ECON 504 HW3: Commentary

Peter Donahue

Fall 2024

Running The Code

Whenever it seemed like a task was about to require some serious re-tooling to the existing structure of the code, I split it off into a different module. Unfortunately, this means I have 3 modules that duplicate (or almost duplicate) quite a lot of code across them. Here is a breakdown of the three:

1. task1.jl
2. task2.jl
3. task3.jl (actually covers everything from task 3 - task 7)

Here the code to ensure you have all the packages installed required by any of the three modules:

```
import Pkg; Pkg.add(["JSON", "Distributed", "Profile", "Distributions", "Random", "JuMP",
↪  "Ipopt", "MadNLP", "LinearAlgebra", "DistributedSparseGrids", "FastGaussQuadrature",
↪  "SparseGrids", "Printf", "PrettyTables", "DataFrames"])
```

To run the three modules:

1. Windows: run the batch file "run.bat"
2. Mac: run the equivalent bash file "run.sh"

Step-By-Step Responses

Task 1

- Log Marginal Cost Function

  The log marginal cost function is defined as:

$$\log c_{j,t} = 0.5 + 0.1 w_{j,1,t} + 0.1 w_{j,2,t} + \omega_{j,t} \tag{1}$$

  Corresponding Code:

```
    log_mc = 0.5 + 0.1 * w_t[j, 1] + 0.1 * w_t[j, 2] + omega_t[j]
    mc_t[j] = exp(log_mc)
```

- Utility Function for Inside Good

  The utility function for the inside good j in market t for consumer i is:

$$U_{i,j,t} = \beta_{0,i,t} + \beta_{1,i,t} x_{j,1,t} + \beta_{2,i,t} x_{j,2,t} - |\beta_{3,i,t}| p_{j,t} + \xi_{j,t} + \epsilon_{i,j,t} \tag{2}$$

  Corresponding Code:

```
@NLexpression(model, V[s=1:_S, k=1:_J],
    beta0_s[s] + beta1_s[s] * x_k1[k] + beta2_s[s] * x_k2[k]
    - beta3_s[s] * p[k] + ksi_k[k])
```

- Choice Probabilities

  Given the Type 1 extreme value distribution for $\epsilon_{i,j,t}$, the choice probabilities are:

$$P_{i,j,t} = \frac{\exp(U_{i,j,t})}{1 + \sum_{k=1}^{J} \exp(U_{i,k,t})} \tag{3}$$

  Corresponding Code:

```
@NLexpression(model, exp_V[s=1:_S, k=1:_J], exp(V[s, k]))
@NLexpression(model, denom[s=1:_S], 1 + sum(exp_V[s, k] for k in 1:_J))
@NLexpression(model, P[s=1:_S, k=1:_J], exp_V[s, k] / denom[s])
```

- Market Share (Simulated)

  The aggregate market share for product j is approximated by simulation:

$$s_{j,t} = \frac{1}{S} \sum_{s=1}^{S} P_{s,j,t} \tag{4}$$

  Corresponding Code:

```
@NLexpression(model, s_p[j=1:_J],
    (1 / _S) * sum(P[s, j] for s in 1:_S))
```

- Derivative of Market Share w.r.t. Price

  The derivative of the market share with respect to price is:

$$\frac{\partial s_{j,t}}{\partial p_{j,t}} = \frac{1}{S} \sum_{s=1}^{S} P_{s,j,t}(1 - P_{s,j,t})(-\beta_{3,s}) \tag{5}$$

  Corresponding Code:

```
@NLexpression(model, ds_dp[j=1:_J],
    (1 / _S) * sum(P[s, j] * (1 - P[s, j]) * (-beta3_s[s])
    for s in 1:_S))
```

- First Order Condition

  The first order condition for price optimization is:

$$s_{j,t} + (p_{j,t} - c_{j,t}) \frac{\partial s_{j,t}}{\partial p_{j,t}} = 0 \tag{6}$$

  Corresponding Code:

```
@NLconstraint(model, [j=1:_J],
    s_p[j] + (p[j] - mc_k[j]) * ds_dp[j] == 0)
```

