Regression

Manoel Horta Ribeiro
manoel@cs.princeton.edu

humans & machines lab

COS 598D / Spring 2026

Game Plan!

- **This week**:
  - More advanced topics on regression
  - Some time at the end for group formation
  - **This Friday EOD**: Homework #2 is released

- **Next week**: Spring break!

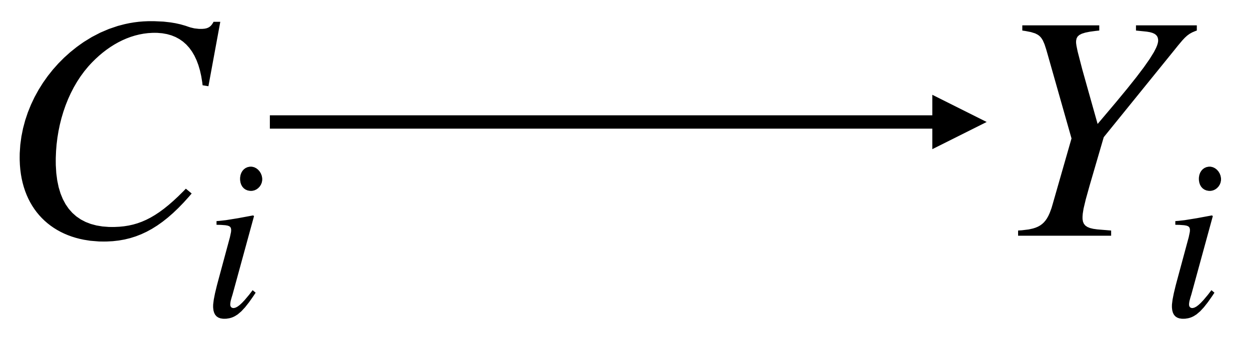
- **Next module**: Advanced topics!
  - *Guests come in person (Mar 19/April 09)*, class is a guest presentation plus open discussion!
  - *Guests present virtually (Mar 26/April 02)*, class is a guest presentation plus open discussion!

Olawale Salaudeen

Recap: Univariate OLS

- We care about how varying questions difficulty level changes correctness.
- We observe $\langle Y_i, C_i \rangle \; i = 1, \ldots, n$ where $Y_i \in \{0,1\}$ and $C_i \in \mathbb{R}$
- We assume the model:

$$Y_i = \beta_0 + \beta_1 C_i + \epsilon_i$$



Univariate OLS: Variance

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n} (C_i - \bar{C})(Y_i - \bar{Y})}{\sum_{i=1}^{n} (C_i - \bar{C})^2}$$

- The residuals are: $\hat{\varepsilon}_i = Y_i - \hat{\beta}_0 - \hat{\beta}_1 C_i$
- The traditional way to calculate $\text{Var}(\hat{\beta}_1 \mid C_1, \ldots, C_n)$
  - Assume Var$(\epsilon_i | C_i) = \sigma^2$
  - You can show with some algebra that:

$$C_i \longrightarrow Y_i$$

$$\text{Var}(\hat{\beta}_1 \mid C_1, \ldots, C_n) = \frac{\sigma^2}{\sum_{i=1}^{n} (C_i - \bar{C})^2} = \frac{\frac{1}{n-2} \sum_{i=1}^{n} \hat{\varepsilon}_i^2}{\sum_{i=1}^{n} (C_i - \bar{C})^2}$$

- We'll use this variance to do the Z tests!

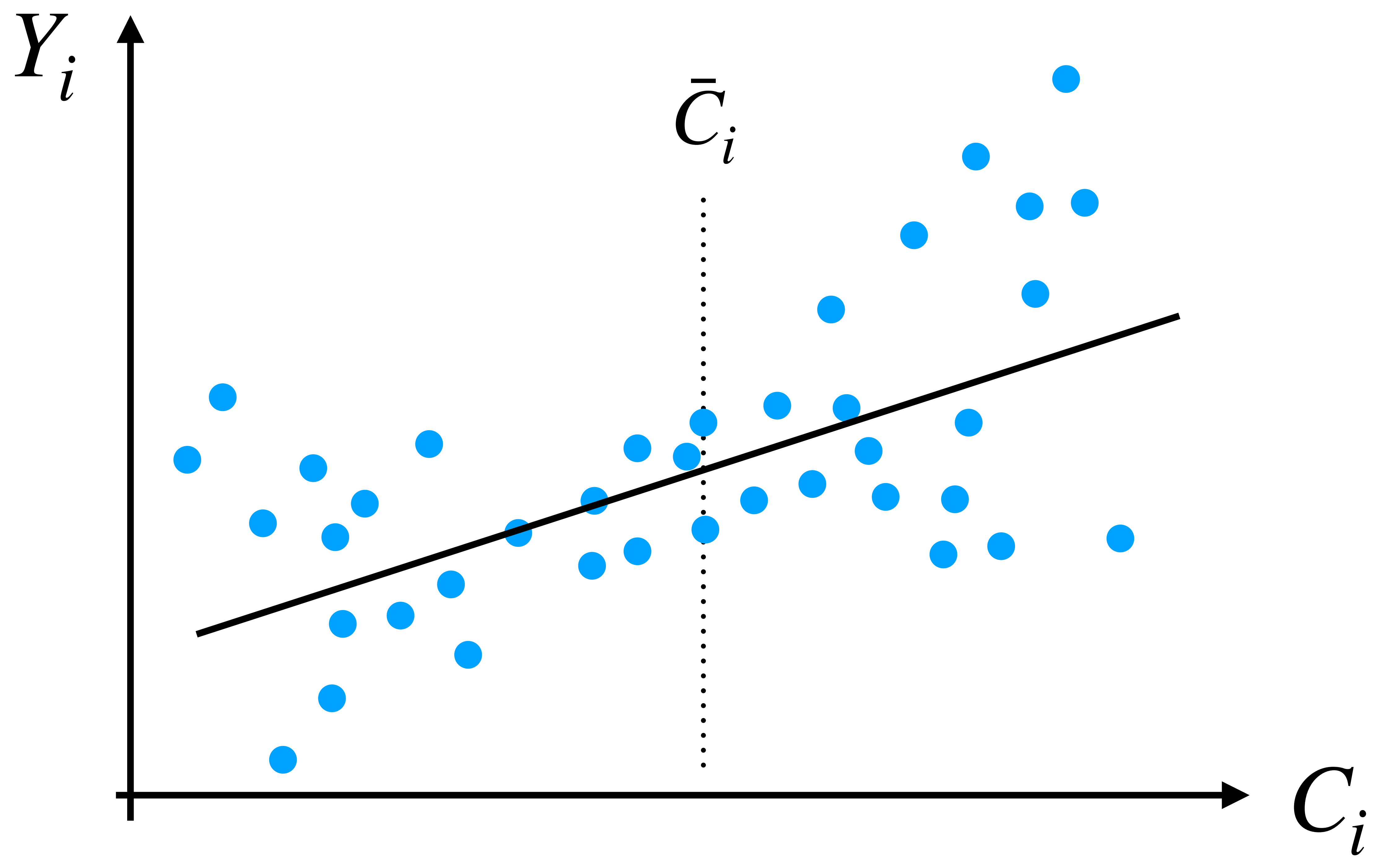
Reality kicks in: Heteroskedasticity

- Var$(\epsilon_i \mid C_i) \neq \sigma^2$
- **CoT example**: the variance in problems is bigger for challenging problems! Not all units have the same variance!
- Under heteroskedasticity, the usual SE estimator is biased/inconsistent;

