Inferență statistică în ML

Cap 5. Power. Testarea multiplă. Resampling.

April 16, 2019

Power

- **power** este probabilitatea de a rejecta ipoteza nulă atunci când aceasta este FALSE
- power este un lucru bun; vrem 'mai multă putere'
- power intră în discuție mai mult când eșuăm să rejectăm ipoteza nulă, decât atunci când o rejectăm
- exemplu:
  - realizăm un trial test randomizat pentru testarea unui nou medicament
  - sunt două grupuri, grupul de test (cel care primește medicamentul) și grupul de control (primește placebo)
  - fiecare grup are însă doar trei pacienți
  - cel mai probabil rezultat al testului este că eșuăm să rejectăm H_0
  - cumva așteptat, avem foarte puțină 'putere' pentru a detecta diferența dintre grupuri
  - pentru 300 de pacienți în fiecare grup, dacă rezultatul este eșuarea de a rejecta ipoteza nulă, testul este declarat concludent, pentru că am strâns multe date
  - power este mai importantă când obținem H_0 față de obținerea H_a

Power (2)

- power intră în discuție la proiectarea testării ipotezei: ne interesează un test care să aibă o șansă rezonabilă de a detecta H_a dacă aceasta este TRUE
- type I error este situația în care rejectăm H_0 (acceptăm H_a) atunci când H_0 este adevărată; de acestă situație ne-am ocupat prin setarea parametrului α la 5%
- type II error este situația de a accepta H_0 atunci când aceasta este FALSĂ; este un lucru prost; probabilitatea type II error este notată cu β; acest β de asemenea trebuie să fie cât mai mic

$$Power = 1 - \beta$$

- dar power $(1 - \beta)$ trebuie să fie mare

Exemplu

- considerăm exemplul anterior cu Respiratory Distress Index (rata de evenimente respiratorii)
- $H_0 : \mu = 30$ vs. $H_a : \mu > 30$
- atunci power este:

$$P\left(\frac{\bar{X} - 30}{s/\sqrt{n}} > t_{1-\alpha,n-1} \,;\, \mu = \mu_a\right)$$

- funcție ce depinde de valoarea specifică a lui μ_a
- aceeași cu cea care calculează probabilitatea ca statistica să fie mai mare ca quantila T distribution pentru $1 - \alpha$
- pe măsură ce μ_a tinde la 30, power tinde la α (type I error)
- centrăm distribuția T nu în $\mu = 30$, ci în $\mu_a > 30$
- power pentru $\mu_a = 60$ (mare) vs. power pentru $\mu_a = 30.00001$ (mică)
- power e o funcție ce depinde și de μ, valoarea asumată de H_0

Power pentru date Gaussiene

- pentru testarea ipotezei, vom rejecta H_0 dacă $\frac{\bar{X}-30}{\sigma/\sqrt{n}} > z_{1-\alpha}$
- relația echivalentă pentru cazul reject, pentru sample mean este $\bar{X} > 30 + Z_{1-\alpha}\frac{\sigma}{\sqrt{n}}$
- sub $H_0 : \bar{X} \sim N(\mu_0, \sigma^2/n)$
- sub $H_a : \bar{X} \sim N(\mu_a, \sigma^2/n)$
- depinzînd de ipoteză, sample mean poate urma o distribuție sau alta

Exemplu

```
mu_0 = 30
mu_a = 32
alpha = 0.05
sigma = 4
n = 16
z = stats.norm.ppf(1 - alpha)
print(stats.norm.sf(mu_0 + z * sigma/np.sqrt(n), loc=mu_0, scale=sigma/np.sqrt(n)))
print(stats.norm.sf(mu_0 + z * sigma/np.sqrt(n), loc=mu_a, scale=sigma/np.sqrt(n)))
```

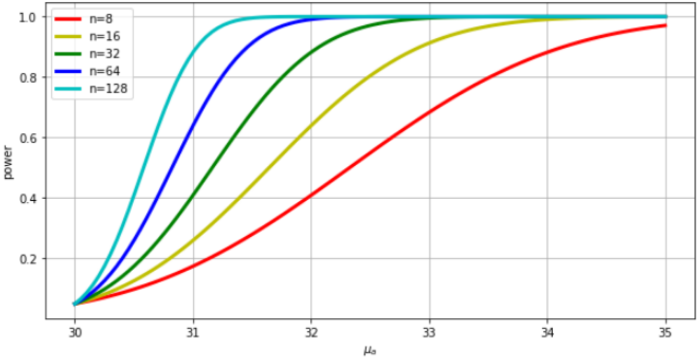
```
0.04999999999999954
0.6387600313123348
```

- pentru un studiu, autorii sunt interesați să calculeze probabilitatea de a obține sample means mai mari ca 30, însă îi interesează în particular valoari la fel de mari ca 32
- dacă inserăm $\mu = \mu_0$, atunci calculăm probabilitatea de a obține valori mai mari decât quantila 95%, care este chiar 5%
- există o probabilitate de 64% de a detecta o medie la fel de mare ca 32 sau mai mare dacă realizăm studiul

