Regression Results

2023-06-18

Libraries

```
library(TensorEconometrics)
library(MultiwayRegression)
library(dplyr)
set.seed(20230615)
```

Convergence of Tucker Regression

For convergence, I estimate all of $\mathcal{G}, \mathbf{U}^{(1)}, \ldots, \mathbf{U}^{(4)}$, while saving the difference between the current and previous iteration. I then sum all these differences and check whether this sum is less than a preset convergence criterion.

To check whether I jump between solutions, I plot the differences for each of $\mathcal{G}, \mathbf{U}^{(1)}, \ldots, \mathbf{U}^{(4)}$ to see convergence plots. First, we simulate random normal predictor tensors and response tensors of size $100 \times 20 \times 4$ and preset rank of `R=c(8,3,8,3)`. Congergence occurs after 151 iterations. The convergence threshold is 1e-04.

Convergence Plots are given in figure 1

I also do this for real data. Here, the difference is I set `R = c(6,3,6,3)` and I have tensor data of size $161 \times 32 \times 3$. Results are found in figure 2.

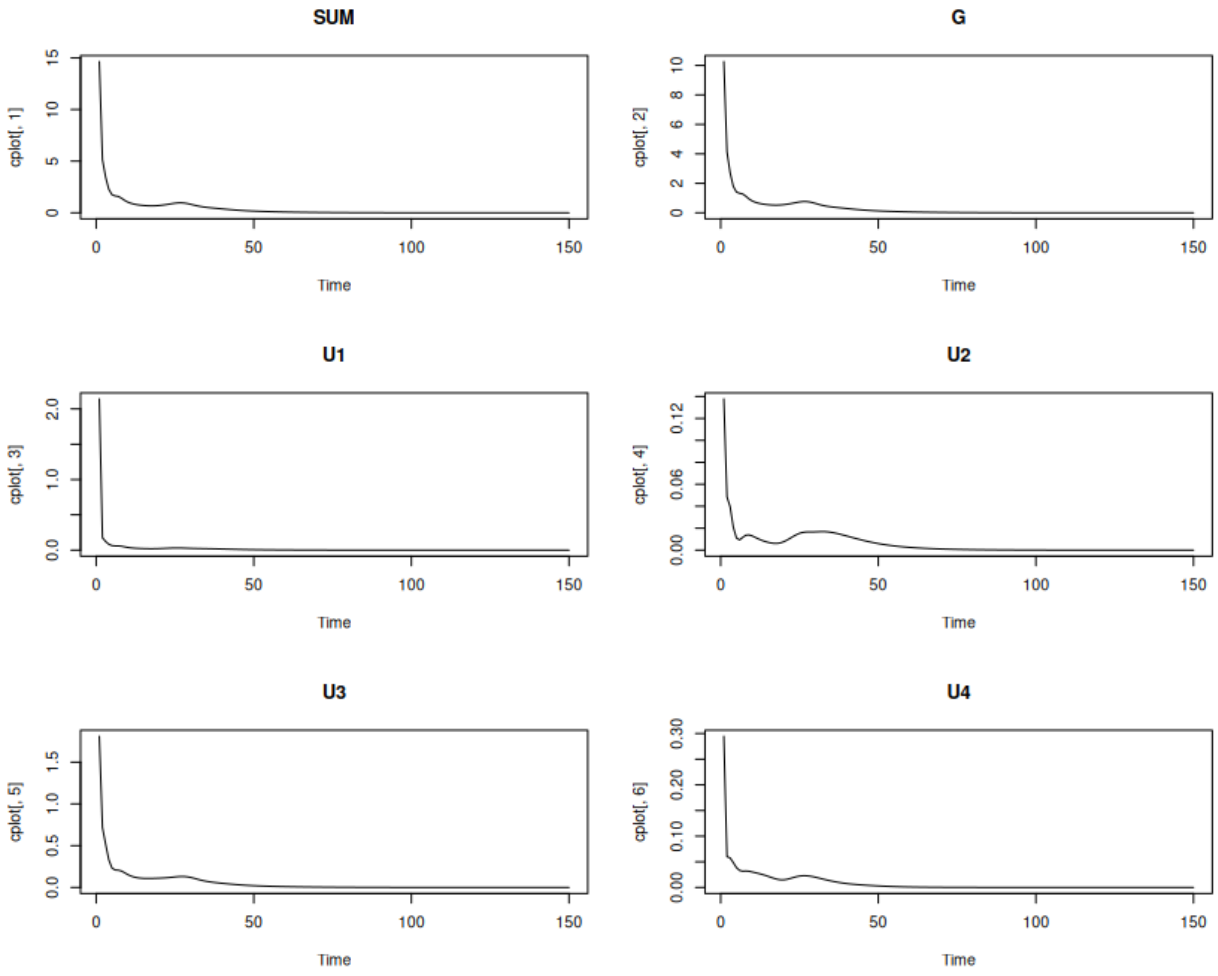

Figure 1: Convergence Plots

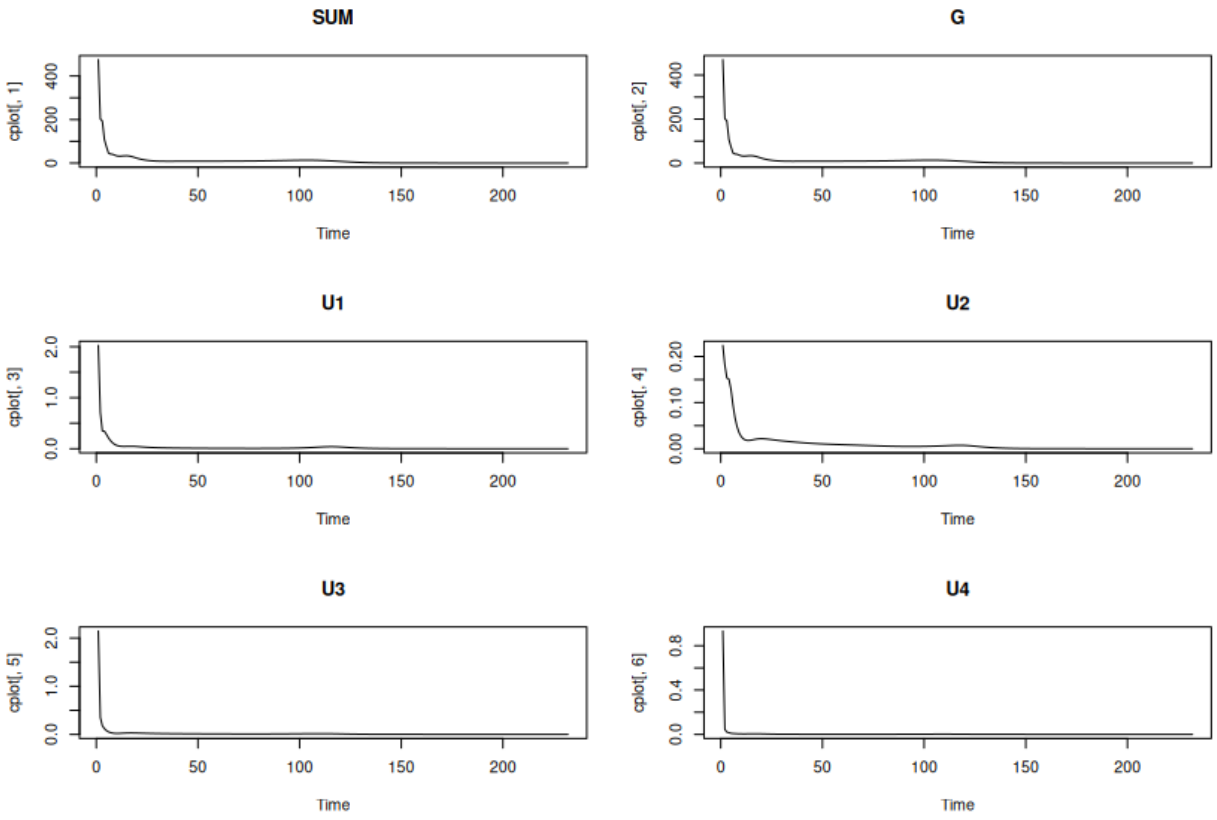

Figure 2: Real Data Plots

CV Results

CP Regression

For choosing a rank, I use a cross-validation type rolling window forecast to see how well our models do out of sample. The data is split 70% training and 30% testing for a total of 113 training observations and 48 testing observations. I then estimate a CP regression of the form