$$C_i \longrightarrow Y_i$$

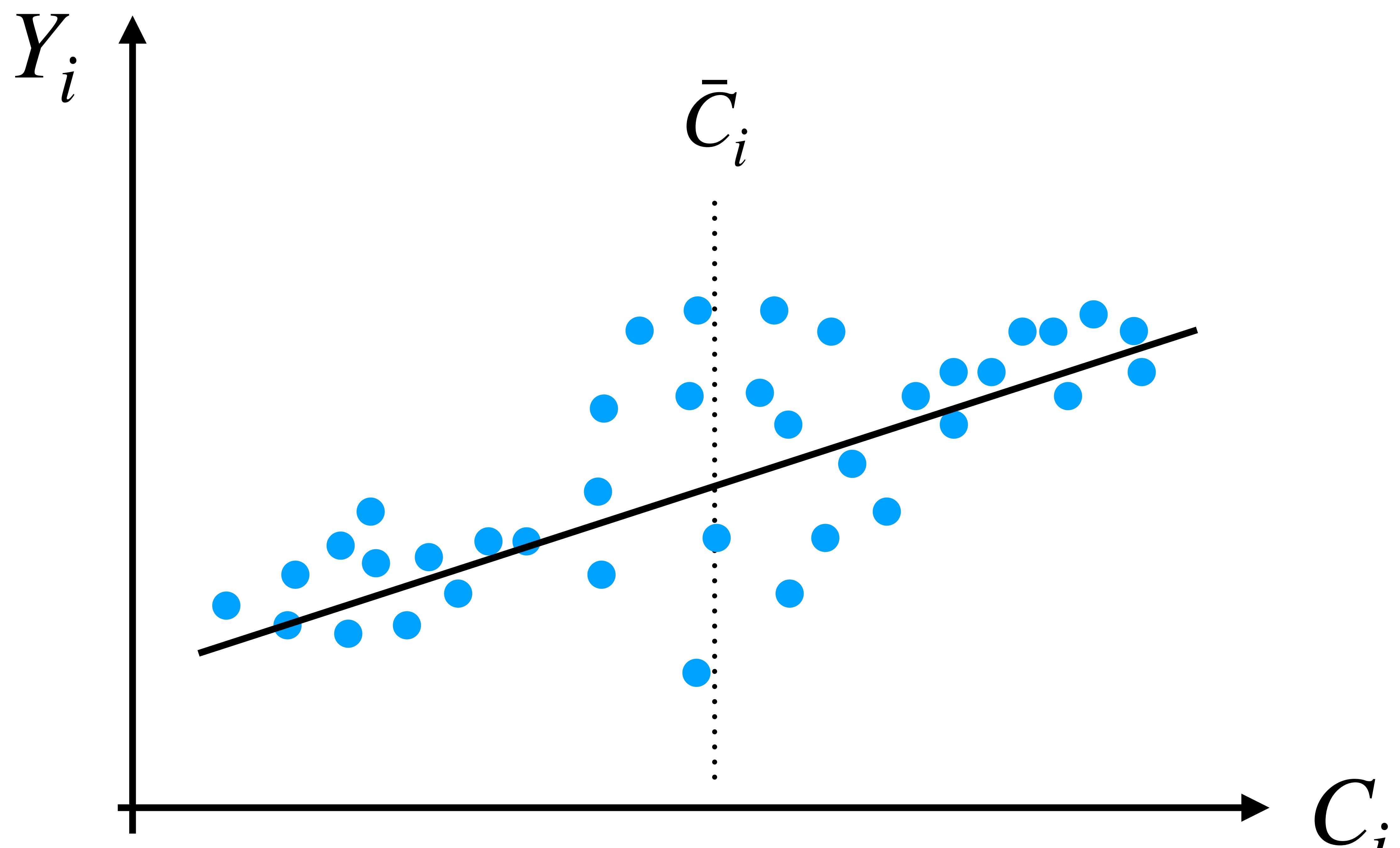
Intuition

- Low noise near $\bar{C}_i$
- High noise far from $\bar{C}_i$
- This makes the slope vary *more* than if the noise were equally distributed throughout the sample!



Intuition

- High noise near $\bar{C}_i$
- Low noise far from $\bar{C}_i$
- This makes the slope vary *less* than if the noise were equally distributed throughout the sample!



Solution: Robust SEs

- Allow variance to vary with i

$$\text{Var}(\epsilon_i \mid C_i) = \sigma_i^2$$

- We can estimate:

$$\text{Var}(\hat{\beta}_1 \mid C_1, \ldots, C_n) = \frac{\sum_{i=1}^{n} (C_i - \bar{C})^2 \, \hat{\epsilon}_i^2}{\left(\sum_{i=1}^{n} (C_i - \bar{C})^2 \right)^2},$$

- **Classical SE**: Use the average squared residual as the noise level.
- **Robust SE**: Each squared residual is its own noise level (weighted by extremity)

Reality kicks in: Correlated Errors

- $\text{Cov}(\epsilon_{pi}, \epsilon_{pj} \mid T_i) \neq 0$ within problem p
- **CoT example**: we have multiple runs per problem!
- If we ignore this and assume i.i.d. errors, standard errors are too optimistic.
- Traditional SE treats your dataset as if it has way more independent variation than it has!

$$T_i \longrightarrow Y_i$$

Intuition

- Suppose that for a problem p and for a run r, we have:

$$Y_{pr} = \beta_1 + \beta_2 T_{pr} + \overbrace{u_p + v_{pr}}^{\epsilon_{pr}}$$

- u_p is a problem-specific shock shared by all runs in p
- v_{pr} is idiosyncratic run noise (sampling, decoding randomness).
- The outcomes of runs r_1 and r_2 are correlated within problem p

$$\text{Cov}(Y_{pr_1}, Y_{pr_2}) = \text{Var}(u_p) \;>\; 0$$

Intuition

- $Y_{pr} = \beta_1 + \beta_2 T_{pr} + \overbrace{u_p + v_{pr}}^{\epsilon_{pr}}$

- Increasing the number of runs without increasing the number of clusters can only help reduce uncertainty on v_{pr}, but not on u_p!

- E.g., if you have thousands of runs for the same problem, you cannot know how much variability there is on the problem-level!

- Your traditional standard error ignores this variability altogether!

Solution: Cluster Robust SEs

- We can estimate:

$$\mathrm{Var}(\hat{\beta}_1 \mid T_1, \ldots, T_n) = \frac{\sum_{g=1}^{G} \left(\sum_{i \in g} (T_i - \bar{T})\, \hat{\varepsilon}_i \right)^2}{\left(\sum_{i=1}^{n} (T_i - \bar{T})^2 \right)^2}$$

- **Classical SE**: Treats each observation as independent noise.
- **Robust SE**: treats each *cluster* as one "independent unit."

