

3. רגרסיה לינארית מרובה

ניתוח נתונים סטטיסטי · הרצאה 3

ד"ר עדי שריד

2026-10-03

על מה נדבר היום

- **תזכורת:** רגרסיה לינארית פשוטה, שיטת הריבועים הפחותים, ואיך קוראים את הפלט
- **רגרסיה מרובה:** כמה משתנים מסבירים, רישום מטריציוני, ופרשנות של “כל השאר קבוע”
- **מבחני השערות:** מבחן t לכל מקדם, מבחן F לכל המודל, ומבחן F חלקי
- R^2 ו- R^2 מתוקן
- **משתנים קטגוריים ואינטראקציות**
- **בדיקת ההנחות:** שאריות ומולטיקולינאריות
- **בחירת משתנים, רווחי סמך ורווחי חיזוי**

תזכורת: רגרסיה לינארית פשוטה (Simple linear regression)

השאלה: כמה מעבד צורכת כל אלף בקשות?

צוות התשתיות מתכנן קיבולת (capacity planning) לשירות חדש. בעשרה חלונות זמן נמדדו העומס (אלפי בקשות בדקה) וניצולת המעבד (CPU, באחוזים).

השאלות: בכמה עולה ניצולת המעבד עם כל אלף בקשות נוספות בדקה? והאם הקשר הזה "אמיתי", או שהוא יכול להיות מקרי?

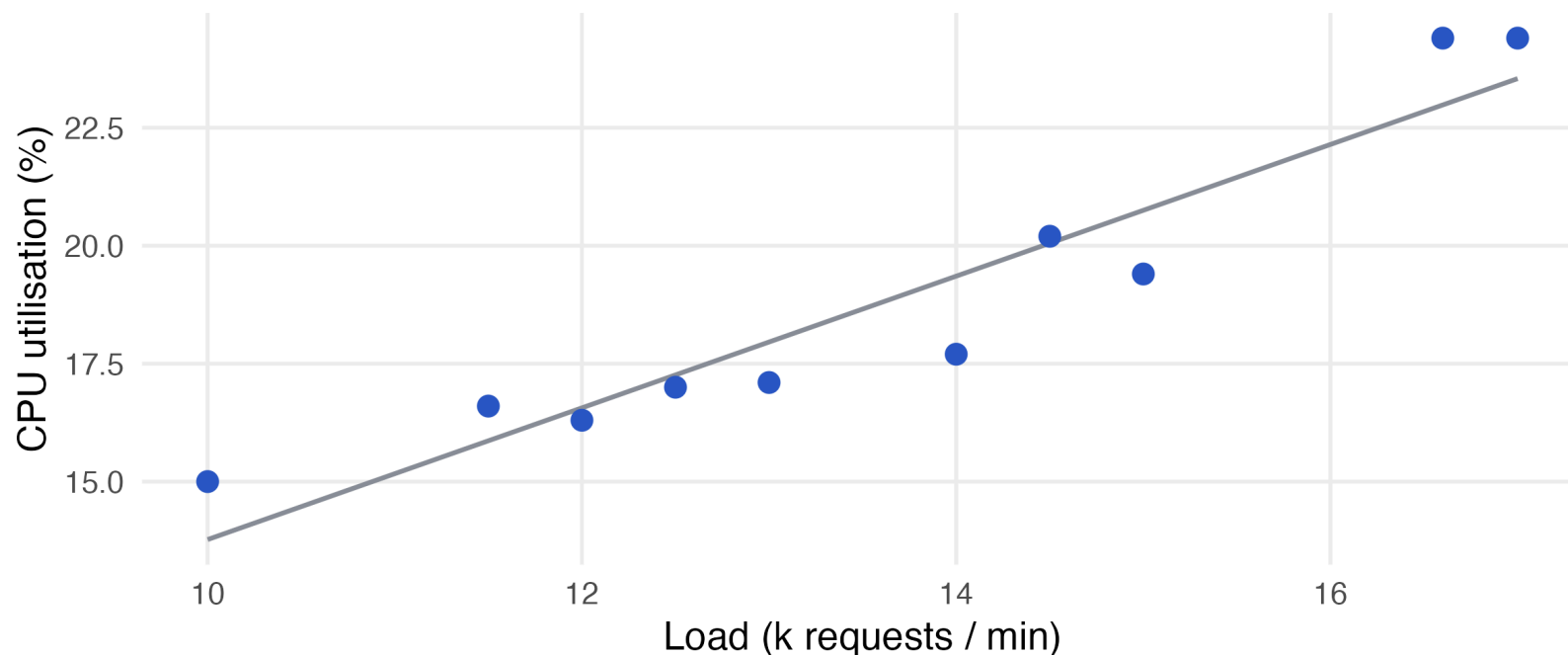
```
cpu <- tibble(  
  load = c(10, 11.5, 14.5, 12, 15, 16.6, 12.5, 13, 14, 17),      # k requests / min  
  util = c(15, 16.6, 20.2, 16.3, 19.4, 24.4, 17, 17.1, 17.7, 24.4) # CPU %  
)
```

הנתונים

```
ggplot(cpu, aes(load, util)) +  
  geom_smooth(method = "lm", se = FALSE, colour = nanas_pal[["grey"]], linewidth = 0.8) +  
  geom_point(size = 3) +  
  labs(title = "ככל שהעומס עולה, המעבד עובד קשה יותר",  
        subtitle = "עשרה חלונות זמן · הקו: קו הריבועים הפחותים",  
        x = "Load (k requests / min)", y = "CPU utilisation (%)")
```

ככל שהעומס עולה, המעבד עובד קשה יותר

עשרה חלונות זמן · הקו: קו הריבועים הפחותים



$$Y_i = \beta_0 + \beta_1 x_i + \varepsilon_i \quad i = 1, \dots, n$$

- β_0 הוא **החותך** (intercept) ו- β_1 הוא **השיפוע** (slope): בכמה משתנה התוחלת של Y כש- x עולה ביחידה אחת

- $E(Y | x) = \beta_0 + \beta_1 x$ ו- $V(Y | x) = \sigma^2$

ההנחות על השגיאות ε_i :

1. **לינאריות:** התוחלת של Y היא פונקציה לינארית של x
2. **אי-תלות:** השגיאות בלתי תלויות זו בזו
3. **שונות קבועה** (homoscedasticity): אותה σ^2 לכל ערך של x
4. **נורמליות:** $\varepsilon_i \sim N(0, \sigma^2)$. נדרשת למבחנים ולרווחי הסמך, לא לאמידה עצמה

שיטת הריבועים הפחותים (Least squares)

מחפשים את הקו שממזער את סכום ריבועי הסטיות האנכיות:

$$L(\beta_0, \beta_1) = \sum_{i=1}^n (y_i - \beta_0 - \beta_1 x_i)^2$$

גוזרים לפי β_0 ולפי β_1 ומשווים לאפס (משוואות נורמליות), ומקבלים:

$$\hat{\beta}_1 = \frac{S_{xy}}{S_{xx}} \quad \hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$$

$$S_{xx} = \sum_{i=1}^n (x_i - \bar{x})^2 = \sum x_i^2 - n\bar{x}^2 \quad S_{xy} = \sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y}) = \sum x_i y_i - n\bar{x}\bar{y}$$

