Intervalos de confiança em regressão linear

Renato

Kaggle Dataset

- Aim: To predict the compressive strength of concrete based on material composition.

🎯 Target Variable (Response Variable)

Feature Name	Description	Units	Typical Range
Compressive Strength	The maximum compressive stress the concrete can withstand.	MPa (MegaPascals)	2.33 - 82.6

- Number of Samples: 1,030 observations
- Number of Features: 8 predictors

```
#generate OLS regression results for all features
import statsmodels.api as sm

X_sm = sm.add_constant(X)
model = sm.OLS(y,X_sm)
print(model.fit().summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  csMPa   R-squared:                       0.616
Model:                            OLS   Adj. R-squared:                  0.613
Method:                 Least Squares   F-statistic:                     204.3
Date:                Fri, 15 Oct 2021   Prob (F-statistic):          6.29e-206
Time:                        16:43:15   Log-Likelihood:                -3869.0
No. Observations:                1030   AIC:                             7756.
Df Residuals:                    1021   BIC:                             7800.
Df Model:                           8
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const        -23.3312     26.586     -0.878      0.380     -75.500      28.837
cement         0.1198      0.008     14.113      0.000       0.103       0.136
slag           0.1039      0.010     10.247      0.000       0.084       0.124
flyash         0.0879      0.013      6.988      0.000       0.063       0.113
water         -0.1499      0.040     -3.731      0.000      -0.229      -0.071
superplasticizer  0.2922      0.093      3.128      0.002       0.109       0.476
coarseaggregate  0.0181      0.009      1.926      0.054      -0.000       0.037
fineaggregate  0.0202      0.011      1.887      0.059      -0.001       0.041
age            0.1142      0.005     21.046      0.000       0.104       0.125
==============================================================================
```

```python
#generate OLS regression results for all features
import statsmodels.api as sm

X_sm = sm.add_constant(X)
model = sm.OLS(y,X_sm)
print(model.fit().summary())
```

OLS Regression Results

Dep. Variable:	csMPa	R-squared:	0.616
Model:	OLS	Adj. R-squared:	0.613
Method:	Least Squares	F-statistic:	204.3
Date:	Fri, 15 Oct 2021	Prob (F-statistic):	6.29e-206
Time:	16:43:15	Log-Likelihood:	-3869.0
No. Observations:	1030	AIC:	7756.
Df Residuals:	1021	BIC:	7800.
Df Model:	8		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-23.3312	26.586	-0.878	0.380	-75.500	28.837
cement	0.1198	0.008	14.113	0.000	0.103	0.136
slag	0.1039	0.010	10.247	0.000	0.084	0.124
flyash	0.0879	0.013	6.988	0.000	0.063	0.113
water	-0.1499	0.040	-3.731	0.000	-0.229	-0.071
superplasticizer	0.2922	0.093	3.128	0.002	0.109	0.476
coarseaggregate	0.0181	0.009	1.926	0.054	-0.000	0.037
fineaggregate	0.0202	0.011	1.887	0.059	-0.001	0.041
age	0.1142	0.005	21.046	0.000	0.104	0.125

Assuma que o modelo de regressão linear e' o mecanismo gerador dos dados.

Existe um vetor desconhecido de coeficientes que gera os dados e que queremos aprender

$$\boldsymbol{\beta}^* = \begin{bmatrix} \beta_0^* \\ \beta_1^* \\ \vdots \\ \beta_p^* \end{bmatrix}$$

```python
#generate OLS regression results for all features
import statsmodels.api as sm

X_sm = sm.add_constant(X)
model = sm.OLS(y,X_sm)
print(model.fit().summary())
```

OLS Regression Results

Dep. Variable:	csMPa	R-squared:	0.616
Model:	OLS	Adj. R-squared:	0.613
Method:	Least Squares	F-statistic:	204.3
Date:	Fri, 15 Oct 2021	Prob (F-statistic):	6.29e-206
Time:	16:43:15	Log-Likelihood:	-3869.0
No. Observations:	1030	AIC:	7756.
Df Residuals:	1021	BIC:	7800.
Df Model:	8		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-23.3312	26.586	-0.878	0.380	-75.500	28.837
cement	0.1198	0.008	14.113	0.000	0.103	0.136
slag	0.1039	0.010	10.247	0.000	0.084	0.124
flyash	0.0879	0.013	6.988	0.000	0.063	0.113
water	-0.1499	0.040	-3.731	0.000	-0.229	-0.071
superplasticizer	0.2922	0.093	3.128	0.002	0.109	0.476
coarseaggregate	0.0181	0.009	1.926	0.054	-0.000	0.037
fineaggregate	0.0202	0.011	1.887	0.059	-0.001	0.041
age	0.1142	0.005	21.046	0.000	0.104	0.125

Sabemos que

$$\beta^* \neq \hat{\beta}$$

Mas esperamos que sejam próximos, que o erro de estimação seja pequeno.

E’ claro que não podemos saber exatamente qual foi nosso erro

(se soubéssemos, bastaria subtrair o erro e ter o valor verdadeiro e desconhecido).

```python
#generate OLS regression results for all features
import statsmodels.api as sm

X_sm = sm.add_constant(X)
model = sm.OLS(y,X_sm)
print(model.fit().summary())
```

OLS Regression Results

Dep. Variable:	csMPa	R-squared:	0.616
Model:	OLS	Adj. R-squared:	0.613
Method:	Least Squares	F-statistic:	204.3
Date:	Fri, 15 Oct 2021	Prob (F-statistic):	6.29e-206
Time:	16:43:15	Log-Likelihood:	-3869.0
No. Observations:	1030	AIC:	7756.
Df Residuals:	1021	BIC:	7800.
Df Model:	8		
Covariance Type:	nonrobust		