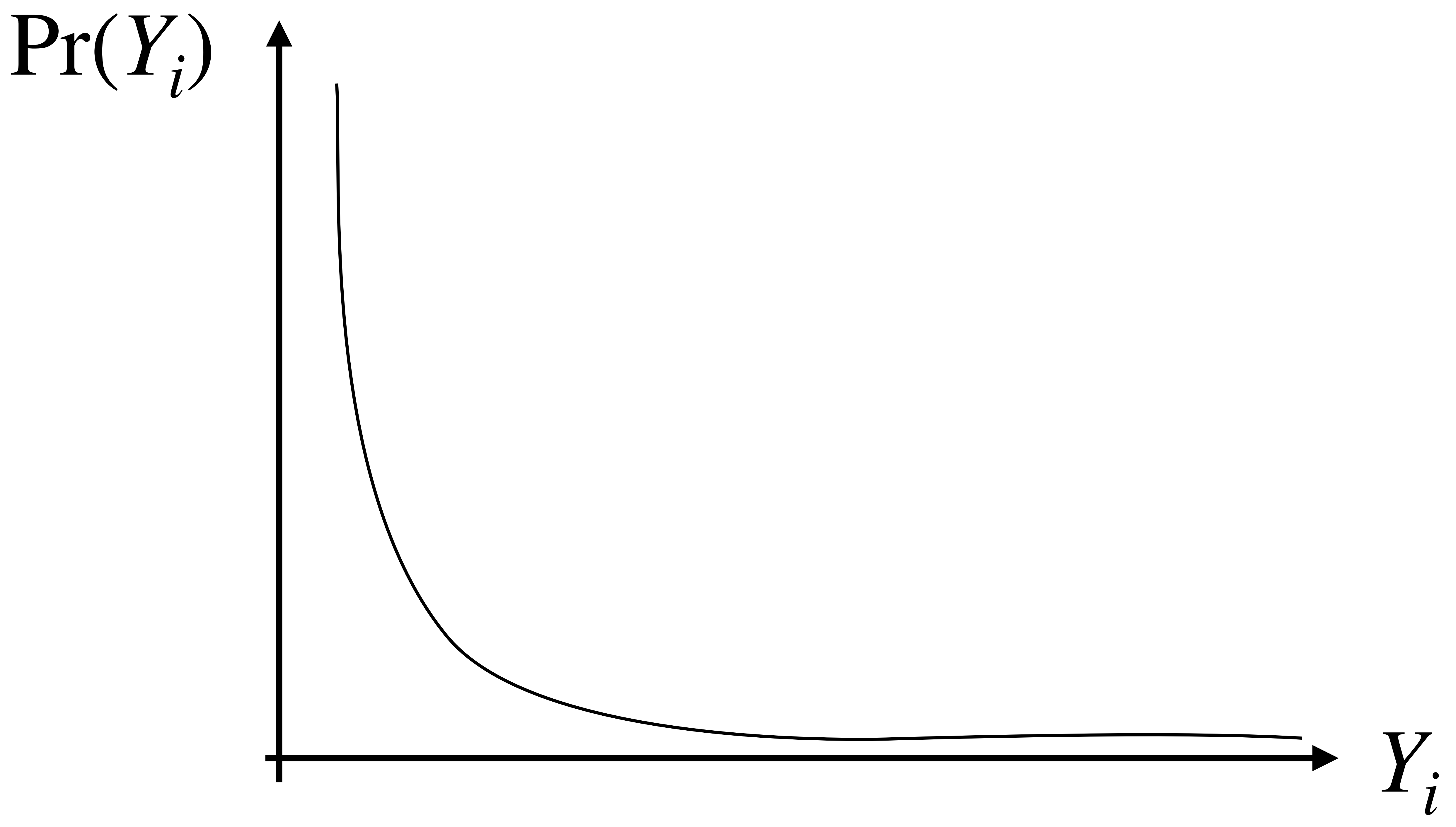
Let’s change the running example

Impact of GenAI on Coding

We are interested in estimating the effect of GenAI (T_i) on the number of *issues* (Y_i) opened on GitHub projects (as a proxy for code quality).

$$Y_i = \beta_0 + \beta_1 T_i + \epsilon_i$$

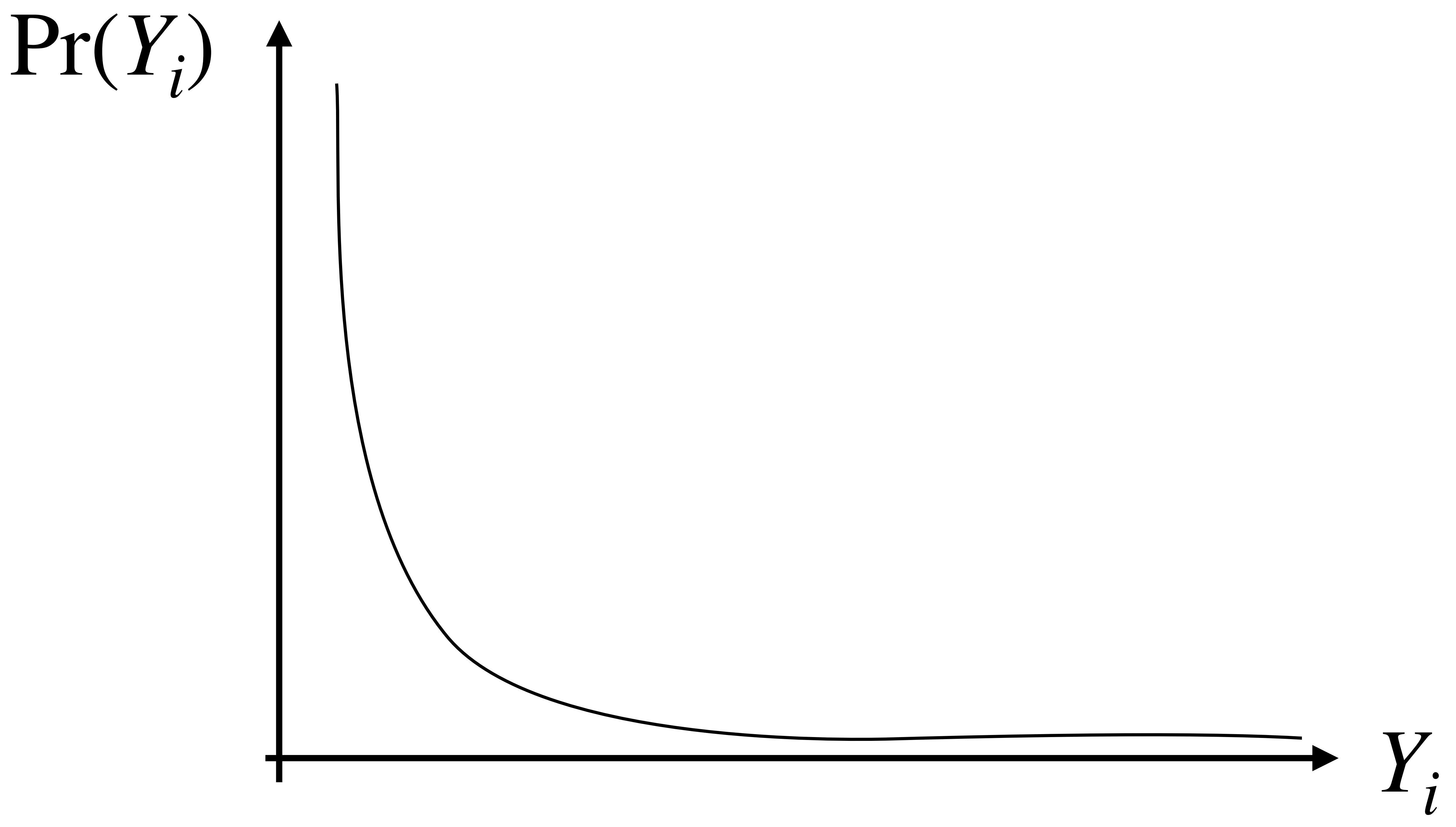
Problem: Data is very skewed, a minority of projects have the majority of Issues!