$$\mathcal{Y}_t = \mathcal{Y}_{t-1} \bar{\times}_2 \mathcal{B} + \mathcal{E}_t$$

where $\mathcal{Y}_t : 32 \times 3$, $\mathcal{B} : 32 \times 3 \times 32 \times 3$ for $t = 1, \ldots, 161$. Error metrics are obtained from the Frobenius norm of the difference between predicted and actual results. In other words, we have

$$\|\mathcal{Y}_t - \mathcal{Y}_t^*\|_2^2 = \sqrt{\sum_{t=1}^{113} \sum_{i=1}^{32} \sum_{j=1}^{3} (y_{tij} - y_{tij}^*)^2}$$

where $\mathcal{Y}_t$ is the actual result at time period t and $\mathcal{Y}_t^*$ is the predicted result at time period t.

This metric is found for 48 out of sample time points to form a vector. We do this for ranks $R = 1, \ldots, 7$ and find the means for each specified rank over the 48 time points. The results for my CP Regression and the one from Lock (2017) are presented below, along with the OLS estimates for comparison.

```
CP_rw <- readRDS("CP_rw.rds")
rrr_rw <- readRDS("rrr_rw.rds")
```

```
hools_rw <- readRDS("HOOLS_rw.rds")

colMeans(CP_rw)
```

```
## [1] 0.01736554 0.01757437 0.01727022 0.01737536 0.01745471 0.01747801 0.01809324
```

```
colMeans(rrr_rw)
```

```
## [1] 0.01810605 0.02041707 0.02155864 0.02815635 0.02654711 0.02973468 0.03102092
```

```
colMeans(hools_rw)
```

```
## [1] 0.04065782 0.04065782 0.04065782 0.04065782 0.04065782 0.04065782 0.04065782
```

where `cp_rw` are the rolling window results for my CP regression, `rrr_rw` is the rolling window results for Lock's CP regression, and `hools_rw` is the rolling window results for the OLS estimate. Each element in the vector corresponds to a chosen rank from $r = 1, \ldots, 7$.

Tucker Regression

The same rolling window exercise is done for the Tucker regression. Because the ranks can vary for each dimension, I perform the rolling window forecast for all combinations of R_1, R_2, R_3, R_4 from 2 to 5. This yields a total of 16 possibilities.

Rank	RMSFE	Rank	RMSFE	Rank	RMSFE	Rank	RMSFE
[2 3 2 3]	0.1572	[3 3 2 3]	0.1572	[4 3 2 3]	0.1603	[5 3 2 3]	0.1472
[2 3 3 3]	0.1632	[3 3 3 3]	0.1470	[4 3 3 3]	0.1602	[5 3 3 3]	0.1416
[2 3 4 3]	0.1569	[3 3 4 3]	0.1630	[4 3 4 3]	0.1392	[5 3 4 3]	0.1419
[2 3 5 3]	0.1574	[3 3 5 3]	0.1603	[4 3 5 3]	0.1575	[5 3 5 3]	0.1367

Figure 3: CV Rank Results

I also do this for R=c(r, 3, r, 3). In other words, always estimate a model with three components such that I can see how models with R = c(1,3,1,3), R = c(2,3,2,3), R = c(3,3,3,3), etc. perform. The results are as follows.

```
tucker_rw <- readRDS("tucker_rw.rds")
colMeans(tucker_rw)
```

```
## [1] 0.01901742 0.02117117 0.02439302 0.02699167 0.03306718 0.03280443 0.03465881
```