Computational Economics
https://doi.org/10.1007/s10614-020-09995-z

Implementing Convex Optimization in R: Two Econometric Examples

Zhan Gao[1] · **Zhentao Shi**[2]

Accepted: 8 May 2020

Abstract
Economists specify high-dimensional models to address heterogeneity in empirical studies with complex big data. Estimation of these models calls for optimization techniques to handle a large number of parameters. Convex problems can be effectively executed in modern programming languages. We complement Koenker and Mizera (J Stat Softw 60(5):1–23, 2014)'s work on numerical implementation of convex optimization, with focus on high-dimensional econometric estimators. Combining `R` and the convex solver `MOSEK` achieves speed gain and accuracy, demonstrated by examples from Su et al. (Econometrica 84(6):2215–2264, 2016) and Shi (J Econom 195(1):104–119, 2016). Robust performance of convex optimization is witnessed across platforms. The convenience and reliability of convex optimization in `R` make it easy to turn new ideas into executable estimators.

Keywords Big data · Convex optimization · High-dimensional model · Numerical solver

JEL Classification C13 · C55 · C61 · C87

1 Introduction

Equipped with tremendous growth of computing power over the last few decades, we econometricians endeavor to tackle high-dimensional real-world problems that we could hardly have imagined before. Along with the development of

✉ Zhentao Shi
zhentao.shi@cuhk.edu.hk

Zhan Gao
zhangao@usc.edu

1 Department of Economics, University of Southern California, 3620 South Vermont Ave. Kaprielian Hall, 300, Los Angeles, CA 90089-0253, USA

2 Department of Economics, 9F Esther Lee Building, The Chinese University of Hong Kong, Sha Tin, New Territories, Hong Kong SAR, China

Published online: 02 June 2020

modern asymptotic theory, computation has gradually ascended onto the central stage. Today, discussion of numerical algorithms is essential for new econometric procedures.

Optimization is at the heart of statistical estimation, and convex optimization is the best understood category. Convex problems are ubiquitous in econometric textbooks. The least square problem is convex, and the classical normal regression is also convex after straightforward reparametrization. Given a linear single-index form, the Logit or Probit binary regression, the Poisson regression and the regressions with a censored or truncated normal distributions are all convex. Another prominent example is the quantile regression (Koenker and Bassett 1978).

With the advent of big data, practitioners attempt to build flexible models that involve hundreds or even more parameters in the hope to capture complex heterogeneity in empirical economic studies. Convex optimization techniques lay out the foundation of estimating these high-dimensional models. Recent years witnesses Bajari et al. (2015), Gu and Koenker (2017) and Doudchenko and Imbens (2016), to name a few, exploring new territories by taking advantage of convexity.

To facilitate practical implementation, Koenker and Mizera (2014) summarize the operation in `R` by `MOSEK` via `Rmosek` to solve linear programming, conic quadratic programming, quadratic programming, etc. `R` is open-source software, `MOSEK` is a proprietary convex optimization solver but it offers free academic licenses, and `Rmosek` is the `R` interface that communicates with `MOSEK`. `MOSEK` specializes in convex problems with reliable performance, and is competitive in high-dimensional problems.

This paper complements Koenker and Mizera (2014)'s work. We revisit two examples of high-dimensional estimators, namely Su et al. (2016)'s classifier-Lasso (C-Lasso) and Shi (2016)'s relaxed empirical likelihood (REL) by `Rmosek`. Other than Monte Carlo simulations, we also replicate a real data application that examines China's GDP growth rate (Chen et al. 2019). These exercises highlight two points. Firstly, the `R` environment is robust in numerical accuracy for high-dimensional convex optimization and `Rmosek` takes the lead in computational speed. Second, we showcase the ease of creating new econometric estimators—often no more than a few lines of code—by the code snippets (in the supplement due to space limitations). Such convenience lowers the cost of turning an idea into a prototype, and enables researchers to glean valuable insights about their archetypes by experimenting new possibilities.

