Appendix.

Anonymous Author(s)
Affiliation
Address
`email`

In this appendix, we provide additional details on our experiments. We recall that we compare our *BKKs* procedures against RidgeCV, LassoCV and ElasticnetCV (as implemented in Scikit-learn [3]) on simulated and real datasets. Our procedures are implemented on the centered and rescaled response $\mathbf{Y}$ using PyTorch.

Our Python code is released as an open source package for replication: github/AnonymousSubmissionNeurips2020/.

Configuration machine. We run our experiments with this configuration:

Cloud:	Google Cloud Plateform
CPU:	Intel Haswell 16 vCPUs
GPU:	NVIDIA Tesla P100
RAM:	104 GB
MEM:	500 GB SSD
Image:	rapids-0-7-gpu-experimental-notebooks

1 Initialization for the *BKKs* procedures

We investigated empirically the initialization values of the regularization parameters as well as hyperparameters of ADAM in the $n > p$ setting as summarized in Table 1 below. We found out that a set of standard values produces optimal results in the wide variety of cases we tested. We did not attempt to optimize these parameters in the setting $n \leq p$ as they already produce good results in that case. Nevertheless, investigating the optimal choice of parameters in this setting is an interesting open problem. For all other hyperparameters, we use the default values in the PyTorch implementation.
To replicate ours results, we underline again that these values do not require any tuning for all the datasets we considered.

Optimization parameters		**Parameter initialization**	
Tolerance	10^{-4}	T	30
Max. iter.	10^3	λ	10^3
Learning rate	0.5	γ	0_p
Adam β_1	0.5	κ	0.1
Adam β_2	0.9	μ	0

Table 1: Parameters for the *BKKs* procedures.

2 Data Processing

2.1 Synthetic data

We generate $n = n_{train} + n_{test}$ i.i.d. observations from the model $(\mathbf{x}, Y) \in \mathbb{R}^p \times \mathbb{R}$, $p = 80$ $s.t.$ $Y = \mathbf{x}^\top \beta^* + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma)$, $\mathbf{x} \sim N(0_p, \Sigma)$ and $\beta^* \in \mathbb{R}^p$ are mutually independent.

Scenario A (Correlated features) corresponds to the case where the LASSO is prone to fail and Ridge should perform better. We use the following parameters to generate the data: $\sigma = 50$, $\Sigma = \mathbb{I}_p + H$ with $H_{i,j} = 0.5^\rho$ and $\rho \sim \mathcal{U}([1, 2])$ and the components of β^* are $i.i.d.$ random variables $s.t.$ $\beta^*_j = R_j \times U_j$, where $U_j \sim \mathcal{U}([9, 15])$ is independent from R_j and $\mathbb{P}(R_j = \pm 1) = 1/2$.

Scenario B (Sparse setting) corresponds to a case known to be favorable to LASSO . Here we have $\sigma = 10$, $\Sigma = \mathbb{I}_p$, sparsity level $s = 11$, $\beta^*_j = 0$ if $j > s$ and the $\beta^*_j = R_j \times U_j$ are $i.i.d.$ for $1 \leq j \leq s$ where U_j and R_j are generated as in Scenario A.

Scenario C (Sparsity and correlated features) combines the two previous scenarii with $\sigma = 50$, the covariance Σ from Scenario A and the sparse vector β^* from Scenario B with sparsity level $s = 35$.

For each scenario we sample a $train$-dataset of size $n_{train} = 100$ and a $test$-dataset of size $n_{test} = 1000$ and then perform $M = 100$ repetitions of the data generation process.

2.2 Real data

We test our methods on several commonly used real datasets (UCI [1] and Svmlib [2] repositories) described in Table 2.