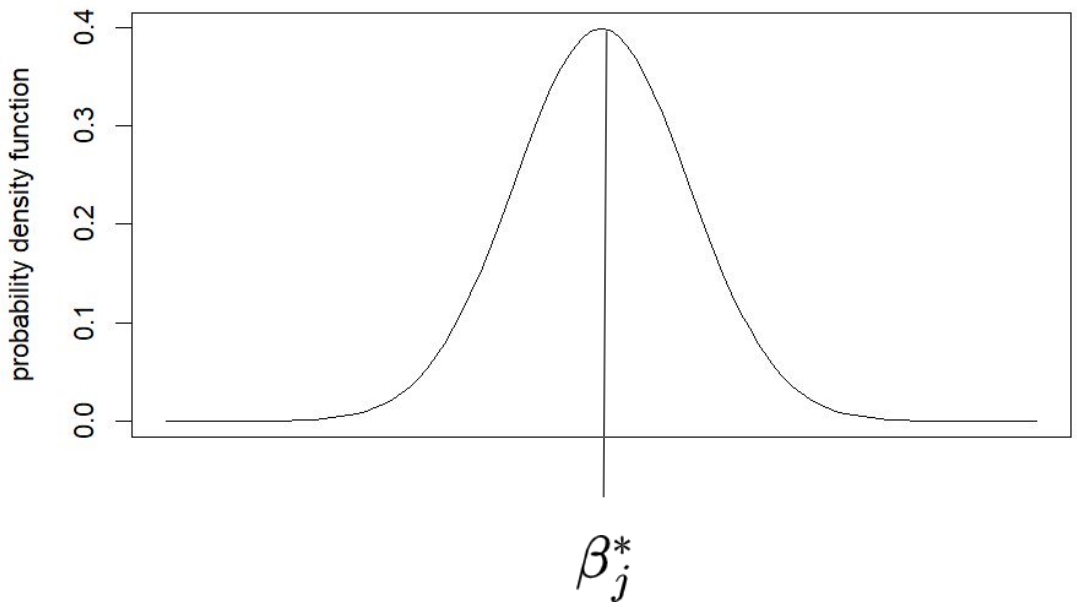
	coef	std err	t	P>|t|	[0.025	0.975]
const	-23.3312	26.586	-0.878	0.380	-75.500	28.837
cement	0.1198	0.008	14.113	0.000	0.103	0.136
slag	0.1039	0.010	10.247	0.000	0.084	0.124
flyash	0.0879	0.013	6.988	0.000	0.063	0.113
water	-0.1499	0.040	-3.731	0.000	-0.229	-0.071
superplasticizer	0.2922	0.093	3.128	0.002	0.109	0.476
coarseaggregate	0.0181	0.009	1.926	0.054	-0.000	0.037
fineaggregate	0.0202	0.011	1.887	0.059	-0.001	0.041
age	0.1142	0.005	21.046	0.000	0.104	0.125

Talvez seja uma surpresa que, embora não saibamos o erro exato, sabemos o quanto podemos esperar **tipicamente** desse erro.

Sabemos que

$$\hat{\boldsymbol{\beta}} \sim N_{p+1}(\boldsymbol{\beta}^*, \sigma^2(\mathbf{X}'\mathbf{X})^{-1})$$

Então: cada coordenada $\hat{\beta}_j$ do vetor estimado $\hat{\boldsymbol{\beta}}$ oscila como uma Gaussiana em torno do seu verdadeiro e desconhecido valor β_j

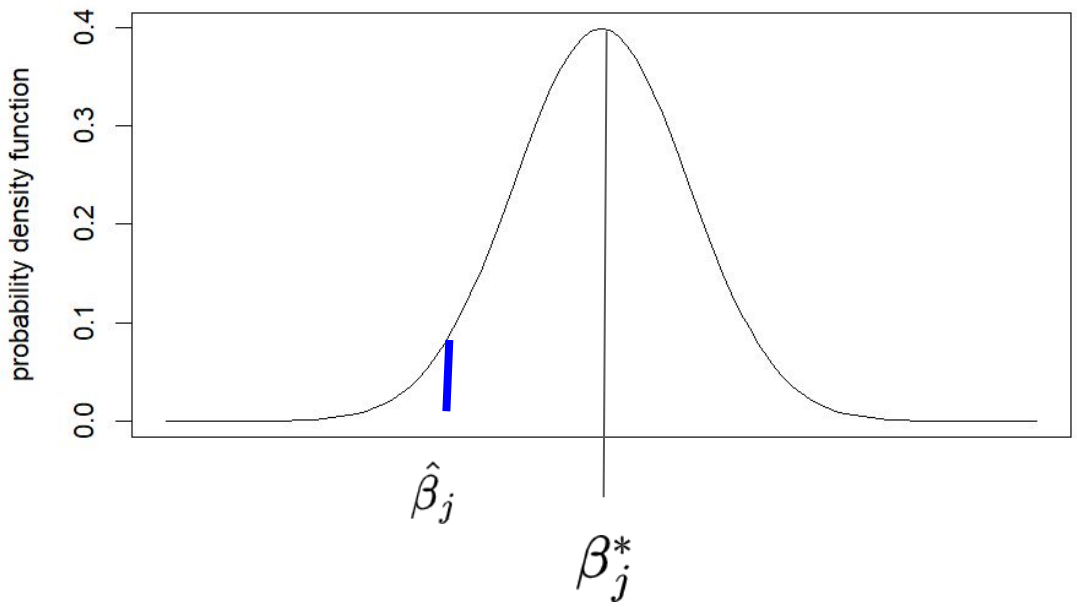


Considere uma dessas coordenadas:

$$\hat{\beta}_j \sim N(\beta_j^*, \sigma^2(\mathbf{X}'\mathbf{X})^{-1}[jj])$$

$\hat{\beta}_j$ e' um valor aleatório que oscila como uma Gaussiana em torno de um centro desconhecido que e' β_j^*

