Penjelasan Soal Seleksi Asisten Lab

Anyelyra Kantata
2206048625

21 Agustus 2024

Pada setiap penjelasan soal, akan dijelaskan mengenai tujuan program, input dan ouput, serta proses penyusunan program tersebut.

Python #2: *Probabilitas Ditargetkan*

Terdapat grup pahlawan bernama ***High Cloud Quintet***, yang terdiri dari 4 pahlawan, yaitu **Jingliu**,**Jing Yuan**, **Yingxin**, dan **Dan Heng**. Dengan kekuatan keempat karakter ini secara berturut-urut, yaitu "Destruction". "Erudition", "Destruction", dan "The Hunt".

- Menghitung probabilitas "High Cloud Quintet" ditargetkan
- Melakukan simulasi penyerangan monster terhadap keempat karakter

Saat pertempuran, Baiheng memberi mukjizat kepada anggota High Cloud Quintet (selain Jing Yuan) dengan rincian:

- Jingliu: 200%
- Yingxing: 500%
- Dan Heng: 300%

Input:

1. Nama karakter
2. Banyak mukjizat

Output:

1. Probabilitas ditargetkan
2. Simulasi penyerangan monster

Program ini dimulai dengan pembuatan dua dictionary, yaitu **karakter** dan **base_aggro**. **karakter** menyimpan nama karakter dan kekuatannya, lalu **base_aggro** menyimpan karakter dan besaran BA (Base Aggro) masing-masing karakter. Sehingga, ketika dimasukkan nama karakter, program sekaligus mendapatkan besaran Base Aggro yang sesuai dengan karakter dan kekuatannya.

```
karakter = {
    "jingliu": "destruction",
    "jing yuan": "erudition",
    "yingxing": "destruction",
    "dan heng": "the hunt"
}

base_aggro = {
    "the hunt": 3,
    "erudition": 3,
    "harmony": 4,
    "nihility": 4,
    "abundance": 4,
    "destruction": 5,
    "preservation": 6
}
```

Sebelum membahas fungsi-fungsi dalam program yang harus didefinisikan, akan dijelaskan terlebih dahulu seperti apa input yang diterima oleh program. Berikut ini langkah-langkah penyimpanan input:

1. Siapkan dictionary yang akan menyimpan nama dan besaran mukjizat, dinamakan **mukijzat_dict**. Lalu, disiapkan juga list yang akan menyimpan nama keempat karakter yang diinput, dinamakan **list_nama**.

2. Terima input **nama**, lalu masukkan dalam list **list_nama**.

3. Terima input **mukjizat**, lalu masukkan dalam dictionary **mukijzat_dict**.

4. Untuk simulasi penyerangan monster, maka akan diminta untuk memasukkan banyaknya simulasi dalam input **simulasi**.

```
for n in range(4):
    nama = input("Input the name of the character: ").lower()
    list_nama.append(nama)
    kekuatan = karakter.get(nama)
    mukjizat = int(input("Input the 'Mukjizat': "))
    mukjizat_dict[nama] = mukjizat

# Input jumlah serangan monster
simulasi = int(input("How many attacks from the monster?: "))
```

Pada soal, telah diketahui formula untuk menghitung probabilitas ditargetkan suatu karakter, yaitu:

$$A_k = BA \times (1 + M)$$

$$P = \frac{A_k}{\Sigma A_p}$$

Keterangan:
A_k = Aggro dari karakter yang dihitung
BA (Base Aggro) = Aggro standar dari kekuatan karakter
M = Mukjizat
P = Probabilitas karakter tersebut ditargetkan
A_p= Sumasi Aggro 1 party

Maka, akan dibuat fungsi yang dapat menghitung A_k dengan hanya menerima nama saja, dinamakan **def cari_aggro_karakter(nama)**. Fungsi ini akan mencari besar besar BA berdasarkan kekuatan, yang telah didapatkan dari nama karakter. Lalu, memasukkannya pada formula yang telah diberikan.

```
def cari_aggro_karakter(nama):
    kekuatan = karakter.get(nama)
    base = base_aggro[kekuatan]
    mukjizat = mukjizat_dict.get(nama)
    aggro_karakter = base * (1 + mukjizat)
    return aggro_karakter
```

Selanjutnya dibuat fungsi **def prob_targeted(nama)** yang berisi formula dari P, yaitu aggro karakter tersebut dibagi dengan penjumlahan aggro seluruh karakter dalam suatu kelompok (party) untuk mencari probabilitas suatu karakter ditargetkan.

```
def prob_targeted(nama):
    total_aggro = sum(cari_aggro_karakter(k) for k in list_nama)
    if total_aggro == 0:
    return 0
    return cari_aggro_karakter(nama) / total_aggro
```

Selanjutnya, akan dilakukan simulasi penyerangan monster yang juga dilakukan dengan sebuah fungsi, dinamakan **def simulate_attack**. Berikut ini adalah proses dari fungsi tersebut:

1. Ketika fungsi menerima nama, maka serangan akan disebar sesuai dengan probabilitas yang dimiliki.

2. Lalu, hitung sisa serangan yang tersedia dengan mengurangi jumlah simulasi serangan dengan penjumlahan dari seluruh serangan yang telah dibagikan.