Impact of GenAI on Coding

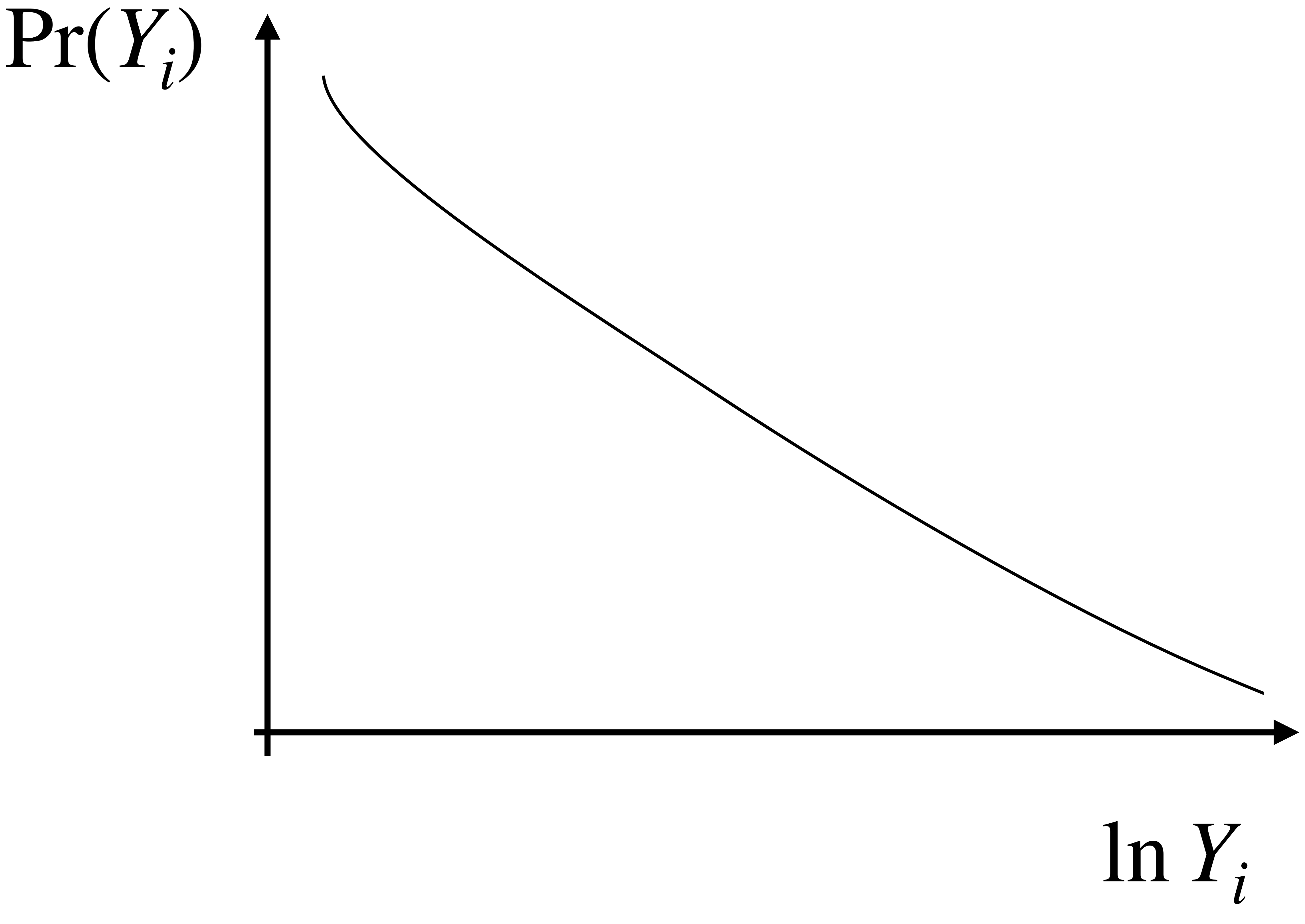
OLS is still "fine," but interpretation gets finicky…

Results get dominated by the outliers; they mean little for the average GitHub project.



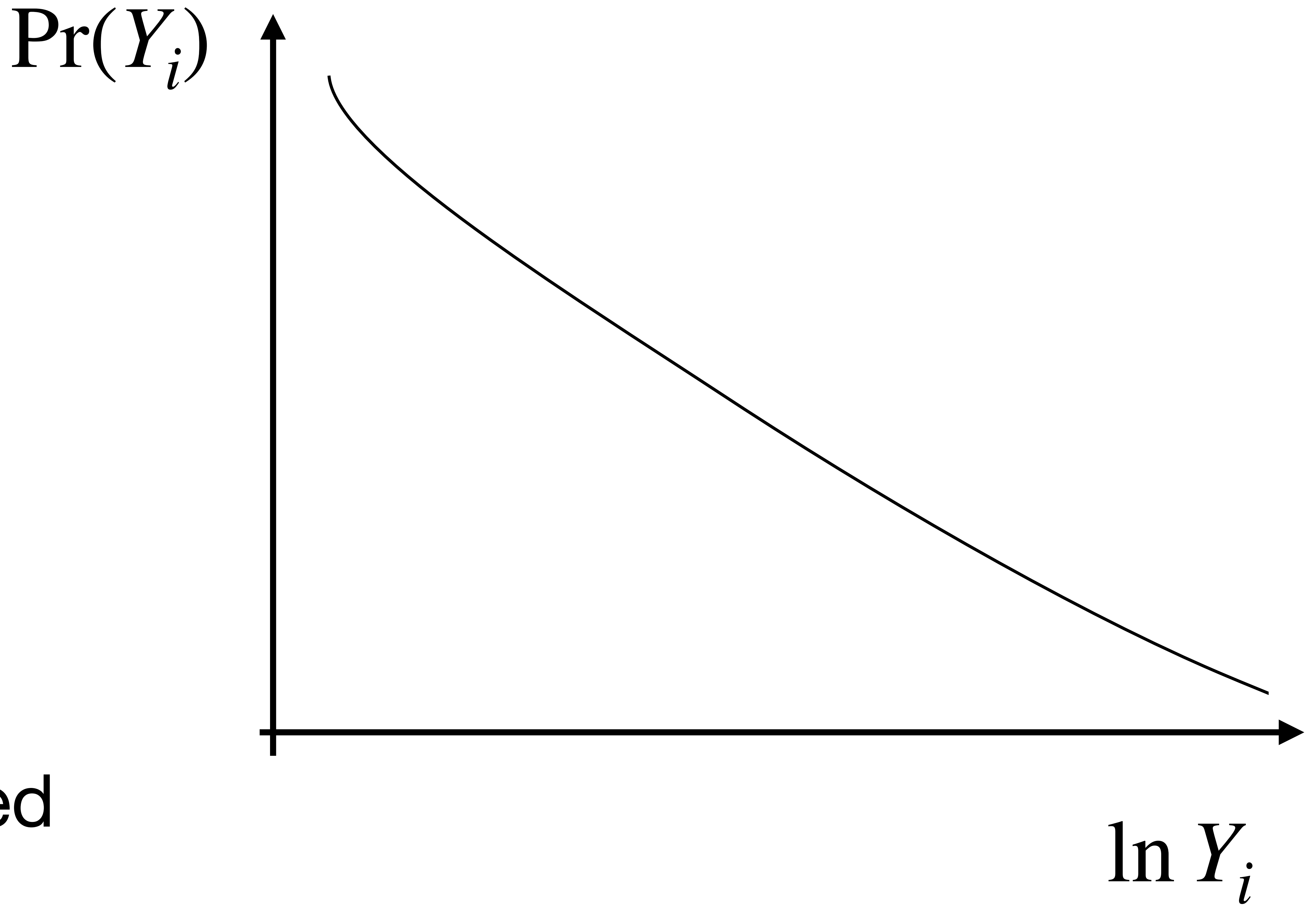
Solution: “Log” your outcome!

- $\ln Y_i = \beta_0 + \beta_1 T_i + \epsilon_i$
- $e^{\ln Y_i} = e^{\beta_0 + \beta_1 T_i + \epsilon_i}$
- $Y_i = e^{\beta_0} e^{\beta_1 T_i} e^{\epsilon_i}$
- $\dfrac{E[Y_i \mid T_i = 1]}{E[Y_i \mid T_i = 0]} = \dfrac{e^{\beta_0} e^{\beta_1} e^{\epsilon_i}}{e^{\beta_0} e^{\epsilon_i}} = e^{\beta_1}$
- $\beta_1 = \ln \dfrac{E[Y_i \mid T_i = 1]}{E[Y_i \mid T_i = 0]}$



Solution: "Log" your outcome!