- Distribution of Product Characteristics and Cost Shifters

  The vector of product characteristics and marginal cost shifters is distributed as:

$$\begin{pmatrix} x_{j,1,t} \\ x_{j,2,t} \\ w_{j,1,t} \\ w_{j,2,t} \end{pmatrix} \sim \mathcal{N}\left(\mathbf{0}, \Sigma_{XW}\right) \tag{7}$$

  with covariance matrix:

$$\Sigma_{XW} = \begin{pmatrix} 0.25 & 0.05 & 0.05 & 0.05 \\ 0.05 & 0.25 & 0.05 & 0.05 \\ 0.05 & 0.05 & 0.1 & 0.05 \\ 0.05 & 0.05 & 0.05 & 0.1 \end{pmatrix} \tag{8}$$

  Corresponding Code:

```
const _SIGMA_XW = [0.25 0.05 0.05 0.05;
                   0.05 0.25 0.05 0.05;
                   0.05 0.05 0.1  0.05;
                   0.05 0.05 0.05 0.1 ]
const _MEAN_XW = zeros(4)

xw = rand(MvNormal(_MEAN_XW, _SIGMA_XW))
x_t[j, :] = xw[1:2]
w_t[j, :] = xw[3:4]
```

- Distribution of Unmeasured Demand and Cost Shifters

  The unmeasured demand and cost shifters are distributed as:

$$\begin{pmatrix} \xi_{j,t} \\ \omega_{j,t} \end{pmatrix} \sim \mathcal{N}\left(\mathbf{0}, \Sigma_{\xi\omega}\right) \tag{9}$$

  with covariance matrix:

$$\Sigma_{\xi\omega} = \begin{pmatrix} 0.2 & 0.1 \\ 0.1 & 0.2 \end{pmatrix} \tag{10}$$

  Corresponding Code:

```
const _SIGMA_KSI_OMEGA = [0.2 0.1; 0.1 0.2]
const _MEAN_KSI_OMEGA = zeros(2)

ksi_omega = rand(MvNormal(_MEAN_KSI_OMEGA, _SIGMA_KSI_OMEGA))
ksi_t[j] = ksi_omega[1]
omega_t[j] = ksi_omega[2]
```

- Distribution of Random Coefficients

  Each random coefficient is independently normally distributed:

$$\beta_{k,i,t} \sim \mathcal{N}\left(\mu_k, \sigma_k^2\right), \quad k = 0, 1, 2, 3 \tag{11}$$

  with means and standard deviations:

$$\mu = \begin{pmatrix} 1.5 \\ 1.0 \\ 1.0 \\ 1.0 \end{pmatrix}, \qquad \sigma = \begin{pmatrix} 1.0 \\ 1.0 \\ 1.0 \\ 0.2 \end{pmatrix}$$

  Corresponding Code:

```
const _BETA_MEANS = [1.5, 1.0, 1.0, 1.0]
const _BETA_SDS = [1.0, 1.0, 1.0, 0.2]

beta_i_t = zeros(_S, 4)
for s in 1:_S
    beta_i_t[s, :] = [rand(Normal(_BETA_MEANS[k], _BETA_SDS[k]))
                      for k in 1:4]
end
```

- Note on the Absolute Value of Price Coefficient

  The absolute value on the price coefficient makes its sign unidentified:

$$\beta_{3,i,t} = |\beta_{3,i,t}| \tag{12}$$

  Corresponding Code:

```
beta3_s = abs.(beta_i_t[:, 4])
```

Results

Task 1 (Subset)

Market	Product	Price	Marginal Cost	Market Share	Characteristic 1	Characteristic 2
1	1	2.54	1.41	0.08	-0.18	0.5
1	2	3.21	2.02	0.1	1.03	0.36
1	4	2.31	1.24	0.02	0.08	-0.43
1	8	6.9	5.68	0.0	-0.39	-0.4
1	9	2.92	1.81	0.06	0.16	0.65
1	10	2.9	1.55	0.21	0.74	0.58
2	7	2.13	0.94	0.04	-1.34	0.19
2	9	2.33	1.21	0.04	-0.63	-0.61
3	1	3.53	1.94	0.19	1.11	0.69
9	1	3.47	1.88	0.24	0.87	0.83