UCI data. We select only datasets that (a) correspond to non trivial regression models (neither the OLS nor the intercept are optimal solutions); (b) contain only continuous or ordinal features; (c) are not time series; (d) correspond to tabular data (eg. no text or image).
We replace the missing values using the scikit learn class ”SimpleImputer” with strategy='mean'. We divide UJIndoorLoc into two regression tasks, predicting the columns LONGITUDE and LATITUDE. A script file named ”UCI Datasets Preprocessing” is provided to download and preprocess all UCI dataset files used. We split each dataset randomly into a 80% $train$-dataset and a 20% $test$-dataset. Again, we repeat this operation $M = 100$ times to produce M pairs of *train/test* datasets.

news20 data (Svmlib) contains a $train$ dataset of size $N_{train} = 16087$ and a $test$ dataset with $N_{test} = 3308$ where the number of features is $P = 34580$. This dataset corresponds to a classification task that we turn into a regression problem with the label as the response variable.
In order to test our procedures in the setting $n \leq p$, we reduce the dimension of the problem by selecting the $p = 2875$ most important features (using PCA). For these selected features (from now on $p = 2875$ is fixed), we sample six new $train$-datasets (in bold in Table 2 in the main paper) of different sizes n from the initial `news20` $train$-dataset. Again, for each size n of dataset, we perform $M = 100$ repetitions of the sampling process to produce M $train$ datasets. We use the initial $test$-dataset where we keep only the p previously selected features to evaluate the performance of the different procedures.

Dataset name	n	p	Dataset name	n	p
airfoil-self -noise.dat	1503	5	communities.data	1994	122
yacht-hydro dynamics.data	308	6	slice-localization-data.csv	53500	379
slump-test-0.data	103	7	UJIndoor LocLAT	19937	465
slump-test-1.data	103	7	UJIndoor LocLON	19937	465
machine.data.data	209	7	**news20_a**	500	2875
Concrete -Data.xls	1030	8	**news20_b**	1000	2875
2014 and 2015 CSM dataset	231	9	**news20_c**	1500	2875
CASP.csv	45730	9	**news20_d**	2000	2875
transcoding-mesurment.tsv	68784	17	**news20_e**	2500	2875
Relation Network (Directed).data	53413	20	**news20_f**	2875	2875

Table 2: UCI datasets and **Svmlib datasets**.

3 Results

3.1 Synthetic data

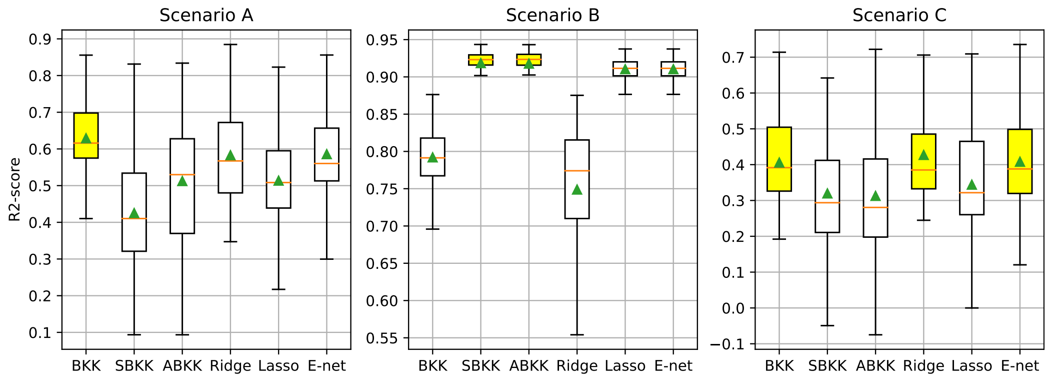

Figure 1: **Synthetic data**: R^2-score.