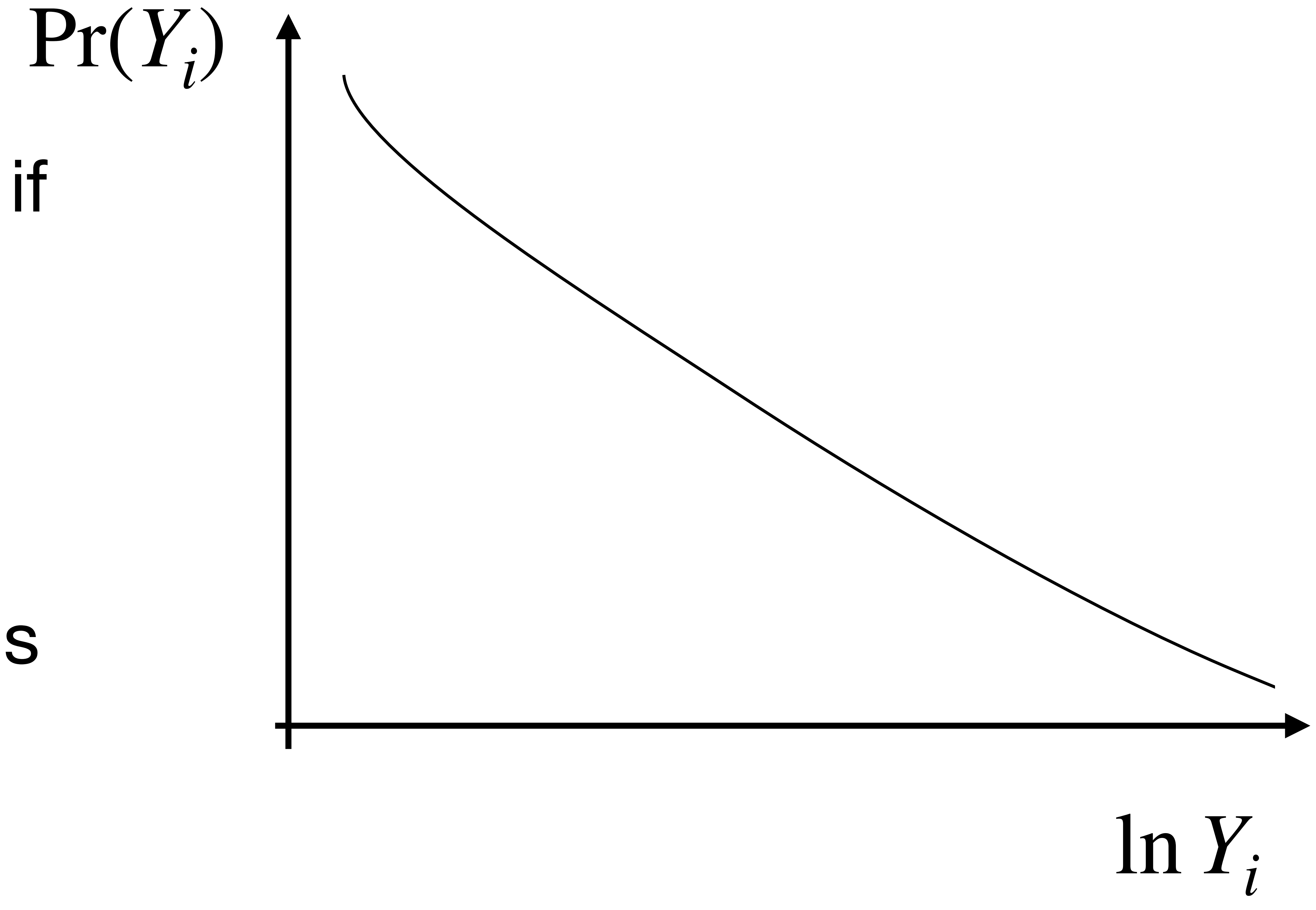
- $\frac{E[Y_i | T_i = 1]}{E[Y_i | T_i = 0]} = e^{\beta_1}$
- $E[Y_i | T_i = 1] = e^{\beta_1} E[Y_i | T_i = 0]$
- The treatment multiplies the baseline by a constant factor!
  - If $\beta_1 = 0.1$, $e^{0.1} \approx 1.105$: treated outcomes are about 10.5% higher!



Solution: “Log” your outcome!

- **Problem**: $\ln 0$ is undefined! What if you have GitHub projects with 0 issues?

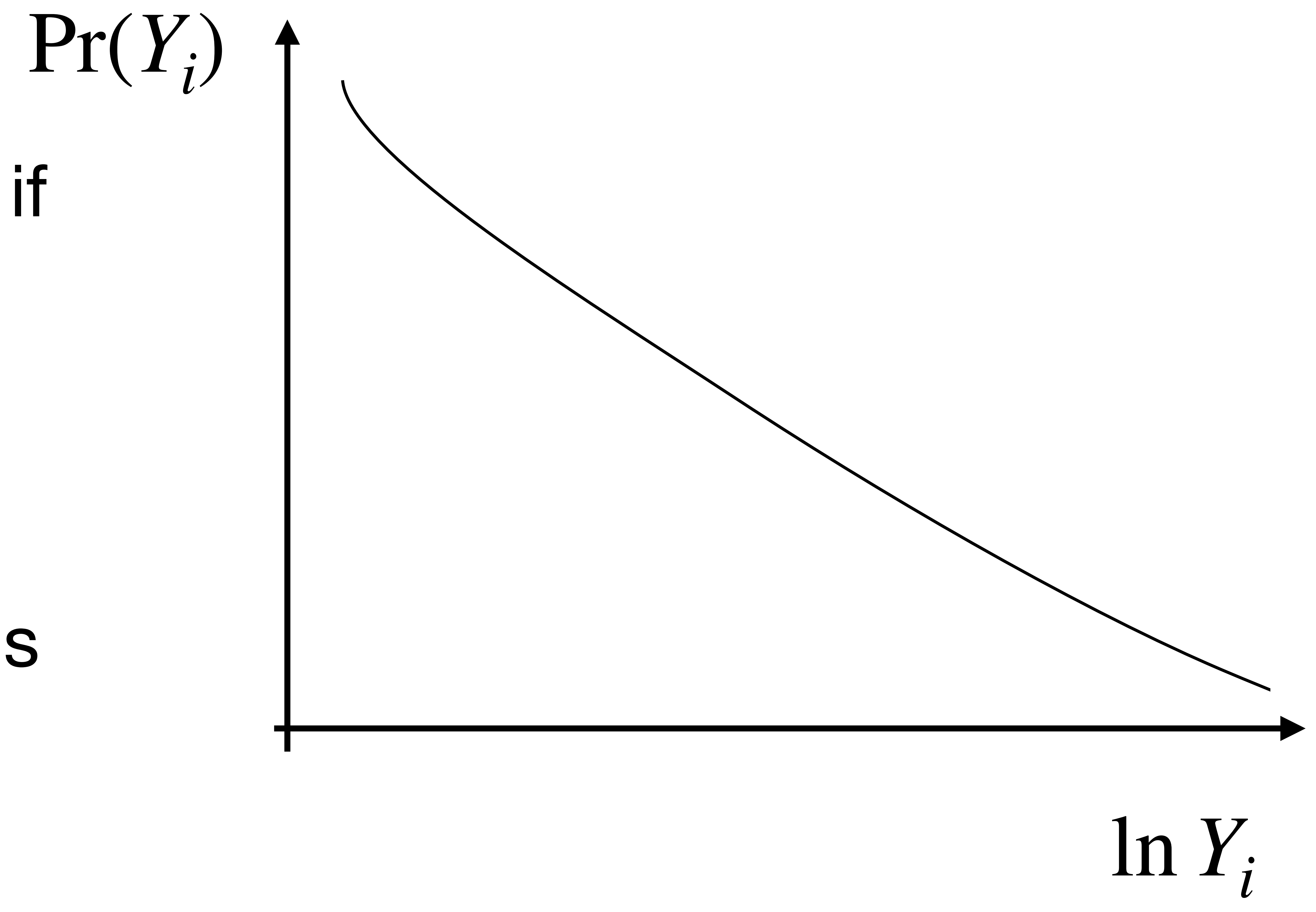
- **(Imperfect) Solution**: Transform your outcome with $\ln(Y_i + 1)$. This can be a problem if your data contains a lot of 0s.