Replication code and supplementary materials are hosted at https://github.com/zhan-gao/convex_prog_in_econometrics. The supplement provides the details of the data generating processes (DGP) of the simulation, additional results of the empirical application, and code snippets of convex optimization formulation.

2 Classifier-Lasso

In linear fixed-effect panel data models, researchers routinely assume that the cross-sectional units are heterogeneous in terms of the time-invariant individual intercept, while they all share the same slope coefficient. This pooling assumption can be tested but is often rejected in real-world applications. In recent years panel data group structure has been developed into a burgeoning literature in econometrics.

When the slope coefficients exhibit group structure, Su et al. (2016) propose C-Lasso to identify the latent group pattern in the likelihood estimation framework. Here we illustrate with a special linear case of C-Lasso, the penalized least square (PLS). Given a tuning parameter λ and the number of groups K, PLS is defined as the solution to

$$\min_{\beta,(\alpha_k)_{k=1}^K} \frac{1}{nT}\sum_{i=1}^{n}\sum_{t=1}^{T}\left(y_{it}-x_{it}'\beta_i\right)^2+\frac{\lambda}{n}\sum_{i=1}^{n}\prod_{k=1}^{K}\|\beta_i-\alpha_k\|_2$$

where $x_{it}\in\mathbb{R}^p$ is the regressor, β_i is its slope coefficient, $\boldsymbol{\beta}=\left(\beta_i\right)_{i=1}^n$, and $\|\cdot\|_2$ is the l_2-norm of a vector. The additive-multiplicative penalty pushes the individual slope coefficients β_i in the same group toward a common coefficient α_k.

Although this subjective with the additive-multiplicative penalty is not a convex function, Su et al. (2016, Supplement Section S3.1) approximate the solution by an iterative algorithm of convex optimization until numerical convergence. Procedures based on such iteration have been successfully applied to Su and Ju (2018), Su and Lu (2017) and Su et al. (2019). The iterative algorithm initiates at the within-group estimator, which is consistent when T is large. In the kth sub-step of the rth iteration, $(\boldsymbol{\beta},\alpha_{\tilde{k}})$ is chosen to minimize

$$\min_{\beta,\alpha_{\tilde{k}}} \frac{1}{nT}\sum_{i=1}^{n}\sum_{t=1}^{T}\left(y_{it}-x_{it}'\beta_i\right)^2+\frac{\lambda}{n}\sum_{i=1}^{n}\|\beta_i-\alpha_{\tilde{k}}\|_2\gamma_i \tag{1}$$

where $\gamma_i=\prod_{k=1}^{\tilde{k}-1}\|\hat{\beta}_i^{(r,k)}-\hat{\alpha}_k^{(r)}\|_2\cdot\prod_{k=\tilde{k}+1}^{K}\|\hat{\beta}_i^{(r-1,k)}-\hat{\alpha}_k^{(r-1)}\|_2$. Given the multiplier γ_i, the above optimization problem is convex in the high-dimensional parameter $(\boldsymbol{\beta},\alpha_{\tilde{k}})\in\mathbb{R}^{p(n+1)}$ and the structure is close to Lasso. While standard Lasso shrinks β to 0, PLS shrinks the slope coefficients to the center $\alpha_{\tilde{k}}$, which is to be optimized. Thus it cannot be solved by `R` packages such as `LARS` or `glmnet`. Reviewing Koenker and Mizera (2014)'s solution to Lasso offers guidance.

