Programare probabilistă folosind PyMC - continuare

Cursul 5

Programare şi modelare probabilistă - anul III

Facultatea de Informatică, UAIC

e-mail: adrian.zalinescu@uaic.ro

web: https://sites.google.com/view/fiicoursepmp/home

6 Noiembrie 2023

Plan

1. Inferenţă Gaussiană
   - Verificarea predictivă a posteriori
2. Inferenţă robustă
3. Comparaţia grupurilor
   - Diferenţa standardizată a mediilor
   - Probabilitatea superiorităţii
4. Modele ierarhice

Distribuţii Gaussiene

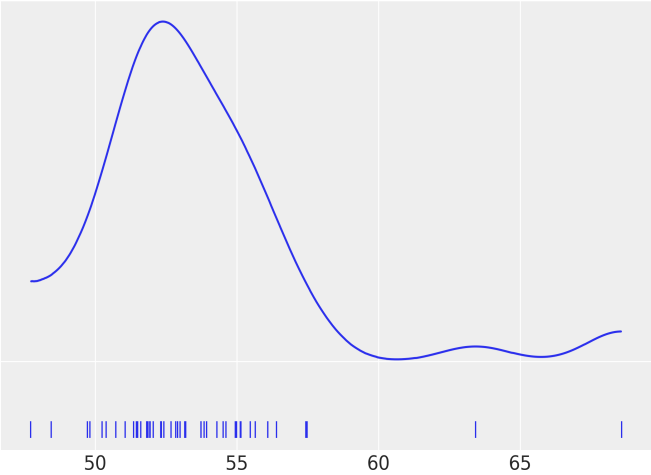
- Am introdus principalele noţiuni Bayesiene folosind modelul beta-binomial datorită simplicităţii sale
- Un alt model foarte simplu este modelul *Gaussian* sau modelul *normal*.
- Distribuţiile Gaussiene (sau normale) sunt foarte folosite datorită faptului că este uşor de lucrat cu ele din punct de vedere matematic: *conjugata a priori* a unei distribuţii Gaussiene este tot o distribuţie Gaussiană.
- Pe de altă parte, foarte multe fenomene pot fi bine aproximate folosind acest tip de distribuţii: aproape de fiecare dată când vrem să măsurăm media unor cantităţi, folosind un eşantion mare, aceasta este distribuită (aproximativ) normal, datorită teoremei limită centrală (TLC).
- În realitate, foarte multe fenomene sunt bazate pe medii:
  - de exemplu, *înălţimea* este rezultatul multor factori de mediu şi genetici, aşadar este de presupus că înălţimile indivizilor urmează o distribuţie Gaussiană.

Inferenţă Gaussiană

Considerăm următorul exemplu din *rezonanţa magnetică nucleară* (*RMN*), folosită la studiul moleculelor.

- RMN ne permite să măsurăm diverse cantităţi observabile ce sunt legate de proprietăţi moleculare interesante, dar inobservabile.
- Una dintre aceste proprietăţi observabile este cunoscută drept *deplasare chimică* a unor nuclee ale anumitor tipuri de atomi.
  - "deplasare" se referă la faptul că se măsoară o anumită diferenţă faţă de o anumită valoare de referinţă;
  - "chimică" deoarece această deplasare depinde de mediul chimic al nucleelor analizate.

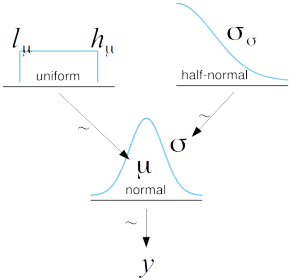
O distribuţie Gaussiană pare să fie o bună aproximare pentru această variabilă.
Un grafic KDE pentru 48 de deplasări chimice:



Un model rezonabil este deci:

$$
\begin{aligned}
\mu &\sim \mathcal{U}(l,h)\text{: distribuţie uniformă;}\\
\sigma &\sim \left|\mathcal{N}(0,\sigma_{sigma})\right|\text{: distribuţie “half-normal”;}\\
y &\sim \mathcal{N}(\mu,\sigma)
\end{aligned}
$$

Pentru vizualizarea modelui, putem folosi diagrame de *tip Kruschke*:

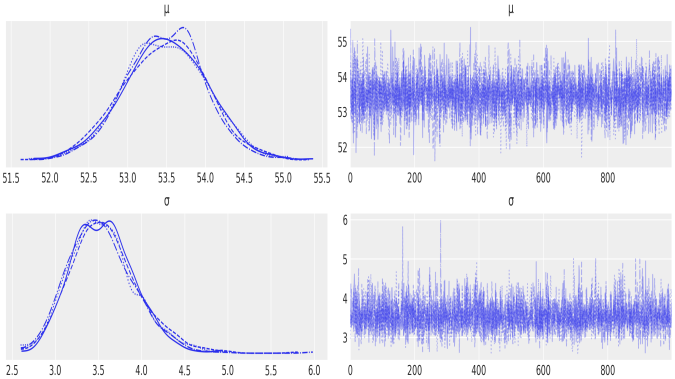


Alegem valorile posibile ale lui μ şi σ, folosind distribuţii a priori ce reflectă gradul nostru de ignoranţă (asupra acestora):

- Pentru μ, o opţiune ar fi să alegem o distribuţie uniformă cu $l = 0$ şi $h = 100$, deoarece ştim că nu este fizic posibil să obţinem valori în afara intervalului $[0, 100]$;
- O altă posibilitate este să restrângem intervalul la $l = 40$ şi $h = 75$, bazându-ne pe datele pe care le avem la dispoziţie;
- Pentru σ (care este întotdeauna > 0) putem alege $\sigma_{sigma} = 10$, care este o valoare suficient de mare pentru datele noastre.

Folosind PyMC, obţinem:

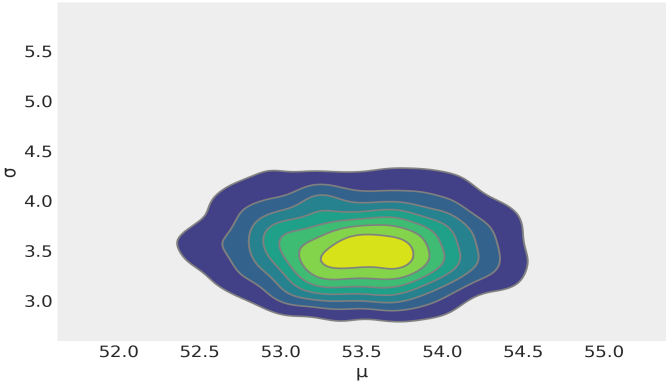
```
with pm.Model() as model_g:
    μ = pm.Uniform('μ', lower=40, upper=70)
    σ = pm.HalfNormal('σ', sigma=10)
    y = pm.Normal('y', mu=μ, sigma=σ, observed=data)
    idata_g = pm.sample(1000, return_inferencedata=True)
```



Sunt două grafice, pentru fiecare parametru (μ şi σ).

Dacă dorim să vizualizăm distribuţia bidimensională, folosim `plot_pair` din libraria `arviz`:

```
az.plot_pair(idata_g, kind='kde')
```



Verificarea predictivă a posteriori

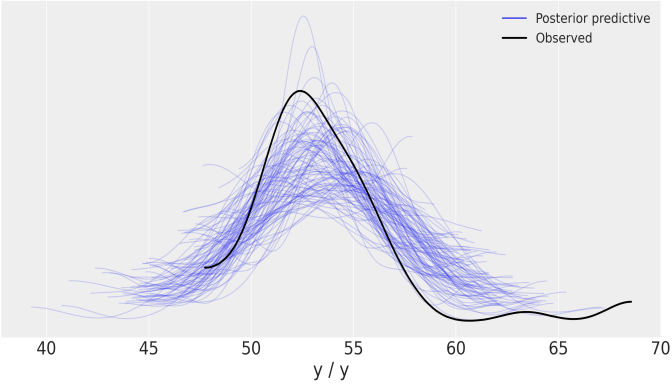
- Odată determinată distribuţia a posteriori, o putem folosi pentru a:
  - simula date, deci *a face predicţii* şi
  - a folosi aceste predicţii pentru *a verifica* modelul.
- În PyMC, este uşor de eşantionat din distribuţia a posteriori, folosind funcţia `sample_posterior_predictive`:

```
pm.sample_posterior_predictive(idata_g, model=model_g, extend_inferencedata=True)
```

- Variabila `y_pred_g` este un dicţionar, cu *cheile* (`keys`) fiind numele variabilelor observate în modelul nostru, în timp ce *valorile* (`values`) este un `array` de `shape=(samples, size)` - în cazul nostru, `(1000, len(data))`.

