Improving Covariance Matrices using Machine Learning

Natalí Soler Matubaro de Santi
natalidesanti@gmail.com

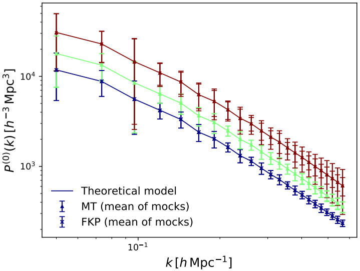
Advisor: Prof. Dr. Luis Raul Weber Abramo

Mathematical Physics Department
University of São Paulo

December 17, 2020

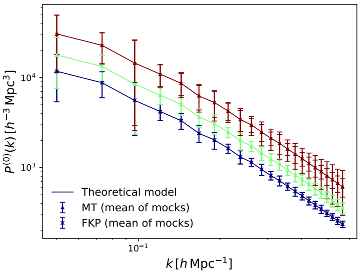
Motivation: Why improving covariance matrices?

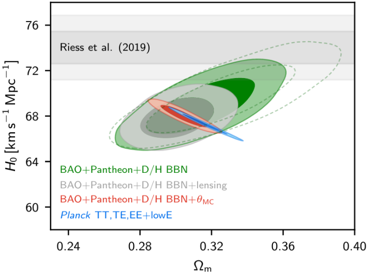
- They reflect the propagation of the statistical errors:



Motivation: Why improving covariance matrices?

- They reflect the propagation of the statistical errors:

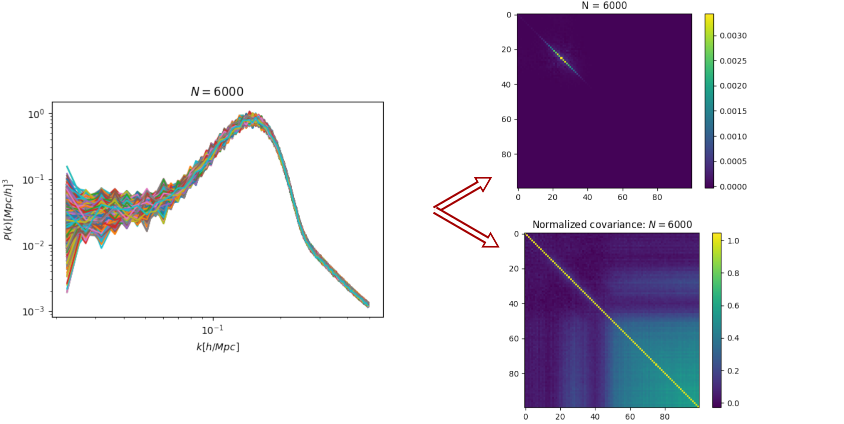




Planck 2018 results: VI Cosmological parameters.

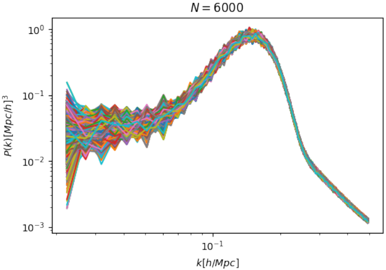
Problem

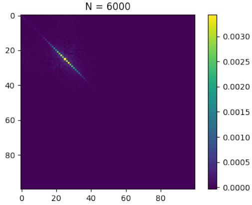
- To represent the true statistical errors we need a lot of data to build these matrices:

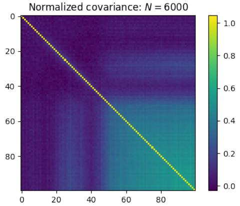


Problem

- To represent the true statistical errors we need a lot of data to build these matrices:

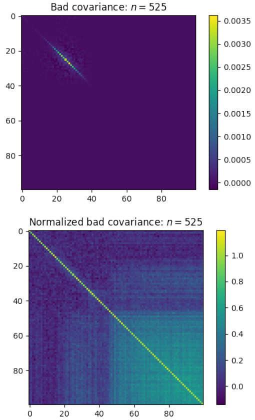


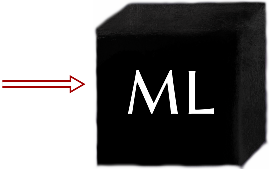




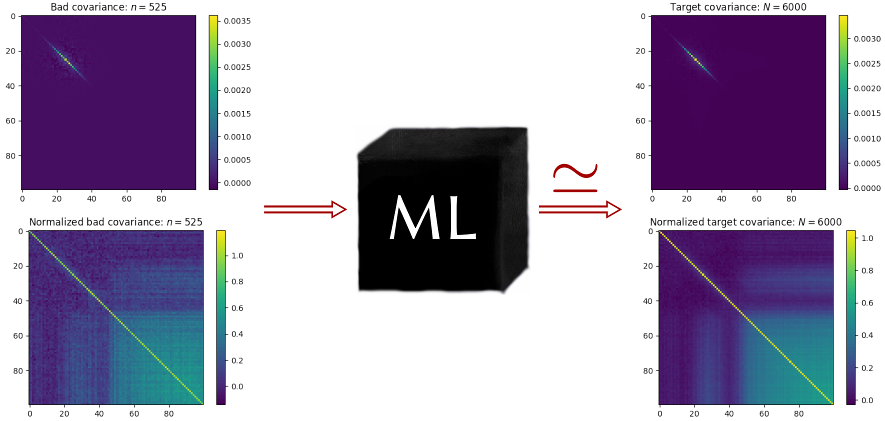
But we can not always do it in practice...

Proposed solution

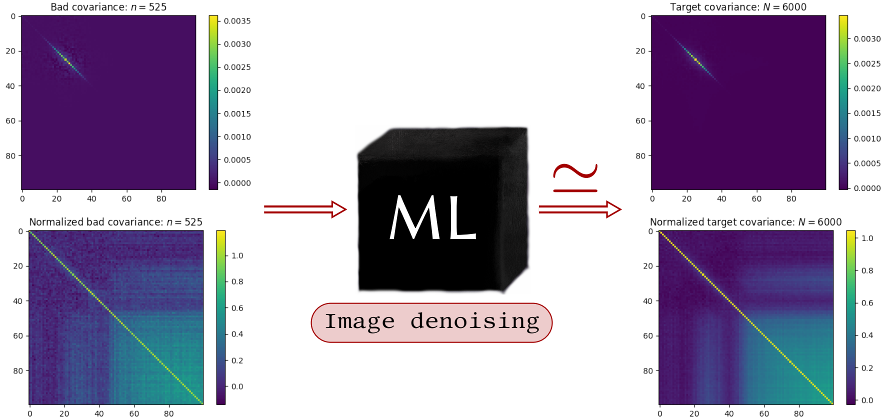




Proposed solution



Proposed solution



Methodology - Toy project simulation

- The simulations followed the theoretical function:

$$P(k) = P_0 \exp\left[-\frac{(k-k_0)^2}{2\sigma_0^2}\right]$$

	Value
P_0	$1 Mpc/h$
k_0	$0.15 h/Mpc$
σ_0	$0.03 h/Mpc$

Methodology - Toy project simulation

- The simulations followed the theoretical function:

$$P(k) = P_0 \exp\left[-\frac{(k-k_0)^2}{2\sigma_0^2}\right]$$

	Value
P_0	$1Mpc/h$
k_0	$0.15h/Mpc$
σ_0	$0.03h/Mpc$

- We trained different **denoisers** using, each time:
  - `input_train`: **bad cov. matrices** (hundreds of spectra) $\Rightarrow n$;
  - `target_train`: **good cov. matrices** $\Rightarrow N = 6000$;