Above is a subset of the full results table – the full table is in the appendix. The main takeaways are relatively straightforward:

- Firms are profit maximizing, so price is always above marginal cost. Even if this means effectively "shutting down production", which in this case means having a market share of 0.
- For the most part, products with a higher marginal cost and thus price have a lower market share, for example product 8 in market 1 has by far the highest price point and is the only one with 0 market share.
- Products that do well in the two non-price characteristics can charge a higher margin without sacrificing too much market share – see product 1 in market 3 which has the highest profit margin (1.59), highest price (3.53) and highest market share (0.19) in market 3.

Task 2

Task 2

Method	Draws/Nodes	Time (s)	Avg Price Deviation
Monte Carlo S=1,000,000	10000	1010.76	N/A
Monte Carlo S=100	100	7.22	0.02666
Monte Carlo S=1,000	1000	60.48	0.00358
Gauss-Hermite N=3	3	6.4	0.00218
Gauss-Hermite N=5	5	38.3	0.00180
Sparse Grid Accuracy=5	5	25.0	0.00215

The first thing I notice is that the Monte Carlo methods performed the worst, and the better performing Monte Carlo run (S=1,000) took almost twice as long as the next slowest method. The Gauss-Hermite N=5 method was the more accurate, followed by the Sparse Grid Accuracy=5 and Gauss-Hermite N=3 which performed equally well – in terms of average price deviations from the "data." In terms of runtime, the similarly precise Gauss-Hermite N=3 was almost 5 times faster than the Sparse Grid Accuracy=5 method.

Task 5

Excluded Instruments

- Equation 12 from Gandhi and Houde (2020) defines the set of differentiation instruments:

$$A_j\left(\boldsymbol{x}_t, \boldsymbol{w}_t\right) = \begin{cases} \boldsymbol{w}_{jt} & \text{Price IVs} \\ \sum_{j' \neq j} \left(d^{\hat{p}}_{jt,j'}\right)^2 & \text{Isolation of } j \text{ in price} \\ \sum_{j' \neq j} \left(d^{k}_{jt,j'}\right)^2 & \text{Isolation of } j \text{ in } x_k \\ \sum_{j' \neq j} d^{k}_{jt,j'} \times d^{l}_{jt,j'} & \text{Interaction between } k \text{ and } l \end{cases} \tag{13}$$

  where $d^k_{jt,j'} = x_{jkt} - x_{j'kt}$ is the difference in characteristic k between products j and j' in market t.

- Instrument 1: Intercept

  Mathematical Definition:

$$Z_{jt,1} = 1$$

  Corresponding Code:

```
Z[idx, 1] = 1
```

- Instruments 2 and 3: Marginal Cost Shifters for Own Product

  Mathematical Definitions:

$$Z_{jt,2} = w_{j1t}$$
$$Z_{jt,3} = w_{j2t}$$

  Corresponding Code:

```
Z[idx, 2] = w1[t, j]
Z[idx, 3] = w2[t, j]
```

- Instruments 4 and 5: Squares of Marginal Cost Shifters

  Mathematical Definitions:

$$Z_{jt,4} = w^2_{j1t}$$
$$Z_{jt,5} = w^2_{j2t}$$

  Corresponding Code:

```
    Z[idx, 4] = w1[t, j]^2
    Z[idx, 5] = w2[t, j]^2
```

- Instruments 6 and 7: Sum of Characteristic Differences

  Mathematical Definitions:

  For each non-price characteristic $k = 1, 2$,

$$Z_{jt,6} = \sum_{j' \neq j} (x_{jkt} - x_{j'kt})$$
$$Z_{jt,7} = \sum_{j' \neq j} (x_{jkt} - x_{j'kt}) \quad (\text{for } k = 2)$$

  Corresponding Code:

```
    other_indices = [k for k in 1:J if k != j]

    sum_d_x1 = sum(x1[t, j] .- x1[t, other_indices])
    Z[idx, 6] = sum_d_x1

    sum_d_x2 = sum(x2[t, j] .- x2[t, other_indices])
    Z[idx, 7] = sum_d_x2
```