Putem folosi `az.plot_ppc` pentru o verificare vizuală:

```
ax = az.plot_ppc(idata_g, num_pp_samples=100, figsize=(12, 6), mean=False)
```



Predicţii bazate pe date (remember: Cursul 2)
Odată ce distribuţia a posteriori a fost generată pe baza analizei Bayesiene, o proprietate foarte utilă este utilizarea acestei pentru a face/genera predicţii ale sistemului:

- Distribuţia *predictivă a posteriori* este dată de:

$$p(\hat{y}|y) = \int p(\hat{y}|\theta)p(\theta|y)d\theta$$

- Această integrală se aproximează printr-un proces în doi paşi:
  - se generează o valoare θ conform distribuţiei a posteriori, $p(\cdot|y)$;
  - se *hrăneşte* (*feed*) verosimilitatea $p(\cdot|\theta)$ cu această valoare θ, generând o valoare prezisă $\tilde{y}$.
- Pe lângă generarea de predicţii, acest procedeu serveşte şi la analiza critică a modelului, prin compararea setului de date y cu setul de date prezise $\tilde{y}$.
- Metoda de mai sus se mai numeşte *verificare predictivă a posteriori* (*ulterioară*).

Inferenţă robustă

O obiecţie pe care putem să o aducem acestui model este că

- am presupus o distribuţie normală pentru verosimilitate, dar
- avem două date la marginea distribuţiei, care fac ca prezumţia de normalitate să fie puţin forţată.

Putem să ne imaginăm că cele două puncte primesc o pondere *prea mare* atunci când dorim să calculăm parametrii μ şi σ ai distribuţiei normale.

O primă opţiune este să considerăm aceste puncte ca *aberante* (*outliers*) şi să le eliminăm din analiza datelor.

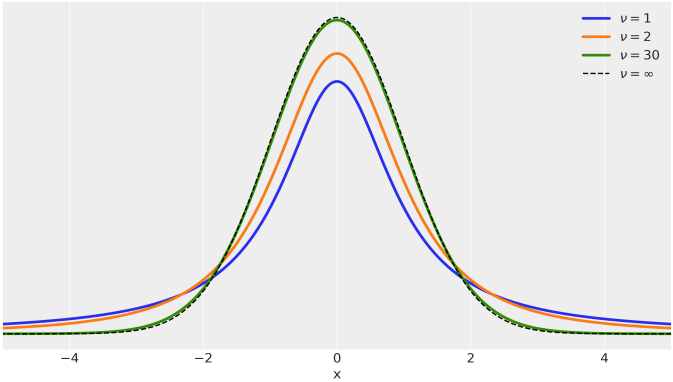
- Un motiv este dat de faptul că ele pot să provină dintr-o proastă funcţionare a achipamentului sau dintr-o eroare umană.
- Se poate automatiza procesul de eliminare a valorilor aberante, de exemplu prin considerarea acestora ca fiind:
  - valorile ce nu se află în intervalul $(Q_1 - 1.5 * IQR, Q_3 + 1.5 * IQR)$, unde Q_1, Q_3 sunt prima, respectiv a treia cvartilă, iar $IQR := Q_3 - Q_1$;
  - valorile ce nu se află în intervalul $(m - 2 * sd, m + 2 * sd)$, unde m şi sd reprezintă media, respectiv deviaţia standard a datelor.

Distribuţia Student

O altă opţiune este să ţinem cont de aceste valori şi să modificăm modelul, considerând distribuţii a priori şi verosimilităţi modificate.
Astfel, în loc de distribuţia normală, putem considera distribuţia t a lui Student:

- Aceasta are 3 parametri: *media*, *scara* (analoagă deviaţiei standard) şi *nr. de grade de libertate*, notat de obicei cu ν (în intervalul $[0, +\infty]$).
- ν se mai numeşte şi parametrul de normalitate, deoarece controlează cât de mult seamănă distribuţia t cu distribuţia normală.
- Pentru $\nu = 1$, aceasta se mai numeşte şi distribuţia *Cauchy* sau *Lorentz*; capetele acesteia sunt mai groase ca cele ale distribuţiei normale: 95% de valori generate cu o astfel de distribuţie sunt în intervalul $[-12.7, 12.7]$ (faţă de intervalul $[-1.96, 1.96]$ pentru cea normală).
- Distribuţia normală coincide cu t în cazul în care $\nu = +\infty$.

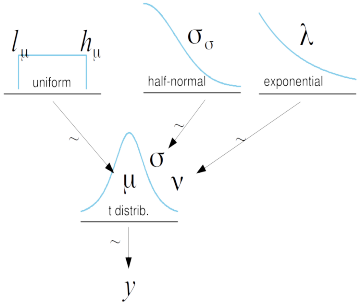
În libraria `stats`, o distribuţie Student poate fi apelată prin `stats.t(loc=0, scale=1, df=1)`.