חישוב ידני

```
cpu |> summarise(n = n(), x_bar = mean(load), y_bar = mean(util),  
                Sxx = sum((load - x_bar)^2),  
                Sxy = sum((load - x_bar) * (util - y_bar)),  
                Syy = sum((util - y_bar)^2))
```

```
# A tibble: 1 × 6  
  n x_bar y_bar  Sxx  Sxy  Syy  
<int> <dbl> <dbl> <dbl> <dbl> <dbl>  
1    10 13.6 18.8 45.0 62.8 97.9
```

$$\hat{\beta}_1 = \frac{62.80}{44.99} = 1.396 \quad \hat{\beta}_0 = 18.81 - 1.396 \cdot 13.61 = -0.188$$

$$SS_R = \hat{\beta}_1 S_{xy} = 87.66 \quad SS_E = S_{yy} - SS_R = 97.91 - 87.66 = 10.25 \quad \hat{\sigma}^2 = \frac{SS_E}{n - 2} = 1.281$$

$$se(\hat{\beta}_1) = \sqrt{\frac{\hat{\sigma}^2}{S_{xx}}} = \sqrt{\frac{1.281}{44.99}} = 0.169 \quad t = \frac{1.396}{0.169} = 8.27 \quad R^2 = \frac{SS_R}{S_{yy}} = 0.895$$

אותו חישוב ב-R

```
fit_cpu <- lm(util ~ load, data = cpu)
summary(fit_cpu)
```

Call:

```
lm(formula = util ~ load, data = cpu)
```

Residuals:

Min	1Q	Median	3Q	Max
-1.6544	-0.7095	-0.0565	0.8273	1.4163

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	-0.188	2.324	-0.08	0.94
load	1.396	0.169	8.27	3.4e-05 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.13 on 8 degrees of freedom

Multiple R-squared: 0.895, Adjusted R-squared: 0.882

F-statistic: 68.4 on 1 and 8 DF, p-value: 3.43e-05

איך קוראים את הפלט

- **Estimate**: המקדמים. כל אלף בקשות נוספות בדקה מעלות את ניצולת המעבד ב-1.4 נקודות אחוז בממוצע
- **Std. Error**: שגיאת התקן של כל מקדם, $se(\hat{\beta}_j)$
- **t value** ו-**Pr(>|t|)**: מבחן $H_0 : \beta_j = 0$, עם $t = \hat{\beta}_j / se(\hat{\beta}_j)$ ו- $n - 2 = 8$ דרגות חופש. לשיפוע $p = 0.00003$, ולכן **דוחים**: הקשר לא מקרי
- **Residual standard error**: $\hat{\sigma} = \sqrt{MSE} = 1.13$. זו הסטייה הטיפוסית של תצפית מהקו, ביחידות של Y
- **Multiple R-squared**: 89.5% מהשונות בניצולת המעבד מוסברים על ידי העומס
- **F-statistic**: מבחן לכל המודל. ברגסיה פשוטה $F = t^2$: $8.27^2 = 68.4$

מה עושים עם החותך? מירכוז (Centering)

החותך הוא הניצולת הצפויה כש- $x = 0$, הרחק מטווח הנתונים (10 עד 17). לכן השונות שלו גדולה:

$$V(\hat{\beta}_0) = \sigma^2 \left[\frac{1}{n} + \frac{\bar{x}^2}{S_{xx}} \right]$$

אם מחסירים מ- x את הממוצע שלו, $\bar{x} = 0$, והחותך הופך ל- $\bar{y}$: הניצולת הצפויה **בעומס הממוצע**.

```
lm(util ~ I(load - mean(load)), data = cpu) |> summary() |> coef()
```

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	18.810	0.3579	52.551	1.906e-11
I(load - mean(load))	1.396	0.1688	8.272	3.431e-05

- השיפוע לא השתנה, ושגיאת התקן של החותך ירדה מ-2.32 ל-0.358: $\hat{\sigma}/\sqrt{n} = 1.132/\sqrt{10} = 0.358$
- הקו הוא אותו קו. השתנתה רק השאלה שהחותך עונה עליה

פירוק השונות וטבלת ANOVA

$$\underbrace{\sum_i (y_i - \bar{y})^2}_{SS_T} = \underbrace{\sum_i (\hat{y}_i - \bar{y})^2}_{SS_R} + \underbrace{\sum_i (y_i - \hat{y}_i)^2}_{SS_E}$$

מקור	סכום ריבועים	דרגות חופש	ממוצע ריבועים	F
רגרסיה	SS_R	1	$MS_R = SS_R/1$	MS_R/MS_E
שגיאה	SS_E	$n - 2$	$MS_E = SS_E/(n - 2)$	
סך הכול	SS_T	$n - 1$		

```
anova(fit_cpu)
```

Analysis of Variance Table

Response: util

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
load	1	87.7	87.7	68.4	3.4e-05 ***
Residuals	8	10.2	1.3		

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

מקדם המתאם המרובה: $R^2 = SS_R/SS_T = 87.66/97.91 = 0.895$
שני הרכיבים אי-שליליים, ולכן תמיד $0 \leq R^2 \leq 1$.

רגרסיה לינארית מרובה (Multiple linear regression)

השאלה: מה קובע את זמן התגובה של מודל שפה?

חברה משתמשת ב-API של מודל שפה, בשלושה גדלים: `small`, `medium` ו-`large`. המשתמשים מתלוננים על המתנה ארוכה.

השאלות:

- כמה זמן מוסיף כל טוקן (token) בתשובה? וכל אלף טוקנים בקלט?
- האם מודל השפה הגדול **עצמו** איטי יותר, או שהוא פשוט כותב תשובות ארוכות יותר?
- כמה זמן תיקח קריאה עם 2,000 טוקנים בקלט ו-500 בפלט למודל השפה הגדול?

הנתונים: 250 קריאות מהלוגים (נתונים מדומים). לכל קריאה: גודל מודל השפה, מספר הטוקנים בקלט ובפלט, מספר המסמכים שצורפו לפרומפט (RAG), השעה ביום, וזמן התגובה בשניות.

הנתונים

```
set.seed(303) # own seed, so edits elsewhere in the lecture don't change this example
n_calls <- 250
sizes <- c("small", "medium", "large")
median_output <- c(small = 250, medium = 350, large = 500) # tokens
sec_per_token <- c(small = 0.010, medium = 0.018, large = 0.030)

llm <- tibble(
  model = factor(sample(sizes, n_calls, replace = TRUE, prob = c(0.5, 0.3, 0.2)), sizes),
  rag_docs = rpois(n_calls, 3), # documents added to the prompt
  input_tokens = round(300 + 550 * rag_docs + rlnorm(n_calls, log(400), 0.6)),
  output_tokens = round(rlnorm(n_calls, log(median_output[as.character(model)]), 0.5)),
  hour = sample(0:23, n_calls, replace = TRUE)
) |>
  mutate(latency = 0.8 + 0.25 * input_tokens / 1000 +
         sec_per_token[as.character(model)] * output_tokens + rnorm(n_calls, 0, 1),
         latency = round(latency, 2),
         input_k = input_tokens / 1000)

llm |> select(model, input_tokens, output_tokens, rag_docs, hour, latency) |> head(4)
```

A tibble: 4 × 6

	model	input_tokens	output_tokens	rag_docs	hour	latency
	<fct>	<dbl>	<dbl>	<int>	<int>	<dbl>
1	small	3585	155	5	21	1.71
2	large	1851	478	0	1	15.7
3	medium	3140	161	5	16	3.89
4	medium	2382	200	3	6	4.83