3. Jika masih ada serangan, maka serangan akan dibagikan secara random, tetapi dengan weight. Weight ini diperlukan karena pembagian acak perlu dibagikan secara proporsional sesuai probabilitas yang dimiliki oleh karakter.

```
def simulate_attack(nama):
    attack_counts = {k: 0 for k in list_nama}

    for k in list_nama:
        attack_counts[k] = int(prob_targeted(k) * simulasi)

    allocated_attacks = sum(attack_counts.values())
    remaining_attacks = simulasi - allocated_attacks

    if remaining_attacks > 0:
        for _ in range(remaining_attacks):
            selected = random.choices(list_nama,
            weights=[prob_targeted(k) for k in list_nama], k=1)[0]
            attack_counts[selected] += 1

return attack_counts
```

Terakhir, probabilitas ditargetkannya suatu karakter dan simulasi serangan akan ditampilkan dengan memanggil fungsi-fungsi yang telah dibuat.

```
for nama in list_nama:
    final_probs = prob_targeted(nama)
    print('The chance that {} gets targeted
    is: {:.3%}'.format(nama, final_probs))


attack_results = simulate_attack(None)
for nama, count in attack_results.items():
    print('The number of attacks on {} is: {}'.format(nama, count))
```

Python #3: *Pembagian Kue*

Seorang ibu mempunyai 3 orang anak. Berikut ini adalah makanan kesukaan anak-anak ibu tersebut:

- Anak pertama: kue coklat
- Anak kedua: brownies
- Anak ketiga: nastar

Mereka sepakat untuk membagi memberikan kue-kue buatan ibu dengan aturan, yaitu $\frac{1}{2}$ dari masing-masing kue ke anak yang paling menyukai, lalu sisanya dibagi rata ke anak yang lain. Namun, tahun ini ibu mengangkat seorang anak kecil yang belum tahu makanan kesukaannya. Akhirnya diputuskan bahwa akan membagi N% dari bagian kue masing-masing ke adik baru.

Input:

- **X**: jumlah kue coklat
- **Y**: jumlah potong brownies
- **Z**: jumlah nastar
- **N**: persentase bagian untuk adik baru

Output: Banyak bagian yang akan didapat masing-masing anak dari setiap jenis kue.

```
X = int(input("Masukkan jumlah kue coklat: "))
Y = int(input("Masukkan jumlah potong brownies: "))
Z = int(input("Masukkan jumlah nastar: "))
N = float(input("Masukkan persentase bagian untuk adik baru: "))
```

Pertama, akan ditentukan berapa banyak kue yang akan diberi kepada adik baru. Bagian yang diambil untuk anak baru berasal dari separuh jumlah kue.

```
adik_coklat = math.floor((N / 100) * (X/2))
adik_brownies = math.floor((N / 100) * (Y/2))
adik_nastar = math.floor((N / 100) * (Z/2))
```

Hitung sisa kue setelah dibagi kepada adik baru. Dengan fungsi floor, akan dicari separuh dari masing-masing sisa kue untuk diberi ke anak yang paling menyukai.

```
sisa_coklat = X - adik_coklat
sisa_brownies = Y - adik_brownies
sisa_nastar = Z - adik_nastar

anak1_coklat = math.floor(sisa_coklat / 2)
anak2_brownies = math.floor(sisa_brownies / 2)
anak3_nastar = math.floor(sisa_nastar / 2)
```

Lalu, hitung sisa kue yang masih tersisa. Jika masih ada, maka akan dibagi rata kepada anak lainnya, termasuk anak yang menyukai juga.

```
sisa_akhir_coklat = X - (adik_coklat + anak1_coklat)
sisa_akhir_brownies = Y - (adik_brownies + anak2_brownies)
sisa_akhir_nastar = Z - (adik_nastar + anak3_nastar)

if sisa_akhir_coklat > 0:
    bagi_coklat = math.floor(sisa_akhir_coklat/3)
    anak1_coklat = anak1_coklat + bagi_coklat
    anak2_coklat = bagi_coklat
    anak3_coklat = bagi_coklat

if sisa_akhir_brownies > 0:
    bagi_brownies = math.floor(sisa_akhir_brownies/3)
    anak2_brownies = anak2_brownies + bagi_brownies
    anak1_brownies = bagi_brownies
```

```
        anak3_brownies = bagi_brownies

    if sisa_akhir_nastar > 0:
        bagi_nastar = math.floor(sisa_akhir_nastar/3)
        anak3_nastar = anak3_nastar + bagi_nastar
        anak1_nastar = bagi_nastar
        anak2_nastar = bagi_nastar
```

Terakhir, akan ditampilkan jumlah dari masing-masing kue yang diterima oleh masing-masing anak.