Methodology - Toy project simulation

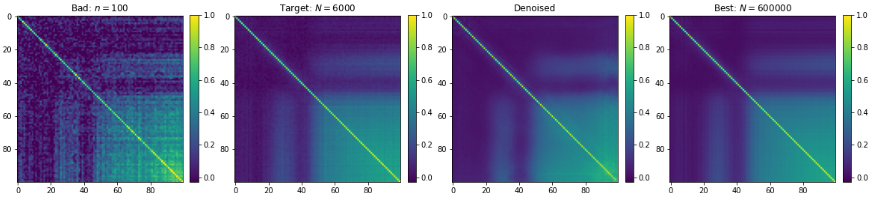
- The simulations followed the theoretical function:

$$P(k) = P_0 \exp\left[-\frac{(k-k_0)^2}{2\sigma_0^2}\right]$$

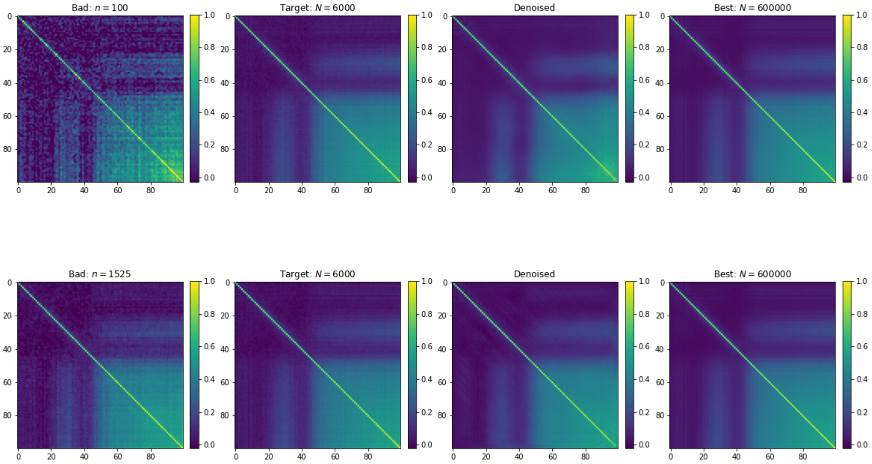
	Value
P_0	$1Mpc/h$
k_0	$0.15h/Mpc$
σ_0	$0.03h/Mpc$

- We trained different **denoisers** using, each time:
  - `input_train`: **bad cov. matrices** (hundreds of spectra) $\Rightarrow n$;
  - `target_train`: **good cov. matrices** $\Rightarrow N = 6000$;
- Then, we test the each **denoiser** with:
  - `input_test`: **bad cov. matrices** (hundreds of spectra) $\Rightarrow n$.

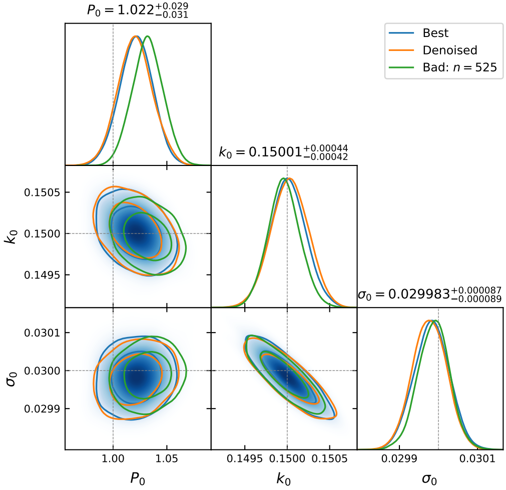
Results - Visual Results



Results - Visual Results

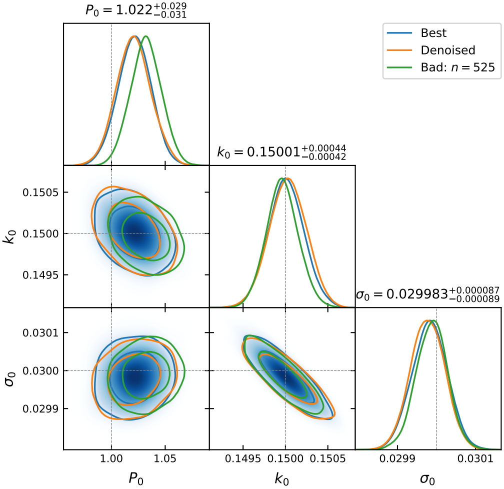


Results - Markov Monte Carlo Chain (MCMC)[1]: Recovering Parameters



[1] Ivezić, Z. et al., Statistics, Data Mining, and Machine Learning in Astronomy: A Practical Python Guide for the Analysis of Survey Data, 2014.