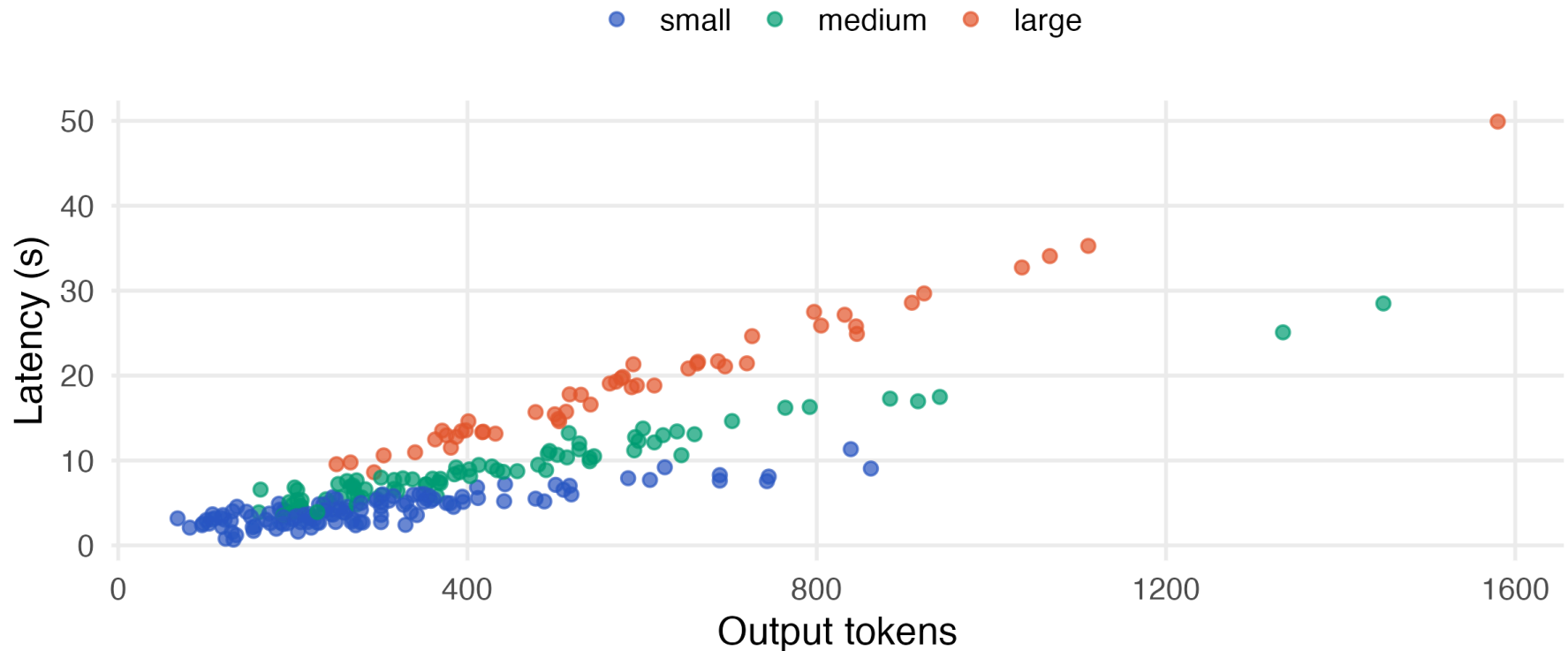
זמן התגובה מול אורך התשובה

```
ggplot(llm, aes(output_tokens, latency, colour = model)) +  
  geom_point(alpha = 0.7) +  
  scale_colour_manual(values = model_cols) +  
  labs(title = "תשובה ארוכה יותר, המתנה ארוכה יותר",  
        subtitle = "וכל מודל שפה עם שיפוע משלו",  
        x = "Output tokens", y = "Latency (s)", colour = NULL)
```

זמן התגובה מול אורך התשובה

תשובה ארוכה יותר, המתנה ארוכה יותר

וכל מודל שפה עם שיפוע משלו



$$Y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_k x_{ik} + \varepsilon_i \quad \varepsilon_i \sim N(0, \sigma^2)$$

- אותן הנחות כמו ברגרסיה פשוטה, עם k משתנים מסבירים (predictors)
 - הפרשנות של β_j : השינוי בתוחלת של Y כש- x_j עולה ביחידה אחת, וכל שאר המשתנים במודל נשארים קבועים
 - “לינארי” פירושו לינארי בפרמטרים β . אפשר להכניס x^2 , $\log x$ או מכפלה $x_1 x_2$, והמודל עדיין לינארי
- נתחיל משני משתנים מסבירים: הטוקנים בקלט (באלפים) והטוקנים בפלט.

```
fit2 <- lm(latency ~ input_k + output_tokens, data = llm)
coef(fit2)
```

(Intercept)	input_k	output_tokens
-1.75379	0.07958	0.02634

רישום מטריציוני (Matrix form)

$$\mathbf{y} = X\boldsymbol{\beta} + \boldsymbol{\varepsilon} \quad X = \begin{pmatrix} 1 & x_{11} & \cdots & x_{1k} \\ \vdots & \vdots & & \vdots \\ 1 & x_{n1} & \cdots & x_{nk} \end{pmatrix}_{n \times (k+1)}$$

ממזערים את $L(\boldsymbol{\beta}) = (\mathbf{y} - X\boldsymbol{\beta})'(\mathbf{y} - X\boldsymbol{\beta})$. הנגזרת היא $-2X'\mathbf{y} + 2X'X\boldsymbol{\beta}$, ומשווים אותה לאפס:

$$X'X\hat{\boldsymbol{\beta}} = X'\mathbf{y} \implies \hat{\boldsymbol{\beta}} = (X'X)^{-1}X'\mathbf{y}$$

- העמודה של אחדות ב- X היא החותך
- צריך ש- $X'X$ תהיה הפיכה: אף משתנה לא יכול להיות צירוף לינארי מדויק של האחרים

```
X <- model.matrix(~ input_k + output_tokens, data = llm) # adds the column of ones
y <- llm$latency
solve(t(X) %*% X) %*% t(X) %*% y # same as coef(fit2)
```

```
      [,1]
(Intercept) -1.75379
input_k      0.07958
output_tokens 0.02634
```

תכונות האומדים

- חסרי הטיה: $E(\hat{\beta}) = \beta$
- מטריצת השוניות: $V(\hat{\beta}) = \sigma^2(X'X)^{-1}$. נסמן $C = (X'X)^{-1}$
- אומד ל- σ^2 : $\hat{\sigma}^2 = MS_E = \frac{SS_E}{n - k - 1}$. מאבדים דרגת חופש על כל מקדם, כולל החותך

$$se(\hat{\beta}_j) = \sqrt{\hat{\sigma}^2 C_{jj}} \quad t = \frac{\hat{\beta}_j}{se(\hat{\beta}_j)} \sim t_{n-k-1} \quad \hat{\beta}_j \pm t_{1-\alpha/2, n-k-1} \cdot se(\hat{\beta}_j)$$

```
sigma2_hat <- sum(resid(fit2)^2) / (nrow(X) - ncol(X)) # SSE / (n - k - 1)
sqrt(diag(sigma2_hat * solve(t(X) %*% X)))           # = "Std. Error" in summary()
```

(Intercept)	input_k	output_tokens
0.7254406	0.2442116	0.0009362

הפלט המלא

```
summary(fit2)
```

Call:

```
lm(formula = latency ~ input_k + output_tokens, data = llm)
```

Residuals:

	Min	1Q	Median	3Q	Max
	-12.084	-1.880	-0.418	1.786	9.959

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	-1.753788	0.725441	-2.42	0.016 *
input_k	0.079583	0.244212	0.33	0.745
output_tokens	0.026340	0.000936	28.13	<2e-16 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.56 on 247 degrees of freedom