Wolfram #2: *Segitiga*

Input:

- Satu list $\{x_1, x_2, x_3, x_4, x_5, x_6\}$ yang berisi 6 bilangan bulat non negative antara 0 sampai 9
- Satu bialngan bulat positif n

Output:

- Hasil tiap iterasi
- Tiga titik yang didapatkan
- Apakah ketiga titik bisa membentuk segitiga?
- Jika bisa membentuk segitia, berapa besar tiap sudut?
- Dan gambar segitiga dalam bidang Kartesius?

Tujuan:
Membuat fungsi bernama **NGAB** yang menerima list dan bilangan bulat positif. Dimana fungsi ini akan melalukan sebuah iterasi (dengan aturan) terhadap list, membuatnya menjadi tiga titik, dan membuat segitiga (apabila memungkinkan) dari ketiga titik tersebut.

Pada soal telah diberikan arahan pembuatan fungsi, maka seluruh langkah-langkah pembuatan fungsi akan sesuai dengan arahan pada soal.
Input list $\{x_1, x_2, x_3, x_4, x_5, x_6\}$ yang diberikan akan dilakukan iterasi sebanyak n kali (yang setiap hasil iterasi akan ditampilkan). Misal list $\{y_1, y_2, y_3, y_4, y_5, y_6\}$ adalah list hasil 1 kali iterasi, berikut ini adalah aturan iterasinya:

$$
\begin{aligned}
y_1 &= (x_1 + x_2) \mod 10 \\
y_2 &= (x_2 + x_3) \mod 10 \\
y_3 &= (x_3 + x_4) \mod 10 \\
y_4 &= (x_4 + x_5) \mod 10 \\
y_5 &= (x_5 + x_6) \mod 10 \\
y_6 &= (x_6 + x_1) \mod 10
\end{aligned}
$$

```
NGAB[list_, n_] := Module[{currentList = list, nextList, i, points},

  Print["Hasil Iterasi:"]
    Print[currentList]
    For[i = 1, i <= n, i++,
     nextList = Mod[{
        currentList[[1]] + currentList[[2]],
        currentList[[2]] + currentList[[3]],
        currentList[[3]] + currentList[[4]],
        currentList[[4]] + currentList[[5]],
        currentList[[5]] + currentList[[6]],
        currentList[[6]] + currentList[[1]]
        }, 10];
     Print[nextList];
     currentList = nextList;
     ];
```

Setelah iterasi selesai, maka akan dibuat tiga titik yang diambil dari setiap 2 angka berurutan dari list angka terakhir.
Misal list terakhir adalah: $\{1, 2, 3, 4, 5, 6\}$
Maka, titik yang terbentuk adalah: {1,2}, {3,4}, {5,6}

```
titik1 = {currentList[[1]], currentList[[2]]};
titik2 = {currentList[[3]], currentList[[4]]};
titik3 = {currentList[[5]], currentList[[6]]};
titikList = {titik1, titik2, titik3};
```

Karena tujuan kita adalah untuk membuat sebuah segitiga, maka kita perlu memeriksa apakah titik-titik yang telah didapatkan dapat membentuk segitiga atau tidak. Untuk memeriksa, akan digunakan ***Triangle Inequality*** untuk memeriksa apakah segitiga dengan panjang-panjang sisi a, b, c terdapat atau tidak.

Jika $a + b > c$, $a + c > b$, dan $b + c > a$ terpenuhi. Maka, segitiga dengan panjang sisi tersebut ada. Namun, karena yang akan diperiksa adalah dalam bentuk titik, maka akan dihitung dulu jarak antara dua titik.

