README : Model

Non-GAAP Reporting and Investment

Charles G. McClure Anastasia A. Zakolyukina*

20 August 2023

This note describes the code shared in `mz_share/model` that evaluates the model once. The model code is written in `Julia` and executed on the computing cluster that runs Linux.

```
Version 1.4.2 (2020-05-23)
Official https://julialang.org/ release
```

Evaluating model once

The code for evaluating the SMM objective function once is `eval.jl`. It evaluates the objective function in the baseline estimates from Table 4. You can run `eval.jl` in `Julia` interactively, which produces the following output:

```
Price coeffs = [736.4262405352564, 22.187192533628405, -1.4622654396171029]
Price coeffs = [754.0997731429491, 22.5918157936602, -1.5270770545944194]
x: 0.5745409793455529 0.3381737012722537 0.4884917981470520 0.1267994934858418 0.0212190418166888 0.3859639932137643 28.5047986163777587 0.011092679325
Flags header prc       rho       sing      coef      vf        qmid      qu        ql        wu        bu        bl
Flags        0         0         0         0         0         0         0         0         0         0         0
F(x)         = 4.313855312274457

 61.706351 seconds (50.50 M allocations: 9.896 GiB, 4.57% gc time)
```

*https://www.charlesgmcclure.com/, https://www.anastasiazakolyukina.xyz/

The single evaluation will create `/Output/moments.txt` with model-based moments computed from simulated data. We have already provided `moments.txt` file in the `/Output` folder. This file will be overwritten once you run the code.

This code uses various data-based statistics such as moments, weight matrix, and depreciation parameter. The files with these statistics are provided in `../moments_weights/csv`

Standard errors

The code snippet below shows the formula we used for computing standard errors. The code snippet below is in `Julia`.

```
....
# See get_moments.Rmd for file naming
# Two-way clustered optimal weight matrix
# Qmm_opt_clust_firm_xxx (clustered by {gvkey})
# Qmm_opt_clust_twoway_xxx (clustered by {gvkey, fyear})
# "./csv/Qmm_opt_clust_firm"//"_"//trim(suffix)".csv"
file = string("../moments_weights/csv/Qmm_opt_clust_twoway_", suffix, ".csv")
Omega_all = convert(Matrix{Float64}, readdlm(file, ',', skipstart = 1)[:, 2:end])

# Load weight matrix
# "csv/Qmm_"//trim(suffix)//".csv"
file = string(path, "/../moments_weights/csv/Qmm_", suffix, ".csv")
Qmm_all = convert(Matrix{Float64}, readdlm(file, ',', skipstart = 1)[:, 2:end]) # covariance matrix of moments

# Pick out moments used in estimation for W and Omega
W = inv(Qmm_all[mom_index, mom_index]) # fval = (sim_mom - data_mom)'*W*(sim_mom - data_mom)
Omega = Omega_all[mom_index, mom_index]
....
function se_calc(x, # optimal set of parameters from objective minimizations
                 jacob, # Jacobian matrix
                 Omega, # Optimal (clustered) co-variance matrix of moments
                 W, # weight matrix for GMM
                 nobs, nsim)
    # This function computes parameter estimate standard errors

    # Long formula for non-optimal weight matrix
    # sum(eigvals(W) .< 0.0)
    outer = inv(jacob * W * jacob')
    x_cov = (1/nobs) * (1 + 1/nsim) * outer * (jacob * W * Omega * W * jacob') * outer

    # Compute SE
    x_se = sqrt.(diag(x_cov))
```

```
    return x_se, x_cov
end
```