Solution: “Log” your outcome!

- **Problem**: $\ln 0$ is undefined! What if you have GitHub projects with 0 issues?

- **(Imperfect) Solution**: Transform your outcome with $\ln(Y_i + 1)$. This can be a problem if your data contains a lot of 0s.



Broader solution

- Logging the outcome is a “hack.”
- Represents a broader idea: create a model that better captures what the data looks like.
- Instead of OLS, you may want to choose a model whose assumptions match the data-generating process.
  - This is usually “the cherry on the cake,” basic OLS can get you very far.

Generalized Linear Models

- It turns out OLS is a special case of a broader family of models.
- Ingredients:
  - Choose a distribution for $Y \mid X$ from the exponential family.
  - Choose a link function $g(\cdot)$, such that $g(E[Y_i | X_i]) = x_i^\top \beta$.
- For OLS
  - $Y_i \mid X_i \sim \mathcal{N}(\mu_i, \sigma^2)$
  - $g(\cdot)$ is the identity function!

Generalized Linear Models

- What we artificially did with logging the outcome can be better achieved with a Poisson regression.
  - $Y \mid X \sim \text{Poisson}(\mu_i)$
  - $g(\mu_i) = \log(\mu_i)$
  - This model comports 0 outcomes!

- We can fit any GLM with another technique (Maximum Likelihood Estimation).

- This method is OLS when we assume:
  - $Y_i \mid X_i \sim \mathcal{N}(\mu_i, \sigma^2)$
  - $g(\cdot)$ is the identity function!

Generalized Linear Models

- Most models you hear about GLMs:
  - Logistic Regression
  - Poisson Regression
  - Gamma regression

- Don’t overestimate OLS; it has neat properties that not all GLMs have
  - Exact finite-sample unbiasedness of $\hat{\beta}$.
  - Frisch–Waugh–Lovell (FWL) exact partialling out.
  - Interpretation is very simple.

Back to the original problem…

We are interested in estimating the effect of GenAI (T_i) on the number of *issues* (Y_i) opened on GitHub projects (as a proxy for code quality).

A bunch of other factors may influence both issues and AI usage, e.g., how big the project is!

If you have panel data, there's a technique that will get you very far…

But what is *panel data*?

Panel data tracks the *same units* repeatedly over time! For example:

- Unit i: a GitHub project
- Time t: week/month
- Outcome Y_{it}: issues opened for project i in time t
- Treatment T_{it}: GenAI adoption

But what is *panel data*?

Project	T	Y	T
repoA	2025-01	7	0
repoA	2025-02	4	1
repoB	2025-01	0	0
repoB	2025-02	1	0

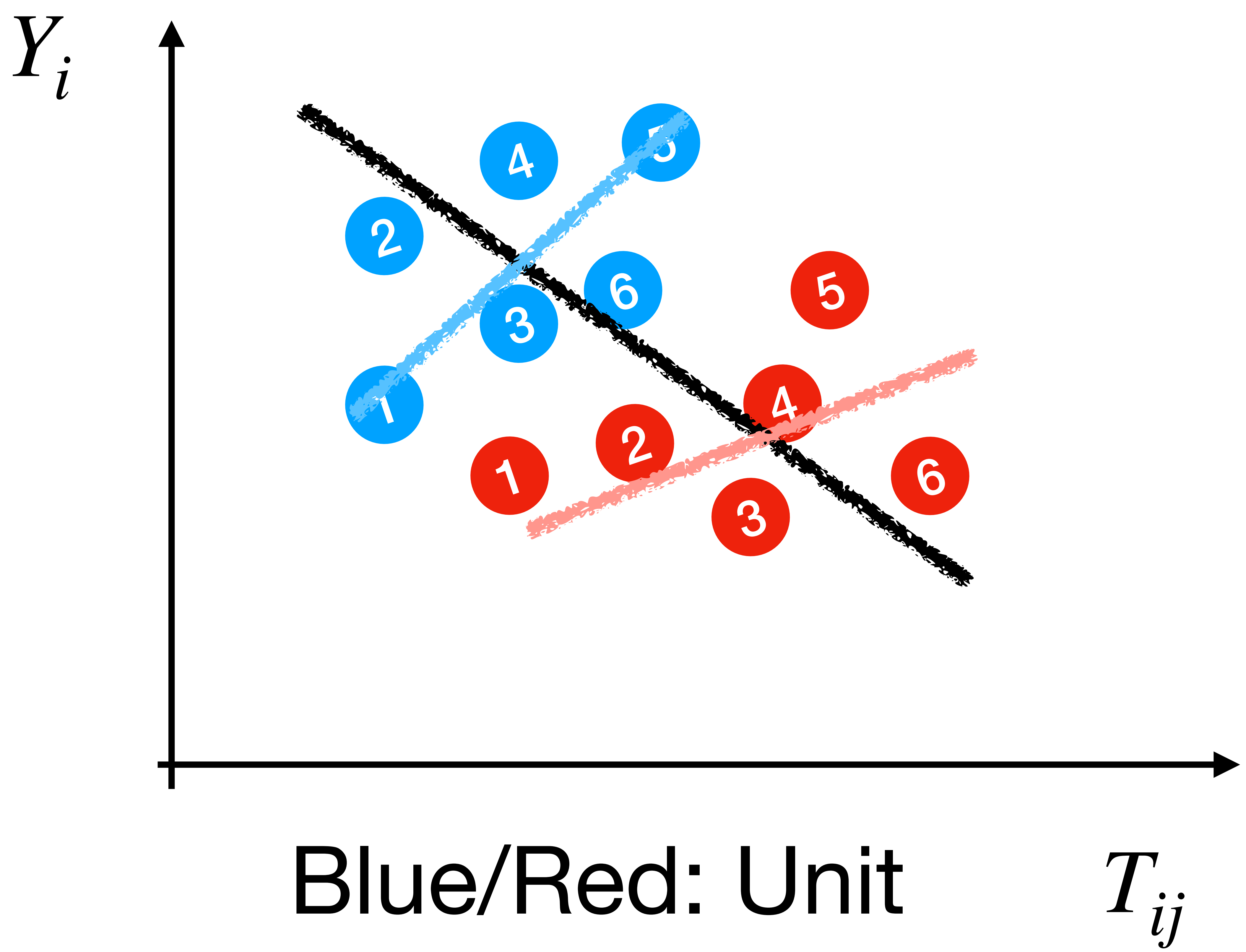
The fixed effects model

Idea: add a dummy variable for each unit!

$$Y_{it} = \beta_1 T_{it} + \sum_i \alpha_i M_i + \epsilon_{it}$$

Note the absence of an intercept!

Remember FWL!