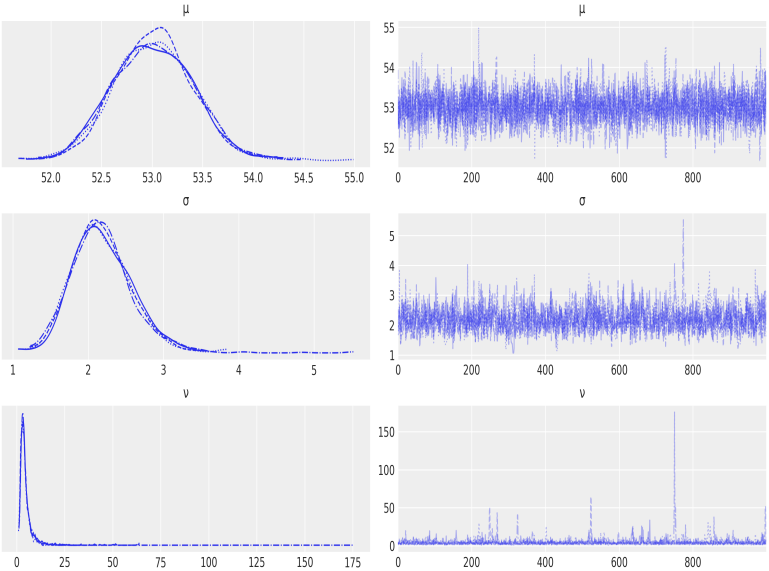
Modelul devine:

$$
\begin{aligned}
\mu &\sim U(l,h);\\
\sigma &\sim \left|\mathcal{N}(0,\sigma_{sigma})\right|;\\
\nu &\sim \mathrm{Exp}(\lambda);\\
y &\sim t(\mu,\sigma,\nu)
\end{aligned}
$$

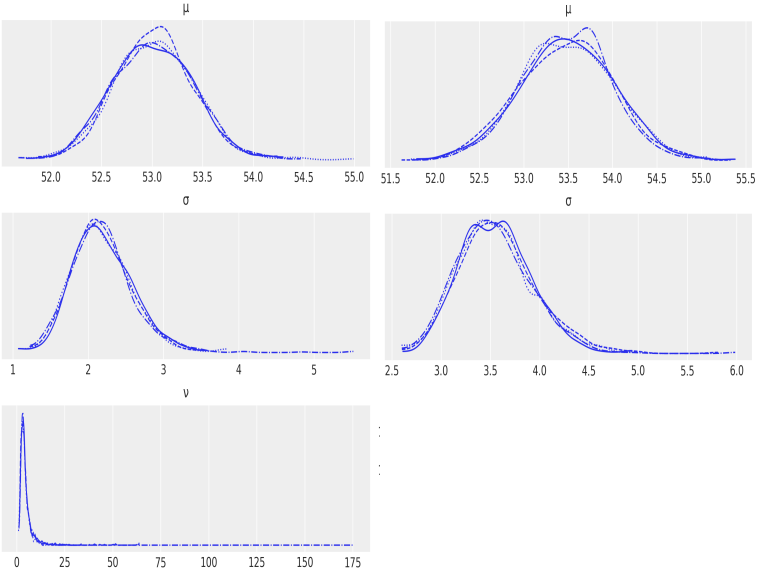
vizualizat cu o diagramă de *tip Kruschke*:



- Vom alege ca distribuţie a priori pentru ν o distribuţie exponenţială de medie 30, deoarece pentru $\nu = 30$, distribuţia Student este foarte similară cu cea normală.
- Prin urmare, luăm $\lambda = 1/30$.
- În PyMC, operăm câteva modificări ale modelului:
-
```
with pm.Model() as model_t:
    μ = pm.Uniform('μ', 40, 75)
    σ = pm.HalfNormal('σ', sigma=10)
    ν = pm.Exponential('ν', 1/30)
    y = pm.StudentT('y', mu=μ, sigma=σ, nu=ν, observed=data)
    idata_t = pm.sample(1000, return_inferencedata=True)
```


μ
52.0
52.5
53.0
53.5
54.0
54.5
55.0
μ
55
54
53
52
0
200
400
600
800
σ
1
2
3
4
5
σ
5
4
3
2
1
0
200
400
600
800
ν
0
25
50
75
100
125
150
175
ν
150
100
50
0
0
200
400
600
800

comparat cu modelul Gaussian:



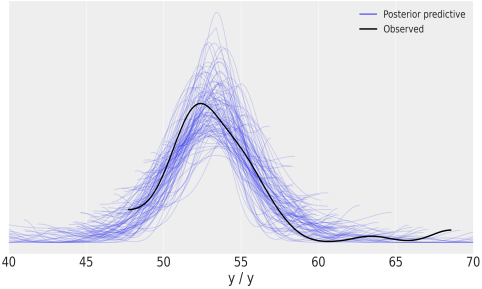
Comparaţia cu `az.summary`:

	mean	sd	hdi_3%	hdi_97%	mcse_mean	mcse_sd	ess_bulk	ess_tail	r_hat
μ	53.484	0.527	52.475	54.449	0.009	0.006	3423.0	2504.0	1.0
σ	3.554	0.380	2.881	4.277	0.007	0.005	3394.0	2449.0	1.0

	mean	sd	hdi_3%	hdi_97%	mcse_mean	mcse_sd	ess_bulk	ess_tail	r_hat
μ	53.013	0.398	52.282	53.766	0.007	0.005	3107.0	2434.0	1.0
σ	2.187	0.401	1.492	2.959	0.009	0.006	1993.0	2301.0	1.0
ν	4.766	6.159	1.186	9.482	0.185	0.131	2025.0	1740.0	1.0

Comparaţia prin verificarea predictivă a posteriori:

```
pm.sample_posterior_predictive(idata_t, model=model_t, extend_inferencedata=True)
az.plot_ppc(idata_t, figsize=(12, 6), num_pp_samples=100)
plt.xlim(40, 70)
```



- Datele se potrivesc mai bine, iar unele date generate sunt mai depărtate de grosul acestora.
- Avem o estimare mai robustă, deoarece outlier-ii au efectul de a-l descreşte pe ν în loc de a trage media spre ei şi a creşte deviaţia standard.

Comparaţia grupurilor

O metodă de analiză statistică este *comparaţia grupurilor*, în situaţii în care vrem să estimăm *mărimea efectului*, adică cuantificarea diferenţei între două grupuri. De exemplu, o putem folosi atunci când dorim să măsurăm:

- răspunsul pacienţilor la un medicament nou;
- reducerea accidentelor de maşină prin introducerea unor noi reguli în codul rutier;
- performanţa studenţilor sub diferite metode pedagogice, etc.

Exemplu: atunci când se introduce un nou medicament, se alcătuiesc un grup de control (sau mai multe) şi un grup de tratament.
Astfel se doreşte compararea efectului medicamentului (în grupul de tratament) faţă de a nu face nimic (efectul placebo în grupul de control).

Pentru a compara grupurile, trebuie să decidem mai întâi ce caracteristici ale acestora vom folosi în comparaţie. O caracteristică des întâlnită este media grupurilor. Abordarea Bayesiană lucrează cu distribuţia a posteriori a diferenţei mediilor între grupuri şi nu doar cu o estimare punctuală a diferenţelor. Avem la îndemână trei unelte pentru estimarea mărimii efectului:

- *graficul distribuţiei a posteriori cu o valoare de referinţă*;
- *diferenţa standardizată a mediilor* (sau *Cohen's d*);
- *probabilitatea superiorităţii*.

Diferenţa standardizată a mediilor

Acesta (sau *Cohen's d*) se defineşte ca

$$d = \frac{\mu_2 - \mu_1}{\sqrt{\frac{\sigma_2^2 + \sigma_1^2}{2}}},$$

unde μ_i, σ_i reprezintă media, respectiv deviaţia standard a grupului i, $i = 1, 2$. Astfel, diferenţa de medii, $\mu_2 - \mu_1$ este standardizată prin luarea în calcul a deviaţiilor standard a ambelor grupuri (ce pot fi de mărimi diferite).

De exemplu, $d = 0.5$ reprezintă o diferenţă de 0.5 deviaţii standard a unui grup faţă de celălalt.

Evident, experienţa ne facem să spunem când anume (în ce context) o anumită valoare d este *mare*, *mică*, *medie*, etc.

Probabilitatea superiorităţii

Este o altă metodă pentru a estima mărimea efectului şi este definită ca:

- probabilitatea ca un punct luat la întâmplare dintr-un grup să aibă o valoare mai mare decât unul luat la întâmplare din celălalt grup.

Presupunând că datele folosite sunt distribuite normal, o putem calcula astfel:

$$p_s := \Phi\left(\frac{d}{\sqrt{2}}\right),$$

unde Φ este funcţia de repartiţie a distribuţiei normale, iar d este diferenţa standardizată a mediilor.

Exemplu: Date legate de bacşiş-uri[1]

Dorim să studiem efectul zilei din săptămână asupra sumei totale rezultate din bacşiş la un restaurant.