Example 1 (Lasso) The standard Lasso problem is

$$\min_{\beta}\frac{1}{n}\|y-X\beta\|_2^2+\lambda\|\beta\|_1$$

where $y\in\mathbb{R}^n$ and $X\in\mathbb{R}^{n\times p}$ are observed data, λ is the tuning parameter and β is the parameter of interests. Koenker and Mizera (2014) introduce new parameters to transform the l_1-penalized problem into a conic optimization to overcome the difficulty that `Rmosek` does not accept the l_1 norm. We first deal with $\|\beta\|_1$. The $p\times 1$ vector β can be decomposed into a positive part $\beta^+=\left(\max\left\{0,\beta_j\right\}\right)_{j=1}^p$ and a negative part $\beta^-=\left(\max\left\{0,-\beta_j\right\}\right)_{j=1}^p$, so that $\beta=\beta^+-\beta^-$ and $\|\beta\|_1=e'\beta^++e'\beta^-$, where e is the $p\times 1$ vector with all elements equal to 1. Next, we transform the l_2-norm $\|y-X\beta\|_2^2$ to a second-order conic constraint. Consider a minimization problem with $\|v\|_2^2$ in the objective function. We can use a new parameter t to replace it and add a conic constraint $\|v\|_2^2 \le t$, which is equivalent to $\left\| \left(v, \frac{t-1}{2} \right) \right\|_2 \le \frac{t+1}{2}$. Thus we obtain a standard conic constraint $\|(v,s)\|_2 \le r$, where $s = \frac{t-1}{2}$ and $r = \frac{t+1}{2}$. We rewrite the Lasso problem as

$$\min_{\theta} \lambda\left(e'\beta^{+} + e'\beta^{-}\right) + \frac{t}{n} \text{ s.t. } v = y - X\left(\beta^{+} - \beta^{-}\right),\ \|(v,s)\|_2 \le r,\ s = \frac{t-1}{2},\ r = \frac{t+1}{2}$$

where $\theta = \left(\beta^{+}, \beta^{-}, v, t, s, r\right)$. This problem is of the standard form of second-order conic programming and hence can be executed in `Rmosek`.

Koenker and Mizera (2014)'s idea of transformation Lasso into a conic programming can be carried over to (1):

$$\begin{aligned} &\min_{\alpha_{\tilde{k}},\theta} \sum_{i=1}^{n} \frac{t_i}{nT} + \frac{\lambda}{n}\gamma_i \cdot w_i \\ &\text{s.t. } v_i = y_i - x_i\beta_i,\ \beta_i - \alpha_{\tilde{k}} = \mu_i,\ s_i = \frac{t_i - 1}{2},\ r_i = \frac{t_i + 1}{2}, \\ &\left\| \left(v_i, s_i\right) \right\|_2 \le r_i,\ \left\|\mu_i\right\|_2 \le w_i,\ t_i \ge 0, \quad \text{for all } i = 1, 2, \ldots, n \end{aligned}$$

where $\theta = \left\{\beta_i, v_i, \mu_i, s_i, r_i, t_i, w_i\right\}_{i=1}^{n}$. This is the formulation we will feed into `Rmosek`.

2.1 Simulation

We replicate the simulation study in Su et al. (2016, Section 4) in `R` via `Rmosek` and compare the performance of different numerical optimization approaches. Su et al. (2016) conduct their numerical work in `MATLAB` via `CVX` (Grant and Boyd 2014). `CVX` is a `MATLAB` add-on optimization modeling package that provides an interface to communicate with commercial or open-source solvers. In the `R` environment, the *de facto* solver is `optimx` (Nash and Varadhan 2011); another option is the interface `nloptr` (Ypma 2017) which hooks optimization solver `NLopt` (Johnson 2017). They are general-purpose optimization solvers not tailored for convexity. Most recently, Fu et al. (2019) are actively developing `CVXR`, `CVX`'s counterpart in `R`. By default it is integrated with the open-source solver `ECOS` (Domahidi et al. 2013), and starting from version 1.0, it can invoke `Rmosek`.[1] We also consider the counterpart of `CVX` in `Python` environment, `CVXPY` (Diamond and Boyd 2016), to verify the stability of the algorithm across platforms.