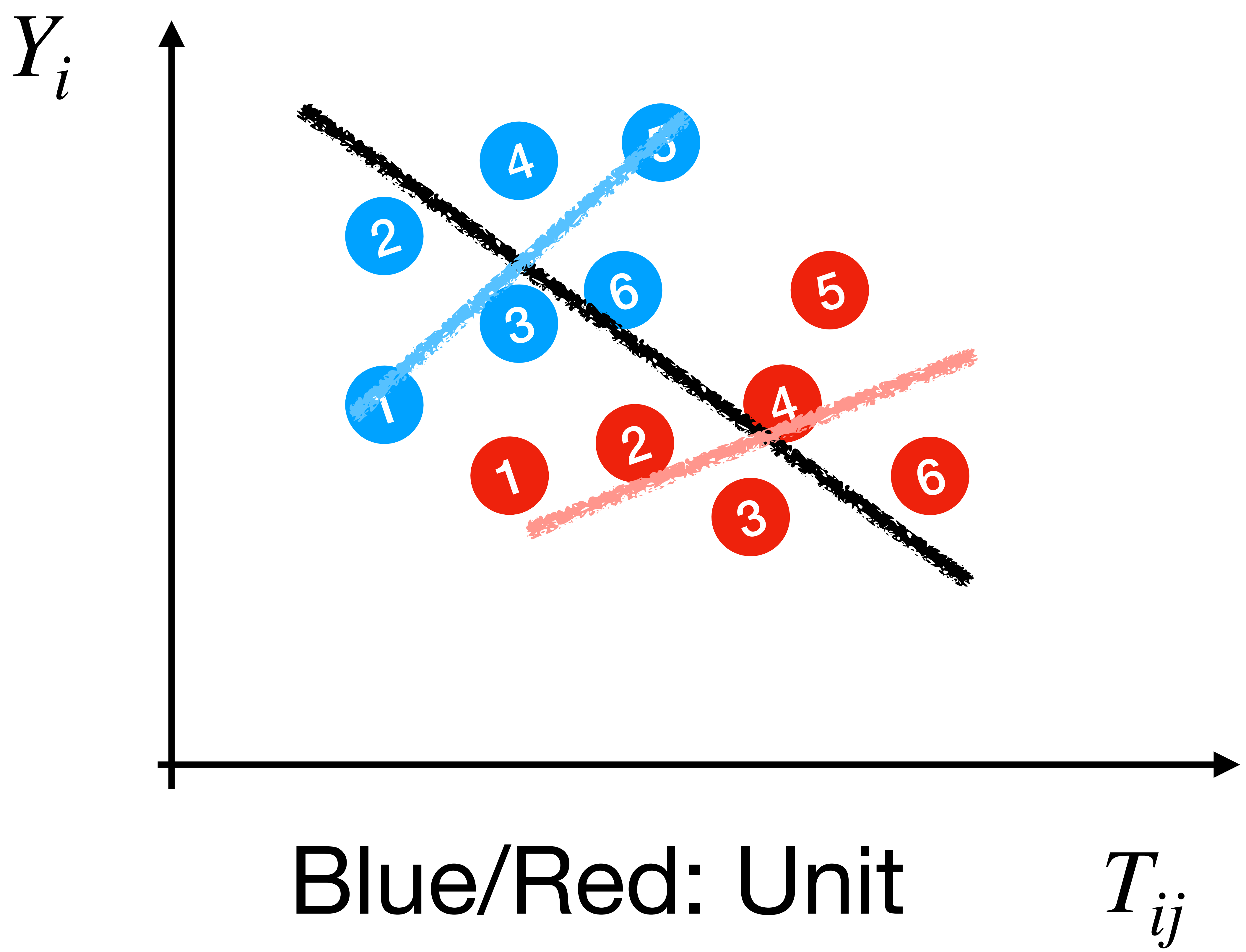
Blue/Red: Unit

Text: Week

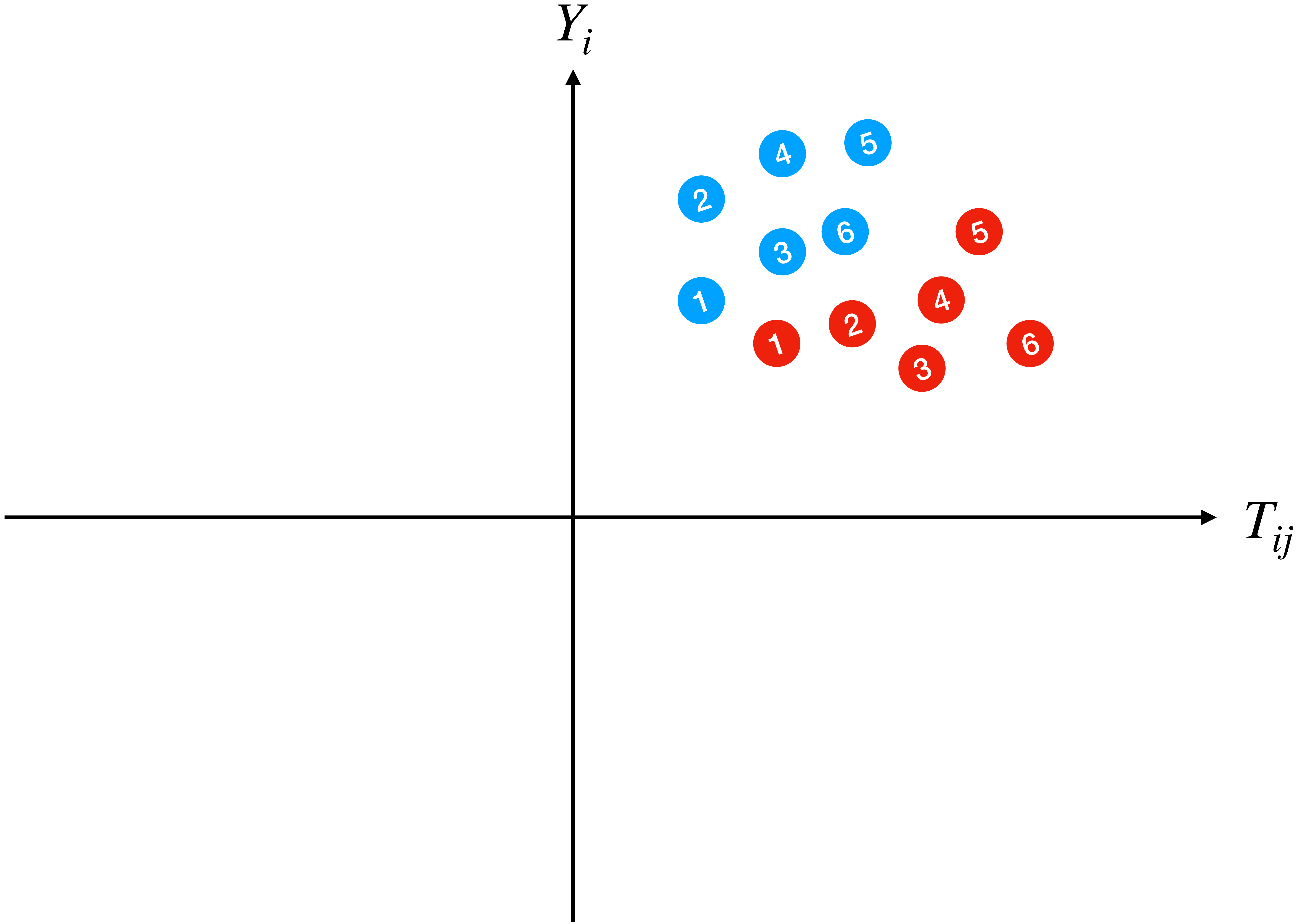
The fixed effects model

For a given unit, the dummy will capture the average Y_i!

