Studio 07 - Robust Regression with Heavy-Tailed Errors

Groups: work in teams of 2-4.

1. Objective

The goal of today's lab is to **better understand the effect of heavy-tailed errors on the performance of linear regression estimators** in high-dimensional settings. You will implement and analyze a simulation study to compare several estimators of the regression coefficients, varying factors such as tail heaviness, signal strength, dimensionality and design matrix structure.

---

2. Experimental setup

Consider the following linear model

$$y = X\beta + \varepsilon, \quad \varepsilon_i \overset{\text{iid}}{\sim} F,$$

where F is some univariate error distribution, $X \in \mathbb{R}^{n \times p}$ and $\beta \in \mathbb{R}^p$.

Parameters to vary

Parameter	Meaning	Examples / Suggestions
Tail heaviness	Tails of error distribution F.	Student-t ($\text{df} = 1, 2, 3, 20, +\infty$)
Aspect ratio	$\gamma = p/n$ controls the dimensionality regime.	Try $\gamma \in \{0.2, 0.5, 0.8\}$.
Design structure	Correlation among predictors.	Autoregressive: ($\text{Cov}(X_j, X_k) = \rho^{\|j-k\|}$).
Signal-to-noise ratio	Overall signal strength.	Vary by scaling β or noise variance so that $\text{SNR} = \frac{\beta^\top X' X \beta}{\sigma^2} \in \{1, 5, 10\}$.

---

3. Estimators to Compare

Estimate β using the following methods:

Method	Implementation	Description
Ordinary Least Squares (OLS)	`sklearn.linear_model.LinearRegression()`	Sensitive to outliers and heavy tails.
Least Absolute Deviation (LAD)	`sklearn.linear_model.QuantileRegressor(alpha=0.)`	Minimizes the sum of absolute residuals; robust to outliers.
Huber Regression	`sklearn.linear_model.HuberRegressor()`	Interpolates between OLS and LAD.

---

4. Evaluation Metric

For each scenario and each method, estimate the mean-squared error (MSE) $\mathbb{E}\|\hat{\beta} - \beta\|^2$.

---

5. Implementation Guidelines

- **Divide tasks:** Within your group, assign roles for simulation generation, estimator fitting, and analysis/visualization.

- **Design a results schema:** Agree on a shared data contract (e.g. a `pandas.DataFrame` where each row corresponds to one simulation run, including columns like `method`, `n`, `p`, `df`, `rho`, `SNR`, `rep`, `mse`).
- **Write modular code:** You do *not* need to use classes, but functions should be modular (e.g. `generate_data()`, `fit_model()`, `evaluate()`).
- **Store reproducible results:** Use consistent random seeds, and save results to CSV or pickle file for analysis.

---

6. Analysis/Visualization

Create at least one publication-quality figures summarizing your findings. Each figure should be:

- Appropriately sized (roughly 4-6" wide, 2.5-4" tall)
- Labeled with readable fonts and clear legends
- Styled for clarity and professionalism: avoid unnecessary gridlines, use consistent color palettes. Generally question all of the defaults of matplotlib.

Suggested plot:

- MSE versus degrees of freedom (t-distribution) for each method
- Small multiples varying other factors (e.g. SNR and aspect ratio)

---

7. Deliverables

Each group should submit the full pipeline, along with generated figure(s),