Tutorial Komputasi Statistika 2 2024

Raditya Arviandana

June 9, 2024

1 UAS 2022/2023

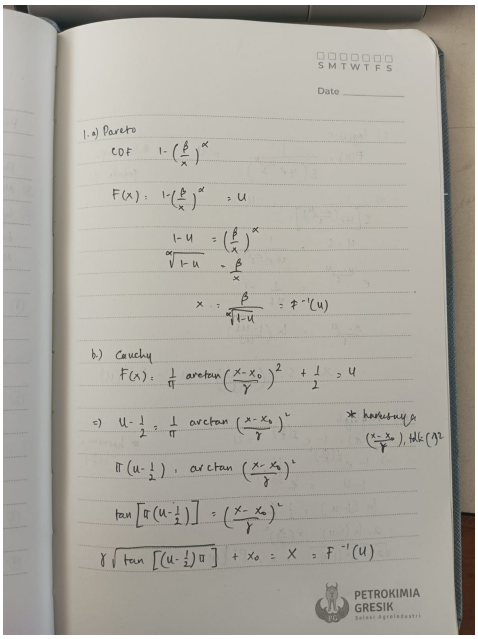
1.1 NOMOR 1

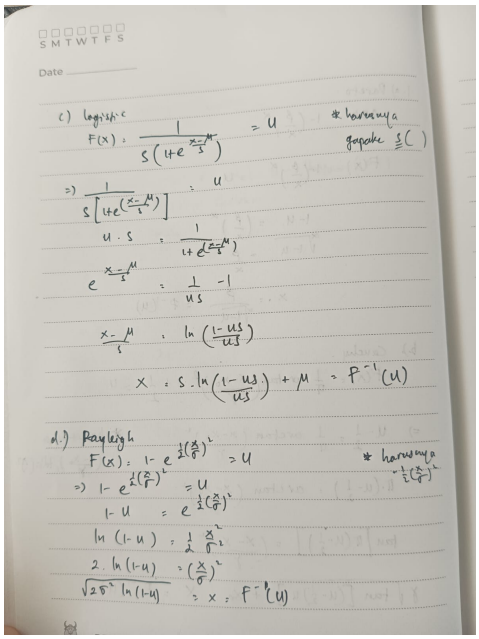
1.1.1 Buatlah skrip dengan R dan Python untuk membangkitkan bilangan random dari ke empat distribusi peluang inimenggunakan metode inverse transform

1. Diberikan fungsi pdf dan cdf dari distribusi peluang berikut. Buatlah skrip dengan R dan Python untuk membangkitkan bilangan random dari ke empat distribusi peluang ini menggunakan metode *inverse transform*.

Distribution Name	Probability Density Function	Cumulative Density Function
Pareto	$\frac{\alpha\beta^{\alpha}}{x^{\alpha+1}}$	$1-(\frac{\beta}{x})^{\alpha}$
Cauchy	$\frac{1}{\pi\gamma[1+(\frac{x-x_0}{\gamma})^2]}$	$\frac{1}{\pi}\,arctan(\frac{x-x_0}{\gamma})^2+\frac{1}{2}$
Logistic	$\frac{1}{4}sech^2(\frac{x-\mu}{2s})$	$\frac{1}{s(1+e^{\frac{x-\mu}{s}})}$
Rayleigh	$\frac{x}{\sigma^2}e^{\frac{x^2}{2\sigma^2}}$	$1-e^{\frac{x^2}{2\sigma^2}}$

Contoh sederhana, ingin dicari nilai c yang peluang kumulatifnya kurang dari 0.95 pada distribusi normal standar. Langkah yang dilakukan adalah mencari nilai kuantil dari distribusi normal standar pada titik 0.95, atau dituliskan $F^{-1}(x < c) = 0.95$ maka diperoleh nilai $c = 1.644854$





Link Komputasi:

- Komputasi Python
- Komputasi R

1.2 NOMOR 2

1.2.1 Sebutkan setidaknya 3 bahasa pemrograman yang dapat digunakan untuk melakukan kajian Analisadan pengambilan kesimpulan berbasis data

1. Diberikan fungsi pdf dan cdf dari distribusi peluang berikut. Buatlah skrip dengan R dan Python untuk membangkitkan bilangan random dari ke empat distribusi peluang ini menggunakan metode *inverse transform*.