Su et al. (2016, Section 4)'s DGP 1 serves as a benchmark. Table 1 reports the root-mean-square error (RMSE) of $\widehat{\alpha}_1$ and the probability of correct group classification (correct ratio) under various combinations of the cross sectional units n and the

[1] At the writing of this note, `CVXR + MOSEK` takes from minutes to hours to compute one estimation depending on sample sizes, which makes the full-scale simulation exercise computational infeasible.

Table 1 Classification and point estimation of α_1: Su et al. (2016, DGP 1)

(n, T)	(100, 15)	(100, 25)	(100, 50)	(200, 15)	(200, 25)	(200, 50)
RMSE						
`R::Rmosek`	0.0762	0.0386	0.0247	0.0428	0.0278	0.0174
`R::CVXR`	0.0762	0.0386	0.0247	0.0427	0.0278	0.0174
`MATLAB::CVX`	0.0767	0.0399	0.0253	0.0443	0.0286	0.0179
`Python::CVXPY`	0.0741	0.0394	0.0253	0.0424	0.0271	0.0173
Correct ratio						
`R::Rmosek`	0.8987	0.9645	0.9965	0.9019	0.9668	0.9969
`R::CVXR`	0.8986	0.9645	0.9965	0.9020	0.9668	0.9969
`MATLAB::CVX`	0.8991	0.9647	0.9965	0.9026	0.9667	0.9968
`Python::CVXPY`	0.8988	0.9644	0.9965	0.9020	0.9668	0.9969
Running time (in minute)						
`R::Rmosek`	16.73	10.97	8.89	25.85	16.45	14.08
`R::CVXR`	147.94	89.71	68.16	172.89	114.61	102.65
`MATLAB::CVX`	81.21	48.55	34.07	87.71	51.77	37.98
`Python::CVXPY`	21.11	11.37	18.42	27.78	20.72	22.66

To compare the running time, each case is executed in a single thread on the same computing platform of Intel(R) Core(TM) i7-8750H CPU @ 2.20 GHz. The iterative algorithm optimizes $2(n+1)$ parameters in each iteration until numerical convergence

time length T. The DGP, simulation settings and the indicators are relegated to the supplement.

In our experiment, we find in the `R` environment `optimx` breaks down when solving such high dimensional problems. `nloptr` takes more than a few hours to finish one estimation, which makes the full-scale simulation exercise computational infeasible, and what is worse, `nloptr` fails to attain an accurate solution in most cases. The numerical results of estimation error and classification correct ratio by `Rmosek` are almost identical to `CVXR` up to rounding errors. In addition, we implement the simulation in `MATLAB` via `CVX` and in `Python` via `CVXPY`, and the results are largely similar. This extensive simulation exercise demonstrates the robustness of the numerical performance of C-Lasso across a multitude of computing platforms.

Practitioners may need to try out different specifications for robustness check in real applications. Without fast optimization solvers, computational cost can become a bottleneck. `Rmosek` significantly outperforms all alternatives in terms of computing time. According to the lower panel of Table 1, `CVX` in `MATLAB` is about 2.70–4.85 times slower and `CVXPY` is about 1.03–2.07 times slower than `Rmosek`. Although `CVX` and `CVXPY` are also powered by `MOSEK`, the optimization modeling package takes time to check the convexity of the input problem and automate the formulation. `CVXR` is 6.68–8.84 times slower than `Rmosek`. In summary, an optimization modeling package is useful at the trial-and-error stage. However, for problems that are mathematically verified to be convex, directly calling `MOSEK` saves much computational time in full-scale implementation.

Table 2 Running time (in second): replication of Chen et al. (2019, Table A11)

	With light	No light	No light and Tax
`R::Rmosek`	87.37	102.93	42.50
`R::CVXR`	783.21	564.23	557.53
`MATLAB::CVX`	578.05	699.44	294.69

2.2 Empirical Application

In real data applications, experimenting with various specifications and tuning parameters is time-consuming, particularly for C-Lasso in panel data of a short T as the iterative algorithm has to iterate many times until convergence. We demonstrate the speed gain by Rmosek via the short panel example of Chen et al. (2019).