Dengan formula untuk menghitung jarak antara dua titik, yaitu:
Misal titik $A(x_1, y_1)$ dan $B(x_2, y_2)$.
Maka, untuk menghitung jarak antara titik A dan B adalah: $AB = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$.

```
AB = Sqrt[(titik2[[1]] - titik1[[1]])^2 + (titik2[[2]] -
       titik1[[2]])^2];
BC = Sqrt[(titik3[[1]] - titik2[[1]])^2 + (titik3[[2]] -
       titik2[[2]])^2];
CA = Sqrt[(titik3[[1]] - titik1[[1]])^2 + (titik3[[2]] -
       titik1[[2]])^2];
```

```
isSegitiga = AB + BC > CA && AB + CA > BC && BC + CA > AB;
```

Jika **isSegitiga** tidak terpenuhi, maka fungsi akan berhenti dan memberikan pesan bahwa ketiga titik tersebut tidak dapat membentuk segitiga.

Namun, jika **isSegitiga** terpenuhi, maka fungsi akan melanjutkan untuk mencari besar ketiga sudut dari segitiga yang terbentuk dengan menggunakan **Aturan Cosinus**.

```
sudutA = N[ArcCos[(BC^2 + CA^2 - AB^2)/(2*BC*CA)]*180/Pi];
sudutB = N[ArcCos[(AB^2 + CA^2 - BC^2)/(2*AB*CA)]*180/Pi];
sudutC = N[ArcCos[(AB^2 + BC^2 - CA^2)/(2*AB*BC)]*180/Pi];
sudutList = Sort[{sudutA, sudutB, sudutC}];
```

Lalu, sudut-sudut tersebut akan ditampilkan secara terurut.

```
sudutList = Sort[{sudutA, sudutB, sudutC}];
```

Terakhir, akan dibuat akan ditampilkan gambar segitiga dalam bidang kartesius apabila **isSegitiga** terpenuhi.

```
If[isSegitiga, ListLinePlot[{titik1, titik2, titik3, titik1}]]
```

EDA & R #1: *Perencanaan Pembuatan Film*

John Doe adalah seorang pembuat film. Ia menghadapi tantangan dalam menentukan aspek terbaik untuk menjadi pertimbangan dalam pembuatan filmnya, seperti anggaran, genre, dan perusahaan produksi. Oleh karena itu akan dilakukan *Exploratory Data Analysis* dengan **RStudio** untuk memberi rekomendasi kepada John Doe.

Setelah dilakukan pembersihan data, tidak ada data yang hilang atau terduplikasi. Selanjutnya, akan dilakukan penghitungan statistik deskriptif, untuk kolom numerik yang sesuai, yaitu **"budget"**, **"runtime"**, **"box_office"**, dan **"vote_average"**.

Kolom Numerik	Min.	1st Qu.	Median	Mean	3rd Qu.	Max
"budget"	5305554	55956135	108357038	103815079	153487338	199935700
"runtime"	80.0	103.8	127.0	128.4	153.0	179.0
"box_office"	4770194	109426694	222322439	273842431	404061613	976720229
"vote_average"	1.000	3.000	5.300	5.396	7.800	10.000

Table 1: Statistik Deskriptif

Terlihat bahwa rata-rata anggaran yang perlu dikeluarakan untuk pembuatan sebuah film adalah sekitar 100 juta. Lalu, untuk durasi, rata-rata durasi film adalah sekitar 120 menit. Ini bisa menjadi rekomendasi bagi John untuk mempersiapkan anggaran dan durasi film yang harus dibuat supaya bisa bersaing dengan film di pasar.

Selanjutnya, dilakukan beberapa visualisasi untuk analisis hubungan antara fitur yang mungkin berpengaruh.