Multiple R-squared: 0.762, Adjusted R-squared: 0.76

F-statistic: 396 on 2 and 247 DF, p-value: <2e-16

איך קוראים את הפלט

- **output_tokens = 0.0263**: כל טוקן נוסף בתשובה מוסיף כ-26 מילישניות, כשמספר הטוקנים בקלט קבוע.
 $t = 28$, מובהק מאוד
- **input_k**: לא מובהק ($p = 0.75$). האם הקלט באמת לא משפיע? נחזור לזה
- **F-statistic: 396 on 2 and 247 DF**: מבחן לכל המודל. $k = 2$ ו- $n - k - 1 = 247$
- **R-squared = 0.762**: שני המשתנים יחד מסבירים 76% מהשונות בזמן התגובה
- **Residual standard error = 3.56**: קריאה טיפוסית רחוקה 3.6 שניות מהחיזוי. זה הרבה, כשהחציון הוא 6 שניות

מבחני השערות במודל המרובה

מבחן F לכל המודל

לפחות מקדם אחד שונה מאפס: H_1 $H_0 : \beta_1 = \beta_2 = \dots = \beta_k = 0$

מקור	סכום ריבועים	דרגות חופש	ממוצע ריבועים	F
רגרסיה	SS_R	k	$MS_R = SS_R/k$	MS_R/MS_E
שגיאה	SS_E	$n - k - 1$	$MS_E = SS_E/(n - k - 1)$	
סך הכול	SS_T	$n - 1$		

דוחים את H_0 אם $F > F_{1-\alpha; k, n-k-1}$.

- דחייה אומרת רק שלפחות משתנה אחד עוזר. היא לא אומרת איזה
- מבחני t-בדקים כל מקדם בנוכחות כל השאר. משתנה יכול להיות לא מובהק לבד, ובכל זאת חשוב

R^2 ו- R^2_{adj} מתוקן (Adjusted R^2)

$$R^2 = \frac{SS_R}{SS_T} = 1 - \frac{SS_E}{SS_T} \quad R^2_{adj} = 1 - \frac{SS_E / (n - k - 1)}{SS_T / (n - 1)} = 1 - (1 - R^2) \frac{n - 1}{n - k - 1}$$

- הבעיה של R^2 : הוא אף פעם לא יורד כשמוסיפים משתנה, גם אם המשתנה הוא רעש טהור
- R^2 המתוקן "מעניש" על כל משתנה נוסף. הוא עולה רק אם המשתנה מקטין את MS_E . למקסם אותו זה כמו למזער את MS_E

```
set.seed(8)
llm_noise <- llm |> mutate(noise1 = rnorm(n()), noise2 = rnorm(n()), noise3 = rnorm(n()))
fit_noise <- lm(latency ~ input_k + output_tokens + noise1 + noise2 + noise3, data = llm_noise)
rbind(fit2      = c(R2 = summary(fit2)$r.squared,      adj = summary(fit2)$adj.r.squared),
      fit_noise = c(R2 = summary(fit_noise)$r.squared, adj = summary(fit_noise)$adj.r.squared))
```

	R2	adj
fit2	0.7622	0.7603
fit_noise	0.7628	0.7580

תרגיל: השלימו את הפלט

פלט של רגרסיה עם $n = 30$ תצפיות ו- $k = 3$ משתנים מסבירים. חלק מהערכים נמחקו:

מקור	סכום ריבועים	דרגות חופש	ממוצע ריבועים	F
רגרסיה	600	(א)	(ב)	(ג)
שגיאה	(ד)	(ה)	(ו)	
סך הכול	860	29		

השלימו את הטבלה, וחשבו את R^2 , את R^2_{adj} ואת $\hat{\sigma}$. האם המודל מובהק ב- $\alpha = 0.05$? ($F_{0.95; 3, 26} = 2.98$)

משתנים קטגוריים (Categorical variables)

משתני דמה (Dummy variables)

למשתנה קטגורי עם m רמות מוסיפים $m - 1$ משתני דמה (אינדיקטורים). רמה אחת היא רמת הבסיס (reference level), ולה אין משתנה משלה.

$$D_{medium} = \begin{cases} 1 & \text{medium} \\ 0 & \text{אחרת} \end{cases} \quad D_{large} = \begin{cases} 1 & \text{large} \\ 0 & \text{אחרת} \end{cases}$$

$$Y = \beta_0 + \beta_1 x_{input} + \beta_2 x_{output} + \beta_3 D_{medium} + \beta_4 D_{large} + \varepsilon$$

- β_4 : בכמה קריאה למודל השפה הגדול איטית יותר מקריאה למודל השפה הקטן, באותו מספר טוקנים
- למה לא m משתנים? כי אז $D_{small} + D_{medium} + D_{large} = 1$, בדיוק כמו עמודת החותך, ו- $X'X$ לא הפיכה
- R יוצר את משתני הדמה בעצמו מכל `factor`. רמת הבסיס היא הרמה הראשונה (`levels`)

```
model.matrix(~ model, data = llm) |> head(3)
```

	(Intercept)	modelmedium	modellarge
1	1	0	0
2	1	0	1
3	1	1	0

האם מודל השפה הגדול איטי יותר?

```
fit_raw <- lm(latency ~ model, data = llm) # model only
fit3 <- lm(latency ~ input_k + output_tokens + model, data = llm) # same length of answer
rbind(raw = coef(fit_raw)[c("modelmedium", "modellarge")],
      adjusted = coef(fit3)[c("modelmedium", "modellarge")])
```

	modelmedium	modellarge
raw	4.930	15.182
adjusted	2.047	9.018

- בלי שלוט באורך התשובה: קריאה למודל השפה הגדול איטית ב-15.2 שניות מקריאה למודל השפה הקטן
- באותו מספר טוקנים: הפער יורד ל-9 שניות
- חלק מהפער נובע מכך שמודל השפה הגדול כותב תשובות ארוכות יותר. זה **משתנה מערבב** (confounder)
- “שליטה” במשתנה (controlling for) פירושה להשוות קריאות עם אותו ערך שלו

מבחן F חלקי (Partial F test)

השאלה: האם גודל מודל השפה משפיע בכלל? זו השערה על שני מקדמים יחד, $H_0 : \beta_3 = \beta_4 = 0$. אי אפשר לענות עליה עם שני מבחני t נפרדים.

משווים בין המודל המלא (full) לבין המודל המצומצם (reduced), שבו H_0 נכונה:

$$F = \frac{(SS_E^R - SS_E^F)/q}{SS_E^F/(n - k - 1)} \sim F_{q, n-k-1}$$

q הוא מספר המקדמים שנבדקים, ו- k הוא מספר המשתנים במודל המלא.

```
anova(fit2, fit3)
```

Analysis of Variance Table

Model 1: latency ~ input_k + output_tokens

Model 2: latency ~ input_k + output_tokens + model

	Res.Df	RSS	Df	Sum of Sq	F	Pr(>F)
1	247	3125				
2	245	911	2	2213	297	<2e-16 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

איך קוראים את הפלט

- **RSS**: סכום ריבועי השגיאות (SS_E) של כל מודל. משתני הדמה הורידו אותו מ-3125 ל-911
- **Df = 2**: הוספנו $q = 2$ מקדמים
- **בחישוב ידני** (דוחים את H_0 : גודל מודל השפה משפיע):

$$F = \frac{(3125 - 911)/2}{911/245} = 297$$

- כש- $q = 1$, מבחן F החלקי שקול למבחן t של אותו מקדם: $F = t^2$
- מבחן F לכל המודל הוא מקרה פרטי: המודל המצומצם הוא מודל עם חותך בלבד

אינטראקציות (Interactions)

האם הקווים מקבילים?