Curba puterii

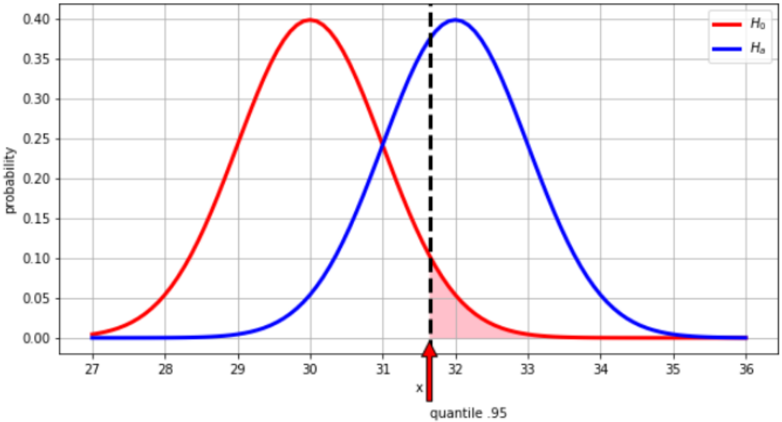
```python
def power(mu_a, n):
    mu_0, alpha, sigma = 30, 0.05, 4
    z = stats.norm.ppf(1 - alpha)
    return stats.norm.sf(mu_0 + z * sigma/np.sqrt(n), loc=mu_a, scale=sigma/np.sqrt(n))

mu_a = np.linspace(30, 35, 100)

plt.figure(figsize=(10,5))
[plt.plot(mu_a, power(mu_a, n), lw=3, color=c)
     for n,c in [(8, 'r'), (16, 'y'), (32, 'g'), (64, 'b'), (128, 'c')]
]
plt.grid()
plt.legend(['n=8', 'n=16', 'n=32', 'n=64', 'n=128'])
plt.xlabel('$\mu_a$')
plt.ylabel('power')
plt.show()
```

Curba puterii (2)



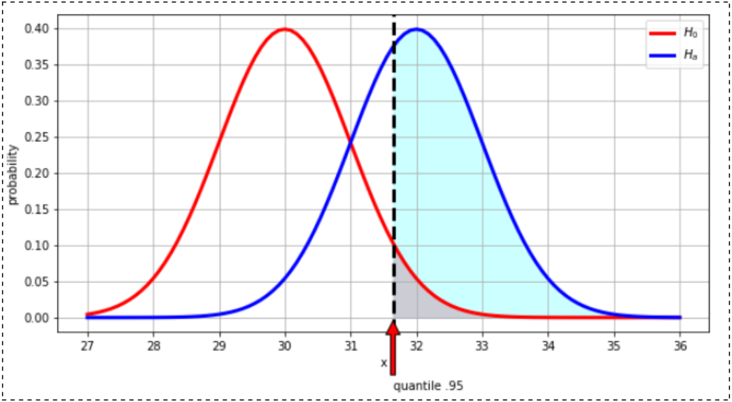
- toate liniile converg la 0.5 când $\mu = 30$
- power crește când μ_a crește: e mult mai probabil să detectăm o diferență când acea diferență e foarte mare
- când sample size crește, power crește mai rapid: mai multe date colectate, mai probabil de detectat acea diferență

Distribuțiile H0 și Ha pentru sample mean



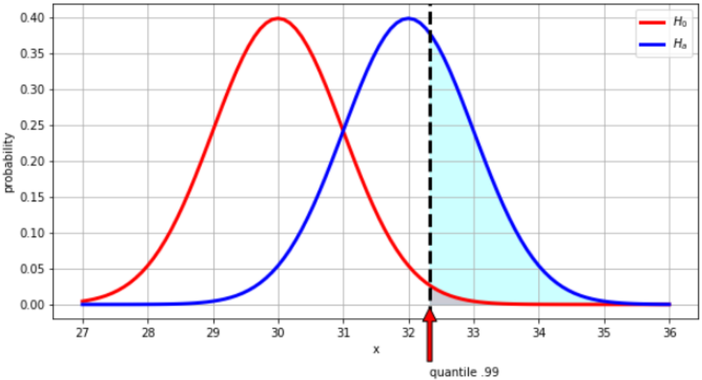
- $H_0 : \bar{X} \sim N(\mu_0 = 30, \sigma^2/n)$, $H_a : \bar{X} \sim N(\mu_a = 32, \sigma^2/n)$
- linia neagră : threshold, pentru valori mai mari ca ea, rejectăm

Power ca probabilitatea de sub Ha



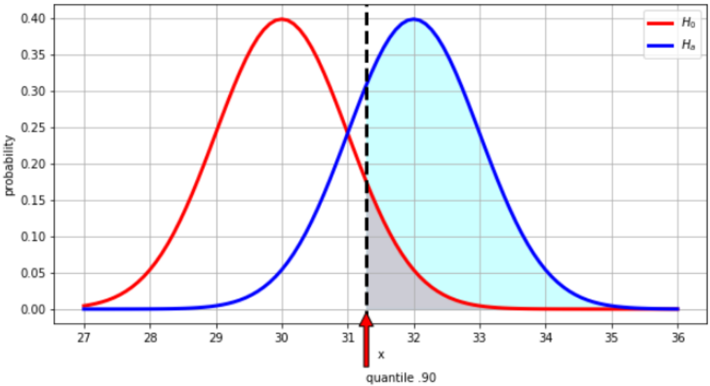
- power = probabilitatea ca sample mean să fie mai mare ca black line (calibrată pentru AuC roșie de 5%), adică probabilitatea de reject dacă H_a (albastră) este TRUE
- partea din stânga este type II error rate (1 - power)

alpha = 1%



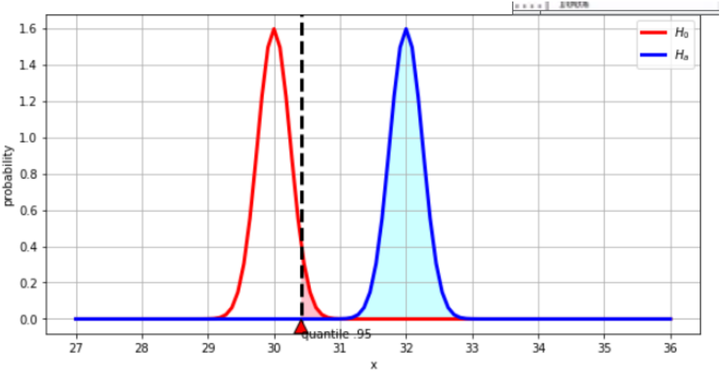
- $\alpha = 1\%$, scădem type I error rate (probabilitatea de rejecție H_0 dacă aceasta e adevărată), dar vedem cum scade power (de la 64% la sub 50%), $\beta = 1 - power$, crește type II error rate (P(fail to reject ; H_0 este FALSĂ))