- Instruments 8 and 9: Sum of Squared Characteristic Differences

  Mathematical Definitions:

$$Z_{jt,8} = \sum_{j' \neq j} (x_{jkt} - x_{j'kt})^2$$
$$Z_{jt,9} = \sum_{j' \neq j} (x_{jkt} - x_{j'kt})^2 \quad (\text{for } k = 2)$$

  Corresponding Code:

```
    sum_d_x1_sq = sum((x1[t, j] .- x1[t, other_indices]).^2)
    Z[idx, 8] = sum_d_x1_sq

    sum_d_x2_sq = sum((x2[t, j] .- x2[t, other_indices]).^2)
    Z[idx, 9] = sum_d_x2_sq
```

- Instrument 10: Interaction Term

  Mathematical Definition:

$$Z_{jt,10} = \sum_{j' \neq j} (x_{j1t} - x_{j'1t}) \times (x_{j2t} - x_{j'2t})$$

  Corresponding Code:

```
    interaction_term = sum((x1[t, j] .- x1[t, other_indices]) .*
                           (x2[t, j] .- x2[t, other_indices]))
    Z[idx, 10] = interaction_term
```

Price Endogeneity and Instrument Validity

- In our model, prices are determined simultaneously with quantities (market shares), and they may be correlated with unobserved factors affecting demand (e.g., unmeasured product quality). This correlation leads to endogeneity.
- The instruments are functions of observed product characteristics and cost shifters, which are assumed to be exogenous. They affect prices through their influence on marginal costs and the competitive environment but are uncorrelated with the unmeasured demand shocks.

Task 6

I will now detail the estimation procedure. The estimation is performed using the Method of Simulated Moments (MSM) within the Mathematical Programming with Equilibrium Constraints (MPEC) framework. The implementation leverages JuMP and the IPOPT solver for optimization. A two-step Generalized Method of Moments (GMM) estimator with an optimal weighting matrix is employed.

1. Model Specification

   We consider a random coefficients logit model where the utility of consumer i for product j in market t is given by:

   $$U_{i,j,t} = \beta_{0,i} + \beta_{1,i} x_{j,1,t} + \beta_{2,i} x_{j,2,t} - |\beta_{3,i}| p_{j,t} + \epsilon_{i,j,t} \tag{14}$$

   where:

   - $x_{j,1,t}$ and $x_{j,2,t}$ are observed product characteristics.
   - $p_{j,t}$ is the price of product j in market t.
   - $\beta_{k,i}$ are individual-specific random coefficients defined as:

     $$\beta_{k,i} = \bar{\beta}_k + \sigma_k v_{k,i}, \quad v_{k,i} \sim \mathcal{N}(0,1)$$

   - $\epsilon_{i,j,t}$ is an i.i.d. Type I Extreme Value error term.

   Corresponding Code:

```
@NLexpression(model, V[i=1:N_data, n=1:N_nodes],
    (beta_means[1] + sqrt(2)*sigma[1]*NODES[n,1])
    + (beta_means[2] + sqrt(2)*sigma[2]*NODES[n,2]) * x1[i]
    + (beta_means[3] + sqrt(2)*sigma[3]*NODES[n,3]) * x2[i]
    - abs(beta_means[4] + sqrt(2)*sigma[4]*NODES[n,4]) * prices_vec[i]
)
```

2. Choice Probabilities and Market Shares

   Given the utility specification and the distribution of the error term, the individual choice probabilities are:

   $$P_{i,j,t} = \frac{\exp(U_{i,j,t})}{1 + \sum_{k=1}^{J} \exp(U_{i,k,t})} \tag{15}$$

   We approximate the market shares by integrating over the distribution of random coefficients using sparse grid quadrature with nodes v_n and weights w_n:

   $$\hat{s}_{j,t} = \frac{\sum_{n=1}^{N} w_n P_{j,t}(v_n)}{\sum_{n=1}^{N} w_n} \tag{16}$$