Results - Markov Monte Carlo Chain (MCMC)[1]: Recovering Parameters



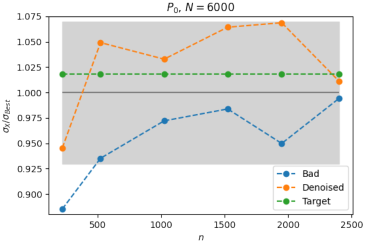
$$\frac{\Delta\mu}{\sigma_{Best}} = \frac{\mu_X - \mu_{Best}}{\sigma_{Best}}$$

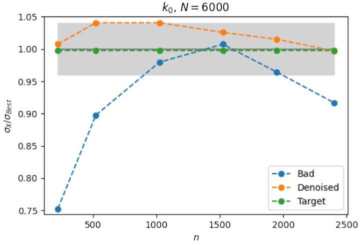
$\Delta\mu/\sigma_{Best}$	**Bad**	**Denoised**
P_0	0.685	0.054
k_0	0.218	0.092
σ_0	0.118	0.120

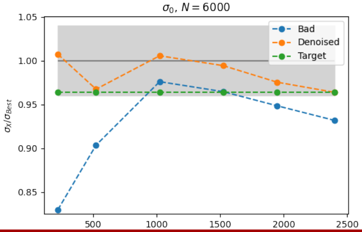
[1] Ivezić, Z. et al., Statistics, Data Mining, and Machine Learning in Astronomy: A Practical Python Guide for the Analysis of Survey Data, 2014.

Results - Sigma fraction: σ_X/σ_{Best}

- Varying the number of spectra n in the bad/input covariance matrices;
- Same number of spectra N in the good/target covariance matrices;

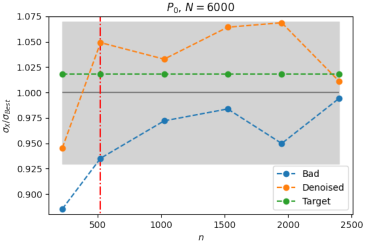


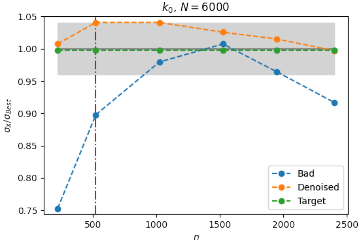


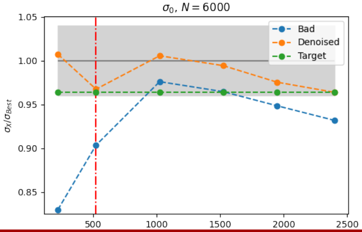


Results - Sigma fraction: σ_X/σ_{Best}

- Varying the number of spectra n in the bad/input covariance matrices;
- Same number of spectra N in the good/target covariance matrices;







Conclusions and Next Steps

- We have achieved great results using **image denoising** techniques to improve the *covariance matrices*;

Conclusions and Next Steps

- We have achieved great results using **image denoising** techniques to improve the *covariance matrices*;
- We showed that even with a **low** number of simulations, we can achieve the same results as a **higher** number of them;

Conclusions and Next Steps

- We have achieved great results using **image denoising** techniques to improve the *covariance matrices*;
- We showed that even with a **low** number of simulations, we can achieve the same results as a **higher** number of them;
- Once all this work is a really controlled "toy project", we want to apply the same method in **realistic simulations** (e.g., `ExSHalos`, `LogNormals`, `N-body`).