alpha = 10%



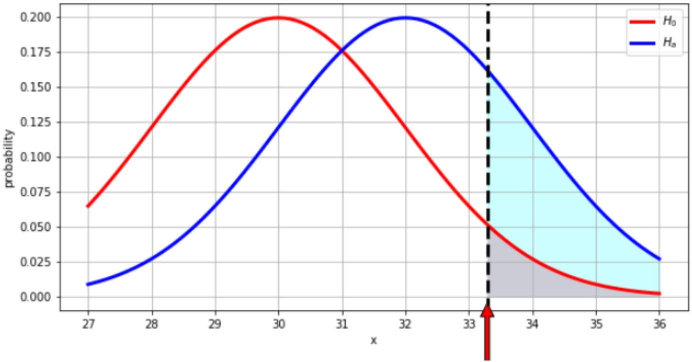
- $\alpha = 10\%$, creștem type I error rate, și vedem cum crește power (peste 64%), scade type II error rate (P(fail to reject ; H_0 este FALSĂ)) - partea de sub curba albastră

Variabilitatea scade, sigma = 1



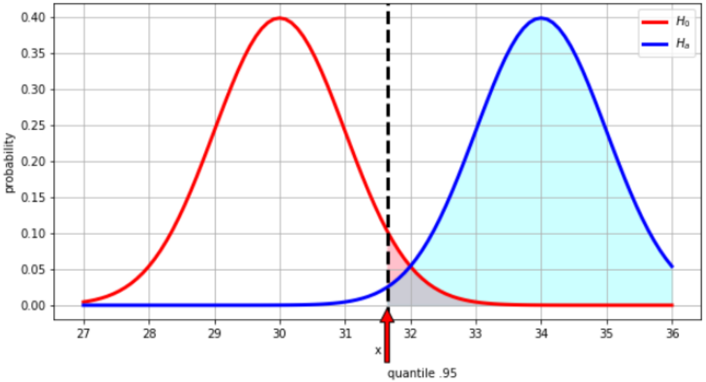
- $\alpha = 5\%$; $\sigma = 1$ (variabilitate mică)
- deși type I error este 5%, power e aproape 100%, type II error rate aproape de 0

Variabiliatea crește, sigma = 8



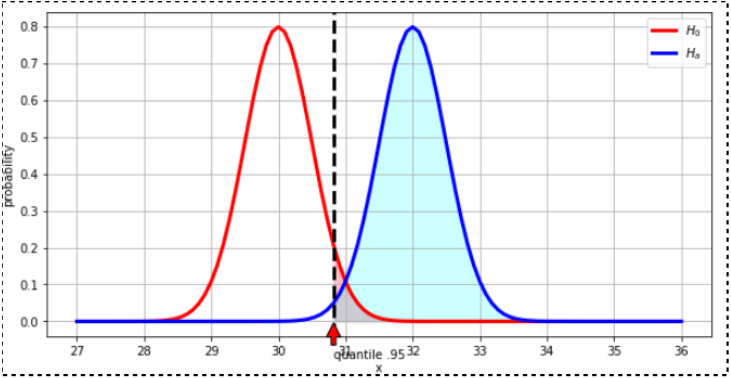
- $\alpha = 5\%$; $\sigma = 8$ (variabilitate foarte mare)
- mult zgomot în măsurători $\rightarrow$ lower power

M_a se deplasează la dreapta



- $\alpha = 5\%$; $\sigma = 4$; $\mu_a = 34$
- distribuția alternativă se mută mai la dreapta $\rightarrow$ avem mai multă putere

Sample size crește, n = 64



- pe măsură ce crește n, avem mai puțină variabilitate (dispersia distribuției sample mean scade), densitățile se 'strâng', power crește

Ecuația pentru power

- când testăm $H_a : \mu > \mu_0$, power este $1 - \beta$, adică

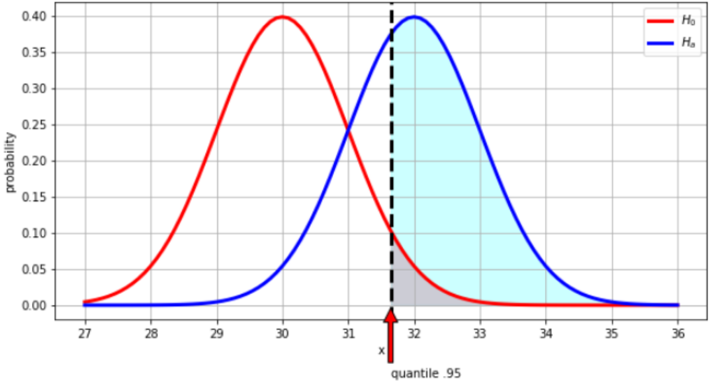
$$1 - \beta = P\left(\bar{X} > \mu_0 + z_{1-\alpha}\frac{\sigma}{\sqrt{n}}\ ;\ \mu = \mu_a\right)$$

- unde $\bar{X} \sim N(\mu_a, \sigma^2/n)$
- necunoscute: μ_a, σ, n, β
- concrete: μ_0, α
- putem specifica oricare 3 pentru a o afla pe a patra
- de cele mai multe ori impunem β, și determinăm n necesar în funcție de variabilitate