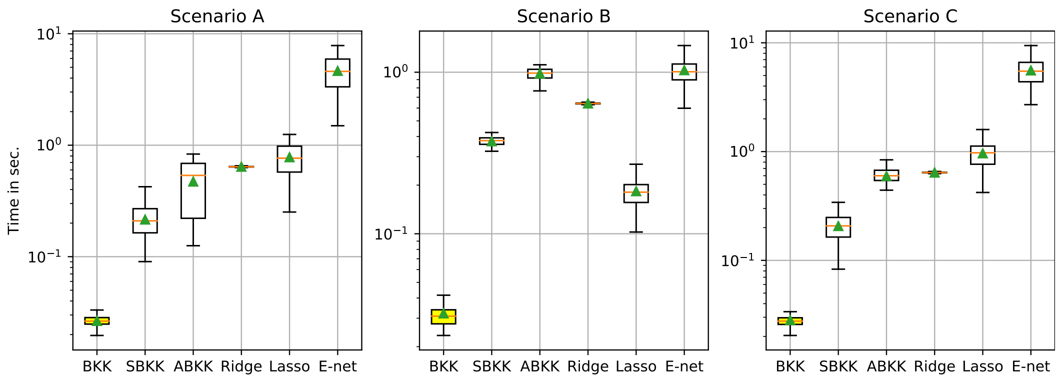

Figure 2: **Synthetic data**: running times in seconds.

Figures 1 and 2 contain the boxplots and empirical means (green triangle) of the $\mathbf{R^2}$-scores and computation times of each procedures for the synthetic datasets. The boxplot(s) highlighted in yellow correspond(s) to the best procedure(s) according to the Mann-Whitney (MW) test applied on the $\mathbf{R^2}$-scores and computation times.

For the synthetic data, even in cases most favorable to the benchmark procedures, our *BKKs* procedures attain equivalent to better performances. In **Scenario A**, *BKK* is the best procedure in terms of both $\mathbf{R}^2$-score and computation time according to the MW test. In **Scenario B**, $\mathcal{S}$*BKK* and $\mathcal{A}$*BKK* rank first in term of $\mathbf{R}^2$-score according to the MW test. All three *BKKs* procedures are computed in less than one second. In **Scenario C**, *BKK*, Ridge and Elastic-net are equivalent in term of $\mathbf{R}^2$-score according to MW test. However *BKK* is the fastest to compute according to MW test. Elastic-net is considerably slower.

The details of our experiments are summarized in the table below.

Scenario A		*BKK*	*SBKK*	*ABKK*	**Ridge**	**Lasso**	**Enet**
Time in sec.	**Mean**	**0.02**	0.21	0.47	0.64	0.78	4.64
	Median	0.02	0.20	0.53	0.64	0.76	4.59
	Std	0.00	0.07	0.23	0.00	0.27	1.64
	Rank	1	2	3	4	5	6
R^2**-score**	**Mean**	**0.62**	0.42	0.51	0.58	0.51	0.58
	Median	0.61	0.41	0.52	0.56	0.50	0.56
	Std	0.10	0.16	0.17	0.13	0.15	0.11
	Rank	1	6	4	2	4	2
Scenario B		*BKK*	*SBKK*	*ABKK*	**Ridge**	**Lasso**	**Enet**
Time in sec.	**Mean**	**0.03**	0.37	0.98	0.64	0.18	1.03
	Median	0.03	0.37	0.98	0.64	0.18	1.00
	Std	0.00	0.02	0.07	0.00	0.03	0.20
	Rank	1	3	5	4	2	5
R^2**-score**	**Mean**	0.79	**0.91**	**0.91**	0.74	0.91	0.91
	Median	0.79	0.92	0.92	0.77	0.91	0.91
	Std	0.04	0.02	0.02	0.08	0.01	0.01
	Rank	5	1	1	6	3	3
Scenario C		*BKK*	*SBKK*	*ABKK*	**Ridge**	**Lasso**	**Enet**
Time in sec.	**Mean**	**0.02**	0.20	0.59	0.64	0.96	5.57
	Median	0.02	0.20	0.59	0.64	0.97	5.48
	Std	0.00	0.05	0.11	0.00	0.26	1.55
	Rank	1	2	3	4	5	6
R^2**-score**	**Mean**	**0.40**	0.31	0.31	**0.42**	0.34	**0.40**
	Median	0.39	0.29	0.28	0.38	0.32	0.38
	Std	0.17	0.16	0.17	0.12	0.17	0.14
	Rank	1	4	6	1	4	1

Table 3: **Synthetic data**: mean, median, standard deviation and rank according to the MW test.

3.2 Real data

UCI dataset $n > p$ Figure 3 reveals that the *BKKs* are better than the benchmark procedures both in term of $\mathbf{R^2}$-scores and running times according to the MW test. The average speed-up is about 123 times for *BKK* over Ridge and goes up to 236 times for *BKK* against Elastic-net.