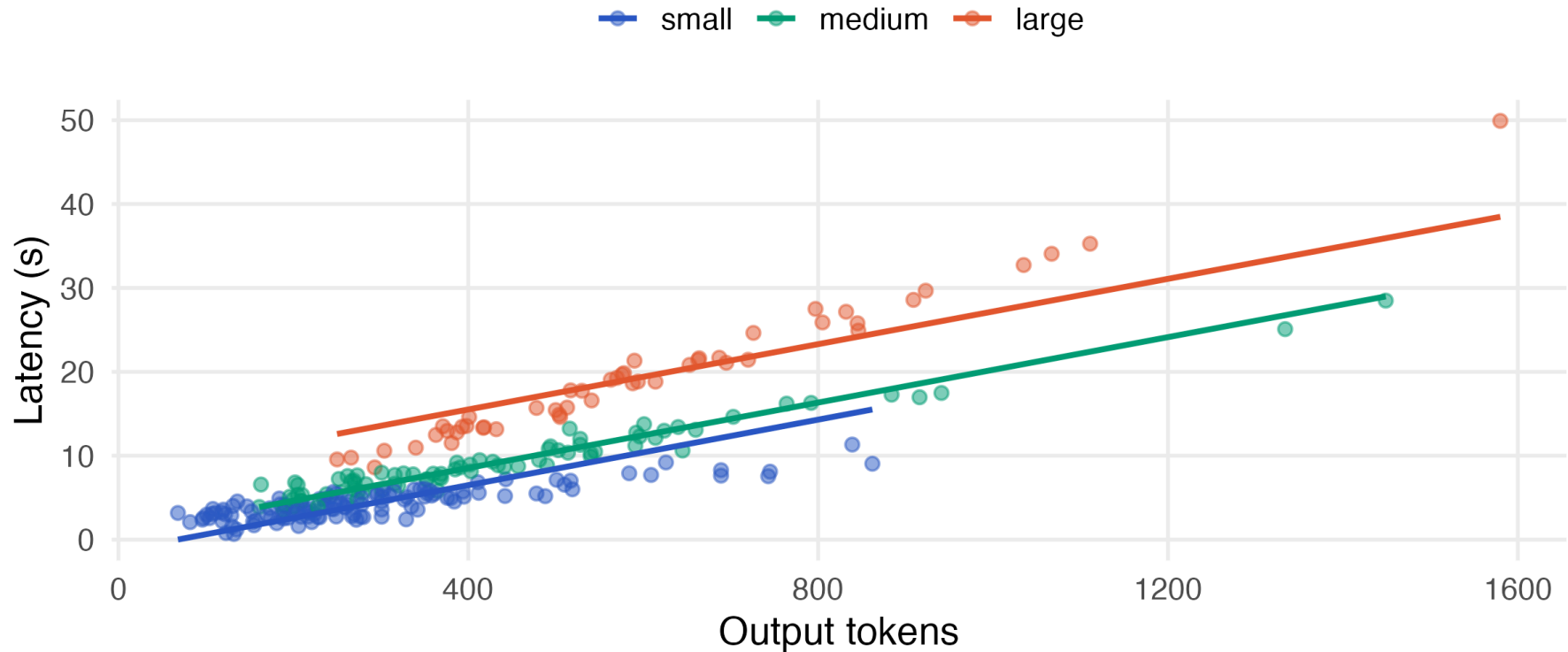
ברגרסיה `fit3` לכל מודל שפה יש חותך משלו, אבל אותו שיפוע: אותו זמן לכל טוקן בתשובה. זה סביר?

```
ggplot(augment(fit3), aes(output_tokens, colour = model)) +  
  geom_point(aes(y = latency), alpha = 0.5) +  
  geom_smooth(aes(y = .fitted), method = "lm", se = FALSE, linewidth = 1) +  
  scale_colour_manual(values = model_cols) +  
  labs(title = "קווים מקבילים לא מתאימים לנתונים",  
        subtitle = "הקווים: החיזוי של רגרסיה בלי אינטראקציה (בממוצע הקלט)",  
        x = "Output tokens", y = "Latency (s)", colour = NULL)
```

האם הקווים מקבילים?

קווים מקבילים לא מתאימים לנתונים

הקווים: החיזוי של רגרסיה בלי אינטראקציה (בממוצע הקלט)



מודל עם אינטראקציה

אינטראקציה פירושה שההשפעה של משתנה אחד תלויה בערך של משתנה אחר. כאן: הזמן לכל טוקן תלוי בגודל מודל השפה.

$$Y = \beta_0 + \beta_1 x_{in} + \beta_2 x_{out} + \beta_3 D_{med} + \beta_4 D_{lrg} + \beta_5 x_{out} D_{med} + \beta_6 x_{out} D_{lrg} + \varepsilon$$

מודל שפה		השיפוע של x_{out}
small		β_2
medium		$\beta_2 + \beta_5$
large		$\beta_2 + \beta_6$

ב-R, `output_tokens * model` הוא קיצור של `output_tokens: model + model`.

```
fit4 <- lm(latency ~ input_k + output_tokens * model, data = llm)
summary(fit4)
```

מודל עם אינטראקציה

Call:

```
lm(formula = latency ~ input_k + output_tokens * model, data = llm)
```

Residuals:

Min	1Q	Median	3Q	Max
-2.3999	-0.7069	0.0906	0.6431	2.3889

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	0.917850	0.224929	4.08	6.1e-05	***
input_k	0.220542	0.064231	3.43	0.0007	***
output_tokens	0.009887	0.000532	18.57	< 2e-16	***
modelmedium	-0.075828	0.278457	-0.27	0.7856	
modellarge	-0.613472	0.384691	-1.59	0.1121	
output_tokens:modelmedium	0.008135	0.000690	11.80	< 2e-16	***
output_tokens:modellarge	0.021030	0.000749	28.09	< 2e-16	***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.931 on 243 degrees of freedom

Multiple R-squared: 0.984, Adjusted R-squared: 0.984

F-statistic: 2.49e+03 on 6 and 243 DF, p-value: <2e-16

איך קוראים את הפלט

- זמן לכל טוקן בתשובה: 9.9 מילישניות במודל השפה הקטן, 18 בינוני ו-30.9 בגדול (האמת בסימולציה: 10, 18 ו-30)
- `input_k` מובהק עכשיו: 0.22 שניות לכל אלף טוקנים בקלט. במודלים הקודמים הוא "הוסתר" בגלל שגיאה גדולה ($\hat{\sigma} = 3.6$). עכשיו $\hat{\sigma} = 0.93$
- `modellarge`, `modelmedium` לא מובהקים, אבל **לא מוחקים אותם**. כשיש אינטראקציה, הם ההפרש בין מודלי השפה כשאורך התשובה **אפס**, ולא "ההשפעה של גודל מודל השפה"
- **כלל ההיררכיה**: אם אינטראקציה נמצאת במודל, משאירים בו גם את המשתנים שמרכיבים אותה

```
anova(fit3, fit4)
```

Analysis of Variance Table

Model 1: latency ~ input_k + output_tokens + model

Model 2: latency ~ input_k + output_tokens * model

	Res.Df	RSS	Df	Sum of Sq	F	Pr(>F)
1	245	911				
2	243	211	2	701 404	<2e-16	***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

בדיקת ההנחות (Diagnostics)