Power pentru two-sided test

- raționamentul pentru $H_a : \mu < \mu_a$ este similar
- pentru $H_a : \mu \neq \mu_a$, calculăm one-sided test power folosind $\alpha/2$
- power crește când α crește
- power pentru one-sided test este mai mare decât power pentru two-sided test (din cauza lui α)
- power crește când μ_a se depărtează de μ_0
- power crește cu creșterea lui n
- power crește când σ scade
- de fapt, depinde de $\frac{\sqrt{n}(\mu_a - \mu_0)}{\sigma}$
- cantitatea $\frac{\mu_a - \mu_0}{\sigma}$ se numește **effect size**, diferența de medii în unități de deviație standard (unit free)

Calculul power în Python




```
print(stats.norm.sf(x_black, loc=mu_a, scale=sigma/np.sqrt(n)))
print(statsmodels.stats.power.normal_power(
    effect_size=(mu_a - mu_0)/sigma,
    nobs=16, alpha=0.05, alternative='larger')
)
```

```
0.6387600313123348
0.638760031312335
```

T-test power

- considerăm calculul power pentru testul T Gosset pentru exemplul anterior
- power este:

$$P\left(\frac{\bar{X}-\mu_0}{S/\sqrt{n}} > t_{1-\alpha,n-1}\,;\ \mu=\mu_a\right)$$

- rejectăm dacă statistica este acum mai mare decât quantila T, unde distribuția T este calculată sub ipoteza $\mu = \mu_a$
- statistica nu urmărește o distribuție T dacă $\mu = \mu_a$, ci o distribuție non-centrală de tip T
- putem calcula un parametru (omitem unul și îi specificăm pe ceilalți 3)

Calculul T test power

```
print('mu_0: %d, mu_a: %d' % (mu_0, mu_a))
print('sigma: ', sigma)
print('power: ', statsmodels.stats.power.ttest_power(
    effect_size=(mu_a - mu_0)/sigma,
    nobs=16, alpha=0.05, alternative='larger'))
```

```
mu_0: 30, mu_a: 32
sigma:  4
power:  0.6040328683316007
```

```
print(statsmodels.stats.power.tt_solve_power(
    effect_size=(mu_a - mu_0)/sigma,
    alpha=0.05, nobs=16, alternative='larger'))
print(statsmodels.stats.power.tt_solve_power(
    effect_size=(mu_a - mu_0)/sigma,
    alpha=0.05, power=0.6040328, alternative='larger'))
```

```
0.6040328683316007
15.999997299353081
```

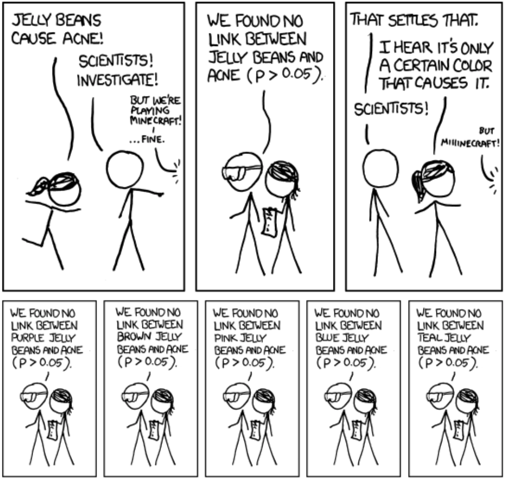
Testarea multiplă

- atunci când se realizează multe testări ale ipotezei, vom realiza corecții
- testarea ipotezei este o tehnică supra-utilizată
- ce facem când obținem mai multe p-values pentru același dataset?
- pentru teste multiple, corecția previne apariția false-positive-lor sau a falselor descoperiri
- cum definim măsura erorii pe care vrem să o controlăm
- cum definim corecția ca metodă statistică de a controla eroarea

Cele trei 'epoci' ale statisticii ca știință

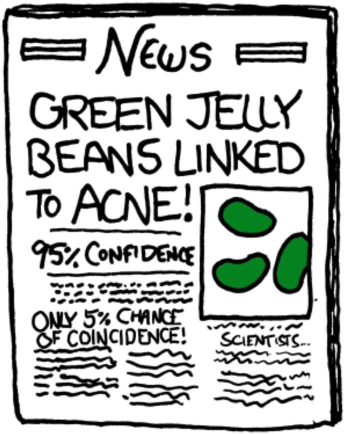
- epoca lui Quetelet (Belgia sec. XIX), în care recensămintele dădeau răspunsurile la întrebări precum 'se nasc mai multe fetițe decât băieței?', 'crește rata nebuniei?'
- perioada clasică, Pearson, Fisher, Hotelling și succesorii lor (sec. XX), ce au dezvoltat teoria inferenței optime pentru a trage concluzii asupra populației folosind datele limitate ale unui experiment, 'este tratamentul A mai bun ca tratamentul B?'
- epoca producției științifice de masă, cu dataset-uri imense, mii de teste ale ipotezei; care dintre miile de p-values sunt relevante și care apar din pură întâmplare?
- B. Efron, "Large-Scale Inference: Empirical Bayes Methods for Estimation, Testing and Prediction"

De ce e nevoie de corecție pentru teste multiple



- `https://xkcd.com/882/`

De ce e nevoie de corecție pentru teste multiple (2)



- $\alpha = 0.05$: șansa ca din întâmplare să apară un eveniment cu această p-valoare sau mai extremă
- dacă s-au făcut 20 de teste, șansa nu mai e de 5%
- din aceste 20 de teste, ne așteptăm să găsim cel puțin o eroare (p-valoare < 0.05 din întâmplare)