Distribution Name	Probability Density Function	Cumulative Density Function
Pareto	$\frac{\alpha\beta^{\alpha}}{x^{\alpha+1}}$	$1-(\frac{\beta}{x})^{\alpha}$
Cauchy	$\frac{1}{\pi\gamma[1+(\frac{x-x_0}{\gamma})^2]}$	$\frac{1}{\pi}\,arctan(\frac{x-x_0}{\gamma})^2+\frac{1}{2}$
Logistic	$\frac{1}{4}sech^2(\frac{x-\mu}{2s})$	$\frac{1}{s(1+e^{\frac{x-\mu}{s}})}$
Rayleigh	$\frac{x}{\sigma^2}e^{\frac{x^2}{2\sigma^2}}$	$1-e^{\frac{x^2}{2\sigma^2}}$

a) Python

```
from sklearn.ensemble import RandomForestClassifier
import numpy as np
X, y = make_classification(n_samples=1000, n_features=4,
    n_informative=2, n_redundant=0, random_state=0, shuffle
    =False)

clf = RandomForestClassifier(max_depth=2, random_state=0)
clf.fit(X, y)
print(clf.predict([[0, 0, 0, 0]]))
```

b) R

```
install.packages("AER",dep=T)
library(AER)
data(CASchools)
linear_model<- lm(score ~ income, data = CASchools)
summary(linear_model)
```

c) SQL

```
SELECT AVG(Age)
FROM Costumer
WHERE Country = 'Spain' AND CustomerName LIKE 'G%';
```

d) JavaScript
Membuat model machine learning sekaligus *deploy* ke dalam website

Setiap bahasa pemrograman memiliki kelebihan dan kekurangan dan dapat dikatakan sebagai bahasa pemrograman terbaik berdasarkan konteksnya. R dapat menjadi bahasa pemrograman terbaik dalam konteks analisis data secara statistika karena R menyediakan berbagai *packages* dan *library* untuk mendukung analisis statistika. Python akan baik digunakan ketika masalah yang dihadapi merupakan permasalahan *machine learning* atau berbasis *computer science*. dst...

1.2.2 Berikan satu contoh penggunaan Python sebagai alat bantu komputasi untuk melakukan Analisa data ril pada bidang minat yang anda suka (jelaskan data, metode statistika, pengolahan data, dan komputasi metode dengan R). berikan pula skrip R yang setara menghasilkan output yang relatif sebanding dengan versi skrip Python juga.

Contoh analisis data dengan regresi linear sederhana dengan data numerik yang mengikuti persamaan $Y = 1 + 0.5X$. Pertama-tama, data akan di analisis dengan bahasa Python.

```
# KOMPUTASI PYTHON
import statsmodels.api as sm
import numpy as np

X = np.array([-1.0, 0.0, 1.0, 2.0, 3.0, 4.0, 5.0], dtype=float)
Y = np.array([0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5], dtype=float)

X = sm.add_constant(X)
model = sm.OLS(Y,X)
model.fit()

print(model.summary())
```

Kemudian akan digunakan bahasa R, dengan cara yang cukup serupa

```
# KOMPUTASI R
X = c(-1.0, 0.0, 1.0, 2.0, 3.0, 4.0, 5.0)
Y = c(0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5)
```

```
df = data.frame(X, Y)
names(df) = c('X','Y')

model = lm(Y ~ X, data = df)
summary(model)
```

2 UAS 2021/2022

Referensi Lain: Jawaban UAS Mas Zaki

2.1 Buatlah suatu program dalam bahasa R yang dapat digunakan untuk mencari estimator $\theta_t = (\beta_{1,t}, \tau_t)$ yang meminimumkan fungsi di atas! (Catatan: perhatikan parameter merupakan vektor)

1. Salah satu model penting dalam terapan di bidang keuangan adalah model kurva yield obligasi yang disebut sebagai model nelson siegel (NS) dan perluasannya. Secara sederhana model ini dapat dinyatakan dalam bentuk model regresi nonlinier yang dibentuk untuk menggambarkan kondisi punuk (hump) pada waktu sampai engan jatuh tempo (time to maturity) jangka pendek, menengah, dan panjang. Bentuk sederhana dari fungsi regresi nonlinear ini yang hanya mengandung hump jangka menengah dapat diberikan dalam bentuk fungsi berikut. Diberikan fungsi dengan parameter $\theta_t = (\beta_{1,t}, \tau_t)$

$$i_t(x,\theta) = \beta_{1,t}\left(\frac{1 - exp(-x_t/\tau_t)}{(-x_t/\tau_t)}\right)$$

Diberikan data waktu jatuh tempo $x = (X_1, X_2, \ldots, X_t)$ buatlah suatu program dalam bahasa R yang dapat digunakan untuk mencari estimator $\theta_t = (\beta_{1,t}, \tau_t)$ yang meminimumkan fungsi di atas! (**Catatan:** perhatikan parameter merupakan vektor)

a) Dengan menggunakan metode Newton-Raphson menggunakan R dan python
b) Menggunakan library lain yang tersedia pada R dan python.