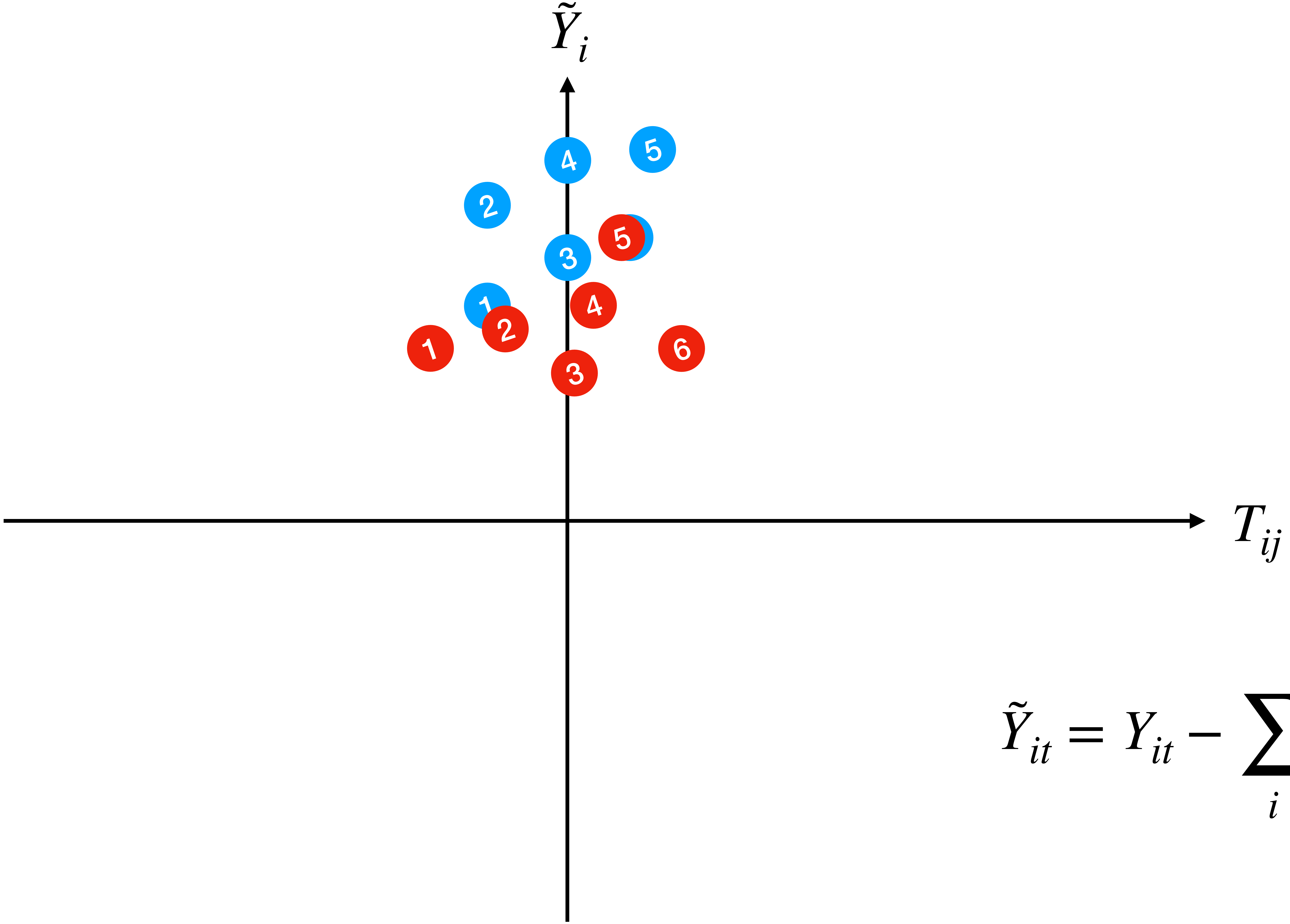
$$\hat{\beta}_1 = \frac{\text{Cov}(Y_i, \tilde{T}_i)}{\text{Var}(\tilde{T}_i)}$$



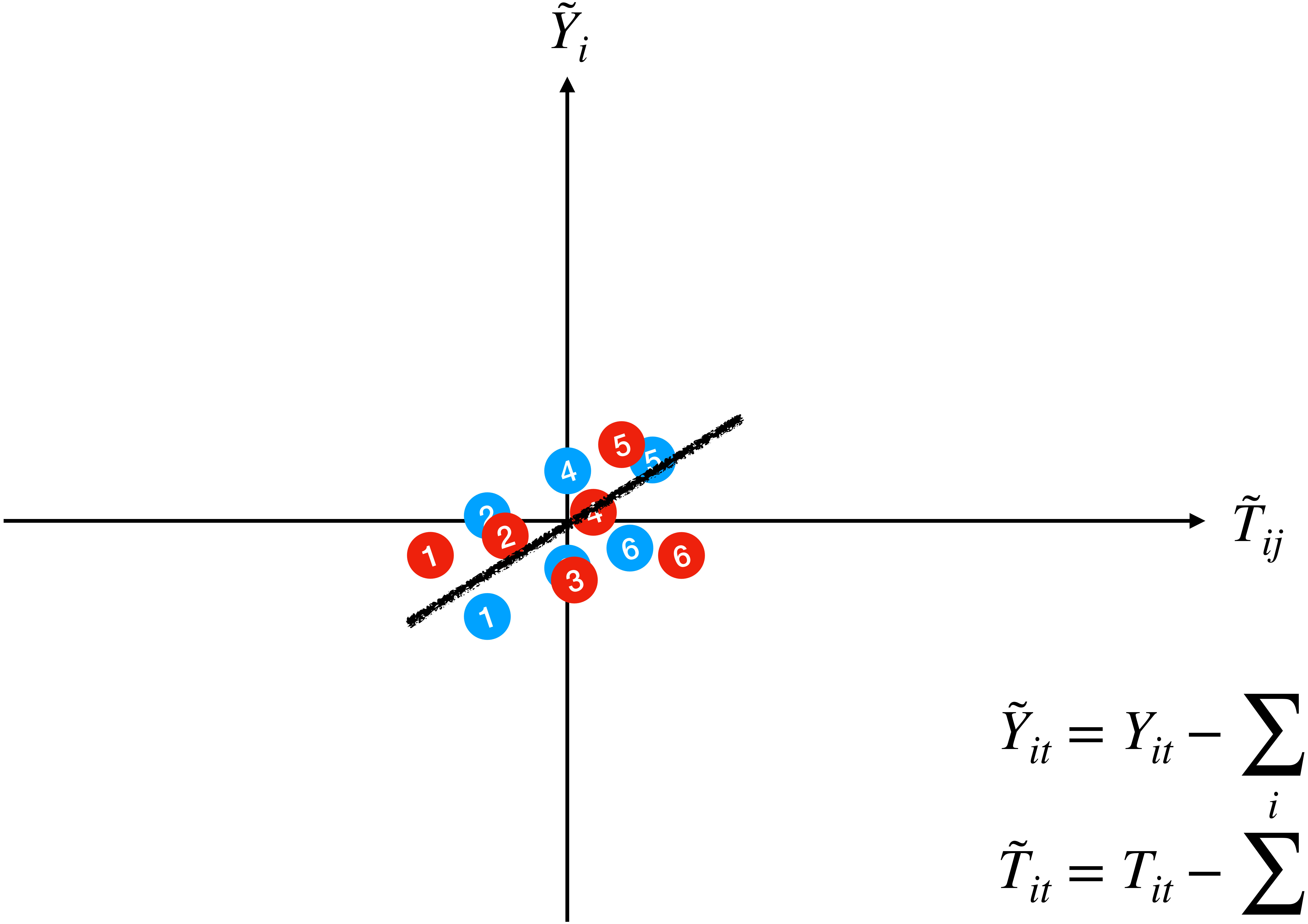
Blue/Red: Unit
Text: Week



Y_i
T_{ij}
4
5
2
6
5
3
1
4
1
2
6
3



$$\tilde{Y}_{it} = Y_{it} - \sum_i \alpha_i M_i$$



$$\tilde{Y}_{it} = Y_{it} - \sum_i \alpha_i M_i$$

$$\tilde{T}_{it} = T_{it} - \sum_i \alpha_i M_i$$

Fixed effects is magic!

- Fixed effects feels like magic because it lets you control for a huge number of confounders without measuring them explicitly.

- The project-level fixed effects soak up anything about the project that's *constant over time* (baseline size, popularity, culture, etc.).

Some warnings!

- **Beware**: measures from the same project are correlated!

  Solution: Use cluster robust standard errors!

- **Beware:** But projects don't evolve in a vacuum. There are shocks and trends that affect everyone at once! E.g., Claude Code is being released!

  Solution: Add time fixed effects! $Y_{it} = \beta_1 T_{it} + \sum_i \alpha_i M_i + \sum_t \gamma_t N_t + \epsilon_{it}$

- **Beware:** Fixed effects are not super "efficient," maybe you want to draw statistical conclusions from the between-units variation!

Project Discussion!