Tipuri de erori

- 'potrivim' (engl. fit) o regresie liniară[1], $y \sim w * x + b$, unde x și y sunt vectori, iar w și b scalari
- testăm parametrul w al unui modelului de regresie pentru a vedea dacă e diferit de zero, adică dacă există o asociere între valorile lui x și cele ale lui y
- $H_0 : \gamma = 0$ vs. $H_a : \gamma \neq 0$ (two-sided test), verificăm p-value

	Real: w = 0	Real: $w \neq 0$	Ipoteze
Presupunem H_0 : w = 0	U	T	m - R
Presupunem $H_a : w \neq 0$	V	S	R
Presupuneri	m_0	$m - m_0$	m

- type I error sau False Positive (V): considerăm $w \neq 0$ greșit
- type II error sau False Negative (T): considerăm $w = 0$ greșit

[1] w este panta ('slope'), coeficientul variabilei x denumită 'covariate'; b este denumit 'intercept' (`https://en.wikipedia.org/wiki/Linear_regression`)

Error rates

	Real: w = 0	Real: $w \neq 0$	Ipoteze
Presupunem H_0 : w = 0	U (TN)	T (FN)	m - R
Presupunem H_a : $w \neq 0$	V (FP)	S (TP)	R
Presupuneri	m_0	$m - m_0$	m

- False Positive Rate[2]: rata la care rezultatele false (w=0) sunt semnificative, $E\left[\frac{V}{m_0}\right]$ = False Positives / (False Positives + True Negatives)
- Family Wise Error Rate (FWER): probabilitatea de a obține cel puțin o False Positive, $Pr(V \geq 1)$
- False Discovery Rate (FDR): rata la care presupunerile de relevanță statistică sunt false, $E\left[\frac{V}{R}\right]$ = False Positives / (False Positives + True Positives)

[2] https://en.wikipedia.org/wiki/False_positive_rate

Controlul False Positive Rate (FPR - type I error)

- controlul se face cu α (de regulă 5%)
- regulă cu care să putem controla câte detecții pozitive false facem, din totalul w $= 0$
- controlul se face cu procedura care presupune p-value semnificativă statistic dacă $pValue < \alpha$; în medie, rata este acest α
- presupunem că realizăm 10.000 de teste și w=0 pentru toate
- presupunem că stabilim ca toate $P < 0.05$ să fie semnificative
- numărul așteptat de false positives este $10.000 \times 0.05 = 500$ false positives
- cum putem preveni atât de multe false positives?

Controlul Family-wise Error Rate (FWER)

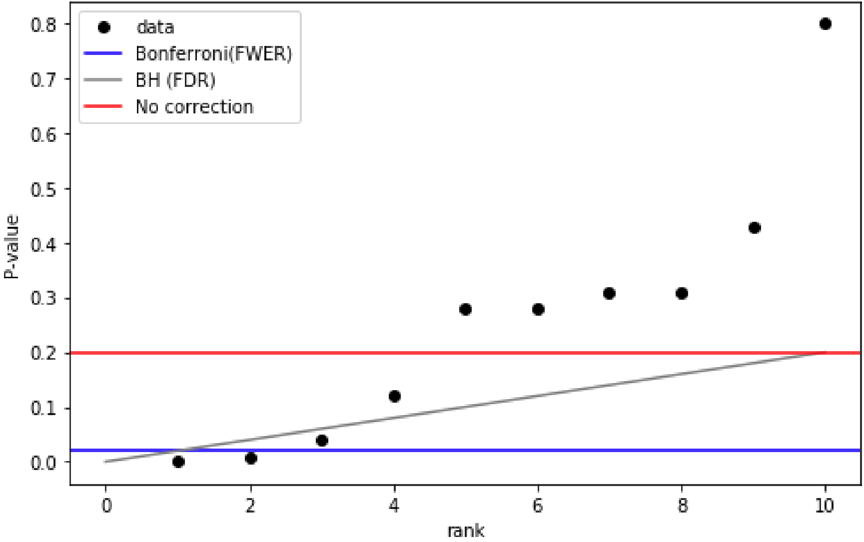
- cea mai veche corecție este corecția Bonferroni[3]
- principiu:
  - facem m teste
  - vrem să limităm nivelul FWER la valoarea α astfel ca $P(V \geq 1) < \alpha$ (probabilitatea de a face chiar și o singură eroare să fie mai mică decât α)
  - calculăm p-values în modul obișnuit
  - setăm $\alpha_{fwer} = \alpha/m$
  - denumim semnificative doar p-values mai mici ca α_{fwer}
- ușor de calculat, garantat vom face foarte puține erori,
- **însă** poate fi o corecție extrem de conservatoare

[3] https://en.wikipedia.org/wiki/Bonferroni_correction

Controlul False Discovery Rate (FDR)

- cea mai populară corecție când realizăm multe teste în genetică, procesare de imagini, astronomie
- principiu:
  - realizăm m teste
  - dorim să controlăm $\text{FDR} = E\left[\frac{V}{R}\right]$ la un nivel α
  - calculăm p-values în modul obișnuit
  - ordonăm p-values crescător, $P_{(1)} < P_{(2)} < \cdots < P_{(m)}$
  - denumim semnificative valorile $P_{(i)} \leq \alpha \times \frac{i}{m}$
- ușor de calculat, mult mai puțin conservatoare
- **însă** lasă să treacă mai multe false positives; se poate comporta ciudat în condiții de variabile dependente

Exemplu de corecție (FPR/type I error/$\alpha = 0.2$)



Ajustarea p-valorilor

- până acum am ajustat llimita α
- o altă abordare este ajustarea p-valorilor
- prin ajustare, nu mai sunt p-values
- dar pot fi folosite direct fără a mai modifica α

- principiu pentru FWER:
  - presupunem că p-values sunt $P_1, P_2, \dots P_m$
  - le ajustăm prin $P_i^{fwer} = \max(m \times P_i, 1)$ pentru fiecare p-value P_i
  - vom considera semnificative doar valorile $P_i^{fwer} < \alpha$, realizând astfel corecția Bonferroni