$$\hat{\beta}_j \sim N(\beta_j^*, \sigma^2(\mathbf{X}'\mathbf{X})^{-1}[jj])$$



Observamos um valor dessa distribuição, o valor observado $\hat{\beta}_j$

$$\hat{\beta}_j \sim N(\beta_j^*, \sigma^2(\mathbf{X}'\mathbf{X})^{-1}[jj])$$

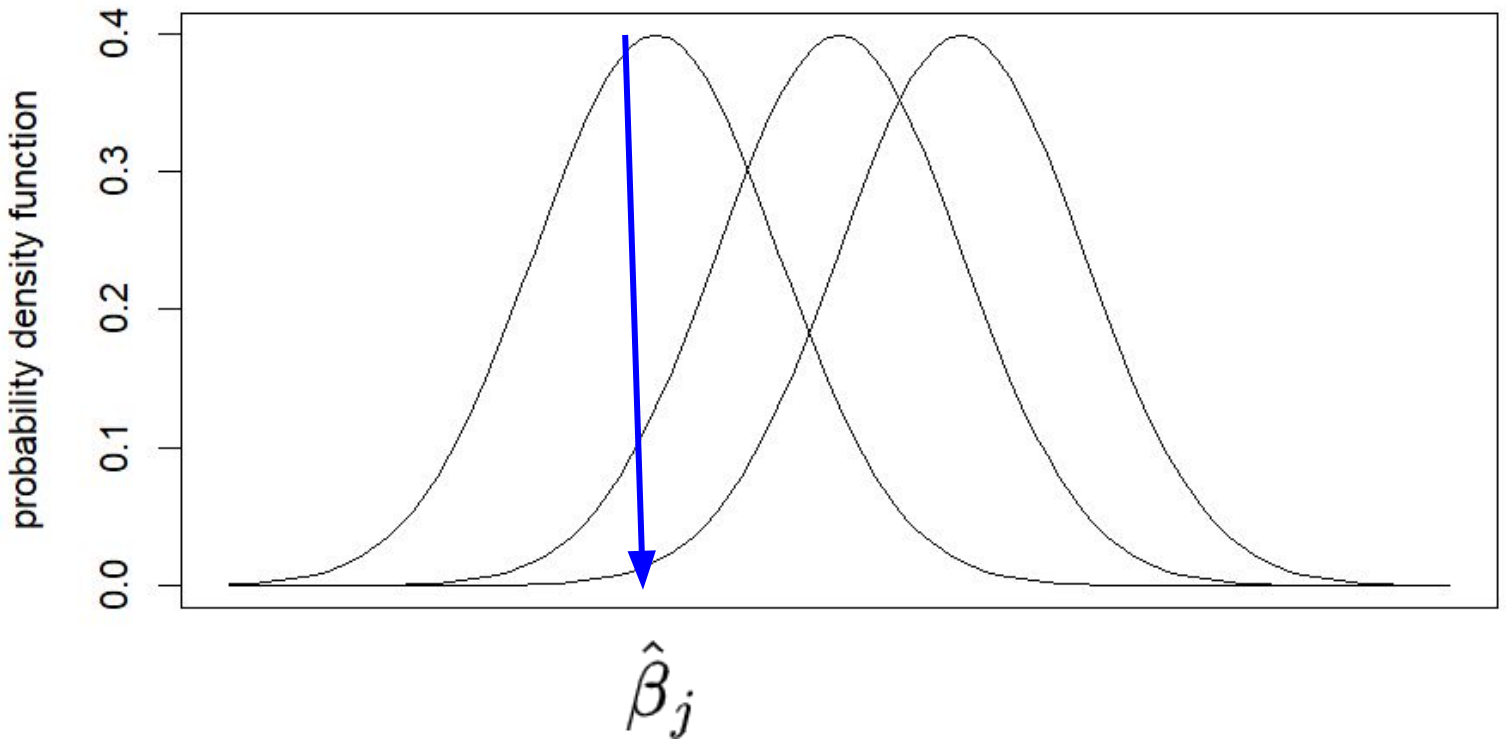
O que REALMENTE observamos:

apenas $\hat{\beta}_j$

Como ter noção do erro que estamos cometendo?

A Gaussiana está centrada no β_j^* desconhecido

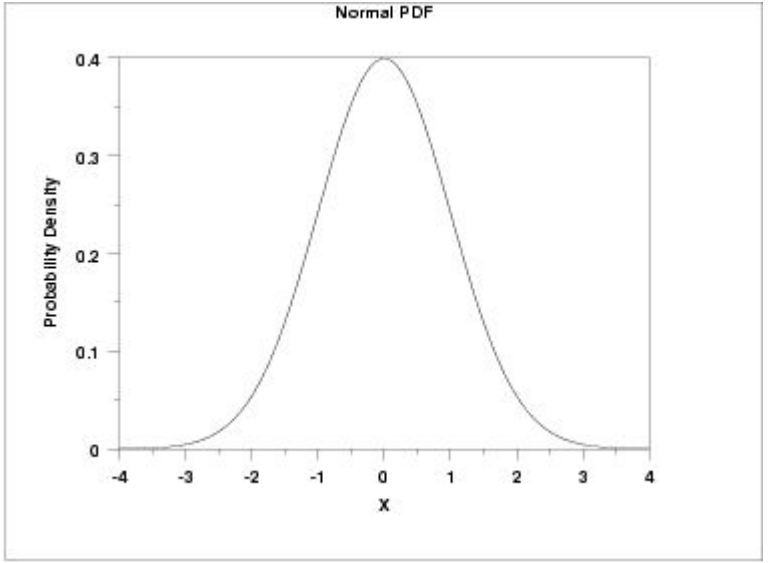
Não sabemos quais dessas Gaussianas e' a verdadeira pois não sabemos o valor de β_j^*



Gaussiana padrão

Mas sabemos muito sobre o comportamento de variáveis aleatórias Gaussianas.