שאריות מול ערכים חזויים

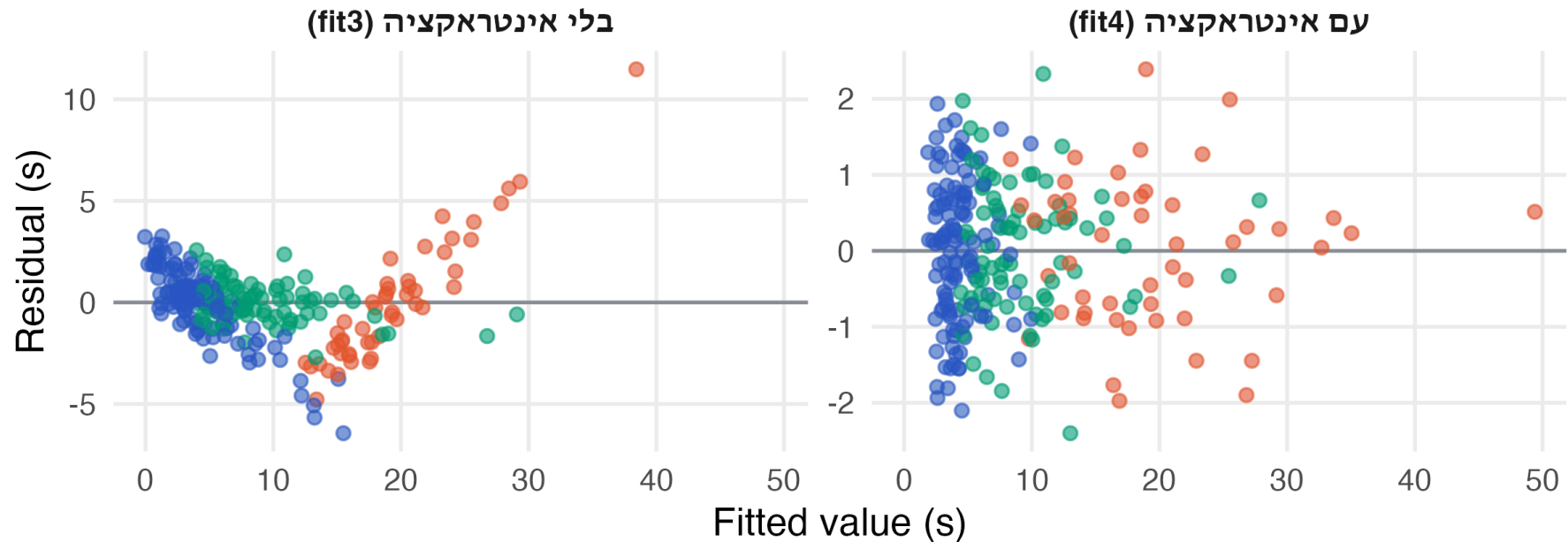
```
bind_rows(`אינטראקציה בלי (fit3)` = augment(fit3),
          `אינטראקציה עם (fit4)` = augment(fit4), .id = "fit") |>
  ggplot(aes(.fitted, .resid, colour = model)) +
  geom_hline(yintercept = 0, colour = nanas_pal[["grey"]]) +
  geom_point(alpha = 0.6) +
  facet_wrap(~ fit, scales = "free_y") +
  scale_colour_manual(values = model_cols) +
  labs(title = "השאריות חושפות את מה שהרגרסיה פספסה",
       subtitle = "משמאל: דפוס ברור לפי גודל מודל השפה · מימין: ענן אקראי סביב אפס",
       x = "Fitted value (s)", y = "Residual (s)", colour = NULL)
```

שאריות מול ערכים חזויים

השאריות חושפות את מה שהרגרסיה פספסה

משמאל: דפוס ברור לפי גודל מודל השפה · מימין: ענן אקראי סביב אפס

● small ● medium ● large



מה מחפשים בגרף השאריות

מה עושים	מה זה אומר	מה רואים
	ההנחות סבירות	ענן אקראי סביב אפס
מוסיפים x^2 , טרנספורמציה ל- x	הקשר לא לינארי	עקומה (קשת)
טרנספורמציה ל- Y (למשל $\log Y$)	השונות לא קבועה	משפך: הפיזור גדל עם $\hat{y}$
מוסיפים אותם למודל	חסר משתנה או אינטראקציה	קבוצות עם דפוסים שונים
בודקים אותו. לא מוחקים אוטומטית	ערך חריג	נקודה בודדת רחוקה

ועוד שתי בדיקות:

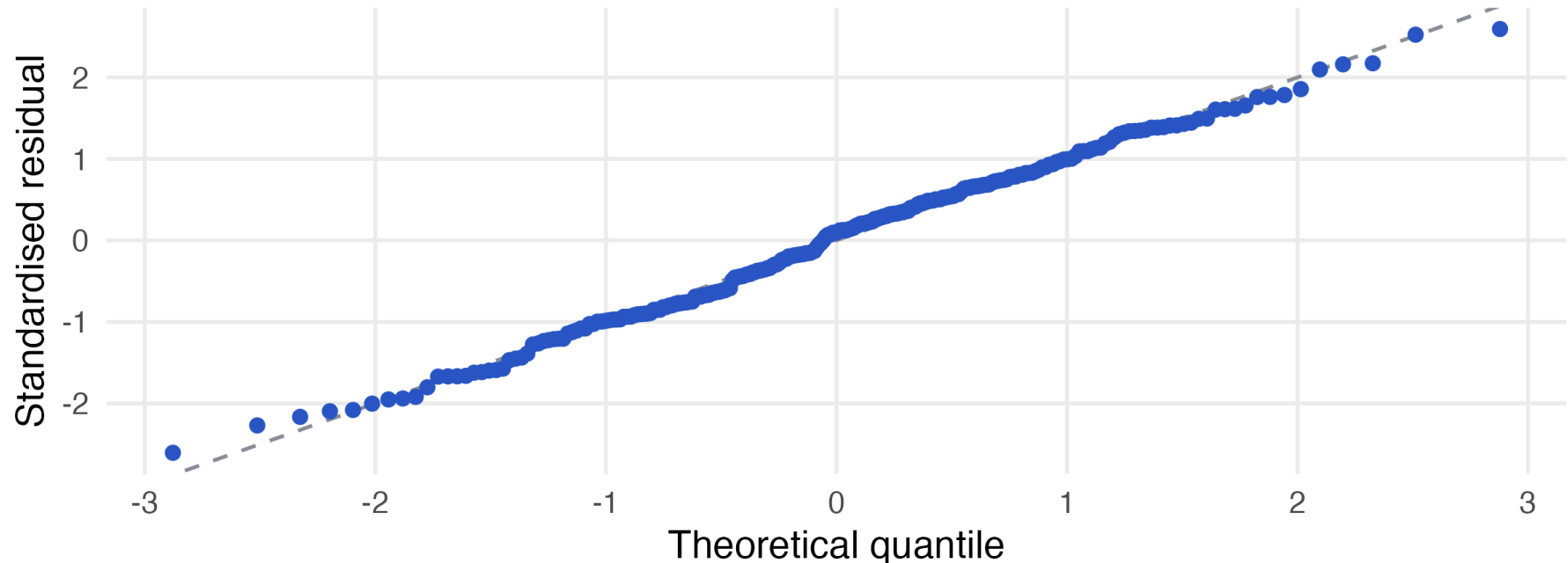
- **נורמליות:** גרף Q-Q של השאריות (`stat_qq()`). עם מדגם גדול, סטיות קטנות לא מזיקות למבחנים
- **אי-תלות:** אם הנתונים נאספו לאורך זמן, משרטטים את השאריות לפי סדר האיסוף