Exemplu 1: no True Positives: no correction

```
n = 1000
pValues = np.zeros(n)
for i in range(n):
    x = np.random.randn(20)
    y = np.random.randn(20)
    x = sm.add_constant(x)
    est = sm.OLS(y, x).fit()
    pValues[i] = est.pvalues[1]

print('no correction: ', np.sum(pValues < 0.05))
```

```
no correction:  55
```

Exemplu 1: interpretarea pValue pentru regresia liniară

- $y \sim w * x + b$
- x: predictor; y: predicted; w: slope sau coeficientul lui x; b: intercept

- "The p-value for each independent variable tests the null hypothesis that the variable [x] has no correlation with the dependent variable [y]. If there is no correlation, there is no association between the changes in the independent variable and the shifts in the dependent variable. In other words, there is insufficient evidence to conclude that there is effect at the population level."
- `https://statisticsbyjim.com/regression/interpret-coefficients-p-values-regression/`

Exemplu1: no True Positives: corrected

```
res = mt.multipletests(pValues, method='bonferroni')
print('Bonferroni correction: ', np.sum(res[1] < 0.05))

res = mt.multipletests(pValues, method='fdr_bh')
print('Benjamini/Hochberg correction: ', np.sum(res[1] < 0.05))
```

```
Bonferroni correction:  0
Benjamini/Hochberg correction:  0
```

Exemplu 2: 50% True Positives (1)

```
# exemplu pentru care coeficientul w al regresiei liniare este:
# 0 pentru primele 500 de situatii construite
# 2 pentru urmatoarele
n = 1000
pValues = np.zeros(n)
for i in range(n):
    x = np.random.randn(20)
    y = np.random.randn(20) if i < n//2 else (np.random.randn(20) + 2*x)
    x = sm.add_constant(x)
    est = sm.OLS(y, x).fit()
    pValues[i] = est.pvalues[1]

print('no correction:', np.sum(pValues < 0.05))
res = mt.multipletests(pValues, method='bonferroni')
print('corectie FWER:' , np.sum(res[1] < 0.05))
```

```
no correction: 524
corectie FWER: 483
```

Exemplu 2: 50% True Positives (2)

```
df = pd.DataFrame({
    'trueStatus': ['zero'] * 500 + ['not zero'] * 500,
    'pValue': pValues
})
pd.crosstab(df.pValue < 0.05, df.trueStatus)
```

trueStatus	not zero	zero
pValue		
False	0	476
True	500	24

Exemplu 2: 50% True Positives (3)

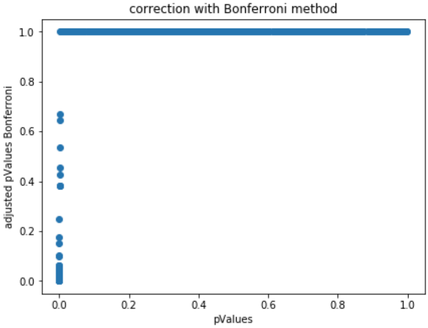
```
df = pd.DataFrame({
    'trueStatus': ['zero'] * 500 + ['not zero'] * 500,
    'pValueAdjusted': mt.multipletests(pValues, method='bonferroni')[1]
})
pd.crosstab(df.pValueAdjusted < 0.05, df.trueStatus)
```

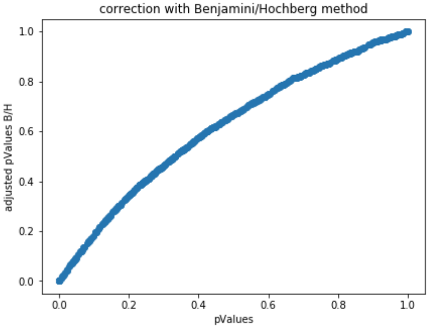
trueStatus	not zero	zero
pValueAdjusted		
False	17	500
True	483	0

```
df = pd.DataFrame({
    'trueStatus': ['zero'] * 500 + ['not zero'] * 500,
    'pValueAdjusted': mt.multipletests(pValues, method='fdr_bh')[1]
})
pd.crosstab(df.pValueAdjusted < 0.05, df.trueStatus)
```

trueStatus	not zero	zero
pValueAdjusted		
False	0	489
True	500	11

Exemplu 2: 50% True Positives (4)





Resurse

- studiul testelor multiple este un domeniu în sine
- o corecție Bonferroni sau B/H e deseori suficientă
- dacă există dependență între teste se poate considera corecția Benjamini-Yekutieli
- https://en.wikipedia.org/wiki/Multiple_comparisons_problem
- Q-Q plot arată cât de 'normală' e distribuția (quantilele distribuției vs. quantilele distribuției normale)
- "The practice of trying many unadjusted comparisons in the hope of finding a significant one is a known problem, whether applied unintentionally or deliberately, is sometimes called *p-hacking*." (Wikipedia)
- S.Y.Chen et al., "A general introduction to adjustment for multiple comparisons", https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5506159/