While China is the second largest economy in the world in terms of aggregate GDP, the accuracy of its reported national income accounting has been a topic of constant debate over the years. Chen et al. (2019) utilize local economic indicators directly associated with economic activities to estimate China's local and aggregate GDP in order to assess the quality of these numbers. Different regions of this continent-size country are growing at varying pace, thereby resulting in tremendous heterogeneity among its provinces. To control the hidden heterogeneity, Chen et al. (2019) specify a linear fixed effect model with latent group structure

$$y_{it} = x_{it}'\beta_i + v_i + \varepsilon_{it},$$

where y_{it} is the logarithm of GDP for province i at year t, x_{it} includes local economic indicators of interests, v_i characterizes the fixed effect of province i, and ε_{it} is the idiosyncratic error. The heterogeneous slope coefficients β_i capture latent group structures across regions to be determined by C-Lasso. The data span from year 2000 to 2007, i.e. $T = 8$. Five indicators are employed as regressors to control observable heterogeneity, namely *satellite night lights*, *national tax revenue*, *exports*, *imports*, and *electricity consumption*. These indicators are less susceptible to local officials' manipulation and thus better reflect economic activities for real businesses. Two alternative specifications, one with no *satellite night lights* and the other with neither *satellite night lights* nor *national tax revenue*, are also considered.

In our implementation, the number of groups K and tuning parameter λ are determined by the information criterion proposed in Su et al. (2016, Section 2.5). Chen et al. (2019) estimate the model in `MATLAB` via `CVX` and the classification results are reported in their Table A11. We replicate the classification results by `Rmosek` and compare the accuracy and speed to `CVX`. The same as in the original paper, in all three specifications the information criterion determines two groups. The results are detailed in the supplement though, here we report the CPU time consumed by the parameter tuning process in Table 2. `Rmosek` is about 6.61–6.93 times faster than `CVX`, and about 5.48–13.12 times faster than `CVXR`.

3 Relaxed Empirical Likelihood

Besides the regression setting in Sect. 2, convex programming is also useful in structural econometric estimation. Consider the models with a "true" parameter β_0 satisfying the unconditional moment condition $\mathbb{E}\left[g\left(Z_i, \beta_0\right)\right] = \mathbf{0}_m$, where $\left\{Z_i\right\}_{i=1}^n$ is the observed data, $\beta \in \mathbb{R}^p$ is a finite-dimensional vector, and g is an $\mathbb{R}^m$-valued moment function. The generalized method of moments (GMM) and empirical likelihood (EL) are two workhorses dealing with moment restriction models in econometrics. In particular, EL solves

$$\max_{\beta \in \mathcal{B}, \pi \in \Delta_n} \sum_{i=1}^n \log \pi_i \quad \text{s.t.} \quad \sum_{i=1}^n \pi_i g\left(Z_i, \beta\right) = \mathbf{0}_m$$

where $\Delta_n = \left\{\pi \in [0,1]^n : \sum_{i=1}^n \pi_i = 1\right\}$ is the n-dimensional probability simplex. However, neither GMM nor EL is capable of estimating a model with more moment equalities than observations, i.e. $m > n$. To make the optimization feasible, Shi (2016) relaxes the equality restriction $\sum_{i=1}^n \pi_i g_i(\beta) = \mathbf{0}_m$ in EL. The relaxed empirical likelihood (REL) estimator is defined as the solution to

$$\max_{\beta \in \mathcal{B}} \max_{\pi} \sum_{i=1}^n \log \pi_i, \quad \text{s.t. } \pi \in \left\{ \Delta_n : \left| \sum_{i=1}^n \pi_i g_{ij}(\beta) \right| \le \lambda, j = 1, 2, \ldots, m \right\},$$

where $\lambda \ge 0$ is a tuning parameter, $g_{ij}(\beta) = g_j\left(Z_i, \beta\right)$ is the jth component of the vector $g\left(Z_i, \beta\right)$.