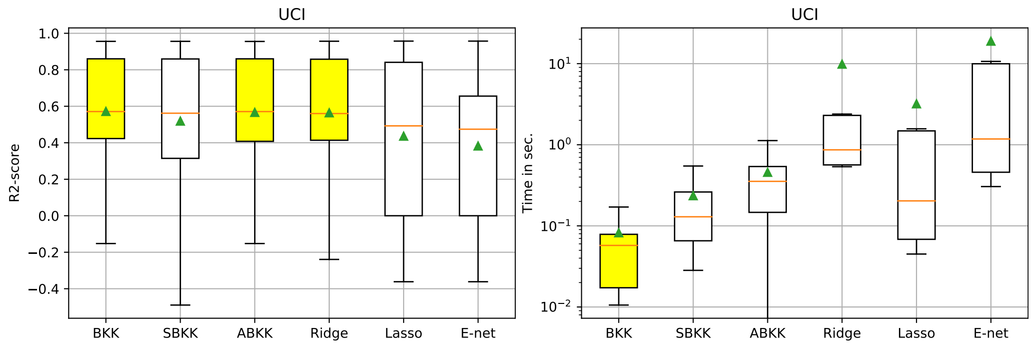

Figure 3: **UCI data**: R^2-score boxplots (left) and running times in seconds (right).

The details of our experiments are summarized in the table below.

UCI data		*BKK*	*SBKK*	*ABKK*	**Ridge**	**Lasso**	**E-net**
Time in sec.	**Mean**	**0.08**	0.23	0.45	9.87	3.20	18.9
	Median	0.05	0.12	0.35	0.86	0.20	1.17
	Std	0.09	0.30	0.48	18.1	6.03	35.3
	Rank	1	2	3	5	4	6
R^2**-score**	**Mean**	**0.57**	0.51	0.56	0.56	0.43	0.38
	Median	0.57	0.56	0.57	0.56	0.49	0.47
	Std	0.28	0.40	0.29	0.28	0.37	0.37
	Rank	1	4	1	1	5	6

Table 4: **UCI data**: mean, median, standard deviation and rank according to the MW test.

20news dataset $n \leq p$ Our methods are always significantly faster to compute for a $\mathbf{R^2}$- score always within 0.05 of the best. An average speed-up up to 20 times is achieved for $n \leq p$. See Table 5 and Figure 4.

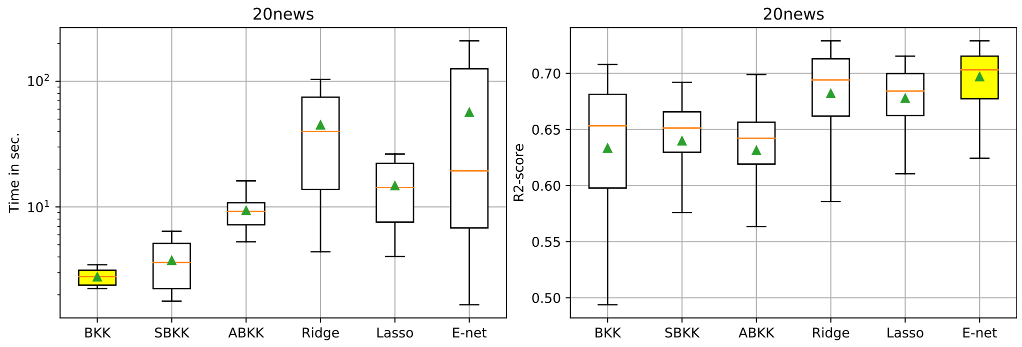

Figure 4: **20news data**: R^2-score boxplots (left) and running times in seconds (right).

The details of our experiments are summarized in the table below.