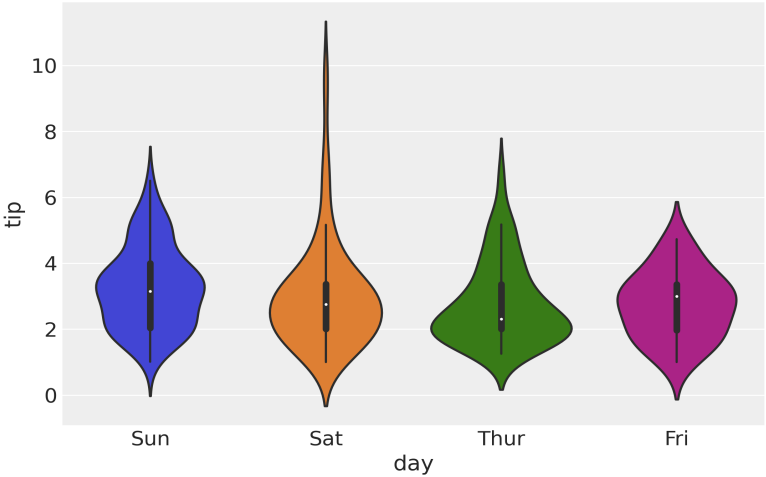
- aici, diversele grupuri sunt date de zilele din săptămână;
- nu avem grupe de control;
- citim datele dintr-un fişier `.csv` ca un `DataFrame` din librăria `pandas`.

```
tips = pd.read_csv('./data/tips.csv')
tips.tail()
```

	total_bill	tip	sex	smoker	day	time	size
239	29.03	5.92	Male	No	Sat	Dinner	3
240	27.18	2.00	Female	Yes	Sat	Dinner	2
241	22.67	2.00	Male	Yes	Sat	Dinner	2
242	17.82	1.75	Male	No	Sat	Dinner	2
243	18.78	3.00	Female	No	Thur	Dinner	2

[1] Bryant, P. G., Smith, M in *Practical Data Analysis: Case Studies in Business Statistics*, 1995.

Vom folosi doar coloanele `day` şi `tip`, pe care putem să le vizualizăm cu funcţia `violinplot` din librăria `seaborn`:



Creăm mai întâi variabilele:

```
tip = tips['tip'].values
idx = pd.Categorical(tips['day'],
                     categories=['Thur', 'Fri', 'Sat', 'Sun']).codes
groups = len(np.unique(idx))
```

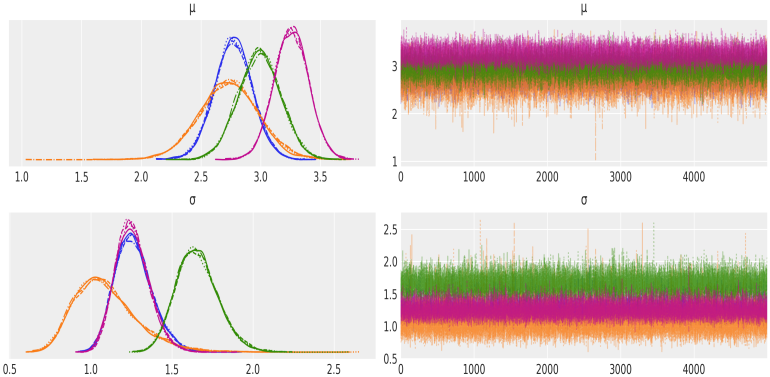
şi apoi modelul:

```
with pm.Model() as comparing_groups:
    μ = pm.Normal('μ', mu=0, sd=10, shape=groups)
    σ = pm.HalfNormal('σ', sd=10, shape=groups)

    y = pm.Normal('y', mu=μ[idx], sd=σ[idx], observed=tip)

    idata_cg = pm.sample(5000, return_inferencedata=True)
az.plot_trace(idata_cg)
```

Rezumatul grafic al inferenţei:



Codul de mai jos realizează graficele diferențelor a posteriori $\mu_i - \mu_j$ cu $0 \leq i < j \leq 3$:

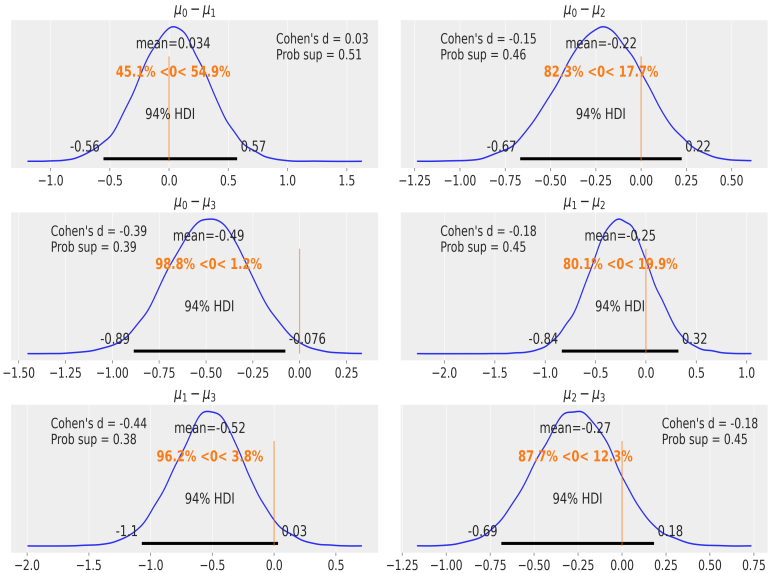
```
dist = stats.norm()

_, ax = plt.subplots(3, 2, figsize=(14, 8), constrained_layout=True)

comparisons = [(i, j) for i in range(4) for j in range(i+1, 4)]
pos = [(k, l) for k in range(3) for l in (0, 1)]

posterior_cg = idata_cg.posterior.stack(samples={"chain", "draw"})

for (i, j), (k, l) in zip(comparisons, pos):
    means_diff = posterior_cg["μ"].sel({"μ_dim_0":i}) - posterior_cg["μ"].sel({"μ_dim_0":j})
    d_cohen = (means_diff / np.sqrt((posterior_cg["σ"].sel({"σ_dim_0":i})**2 +
                                     posterior_cg["σ"].sel({"σ_dim_0":j})**2) / 2)).mean().item()
    ps = dist.cdf(d_cohen/(2**0.5))
    az.plot_posterior(means_diff.values, ref_val=0, ax=ax[k, l])
    ax[k, l].set_title(f'$\mu_{i}-\mu_{j}$')
    ax[k, l].plot(
        0, label=f"Cohen's d = {d_cohen:.2f}\nProb sup = {ps:.2f}", alpha=0)
    ax[k, l].legend()
```


$\mu_0 - \mu_1$
mean=0.034
Cohen's d = 0.03
Prob sup = 0.51
45.1% <0< 54.9%
94% HDI
-0.56
0.57
$\mu_0 - \mu_2$
Cohen's d = -0.15
Prob sup = 0.46
mean=-0.22
82.3% <0< 17.7%
94% HDI
-0.67
0.22
$\mu_0 - \mu_3$
Cohen's d = -0.39
Prob sup = 0.39
mean=-0.49
98.8% <0< 1.2%
94% HDI
-0.89
-0.076
$\mu_1 - \mu_2$
Cohen's d = -0.18
Prob sup = 0.45
mean=-0.25
80.1% <0< 19.9%
94% HDI
-0.84
0.32
$\mu_1 - \mu_3$
Cohen's d = -0.44
Prob sup = 0.38
mean=-0.52
96.2% <0< 3.8%
94% HDI
-1.1
0.03
$\mu_2 - \mu_3$
mean=-0.27
Cohen's d = -0.18
Prob sup = 0.45
87.7% <0< 12.3%
94% HDI
-0.69
0.18

Modele ierarhice

Să presupunem că dorim să analizăm calitatea apei dintr-un oraş. Prin urmare, prelevăm eşantioane împărţind oraşul în cartiere. Avem două opţiuni:

- să studiem fiecare cartier ca o entitate separată;
- să strângem laolaltă toate datele şi să estimăm calitatea apei din întregul oraş.

Deşi fiecare metodă are avantajul său (o vedere mai detaliată asupra problemei vs. un eşantion mai mare ce duce la o acurateţe sporită), putem alege o cale intermediară:

- putem clădi un model care să estimeze atât calitatea apei la nivel de cartier, cât şi la nivel de oraş.

Un astfel de model se numeşte model *ierarhic* sau *multietajat*.

Cum construim un astfel de model ierarhic?

- În loc să fixăm parametrii distribuţiilor a priori, îi vom estima conform altor distribuţii a priori, numite distribuţii *hiper a priori*, cu parametrii numiţi *hiperparametri*.
- Putem etaja astfel modelul cât se poate de mult, dar dacă sistemul nu cere o astfel de structură, adăugând niveluri suplimentare, modelul devine dificil de înţeles şi inferenţele nu neapărat mai bune.
- Astfel, modelele ierarhice trebuie să reflecte structura datelor.