שאריות מול התפלגות נורמלית

```
ggplot(augment(fit4), aes(sample = .std.resid)) +  
  geom_abline(colour = nanas_pal[["grey"]], linetype = "dashed") +  
  stat_qq(colour = nanas_pal[["primary"]]) +  
  labs(title = "השאריות קרובות לנורמליות",  
        subtitle = "N(0,1) מול הכמויות של fit4 שאריות מתוקננות של",  
        x = "Theoretical quantile", y = "Standardised residual")
```

השאריות קרובות לנורמליות

שאריות מתוקננות של fit4 מול הכמויות של $N(0,1)$



מבחן Kolmogorov–Smirnov לשאריות

גרף Q-Q הוא בדיקה בעין. אפשר גם לבדוק את הנורמליות במבחן, עם מבחן KS מההרצאה על שיטות א-פרמטריות:

$$H_0 : N(0, 1) \text{ השאריות המתוקננות מתפלגות} \quad D = \max_x |F_n(x) - \Phi(x)|$$

```
ks.test(rstandard(fit4), "pnorm")
```

Asymptotic one-sample Kolmogorov–Smirnov test

```
data: rstandard(fit4)
D = 0.046, p-value = 0.7
alternative hypothesis: two-sided
```

- $D = 0.046 < 0.086 \approx 1.36/\sqrt{250}$, ולכן **לא דוחים** את H_0 . אין עדות לסטייה מנורמליות
- **הסתייגות:** השאריות מחושבות מהמקדמים שאמדנו, ולכן ה-p-value רק בקירוב (כמו ב-KS עם פרמטרים אמודים). מבחן ייעודי לנורמליות הוא `shapiro.test()`
- עם מדגם גדול מאוד, KS ידחה גם סטייה זניחה. עם מדגם קטן, הוא לא יזהה גם סטייה גדולה. לכן מסתכלים גם על הגרף

מולטיקולינאריות (Multicollinearity)

מספר המסמכים מ-RAG קובע כמעט לגמרי את מספר הטוקנים בקלט: המתאם ביניהם הוא 0.94. מה קורה כשמכניסים את שניהם?

```
fit5 <- lm(latency ~ input_k + rag_docs + output_tokens * model, data = llm)
rbind(fit4 = coef(summary(fit4))["input_k", 1:2],
      fit5 = coef(summary(fit5))["input_k", 1:2])
```

	Estimate	Std. Error
fit4	0.2205	0.06423
fit5	0.4372	0.18807

- שגיאת התקן של `input_k` גדלה פי 2.9. הרגרסיה לא יכולה להפריד בין שני משתנים שזזים יחד
 - המקדמים "לא יציבים": שינוי קטן בנתונים מזיז אותם הרבה. **החזוי** כמעט לא נפגע. **הפרשנות** נפגעת.
- מדד VIF** (Variance Inflation Factor): R_j^2 הוא ה- R^2 של רגרסיה של x_j על כל שאר המשתנים המסבירים.

$$VIF_j = \frac{1}{1 - R_j^2}$$

```
r2_input <- summary(lm(input_k ~ rag_docs + output_tokens * model, data = llm))$r.squared
1 / (1 - r2_input)      # VIF of input_k. Rule of thumb: above 5-10 is a problem
```

```
[1] 8.672
```

בחירת משתנים (Variable selection)

שלוש גישות

- **קדימה (Forward):** מתחילים ממודל בלי משתנים. בכל צעד מוסיפים את המשתנה שמשפר את הקריטריון הכי הרבה. עוצרים כשאף הוספה לא משפרת
 - **אחורה (Backward):** מתחילים מכל המשתנים. בכל צעד מוציאים את המשתנה שתורם הכי מעט, כל עוד ההוצאה לא פוגעת בקריטריון
 - **בצעדים (Stepwise):** כמו קדימה, אבל אחרי כל הוספה בודקים אם כדאי להוציא משתנה שכבר נכנס
- קריטריונים נפוצים:** R^2_{adj} , ה-p-value של מבחן F חלקי, או AIC (מדד שמאזן בין טיב ההתאמה למספר הפרמטרים. קטן יותר = טוב יותר)

```
fit_all <- lm(latency ~ input_k + rag_docs + hour + output_tokens * model, data = llm)
fit_step <- step(fit_all, direction = "both", trace = 0) # stepwise by AIC
formula(fit_step)
```

```
latency ~ input_k + hour + output_tokens + model + output_tokens:model
```

מה יכול להשתבש בבחירה אוטומטית

```
coef(summary(fit_step))["hour", ]
```

Estimate	Std. Error	t value	Pr(> t)
-0.014070	0.008184	-1.719055	0.086883

- **hour נשאר** במודל, אף שבסימולציה לשעה אין שום השפעה על זמן התגובה. ה-p-value שלו אפילו קרוב ל-0.05
- כשבדקים הרבה משתנים, חלקם ייראו מועילים במקרה. האלגוריתם בוחר בדיוק אותם
- ה-p-values **אחרי בחירה אופטימיים מדי**. המודל נבחר על סמך אותם נתונים שעליהם הוא נבדק
- **עדיף:** לבנות את המודל מתוך הבנה של המנגנון, ולבדוק אותו על נתונים שלא השתתפו בבנייתו. בפועל מחלקים את הנתונים לשני חלקים: **סט אימון** (train) שעליו בוחרים ואומדים את המודל, ו**סט בדיקה** (test) שעליו בודקים את החיזוי. בקורס בלמידת מכונה תלמדו על כך יותר

חיזוי (Prediction)

רווח סמך מול רווח חיזוי

לנקודה חדשה $\mathbf{x}_0 = (1, x_{01}, \dots, x_{0k})'$, החיזוי הוא $\hat{y}_0 = \mathbf{x}_0' \hat{\beta}$. יש שתי שאלות שונות:

- **רווח סמך** (confidence interval): איפה התוחלת $E(Y | \mathbf{x}_0)$, זמן התגובה הממוצע של כל הקריאות האלה?
- **רווח חיזוי** (prediction interval): איפה תיפול **קריאה בודדת** חדשה?

$$\text{CI: } \hat{y}_0 \pm t_{1-\alpha/2, n-k-1} \sqrt{\hat{\sigma}^2 \mathbf{x}_0' (X' X)^{-1} \mathbf{x}_0} \quad \text{PI: } \hat{y}_0 \pm t_{1-\alpha/2, n-k-1} \sqrt{\hat{\sigma}^2 (1 + \mathbf{x}_0' (X' X)^{-1} \mathbf{x}_0)}$$

ה-1 הנוסף ברווח החיזוי הוא השונות של התצפית החדשה עצמה. גם עם אינסוף נתונים הוא לא נעלם.

ברגרסיה פשוטה, $\mathbf{x}_0' (X' X)^{-1} \mathbf{x}_0 = \frac{1}{n} + \frac{(x_0 - \bar{x})^2}{S_{xx}}$, הרווחים צרים ביותר ב- $\bar{x}$ ומתרחבים ככל שמתרחקים ממנו.

חזרה לשאלה: כמה זמן תיקח הקריאה?