1. Power

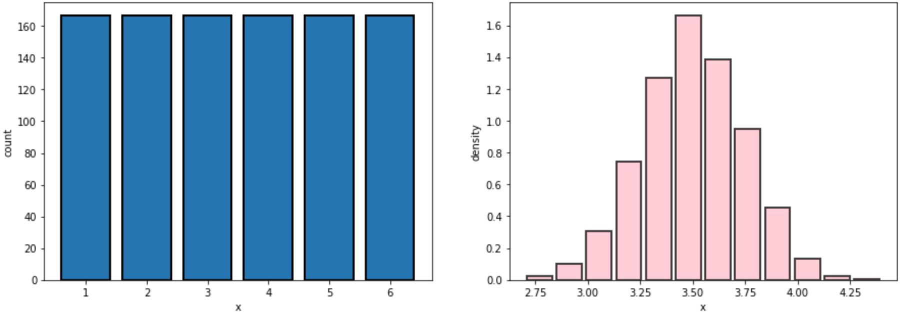
2. Testarea multiplă

3. Resampling

Procedura bootstrap

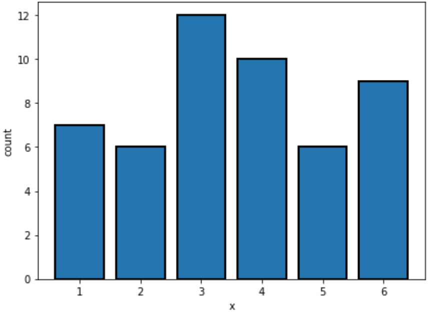
- bootstrap este o procedură folostiă pentru calculul intervalelor de confidență și calculul erorii standard pentru statistici dificile
- inventată în 1979 de către Bradley Efron
- metodă care simplifică munca analiștilor de date aparatul matematic necesar calculării proprietăților statisticilor
- "Circa 1900, to pull (oneself) up by (one's) bootstraps was used figuratively of an impossible task (among the *practical questions* at the end of chapter one of Steele's *Popular Physics* schoolbook (1888) is, *30. Why can not a man lift himself by pulling up on his boot-straps?* By 1916 the meaning of the phrase had expanded to include *better oneself by rigorous, unaided effort.* The meaning *fixed sequence of instructions to load the operating system of a computer* (1953) is from the notion of the first-loaded program pulling itself (and the rest) up by the bootstrap."
(`https://www.etymonline.com/word/bootstrap`)

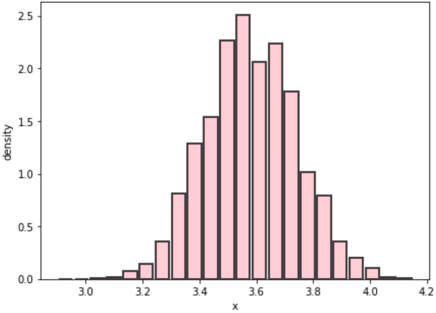
Sample cu 50 de valori de tip zar



- se aruncă de 1.000 de ori, fiecare față este echiprobabilă (stânga)
- se aruncă de 50 de ori (sample) și se face media sample-ului; se repetă procedura de 10.000 de ori (dreapta) $\rightarrow$ distribuția sample means

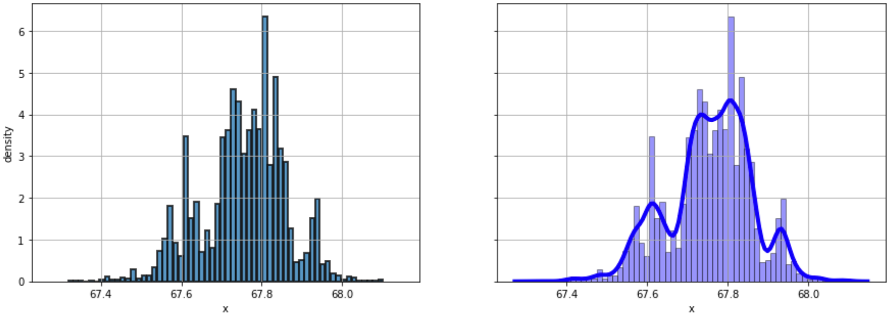
Bootstrap: resample cu înlocuire din același sample





- obținem un sample de 50 de valori din distribuția originală; distribuția sample-ului nu mai este echiprobabilă (stânga)
- realizăm un resampling cu replacement, extragem 100 de valori; se repetă procedura de 10.000 de ori (dreapta) $\rightarrow$ distribuția resample means

Distribuția estimată a medianei resample-urilor



- din sample-ul extras inițial, am re-extras sample-uri de aceeași dimensiune, cu înlocuire
- estimăm distribuția populației folosind un singur sample din ea
- putem lua deviația standard (estimand al standard error pentru mediană)

Principiul bootstrap

- presupunem că avem o statistică sampled ce estimează un parametru al populației, dar nu știm distribuția sa
- principiul bootstrap sugerează că putem folosi distribuția definită de date pentru a aproxima sampling distribution
- în practică, principiul bootstrap e realizat folosind simulări
- se realizează extrageri **cu înlocuire** din datele sample-ului pe care îl avem
- ideea e că unele date sunt duplicate
- vom calcula o statistică pentru fiecare dataset simulat
- folosim statistica pentru a calcula confidence intervals sau standard error