1) Considere a Gaussiana padrão: N(0,1):
   a) Dificilmente sai de (-2, 2)
   b) Probab(estar em (-2,2)) = 0.95 (ou 95%)



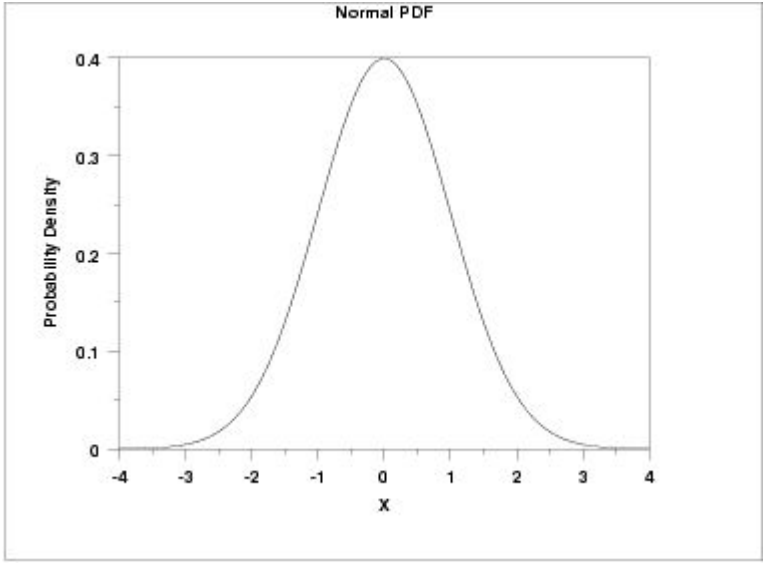
Gaussiana padrão

Mas sabemos muito sobre o comportamento de variáveis aleatórias Gaussianas.

1) Considere a Gaussiana padrão: N(0,1):
   a) Dificilmente sai de (-2, 2)
   b) Probab(estar em (-2,2)) = 0.95 (ou 95%)

2) Toda Gaussiana $X \sim N(\mu, \sigma^2)$

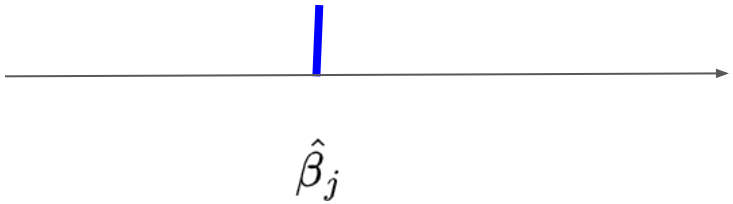
   pode ser transformada em Gaussiana padrão: $Z = \frac{X-\mu}{\sigma} \sim N(0,1)$



De volta para regressão linear…

$$\hat{\beta}_j \sim N(\beta_j^*, \sigma^2(\mathbf{X}'\mathbf{X})^{-1}[jj])$$

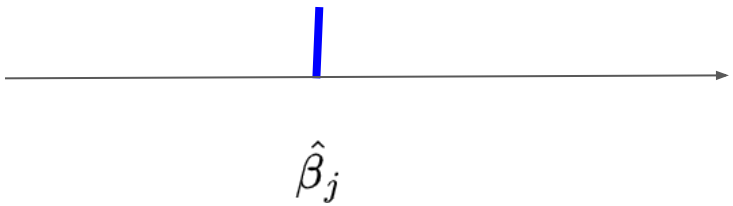
$$Z = \frac{\hat{\beta}_j - \beta_j^*}{\sqrt{\sigma^2(\mathbf{X}'\mathbf{X})^{-1}[jj]}} \sim N(0,1)$$



De volta para regressão linear…

$$\hat{\beta}_j \sim N(\beta_j^*, \sigma^2(\mathbf{X}'\mathbf{X})^{-1}[jj])$$

$$Z = \frac{\hat{\beta}_j - \beta_j^*}{\sqrt{\sigma^2(\mathbf{X}'\mathbf{X})^{-1}[jj]}} \sim N(0,1)$$



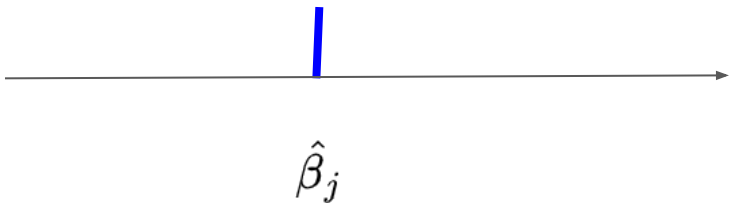
Esta variável aleatória Z dificilmente sairá fora do intervalo (-2, 2).

95% de chance de estar em (-2, 2)

De volta para regressão linear…

$$\hat{\beta}_j \sim N(\beta_j^*, \sigma^2(\mathbf{X}'\mathbf{X})^{-1}[jj])$$

$$Z = \frac{\hat{\beta}_j - \beta_j^*}{\sqrt{\sigma^2(\mathbf{X}'\mathbf{X})^{-1}[jj]}} \sim N(0,1)$$



Esta variável aleatória Z dificilmente sairá fora do intervalo (-2, 2).

95% de chance de estar em (-2,2)

Assim, com alta probabilidade (confiança de 95%) Z estará em (-2,2)

E daí?

Simplifique a expressão Z:

$$Z = \frac{\hat{\beta}_j - \beta_j^*}{\sqrt{\sigma^2 (\mathbf{X}'\mathbf{X})^{-1}[jj]}} = \frac{\hat{\beta}_j - \beta_j^*}{v}$$

$$Z = \frac{\hat{\beta}_j - \beta_j^*}{\sqrt{\sigma^2 (\mathbf{X}'\mathbf{X})^{-1}[jj]}} = \frac{\hat{\beta}_j - \beta_j^*}{v}$$

Com probabilidade 0.95 (ou 95% de chance) Z está em (-2, 2) o que implica:

$$-2 \leq \frac{\hat{\beta}_j - \beta_j^*}{v} \leq 2$$

$$Z = \frac{\hat{\beta}_j - \beta_j^*}{\sqrt{\sigma^2 (\mathbf{X}'\mathbf{X})^{-1}[jj]}} = \frac{\hat{\beta}_j - \beta_j^*}{v}$$

Com probabilidade 0.95 (ou 95% de chance) Z está em (-2, 2) o que implica:

$$-2 \leq \frac{\hat{\beta}_j - \beta_j^*}{v} \leq 2$$

$$-2v \leq \quad (\hat{\beta}_j - \beta_j^*) \quad \leq 2v$$

Repetindo:

$$-2v \leq \quad (\hat{\beta}_j - \beta_j^*) \quad \leq 2v$$

Podemos manipular a expressão acima passando qualquer um dos dois termos centrais para os extremos.