20news data		*BKK*	*SBKK*	*ABKK*	**Ridge**	**Lasso**	**E-net**
Time in sec.	**Mean**	**2.77**	3.75	9.37	44.9	14.7	56.5
	Median	2.79	3.61	9.22	39.8	14.2	19.3
	Std	0.40	1.38	2.59	33.2	7.66	65.9
	Rank	1	2	3	5	4	6
R^2**-score**	**Mean**	0.63	0.63	0.63	0.68	0.67	**0.69**
	Median	0.65	0.65	0.64	0.69	0.68	0.70
	Std	0.06	0.03	0.03	0.03	0.02	0.02
	Rank	4	4	6	2	3	1

Table 5: **20news data**: mean, median, standard deviation and rank according to the MW test.

3.3 Running time

We observe in Figures 5 and 6 that the running times of the *BKKs* procedures increase far more slowly w.r.t. p for UCI and n for 20news (since p is fixed for 20news) than the benchmark procedures for both the UCI and 20news datasets.
On the left graph, p (max dimension $= 465$) seems to have a little impact on the *BKKs* procedures as the running time remains below 1 second. Indeed, inverting a matrix is not an expensive operation in PyTorch as long as the matrix can be stored into the GPU memory. Even for $p = 2875$ for the 20news dataset, the running time is approximately 10 seconds or less on average as compared to 56.5 seconds for Elastic-net.

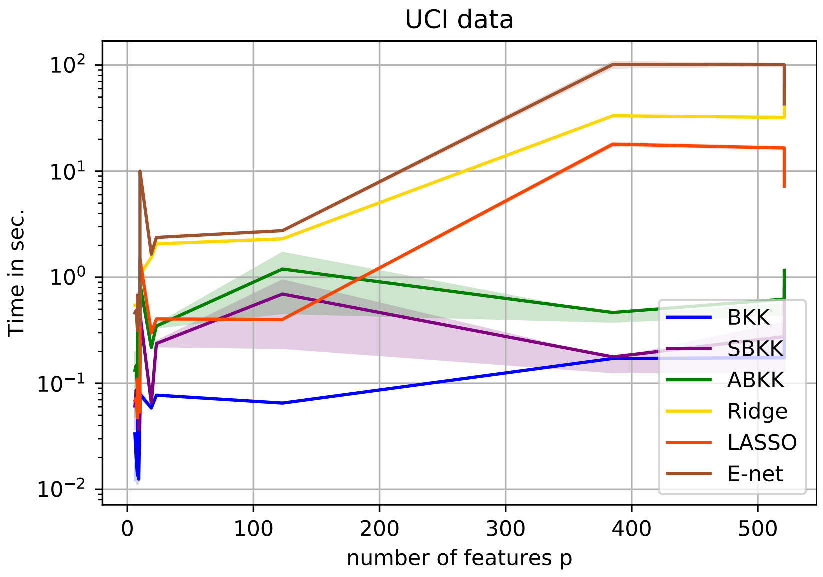

Figure 5: **UCI data:** Running time as a function of p

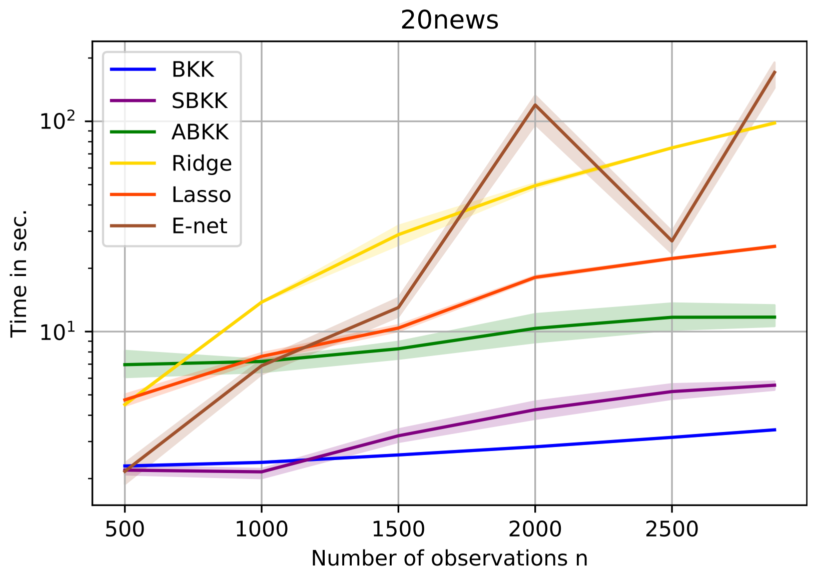

Figure 6: **20news data:** Running time as a function of n

3.4 Number of Iterations

In all experiments, the number of iterations required for the convergence of our *BKKs* procedures is about a few dozen (see Figure 7).

