm{u}}_1 = -56.15)$

We see that all machines execpt machine 1 makes a profit. We should advise to stop using machine 1 instantly. The other 5 machines (2-6) all makes profit, altough the amount of profit varies much. The best profitable machines are 4 and 2.

For the 7th machine, we predict the followin utility:

In [64]:

```
mu7 = vec_utility(params['mu_new']).mean()
display(Markdown(r'$\bar{\mathrm{u}}_7 = ' + '{:.2f}'.format(mu7) + r'$'))
```

$\bar{\mathrm{u}}_7 = 40.55$

Since the predicted utility for the 7th machine is positive, we advise the owners to buy a 7th machine.