Por exemplo, some β_j^* nos três lados (isto não altera as desigualdades):

$$\beta_j^* - 2v \leq \quad (\hat{\beta}_j - \beta_j^*) + \beta_j^* \quad \leq \beta_j^* + 2v$$

$$\beta_j^* - 2v \leq \quad \hat{\beta}_j \quad \leq \beta_j^* + 2v$$

Isto não é' muito útil pois β_j^* é desconhecido

$$-2v \leq \quad (\hat{\beta}_j - \beta_j^*) \quad \leq 2v$$

Subtraindo $\hat{\beta}_j$ dos três lados da desigualdade inicial

$$-\hat{\beta}_j - 2v \leq \quad -\hat{\beta}_j + (\hat{\beta}_j - \beta_j^*) \quad \leq -\hat{\beta}_j + 2v$$

Ou seja: $$-\hat{\beta}_j - 2v \leq \quad (-\beta_j^*) \quad \leq -\hat{\beta}_j + 2v$$

Multiplique todos os três lados por (-1) (e troque as desigualdades):

$$(-1) * (-\hat{\beta}_j - 2v) \geq \quad (-1) * (-\beta_j^*) \quad \geq (-1) * (-\hat{\beta}_j + 2v)$$

$$\hat{\beta}_j + 2v \geq \quad \beta_j^* \quad \geq \hat{\beta}_j - 2v$$

$$\hat{\beta}_j + 2v \geq \quad \beta_j^* \quad \geq \hat{\beta}_j - 2v$$

Ou na ordem em que eu gosto mais (do menor para o maior):

$$\hat{\beta}_j - 2v \leq \quad \beta_j^* \quad \leq \hat{\beta}_j + 2v$$

Terminamos a matemática. Vamos interpretar:

Os extremos da desigualdade formam um intervalo IC:

$$IC = [\ \ \hat{\beta}_j - 2v, \quad \hat{\beta}_j + 2v\ \ ]$$

Este intervalo é' aleatório!!!! (pois $\hat{\beta}_j$ é aleatório)

Com probabilidade 0.95, este IC aleatório cobre o verdadeiro e desconhecido β_j^*

Resumindo:

O vetor estimado de coeficientes: $\hat{\boldsymbol{\beta}} \sim N_{p+1}(\boldsymbol{\beta}^*, \sigma^2(\mathbf{X}'\mathbf{X})^{-1})$

Considerando uma das coordenadas: $\hat{\beta}_j \sim N(\beta_j^*, \sigma^2(\mathbf{X}'\mathbf{X})^{-1}[jj])$

Notação para simplificar: $v^2 = \sigma^2(\mathbf{X}'\mathbf{X})^{-1}[jj]$

Construa o intervalo aleatório: $IC = [\ \ \hat{\beta}_j - 2v, \quad \hat{\beta}_j + 2v\ \ ]$

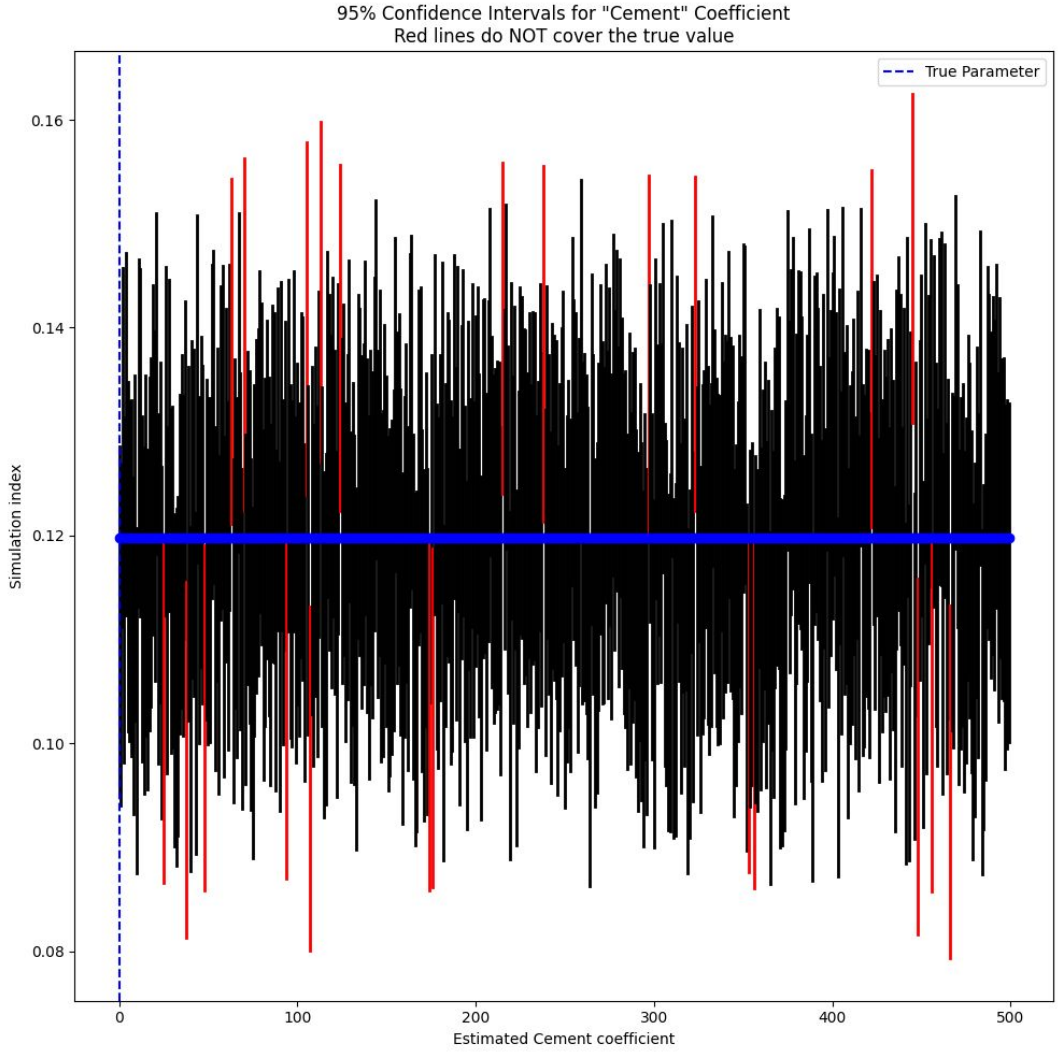
O verdadeiro e desconhecido valor β_j^* do coeficiente pode ou não estar dentro desse IC aleatório.