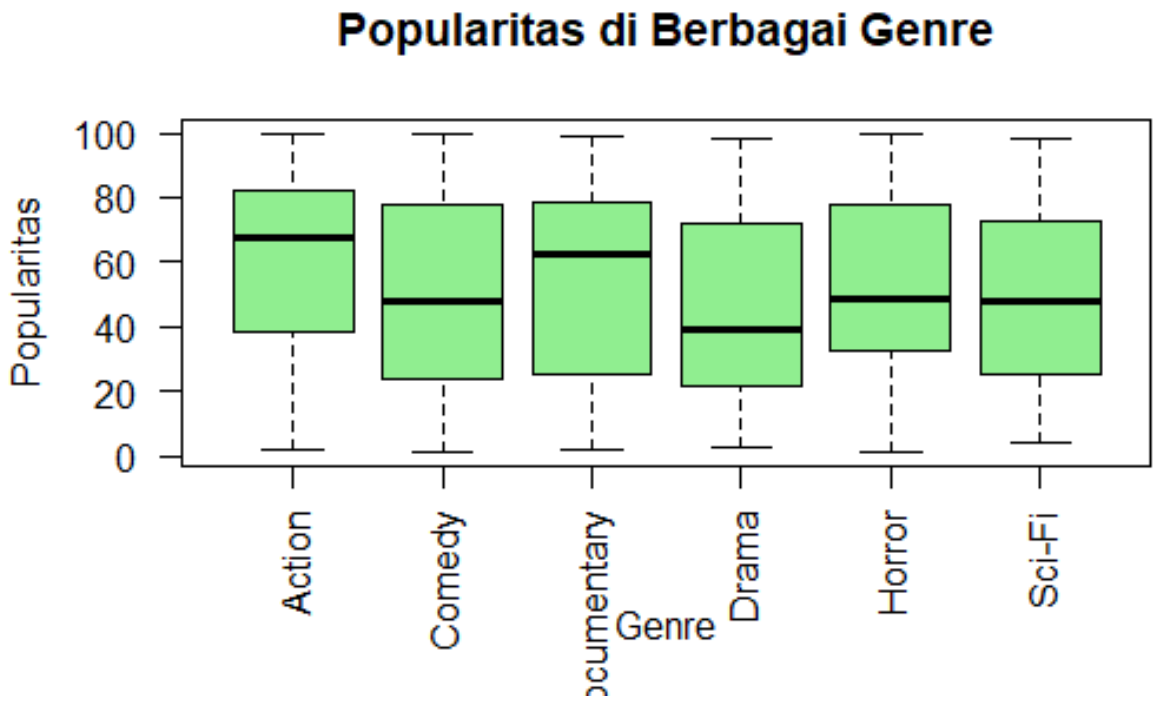

Figure 1: Popularitas di Berbagai Genre

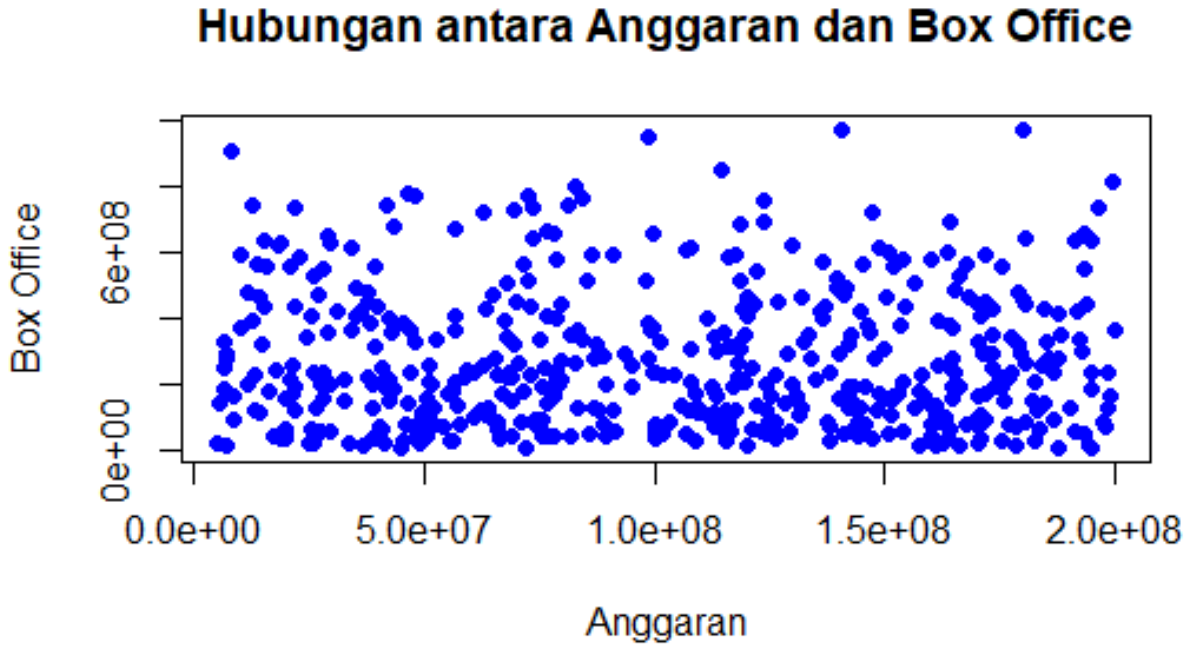

Figure 2: Hubungan antara Anggaran dan Box Office

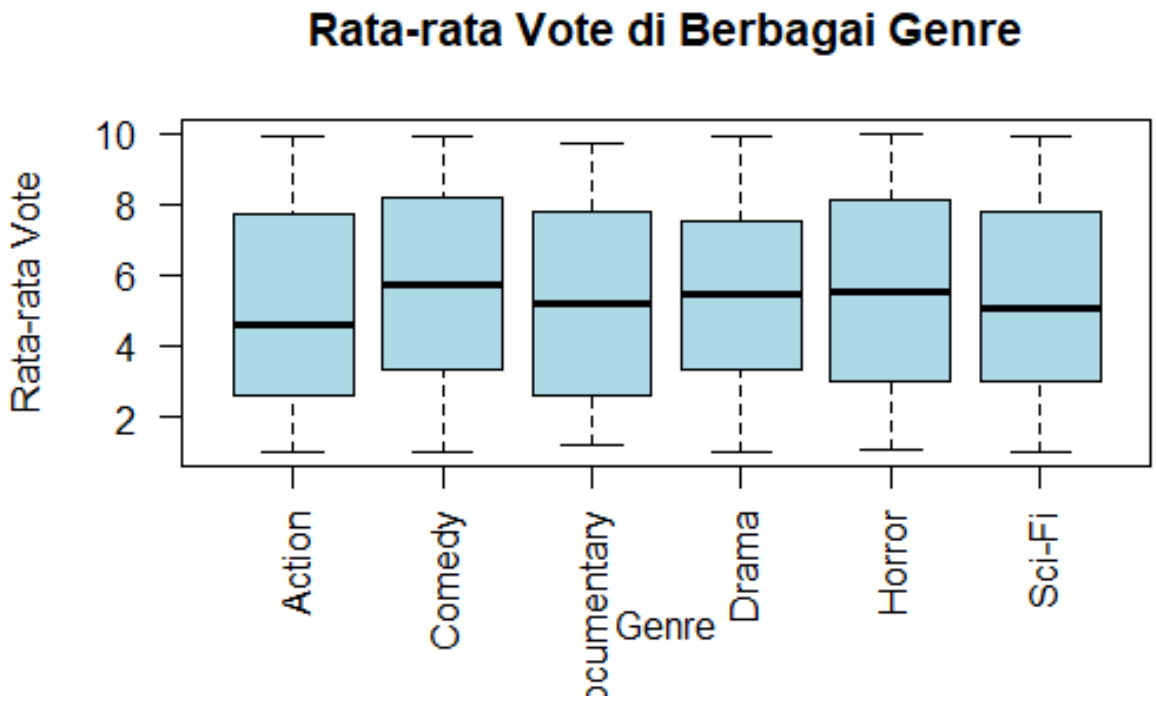

Figure 3: Rata-rata Vote di Berbagai Genre

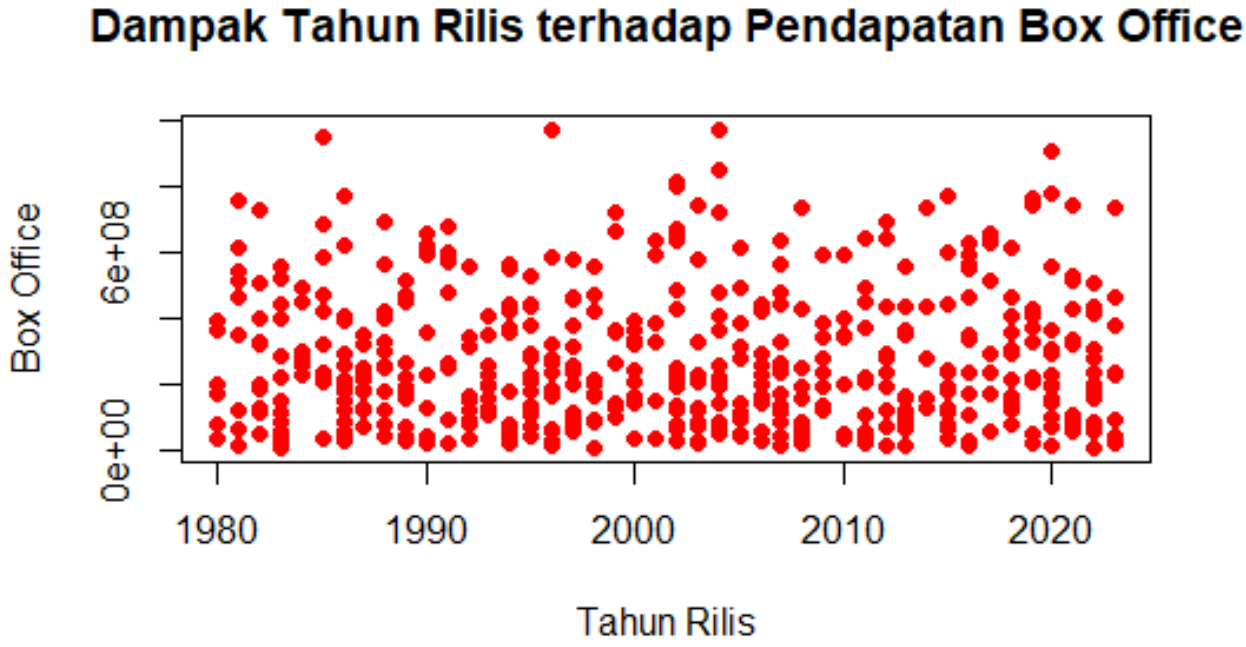

Figure 4: Dampak Tahun Rilis terhadap Pendapatan Box Office

Berdasarkan scatter plot, hubungan Tahun Rilis dan Anggaran dengan Box Office ternyata tidak begitu berpengaruh. Karena terlihat bahwa film dengan anggaran yang kecil juga mungkin untuk mendapat pendapatan Box Office yang besar.