2,000 טוקנים בקלט, 500 בפלט, מודל השפה הגדול:

```
new_call <- tibble(input_k = 2, output_tokens = 500, model = "large")
rbind(confidence = predict(fit4, new_call, interval = "confidence")[1, ],
      prediction = predict(fit4, new_call, interval = "prediction")[1, ])
```

	fit	lwr	upr
confidence	16.2	15.92	16.49
prediction	16.2	14.35	18.06

- **בממוצע**, קריאות כאלה לוקחות בין 15.9 ל-16.5 שניות
- **קריאה בודדת** תיקח, ב-95% מהמקרים, בין 14.3 ל-18.1 שניות. זה המספר שרלוונטי להתחייבות לזמן תגובה (SLA)

איך זה נראה

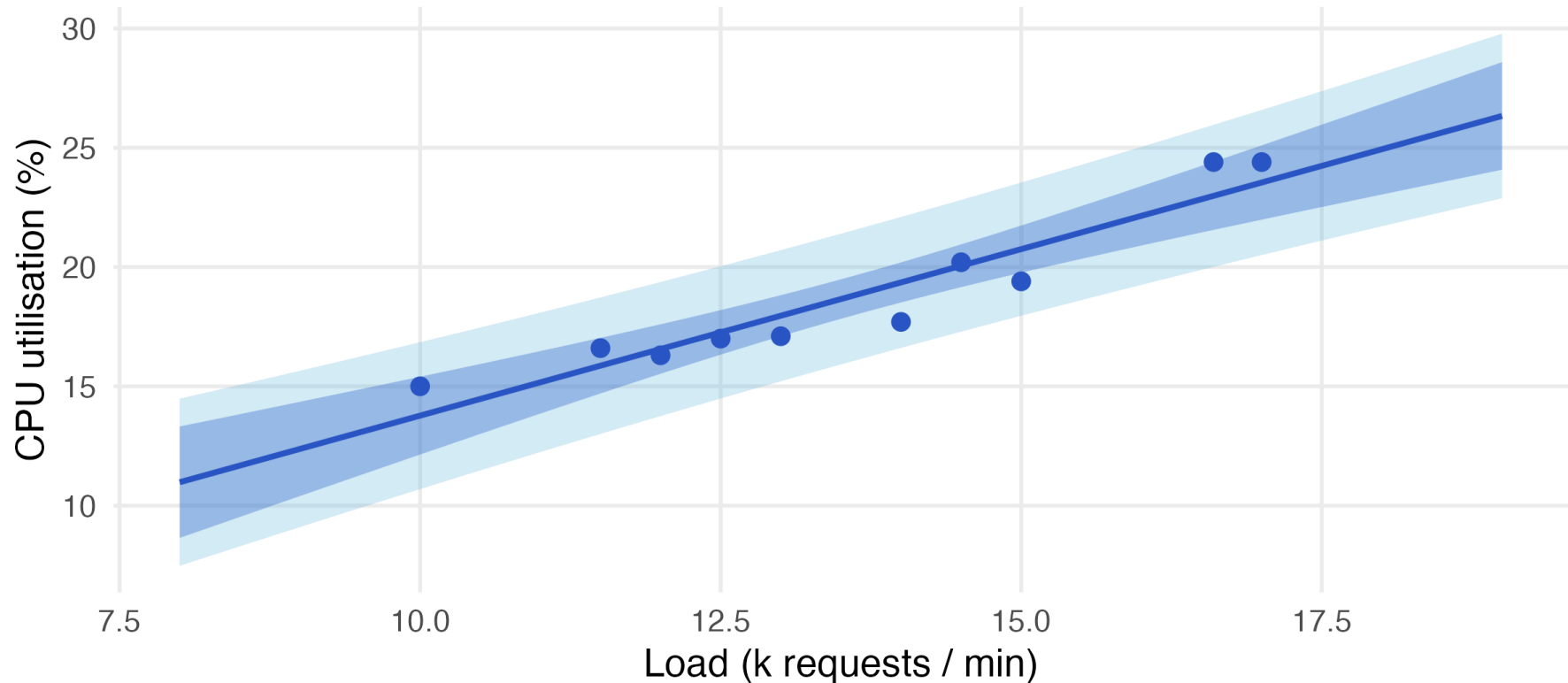
```
grid_cpu <- tibble(load = seq(8, 19, by = 0.1))
bands <- bind_cols(grid_cpu,
  predict(fit_cpu, grid_cpu, interval = "confidence") |> as_tibble(),
  predict(fit_cpu, grid_cpu, interval = "prediction") |> as_tibble() |>
  select(pi_lwr = lwr, pi_upr = upr))

ggplot(bands, aes(load)) +
  geom_ribbon(aes(ymin = pi_lwr, ymax = pi_upr), fill = nanas_pal[["sky"]], alpha = 0.25) +
  geom_ribbon(aes(ymin = lwr, ymax = upr), fill = nanas_pal[["primary"]], alpha = 0.35) +
  geom_line(aes(y = fit), colour = nanas_pal[["primary"]], linewidth = 1) +
  geom_point(aes(load, util), data = cpu, size = 2.5) +
  labs(title = "רווח החיזוי רחב יותר מרווח הסמך, ושניהם מתרחבים בקצוות",
    subtitle = "דוגמת המעבד • כהה: רווח סמך 95% לתוחלת • בהיר: רווח חיזוי 95% לתצפית בודדת",
    x = "Load (k requests / min)", y = "CPU utilisation (%)")
```

איך זה נראה

רווח החיזוי רחב יותר מרווח הסמך, ושניהם מתרחבים בקצוות

דוגמת המעבד · כהה: רווח סמך 95% לתוחלת · בהיר: רווח חיזוי 95% לתצפית בודדת



מה יכול להשתבש

חמש מלכודות

1. **אקסטרפולציה:** לפי מודל המעבד, ניצולת של 80% תגיע בעומס של כ-57 אלף בקשות בדקה. אבל מדדנו רק בין 10 ל-17. מעבדים בעומס גבוה מתנהגים אחרת (תורים, throttling), והמודל לא יודע על זה כלום
2. **מתאם אינו סיבתי:** נתוני הלוגים הם תצפיתיים. אם משתמשים בוחרים במודל השפה הגדול למשימות קשות, ייתכן שגם משהו אחר במשימות האלה מאט אותן. כדי לדעת בוודאות, צריך ניסוי (על כך בהרצאה על תכנון ניסויים)
3. **R^2 גבוה אינו הוכחה:** הוא תמיד עולה כשמוסיפים משתנים. משווים מודלים לפי R^2_{adj} , מבחן F חלקי, או ביצועים על נתונים חדשים
4. **מולטיקולינאריות:** מקדמים לא יציבים ושגיאות תקן גדולות, אף שהחיזוי בסדר
5. **מודל שגוי מסתיר אפקטים:** `input_k` נראה חסר השפעה עד שהוספנו את גודל מודל השפה והאינטראקציה. **תמיד** מסתכלים על השאריות

סיכום

מה לקחת מההרצאה

השאלה	הכלי	R-ב
מה הקשר בין Y לכל משתנה, כשהשאר קבועים?	$\hat{\beta} = (X'X)^{-1}X'y$	<code>lm(y ~ x1 + x2), summary()</code>
האם משתנה בודד תורם?	מבחן t , t_{n-k-1}	<code>summary()</code>
האם המודל כולו מסביר משהו?	מבחן F , $F_{k, n-k-1}$	שורה אחרונה ב- <code>summary()</code>
האם קבוצת משתנים תורמת?	מבחן F חלקי, $F_{q, n-k-1}$	<code>anova(reduced, full)</code>
משתנה קטגורי עם m רמות	$m - 1$ משתני דמה	<code>factor</code>
האם השיפוע תלוי בקבוצה?	אינטראקציה	<code>x * group</code>
האם ההנחות מתקיימות?	שאריות, Q-Q, VIF	<code>augment(), stat_qq()</code>
מה צפוי לתצפית חדשה?	רווח סמך / חיזוי	<code>predict(interval =)</code>

- כל מקדם מתפרש **בהינתן שאר המשתנים במודל**. כשמשנים את המודל, משתנה גם הפרשנות
- את המודל בונים מתוך הבנה של המנגנון. אלגוריתם בחירה הוא כלי עזר, לא תחליף