A probabilidade de estar no IC e' 0.95: $\mathbb{P}(\text{IC contains } \beta_j^*) = 0.95$

Exemplo: cimento

Ver código no notebook.

Cobertura empírica: 95.4%



```python
#generate OLS regression results for all features
import statsmodels.api as sm

X_sm = sm.add_constant(X)
model = sm.OLS(y,X_sm)
print(model.fit().summary())
```

OLS Regression Results

Dep. Variable:	csMPa	R-squared:	0.616
Model:	OLS	Adj. R-squared:	0.613
Method:	Least Squares	F-statistic:	204.3
Date:	Fri, 15 Oct 2021	Prob (F-statistic):	6.29e-206
Time:	16:43:15	Log-Likelihood:	-3869.0
No. Observations:	1030	AIC:	7756.
Df Residuals:	1021	BIC:	7800.
Df Model:	8		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-23.3312	26.586	-0.878	0.380	-75.500	28.837
cement	0.1198	0.008	14.113	0.000	0.103	0.136
slag	0.1039	0.010	10.247	0.000	0.084	0.124
flyash	0.0879	0.013	6.988	0.000	0.063	0.113
water	-0.1499	0.040	-3.731	0.000	-0.229	-0.071
superplasticizer	0.2922	0.093	3.128	0.002	0.109	0.476
coarseaggregate	0.0181	0.009	1.926	0.054	-0.000	0.037
fineaggregate	0.0202	0.011	1.887	0.059	-0.001	0.041
age	0.1142	0.005	21.046	0.000	0.104	0.125

Na prática, temos apenas um único IC que pode ou não estar realmente cobrindo o verdadeiro valor desconhecido do coeficiente.

IC: coef +- 2.0 * std. error

Por exemplo, “cement” tem IC:

0.1198 +- 2.0 * 0.008 =

[0.1038, 0.1358]

Um detalhe:

Definimos $IC = [\ \ \hat{\beta}_j - 2v, \quad \hat{\beta}_j + 2v\ \ ]$

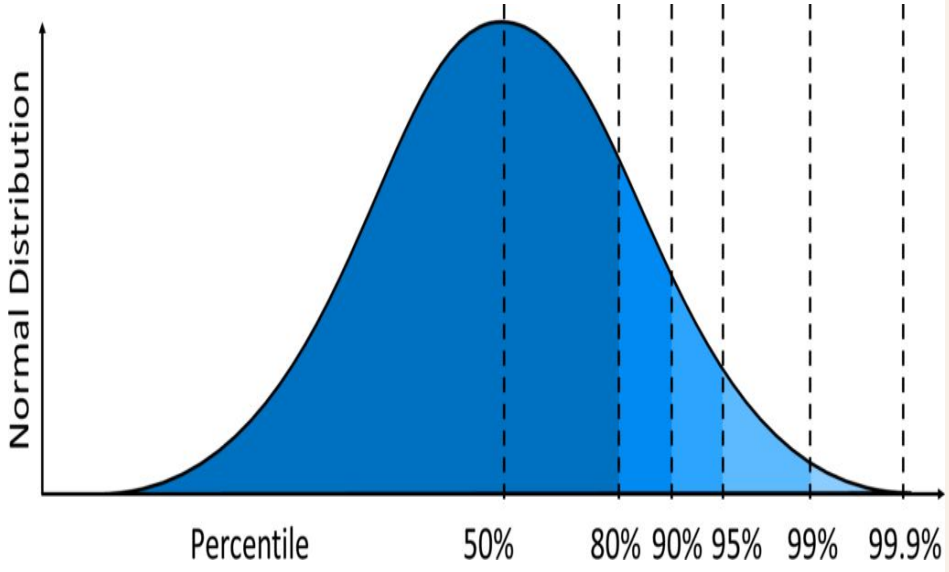
A constante “2” surge por causa da distribuição Gaussiana N(0,1) e da probabilidade 0.95.

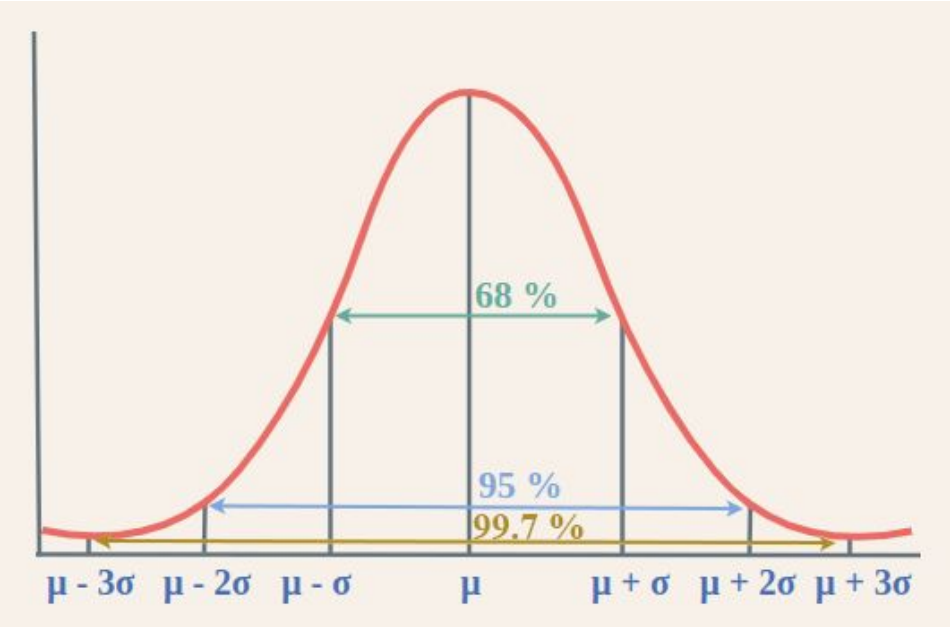
Eu usei “2” como uma aproximação para o valor exato 1.96 (Isto ‘e, o IC deveria estar usando 1.96 mas a diferença será irrisória na maioria dos casos e “2” ‘e mais fácil de memorizar.