2.1.1 Dengan menggunakan metode Newton-Raphson menggunakan R dan python

Bentuk umum rumus Newton-Raphson

$$x_{n+1} = x_n - \frac{f(x)}{f'(x)} \tag{1}$$

Langkah-langkah:

a) Gunakan metode MLE untuk estimasi parameter

b) Dari fungsi log-likelihood yang didapatkan, cari turunan pertama dan keduanya untuk tiap parameter

c) Buat fungsi Newton-Raphson dengan rumus Newton-Raphson di atas dan lakukan perulangan di dalamnya

d) Perulangan dilakukan hingga hasilnya konvergen, dengan kata lain $x_{n+1} \approx x_n$

2.1.2 Menggunakan library lain yang tersedia pada R dan python.

- Library newtonRaphson dalam R: pracma, dengan fungsi *newtonRaphson* atau *newton*
- Library dalam Python: scipy, dengan fungsi *scipy.optimize.newton*

2.2 Diketahui dalam literatur, terdapat banyak algoritma yang dapat digunakan untuk membangkitkan bilangan random berdistribusi normal (standar), diantaranya menggunakan pendekatan inverse transform, Box-Muller, Ziggurat Algorithm dan lain-lain.

a) Variabel random berdistribusi normal standar dapat dibangkitkan menggunakan algoritma Box-Muller, dengan cara membangkitkan dua bilangan random uniform U_1 dan U_2 yang independen berdistribusi U(0,1), maka transformasi

$$Z_1 = \sqrt{-2\ln(U_1)}\cos(2\pi U_2)$$
$$Z_2 = \sqrt{-2\ln(U_1)}\sin(2\pi U_2)$$

Akan menghasilkan dua bilangan random independen Z_1 dan Z_2 yang berdistribusi normal standar N(0,1). Implementasikan algoritma ini untuk membangkitkan n data random normal standar menggunakan R dan python

b) Salah satu algoritma terbaru adalah membangkitkan data distribusi normal dengan menggunakan pendekatan distribusi generalized exponential (lihat https://home.iitk.ac.id/~kundu/paper104.pdf). Implementasikan algoritma pada paper ini untuk membangkitkan n data random normal standar menggunakan R dan python.

2.2.1 Variabel random berdistribusi normal standar dapat dibangkitkan menggunakan algoritma Box-Muller, dengan cara membangkitkan dua bilangan random uniform U_1 dan U_2 yang independen berdistribusi U(0,1)

Python:

```
import numpy as np
import math

def Box_Muller(n_samples, random_state = None):
  np.random.seed(random_state)
  u1 = np.random.uniform(size = n_samples)
  u2 = np.random.uniform(size = n_samples)

  z1 = np.sqrt(-2*np.log(u1)) * np.cos(2 * math.pi * u2)
  z2 = np.sqrt(-2*np.log(u1)) * np.sin(2 * math.pi * u2)
  return z1, z2

normal1, normal2 = Box_Muller(300, random_state = 26)
```

R:

```
box_muller = function(n_samples){
  u1 = runif(n_samples)
  u2 = runif(n_samples)

  z1 = sqrt(-2*log(u1)) * cos(2*pi*u2)
  z2 = sqrt(-2*log(u1)) * sin(2*pi*u2)
  return (list(normal_1 = z1, normal_2 = z2)) # list(z1, z2) # c(
      z1, z2)
}

generate = box_muller(300)
normal1 = generate$normal_1
normal2 = generate$normal_2
```

2.2.2 Salah satu algoritma terbaru adalah membangkitkan data distribusi normal dengan menggunakan pendekatan distribusi generalized exponential (lihat https://home.iitk.ac.id/kundu/paper104.pdf). Implementasikan algoritma pada paper ini untuk membangkitkan n data random normal standar menggunakan R dan python.

Link paper

2.3 Berikan masing-masing satu contoh terapan dan komputasinya dengan R dan python dalam masalah Data Science/Machine Learning.

Python:

```
import pandas as pd
from sklearn.preprocessing import OneHotEncoder
from sklearn.linear_model import LogisticRegression

data = {
    'Gender': ['Male', 'Female', 'Female', 'Unknown'],
    'Number': [30, 20, 23, 29],
    'Buy or Not': [1, 0, 0, 1]
}
df = pd.DataFrame(data)

X = df.iloc[:, :2]
Y = df['Buy or Not']


ohe = OneHotEncoder(sparse_output = False, handle_unknown = '
   ignore')
encode = ohe.fit_transform(X[['Gender']])
X[ohe.get_feature_names_out()] = encode
X = X.drop('Gender', axis = 1)

logit = LogisticRegression()
logit.fit(X, Y)

logit.predict(np.array([[33, 0, 1, 0]]))
```

R:

```
library(fastDummies)
```

```
df = data.frame(
  Gender = c("Male", "Female", "Female", "Unknown"),
  Age = c(30, 20, 23, 29),
  Buying = c(1, 0, 0, 1)
)

df_encoded = dummy_cols(df)

logit <- glm(Buying ~ Age + Gender_Female + Gender_Male + Gender_
   Unknown, data = df_encoded, family = "binomial")
summary(logit)
```