   Corresponding Code:

```
@NLexpression(model, P[i=1:N_data, n=1:N_nodes],
    exp(V[i, n]) / (1 + exp(V[i, n]))
)

@NLexpression(model, s_hat[i=1:N_data],
    sum(WEIGHTS[n] * P[i, n] for n = 1:N_nodes) / total_weight
)
```

3. Moment Conditions

   We construct moment conditions using instruments $Z_{i,j}$:

$$m_{i,j}(\theta) = (s_{j,t} - \hat{s}_{j,t}(\theta))Z_{i,j} \tag{17}$$

   where $s_{j,t}$ are the observed market shares and $\hat{s}_{j,t}(\theta)$ are the predicted market shares given parameters $\theta = (\bar{\beta}, \sigma)$.

   Corresponding Code:

```
@NLexpression(model, moments[i=1:N_data, j=1:num_instruments],
    (shares_vec[i] - s_hat[i]) * Z[i, j]
)
```

4. GMM Objective Function

   We define the GMM objective function as:

$$Q(\theta) = m(\theta)' W m(\theta) \tag{18}$$

   where W is the weighting matrix, initially set to the identity matrix.

   Corresponding Code:

```
W = I(num_instruments)

@NLobjective(model, Min, sum(
    moments[i, j] * W[j, k] * moments[i, k]
    for i in 1:N_data, j in 1:num_instruments, k in 1:num_instruments
))
```

5. First-Stage Estimation

   We minimize the GMM objective function to obtain initial estimates of θ:

$$\hat{\theta}_1 = \arg\min_{\theta} Q(\theta) \tag{19}$$

   This is implemented using the IPOPT solver via JuMP.

   Corresponding Code:

```
optimize!(model)
```

6. Compute Optimal Weighting Matrix

   Using the residuals from the first-stage estimation, we compute the optimal weighting matrix:

$$W_{\text{opt}} = \left(\frac{1}{N} \sum_{i=1}^{N} m_i(\hat{\theta}_1) m_i(\hat{\theta}_1)' \right)^{-1} \tag{20}$$

Corresponding Code:

```
residuals = shares_vec .- [value(s_hat[i]) for i in 1:N_data]

moments_matrix = residuals .* Z

W_opt = inv((moments_matrix' * moments_matrix) / (T * J))
```

7. Second-Stage Estimation

   We re-estimate the model using the optimal weighting matrix to obtain the efficient GMM estimates:

$$\hat{\theta}_2 = \arg\min_{\theta} m(\theta)' W_{\text{opt}} m(\theta) \tag{21}$$

   Corresponding Code:

```
@NLobjective(model, Min, sum(
    moments[i, j] * W_opt[j, k] * moments[i, k]
    for i in 1:N_data, j in 1:num_instruments, k in 1:num_instruments
))
optimize!(model)
```

8. Parameter Estimates

   The final parameter estimates are obtained from the second-stage optimization:

$$\hat{\theta} = (\hat{\bar{\beta}}, \hat{\sigma})$$

   Corresponding Code:

```
beta_means_est = value.(beta_means)
sigma_est = value.(sigma)
final_estimates = (beta_means_est = beta_means_est, sigma_est = sigma_est)
```

9. Additional Details

   Variables and Constraints in the Optimization Model We define the variables for the parameters to be estimated:

   - `beta_means`: $\bar{\beta}_k$, for $k = 1, \ldots, 4$
   - `sigma`: $\sigma_k \geq 0$, for $k = 1, \ldots, 4$

   Corresponding Code:

```
@variable(model, beta_means[i=1:4], start=beta_init[i])
@variable(model, sigma[i=1:4] >= 0, start=abs(beta_init[4 + i]))
```

Integration using Sparse Grid Quadrature We approximate the integrals over the distribution of random coefficients using a Smolyak sparse grid quadrature method with level 5 accuracy.

Corresponding Code:

```
const NODES, WEIGHTS = smolyak_sparse_grid(5, 4)
const N_nodes = size(NODES, 1)
const total_weight = sum(WEIGHTS)

function smolyak_sparse_grid(level, dimension)
    nodes, weights = sparsegrid(dimension, level, FastGaussQuadrature.gausshermite)
    return hcat(nodes...)', weights
end
```

Loop Over Starting Values To ensure global optimality and avoid local minima, we loop over multiple starting values for the parameters.