Se você quiser um intervalo de confiança de 99% deverá usar um valor diferente de 2 (deveria ser 2.32).

2.32*sigma → 99%

3*sigma → 99.7%





Outro detalhe:

Definimos: $IC = [\quad \hat{\beta}_j - 2v, \quad \hat{\beta}_j + 2v \quad]$

com $v^2 = \sigma^2 (\mathbf{X}'\mathbf{X})^{-1}[jj]$

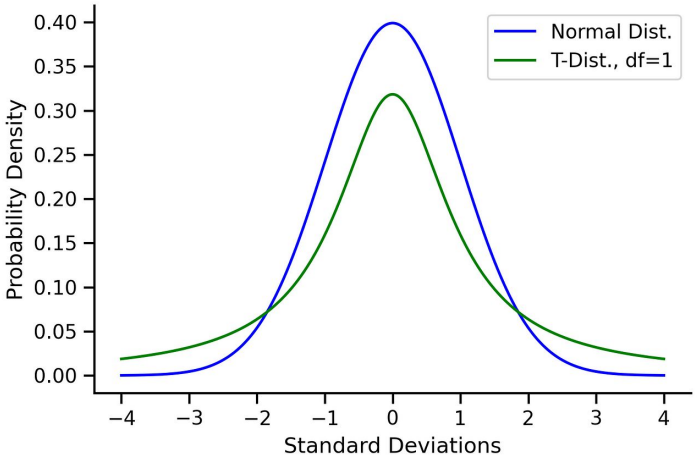
Mas σ^2 também é desconhecido.

Resolvemos isso substituindo σ^2 em v^2 por : $S^2 = \frac{1}{n-(p+1)} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$

Qual a consequência disso?

A distribuição Gaussiana padrão deve ser substituída pela t-Student.

T-Student



Densidade ‘e bem similar `a N(0,1)

Diferenca principal: caudas mais “pesadas”

Densidade nao cai para zero tao depressa quanto a N(0,1)

As constantes

“2” (para IC de 95%)

ou “2.32” (para IC de 99%)

são MAIORES.

E qual o valor dessas constantes?

Depende dos graus de liberdade…

df = Graus de liberdade (degrees of freedom)

df = dimensão do espaço vetorial onde vive o vetor de resíduos.

n = número de observações (tamanho do dataset)

p = número de features

p+1 = no. de features + (termo de bias, coluna de 1’s)

df = n - (p+1)

A distribuição t varia um pouquinho com df.

Quando df → infinito temos a N(0,1)

Figura exagerando as diferenças (só ilustrativa)