Similar to standard EL, REL's optimization involves an inner loop and an outer loop. The outer loop for β is a low-dimensional nonlinear optimization, which can be solved by Newton-type methods. With the linear constraints and the logarithm objective, the inner loop is convex in $\pi = \left(\pi_i\right)_{i=1}^n$. Start from the version 9.0, `MOSEK` supports the exponential cone, which can be used to model a variety of expressions with logarithm and exponential. By introducing auxiliary variables $t = \left(t_i\right)_{i=1}^n$, the logarithm objective can be reformulated as a linear objective function $\sum_{i=1}^n t_i$ and n exponential conic constraints, $\left(\pi_i, 1, t_i\right) \in \mathcal{K}_{\exp} := \left\{\left(x_1, x_2, x_3\right) : x_1 \ge x_2 \exp\left(x_3/x_2\right), x_2 > 0\right\} \cup \left\{\left(x_1, 0, x_3\right) : x_1 \ge 0, x_3 \le 0\right\}$, $i = 1, 2, \ldots, n$. For each β, the inner problem can be then formulated as a conic programming problem

$$\max_{\pi, t} \sum_{i=1}^n t_i \ \text{ s.t. } \begin{bmatrix} 1 \\ -\lambda \mathbf{1}_m \end{bmatrix} \le \begin{bmatrix} \mathbf{1}_n' \\ \left[g_{ij}(\beta)\right]_{i,j} \end{bmatrix} \pi \le \begin{bmatrix} 1 \\ \lambda \mathbf{1}_m \end{bmatrix}, \ \left(\pi_i, 1, t_i\right) \in \mathcal{K}_{\exp}, \ 0 \le \pi_i \le 1,$$

and it is readily solvable in `Rmosek` by translating the mathematical expression into computer code.

Table 3 REL estimation in linear IV model: replication of Shi (2016)

(n, m)	(120, 80)	(120, 160)	(240, 80)	(240, 160)
Estimation of $\widehat{\beta}_1$ in the outer loop				
`R::nloptr`				
Bias	– 0.020	– 0.018	– 0.004	– 0.008
RMSE	0.135	0.162	0.078	0.093
`MATLAB::fmincon`				
Bias	– 0.004	– 0.012	– 0.006	– 0.009
RMSE	0.113	0.143	0.071	0.077
Running time of REL's inner loop (in second)				
`R::Rmosek`	3.250	5.424	5.778	9.147
`R::nloptr`	31.621	67.338	65.845	147.961
`R::CVXR`	37.266	44.651	43.365	101.814
`MATLAB::CVX`	18.953	20.049	20.951	23.900

3.1 Simulation

We follow the simulation design in Shi (2016, Section 4), described in the supplement. The upper panel of Table 3 reports the bias and RMSE of the estimation of $\hat{\beta}_1$, the first element of the vector β, implemented purely in `R` with the inner loop by `Rmosek` and the outer loop by `nloptr`. The results are close to those in Shi (2016), where the code was written in `MATLAB` with the outer loop handled by the function `fmincon` and the inner loop by `CVX` solved by `MOSEK`.

We also experiment with other numerical alternatives. Since the scale of the optimization problems here is smaller than C-Lasso, the convex inner loop can be correctly solved by `Rmosek`, `CVXR`, `CVX` in `MATLAB`, or even `nloptr`.[2] These four methods produce virtually identical inner loop results up to rounding errors. This finding confirms the robustness of the `R` environment in high-dimensional optimization. The small difference in the upper panel of Table 3, therefore, is attributed to the outer loop between the function `nloptr` in `R` and the function `fmincon` in `MATLAB`.

We report the running time of each method in the lower panel of Table 3 to evaluate the computational cost in the inner loop. We fix $\beta = (0.9, 0.9)$, simulate 100 sets of data for each sample size, and numerically solve the inner loop only. Although `CVXR` and `nloptr` are able to correctly solve the problem thanks to its small scale, `Rmosek` remains 2.61 and 16.18 times faster than these alternatives.