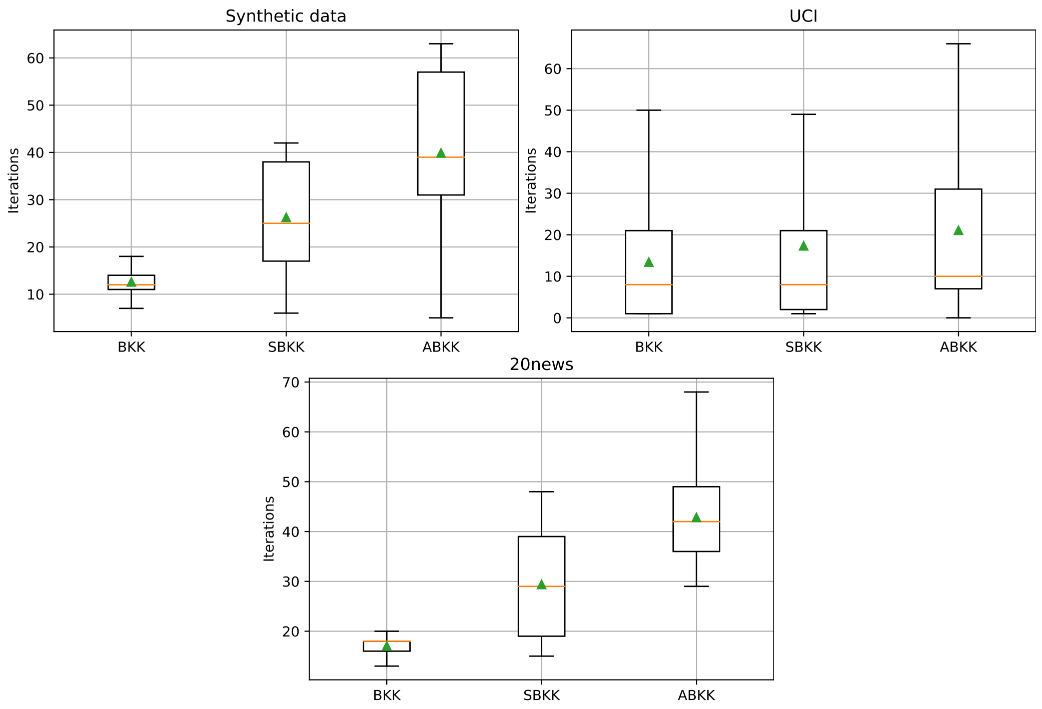

Figure 7: **All data:** Number of iterations

The details of our experiments are summarized in the table below.

		Scenario A	**Scenario B**	**Scenario C**	**UCI**	**20news**
BKK	**Mean**	12.0	14.6	**11.1**	13.4	17.0
	Median	12	13	11	8	18
	Std	2.35	4.76	3.46	14.9	1.93
$\mathcal{S}$*BKK*	**Mean**	20.8	38.5	19.3	**17.3**	29.3
	Median	21	39	19	8	29
	Std	7.63	2.11	5.83	30.7	9.74
$\mathcal{A}$*BKK*	**Mean**	27.9	58.1	33.5	**21.0**	42.8
	Median	33	58	34	10	42
	Std	14.6	2.45	6.69	28.4	8.51

Table 6: **Number of Iterations**: mean, median and standard deviation.

3.5 Impact of parameter T

Figure 8 plots the impact of parameter T on the performances of all the *BKKs* procedures on the synthetic data.