Corresponding Code:

```
for beta_init in starting_values_list
    # Estimation steps
end
```

Moment Conditions and Weighting Matrix In the first stage, the weighting matrix W is set to the identity matrix. After obtaining initial estimates, we compute the residuals and update the weighting matrix to the optimal one based on the empirical variance of the moments.

Corresponding Code for First Stage:

```
W = I(num_instruments)

# Define the GMM objective function
@NLobjective(model, Min, sum(
    moments[i, j] * W[j, k] * moments[i, k]
    for i in 1:N_data, j in 1:num_instruments, k in 1:num_instruments
))
```

Corresponding Code for Second Stage:

```
# Update the weighting matrix W_opt
W_opt = inv((moments_matrix' * moments_matrix) / (T * J))

# Redefine the GMM objective function with W_opt
@NLobjective(model, Min, sum(
    moments[i, j] * W_opt[j, k] * moments[i, k]
    for i in 1:N_data, j in 1:num_instruments, k in 1:num_instruments
))
```

Parallelization and Multiple Simulations The estimation is performed within a Monte Carlo study, evaluating several combinations of J and T over multiple simulated datasets. Parallelization is utilized to speed up computations.

Corresponding Code Snippet:

```
addprocs(3)

@sync begin
    for (J_T) in JT_combinations
        @async begin
            # Simulation and estimation code
        end
    end
end
```

The estimation procedure combines the MPEC approach with a two-step GMM estimator to efficiently estimate the means and variances of the random coefficients in the model. The use of sparse grid quadrature allows for accurate approximation of integrals over the random coefficients, and the implementation in JuMP with IPOPT leverages automatic differentiation and efficient optimization algorithms.