Statistici pe distribuția resampled

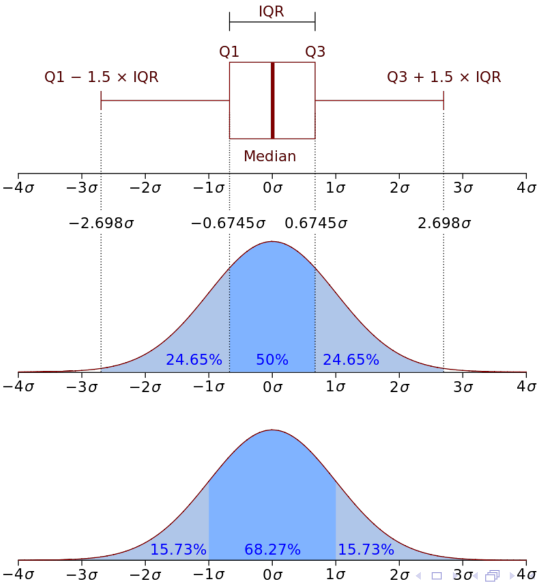
```
import seaborn

# vezi https://towardsdatascience.com/histograms-and-density-plots-in-python-f6bda88f5ac0

x = father_son.fheight.values
n, nosims = len(x), 10000
resamples = np.random.choice(x, size=(nosims, n), replace=True)
resampledMedians = np.median(resamples, axis=1)

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(15, 5), sharey=True, sharex=True)
kwargs = dict(rwidth=0.85, density=True, alpha=0.75, ec='k', linewidth=2)
ax1.hist(resampledMedians, **kwargs, bins=60)
ax1.grid()
ax1.set_xlabel('x')
ax1.set_ylabel('density')
seaborn.distplot(resampledMedians, hist=True, kde=True, bins=60, ax=ax2,
    color = 'blue',  hist_kws={'edgecolor':'black'}, kde_kws={'linewidth': 4})
ax2.grid()
ax2.set_xlabel('x')

print('deviatia standard: ', np.std(resampledMedians))
print('quantila 2.5%:', np.quantile(resampledMedians, 0.025))
print('quantila 97.5%:', np.quantile(resampledMedians, 0.975))
plt.show()
```

```
deviatia standard:  0.10407116223529014
quantila 2.5%: 67.550215
quantila 97.5%: 67.94195
```

Whiskers plots

- "A box plot is a method for graphically depicting groups of numerical data through their quartiles. The box extends from the Q1 (25%) to Q3 (75%) quartile values of the data, with a line at the median (Q2, 50%). The whiskers extend from the edges of box to show the range of the data. The position of the whiskers is set by default to 1.5 * IQR (IQR = Q3 - Q1) from the edges of the box. Outlier points are those past the end of the whiskers."
(https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.boxplot.html)
- https://en.wikipedia.org/wiki/Quartile

Whiskers plots (2) - Wikipedia



Comparații între grupuri

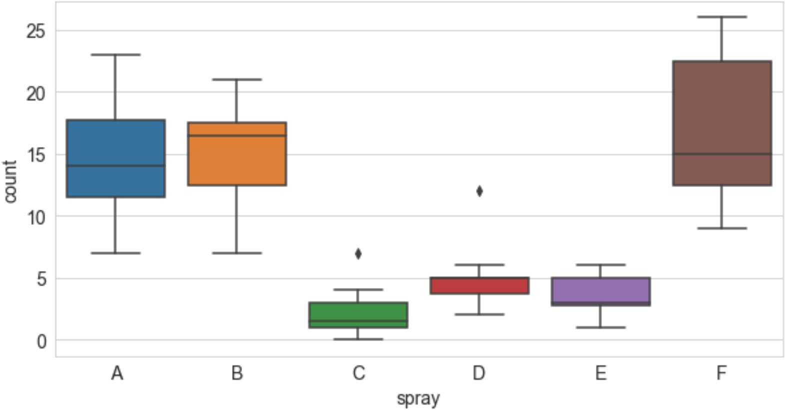
- comparăm două grupuri independente, de exemplu B cu C
- coloana counts numără numărul de insecte omorâte când cultura e tratată cu un spray

```
insect_sprays = pd.read_csv('insect_sprays.csv')
insect_sprays.head()
```

	Unnamed: 0	count	spray
0	1	10	A
1	2	7	A
2	3	20	A
3	4	14	A
4	5	14	A

InsectSprays dataset

```
plt.rcParams.update({'font.size': 14})
fig, ax = plt.subplots(1, 1, figsize=(10, 5))
seaborn.set_style('whitegrid')
seaborn.boxplot(x='spray', y='count', data=insect_sprays, ax=ax)
plt.show()
```



Teste de permutare

- considerăm ipoteza nulă că distribuțiile observațiilor din fiecare grup sunt identice (una și aceeași distribuție)
- atunci etichetele grupurilor nu contează (oricum le-am schimba între ele dăm peste aceeași distribuție)
- considerăm un dataframe cu coloanele count și spray
- permutăm etichetele din coloana spray
- recalculăm diferența între oricare din statisticile:
  - medie aritmetică
  - media geometrică
  - o statistică T
- calculăm procentajul de simulări pentru care statistica pe sample-ul simulat a fost mai extremă (către ipoteza alternativă) față de cea observată

Variațiuni pe tema teste de permutare

Tipuri de date	Statistica	Test statistic
Ranks	rank sum	Rank sum test
Binary	prob. hipergeometrică	Fisher exact test
Raw		Permutation test

- testele de randomizare sunt exact permutation tests
- testele de permutare funcționează și pentru regresii - permutăm regresorul[4] care ne interesează
- testul de permutare funcționează excelent în distribuții multivariate[5], pentru că putem calcula o statistică ce controlează FWER

[4] coeficientul variabilei x_i

[5] în care y depinde de $x_1, x_2, \ldots$

Test de permutare D vs. C

```
def testStat(a, b, groups):
    return np.mean([y for (x, y) in zip(a, b) if x == groups[0]]) \
        - np.mean([y for (x, y) in zip(a, b) if x == groups[1]])

groups = ['D', 'C']
subdata = insect_sprays[insect_sprays.spray.isin(groups)]

observedStat = testStat(subdata['spray'], subdata['count'], groups)

n = subdata['spray'].values.shape[0]
nosims = 10000
permutations = np.array(list(map(
    lambda x: testStat(x, subdata['count'], groups),
    [np.random.choice(subdata['spray'].values, n) for i in range(nosims)]
)))

print(observedStat)
pValue = np.mean(permutations > observedStat)
print(pValue)
```

```
2.8333333333333335
0.0014
```

Test de permutare D vs. C (2)

```
deviatia standard:  1.1057764723853005
quantila 2.5%: -2.0625
quantila 97.5%: 2.0625
```