- Să presupunem că avem 3 cartiere şi că am măsurat nivelul de plumb din apă.
- Pentru simplitate, eşantioanele ce depăşesc nivelul maxim admis de OMS vor fi notate cu 0, iar celelalte cu 0.
- Putem genera datele (artificial produse) cu:

```
N_samples = [30, 30, 30]
G_samples = [18, 18, 18]  # [3, 3, 3]  [18, 3, 3]

group_idx = np.repeat(np.arange(len(N_samples)), N_samples)
data = []
for i in range(0, len(N_samples)):
    data.extend(np.repeat([1, 0], [G_samples[i], N_samples[i]-G_samples[i]]))
```

Modelul este esenţial acelaşi cu cel folosit pentru aruncarea banului (modelul beta-binomial), exceptând:

- vom defini două distribuţii hiper a priori ce vor influenţa distribuţia a priori $\text{Beta}(\alpha, \beta)$;
- în loc să definim parametrii α şi β ca v.a. hiper a priori, îi vom defini indirect cu ajutorul a doi termeni, μ şi κ, ce reprezintă:
  - *media* distribuţiei $\text{Beta}(\alpha, \beta)$ şi
  - *precizia* (*concentraţia*) distribuţiei $\text{Beta}(\alpha, \beta)$ – analog inversei deviaţiei standard.
- Astfel, $\alpha = \mu \cdot \kappa$, $\beta = (1 - \mu) \cdot \kappa$.

Modelul are deci forma:

$$
\begin{aligned}
\mu &\sim \text{Beta}(\bar{\alpha}, \bar{\beta}) \\
\kappa &\sim |\mathcal{N}(0, \sigma_\kappa)| \\
\alpha &= \mu \cdot \kappa \\
\beta &= (1-\mu) \cdot \kappa \\
\theta_i &\sim \text{Beta}(\alpha_i, \beta_i) \\
y_i &\sim \text{Bernoulli}(\alpha_i, \beta_i)
\end{aligned}
$$

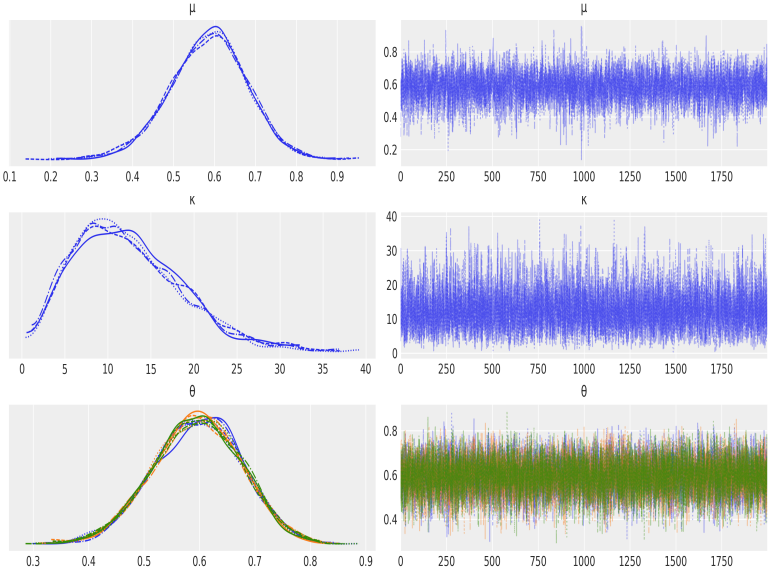
Folosirea indicelui i subliniază faptul că unii parametri au diverse valori pentru grupuri (cartiere) diferite.

```
with pm.Model() as model_h:
    μ = pm.Beta('μ', 1., 1.)
    κ = pm.HalfNormal('κ', 10)

    θ = pm.Beta('θ', alpha=μ*κ, beta=(1.0-μ)*κ, shape=len(N_samples))
    y = pm.Bernoulli('y', p=θ[group_idx], observed=data)

    idata_h = pm.sample(2000, return_inferencedata=True)
```

Rezumatul grafic al inferenţei:



Ce se întâmplă când schimbăm datele?

G_samples			θ (media)		
18,	18,	18	0.6,	0.6,	0.6
3,	3,	3	0.1,	0.1,	0.1
18,	3,	3	0.55,	0.13,	0.13

Apare aşa-numitul fenomen de *contracţie*, mediile estimate pe grupuri se apropie de media generală (care este $(18 + 3 + 3)/90 = 0.2(6)$ pentru ultima linie).