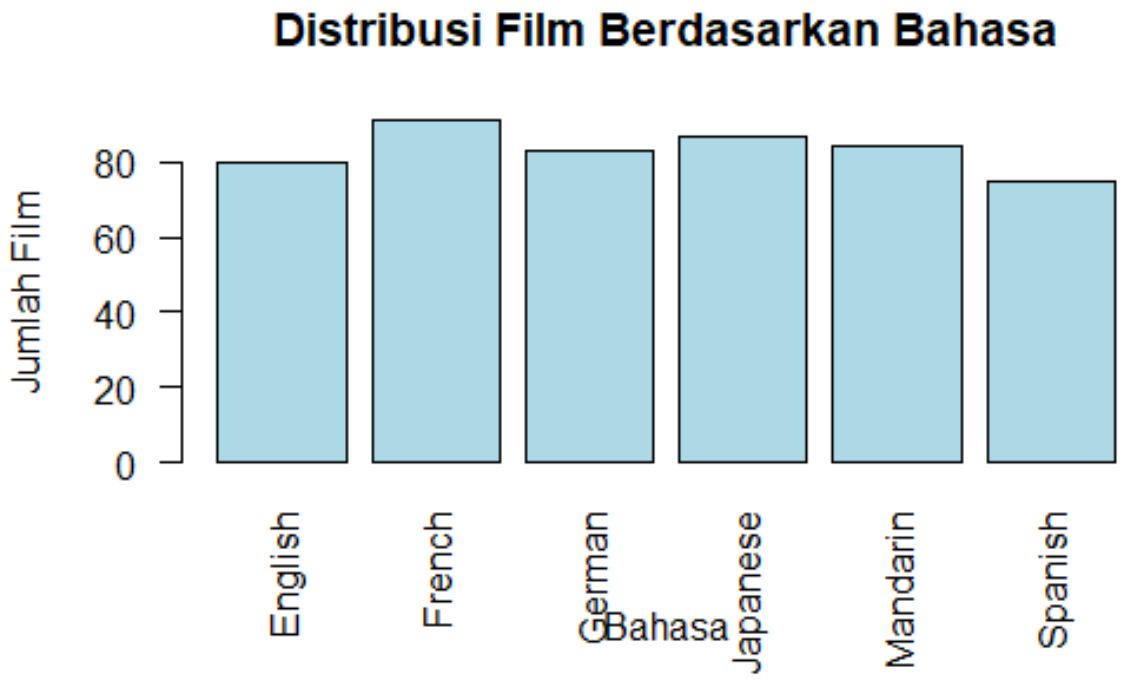

Figure 5: Distribusi Film Berdasarkan Bahasa

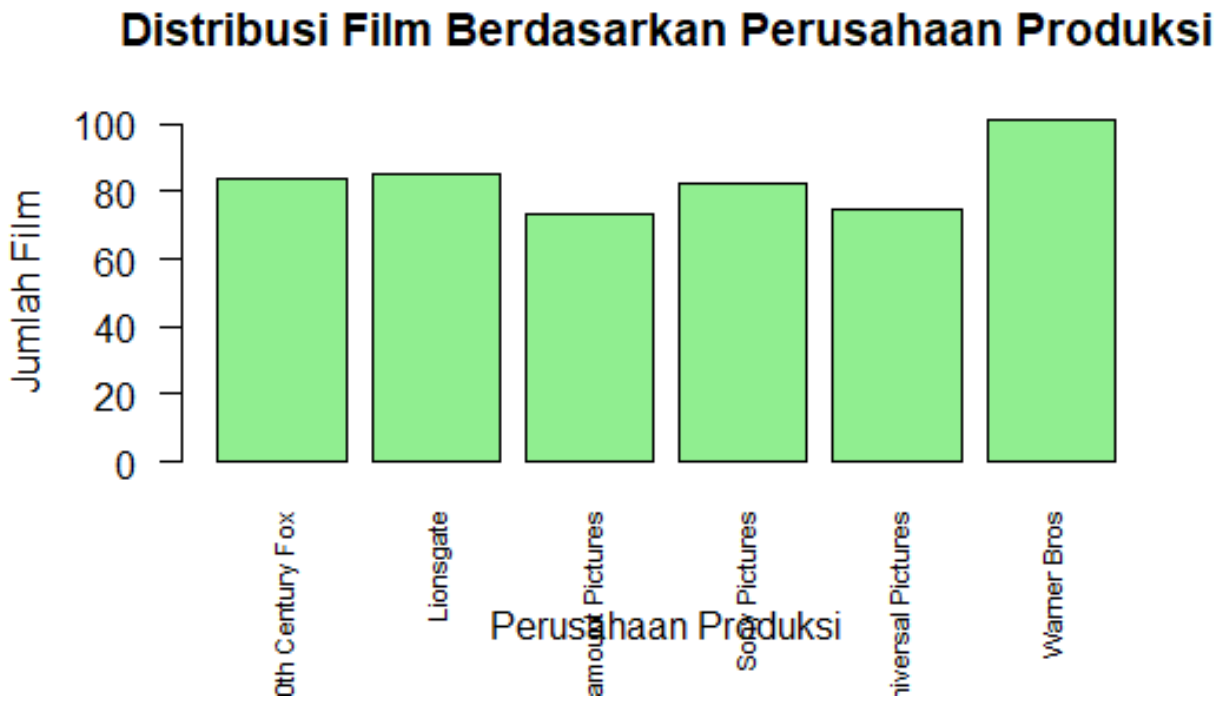

Figure 6: Distribusi Film Berdasarkan Perusahaan Produksi

Didapatkan bahwa film dengan bahasa Prancis paling banyak diproduksi, meskipun selisih dengan jumlah film yang berbahasa lain tidak begitu jauh. Berdasarkan perusahaan produksi, Warner Bros merupakan perusahaan yang paling banyak memproduksi film. Hal ini dapat menjadi pertimbangan John Doe untuk memilih perusahaan produksi mana yang akan diajak untuk bekerja sama.

Lalu, diberikan juga 3 genre teratas yang diurutkan berdasarkan rata-rata **"box_office"**, rata-rata **"popularity"**, dan rata-rata **"vote_average"** setiap genre.

No	Genre	Rata-rata **"box_office"**
1	Documentary	299882057
2	Comedy	286376699
3	Horror	270516402

Table 2: 3 Genre Teratas Berdasarkan Box Office

No	Genre	Rata-rata **"popularity"**
1	Action	59.11796
2	Documentary	54.22829
3	Horror	52.12329

Table 3: 3 Genre Teratas Berdasarkan Popularitas

No	Genre	Rata-rata **"vote_average"**
1	Horror	5.626829
2	Comedy	5.548750
3	Drama	5.475000

Table 4: 3 Genre Teratas Berdasarkan Rata-rata Vote

Terlihat bahwa berdasarkan ketiga fitur tersebut, genre Horror selalu ada dalam 3 genre teratas. Hasil ini juga bisa menjadi pertimbangan bagi John Doe untuk memilih genre film yang akan dibuat.