Appendix

Task 1

Market	Product	Price	Marginal Cost	Market Share	Characteristic 1	Characteristic 2
Int64	Int64	Float64	Float64	Float64	Float64	Float64
1	1	2.54	1.41	0.08	-0.18	0.5
1	2	3.21	2.02	0.1	1.03	0.36
1	3	2.09	1.01	0.05	0.01	0.05
1	4	2.31	1.24	0.02	0.08	-0.43
1	5	2.26	1.06	0.13	0.14	0.8
1	6	3.2	2.08	0.06	-0.1	0.36
1	7	2.74	1.58	0.1	0.86	0.25
1	8	6.9	5.68	0.0	-0.39	-0.4
1	9	2.92	1.81	0.06	0.16	0.65
1	10	2.9	1.55	0.21	0.74	0.58
2	1	2.5	1.36	0.1	0.58	0.04
2	2	2.69	1.64	0.03	-0.32	0.13
2	3	2.5	1.43	0.03	-0.51	-0.6
2	4	2.78	1.59	0.12	0.53	0.36
2	5	2.37	1.29	0.07	0.38	0.12
2	6	2.82	1.68	0.09	0.21	0.72
2	7	2.13	0.94	0.04	-1.34	0.19
2	8	2.33	1.26	0.06	0.1	-0.23
2	9	2.33	1.21	0.04	-0.63	-0.61
2	10	2.18	0.86	0.2	-0.07	0.56
3	1	3.53	1.94	0.19	1.11	0.69
3	2	2.29	1.15	0.09	-0.3	0.41
3	3	2.6	1.5	0.05	-0.18	0.11
3	4	2.79	1.66	0.05	-0.75	-0.26
3	5	3.2	2.11	0.02	-0.29	-0.31
3	6	3.14	2.0	0.08	0.28	0.37
3	7	2.43	1.2	0.06	0.14	-0.85
3	8	2.16	1.03	0.07	-0.56	0.24
3	9	2.92	1.82	0.02	-0.1	-0.63
3	10	3.04	1.81	0.14	0.17	0.32
4	1	3.63	2.57	0.03	0.58	-0.11
4	2	2.91	1.88	0.03	0.16	-0.53
4	3	3.38	2.2	0.13	0.22	0.53
4	4	3.06	1.92	0.11	0.7	0.48
4	5	4.05	2.99	0.03	-0.38	0.52
4	6	3.29	1.88	0.21	0.63	0.92
4	7	2.55	1.42	0.08	0.66	-0.35
4	8	4.18	3.16	0.01	-0.56	-0.2
4	9	2.3	1.15	0.11	0.63	-0.14
4	10	2.55	1.52	0.04	-0.01	0.09
5	1	2.26	0.9	0.24	0.53	0.31
5	2	2.66	1.5	0.1	0.19	0.62
5	3	2.3	1.2	0.03	-0.7	-0.12
5	4	3.16	2.06	0.01	-0.09	-0.7
5	5	3.37	2.24	0.02	-0.33	-0.5
5	6	2.96	1.79	0.1	0.31	0.68
5	7	2.62	1.43	0.11	0.36	-0.07
5	8	2.92	1.8	0.04	0.37	-0.42
5	9	2.45	1.3	0.1	0.64	0.08
5	10	3.24	2.13	0.04	0.37	0.17
6	1	2.59	1.44	0.1	0.69	0.12
6	2	2.04	1.01	0.03	-0.48	-0.32
6	3	3.12	2.09	0.02	-0.34	-0.0
6	4	2.55	1.36	0.13	0.61	0.52
6	5	2.97	1.88	0.04	-0.71	-0.1
6	6	2.57	1.47	0.08	0.15	0.05
6	7	2.11	1.05	0.04	-0.4	-0.59
6	8	2.87	1.8	0.05	0.39	0.26
6	9	3.69	2.6	0.03	-0.37	0.86
6	10	2.31	0.65	0.24	0.32	1.46
7	1	2.95	1.57	0.17	0.81	-0.0
7	2	2.78	1.72	0.04	-0.02	-0.33
7	3	4.73	3.58	0.02	0.63	0.41
7	4	2.32	1.27	0.03	-0.88	0.13
7	5	2.57	1.48	0.06	0.58	0.05
7	6	2.35	1.16	0.14	-0.42	0.23
7	7	2.37	1.29	0.06	-0.6	0.07
7	8	2.5	1.38	0.07	0.16	-0.64
7	9	2.4	1.27	0.11	0.24	0.17
7	10	2.95	1.71	0.1	-0.48	0.89
8	1	3.34	2.24	0.04	0.07	0.3
8	2	3.7	2.59	0.03	-0.14	0.35
8	3	3.2	1.84	0.07	-1.48	0.49
8	4	4.96	3.8	0.03	0.11	0.33
8	5	2.55	1.41	0.1	-0.15	0.34
8	6	2.08	1.01	0.04	-0.41	-0.83
8	7	2.28	1.16	0.05	-1.03	-0.26
8	8	2.8	1.71	0.05	-0.25	-0.31
8	9	2.04	0.76	0.18	0.39	0.21
8	10	1.91	0.74	0.11	0.48	-0.59
9	1	3.47	1.88	0.24	0.87	0.83
9	2	3.33	2.23	0.04	0.57	-0.27
9	3	2.39	1.27	0.09	0.23	0.49
9	4	3.1	1.95	0.09	0.45	0.24
9	5	4.09	2.96	0.02	-0.45	-0.48
9	6	2.31	1.19	0.07	-0.32	0.05
9	7	3.91	2.79	0.05	0.86	0.26
9	8	2.65	1.55	0.05	-0.31	0.32
9	9	2.39	1.27	0.06	0.13	-0.31
9	10	2.95	1.85	0.06	0.06	0.18
10	1	2.77	1.66	0.04	-0.81	0.49
10	2	2.25	1.21	0.05	-0.44	-0.35
10	3	2.63	1.62	0.01	-0.74	-1.02
10	4	2.77	1.58	0.13	0.14	0.04
10	5	2.8	1.65	0.09	0.28	-0.26
10	6	1.69	0.68	0.05	-0.27	0.21
10	7	2.06	1.01	0.05	-0.38	0.4
10	8	2.9	1.87	0.02	-0.37	-1.12
10	9	2.09	1.02	0.08	-0.24	-0.3
10	10	1.97	0.66	0.17	0.63	0.16