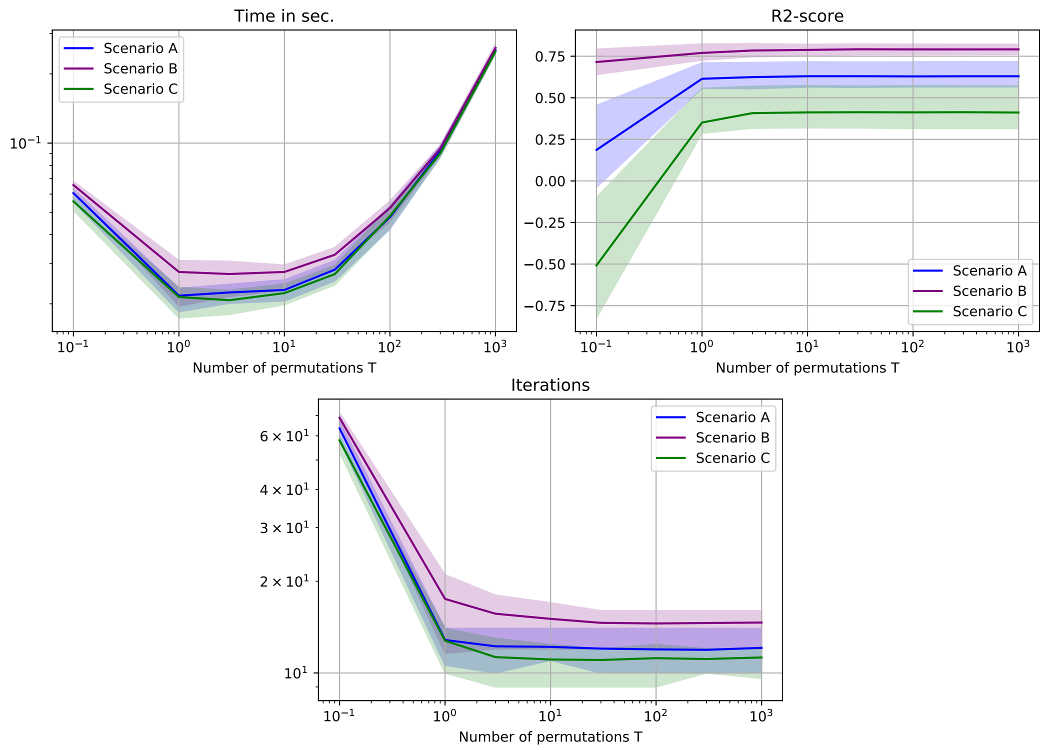

Figure 8: **Synthetic data:** impact of T on the *BKKs* procedures.

4 Loss functions

By applying the underlying idea of *G-ER* within the framework of the linear model, we have developed a new framework where several new criteria and procedures have been derived. In the same way, other frameworks can easily be constructed by choosing the criterion judiciously. As proof, we can easily emulate the *G-ER* idea to build other pertinent frameworks based on the criteria below

$$\mathrm{BKK}^{sq}_{\beta}(\theta) = -\sqrt{\frac{1}{T}\sum_{t=1}^{T}\left(\|\mathbf{Y}-\mathbf{X}\beta(\theta,\mathbf{X},\mathbf{Y})\|_2 - \|\pi^t(\mathbf{Y})-\mathbf{X}\beta(\theta,\mathbf{X},\pi^t(\mathbf{Y}))\|_2\right)^2},$$

$$\mathrm{BKK}^{ln}_{\beta}(\theta) = -\ln\left(1+\frac{1}{T}\sum_{t=1}^{T}\left|\|\mathbf{Y}-\mathbf{X}\beta(\theta,\mathbf{X},\mathbf{Y})\|_2 - \|\pi^t(\mathbf{Y})-\mathbf{X}\beta(\theta,\mathbf{X},\pi^t(\mathbf{Y}))\|_2\right|\right)$$

These criteria have in common that they are not too oscillating. The convexity of these criteria will depend on the choice of the closed form family of estimators $\{\beta_\theta\}_\theta$. In our experiments, we observed equivalent results for these other criteria.

References

[1] Arthur Asuncion and David Newman. Uci machine learning repository, 2007.

[2] Chih-Chung Chang and Chih-Jen Lin. Libsvm: A library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):1–27, 2011.

[3] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.