Kesimpulan & Saran:

1. **Memilih perusahaan produksi**: John disarankan untuk memilih perusahaan produksi yang produktif memproduksi film. Perusahaan yang paling banyak memproduksi film adalah Warner Bros.

2. **Memilih Bahasa**: Bahasa yang paling banyak dipakai dalam produksi film adalah Pransis. Dengan asumsi bahwa para pembuat film banyak menggunakan bahasa itu karena pasar lebih tertarik dengan film berbahasa Prancis, maka John bisa menggunakan bahasa yang sama juga. Bahasa Jepang menggunakan bahasa kedua terbanyak yang dipakai. Sehingga, John bisa mempertimbangkan untuk memilih bahasa Prancis atau Jepang.

3. **Memilih Genre**: Berdasarkan rata-rata Box Office, Popularitas, dan Vote, genre Horror selalu berada pada urutan 3 teratas. Film Dokumenter juga selau menempati urutan 3 teratas berdasarkan rata-rata Box Office dan Popularitas. Lalu, film Komedi juga selalu menempati urutan 3 teratas berdasarkan rata-rata Box Office dan Vote. Oleh karena itu, genre Horror, Dokumenter, dan Komedi dapat menjadi pertimbangan untuk John.

4. **Persiapan Anggaran**: Dari seluruh film yang ada pada dataset, film termurah pernah dibuat dengan anggaran sekitar 5 juta dan film termahal pernah dibuat dengan anggaran sekitar 199 juta. Jika John ingin mempersiapkan anggaran berdasarkan rata-rata anggaran pembuatan film, maka John perlu menyiapkan anggaran sekitara 100 juta.