[2] `CVXR` + `MOSEK` does not support exponential/logarithm objective functions and hence is infeasible for REL.

4 Conclusion

In this note along with its supplement and code, we demonstrate numerical implementation via `Rmosek` of two examples of high-dimensional econometric estimators. The convenience and reliability of high-dimensional convex optimization in `R` will open new possibilities to create estimation procedures. In the era of big data, we are looking forward to witnessing more algorithms blossoming and flourishing together with theoretical research of high-dimensional models.

Acknowledgements Shi thanks Roger Koenker for inspiration and hospitality during his visit to University of Illinois.

References

Bajari, P., Nekipelov, D., Ryan, S. P., & Yang, M. (2015). Machine learning methods for demand estimation. *American Economic Review*, *105*(5), 481.

Chen, W., Chen, X., Hsieh, C.-T., & Song, Z. (2019). A forensic examination of China's national accounts. *Brookings Papers on Economic Activities Spring*, 77–127. https://books.google.com.hk/books?hl=en&lr=&id=8kS8DwAAQBAJ&oi=fnd&pg=PA77&ots=84scMFLqBp&sig=m4yQpWnTftx6ly6mASjzGGO4FEY&redir_esc=y#v=onepage&q&f=false.

Diamond, S., & Boyd, S. (2016). CVXPY: A python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, *17*(83), 1–5.

Domahidi, A., Chu, E., & Boyd, S. (2013). ECOS: An SOCP solver for embedded systems. *European Control Conference (ECC)*, *2013*, 3071–3076.

Doudchenko, N., & Imbens, G. W. (2016). *Balancing, regression, difference-in-differences and synthetic control methods: A synthesis*. Technical report, National Bureau of Economic Research No. 22791.

Fu, A., Balasubramanian, N., & Boyd, S. (2019). CVXR: An R package for disciplined convex optimization. *Journal of Statistical Software*, arxiv:1711.07582.

Grant, M., & Boyd, S. (2014). `CVX`: *Matlab software for disciplined convex programming*, version 2.1. http://cvxr.com/cvx.

Gu, J., & Koenker, R. (2017). Empirical bayesball remixed: Empirical bayes methods for longitudinal data. *Journal of Applied Econometrics*, *32*(3), 575–599.

Johnson, S. G. (2017). *The* `nlopt`*nonlinear-optimization package*.

Koenker, R., & Bassett, G. (1978). Regression quantiles. *Econometrica*, *46*, 33–50.

Koenker, R., & Mizera, I. (2014). Convex optimization in R. *Journal of Statistical Software*, *60*(5), 1–23.

Nash, J. C., & Varadhan, R. (2011). Unifying optimization algorithms to aid software system users: optimx for R. *Journal of Statistical Software*, *43*(9), 1–14.

Shi, Z. (2016). Econometric estimation with high-dimensional moment equalities. *Journal of Econometrics*, *195*(1), 104–119.

Su, L., & Ju, G. (2018). Identifying latent grouped patterns in panel data models with interactive fixed effects. *Journal of Econometrics*, *206*(2), 554–573.

Su, L., & Lu, X. (2017). Determining the number of groups in latent panel structures with an application to income and democracy. *Quantitative Economics*, *8*(3), 729–760.

Su, L., Shi, Z., & Phillips, P. C. (2016). Identifying latent structures in panel data. *Econometrica*, *84*(6), 2215–2264.

Su, L., Wang, X., & Jin, S. (2019). Sieve estimation of time-varying panel data models with latent structures. *Journal of Business & Economic Statistics*, *37*(2), 334–349.

Ypma, J. (2017). `nloptr`: *R interface to NLopt R package version 1.0.4*. https://cran.r-project.org/web/packages/nloptr/index.html.