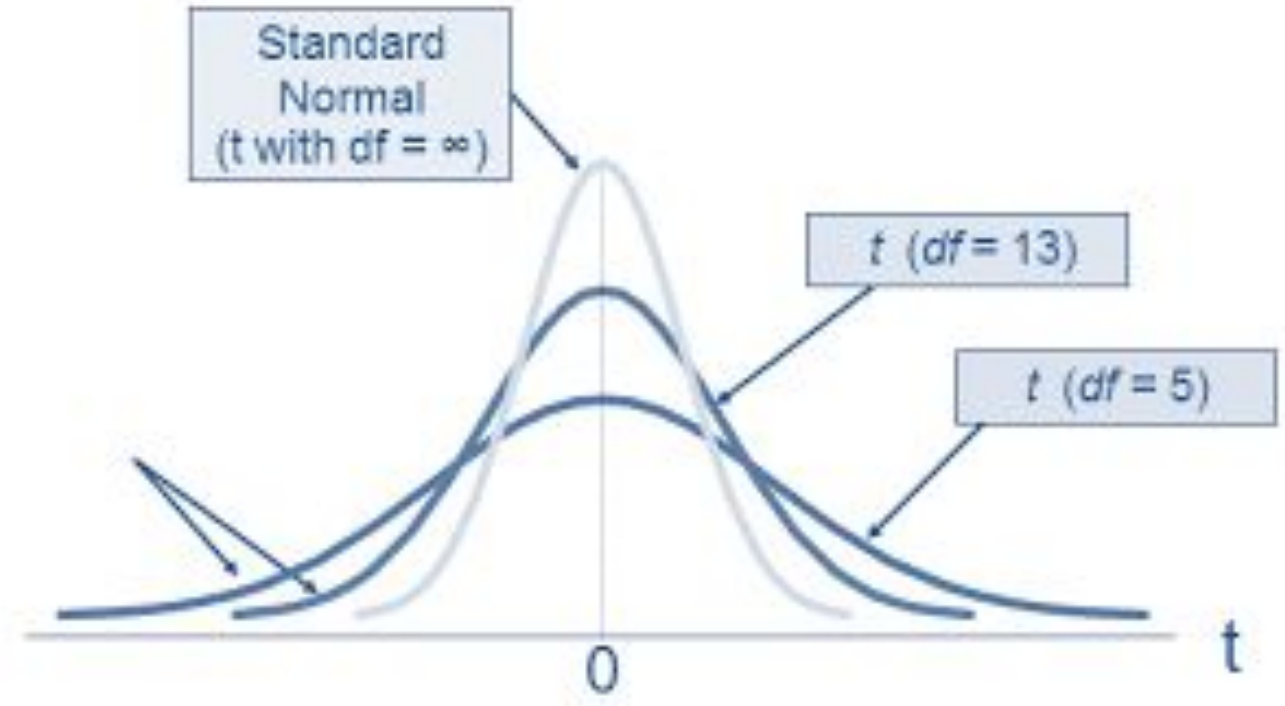
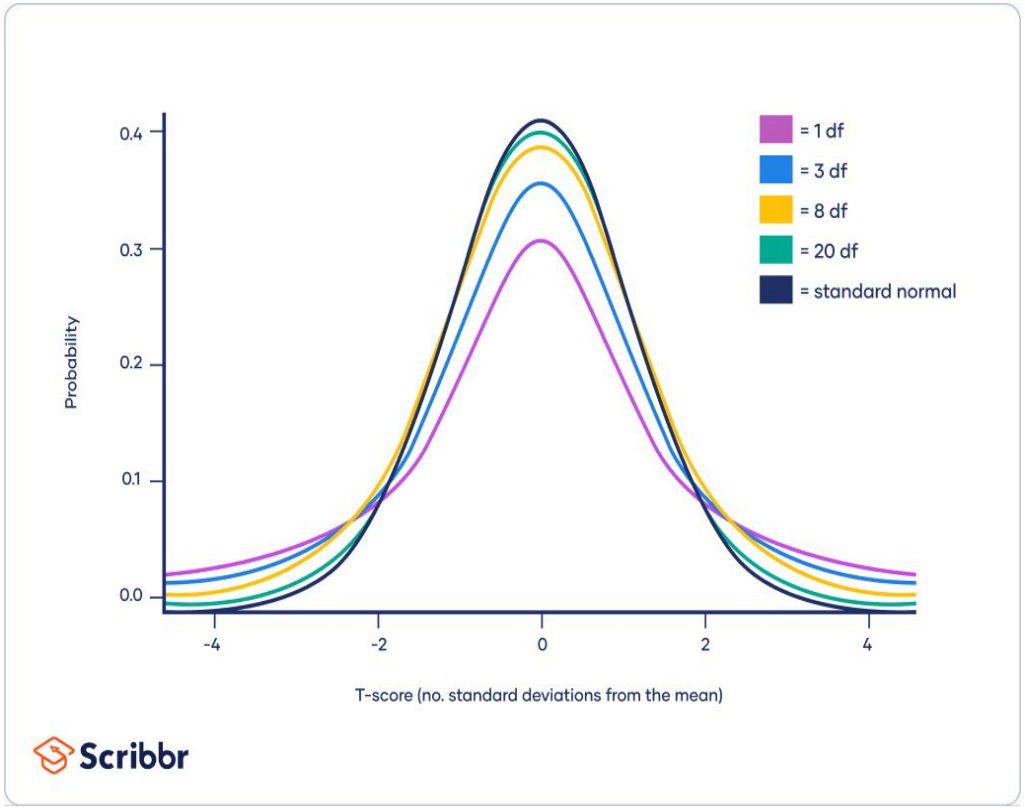


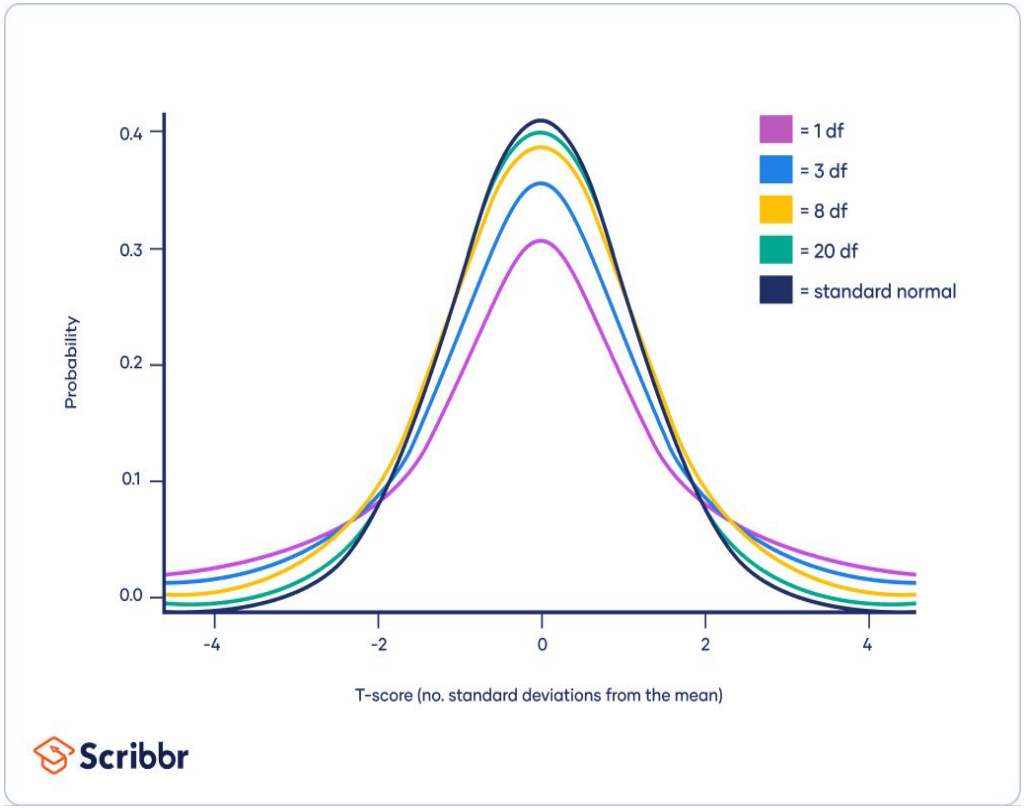
Gráfico com escalas corretas



Com df=20 ja' e' praticamente igual a N(0,1)

from scipy.stats import t
q = t.ppf(0.975, df=20)

Gráfico com escalas corretas



1 - 12.706
2 - 4.303
5 - 2.517
10 - 2.228
20 - 2.086
30 - 2.042
100 - 1.984
200 - 1.972
300 - 1.968
N(0,1) - 1.964